Inhaltsverzeichnis

Einleitung

In den letzten Jahren stieg die Anzahl der Anwendungsgebiete für integrierte Schaltungen ständig.

Wachsende Integrationsdichte und Automatisierung des Entwurfs machten es möglich, für zahlreiche Aufgaben den Einsatz universeller Mikroprozessoren durch anwenderspezifische Spezialschaltkreise (ASIC's = Application Specific Integrated Circuits) zu ersetzen.

Da anwenderspezifische Schaltkreise meist nur in geringen Stückzahlen produziert werden, kommt es immer mehr zu einem Übergewicht der Entwurfskosten gegenüber den Fertigungskosten eines Chips.

Kurze Entwurfszeiten und damit niedrige Entwurfskosten können aber nur dann erreicht werden, wenn möglichst große Teile des Entwurfsprozesses automatisiert werden, d.h. rechnergestützt ablaufen.

Ein wichtiges Teilproblem beim Entwurf integrierter Schaltungen ist der Entwurf kombinatorischer Teilschaltungen, d.h. der Entwurf von Realisierungen zu Schaltfunktionen $f : \{0,1\}^n \rightarrow \{0,1\}^m$. Zu gegebenen Schaltfunktionen sollen Schaltkreise gefunden werden, die diese Funktionen realisieren und möglichst geringe Kosten haben. Als Kostenmaße kommen hierbei z.B. die benötigte Schaltkreisfläche bzw. die Laufzeit der Schaltung in Frage.

Abhängig von dem zu lösenden praktischen Problem sind hierbei die verschiedensten Schaltfunktionen zu realisieren. *Eine* Eigenschaft haben die in der Praxis zu realisierenden Schaltfunktionen jedoch zumeist gemeinsam: Sie unterscheiden sich in ihrem Aufbau wesentlich von Funktionen, deren Funktionstabellen zufällig ausgewürfelt wurden. Die in der Praxis auftretenden Funktionen sind in der Regel *nicht* durch (für große Eingangsvariablenzahlen *riesige*) Funktionstabellen gegeben, sondern entspringen einem praktischen Problem und genügen daher relativ einfachen und kurzen Bildungsgesetzen. Daher weisen solche Schaltfunktionen meistens gewisse Regelmäßigkeiten bzw. Struktureigenschaften auf. Bei der Logiksynthese (d.h. beim Entwurf von kombinatorischen Schaltungen zu diesen Funktionen) müssen solche Struktureigenschaften ausgenutzt werden, um zu guten Realisierungen zu kommen.

Mehrere Gründe sprechen für eine *automatische* Durchführung der Logiksynthese:

- Der Entwurf guter Realisierungen ist häufig mühsam und zeitaufwendig.

- Es sollen auch Anwender, die keine Experten auf dem Gebiet der Schaltkreissynthese sind, in die Lage versetzt werden, Entwürfe durch eine einfache Spezifikation des gewünschten Ein–/Ausgabeverhaltens durchzuführen.

- Häufig enthalten die zu realisierenden Schaltfunktionen zwar Regelmäßigkeiten, aber diese Regelmäßigkeiten sind nicht auf den ersten Blick zu erkennen. Auf diese Weise wird es für den Menschen schwierig, eine Logiksynthese unter Ausnutzung vorhandener Struktureigenschaften durchzuführen.

Oftmals beschränkt man sich bei der Realisierung von Schaltfunktionen auf zweistufige Lösungen. Einerseits lassen sich solche zweistufigen Lösungen durch programmierbare logische Felder (PLA's) leicht umsetzen, andererseits gibt es relativ gute heuristische Verfahren, die in der Lage sind, kostengünstige zweistufige Realisierungen zu finden. Allerdings kommen sehr einfache Schaltfunktionen vor, bei denen auch die beste zweistufige Realisierung *sehr groß* wird (vgl. Kapitel 2). Die Einschränkung des Suchraumes auf zweistufige Realisierungen macht sich oft negativ bemerkbar. Aus diesem Grund befaßt sich die vorliegende Arbeit mit *mehrstufigen* Realisierungen.

Wichtige Struktureigenschaften von Schaltfunktionen sind *Symmetrien* (z.B. Invarianz gegenüber der Vertauschung von Eingangsvariablen) und *nichttriviale Zerlegbarkeit*. Es werden hier Möglichkeiten zur Ausnutzung von Symmetrieeigenschaften bei der Logiksynthese angegeben. Weiterhin wird ein Logiksyntheseverfahren vorgestellt, das auf der Ausnutzung nichttrivialer Zerlegbarkeit beruht.

Die Arbeit ist folgendermaßen gegliedert:
In Kapitel 1 werden grundlegende Definitionen angegeben, die in der weiteren Arbeit benötigt werden. Außerdem wird eine kurze Einführung in das Gebiet der Booleschen Algebren gegeben. Hiebei werden im wesentlichen wichtige Ergebnisse aus [Hot74] wiederholt. Leser, die mit dem Thema vertraut sind, können dieses Kapitel überspringen bzw. lediglich zum Nachschlagen bei Unklarheiten bzgl. Definitionen benutzen.

Kapitel 2 dient zum einen dazu, die *mehrstufige* Logiksynthese zu motivieren, zum anderen dient es zur grundsätzlichen Untersuchung der Komplexität von Schaltfunktionen. Dadurch wird verdeutlicht, was Logiksynthesealgorithmen evtl. leisten können bzw. was man *nicht* von ihnen erwarten kann.

Kapitel 3 befaßt sich mit der Ausnutzung von Symmetrieeigenschaften bei der Logiksynthese (und ihrer Erkennung).

Der Schwerpunkt der Arbeit liegt auf Kapitel 4. Kapitel 4 befaßt sich mit nichttrivialen Zerlegungen von Schaltfunktionen. Es enthält grundsätzliche Überlegungen zur Suche nach geeigneten Zerlegungen, die auch auf partielle (d.h. nicht vollständig spezifizierte) Schaltfunktionen erweitert werden. Weiterhin werden

Zusammenhänge zwischen Symmetrieeigenschaften und nichttrivialer Zerlegbarkeit von Schaltfunktionen aufgezeigt. Ein wichtiger Punkt liegt in der Behandlung von Schaltfunktionen mit mehreren Ausgängen. Es werden Möglichkeiten entwickelt, wie man Schaltkreise finden kann, die die einzelnen Ausgangsfunktionen nicht *getrennt* realisieren, sondern möglichst große Schaltungsteile bei der Realisierung *mehrerer* Ausgangsfunktionen mit Vorteil verwenden. In einem letzten Abschnitt werden einige Beispielentwürfe eines auf diesen Grundlagen implementierten Verfahrens angegeben. Dieser Abschnitt zeigt, daß auch bei Problemstellungen, die schon intensiv unter Einsatz *menschlicher* Intelligenz bearbeitet wurden, mit *automatischer* Logiksynthese konkurrenzfähige Ergebnisse erzielt werden können. Das Beispiel eines Addierers zeigt, daß es sogar denkbar ist, Realisierungen, die von automatischen Logiksynthesewerkzeugen erzeugt wurden, als Anregung für parametrisierte Entwürfe (d.h. Entwürfe mit variabler Bitbreite) zu nutzen.

Bei der vorliegenden Arbeit handelt es sich um die Diplomarbeit des ersten Autors, die am Lehrstuhl von Prof. Dr. G. Hotz, Fachbereich Informatik der Universität des Saarlandes, unter Betreuung des zweiten Autors angefertigt wurde.

Kapitel 1

Grundlagen

1.1 Grundlegende Definitionen

Aufgabe der Logiksynthese ist es, zu einer vorgegebenen booleschen Funktion einen Schaltkreis zu finden, der diese realisiert.

Im folgenden sollen die grundlegenden Begriffe, die in diesem Zusammenhang eine Rolle spielen, formal definiert werden.

1.1.1 Boolesche Funktionen

Definition 1.1 (Totale boolesche Funktionen)

$$B_{n,m} = \{f : \{0,1\}^n \to \{0,1\}^m\}$$

sei die Menge aller **(totalen) booleschen Funktionen** *bzw. Schaltfunktionen von* $\{0,1\}^n$ *nach* $\{0,1\}^m$.

Bezeichnung 1 $B_n := B_{n,1}$

Definition 1.2 (Partielle boolesche Funktionen)

$$BP_{n,m} = \{f : D \to \{0,1\}^m \mid D \subseteq \{0,1\}^n\}$$

sei die Menge aller **partiellen booleschen Funktionen** *bzw. Schaltfunktionen von* $\{0,1\}^n$ *nach* $\{0,1\}^m$.
D heißt Definitionsbereich von $f : D \to \{0,1\}^m$.

Bezeichnung 2 *Für* $D \subseteq \{0,1\}^n$ *sei*

$$S(D) = \{f : D \to \{0,1\}\}$$

die Menge aller **partiellen booleschen Funktionen** *von* D *nach* $\{0,1\}$.

Definition 1.3 (ON–Menge, OFF–Menge, DC–Menge)
Sei $f : D \to \{0,1\}$, $D \subseteq \{0,1\}^n$.
Dann heißt

- $\{0,1\}^n \setminus D$ *die* **don't care–Menge** *(kurz DC–Menge) von* f*,*
- $ON(f) = \{x \in D \mid f(x) = 1\}$ *die* **ON–Menge** *von* f*,*
- $OFF(f) = \{x \in D \mid f(x) = 0\}$ *die* **OFF–Menge** *von* f*.*

Um aus vorgegebenen booleschen Funktionen leicht neue Funktionen definieren zu können, benutzt man die Funktionen aus B_1 und B_2 als Operationen auf den Funktionen aus $S(D)$. Sind $g, h \in S(D)$ und ist f eine Operation aus B_2 dann schreibt man im allgemeinen statt $f(g,h)$ in Infixschreibweise $g\ f\ h$. Es gilt also:

$$\forall x \in D \ : \ (g\ f\ h)(x) = f(g(x), h(x))$$

Ebenso für $f \in B_1$:

$$\forall x \in D \ : \ (f\ g)(x) = f(g(x))$$

Allgemein sind für Funktionen aus B_1 bzw. B_2 folgende Bezeichnungen üblich:

Bezeichnung 3

- $not(x) = 1$ *genau dann, wenn* $x = 0$*. not heißt Negation von* x*. Schreibweise:* $\overline{x}$ *oder* $\neg x$*.*
- $and(x,y) = 1$ *genau dann, wenn* $x = y = 1$*. and heißt Konjunktion von* x *und* y*. Schreibweise:* $x \wedge y$*,* $x \cdot y$ *oder* xy*.*
- $nand(x,y) = 0$ *genau dann, wenn* $x = y = 1$*.*
- $or(x,y) = 0$ *genau dann, wenn* $x = y = 0$*. or heißt Disjunktion von* x *und* y*. Schreibweise:* $x \vee y$ *oder* $x + y$*.*
- $nor(x,y) = 1$ *genau dann, wenn* $x = y = 0$*.*
- $exor(x,y) = 1$ *genau dann, wenn* $x \neq y$*. exor heißt exclusive-or-Funktion von* x *und* y*. Schreibweise:* $x \oplus y$*.*
- $equiv(x,y) = 1$ *genau dann, wenn* $x = y$*. Schreibweise:* $x \equiv y$*.*
- $0(x,y) = 0\ \forall x, y \in \{0,1\}$*. 0 ist die konstante Nullfunktion.*
- $1(x,y) = 1\ \forall x, y \in \{0,1\}$*. 1 ist die konstante Einsfunktion.*

Bemerkung 1 *Sei* $\pi_i^n \in B_n$ *die i. Projektion, d.h.* $\pi_i^n(x_1, \ldots, x_n) = x_i$*. Falls aus dem Zusammenhang hervorgeht, was gemeint ist, wird häufig* π_i^n *einfach als* x_i *bezeichnet.*

1.1.2 Ω–Schaltkreise

Zellenbibliothek

Im allgemeinen hat man zur Realisierung boolescher Funktionen eine gewisse Auswahl an (einfachen) booleschen Funktionen zur Verfügung, mit deren Hilfe komplexere Funktionen ausgedrückt werden können (vgl. Standardzellenentwurf). Die schon verfügbaren Funktionen sind in einer Zellenbibliothek zusammengefaßt.

Definition 1.4 (Zellenbibliothek) *Eine endliche Teilmenge* $\Omega \subseteq \bigcup_{n \in \mathbf{N}} B_n$ *heißt Zellenbibliothek.*

Häufig wird als Zellenbibliothek $\Omega = B_2$ oder eine Teilmenge von B_2 gewählt.

Im allgemeinen bezeichnet man Realisierungen für Funktionen aus einer Zellenbibliothek als Gatter (z. B. „*and*–Gatter“, „*exor*–Gatter“ etc.).

Definition von Ω–Schaltkreisen

Die beiden nächsten Definitionen geben an, wie man mit Hilfe von Funktionen aus einer solchen Zellenbibliothek Ω komplexere Funktionen darstellt. Die Darstellung erfolgt durch einen sogenannten Ω–Schaltkreis, der direkt als Netzliste einer Schaltung interpretiert werden kann.

Definition 1.5 (Schaltkreis)
Sei Ω *eine Zellenbibliothek. Sei* $T = \Omega \cup \{EPAD, APAD\}$.
Ein Ω–**Schaltkreis** S *mit* n *Eingängen und* m *Ausgängen ist ein 4–Tupel*

$$(G = (V, E), typ, pe, pa),$$

wobei gilt:

- G *ist ein gerichteter, azyklischer, knotenorientierter*[1] *Graph.* V *ist die Menge der Knoten bzw. Zellen,* E *die Menge der Kanten bzw. Verbindungssleitungen.*
  Jeder Kante ist eine Richtung zugeordnet. Quelle und Ziel einer Kante $e = (v, w)$ *sind gegeben durch die Abbildungen* $Q : E \rightarrow V$ *und* $Z : E \rightarrow V$, *wobei* $Q(e) = v$, $Z(e) = w$.
  Der Eingangsgrad eines Knotens v *ist definiert durch die Abbildung* $indeg : V \rightarrow \mathbf{N}$, $indeg(v) = |\{e \mid Z(e) = v\}|$.
  Entsprechend ist der Ausgangsgrad eines Knotens v *definiert durch die Abbildung* $outdeg : V \rightarrow \mathbf{N}$, $outdeg(v) = |\{e \mid Q(e) = v\}|$.
  Die Knotenorientierungen von G *sind durch die partiellen injektiven Abbildungen* $I, O : V \times \mathbf{N} \rightsquigarrow E$ *definiert. Falls* $1 \leq i \leq indeg(v)$, *dann ist*

[1] Ein Graph G heißt *knotenorientiert*, wenn es für jeden Knoten v des Graphen sowohl eine Numerierung der einlaufenden Kanten als auch eine Numerierung der auslaufenden Kanten gibt.

$I(v,i)$ definiert, $I(v,i) = e$ mit $Z(e) = v$ und e heißt i. Input des Knotens v; falls $1 \leq i \leq outdeg(v)$, dann ist $O(v,i)$ definiert, $O(v,i) = e$ mit $Q(e) = v$ und e heißt i. Output von v.

- *$typ : V \to T$ ist eine Abbildung, die jedem Knoten einen „Typ" (Funktion der Zellenbibliothek oder EPAD bzw. APAD) zuordnet.*
  Es muß gelten: $|typ^{-1}(EPAD)| = n$, $|typ^{-1}(APAD)| = m$.
  Falls $typ(v) = t$, dann heißt t Typ von v, v ein t–Knoten.
  Falls $typ(v) \in \Omega$, dann gilt $typ(v) \in B_{indeg(v)}$.
  Falls $typ(v) = EPAD$, dann gilt $indeg(v) = 0$,
  falls $typ(v) = APAD$, dann gilt $indeg(v) = 1$, $outdeg(v) = 0$.

- *pe, pa sind „Numerierungen" der EPAD– bzw. APAD–Knoten.*
  $pe : \{1,\ldots,n\} \to \{v \in V \mid typ(v) = EPAD\}$, $pa : \{1,\ldots,m\} \to \{v \in V \mid typ(v) = APAD\}$ sind bijektive Abbildungen.
  $pe(i)$ ($pa(i)$) heißt der i. primäre Eingang (Ausgang).

Die nun folgende Definition stellt den Zusammenhang her zwischen einem Ω–Schaltkreis und den booleschen Funktionen, die durch diesen Schaltkreis realisiert werden.

Definition 1.6 (Durch Ω–Schaltkreis definierte Funktion)
Sei $S = (G, typ, pe, pa)$ ein Ω–Schaltkreis mit n Eingängen und m Ausgängen.
Sei $e \in E$ eine Verbindungsleitung mit $e = O(v,j)$.

- *Falls $typ(v) = EPAD$ und $v = pe(i)$, dann ist die durch Leitung e berechnete Funktion definiert durch*

$$f_e : \{0,1\}^n \to \{0,1\}, \quad f_e(x_1,\ldots,x_n) = x_i.$$

- *Falls $typ(v) = g \in \Omega$, $e_k = I(v,k)$ für $k = 1,\ldots,indeg(v)$, dann ist die durch die Leitung e berechnete Funktion definiert durch*

$$f_e : \{0,1\}^n \to \{0,1\}, \quad f_e(\mathbf{x}) = g(f_{e_1}(\mathbf{x}),\ldots,f_{e_{indeg(v)}}(\mathbf{x}))$$

Seien nun für $1 \leq i \leq m$ $y_i = I(pa(i))$, f_{y_i} jeweils die durch y_i berechneten Funktionen.
Dann ist die durch S realisierte Funktion $f_S : \{0,1\}^n \to \{0,1\}^m$ definiert durch

$$f_S(\mathbf{x}) = (f_{y_1}(\mathbf{x}),\ldots,f_{y_m}(\mathbf{x})).$$

Ein Schaltkreis S soll aber nicht nur als Realisierung einer einzigen totalen Funktion f_S gelten, sondern als Realisierung *aller* Funktionen, die man aus f_S durch Einschränkung des Definitionsbereichs erhält:

Definition 1.7 (S Realisierung von g) *Sei $f_S : \{0,1\}^n \to \{0,1\}^m$ die durch den Ω–Schaltkreis S realisierte Funktion.*

Dann heißt S Schaltkreis bzw. Realisierung für jede *Funktion $g : D \to \{0,1\}^m$, $D \subseteq \{0,1\}^n$, mit $f|_D = g$.*

Beispiel

Die bisher angegebenen Definitionen sollen nun anhand eines Beispiels verdeutlicht werden.

Betrachte einen Volladdierer, d. h. eine Funktion

$$fa : \{0,1\}^3 \rightarrow \{0,1\}^2,\ (x_1, x_2, x_3) \mapsto (c, s) \text{ mit } x_1 + x_2 + x_3 = s + 2c$$

(Hierbei handelt es sich bei „+" um die ganzzahlige Addition. s stellt das Summenbit, c das Übertragsbit der binären Summe dar.)
Es gilt:

$$\begin{aligned} c &= x_1 x_2 \vee (x_1 \oplus x_2) x_3 \\ s &= x_1 \oplus x_2 \oplus x_3 \end{aligned}$$

Als Zellenbibliothek wurde $\Omega = \{\oplus,\ \cdot,\ \vee\}$ gewählt.

Der Graph eines Ω–Schaltkreises, der die Funktion fa realisiert, ist in Abbildung 1.1 angegeben. Die zugehörigen Funktionen typ, pe und pa sind in entsprechenden Tabellen zu finden.

Üblicherweise wird man allerdings den Schaltkreis in einer etwas übersichtlicheren Form wie in Abbildung 1.2 darstellen.

Kosten

Im folgenden werden verschiedene Kostenmaße für Ω–Schaltkreise eingeführt, die es ermöglichen sollen, die Kosten verschiedener Realisierungen der gleichen Funktion zu vergleichen.

Dazu wird der Begriff der Zellenbibliothek noch um Zellenflächen und Zellenlaufzeiten erweitert.

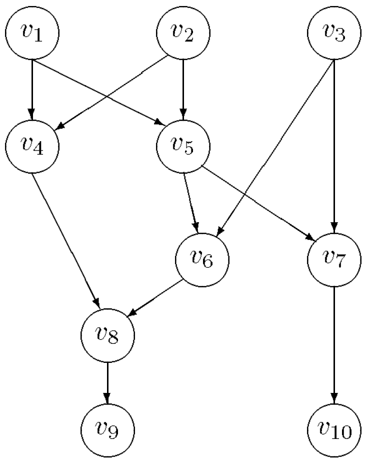
Definition 1.8 (erweiterte Zellenbibliothek) *Eine* **erweiterte Zellenbibliothek** *ist ein Tripel $(\Omega, A_\Omega, T_\Omega)$, das aus einer Zellenbibliothek Ω, und zwei Funktionen $A_\Omega : \Omega \rightarrow \mathbf{R}$ und $T_\Omega : \Omega \rightarrow \mathbf{R}$ besteht. Hierbei ordnet A_Ω jeder Funktion aus Ω eine Gatterfläche zu, T_Ω ordnet jeder Funktion aus Ω eine Gatterlaufzeit zu.*

Definition 1.9 (Kostenmaße)

1. *Die Ω–**Komplexität** eines Ω–Schaltkreises $S = ((V,E), typ, pe, pa)$ ist*

$$C_\Omega(S) = |V \setminus \{v \in V \mid typ(v) \in \{EPAD, APAD\}\}|$$

(D. h. die Ω–Komplexität wird bestimmt durch die Anzahl der Zellen des Schaltkreises, die keine $EPAD$– oder $APAD$–Zellen sind.)



v	$typ(v)$
v_1	$EPAD$
v_2	$EPAD$
v_3	$EPAD$
v_4	and
v_5	$exor$
v_6	and
v_7	$exor$
v_8	or
v_9	$APAD$
v_{10}	$APAD$

i	$pe(i)$
1	v_1
2	v_2
3	v_3

i	$pa(i)$
1	v_9
2	v_{10}

Abbildung 1.1: Schaltkreis zu einem Volladdierer

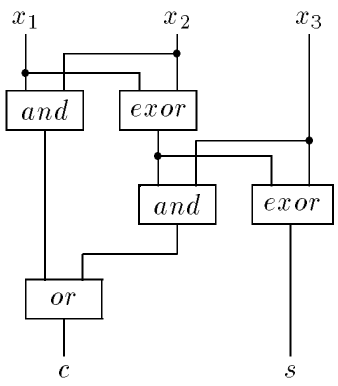


Abbildung 1.2: Schaltkreis zu einem Volladdierer (vereinfachte Darstellung)

Die Ω–Komplexität einer booleschen Funktion $f \in BP_{n,m}$ mit Definitionsmenge D ist

$$C_\Omega(f) = \min\{C_\Omega(S) \mid f_S|_D = f\}$$

Eine Realisierung S von f mit $C_\Omega(S) = C_\Omega(f)$ heißt Ω–optimal.

2. *Die* **Zellfläche** *eines Ω–Schaltkreises $S = ((V,E), typ, pe, pa)$ ist definiert als*

$$A_\Omega(S) = \sum_{\substack{v \in V \\ typ(v) \notin \{EPAD, APAD\}}} A_\Omega(typ(v))$$

(D. h. die Zellfläche ergibt sich als Summe der Gatterflächen sämtlicher Zellen des Schaltkreises.)
Die Zellfläche einer booleschen Funktion $f \in B_{n,m}$ ist entsprechend

$$A_\Omega(f) = \min\{A_\Omega(S) \mid f_S|_D = f\}$$

3. *Die* **Tiefe** *eines Ω–Schaltkreises $S = (G = (V,E), typ, pe, pa)$ ist $D_\Omega(S)$, die größte Anzahl von Zellen auf einem gerichteten Pfad in G ($EPAD$– und $APAD$–Zellen nicht mitgerechnet).*
Die Tiefe einer booleschen Funktion $f \in B_{n,m}$ ist entsprechend

$$D_\Omega(f) = \min\{D_\Omega(S) \mid f_S|_D = f\}$$

4. *Die* **Zellenlaufzeit** *eines Ω–Schaltkreises $S = (G = (V,E), typ, pe, pa)$ $T_\Omega(S)$ ist definiert als die größte Summe der Gatterlaufzeiten auf einem gerichteten Pfad von einem $EPAD$–Knoten zu einem $APAD$–Knoten.*
Die Zellenlaufzeit einer booleschen Funktion $f \in B_{n,m}$ ist

$$T_\Omega(f) = \min\{T_\Omega(S) \mid f_S|_D = f\}$$

Bemerkung 2 *Die Betrachtung der Ω–Komplexität eines Schaltkreises als Kostenmaß ist um so sinnvoller, je ähnlicher die Kosten der einzelnen Vertreter der Zellenbibliothek sind. Dies ist insbesondere dann der Fall, wenn sich die Zellenbibliothek aus „kleinen“ Funktionen zusammensetzt (z. B. Funktionen aus B_2).*

Im folgenden wird Ω meist als $\Omega = B_2$, $\Omega = STD = \{0, 1, and, nand, or, nor, not\}$ oder $\Omega = B_2 \setminus \{exor, equiv\} =: R_2$ gewählt.

(Grundsätzlich läßt sich für $k \geq n$ B_n durch eine Abbildung $i_{n,k}$ in B_k einbetten, wenn man definiert:
$i_{n,k}(f) = f'$ mit

$$f'(x_1, \ldots, x_n, x_{n+1}, \ldots, x_k) = f(x_1, \ldots, x_n)\, \forall (x_1, \ldots, x_k) \in \{0,1\}^k.$$

Daher kann man sich anstelle von $B_1 \cup B_2$ auf die Zellenbibliothek B_2 beschränken.)

Wenn die zugrundliegende Zellenbibliothek aus dem Zusammenhang hervorgeht, wird teilweise auch $C(S)$ bzw. $C(f)$ statt $C_\Omega(S)$ bzw. $C_\Omega(f)$ geschrieben.

1.1.3 Boolesche Ausdrücke

Eine weitere Beschreibungsmöglichkeit boolescher Funktionen aus B_n stellen geklammerte boolesche Ausdrücke dar.

Sie haben eine direkte Entsprechung zu Ω–Schaltkreisen über der Zellenbibliothek $\{0, 1, and, or, not\}$ und werden besonders zur Beschreibung zweistufiger Realisierungen verwendet.

Definition 1.10 (Boolesche Ausdrücke über Var_n**)** *Sei* $Var_n = \{x_1, \ldots, x_n\}$ *eine* n*–elementige Menge, die Menge der* **Variablen**, $E = \{0, 1, (,), \cdot, \vee, \neg\} \cup Var_n$ *und* E^+ *die freie Wörterhalbgruppe über* E*. Dann heißt* $\mathcal{A}(Var_n)$*, der Durchschnitt aller Teilmengen* $L \subseteq E^+$ *mit*

- $\{0, 1\} \cup Var_n \in L$,
- $w_1, \ldots, w_k \in L \Longrightarrow (w_1 \cdot \ldots \cdot w_k) \in L$ *und* $(w_1 \vee \ldots \vee w_k) \in L$,
- $w \in L \Longrightarrow (\neg w) \in L$,

die Menge der **booleschen Ausdrücke über** Var_n*.*

Schreibweise: Für $(\neg w)$ schreiben wir auch $\overline{w}$.

Im folgenden sollen boolesche Ausdrücke über Var_n als Ω–Schaltkreise über der Zellenbibliothek $\Omega = \{0, 1, and, or, not\}$ interpretiert werden. Dies ermöglicht es, Begriffe wie die durch den booleschen Ausdruck definierte Funktion, die Kosten dieser Funktion etc. direkt aus den entsprechenden Definitionen für Ω–Schaltkreise zu übernehmen.

Dazu werden zunächst einige Hilfsfunktionen zum Zusammensetzen von Ω–Schaltkreisen benötigt. (Zur Verdeutlichung, daß es sich bei *and* und *or* um Funktionen aus B_2 handelt, wird hier auch and_2 und or_2 geschrieben.)

- Die Funktion *p_comp* setzt Ω–Schaltkreise mit gleicher Anzahl von Inputs zusammen. Dabei werden die Schaltkreise einfach „parallel" nebeneinandergesetzt und gleiche Inputs werden miteinander „verschmolzen". Es folgt eine formale Definition dieser Funktion:

  Eingabe:
  k Ω–Schaltkreise

  $$S_1 = ((V_1, E_1), typ_1, pe_1, pa_1), \ldots, S_k = ((V_k, E_k), typ_k, pe_k, pa_k),$$

  wobei für $1 \leq i \leq k$ gilt:

  $$|\{v \in V_i \mid typ_i(v) = EPAD\}| = n,$$

  und

  $$|\{v \in V_i \mid typ_i(v) = APAD\}| = m_i$$

$$V_i \cap V_j = \emptyset \quad \forall 1 \leq i, j \leq k,\, i \neq j$$

Ausgabe:
Ein Ω–Schaltkreis

$$S = ((V, E), typ, pe, pa), \text{ wobei gilt:}$$

–
$$V = \{u_1, \ldots, u_n\} \cup \bigcup_{i=1}^{k} (V_i \setminus \{v \in V_i \mid typ_i(v) = EPAD\})$$

Hierbei sollen u_i „neue“ Knoten sein.

–
$$\begin{aligned} E \quad = \quad & \bigcup_{i=1}^{n} \{(u_i, v) \mid \exists j \in \{1, \ldots, k\} \text{ mit } (pe_j(i), v) \in E_j\} \\ & \cup \bigcup_{i=1}^{k} (E_i \setminus \{(u, v) \in E_i \mid typ_i(u) = EPAD\}) \end{aligned}$$

–
$$typ : V \rightarrow (\Omega \cup \{EPAD, APAD\}),$$
$$typ(v) = \begin{cases} typ_i(v), & falls\ v \in V_i \\ EPAD, & falls\ v = u_i \end{cases}$$

–
$$pe : \{1, \ldots, n\} \rightarrow V, \quad pe(i) = u_i$$

–
$$pa : \{1, \ldots, \sum_{i=1}^{k} m_i\} \rightarrow V, \quad pa((\sum_{j=1}^{i-1} m_j) + l) = pa_i(l)$$
$$(1 \leq i \leq k,\ 1 \leq l \leq m_i)$$

- Die Funktion *s_comp* setzt 2 Ω–Schaltkreise zusammen, bei denen der erste genausoviele Ausgänge hat wie der zweite Eingänge. Dabei werden die Schaltkreise „hintereinandergeschaltet“, d. h. die Ausgänge des ersten Schaltkreises werden mit den Eingängen des zweiten „verschmolzen“. Es folgt auch hier eine formale Definition dieser Funktion:

  Eingabe:
  2 Ω–Schaltkreise

  $$S_1 = ((V_1, E_1), typ_1, pe_1, pa_1),\ S_2 = ((V_2, E_2), typ_2, pe_2, pa_2),$$

  wobei gilt:

  $$|\{v \in V_1 \mid typ_1(v) = APAD\}| = |\{v \in V_2 \mid typ_2(v) = EPAD\}|$$
  $$V_1 \cap V_2 = \emptyset$$

  Ausgabe:
  Ein Ω–Schaltkreis

  $$S = ((V, E), typ, pe, pa), \text{ wobei gilt:}$$

–
$$\begin{aligned} V &= (V_1 \cup V_2) \setminus \\ &\quad (\{v \in V_1 \mid typ_1(v) = APAD\} \cup \{v \in V_2 \mid typ_2(v) = EPAD\}) \end{aligned}$$

–
$$\begin{aligned} E &= (E_1 \setminus \{(u,v) \in E_1 \mid typ_1(v) = APAD\}) \\ &\quad \cup (E_2 \setminus \{(u,v) \in E_2 \mid typ_2(u) = EPAD\}) \\ &\quad \cup \{(u,v) \mid \exists (u, pa_1(i)) \in E_1 \text{ und } \exists (pe_2(i), v) \in E_2\} \end{aligned}$$

–
$$typ : V \to (\Omega \cup \{EPAD, APAD\}),$$
$$typ(v) = \begin{cases} typ_1(v), & falls\ v \in V_1 \\ typ_2(v), & falls\ v \in V_2 \end{cases}$$

- $pe = pe_1$
- $pa = pa_2$

- Die Funktion AND_n liefert einen Schaltkreis für die *and*–Funktion mit $n \geq 1$ Eingängen. Der Schaltkreis ist ein balancierter Baum aus and_2–Zellen. AND_n läßt sich folgendermaßen rekursiv definieren:

  - Falls $n = 1$, dann gilt:
  $$AND_1 = ((V,E), typ, pe, pa) \text{ mit}$$
    * $V = \{v_1, v_2\}$
    * $E = \{(v_1, v_2)\}$
    * $typ(v_1) = EPAD$, $typ(v_2) = APAD$
    * $pe : \{1\} \to V$, $pe(1) = v_1$
    * $pa : \{1\} \to V$, $pa(1) = v_2$
  - Falls $n = 2$, dann gilt:
  $$AND_2 = ((V,E), typ, pe, pa) \text{ mit}$$
    * $V = \{v_1, v_2, v_3, v_4\}$
    * $E = \{(v_1, v_3), (v_2, v_3), (v_3, v_4)\}$
    * $typ(v_1) = typ(v_2) = EPAD$, $typ(v_3) = and_2$, $typ(v_4) = APAD$
    * $pe : \{1, 2\} \to V$, $pe(1) = v_1$, $pe(2) = v_2$
    * $pa : \{1\} \to V$, $pa(1) = v_4$
  - Falls $n > 2$: Angenommen
  $$AND_{\lfloor n/2 \rfloor} = ((V_1, E_1), typ_1, pe_1, pa_1),$$
  $$AND_{\lceil n/2 \rceil} = ((V_2, E_2), typ_2, pe_2, pa_2),$$
  $$V_1 \cap V_2 = \emptyset$$

  Dann gilt für AND_n:
  $$AND_n = s\_comp(((V,E), typ, pe, pa), AND_2), \text{ wobei}$$

- $V = V_1 \cup V_2$
- $E = E_1 \cup E_2$
- $$typ : V \to \Omega,\ typ(v) = \begin{cases} typ_1(v), & falls\, v \in V_1 \\ typ_2(v), & falls\, v \in V_2 \end{cases}$$
- $$pe : \{1, \ldots n\} \to V,\ pe(i) = \begin{cases} pe_1(i)\ f\ddot{u}r\ 1 \leq i \leq \lfloor n/2 \rfloor \\ pe_2(i - \lfloor n/2 \rfloor)\ f\ddot{u}r\ \lfloor n/2 \rfloor < i \leq n \end{cases}$$
- $pa : \{1, 2\} \to V,\ pa(1) = pa_1(1),\ pa(2) = pa_2(1)$

- Analog definiert man die Funktion OR_n, die für die *or*–Funktion mit $n \geq 1$ Eingängen einen Schaltkreis liefert.
- Mit INV wird der Ω–Schaltkreis bestehend aus einer Inverterzelle (*not*–Zelle) (und einer $EPAD$– und $APAD$–Zelle) bezeichnet.
- $X_{n,i}$ sei ein Ω–Schaltkreis aus n $EPAD$–Zellen und einer $APAD$–Zelle. Die einzige Kante führt von der Zelle $pe(i)$ zur $APAD$–Zelle.
- Mit $NULL_n$ wird der Ω–Schaltkreis bestehend aus einer 0–Zelle (verbunden mit einer $APAD$–Zelle) und n unverbundenen $EPAD$–Zellen bezeichnet.
- Entsprechend wird mit $EINS_n$ der Ω–Schaltkreis bestehend aus einer 1–Zelle (verbunden mit einer $APAD$–Zelle) und n unverbundenen $EPAD$–Zellen bezeichnet.

Definition 1.11 (Interpretetion boolescher Ausdrücke) *Die* **Interpretation** *boolescher Ausdrücke über Var_n ist eine Abbildung SK von $\mathcal{A}(Var_n)$ auf Ω–Schaltkreise mit $\Omega = \{0, 1, and_2, or_2, not\}$. Sie hat folgende Eigenschaften:*

$$\begin{aligned}
SK(0) &= NULL_n \\
SK(1) &= EINS_n \\
SK(x_i) &= X_{n,i} \\
SK((w_1 \cdot \ldots \cdot w_k)) &= s_comp(p_comp(SK(w_1), \ldots, SK(w_k)), AND_k) \\
SK((w_1 \vee \ldots \vee w_k)) &= s_comp(p_comp(SK(w_1), \ldots, SK(w_k)), OR_k) \\
SK((\overline{w})) &= s_comp(SK(w), INV)
\end{aligned}$$

Definition 1.12 (durch booleschen Ausdruck definierte Funktion) *Die durch einen booleschen Ausdruck w definierte Funktion $\Phi(w)$ ist die Funktion, die durch den zugehörigen Schaltkreis $SK(w)$ definiert wird.*

Definition 1.13 (Kosten eines booleschen Ausdrucks) *Die Kosten eines booleschen Ausdrucks (Ω–Komplexität, Zellfläche, Tiefe, Zellenlaufzeit) sind gegeben durch die Kosten des zugehörigen Ω–Schaltkreises ($\Omega = \{0, 1, and, or, not\}$).*

Definition 1.14 (Äquivalenz boolescher Ausdrücke) *Zwei boolesche Ausdrücke heißen* **äquivalent**, *falls sie die gleiche totale Funktion definieren.*

Zur Beschreibung zweistufiger Realisierungen durch boolesche Ausdrücke werden noch folgende Definitionen benötigt:

Definition 1.15 (boolesches Monom)
$m \in \mathcal{A}(\{x_1, \ldots x_n\})$ *heißt* **boolesches Monom**, *falls*

$$m = x_{i_1}^{\epsilon_1} \cdot \ldots \cdot x_{i_j}^{\epsilon_j} \text{ mit } 1 \leq j \leq n, \text{ und } \epsilon_k \in \{0,1\} \text{ für } k = 1, \ldots, j,$$

$$\text{wobei } x_{i_k}^1 = x_{i_k} \text{ und } x_{i_k}^0 = \overline{x_{i_k}} \text{ gilt.}$$

Unter der **Variablenmenge** *von m versteht man die Menge*

$$V(m) = \{x_{i_1}, \ldots, x_{i_j}\}.$$

Das **Monom** *heißt* **vollständig**, *wenn $V(m) = \{x_1, \ldots, x_n\}$ gilt.*

Definition 1.16 (boolesches Polynom)
$p \in \mathcal{A}(\{x_1, \ldots x_n\})$ *heißt* **boolesches Polynom**, *falls gilt:*

- $p = 0$ *oder*
- $p = 1$ *oder*
- $p = m_1 \vee \ldots \vee m_k$, *wobei $\forall i \in \{1, \ldots, k\}$ m_i ein Monom ist.*

Bei zweistufiger Logiksynthese wird zu einer gegebenen Funktion f aus $S(D)$, $D \subseteq \{0,1\}^n$ ein Polynom $p \in \mathcal{A}(Var_n)$ mit möglichst geringen Kosten (geringer Komplexität) gesucht, so daß $\Phi(p)(x) = f(x)\ \forall x \in D$ gilt.

Bemerkung 3 *Üblicherweise werden bei der Bestimmung der Kosten von booleschen Polynomen Inverter nicht mitgezählt. Man kann dies erreichen, indem man als zugrundeliegende Zellenbibliothek R_2 wählt. Ausgehend von dem durch Definition 1.11 bestimmten $\{0,\ 1,\ and_2,\ or_2,\ not\}$–Schaltkreis $S = SK(p)$ zu einem booleschen Polynom p erhält man einen R_2–Schaltkreis S' zu p, indem man die not–Zellen zusammen mit ihren eindeutig bestimmten Nachfolgerzellen jeweils durch eine einzige Zelle ersetzt, die einen passenden Typ aus R_2 hat. Unter der Komplexität von p versteht man dann die R_2–Komplexität des Schaltkreises S'.*

Im folgenden ist die Komplexität boolescher Polynome in diesem Sinne zu verstehen.

Definition 1.17 (Minimalpolynom)
Sei f eine boolesche Funktion aus $S(D)$, $D \subseteq \{0,1\}^n$. Ein Polynom $p \in \mathcal{A}(Var_n)$ heißt **Minimalpolynom** *von f, falls*

- $\forall x \in D \quad \Phi(p)(x) = f(x)$ *und*
- $C(p) \leq C(p')$ *für alle Polynome p' mit $\Phi(p')(x) = f(x) \quad \forall x \in D$*

Zur Beschreibung boolescher Funktionen aus $B_{n,m}$ werden schließlich noch boolesche Polynome mit m Ausgängen definiert.

Definition 1.18 (Boolesche Polynome mit m Ausgängen)
$P = (p_1, \ldots, p_m)$ heißt **boolesches Polynom mit m Ausgängen** *über Var_n, wenn für alle $i \in \{1, \ldots, m\}$ p_i ein boolesches Polynom über Var_n ist.*

Der Schaltkreis zu einem booleschen Polynom $(p_1, \ldots, p_m)$ mit m Ausgängen ergibt sich im wesentlichen durch paralleles Nebeneinandersetzen der Schaltkreise zu den booleschen Polynomen p_i. Der einzige Unterschied ist, daß Monome, die in mehreren der Polynome p_i vorkommen, nur einmal erzeugt werden.

Somit ergibt sich folgende formale Definition der Interpretetion boolescher Polynome mit m Ausgängen:

Definition 1.19 (Interpretetion boolescher Polynome mit m Ausgängen)
Die **Interpretation** *boolescher Polynome mit m Ausgängen über Var_n ist eine Abbildung $POLSK$ von $(\mathcal{A}(Var_n))^m$ auf Ω–Schaltkreise mit $\Omega = \{0, 1, and, or, not\}$.*
Sei $P = (p_1, \ldots, p_m)$ ein boolesches Polynom mit m Ausgängen über Var_n. Sei für $1 \leq i \leq m$ $p_i = m_1^{(i)} \vee \ldots \vee m_{k_i}^{(i)}$, $p_i = 0$ oder $p_i = 1$.
Sei

$$\{m_1, \ldots, m_k\} = \bigcup_{i=1}^{m} \cup_{j=1}^{k_i} m_j^{(i)}$$

die Menge der Monome, die in $p_1, \ldots, p_m$ vorkommen.

Sei

$$M' = p\_comp(SK(m_1), \ldots, SK(m_k))$$

Falls es ein p_i gibt mit $p_i = 0$, dann

$$M'' = p\_comp(M', NULL_n),$$

sonst

$$M'' = M'.$$

Falls es ein p_i gibt mit $p_i = 1$, dann

$$M = ((V_M, E_M), typ_M, pe_M, pa_M) = p\_comp(M'', EINS_n),$$

sonst

$$M = ((V_M, E_M), typ_M, pe_M, pa_M) = M''.$$

Seien für $1 \leq i \leq m$ *mit* $p_i \neq 0,\ p_i \neq 1$

$$O_i = ((V_i, E_i), typ_i, pe_i, pa_i) = OR_{k_i} \ mit$$

$$V_i \cap V_M = \emptyset \ und \ V_i \cap V_j = \emptyset \ für \ i \neq j$$

Dann gilt

$$POLSK(P) = ((V, E), typ, pe, pa) \ mit$$

-
$$\begin{aligned} V &= V_M \setminus \{v \in V_M \mid typ_M(v) = APAD\} \\ &\cup \bigcup_{\substack{1 \leq i \leq m \\ p_i \neq 0, p_i \neq 1}} (V_i \setminus \{v \in V_i \mid typ_i(v) = EPAD\}) \\ &\cup \bigcup_{p_i = 0} \{u_i\} \\ &\cup \bigcup_{p_i = 1} \{u_i\} \end{aligned}$$

-
$$\begin{aligned} E &= (E_M \setminus \{(u, v) \in E_M \mid typ_M(v) = APAD\}) \\ &\cup \bigcup_{\substack{1 \leq i \leq m \\ p_i \neq 0, p_i \neq 1}} (E_i \setminus \{(u, v) \in E_i \mid typ_i(u) = EPAD\}) \\ &\cup \bigcup_{\substack{1 \leq i \leq m \\ p_i \neq 0, p_i \neq 1}} \{(u, v) \mid \exists (u, pa_M(q)) \in E_M,\, \exists (pe_i(r), v) \in E_i \ und \ m_r^{(i)} = m_q\} \\ &\cup \bigcup_{p_i = 0} \{(u, u_i) \ mit \ u \in V_M,\, typ_M(u) = 0\} \\ &\cup \bigcup_{p_i = 1} \{(w, u_i) \ mit \ w \in V_M,\, typ_M(w) = 1\} \end{aligned}$$

-
$$typ : V \rightarrow (\Omega \cup \{EPAD, APAD\}),$$

$$typ(v) = \begin{cases} typ_M(v), & falls \ v \in V_M \\ typ_i(v), & falls \ v \in V_i,\ 1 \leq i \leq m,\ p_i \neq 0, p_i \neq 1 \\ APAD, & falls \ v = u_i, 1 \leq i \leq m,\ p_i = 0 \ oder \ p_i = 1 \end{cases}$$

- $pe = pe_M$
-
$$pa(i) = \begin{cases} pa_i(1), & falls \ 1 \leq i \leq m,\ p_i \neq 0, p_i \neq 1 \\ u_i, & falls \ p_i = 0 \ oder \ p_i = 1 \end{cases}$$

Definition 1.20 (Kosten boolescher Polynome mit m Ausgängen)
Die **Kosten** *eines booleschen Polynoms P mit m Ausgängen sind gegeben durch die Kosten des zugehörigen Ω–Schaltkreises $POLSK(P)$.*

Definition 1.21 (Minimalpolynom mit m Ausgängen) *Sei f eine boolesche Funktion aus $B_{n,m}$. Ein Polynom $P \in (\mathcal{A}(Var_n))^m$ heißt* **Minimalpolynom** *von f, falls*

- *f gleich der durch $POLSK(P)$ definierten Funktion ist.*
- *$C(P) \leq C(P')$ für alle Polynome P', bei denen die durch $POLSK(P')$ definierte Funktion gleich f ist,*

1.1.4 Funktionsgraphen

Boolesche Funktionen aus B_n kann man auch durch Funktionsgraphen bzw. OBDD's (ordered binary decision diagrams) repräsentieren ([Bry86]). Funktionsgraphen sind wie folgt definiert:

Definition 1.22 (Funktionsgraph)
Sei $X = \{x_1, \ldots, x_n\}$ eine Menge von Variablen mit einer linearen Ordnung $<$ auf X, wobei $x_i < x_j$ genau dann, wenn $i < j$.

Ein **Funktionsgraph (OBDD)** *F über der Menge X ist ein Paar (G, m) aus einem kantenorientierten und knotenorientierten Graphen $G = (V, E)$ und einer Markierungsfunktion $m : V \to (X \cup \{0, 1\})$.*

G hat genau eine Quelle $q \in V$ und genau zwei Senken $s_0, s_1 \in V$. Jeder Knoten $v \in V \setminus \{s_0, s_1\}$ hat genau 2 Söhne: den 0–Sohn v^0 und den 1–Sohn v^1.

Die Markierungsfunktion m ordnet jedem Knoten $v \in V \setminus \{s_0, s_1\}$ eine Markierung $m(v) \in X$ zu. Weiterhin gilt für die Senken s_0 und s_1 $m(s_0) = 0$ und $m(s_1) = 1$.

Für jeden Knoten $v \in V \setminus \{s_0, s_1\}$, für den ein Sohn v^ϵ ($\epsilon \in \{0, 1\}$) keine Senke ist, gilt: $m(v) < m(v^\epsilon)$.

Der Funktionsgraph ist so definiert, daß auf jedem Pfad von der Quelle zu einer der Senken jede Variable aus X höchstens einmal als Markierung auftreten kann und zwar in der durch die Ordnung auf X vorgegebenen Reihenfolge.

Ein Funktionsgraph über $X = \{x_1, \ldots, x_n\}$ definiert eine boolesche Funktion $f \in B_n$:

Definition 1.23 (Durch Funktionsgraph definierte Funktion)
Sei $F = (G, m)$ ein Funktionsgraph über der Menge $X = \{x_1, \ldots, x_n\}$ von Variablen. Sei $v \in V$ ein Knoten von G.

- *Sei $m(v) = 0$. Dann ist die durch v definierte Funktion eine 0–stellige Funktion (Konstante) $f_v \in B_0$, $f_v() = 0$.*
- *Sei $m(v) = 1$. Dann ist die durch v definierte Funktion eine 0–stellige Funktion (Konstante) $f_v \in B_0$, $f_v() = 1$.*

- *Sei* $m(v) = x_i$. *Sei* v^0 *der* 0–*Sohn von* v, v^1 *der* 1–*Sohn. Sei* $f_{v^0} \in B_{n_0}$ *die durch* v^0 *definierte Funktion,* $f_{v^1} \in B_{n_1}$ *die durch* v^1 *definierte Funktion.*

  Dann ist die durch v *definierte Funktion* $f_v \in B_{n-i+1}$ *mit*

$$f_v(x_i, \ldots, x_n) = \overline{x_i} f_{v^0}(x_{n-n_0+1}, \ldots, x_n) + x_i f_{v^1}(x_{n-n_1+1}, \ldots, x_n).$$

Sei die einzige Quelle $q \in V$ *markiert mit* x_j. *Die durch* $F = (G, m)$ *definierte Funktion* $f_F \in B_n$ *ist dann gegeben durch*

$$f_F(x_1, \ldots, x_n) = f_q(x_j, \ldots, x_n) \, \forall (x_1, \ldots, x_n) \in \{0, 1\}^n.$$

Bezeichnung 4 *Sei* $F = (G = (V, E), m)$ *ein Funktionsgraph über der Variablenmenge* $X = \{x_1, \ldots, x_n\}$. *Sei* $\epsilon = (\epsilon_1, \ldots, \epsilon_k) \in \{0, 1\}^k$, $k \leq n$. *Dann ist der durch* ϵ *erreichbare Knoten von* G *durch folgenden Algorithmus definiert:*

1. *Beginne bei der Quelle* q *von* G. $v = q$.

2. *Solange* $m(v) = x_i$ *mit* $i < k$: *Gehe über zum* ϵ_i–*Sohn von* v, *d.h.* $v = v^{\epsilon_i}$.

Der durch ϵ *erreichbare Knoten ist dann der Knoten* v, *an dem man zum Schluß des Verfahrens angekommen ist.*

Bemerkung 4

- *Ist* $\epsilon = (\epsilon_1, \ldots, \epsilon_n)$, *dann ist der durch* ϵ *erreichbare Knoten eine Senke* s *von* G. *Es gilt:*

$$f_F(\epsilon_1, \ldots, \epsilon_n) = m(s).$$

- *Sei* v *der durch* $\epsilon = (\epsilon_1, \ldots, \epsilon_k)$ *mit* $k < n$ *erreichbare Knoten. Dann läßt sich der Untergraph von* G, *der alle Knoten umfaßt, die von* v *aus erreicht werden können, zusammen mit den zugehörigen Markierungen als Funktionsgraph* $F' = (G', m')$ *über der Variablenmenge* $X' = \{x_{k+1}, \ldots, x_n\}$ *interpretieren.*

  Für die durch F' *definierte Funktion gilt:*

$$f_{F'}(x_{k+1}, \ldots, x_n) = f_F(\epsilon_1, \ldots, \epsilon_k, x_{k+1}, \ldots, x_n)$$
$$\forall \, (x_{k+1}, \ldots, x_n) \in \{0, 1\}^{n-k}$$

Bezeichnung 5 *Ist* $f \in B_n$, *so wird die Funktion* $f' \in B_{n-k}$ *mit*

$$f'(x_{k+1}, \ldots, x_n) = f(\epsilon_1, \ldots, \epsilon_k, x_{k+1}, \ldots, x_n)$$
$$\forall \, (x_{k+1}, \ldots, x_n) \in \{0, 1\}^{n-k}$$

als **Kofaktor** *von* f *hinsichtlich*

$$x^{\epsilon_1} \ldots x^{\epsilon_k}$$

bezeichnet. Man schreibt für f' *auch*

$$f_{x_1^{\epsilon_1} \ldots x_k^{\epsilon_k}}.$$

Schließlich wird noch ein spezieller Funktionsgraph, der ROBDD (= reduced ordered binary decision diagram) definiert. ROBDD's haben den Vorteil, daß zu jeder Funktion aus B_n bei vorgegebener Ordnung auf den Variablen ein eindeutiger Graph existiert.

Definition 1.24 (reduzierter Funktionsgraph bzw. ROBDD) *Ein Funktionsgraph $F = (G = (V, E), m)$ heißt genau dann* **reduziert***,*

- *wenn es keinen Knoten gibt, dessen 0–Sohn und dessen 1–Sohn identisch sind und*
- *wenn es keine zwei Knoten v_1 und v_2 in V gibt, so daß der Untergraph aller Knoten, die von v_1 aus erreicht werden können und der Untergraph aller Knoten, die von v_2 aus erreicht werden können, die gleiche Funktion definieren.*

Bemerkung 5 *Ist $F = (G, m)$ ein reduzierter Funktionsgraph über der Variablenmenge $\{x_1, \ldots, x_n\}$, der eine Funktion f definiert, so läßt sich aus der Definition eines reduzierten Funktionsgraphen folgende Aussage schließen:*
Sind 2 Kofaktoren

$$f_{x_1^{\epsilon_1} \ldots x_k^{\epsilon_k}} \textit{ und } f_{x_1^{\delta_1} \ldots x_k^{\delta_k}}$$

gleich, so sind die durch ϵ bzw. δ erreichbaren Knoten von G identisch.

1.2 Boolesche Algebren

Grundlegende Resultate über Verbände und boolesche Algebren sind in entsprechenden Lehrbüchern wie z. B. [Hot74] zu finden. Der Vollständigkeit halber sollen hier die Ergebnisse kurz aufgeführt werden, die zum Verständnis von Kapitel 3 und 4 benötigt werden. Für eine umfassendere Darstellung sei auf [Hot74] verwiesen.

1.2.1 Definition und Gesetze

Zunächst wird der Begriff der „booleschen Algebra" definiert. Eine boolesche Algebra ist ein spezieller Verband. Verbände kann man entweder aus einer ordnungstheoretischen oder einer algebraischen Sicht definieren. Je nach Situation wird auf eine der beiden Definitionen zurückgegriffen. Daß dieses Vorgehen berechtigt ist, zeigt Lemma 1.1.

Definition 1.25 (Verband (ordnungstheoretisch))
$(M, \leq)$ *heißt* **Verband***, wenn*

- $\leq$ *eine Halbordnung auf M ist und*

- *für je 2 Elemente aus M eine kleinste obere Schranke und eine größte untere Schranke existieren, d.h. wenn es zu $a, b \in M$*

$$c =: sup(a, b) \text{ und } d =: inf(a, b) \in M$$

gibt mit

(V1) $c \geq a, b$ *und* $c \leq m \; \forall m \in M$ *mit* $m \geq a, b$

(V2) $d \leq a, b$ *und* $d \geq m \; \forall m \in M$ *mit* $m \leq a, b$

Definition 1.26 (Verband (algebraisch)) $(M, \cup, \cdot)$ *heißt* **Verband**, *wenn* $\cup, \cdot$ *binäre Operationen auf M sind und $\forall a, b, c \in M$ (A1), (A2) und (A3) erfüllt sind.*

(A1)	$a \cup b = b \cup a$	$a \cdot b = b \cdot a$	*(Kommutativität)*
(A2)	$a \cup (b \cup c) = (a \cup b) \cup c$	$a \cdot (b \cdot c) = (a \cdot b) \cdot c$	*(Assoziativität)*
(A3)	$a \cup (a \cdot b) = a$	$a \cdot (a \cup b) = a$	*(Absorption)*

Die Verbindung zwischen den beiden Definitionen stellt Lemma 1.1 her:

Lemma 1.1 *1. Sei $(M, \leq)$ Verband nach ordnungstheoretischer Definition. Seien $\cup : M \times M \to M$, $\cdot : M \times M \to M$ definiert durch:*

$$\begin{aligned} a \cup b &= sup(a, b) \quad \forall a, b \in M \\ a \cdot b &= inf(a, b) \quad \forall a, b \in M. \end{aligned}$$

Dann ist $(M, \cup, \cdot)$ ein Verband nach algebraischer Definition.

2. Sei $(M, \cup, \cdot)$ Verband nach algebraischer Definition. Sei $\leq \subseteq M \times M$ wie folgt definiert:

$$a \leq b \quad :\Longleftrightarrow \quad a \cdot b = a \quad \forall a, b \in M$$

Dann ist $(M, \leq)$ ein Verband nach der ordnungstheoretischen Definition.

Definition 1.27 (Boolesche Algebra)
$(M, \cup, \cdot, \bar{\ })$ *heißt* **boolesche Algebra**, *wenn $(M, \cup, \cdot)$ ein Verband ist mit $|M| > 0$, $\bar{\ }$ eine unäre Operation auf M ist und wenn für alle $a, b, c \in M$ die Axiome (A4) und (A5) gelten:*

(A4)

$$\left.\begin{aligned} a \cup (b \cdot c) &= (a \cup b) \cdot (a \cup c) \\ a \cdot (b \cup c) &= (a \cdot b) \cup (a \cdot c) \end{aligned}\right\} \begin{array}{l}\textit{Distributiv-} \\ \textit{gesetze}\end{array}$$

(A5)

$$\begin{aligned} a \cup (b \cdot \overline{b}) &= a \\ a \cdot (b \cup \overline{b}) &= a \end{aligned}$$

Aus den Axiomen der booleschen Algebra folgt eine Reihe allgemein bekannter Gesetze, die in Satz 1.1 zusammengefaßt sind:

Satz 1.1

- *(Nullelement, Einselement)*
  Jede boolesche Algebra M besitzt genau ein Nullelement 0 und genau ein Einselement 1 mit

  $$a \cup 0 = a \text{ und } a \cdot 0 = 0 \quad \forall a \in M$$

  $$a \cdot 1 = a \text{ und } a \cup 1 = 1 \quad \forall a \in M$$

  Es gilt:

  $$a \cdot \overline{a} = 0 \qquad a \cup \overline{a} = 1 \quad \forall a \in M$$

- *(Eindeutigkeit des Komplements)*
  Falls für $a, b \in M$ gilt:

  $$a \cup b = 1 \text{ und } a \cdot b = 0,$$

  so ist $b = \overline{a}$.
  ($\overline{a}$ heißt Komplement von a.)

- *(Doppeltes Komplement)*

  $$\forall a \in M \text{ gilt: } \overline{\overline{a}} = a$$

- *(Idempotenz)*

  $$\forall a \in M \text{ gilt: } a \cdot a = a, \quad a \cup a = a$$

- *(de Morgan–Formeln)*
  $\forall a, b \in M$ gilt:

  $$\begin{aligned} \overline{a \cup b} &= \overline{a} \cdot \overline{b} \text{ und} \\ \overline{a \cdot b} &= \overline{a} \cup \overline{b} \end{aligned}$$

- $\overline{0} = 1, \qquad \overline{1} = 0$

1.2.2 Beispiele

Es werden nun einige Beispiele für boolesche Algebren aufgezählt, die im folgenden eine Rolle spielen.

- Sei $M = \{0, 1\}$,

  $$\forall x, y \in M \qquad x \vee y = 0 \iff x = y = 0,$$

  $$\forall x, y \in M \qquad x \cdot y = 1 \iff x = y = 1,$$

  $$\overline{0} = 1 \qquad \overline{1} = 0.$$

  $(M, \vee, \cdot, \overline{\ })$ ist eine boolesche Algebra.

- Die Menge $S(D)$ aller booleschen Funktionen mit Definitionsmenge D bildet mit den Operationen der Disjunktion, Konjunktion und Negation eine boolesche Algebra.
  $((S(D), \vee, \cdot, \overline{\ })$ ist eine boolesche Algebra.)

- Sei M die Potenzmenge einer beliebigen Menge S, d.h. $M = 2^S$. Sei $\cup$ die Vereinigung, $\cap$ der Durchschnitt und $\overline{\ }$ das Komplement bezüglich S. Dann ist $(2^S, \cup, \cap, \overline{\ })$ eine boolesche Algebra.

1.2.3 Atome

Jede boolesche Algebra $(M, \cup, \cdot, \overline{\ })$ ist gleichzeitig auch ein Verband. Auf Verbänden ist durch $\leq$ eine Ordnungsrelation eingeführt. Man interessiert sich nun für die kleinsten Elemente von M, die größer als 0 sind. Diese werden als Atome bezeichnet.

Definition 1.28 (Atom)
Ist $a \in M, a \neq 0$ und folgt aus $0 \leq b \leq a$

$$b = 0 \qquad \textit{oder} \qquad b = a,$$

dann ist a Atom von M.

Eine alternative Charakterisierung eines Atoms liefert Satz 1.2.

Satz 1.2
Sei $a \in M$ und $a \neq 0$.
a ist genau dann Atom von M, wenn für alle $c \in M$ gilt:

$$a \cdot c = a \textit{ oder } a \cdot c = 0.$$

Beweis:
Nach Definition der Relation $\leq$ gilt:

$$ba = b \iff b \leq a$$

„$\Longrightarrow$“: Sei $a \in M, a \neq 0$.
Es gelte $\forall c \in M$: $ac = a$ oder $ac = 0$.
Aus $0 \leq b \leq a$ folgt:

$$ba = b \qquad (1)$$

Nach Voraussetzung gilt aber:

$$ab = a \text{ oder } ab = 0 \qquad (2)$$

$$\overset{(1),\,(2)}{\Longrightarrow} \quad b = a \text{ oder } b = 0.$$

„$\Longleftarrow$“: Sei $a \in M$ ein Atom, d.h. falls

$$0 \leq b \leq a, \text{ dann } b = 0 \text{ oder } b = a.$$

Es gilt $\forall c \in M$:

$$0 \leq ac \leq a \text{ (da } 0 \cdot ac = 0 \text{ und } a \cdot ac = ac)$$

$$\Longrightarrow \quad ac = 0 \text{ oder } ac = a$$

□

Aus Satz 1.2 ergibt sich leicht das folgende Lemma:

Lemma 1.2 *Sind a, b Atome von M und ist $a \neq b$, dann ist $ab = 0$.*

Beispiel:
Betrachte die boolesche Algebra B_n.
Definiere für $\epsilon \in \{0,1\}^n$ die Funktionen $X^\epsilon \in B_n$:

$$X^\epsilon(\alpha) = \begin{cases} 1 & \text{falls } \alpha = \epsilon \\ 0 & \text{sonst.} \end{cases}$$

Dann ist

$$\mathbf{A} = \{X^\epsilon \mid \epsilon \in \{0,1\}^n\}$$

die Menge der Atome von B_n.

Beweis:

- $\forall \epsilon \in \{0,1\}^n$ ist X^ϵ Atom von B_n:
  Sei $f \in B_n$ beliebig.
  Für $b = f \cdot X^\epsilon$ gilt

$$b(\alpha) = \begin{cases} f(\epsilon) & \text{für } \alpha = \epsilon \\ 0 & \text{sonst.} \end{cases}$$

$$\Longrightarrow \quad b = 0 \text{ oder } b = X^\epsilon$$

- Jedes Atom ist in $\mathbf{A}$ enthalten:
  Durch Einsetzen von $\alpha \in \{0,1\}^n$ prüft man direkt nach:

$$\forall f \in B_n : \quad f = \bigcup_{\epsilon \in \{0,1\}^n} f(\epsilon) X^\epsilon.$$

  Es gilt:

$$\begin{aligned} f = f \cdot f &= \Big(\bigcup_{\epsilon \in \{0,1\}^n} f(\epsilon) X^\epsilon \Big) \cdot f \\ &= \bigcup_{\epsilon \in \{0,1\}^n} f(\epsilon)(X^\epsilon \cdot f) \end{aligned}$$

  Falls f Atom ist, so gilt nach Lemma 1.2:

$$X^\epsilon \cdot f = \begin{cases} 0 & \text{für } f \neq X^\epsilon \\ X^\epsilon & \text{sonst.} \end{cases}$$

  Wäre $f \neq X^\epsilon$ für alle $\epsilon \in \{0,1\}^n$, dann wäre

$$f = \bigcup_{\epsilon \in \{0,1\}^n} f(\epsilon) \cdot 0 = 0$$

  und f wäre kein Atom.
  $\Longrightarrow \quad f = X^\epsilon$ für ein $\epsilon \in \{0,1\}^n$
  $\Longrightarrow \quad f \in \mathbf{A}$

□

Der nun folgende Satz zeigt, daß die Elemente einer endlichen, mindestens zweielementigen Algebra sich eindeutig als Vereinigung von Atomen darstellen lassen.

Satz 1.3 *Sei M eine endliche boolesche Algebra mit mindestens 2 Elementen und $A(M)$ die Menge der Atome von M.*
Dann gibt es für alle $f \in M$ eine eindeutige Darstellung

$$f = \bigcup_{a \in A_f} a, \text{ wobei } A_f \subseteq A(M).$$

Beweis:
Zu zeigen sind Existenz und Eindeutigkeit einer solchen Darstellung.

Existenz:

1. Zeige zunächst, daß die Menge der Atome nicht leer ist:
   Beweis durch Konstruktion einer Folge paarweise verschiedener Elemente von M, die mit einem Atom abbricht:
   Nach Voraussetzung gibt es ein $a_0 \in M$ mit $a_0 \neq 0$.
   Falls $\forall b \in M$ mit $0 \leq b \leq a_0$ $b = 0$ oder $b = a_0$, so ist a_0 ein Atom.
   Sonst gibt es a_1 mit $0 \leq a_1 \leq a_0$ und $a_1 \neq 0$ und $a_1 \neq a_0$.
   Entweder ist a_1 Atom oder es gibt aufgrund der gleichen Schlußweise ein $a_2 \neq 0$, $a_2 \neq a_1$ mit $0 \leq a_2 \leq a_1$ und so weiter. Man erhält so eine Folge von $a_i \in M$ mit $a_{i+1} \leq a_i$, $a_{i+1} \neq a_i$ für alle Elemente der Folge.
   Somit sind alle Elemente der Folge verschieden, denn würde für $k > i$ $a_k = a_i$ gelten, dann wäre $a_k = \ldots = a_{i+1} = a_i$ im Widerspruch zu $a_{i+1} \neq a_i$ (denn $a_k \leq \ldots \leq a_{i+1} \leq a_i$ und $\leq$ ist eine Halbordnung).
   Da M endlich ist, muß die Konstruktion also nach endlich vielen Schritten mit einem Atom abbrechen.
   $\Longrightarrow \quad A(M) \neq \emptyset$.

2. Als nächstes wird gezeigt, daß

$$v := \bigcup_{a \in A(M)} a = 1.$$

   Annahme: $\overline{v} \neq 0$.
   Dann gibt es nach voriger Konstruktion ein Atom $a \leq \overline{v}$. Es gilt aber auch $a \leq v$.

$$\Longrightarrow \qquad av = a\overline{v} = a$$

   Nach Multiplikation mit $\overline{v}$ ergibt sich:

$$0 = av\overline{v} = a\overline{v}\,\overline{v} = a\overline{v} = a$$

   $\Longrightarrow$ Widerspruch zur Tatsache, daß a Atom ist!
   Also ist $\overline{v} = 0 \Longrightarrow v = 1$.

3. Unter Ausnutzung dieser Beziehung wird die Existenz der gesuchten Darstellung nun bewiesen:
   Sei $f \in M$ beliebig, dann gilt:

$$f = f \cdot 1 = f \cdot \bigcup_{a \in A(M)} a = \bigcup_{a \in A(M)} (f \cdot a) = \bigcup_{a \in A_f} a \text{ mit}$$

$$A_f = \{a \in A(M) \mid f \cdot a = a\},$$

   da $a \in A(M)$ Atome sind und folglich $f \cdot a = a$ oder $f \cdot a = 0$ gilt.

Eindeutigkeit:

Annahme: Sei

$$f = \bigcup_{a \in A_f} a = \bigcup_{a \in A_f{}'} a$$

mit A_f, ${A_f}' \subseteq A(M)$, $A_f \neq {A_f}'$.
O. B. d. A. gebe es a_0 mit $a_0 \in A_f$ und $a_0 \notin {A_f}'$.
Multipliziert man beide Seiten der Gleichung

$$\bigcup_{a \in A_f} a = \bigcup_{a \in {A_f}'} a$$

mit a_0, so ergibt sich wegen Lemma 1.2:
$a_0 = 0$, d.h. a_0 wäre kein Atom. $\Rightarrow$ Widerspruch!
Also muß die Annahme $A_f \neq {A_f}'$ falsch sein.

$\square$

Korollar 1.1 *Ist $f = \bigcup_{a \in A_f} a$ mit $A_f \subseteq A(M)$, so ist*

$$A_f = \{a \in A(M) \mid f \cdot a = a\}.$$

Bezeichnung 6 *Ist M eine boolesche Algebra, $A(M)$ die Menge der Atome von M, $f \in M$, $A_f \subseteq A(M)$, so daß $f = \bigcup_{a \in A_f} a$, dann sei*

$$Atome_M(f) := A_f = \{a \in A(M) \mid f \cdot a = a\}.$$

$Atome_M(f)$ wird auch als die Menge der in f enthaltenen Atome bezeichnet.

Aus dem obigen Beweis läßt sich ein weiteres Korollar herausziehen:

Korollar 1.2

$$\bigcup_{a \in A(M)} a = 1.$$

1.2.4 Unteralgebren

Definition 1.29 (Unteralgebra)
Die boolesche Algebra $(M', \cup, \cdot, \overline{})$ heißt **Unteralgebra** *der booleschen Algebra $(M, \cup, \cdot, \overline{})$, wenn $M' \subseteq M$ und wenn die Operationen beider Algebren auf M' übereinstimmen.*
(Schreibweise: $(M', \cup, \cdot, \overline{}) \subseteq (M, \cup, \cdot, \overline{})$)

Daraus ergibt der folgende einfache Satz:

Satz 1.4
Ist $(M, \cup, \cdot, \overline{})$ eine boolesche Algebra, $M' \subseteq M$ und gilt $\forall a, b \in M'$

$$a \cup b \in M', \ a \cdot b \in M', \ \overline{a} \in M',$$

dann ist $(M', \cup, \cdot, \overline{})$ Unteralgebra von $(M, \cup, \cdot, \overline{})$.

(Da sich jede der beiden binären Verknüpfungen durch die andere binäre Verknüpfung und die Negation ausdrücken läßt, genügt es schon festzustellen, daß M' gegenüber Disjunktion und Negation oder gegenüber Konjunktion und Negation abgeschlossen ist.)

Mit Satz 1.1 erhält man folgendes Lemma:

Lemma 1.3 *Sei M eine boolesche Algebra, M' Unteralgebra von M. Für die Null– bzw. Einselemente von M und M' gilt:*

$$1_M = 1_{M'} \qquad \textit{und} \qquad 0_M = 0_{M'}$$

Beweis: Sei $a \in M'$ ($\Rightarrow a \in M$).
Dann gilt wegen Satz 1.1: $1_M = a \cup \overline{a} = 1_{M'}$ und $0_M = a \cap \overline{a} = 0_{M'}$. □

Um Unteralgebren M' einer booleschen Algebra M genauer zu charakterisieren, kann man die Atome von M' eindeutig als Disjunktion von Atomen von M darstellen (siehe Satz 1.3). Zieht man noch Lemma 1.2 in Betracht, so ergibt sich der folgende Satz:

Satz 1.5
Sei M eine boolesche Algebra, M' Unteralgebra von M. Seien $a_1, \ldots, a_k$ die Atome von M'.
Dann ist

$$\{Atome_M(a_1), \ldots, Atome_M(a_k)\}$$

eine Partition der Menge $A(M)$ der Atome von M.

Beweis:

- Für $i \in \{1, \ldots, k\}$ gilt:
  $Atome_M(a_i) \neq \emptyset$, da $a_i \neq 0$ (a_i Atom von M').
- Es gilt:
  $$Atome_M(a_i) \cap Atome_M(a_j) = \emptyset \text{ für } i \neq j.$$
  Beweis:
  Annahme: Sei $S = Atome_M(a_i) \cap Atome_M(a_j) \neq \emptyset$.
  Es gilt $a_i = \bigcup_{a \in Atome_M(a_i)} a$ und $a_j = \bigcup_{a \in Atome_M(a_j)} a$
  $$\Longrightarrow a_i \cdot a_j = \bigcup_{a \in S} a \neq 0$$
  Da a_i, a_j Atome von M', gilt aber nach Lemma 1.2:
  $a_i \cdot a_j = 0 \qquad \Longrightarrow$ Widerspruch

- Wegen Korollar 1.2 angewendet auf M' gilt:

$$1_{M'} = \bigcup_{1\leq i\leq k} a_i = \bigcup_{1\leq i\leq k} \Big(\bigcup_{a\in Atome_M(a_i)} a\Big) = \bigcup_{a\in\bigcup_{1\leq i\leq k} Atome_M(a_i)} a$$

  Andererseits gilt wegen Korollar 1.2 angewendet auf M:

$$1_M = \bigcup_{a\in A(M)} a$$

  Wegen $1_{M'} = 1_M$ und da die Darstellung als Disjunktion von Atomen eindeutig ist, gilt:

$$\bigcup_{1\leq i\leq k} Atome_M(a_i) = A(M)$$

□

Ist M eine boolesche Algebra und sind durch eine Partition P der Atome von M die Atome einer Unteralgebra M' von M vorgegeben, so erhält man sämtliche Elemente von M' als Disjunktionen dieser Atome.

M' hat dann wegen der Eindeutigkeit der Darstellung als Disjunktion von Atomen $2^{|P|}$ verschiedene Elemente.

1.2.5 Erzeugendensysteme

Definition 1.30 (Von E erzeugte Unteralgebra) *Sei M eine boolesche Algebra und $E \subseteq M$ eine Teilmenge der Elemente von M. Dann heißt die (bzgl. Mengeninklusion) kleinste Unteralgebra $< E >$ von M, die alle Elemente von E enthält, die* **von E erzeugte Unteralgebra** *von M.*

Erst das nun folgende Lemma liefert die Berechtigung für obige Definition, da er aussagt, daß es stets genau eine solche Unteralgebra gibt:

Lemma 1.4 *Sei M eine boolesche Algebra, $E \subseteq M$. Sei $U(M,E)$ die Menge der Unteralgebren von M, die E enthalten. Dann gilt:*

$$< E > := \bigcap_{M'\in U(M,E)} M'$$

ist eine boolesche Algebra, die E enthält.

Daß $< E >$ dann die kleinste aller Unteralgebren von M ist, die E enthalten, folgt nun leicht aus der Tatsache, daß für jede Unteralgebra M'' aus $U(M,E)$ gilt:

$$< E > := \bigcap_{M'\in U(M,E)} M' \subseteq M''.$$

Definition 1.31 (Erzeugendensystem) *$E \subseteq M$ heißt Erzeugendensystem der booleschen Algebra M, wenn $< E >= M$.*

Im allgemeinen gibt es zu einer booleschen Algebra eine Vielzahl von Erzeugendensystemen. *Ein* mögliches Erzeugendensystem ist durch die Menge der Atome gegeben.

Lemma 1.5 *Ist M eine boolesche Algebra und $|M| \geq 2$, dann gilt:*

$$M =< A(M) > .$$

Beweis:

- Da $< A(M) >$ Unteralgebra von M, gilt $< A(M) >\subseteq M$.
- Da jedes Element von M sich als Disjunktion von Atomen schreiben läßt, $A(M) \subseteq< A(M) >$ und $< A(M) >$ hinsichtlich $\cup$ abgeschlossen ist, gilt

$$M \subseteq< A(M) > .$$

$\square$

Eine mehr konstruktive Charakterisierung der von einer Menge E erzeugten Unteralgebra von M liefert der folgende Satz:

Satz 1.6 *Sei M eine boolesche Algebra, $E \subseteq M$. Die von E erzeugte Unteralgebra $< E >$ ist die Menge aller Elemente von M, die durch $\cup$, $\cdot$ und $\overline{}$ aus den Elementen von E erzeugt werden können.*

Da $< E >$ eine boolesche Algebra ist und $E \subseteq< E >$, sind alle Elemente von M, die durch $\cup$, $\cdot$ und $\overline{}$ aus Elementen von E erzeugt werden können, auch in $< E >$ enthalten.

Daß jedes Element aus $< E >$ durch $\cup$, $\cdot$ und $\overline{}$ aus Elementen von E erzeugt werden kann, läßt sich aus dem folgenden Lemma schließen:

Lemma 1.6 *Sei M eine boolesche Algebra, $E \subseteq M$, $E = \{x_1, \ldots, x_n\}$. Dann gilt für die Menge $A(< E >)$ der Atome von $< E >$:*

$$A(< E >) = \{x_1^{\epsilon_1} \cdot \ldots \cdot x_n^{\epsilon_n} \mid (\epsilon_1, \ldots, \epsilon_n) \in \{0,1\}^n,\ x_1^{\epsilon_1} \cdot \ldots \cdot x_n^{\epsilon_n} \neq 0\}.$$

Hierbei sei $x_i^1 = x_i$ und $x_i^0 = \overline{x_i}$.

(Beweis siehe [Hot74].)

1.2.6 Homomorphismen boolescher Algebren

Im Zusammenhang mit der Synthese boolescher Funktionen (siehe Kapitel 3) sind Homomorphismen, insbesondere Automorphismen, boolescher Algebren von Interesse. Homomorphismen boolescher Algebren sind wie folgt definiert:

Definition 1.32 (Homomorphismen boolescher Algebren)
Seien M und M' boolesche Algebren und sei h eine Abbildung von M in M'. h heißt **Homomorphismus** *von M in M', wenn $\forall a, b \in M$ gilt:*

1. $h(a \cup b) = h(a) \cup h(b)$
2. $h(a \cap b) = h(a) \cap h(b)$
3. $h(\overline{a}) = \overline{h(a)}$

Ist h bijektiv, so heißt h **Isomorphismus**. *Ein Isomorphismus von M in M heißt* **Automorphismus**.

Anmerkung: Forderungen $1-3$ sind nicht unabhängig. Mit Hilfe der de Morgan–Formeln kann man aus Forderung 1 und 3 Forderung 2 herleiten und aus Forderung 2 und 3 Forderung 1.

Folgerung:

- $h(1) = h(a \cup \overline{a}) = h(a) \cup \overline{h(a)} = 1$
- $h(0) = 0$ (analog)

Aus der Eindeutigkeit der Darstellung eines Elementes einer booleschen Algebra als Disjunktion von Atomen ergibt sich das folgende Lemma:

Lemma 1.7 *Ein Homomorphismus $h : M \to M'$ ist eindeutig bestimmt durch die Angabe der Bilder der Atome von M, d.h. durch Angabe von $h|_{A(M)}$.*

Ein wichtiges Beispiel für einen Isomorphismus ist die Isomorphie zwischen der booleschen Algebra $S(D)$ und der Potenzmenge von D.

Satz 1.7 *Die Abbildung*
$$ON : S(D) \to 2^D,$$
die jeder booleschen Funktion $f \in S(D)$ ihre ON–Menge $ON(f)$ zuordnet, ist ein Isomorphismus zwischen $(S(D), \cup, \cdot, \overline{\ })$ und $(2^D, \cup, \cap, \overline{\ })$, wobei im zweiten Fall $\cup$, $\cap$ und $\overline{\ }$ die mengentheoretischen Operationen Vereinigung, Durchschnitt und Komplement bzgl. D sind.

Beweis:
Seien $f, g \in S(D)$. Sei $x \in D$ beliebig.

- $(f \cup g)(x) = 1 \quad \Leftrightarrow \quad f(x) = 1$ oder $g(x) = 1$
  $\Longrightarrow \qquad ON(f \cup g) = ON(f) \cup ON(g)$
- $\overline{f}(x) = 1 \quad \Leftrightarrow \quad f(x) \neq 1$
  $\Longrightarrow \qquad ON(\overline{f}) = D \setminus ON(f) = \overline{ON(f)}$
- Es ist klar, daß ON surjektiv und injektiv ist.

Bemerkung 6 *Satz 1.7 ist ein Spezialfall des Satzes von Stone für endliche boolesche Algebren, der aussagt, daß jede endliche boolesche Algebra M mit der Menge A von Atomen isomorph ist zu einer Algebra $(2^A, \cup, \cdot, \overline{\ })$, wobei $\cup$, $\cdot$ und $\overline{\ }$ die mengentheoretischen Operationen sind.* [2]

Im folgenden ist es häufig nützlich, eine boolesche Funktion aus $S(D)$ als Teilmenge von D, nämlich als ihre ON–Menge, aufzufassen.

Ausgehend von einer Permutation auf D läßt sich leicht ein Automorphismus auf $S(D)$ definieren:

Lemma 1.8 *Ist $\sigma \in PerD$[‡] eine Permutation auf $D \subseteq \{0,1\}^n$, so ist*

$$h_\sigma : S(D) \to S(D),\ h_\sigma(f) = g \ \text{mit}\ g(\sigma(x)) = f(x)\ \forall x \in D$$

ein Automorphismus auf $S(D)$.

Beweis:
Zum Beweis des Satzes wird von der Isomorphie zwischen $S(D)$ und der zugehörigen ON–Mengen–Algebra Gebrauch gemacht.

Es gilt nach Definition von h_σ:

$$ON(h_\sigma(f)) = \{\sigma(x) \mid x \in ON(f)\} = \sigma(ON(f))$$

$\Longrightarrow$

- $$\begin{aligned} ON(h_\sigma(f \cup g)) &= \sigma(ON(f \cup g)) \\ &= \sigma(ON(f) \cup ON(g)) \\ &\overset{(*)}{=} \sigma(ON(f)) \cup \sigma(ON(g)) \\ &= ON(h_\sigma(f)) \cup ON(h_\sigma(g)) \\ &= ON(h_\sigma(f) \cup h_\sigma(g)) \end{aligned}$$

  Gleichung (*) gilt, da σ eine Bijektion auf D ist. Es macht keinen Unterschied, ob man 2 Mengen zuerst vereinigt und dann die Elemente mit σ (bijektiv) „umbenennt“ oder zuerst die Elemente „umbenennt“ und dann die beiden Mengen vereinigt.

---

[2] Die Atome von $S(D)$ sind die Funktionen X^ϵ, die genau an einer Stelle $\epsilon \in D$ den Wert 1 annehmen. In Satz 1.7 sind im Bild des Isomorphismus die Funktionen X^ϵ durch ihre einzige 1–Stelle ϵ ersetzt.

[‡] Unter $PerD$ ist die Gruppe aller Permutationen auf D zu verstehen.

- Analog erhält man
$$ON(h_\sigma(\overline{f})) = \overline{ON(h_\sigma(f))} = ON(\overline{h_\sigma(f)})$$
- Da σ eine Bijektion auf D ist, ist σ angewendet auf Teilmengen von D (d.h. auf ON–Mengen) auch eine Bijektion auf 2^D. Aufgrund der Isomorphie zwischen Elementen aus $S(D)$ und ihren ON–Mengen ist auch h_σ auf $S(D)$ eine Bijektion.

□

Bezeichnung 7 *In diesem Zusammenhang wird h_σ als der durch σ induzierte Automorphismus auf $S(D)$ bezeichnet.*

Satz 1.8 *Ein Homomorphismus $h : M \to M$ ist genau dann ein Automorphismus, wenn die Abbildung $h' : A(M) \to A(M)$ mit $h'(a) = h(a)\ \forall a \in A(M)$ wohldefiniert ist und eine Permutation auf den Atomen von M ist.*

Beweis:

„$\Longleftarrow$“: Sei h ein Automorphismus, d.h. sei h bijektiv.

- Zur Wohldefiniertheit:
  Das Bild eines Atoms unter h ist ein Atom, denn:
  Sei $a \in M$ ein beliebiges Atom. Dann gilt $\forall b \in M$: $ab = 0$ oder $ab = a$.
  Da h surjektiv, gilt $\forall c \in M\ c = h(b_c)$ für ein $b_c \in M$.
  $h(a)$ ist ein Atom, da
  - $\forall c \in M$ gilt:
    $h(a) \cdot c = h(a) \cdot h(b_c) = h(ab_c) = h(a)$ oder
    $h(a) \cdot c = h(a) \cdot h(b_c) = h(ab_c) = h(0) = 0$
  - $h(a) \neq 0$, da sonst $h(a) = h(0)$ mit $a \neq 0$ und h wäre nicht injektiv.
- Zur Injektivität:
  Die Injektivität von h' folgt aus der Injektivität von h.

- Zur Surjektivität:
  Da h bijektiv ist, muß es zu jedem Atom a ein Urbild unter h geben. Es ist lediglich noch zu zeigen, daß das Urbild eines Atoms a wiederum ein Atom ist:
  Angenommen es gäbe $b \in M \setminus A(M)$ mit $h(b) = a$, dann wäre $b = \bigcup_{1 \leq i \leq n} a_i$ mit $n \geq 2$.
  $\Longrightarrow$ $h(b) = \bigcup_{1 \leq i \leq n} h(a_i)$, wobei wegen der Injektivität $h(a_i) \neq h(a_j)$ für $i \neq j$.
  $\Longrightarrow$ Widerspruch zu $h(b) = a$, da die Darstellung von $h(b)$ als Disjunktion von Atomen eindeutig ist.

$\Longrightarrow h'$ ist wohldefiniert und eine Permutation auf $A(M)$.

„$\Longrightarrow$“: Sei $h : M \to M$ ein Homomorphismus, h' sei eine Permutation auf $A(M)$ mit $h'(a) = h(a)\ \forall a \in A(M)$.
Die Elemente von M werden als Vereinigung von Atomen dargestellt. Aus der Bijektivität von h' und der Eindeutigkeit der Darstellung als Disjunktion von Atomen folgt die Bijektivität von h.

□

Angewendet auf die boolesche Algebra $S(D)$ liefert Satz 1.8 die nun folgende Aussage. Dabei wird die umkehrbar eindeutige Entsprechung von Atomen von $S(D)$ zu Elementen von D (nämlich gerade zu den Elementen, auf denen die Atome 1 liefern) ausgenutzt.

Satz 1.9 *Zu jedem Automorphismus auf $S(D)$ gibt es eine Permutation $\sigma \in PerD$, die diesen Automorphismus induziert.*

Beweis:
Sei $X^\epsilon \in S(D)$ $(\epsilon \in D)$ definiert durch

$$X^\epsilon(\alpha) = \begin{cases} 1 & \text{falls } \alpha = \epsilon \\ 0 & \text{falls } \alpha \in D \setminus \{\epsilon\} \end{cases}$$

Die Menge der Atome von $S(D)$ ist

$$A(S(D)) = \{X^\epsilon \mid \epsilon \in D\}.$$

Sei h ein Automorphismus auf $S(D)$. Dann gilt nach Satz 1.8

$$h(X^\epsilon) = X^{\sigma(\epsilon)}\ \forall \epsilon \in D,$$

wobei σ eine Permutation auf D ist.
Der von σ induzierte Automorphismus h_σ ist nun gleich h. Es genügt nach Lemma 1.7, die Gleichheit auf den Atomen nachzuweisen.
Es gilt $\forall \alpha \in D$:

$$\begin{aligned} h_\sigma(X^\epsilon)(\alpha) = X^\epsilon(\sigma^{-1}(\alpha)) &= X^{\sigma(\epsilon)}(\alpha) \\ \Longrightarrow \quad h_\sigma(X^\epsilon) &= h(X^\epsilon) \\ \Longrightarrow \quad h_\sigma &= h \end{aligned}$$

□

Ist ein Automorphismus auf $S(D)$ gegeben, so handelt es sich also immer um eine Transformation der entsprechenden ON−Mengen durch eine Permutation auf D.

1.2.7 Automorphismengruppen boolescher Algebren

Hat man bei der Logiksynthese eine Gruppe von Automorphismen gegeben, die die zu realisierenden Funktionen fest lassen, so kann man von diesem Wissen mit Vorteil Gebrauch machen (siehe Kapitel 3). In diesem Abschnitt sollen Eigenschaften von Automorphismengruppen näher behandelt werden.

Sei M eine boolesche Algebra.

$$G(M) := \{h : M \to M \mid h \text{ Automorphismus}\}$$

ist bzgl. Hintereinanderausführung von Abbildungen eine Gruppe.

Die Permutationen auf der Menge $A(M)$ der Atome von M bilden mit der Hintereinanderausführung von Abbildungen ebenfalls eine Gruppe.

Man kann nun Satz 1.8 etwas genauer fassen:

Satz 1.10 *Die Gruppen $(G(M), \circ)$ und $(PerA(M), \circ)$ sind isomorph.*

$$\beta : G(M) \to PerA(M),\ \beta(h) = h' \text{ mit } h'(a) = h(a)\ \forall a \in A(M)$$

ist ein Gruppenisomorphismus.

Beweis:

- Nach Satz 1.8 ist β wohldefiniert.
- Es ist klar, daß die Einschränkung von $h \in G(M)$ auf $A(M)$ ein Gruppenhomomorphismus ist.
- Da sich jede Permutation auf den Atomen von M in natürlicher Weise zu einem Homomorphismus auf M fortsetzen läßt, ergibt sich in Verbindung mit Satz 1.8 die Surjektivität von β.
- Da nach Lemma 1.7 ein Automorphismus h schon durch $h|_{A(M)}$ eindeutig bestimmt ist, ist β auch injektiv.

□

Angewendet auf $S(D)$ ergibt sich also eine Isomorphie zwischen $G(S(D))$ und $PerD$.

Bezeichnung 8 (G(M, E))
Sei M eine boolesche Algebra, $E \subseteq M$.

$$G(M, E) := \{h \in G(M) \mid h(e) = e\ \forall e \in E\}$$

$G(M, E)$ ist die Untergruppe aller Automorphismen, die die Elemente von E festlassen.

Lemma 1.9 *$G(M, E)$ ist eine Untergruppe von $G(M)$.*

Beweis:
Z. z.: $G(M, E)$ ist abgeschlossen gegenüber den Gruppenoperationen.
Seien $h_1, h_2 \in G(M, E)$.

$$\Longrightarrow \quad \forall e \in E \text{ gilt } (h_1 \circ h_2)(e) = h_1(h_2(e)) = h_1(e) = e$$
$$\Longrightarrow \quad h_1 \circ h_2 \in G(M, E)$$
$$\forall e \in E \text{ gilt } h_1^{-1}(e) = h_1^{-1}(h_1(e)) = e$$
$$\Longrightarrow \quad h_1^{-1} \in G(M, E)$$

□

Das nächste Lemma zeigt, daß man sich bei der Betrachtung von Teilmengen von M, die unter einer Untergruppe der Automorphismen auf M fest bleiben, auf Unteralgebren von M beschränken kann.

Lemma 1.10 *Sei $E \subseteq M$ und $M' =< E >$ die durch E erzeugte Unteralgebra. Dann gilt:*
$$G(M, E) = G(M, M')$$

Beweis:
Sei $h \in G(M, E)$.
Z. z.: h läßt alle Elemente fest, die durch $\cup$, $\cdot$ und $^-$ aus Elementen von E erzeugt werden können.
Seien e_1, $e_2 \in E$. Dann gilt:

$$h(e_1 \cup e_2) = h(e_1) \cup h(e_2) = e_1 \cup e_2$$
$$h(e_1 \cdot e_2) = h(e_1) \cdot h(e_2) = e_1 \cdot e_2$$
$$h(\overline{e_1}) = \overline{h(e_1)} = \overline{e_1}$$

Der Beweis, daß für alle Elemente $e \in< E >$ $h(e) = e$ gilt, erfolgt durch Induktion über die Anzahl der Operationen, die benötigt werden, um e aus Elementen von E zu erzeugen.

$$\Longrightarrow \quad h \in G(M, < E >) \qquad \Longrightarrow \quad G(M, E) \subseteq G(M, < E >)$$

Da $E \subseteq< E >$, ist $G(M, < E >) \subseteq G(M, E)$ trivial.

□

Die Tatsache, daß die Atome einer Unteralgebra M' von M gemäß Satz 1.5 eine Partition auf den Atomen von M liefern, führt zu einer näheren Charakterisierung der Untergruppe der Automorphismen, die die Elemente von M' festlassen.

Die Atome von M' müssen unter jedem Automorphismus h aus $G(M, M')$ festbleiben. (Diese Forderung genügt aber auch schon, da sämtliche Elemente von M' unter h festbleiben, wenn die Atome unter h festbleiben.)

Ist a ein Atom von M', so darf h die Atome aus $Atome_M(a)$ permutieren, da eine Permutation innerhalb von $Atome_M(a)$ a selbst festläßt.

h darf aber *nicht* ein Atom aus $Atome_M(a)$ auf ein Atom $\notin Atome_M(a)$ abbilden, da sonst $h(a) \neq a$. (Beachte dazu immer die Eindeutigkeit der Darstellung als Disjunktion von Atomen.)

Aus diesen Überlegungen resultiert eine Verfeinerung von Satz 1.10:

Satz 1.11 *Sei M' Unteralgebra der booleschen Algebra M. Sei die Menge der Atome von M' $A(M') = \{a_1, \ldots, a_k\}$.*
Dann ist $G(M, M')$ isomorph zum direkten Produkt der Permutationsgruppen auf $Atome_M(a_i)$, d.h.

$$G(M, M') \cong PerAtome_M(a_1) \times \ldots \times PerAtome_M(a_k)$$

$$\gamma : G(M, M') \to PerAtome_M(a_1) \times \ldots \times PerAtome_M(a_k),$$
$$\gamma(h) = h' \text{ mit } h'(a) = h(a) \; \forall a \in A(M)$$

ist ein Gruppenisomorphismus.

Gibt man eine Unteralgebra M' von M vor, so liefert $G(M, M')$ die Untergruppe aller Automorphismen, die M' festlassen.

Ist umgekehrt eine Teilmenge G' der Automorphismengruppe $G(M)$ gegeben, so interessiert man sich für die Teilmenge M' von M, die unter allen Automorphismen aus G' festbleibt.

Bezeichnung 9 *Sei M eine boolesche Algebra, $G' \subseteq G(M)$. Mit*

$$M(G') := \{f \in M \mid h(f) = f \; \forall h \in G'\}$$

wird die Teilmenge von M bezeichnet, deren Elemente unter allen Automorphismen aus G' festbleiben.

Lemma 1.11 *$M(G')$ ist eine Unteralgebra von M.*

Beweis:
Seien $f, g \in M(G')$, d.h. $h(f) = f$, $h(g) = g$ $\forall h \in G'$.

$$\Longrightarrow \quad h(f \cup g) = h(f) \cup h(g) = f \cup g$$
$$h(\overline{f}) = \overline{h(f)} = \overline{f}$$

□

Lemma 1.12 *Es gilt für jede Unteralgebra M' von M: $M(G(M,M')) = M'$.*

Beweis:

- $M' \subseteq M(G(M,M'))$:
  Sei $f \in M'$. $\forall \sigma \in G(M,M')$ gilt: $\sigma(f) = f$.
  $\Longrightarrow \quad f \in M(G(M,M'))$
- $M(G(M,M')) \subseteq M'$:
  Sei $f \notin M'$. ($\Rightarrow \ f \neq 0$)
  Da $f \neq 0$, muß es Atome $a_1, \ldots, a_k \in A(M')$ geben ($k > 0$) mit $fa_i \neq 0$ $\forall 1 \leq i \leq k$ (wegen $\bigcup_{a \in A(M')} a = 1$).

  Wäre für alle Atome a_i mit $1 \leq i \leq k$ $fa_i = a_i$, so wäre $f = \bigcup_{1 \leq i \leq k} a_i$ und somit $f \in M'$.

  Somit gibt es $a_p \in A(M')$ mit $0 \neq fa_p \neq a_p$.
  Es gilt: $f = \bigcup_{a \in Atome_M(f)} a$ und $a_p = \bigcup_{a \in Atome_M(a_p)} a$.
  Es gilt also:

  $$\emptyset \neq Atome_M(f) \cap Atome_M(a_p) \neq Atome_M(a_p)$$

  Es gibt also m_1, m_2 mit

  $$m_1 \in Atome_M(f) \cap Atome_M(a_p) \text{ und}$$

  $$m_2 \in Atome_M(a_p) \setminus Atome_M(f)$$

  Nun kann man einen Automorphismus definieren, der M' festläßt, aber f nicht festläßt:
  Sei $p \in PerA(M)$ mit

  $$p(a) = \begin{cases} m_2, \text{falls } a = m_1 \\ m_1, \text{falls } a = m_2 \\ a, \text{sonst} \end{cases}$$

  Sei $h = \beta^{-1}(p)$ der zu p gehörige Automorphismus.
  Dann ist $h \in G(M,M')$, da h M' festläßt, aber

  $$h(f) = \bigcup_{a \in Atome_M(f)} h(a) = \bigcup_{a \in Atome_M(f)} p(a) \neq f.$$

  $\Longrightarrow \quad f \notin M(G(M,M'))$, da es in $G(M,M')$ einen Automorphismus h gibt, der f nicht festläßt.

□

Lemma 1.13 *1.* $M_1 \subseteq M_2 \quad \Longrightarrow \quad G(M,M_2) \subseteq G(M,M_1)$

2. $G_1 \subseteq G_2 \quad \Longrightarrow \quad M(G_2) \subseteq M(G_1)$

Beweis:

zu 1.: Sei $g \in G(M, M_2)$, dann gilt $g(a) = a\ \forall a \in M_2$ und damit auch $\forall a \in M_1 \subseteq M_2$.
$\Longrightarrow \quad g \in G(M, M_1)$

zu 2.: Sei $a \in M(G_2)$, dann gilt $\forall g \in G_2\ g(a) = a$ und damit $\forall g' \in G_1 \subseteq G_2$ $g'(a) = a$.
$\Longrightarrow \quad a \in M(G_1)$ $\square$

Kapitel 2

Motivation für mehrstufige Logiksynthese

2.1 Kosten zweistufiger Realisierungen

Häufig werden zur Realisierung boolescher Funktionen zweistufige Lösungen (Polynome) gewählt. Zum einen kann man boolesche Polynome leicht durch programmierbare logische Felder (PLA's) realisieren, zum anderen gibt es relativ gute heuristische Verfahren (z. B. ESPRESSO II [BHMSV84]), die in der Lage sind, zu einer booleschen Funktion Minimalpolynome bzw. Polynome, deren Kosten die eines Minimalpolynoms nicht wesentlich übersteigen, zu berechnen.

Die Frage, warum man sich nicht auf zweistufige Realisierungen beschränkt, sondern auch *mehrstufige* Logiksynthese durchführt, ist leicht zu beantworten:
Häufig sind Minimalpolynomrealisierungen einer Schaltfunktion *wesentlich* teurer als andere Realisierungen. Es ist i. a. nicht günstig, sich bei der Suche nach guten Realisierungen für eine boolesche Funktion auf Minimalpolynome zu beschränken.

Als Extrembeispiel für eine solche Funktion soll hier die Exklusiv–Oder–Funktion mit n Eingängen dienen:

$$exor_n(x_1,\ldots,x_n) = \begin{cases} 0, & \text{falls } (\sum_{i=1}^n x_i) \bmod 2 = 0 \\ 1, & \text{sonst} \end{cases}$$

Man überlegt sich leicht, daß

$$p = \bigvee_{(\sum_{i=1}^n \epsilon_i) \mathrm{mod} 2 = 1} x_1^{\epsilon_1} \cdot \ldots \cdot x_n^{\epsilon_n}$$

die kleinste zweistufige Realisierung von $exor_n$ ist. Die Kosten betragen

$$C(p) = \underbrace{(n-1)}_{\text{Kosten für 1 Monom}} \cdot 2^{n-1} + \underbrace{(2^{n-1}-1)}_{\text{Oder-Gatter}} = n2^{n-1} - 1.$$

Daß es sich bei der $exor_n$–Funktion sogar um eine *Funktion mit dem teuersten Minimalpolynom* handelt, schließt man aus dem folgenden Lemma und der Tatsache, daß die Monome in p vollständig sind.

Lemma 2.1 *Zu jeder booleschen Funktion $f \in B_n$ gibt es eine Polynomrealisierung mit $\leq 2^{n-1}$ Monomen.*

Beweis:
Beweis durch Induktion über n.

$n = 1:$ Es gilt, daß $f = 0$, $f = x$, $f = \overline{x}$ oder $f = 1$.
Es ist klar, daß es eine Polynomrealisierung mit ≤ 1 Monom gibt.

$n \to n+1:$ Sei $f \in B_{n+1}$.

$$f(x_1, \ldots, x_{n+1}) = \overline{x_{n+1}} \cdot f(x_1, \ldots, x_n, 0) + x_{n+1} \cdot f(x_1, \ldots, x_n, 1)$$

Die Funktionen f_0 und f_1 mit

$$\begin{aligned} f_0(x_1, \ldots, x_n) &:= f(x_1, \ldots, x_n, 0), \\ f_1(x_1, \ldots, x_n) &:= f(x_1, \ldots, x_n, 1) \quad \forall x \in \{0,1\}^n \end{aligned}$$

sind aus B_n und haben nach Induktionsvoraussetzung Polynomdarstellungen p_0 und p_1 mit jeweils $\leq 2^{n-1}$ Monomen. Aus

$$p = \overline{x_{n+1}} \cdot p_0 + x_{n+1} \cdot p_1$$

erhält man durch Anwendung des Distributivgesetzes ein Polynom mit $\leq 2 \cdot (2^{n-1}) = 2^n$ Monomen, das f realisiert.

□

Obwohl die Funktion $exor_n$ hinsichtlich Polynomrealisierung also eine 'schwerste' Funktion ist, kann sie mehrstufig sehr leicht realisiert werden: Wie in Bild 2.1 dargestellt, genügt ein (balancierter) Baum aus $n-1$ $exor_2$–Gattern als Realisierung $EXOR_{B_2}$ ($exor_2$ ist assoziativ!).

Eine Realisierung $EXOR_{R_2}$ durch einen R_2–Schaltkreis erhält man, wenn man in Bild 2.1 die $exor_2$–Gatter durch die Schaltung von Bild 2.2 ersetzt.

Die Kosten von $EXOR_{B_2}$ betragen

$$C_{B_2}(EXOR_{B_2}) = n - 1,$$

die Kosten von $EXOR_{R_2}$ betragen

$$C_{R_2}(EXOR_{R_2}) = 3(n-1),$$

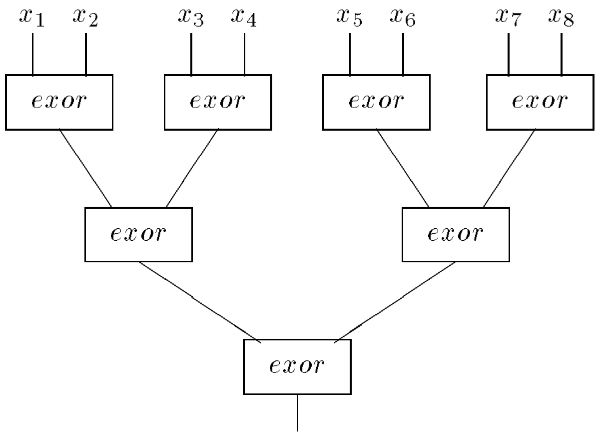


Abbildung 2.1: Realisierung $EXOR_{B_2}$ (Beispiel für $n = 8$)

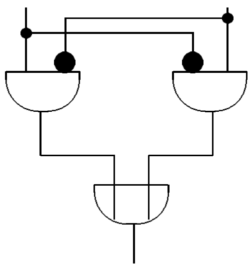


Abbildung 2.2: Schaltung für $exor_2$

d.h. die Kosten des Minimalpolynoms sind exponentiell in den Kosten der angegebenen mehrstufigen Realisierungen.

Anmerkung: Es gilt sogar

$$C_{B_2}(exor_n) = n - 1 = C_{B_2}(EXOR_{B_2}),$$

d.h. die angegebene Realisierung ist B_2–optimal. Dies folgt aus Lemma 2.2 (für Funktionen, die von allen Eingängen abhängen, siehe auch [Weg87], S. 120, Theorem 1.1) und der Tatsache, daß $exor_n$ von allen n Eingängen abhängt.

Lemma 2.2 *Ist eine Funktion $f \in B_n$ von mindestens m Variablen abhängig, so ist $C_{B_2}(f) \geq m - 1$.*

Beweis:
Sei $S = ((V, E), typ, pe, pa)$ ein optimaler B_2–Schaltkreis für f.

- – Da f von mindestens m Eingängen abhängt, gilt für mindestens m Eingangsknoten ($EPAD$–Knoten) e:

  $$outdeg(e) \geq 1.$$

  – Da S optimal ist, gilt f. a. Knoten v mit $typ(v) \in B_2$:

  $$outdeg(v) \geq 1.$$

  (Denn sonst könnte der Knoten eliminiert werden, ohne die berechnete Funktion zu verändern.)

  $$\Rightarrow \qquad |E| \geq m + C_{B_2}(f) \qquad (1)$$

- – F. a. Knoten v mit $typ(v) \in B_2$ gilt: $indeg(v) = 2$

  – Für Ausgangsknoten ($APAD$–Knoten) a gilt: $indeg(a) = 1$

  – F. a. Eingangsknoten ($EPAD$–Knoten) e gilt: $indeg(e) = 0$

  $$\Rightarrow \qquad |E| = 2C_{B_2}(f) + 1 \qquad (2)$$

$$\begin{array}{rcc} (1), (2) & \Longrightarrow & 2C_{B_2}(f) + 1 \geq m + C_{B_2}(f) \\ & \Longleftrightarrow & C_{B_2}(f) \geq m - 1 \end{array}$$

□

Aus Lemma 2.2 folgt darüber hinaus, daß $exor_n$ unter allen Funktionen aus B_n, die von allen Variablen abhängen, minimale Komplexität hat. Insofern ist $exor_n$ bzgl. mehrstufiger Realisierung eine 'leichteste' Funktion, obwohl sie bzgl. zweistufiger Realisierung eine 'schwerste' Funktion ist!

2.2 Der Shannon–Effekt: Ein nur *scheinbar* zufriedenstellendes Resultat

Der nun folgende Abschnitt dient im wesentlichen zwei Zielen: Zum einen soll die Komplexität boolescher Funktionen grundsätzlich untersucht werden, zum anderen soll geklärt werden, was man von Algorithmen zur Logiksynthese erwarten kann und was nicht.

Schon 1949 gelang es Shannon, mit einfachen Abzählargumenten folgenden Satz zu zeigen ([Sha49]):

Satz 2.1 (Shannon '49) *Für hinreichend große n haben „fast alle“[1] Funktionen $f \in B_n$ eine Komplexität von*

$$C_{B_2}(f) \geq \frac{2^n}{n}$$

Auf der anderen Seite gibt es einen Satz von Lupanov ([Lup58]), der folgendes besagt:

Satz 2.2 (Lupanov '58) *Zu jeder Funktion $f \in B_n$ gibt es eine Realisierung L_f (nämlich die Lupanov'sche k–s–Darstellung) mit*

$$C_{B_2}(L_f) \leq \frac{2^n}{n} + o(\frac{2^n}{n}).$$

Aus diesen beiden Sätzen kann man folgern, daß „fast alle“ Funktionen $f \in B_n$ fast die gleiche Komplexität wie die schwerste Funktion in B_n haben. Diesen Effekt nennt man nach Lupanov (70) den **Shannon–Effekt**.

An dieser Stelle könnte es auf den ersten Blick so aussehen, als ob sich eine weitere Behandlung des Themas erübrigen würde: Man kann zu fast allen Funktionen aus B_n mit der Lupanov'schen k–s–Darstellung eine nahezu optimale Realisierung angeben.

Daß man an dieser Stelle nicht aufhört, sondern nach anderen Syntheseverfahren sucht, hat folgende Gründe:

- Bei den eben vorgestellten Aussagen handelt es sich um *asymptotische* Resultate. Für kleine n kann die Komplexität einer Lupanov'schen k–s–Darstellung $\frac{2^n}{n}$ deutlich übersteigen.
- Bei den in der Praxis auftretenden booleschen Funktionen handelt es sich meist *nicht* um zufällig gewürfelte Funktionen, sondern um Funktionen, die von einem menschlichen Benutzer spezifiziert worden sind (z.B. aus

[1] Die Formulierung „fast alle Funktionen $f \in B_n$ haben Eigenschaft P“ ist gleichbedeutend mit der Aussage

$$|\{f \in B_n \mid f \text{ hat Eigenschaft } P \text{ } nicht\}| \;/\; |B_n| \to 0 \text{ für } n \to \infty.$$

einem bestimmten algorithmischen Problem hervorgegangen sind). Daher weisen sie häufig gewisse *Regelmäßigkeiten* bzw. *Struktureigenschaften* auf. Sie sind fast immer mit weit weniger als $\frac{2^n}{n}$ Gattern zu realisieren.

Aufgabe von Logiksyntheseverfahren ist es nun, solche Struktureigenschaften zu erkennen und sie für die Realisierung auszunutzen.

Eine mögliche Struktureigenschaft einer booleschen Funktion ist die *nichttriviale Zerlegbarkeit*. Ein Syntheseverfahren, das auf der Ausnutzung nichttrivialer Zerlegungen beruht, wird in Kapitel 4 vorgestellt.

Eine andere Möglichkeit ist die Ausnutzung von Symmetrien bzw. bestimmten Invarianzen bei der Synthese. Dies wird in Kapitel 3 näher erläutert.

Zunächst soll noch untersucht werden, ob ähnliche Resultate auch für Funktionen mit mehreren Ausgängen (Funktionen aus $B_{n,m}$) gelten.

Tatsächlich kann man nachweisen, daß „fast alle" Funktionen mit n Eingängen und m Ausgängen eine Komplexität $> m\frac{2^n}{n}$ haben. Allerdings gilt dies nur, wenn m nicht zu groß wird.

Mit Abzählargumenten kann man folgenden Satz zeigen:

Satz 2.3 *Für genügend große n haben mindestens*

$$|B_{n,m}|\,(1 - 2^{-s(n)})$$

der $|B_{n,m}| = 2^{m\cdot 2^n}$ Funktionen aus $B_{n,m}$ eine Komplexität $> m\frac{2^n}{n}$, wobei $s(n) = \frac{2^n}{n^2}$[†]*.*

Hierbei darf m allerdings nicht zu groß werden. Der Satz gilt für $m = o(n)$ oder für $m \leq c \cdot n$ mit $c \leq \frac{1}{(1+\delta)64e}$, $\delta > 0$.[‡]

Der Beweis des Satzes erfolgt mit ähnlichen Argumenten wie der Beweis des Satzes von Shannon. Dazu wird zunächst ein Lemma bewiesen, das die Mindestanzahl der B_2–Schaltkreise einer vorgegebenen Komplexität abschätzt.

Lemma 2.3 *Höchstens*

$$S(b,n,m) = \frac{1}{\sqrt{2\pi}}(b+n-1)^{2b}16^b b^{m-b-1/2}e^b$$

Funktionen $f \in B_{n,m}$ können durch B_2–Schaltkreise der B_2–Komplexität b berechnet werden.

[†] Man beachte, daß $2^{-s(n)}$ eine mit wachsendem n sehr schnell gegen 0 strebende Funktion ist. Die Formulierung „fast alle Funktionen haben eine Komplexität $> m\frac{2^n}{n}$" ist also berechtigt.

[‡] Die Aussage des Satzes stimmt auch mit der Anschauung überein: Ist m nicht zu groß und wählt man für großes n m beliebige Funktionen $f_1, \ldots, f_m \in B_n$, so ist es unwahrscheinlich, daß man größere Teile aus der Realisierung einer Funktion f_i mit Vorteil bei der Realisierung von f_j ($j \neq i$) verwenden kann. Die zufällig gewählten Funktionen sind „so verschieden", daß eine getrennte Realisierung der m Funktionen kaum teurer ist als eine gemeinsame Realisierung als Funktion mit m Ausgängen.

Beweis:
Es erfolgt eine Abschätzung der Anzahl verschiedener B_2–Schaltkreise mit n Eingängen, m Ausgängen und b sonstigen Zellen:

- Für jede Zelle, die weder $EPAD$– noch $APAD$–Zelle ist, gibt es $|B_2| = 16$ mögliche Typen.
- Für jeden der beiden Vorgänger eines Gatters gibt es $(b + n - 1)$ Möglichkeiten ($b - 1$ andere Gatter und n Eingänge).
  (Dabei wird allerdings *nicht* berücksichtigt, daß Schaltkreise zyklenfrei sein müssen. Es wird lediglich eine obere Schranke für die tatsächliche Anzahl der verschiedenen Schaltkreise der Größe b bestimmt.)
- Bisher wurden $(b + n - 1)^{2b} \cdot 16^b$ verschiedene Schaltkreise gezählt. Jeder Schaltkreis wurde jedoch $(b!)$–mal gezählt, da $b!$ verschiedene Numerierungen der Gatter möglich sind.
- Bei jedem Schaltkreis muß nun noch festgelegt werden, an welchen der b Gatter die m Ausgangszellen angeschlossen werden sollen. Pro (bisher gezählten) Schaltkreis gibt es dazu b^m Möglichkeiten.

Insgesamt wird so eine obere Schranke von $(b + n - 1)^{2b} 16^b b^m / b!$ für die Anzahl der verschiedenen Schaltkreise der Größe b bestimmt.

Eine weitere Abschätzung mit Hilfe der Stirling–Formel liefert das Ergebnis:
Gemäß Stirling–Formel gilt: $b! \geq \sqrt{2\pi} \cdot \frac{b^{b+1/2}}{e^b}$
Folglich gilt

$$(b + n - 1)^{2b} 16^b b^m / b! \leq \frac{1}{\sqrt{2\pi}} (b + n - 1)^{2b} 16^b b^{m-b-1/2} e^b$$

und die Aussage des Lemmas ist bewiesen. □

Zum Beweis des Satzes betrachtet man eine Teilmenge $B^*_{n,m}$ aller Funktionen aus $B_{n,m}$, die gerade aus den $|B^*_{n,m}|$ Funktionen $\in B_{n,m}$ mit geringster Komplexität besteht. Obwohl $|B^*_{n,m}|$ für große n im Verhältnis zu $|B_{n,m}|$ verschwindend klein ist, kann unter Ausnutzung des Lemmas gezeigt werden, daß die Anzahl verschiedener Schaltkreise mit Komplexität $m \frac{2^n}{n}$ kleiner als $|B^*_{n,m}|$ ist. Da *verschiedene* Funktionen aus $B^*_{n,m}$ trivialerweise nur durch *verschiedene* Schaltkreise realisiert werden können, muß es muß also schon in $B^*_{n,m}$ Funktionen mit Komplexität $> m \frac{2^n}{n}$ geben. Alle *anderen* Funktionen (d.h. „fast alle“ Funktionen) aus $B_{n,m}$ müssen erst recht eine Komplexität $> m \frac{2^n}{n}$ haben, da $B^*_{n,m}$ ja gerade die Funktionen mit geringster Komplexität enthält.

Es folgt der ausführlichere Beweis des Satzes:

Beweis:
Sei $B^*_{n,m}$ eine Teilklasse von $B_{n,m}$ mit

$$|B^*_{n,m}| = 2^{m 2^n - 2^n / n^2}$$

Funktionen.

Sei

$$b = b(n,m) = max\{C_{B_2}(f) \mid f \in B^*_{n,m}\}$$

$\Rightarrow$ Alle Funktionen aus $B^*_{n,m}$ können durch Schaltkreise mit b B_2–Zellen realisiert werden.

Für genügend große n gilt:

$$b \geq n-1 \qquad\qquad (*)$$

Dies folgt leicht aus der Tatsache, daß die Anzahl der Schaltkreise mit n Eingängen, m Ausgängen und Komplexität $n-1$ kleiner ist als $|B^*_{n,m}|$. Setzt man zur Abkürzung $d := \frac{1}{\sqrt{2\pi}}$, so gilt nämlich:

$$\underbrace{\log d + 2(n-1)\log(2(n-1)) + 4(n-1) + (m-n-\frac{3}{2})\log(n-1) + (n-1)\log e}_{\log S(n-1,n,m)}$$

$$< \underbrace{m2^n - \frac{2^n}{n^2}}_{\log |B^*_{n,m}|}$$

$$\Rightarrow \qquad S(n-1,n,m) < |B^*_{n,m}|$$

Realisierungen zu verschiedenen Funktionen sind trivialerweise verschieden[4]. Wenn es in $B^*_{n,m}$ also mehr Funktionen gibt als Schaltkreise mit Komplexität $n-1$, so gibt es Funktionen in $B^*_{n,m}$ mit höherer Komplexität als $n-1$.

Unter Zuhilfenahme dieser groben Abschätzung (*) soll b nun genauer bestimmt werden.

Ausgangspunkt der Überlegung ist wiederum die Tatsache, daß es mindestens so viele verschiedene Schaltkreise mit n Eingängen und m Ausgängen geben muß wie es Funktionen in $B^*_{n,m}$ gibt.

$$\Rightarrow \qquad S(b,n,m) \geq |B^*_{n,m}|$$

$$\Leftrightarrow \qquad \log S(b,n,m) \geq \log |B^*_{n,m}|$$

$$\Leftrightarrow \qquad 2b\log(b+n-1) + 4b + (m-b-1/2)\log b + b\log e + \log d \geq m2^n - \frac{2^n}{n^2}$$

$$\overset{(*)}{\Rightarrow} \qquad 2b(\log b + 1) + 4b + (m-b-1/2)\log b + b\log e + \log d \geq m2^n - \frac{2^n}{n^2}$$

$$\Leftrightarrow \qquad b(\log b + 6 + \log e) + (m-1/2)\log b + \log d \geq m2^n - \frac{2^n}{n^2}$$

Falls $b \leq m\frac{2^n}{n}$ wäre, würde gelten:

$$m\frac{2^n}{n}(n - \log n + \log m + 6 + \log e) + (m-1/2)(n - \log n + \log m) + \log d \geq m2^n - \frac{2^n}{n^2}$$

[4] Hier wird *nicht* ausgenutzt, daß häufig verschiedene Schaltkreise die gleiche Funktion realisieren.

Mit $m \leq \frac{1}{(1+\delta)64e} \cdot n$ gilt:

$m\frac{2^n}{n}(n - \log n + (\log n - \log(1+\delta) - 6 - \log e) + 6 + \log e)$
$+(m-1/2)(n - \log n + \log m) + \log d \geq m2^n - \frac{2^n}{n^2}$

$\Leftrightarrow$

$$m\frac{2^n}{n}(n - \underbrace{\log(1+\delta)}_{:=\varepsilon>0}) + (m-1/2)(n - \log n + \log m) + \log d \geq m2^n - \frac{2^n}{n^2}$$

$\Leftrightarrow$

$$\varepsilon m\frac{2^n}{n} - (m-1/2)(n - \log n + \log m) - \log d \leq \frac{2^n}{n^2}$$

Diese Ungleichung ist falsch für genügend große n.

Folglich ist die Annahme, daß $b \leq m\frac{2^n}{n}$ ist, falsch.
Für genügend große n muß also gelten:

$$b = max\{C_{B_2}(f) \mid f \in B^*_{n,m}\} > m\frac{2^n}{n}$$

Wählt man nun $B^*_{n,m}$ als Menge der $2^{m2^n - 2^n/n^2}$ Funktionen aus $B_{n,m}$ mit geringster Komplexität, so gilt für alle Funktionen $f \in B_{n,m} \setminus B^*_{n,m}$:

$$C_{B_2}(f) \geq \max\{C_{B_2}(g) \mid g \in B^*_{n,m}\} > m\frac{2^n}{n}$$

Da aber

$$|B_{n,m} \setminus B^*_{n,m}| = 2^{m2^n} - 2^{m2^n - 2^n/n^2} = 2^{m2^n} \cdot (1 - \underbrace{2^{-2^n/n^2})}_{\to 0 \text{ für } n\to\infty} =$$

$$= |B_{n,m}| \cdot (1 - 2^{-s(n)}),$$

ist die Aussage des Satzes gezeigt. □

Ist m nicht zu groß, so haben also „fast alle“ Funktionen aus $B_{n,m}$ eine Komplexität $> m\frac{2^n}{n}$. Will man andererseits eine Funktion aus $B_{n,m}$ realisieren, so kann man leicht eine Realisierung mit Kosten $m\frac{2^n}{n} + o(m\frac{2^n}{n})$ finden, indem man die m Ausgangsfunktionen (unter Zuhilfenahme des Satzes von Lupanov) getrennt realisiert. Insofern tritt auch hier ein „Shannon–Effekt“ auf.

Bei zufällig gewählten Funktionen aus $B_{n,m}$ kann man infolgedessen nur für große m darauf hoffen, daß eine gemeinsame Realisierung der Ausgangsfunktionen Vorteile im Vergleich zu einer getrennten Realisierung hat.

Bei Funktionen, die in der Praxis auftreten, sieht die Situation allerdings ganz anders aus:
Häufig weisen bei Funktionen mit mehreren Ausgängen die verschiedenen Ausgangsfunktionen eine ähnliche Struktur auf. Effiziente Realisierungen nutzen diese Tatsache aus, indem sie gleiche Teilschaltungen bei der Realisierung mehrerer

Ausgangsfunktionen benutzen. Die Effizienz solcher Realisierungen beruht oft gerade auf der „Mehrfachverwendung“ gleicher Teilschaltungen.

Es lohnt sich also, bei der Logiksynthese nach Möglichkeiten zur Ausnutzung gemeinsamer Teilschaltungen für mehrere Ausgangsfunktionen zu suchen.

Auch hierzu werden in den Kapiteln 3 und 4 Ansätze vorgestellt.

Kapitel 3

Ausnutzung von Symmetrien

In dem grundlegenden Kapitel über boolesche Algebren wurden Ergebnisse zu Invarianz boolescher Unteralgebren unter Automorphismen aufgeführt.

In diesem Kapitel soll nun die Ausnutzung solcher Eigenschaften bei der Synthese von Schaltkreisen zu booleschen Funktionen aus $S(D)$ im Mittelpunkt stehen.

3.1 G–symmetrische Funktionen

Automorphismen auf $S(D)$ werden induziert durch Permutationen auf $D \subseteq \{0,1\}^n$.

Zur Definition der G–Symmetrie wie in [Hot74] beschränkt man die Betrachtung auf spezielle Automorphismen, die von Permutationen aus einer Untergruppe $\mathbf{P}_n$ von $Per(\{0,1\}^n)$ induziert werden. Hierbei ist $\mathbf{P}_n$ die Gruppe, die durch die Abbildungen

$$\sigma_{ik}, \quad 1 \leq i < k \leq n, \text{ und}$$

$$\nu_i, \quad 1 \leq i \leq n$$

erzeugt werden, wobei $\forall \alpha \in \{0,1\}^n$

$$\sigma_{ik}(\alpha_1, \ldots, \alpha_i, \ldots, \alpha_k, \ldots, \alpha_n) = (\alpha_1, \ldots, \alpha_k, \ldots, \alpha_i, \ldots, \alpha_n)$$

$$\nu_i(\alpha_1, \ldots, \alpha_i, \ldots, \alpha_n) = (\alpha_1, \ldots, \overline{\alpha_i}, \ldots, \alpha_n).$$

Definition 3.1 (G–Symmetrie)
$f \in S(D)$ ($D \subseteq \{0,1\}^n$) heißt G–symmetrisch für $G \subseteq \mathbf{P}_n$, falls für alle $\sigma \in G$ gilt:
$\sigma(D) = D$ und das Bild von f unter dem durch $\sigma|_D$ induzierten Automorphismus h_σ ist $h_\sigma(f) = f$, d. h. es gilt $\forall x \in D$:

$$f(\sigma(x)) = f(x)$$

Definition 3.2 (Erweiterung) *Sei $f \in S(D)$.*
$f' \in S(D')$ mit $D \subseteq D'$ heißt Erweiterung von f, falls $f'(x) = f(x)\ \forall x \in D$.

Bemerkung 7 *Ist eine Funktion $f \in S(D)$ nicht G–symmetrisch, da es $\sigma \in G$ gibt mit $\sigma(D) \neq D$, so kann es trotzdem eine Erweiterung $f' \in S(D)$ geben, die G–symmetrisch ist. Jeder Schaltkreis, der f' realisiert, realisiert auch f. Will man bei der Realisierung von f G–Symmetrie ausnutzen, so ist es häufig vorteilhaft, zu einer Erweiterung von f überzugehen und diese zu realisieren.*

Etliche allgemein gebräuchliche Symmetriedefinitionen sind als Spezialfälle der G–Symmetrie anzusehen:

Bezeichnung 10

- *Ist f $\{\sigma_{ik} \,|\, 1 \leq i < k \leq n\}$–symmetrisch, so bezeichnet man f auch als totalsymmetrisch.*
  Es gilt dann
  $$f(\alpha_1, \ldots, \alpha_n) = f(\beta_1, \ldots, \beta_n) \; \forall \alpha, \beta \text{ mit } \sum_{i=1}^{n} \alpha_i = \sum_{i=1}^{n} \beta_i.$$

- *Ist f $\{\sigma_{ik} \,|\, 1 \leq i < k \leq n;\, i, k \in \lambda\}$–symmetrisch ($\lambda \subseteq \{1, \ldots, n\}$, $|\lambda| > 1$), so bezeichnet man f als teilweise symmetrisch in der Variablenmenge λ.*
  Es gilt dann
  $$f(\alpha_1, \ldots, \alpha_n) = f(\beta_1, \ldots, \beta_n)$$
  $$\forall \alpha, \beta \text{ mit } \sum_{i \in \lambda} \alpha_i = \sum_{i \in \lambda} \beta_i \text{ und } \alpha_i = \beta_i \text{ für } i \in \{1, \ldots, n\} \setminus \lambda.$$

- *Ist f $\{\sigma_{ik}\}$–symmetrisch für $1 \leq i < k \leq n$, so bezeichnet man f als (paarweise) symmetrisch in den Variablen (x_i, x_k).*
  Es gilt dann $\forall \alpha_l \in \{0, 1\}$, $l \in \{1, \ldots, n\} \setminus \{i, k\}$
  $$f(\alpha_1, \ldots, \alpha_{i-1}, 0, \alpha_{i+1}, \ldots, \alpha_{k-1}, 1, \alpha_{k+1}, \ldots, \alpha_n) = f(\alpha_1, \ldots, \alpha_{i-1}, 1, \alpha_{i+1}, \ldots, \alpha_{k-1}, 0, \alpha_{k+1}, \ldots, \alpha_n)$$

- *Ist f $\{\nu_i \circ \sigma_{ik} \circ \nu_i\}$–symmetrisch für $1 \leq i < k \leq n$, so bezeichnet man f als äquivalenzsymmetrisch in den Variablen (x_i, x_k).*
  Es gilt dann $\forall \alpha_l \in \{0, 1\}$, $l \in \{1, \ldots, n\} \setminus \{i, k\}$
  $$f(\alpha_1, \ldots, \alpha_{i-1}, 0, \alpha_{i+1}, \ldots, \alpha_{k-1}, 0, \alpha_{k+1}, \ldots, \alpha_n) = f(\alpha_1, \ldots, \alpha_{i-1}, 1, \alpha_{i+1}, \ldots, \alpha_{k-1}, 1, \alpha_{k+1}, \ldots, \alpha_n)$$

- *Ist f $\{\nu_i\}$–symmetrisch für $1 \leq i \leq n$, so ist f unabhängig von der Eingangsvariablen x_i.*
  Es gilt dann $\forall \alpha_l \in \{0, 1\}$, $l \in \{1, \ldots, n\} \setminus \{i\}$
  $$f(\alpha_1, \ldots, \alpha_{i-1}, 0, \alpha_{i+1}, \ldots, \alpha_n) = f(\alpha_1, \ldots, \alpha_{i-1}, 1, \alpha_{i+1}, \ldots, \alpha_n)$$

Bemerkung 8 *Ohne Beschränkung der Allgemeinheit kann man annehmen, daß G eine Gruppe ist, denn wenn f G–symmetrisch ist, dann ist f auch $< G >$–symmetrisch, wobei $< G >$ die von G erzeugte Untergruppe von $\mathbf{P}_n$ bezeichnet.*

3.2 Realisierungen mit mehreren Polynomstufen

Eine Möglichkeit der mehrstufigen Realisierung einer booleschen Funktion $f \in B_{n,m}$ ist nach [Hot74] die Darstellung als Schaltkreis mit mehreren Polynomstufen. Dabei werden wie in Abbildung 3.1 angedeutet k Schaltkreise zu booleschen Polynomen $P_1, \dots, P_k$ mit mehreren Ausgängen „hintereinandergeschaltet“[1].

Es folgt eine formale Definition für Schaltkreise mit mehreren Polynomstufen. Dabei werden Funktionen $a_i : \mathbf{N} \rightsquigarrow E$ verwendet. $a_i(j)$ liefert eine Kante des j. Ausgangs des i. Polynoms.

Definition 3.3 (Schaltkreis mit mehreren Polynomstufen)

- *Ist $S = ((V,E), typ, pe, pa)$ die Interpretation eines booleschen Polynoms mit m Ausgängen über Var_n, so ist S ein Schaltkreis mit 1 Polynomstufe, n Eingängen und m Ausgängen. Die Ausgangsnumerierungsfunktion $a_1 : \{1, \dots, m\} \to E$ ist folgendermaßen definiert:*

$$a_1(j) = (v, pa(j)) \text{ mit } (v, pa(j)) \in E$$

- *Sei $S_k = ((V_k, E_k), typ_k, pe_k, pa_k)$ ein Schaltkreis mit k Polynomstufen $(k \geq 1)$, n Eingängen und m_k Ausgängen. Seien $a_1{}', \dots, a_k{}'$ die zugehörigen Ausgangsnumerierungsfunktionen der 1. bis k. Stufe. Sei $S_P = ((V_P, E_P), typ_P, pe_P, pa_P)$ die Interpretation eines booleschen Polynoms mit m Ausgängen über Var_{m_k}. Dann ist*

$$S = ((V,E), typ, pe, pa) = s\_comp(S_k, S_P)$$

ein Schaltkreis mit $(k+1)$ Polynomstufen, n Eingängen und m Ausgängen. Für die zugehörigen Ausgangsnumerierungsfunktionen a_1 bis a_{k+1} der 1. bis $(k+1)$. Stufe gilt:

$$a_1 = a_1{}', \dots, a_{k-1} = a_{k-1}{}'$$

$a_k : \{1, \dots, m_k\} \to E,\ a_k(j) = (v, u)$
für beliebiges $(v,u) \in E$ mit $(v, pa_k(j)) \in E_k$

$$a_{k+1} : \{1, \dots, m\} \to E,\ a_{k+1}(j) = (v, pa(j)) \text{ mit } (v, pa(j)) \in E$$

Sei $S = ((V,E), typ, pe, pa)$ nun ein Schaltkreis mit k Polynomstufen, n Eingängen und m Ausgängen und sei für $1 \leq i \leq k$ $a_i : \{1, \dots, m_i\} \to E$ die Ausgangsnumerierungsfunktion der i. Stufe $(m_k = m)$. Die j. Ausgangsfunktion des i. Polynoms ist dann die durch die Leitung $a_i(j)$ berechnete Funktion $f_{a_i(j)}$:

[1] Die Realisierung durch mehrere *Polynom*stufen stellt keine wirkliche Einschränkung dar. Ist ein $\{and_2, or_2, not\}$–Schaltkreis gegeben, so erkennt man anhand einer beliebigen topologischen Sortierung des Schaltkreises sofort, daß es sich um einen Schaltkreis mit mehreren Polynomstufen handelt. Hierbei kann es sich bei den Polynomen auf den einzelnen Stufen allerdings um sehr triviale „Polynome“ handeln, die teilweise nur aus einem Und–Gatter, Oder–Gatter oder Inverter bestehen.

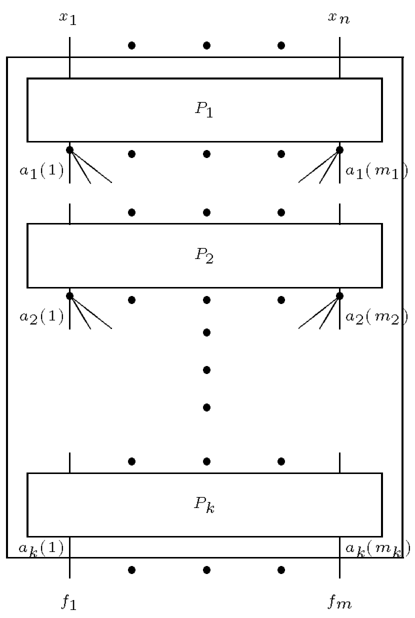


Abbildung 3.1: Schaltkreis mit k Polynomstufen

$\{0,1\}^n \rightarrow \{0,1\}$. Im folgenden Satz wird $f_{a_i(j)}$ durch $f_j^{(i)}$ bezeichnet. Zusätzlich bezeichnet für $1 \leq j \leq n$ $f_j^{(0)}$ die j. Projektion π_j^n aus B_n. Für die durch S definierte Funktion $f \in B_{n,m}$ gilt:

$$f_j = f_j^{(k)} = f_{a_k(j)} \quad \forall 1 \leq j \leq m$$

Satz 3.1 *Unter den oben angegebenen Voraussetzungen gilt f. a.* $1 \leq i \leq k$*:*

$$< f_1^{(i)}, \ldots, f_{m_i}^{(i)} > \subseteq < f_1^{(i-1)}, \ldots, f_{m_{i-1}}^{(i-1)} > .$$

Beweis:
Nach Definition von Schaltkreisen mit k Polynomstufen läßt sich jede Funktion $f_j^{(i)}$ $(1 \leq j \leq m_i)$ mit Hilfe von Operationen $\vee$, $\cdot$ bzw. $^-$ aus den Funktionen $f_1^{(i-1)}, \ldots, f_{m_{i-1}}^{(i-1)}$ gewinnen. Also gilt:

$$\{f_1^{(i)}, \ldots, f_{m_i}^{(i)}\} \subseteq < f_1^{(i-1)}, \ldots, f_{m_{i-1}}^{(i-1)} >$$

und damit auch

$$< f_1^{(i)}, \ldots, f_{m_i}^{(i)} > \subseteq < f_1^{(i-1)}, \ldots, f_{m_{i-1}}^{(i-1)} >,$$

da $< f_1^{(i-1)}, \ldots, f_{m_{i-1}}^{(i-1)} >$ hinsichtlich $\vee$, $\cdot$ und $^-$ abgeschlossen ist.

□

Jeder Schaltkreis mit k Polynomstufen liefert also eine aufsteigende Kette von Unteralgebren von B_n:

$$< f_1 \ldots, f_m > = < f_1^{(k)}, \ldots, f_{m_k}^{(k)} > \subseteq < f_1^{(k-1)}, \ldots, f_{m_{k-1}}^{(k-1)} >$$
$$\subseteq \ldots \subseteq < f_1^{(0)}, \ldots, f_n^{(0)} > = B_n.$$

Dabei bilden die Funktionen $f_1^{(i)}, \ldots, f_{m_i}^{(i)}$ ein ausgezeichnetes Erzeugendensystem von $< f_1^{(i)}, \ldots, f_{m_i}^{(i)} >$.

Hat man eine solche aufsteigende Kette gegeben und zusätzlich die ausgezeichneten Erzeugendensysteme $f_1^{(i)}, \ldots, f_{m_i}^{(i)}$, so kann man leicht einen entsprechenden Schaltkreis mit k Polynomstufen zurückgewinnen[2].

Gemäß Abschnitt 1.2.7 läßt sich zu jeder Unteralgebra M' einer booleschen Algebra M eine Gruppe $G(M, M')$ von Automorphismen auf M finden, die genau die Elemente von M' festlassen. Bestimmt man wiederum die Menge aller Elemente von M, die unter *allen* Automorphismen aus $G(M, M')$ festbleiben, so erhält man $M(G(M, M')) = M'$ (siehe Lemma 1.12).

Also kann man statt der aufsteigenden Kette

$$\underbrace{< f_1^{(k)}, \ldots, f_{m_k}^{(k)} >}_{:=M^{(k)}} \subseteq \ldots \subseteq \underbrace{< f_1^{(0)}, \ldots, f_n^{(0)} >}_{:=M^{(0)}=B_n}$$

ebenso gut eine absteigende Kette von Untergruppen der Automorphismengruppe $G(B_n)$ angeben (vgl. Lemma 1.13):

$$G(B_n, M^{(k)}) \supseteq \ldots \supseteq G(B_n, M^{(1)}) \supseteq \underbrace{G(B_n, M^{(0)})}_{=\{id_{B_n}\}}$$

Bestimmt man jeweils die Menge aller Elemente aus B_n, die unter *allen* Automorphismen aus $G(B_n, M^{(i)})$ festbleiben, so erhält man wegen

$$M(G(B_n, M^{(i)})) = M^{(i)}$$

die ursprüngliche aufsteigende Kette zurück. Nun müssen für die Unteralgebren $M^{(i)}$ noch geeignete Erzeugendensysteme gewählt werden und man erhält dann eine Realisierung als Schaltkreis mit k Polynomstufen.

Auf diesem Gedanken beruht das in [Hot74] vorgeschlagene Logiksyntheseverfahren:
Ausgehend von einer absteigenden Kette

$$G_k \supseteq \ldots \supseteq G_1 \supseteq G_0$$

von Untergruppen der Automorphismengruppe $G(B_n)$ mit

$$G_k = G(B_n, < f_1, \ldots, f_m >) \text{ und } G_0 = \{id_{B_n}\}$$

wird ein Schaltkreis mit k Polynomstufen zur Realisierung von $(f_1, \ldots, f_m)$ bestimmt.

[2]Man wird aufgrund der Mehrdeutigkeit der Darstellung einer Funktion als boolesches Polynom nicht immer exakt den gleichen Schaltkreis mit k Polynomstufen erhalten. Verwendet man Minimalpolynome, so wird man allerdings immer einen Schaltkreis finden, dessen Kosten kleiner oder gleich den Kosten des ursprünglichen Schaltkreises sind.

3.3 Ausnutzung von Symmetrieeigenschaften

3.3.1 Ausnutzung von G–Symmetrie

Wie in [Hot74] vorgeschlagen, kann man sich bei der Wahl der absteigenden Kette von Automorphismengruppen beispielsweise auf Automorphismen beschränken, die durch Permutationen aus der in Abschnitt 3.1 definierten Gruppe $\mathbf{P}_n$ induziert werden.

Soll eine Funktion $f = (f_1, \ldots, f_m) \in B_{n,m}$ realisiert werden, so gibt man eine absteigende Kette $G_k \supseteq \ldots \supseteq G_0$ von Untergruppen von $\mathbf{P}_n$ vor, wobei man weiß, daß alle Funktionen f_j $(1 \leq j \leq m)$ G_k–symmetrisch sind.

Dies bedeutet, daß alle Funktionen f_j $(1 \leq j \leq m)$ invariant sind unter den Automorphismen, die durch die Elemente von G_i $(0 \leq i \leq k)$ induziert werden[3].

Ist eine absteigende Kette $G_k \supseteq \ldots \supseteq G_0$ von Untergruppen von $\mathbf{P}_n$ gegeben, so wird — nach Wahl von Erzeugendensystemen für $M(B_n, G_i)$ — wie oben angedeutet ein Schaltkreis mit k Polynomstufen bestimmt, der f realisiert.

Die dabei verwendeten Symmetrieeigenschaften können wie in Abschnitt 3.4 beschrieben bestimmt werden oder sie sind dem Benutzer aufgrund einer genaueren Kenntnis der zu realisierenden Funktion bekannt und werden dem Logiksynthesealgorithmus als Vorgaben mitgeteilt.

Will man auf diese Weise mehrstufige Realisierungen erzeugen, so muß man in der Lage sein, zu einer vorgegebenen Gruppe G von Permutationen auf $D \subseteq \{0,1\}^n$ die Unteralgebra $M(S(D), G)$ zu bestimmen. Der nun folgende Satz 3.2 gibt an, wie man die Atome von $M(S(D), G)$ erhält. Dabei wird wiederum die Isomorphie zwischen den Funktionen aus $S(D)$ und ihren zugehörigen ON–Mengen benutzt: Die Anwendung eines Automorphismus $h_\sigma \in G(S(D))$, der durch $\sigma \in Per(D)$ induziert wird, auf $f \in S(D)$ läßt sich so als Anwendung von σ auf $ON(f)$ betrachten.

Die Anwendung von Permutationen aus einer Gruppe $G \subseteq PerD$ auf die Elemente aus D teilt D in sog. **Orbits** ein:
Ein Orbit eines Elementes $\alpha \in D$ ist definiert als

$$orbit_G(\alpha) = \{\beta \in D \mid \exists \sigma \in G \text{ mit } \sigma(\alpha) = \beta\}$$

Definiert man durch

$$„\alpha \sim \beta \iff \beta \in orbit_G(\alpha) \iff \exists \sigma \in G \text{ mit } \sigma(\alpha) = \beta“$$

eine Relation auf D, so folgt aus der Tatsache, daß G eine Gruppe ist, leicht, daß „$\sim$“ eine Äquivalenzrelation ist. Die Orbits der Elemente von D sind gerade die Äquivalenzklassen dieser Relation und bilden folglich eine Partition auf D.

[3] Ist aus dem Zusammenhang klar, was gemeint ist, so wird im folgenden der Einfachheit halber nicht mehr zwischen Permutationen aus $\{0,1\}^n$ und den durch diese induzierten Automorphismen auf B_n unterschieden. Ist $G_i \subseteq Per\{0,1\}^n$, so ist mit $M(B_n, G_i)$ die Unteralgebra aller Funktionen aus B_n gemeint, die unter allen Automorphismen fest bleiben, die von Elementen von G_i induziert werden.

Satz 3.2 *Sei $G \subseteq PerD$ eine Untergruppe von $PerD$. Sei für $\alpha \in D$*

$$f_{orbit_G(\alpha)} = ON^{-1}(orbit_G(\alpha)) \in S(D).$$

Dann gilt für die Atome von $M(S(D), G)$:

$$A(M(S(D), G)) = \{f_{orbit_G(\alpha)} \mid \alpha \in D\}.$$

Beweis:

- $\sigma(orbit_G(\alpha)) = orbit_G(\alpha)$ f. a. $\sigma \in G, \alpha \in D$
  $\Longrightarrow$ $f_{orbit_G(\alpha)}$ ist invariant unter dem von σ induzierten Automorphismus
  $\Longrightarrow$ $f_{orbit_G(\alpha)} \in M(S(D), G)$
- $\forall g \in M(S(D), G)$ gilt:

  $$ON(g) \cap orbit_G(\alpha) = \begin{cases} orbit_G(\alpha), \text{ falls } \alpha \in ON(g) \\ \emptyset \text{ sonst} \end{cases}$$

  1. Fall: $\alpha \in ON(g)$
  Sei $\beta \in orbit_G(\alpha)$. $\Rightarrow$ $\exists \sigma \in G$ mit $\sigma(\alpha) = \beta$
  $\Rightarrow$ $\beta \in ON(g)$, da $\sigma(ON(g)) = ON(g)$
  Also gilt $orbit_G(\alpha) \subseteq ON(g)$.
  $\Rightarrow$ $ON(g) \cap orbit_G(\alpha) = orbit_G(\alpha)$.

  2. Fall: $\alpha \notin ON(g)$
  Sei $\beta \in orbit_G(\alpha)$. $\Rightarrow$ $\exists \sigma \in G$ mit $\sigma(\alpha) = \beta$ $\Rightarrow$ $\sigma^{-1}(\beta) = \alpha$
  Angenommen $\beta \in ON(g)$.
  Dann wäre $\alpha \in ON(g)$, da $\sigma^{-1}(ON(g)) = ON(g)$.
  Also gilt $\beta \notin ON(g)$.
  $\Rightarrow$ $orbit_G(\alpha) \cap ON(g) = \emptyset$.

  Folglich gilt für $g \in M(S(D), G)$:

  $$f_{orbit_G(\alpha)} \cdot g = \begin{cases} f_{orbit_G(\alpha)}, \text{ falls } f_{orbit_G(\alpha)} \cdot g \neq \emptyset \\ 0 \text{ sonst} \end{cases}$$

- $$\bigcup_{\alpha \in D} orbit_G(\alpha) = D \quad \Longrightarrow \quad \bigcup_{\alpha \in D} f_{orbit_G(\alpha)} = 1$$

  $\Rightarrow$ $\{f_{orbit_G(\alpha)} \mid \alpha \in D\}$ enthält *alle* Atome von $M(S(D), G)$.

$\square$

Ist Ausnutzung von G–Symmetrie in jedem Fall vorteilhaft?

Man darf allerdings *nicht* erwarten, daß man durch Vorgabe irgendeiner G–Symmetrie schon automatisch eine vorteilhafte Realisierung unter Ausnutzung dieser Symmetrieeigenschaft erhält. Soll eine Realisierung für

$$f = (f_1, \ldots, f_m) \in B_{n,m}$$

gefunden werden und weiß man, daß alle $f_i (1 \leq i \leq m)$ G–symmetrisch sind, so kann man f nach dem beschriebenen Verfahren durch einen Schaltkreis mit 2 Polynomstufen realisieren. Als Erzeugendensystem von $(M(S(D), G)$ sollen die wie in Satz 3.2 berechneten Atome verwendet werden. In der 1. Stufe werden die Atome durch ein Minimalpolynom realisiert, die 2. Stufe liefert die Funktionen f_i als Disjunktionen dieser Atome. Diese Realisierung wird im folgenden als Realisierung **A** bezeichnet.

Es wird nun eine (triviale) Realisierung **B** angegeben, die geringere Kosten als Realisierung **A** hat und somit zeigt, daß es in diesem Fall unnötig war, nach der G–Symmetrie zu suchen und diese auszunutzen.

Bei Realisierung **B** wird G ersetzt durch die Menge *aller* Automorphismen $G(S(D), < f_1, \ldots, f_m >)$, unter denen die Elemente von $< f_1, \ldots, f_m >$ fest bleiben. Es gilt:

$$M(S(D), G(S(D), < f_1, \ldots, f_m >)) =< f_1, \ldots, f_m >$$

Um die Atome von $< f_1, \ldots, f_m >$ zu bestimmen, muß man

$$G(S(D), < f_1, \ldots, f_m >)$$

allerdings nicht kennen! Die Menge der Atome von $< f_1, \ldots, f_m >$ ist nach Lemma 1.6 gegeben durch

$$A(< f_1, \ldots, f_m >) = \{f_1^{\epsilon_1} \cdot \ldots \cdot f_m^{\epsilon_m} \mid (\epsilon_1, \ldots, \epsilon_m) \in \{0,1\}^m,\, f_1^{\epsilon_1} \cdot \ldots \cdot f_m^{\epsilon_m} \neq 0\}.$$

Realisierung **B** berechnet in einer 1. Stufe die Atome von $< f_1, \ldots, f_m >$ durch ein Polynom, die 2. Stufe liefert dann die Funktionen f_i als Disjunktionen dieser Atome. Darstellungen von Funktionen als Disjunktionen ihrer Atome sind eindeutig. Jede Funktion f_i $(1 \leq i \leq m)$ kann also eindeutig dargestellt werden als

$$\bigvee_{\substack{(\epsilon_1, \ldots, \epsilon_m) \in \{0,1\}^m, \epsilon_i = 1, \\ f_1^{\epsilon_1} \cdot \ldots \cdot f_m^{\epsilon_m} \neq 0}} f_1^{\epsilon_1} \cdot \ldots \cdot f_m^{\epsilon_m}.$$

Es gilt $G \subseteq G(S(D), < f_1, \ldots, f_m >)$ und folglich

$$M(S(D), G(S(D), < f_1, \ldots, f_m >)) \subseteq M(S(D), G)$$

d. h.

$$< f_1, \ldots, f_m > \subseteq M(S(D), G)$$

Die Atome von $< f_1, \ldots, f_m >$ liefern also eine Partition auf den Atomen von $M(S(D), G)$. Für ein beliebiges Atom von $< f_1, \ldots, f_m >$ gilt:

$$f_1^{\epsilon_1} \cdot \ldots \cdot f_m^{\epsilon_m} = a_1^{(\epsilon)} \vee \ldots \vee a_{m_\epsilon}^{(\epsilon)},$$

wobei $a_i^{(\epsilon)}$ Atome von $M(S(D), G)$ sind.

Das Polynom der 1. Stufe von Realisierung **B** gehe nun aus dem Minimalpolynom der 1. Stufe von Realisierung **A** hervor, indem zur Bildung eines Atoms $f_1^{\epsilon_1} \cdot \ldots \cdot f_m^{\epsilon_m}$ die Atome von $M(S(D), G)$ $a_1^{(\epsilon)}, \ldots, a_{m_\epsilon}^{(\epsilon)}$ bei $m_\epsilon > 1$ disjunktiv verknüpft werden.

Ist $\sum_{i=1}^{m} \epsilon_i = e > 1$, so wird das Atom $f_1^{\epsilon_1} \cdot \ldots \cdot f_m^{\epsilon_m}$ in e Disjunktionen der 2. Stufe von **B** verwendet. Der Unterschied zwischen Realisierung **A** und **B** besteht darin, daß in Realisierung **B** die disjunktive Verknüpfung von $a_1^{(\epsilon)}, \ldots, a_{m_\epsilon}^{(\epsilon)}$ nur einmal erfolgt (in der 1. Stufe), bei Realisierung **A** jedoch e Mal, nämlich in der 2. Stufe.

Das Atom $\overline{f_1} \cdot \ldots \cdot \overline{f_m} = f_1^0 \cdot \ldots \cdot f_m^0$ wird als einziges in der 2. Stufe von Realisierung **B** nicht verwendet, ebenso werden die Atome $a_1^{(0)}, \ldots, a_{m_0}^{(0)}$ von $M(S(D), G)$, deren Disjunktion $\overline{f_1} \cdot \ldots \cdot \overline{f_m}$ ergibt, in der 2. Stufe von Realisierung **A** nicht verwendet. Läßt man die Realisierung dieser Atome sowohl in Lösung **A** als auch in Lösung **B** weg, so erkennt man: Die Kosten von Realisierung **B** sind kleiner oder gleich den Kosten von Realisierung **A**.

Die Ausnutzung der G–Symmetrie in Realisierung **A** liefert kein besseres Ergebnis als die triviale Realisierung **B**.

Zu einer guten Realisierung durch Ausnutzung von G–Symmetrie kann man nur kommen,

- wenn man längere absteigende Ketten $G_k \supseteq \ldots \supseteq G_0$ betrachtet
  oder

- wenn man „günstige“ Erzeugendensysteme der Unteralgebren $M(S(D), G_i)$ findet (Die Wahl der Atome der Unteralgebren ist offensichtlich nicht immer günstig.)
  oder

- wenn man bei der Verwendung der Atome als Erzeugendensystem diese *nicht* durch ein Polynom realisiert. (Eine Alternative ist in [Hot74] für den Fall angegeben, daß $< f_1, \ldots, f_m >$ G–symmetrisch ist und G sich darstellen läßt als direktes Produkt $G = G_1 \times \ldots \times G_h$, wobei die Gruppen G_i $(1 \leq i \leq h)$ nur auf $< x_{j_1}, \ldots, x_{j_i} >$ echt operieren mit $\{j_1, \ldots, j_i\} \subseteq X_i$. Die Mengen X_i, $1 \leq i \leq h$, bilden dabei eine Partition auf $\{1, \ldots, n\}$.)

Wahl von Erzeugendensystemen

Hat man eine Unteralgebra M von $S(D)$ gegeben und soll man die Funktionen $f_1, \ldots, f_m$ aus einem Erzeugendensystem von M berechnen, so ist die Wahl des

Erzeugendensystems eine nichttriviale Aufgabe:
Einerseits sind nicht alle Erzeugendensysteme mit den gleichen Kosten zu realisieren, andererseits hängt von der Wahl des Erzeugendensystems stark ab, mit welchen Kosten man $f_1, \ldots, f_m$ aus diesem Erzeugendensystem berechnen kann. Für verschiedene Funktionen $f_1, \ldots, f_m$ sind auch verschiedene Erzeugendensysteme günstig: Wählt man $g_1, \ldots, g_k$ als Erzeugendensystem von M, so sind im Extremfall die Kosten zur Berechnung von $f_1, \ldots, f_m$ 0, wenn gerade $f_i = g_i$ $(1 \leq i \leq m \leq k)$, bei anderer Wahl von $f_1, \ldots, f_m$ können die Kosten aber sehr hoch sein.

Trotzdem könnte es sein, daß sich bei der Behandlung spezieller Funktionenklassen bestimmte Erzeugendensysteme anbieten. Dazu soll (auch im Hinblick auf Abschnitt 3.3.2) eine einfaches Beispiel betrachtet werden:

Ein Beispiel zur Wahl von Erzeugendensystemen

Es sollen k totalsymmetrische Funktionen $f_1, \ldots, f_k \in B_n$ realisiert werden. Dazu soll auf einer 1. Stufe ein Erzeugendensystem für die Unteralgebra S aller totalsymmetrischen Funktionen realisiert werden, auf einer 2. Stufe sollen dann $f_1, \ldots, f_k$ realisiert werden. Weder bei der Realisierung des Erzeugendensystems noch bei der Realisierung von $f_1, \ldots, f_k$ mit Hilfe des Erzeugendensystems ist eine Beschränkung auf Polynome vorgegeben.

Konkret werden die Kosten bei 2 verschiedenen Erzeugendensystemen verglichen: Das 1. Erzeugendensystem besteht aus den Atomen von S[§], das 2. Erzeugendensystem besteht aus den Komponenten der Funktion $BSUM_n \in B_{n,\lceil \log(n+1) \rceil}$, die aus $(x_1, \ldots, x_n)$ die Binärdarstellung von $\sum_{i=1}^{n} x_i$ berechnet.

Die Anzahl k der zu realisierenden Funktionen sei so groß, daß die Kosten der 2. Stufe die Kosten der 1. Stufe zur Realisierung des Erzeugendensystems dominieren[5]. Man interessiert sich hier also nur für die Kosten auf der 2. Stufe und zwar für die *mittleren* Kosten der 2. Stufe für den Fall, daß $f_1, \ldots, f_k$ zufällig gewählte totalsymmetrische Funktionen sind. Das Vorkommen aller totalsymmetrischen Funktionen sei dabei gleichwahrscheinlich.

[§] Unter Anwendung von Satz 3.2 ergibt sich, daß $a_0, \ldots, a_n$ mit $a_i(x_1, \ldots, x_n) = 1 \Leftrightarrow \sum_{j=1}^{n} x_j = i$ die Atome von S sind.

[5] $BSUM_n$ läßt sich nach [Weg89] durch einen Schaltkreis B mit $C_{B_2}(B) = 8\frac{1}{3}n + O(\log^2 n)$ realisieren.
Aus $BSUM_n$ lassen sich auch die Atome von S berechnen. $BSUM_n$ liefert auf $(x_1, \ldots, x_n)$ die Binärdarstellung von $\sum_{j=1}^{n} x_j$ und $a_i(x_1, \ldots, x_n) = 1 \Leftrightarrow \sum_{j=1}^{n} x_j = i$. Ist $BSUM_n = (b_{m-1}, \ldots, b_0)$, so gilt $a_i = b_{m-1}^{\epsilon_{m-1}} \cdot \ldots \cdot b_0^{\epsilon_0}$, wobei $(\epsilon_{m-1}, \ldots, \epsilon_0)$ die Binärdarstellung von i ist. *Alle* Atome erhält man durch Realisierung von $n+1$ vollständigen Monomen auf $m = \lceil \log(n+1) \rceil$ Eingangsvariablen. Benutzt man zur Erzeugung der Monome einen divide–and–conquer–Ansatz, so benötigt man dazu (unter Zuhilfenahme von Lemma 4.2.3 und Korollar 4.2.4 aus [Weg89]) $n + O(\sqrt{n})$ Zellen. Man kann die Atome von S also durch einen Schaltkreis C erhalten mit $C_{B_2}(C) = 9\frac{1}{3}n + O(\sqrt{n})$.
(Nach eigenen Abschätzungen gilt für den Schaltkreis B aus [Weg89] sogar $C_{B_2}(B) = 6\frac{1}{9}n + O(\log^2 n)$ und somit für Schaltkreis C $C_{B_2}(C) = 7\frac{1}{9}n + O(\sqrt{n})$.)

Erzeugendensystem $b_{m-1}, \dots, b_0$ mit $BSUM_n = (b_{m-1}, \dots, b_0)$

Sei $s \in B_n$ eine beliebige totalsymmetrische Funktion. Dann gibt es $g \in B_m$ mit

$$s(\mathbf{x}) = g(b_{m-1}(\mathbf{x}), \dots, b_0(\mathbf{x}))\ \forall \mathbf{x} \in \{0,1\}^n.$$

Nach dem Satz von Lupanov läßt sich g mit Kosten

$$\frac{2^m}{m} + o(\frac{2^m}{m}) \leq \frac{n+1}{1/2\log(n+1)} + o(\frac{n+1}{1/2\log(n+1)})$$

realisieren ($m = \lceil \log(n+1) \rceil$). Infolgedessen sind die mittleren Kosten für die Realisierung einer zufällig gewählten totalsymmetrischen Funktion bei vorgegebener Realisierung für $BSUM_n \leq \frac{n+1}{1/2\log(n+1)} + o(\frac{n+1}{1/2\log(n+1)})$.

Erzeugendensystem $a_0, \dots, a_n$ (Atome von S)

Sei $s_I = \bigvee_{j \in I} a_j$ eine beliebige totalsymmetrische Funktion.

Die Wahrscheinlichkeit, daß $|I| = i$ $(0 \leq i \leq n+1)$, beträgt $p_i = \frac{\binom{n+1}{i}}{2^{n+1}}$.

Gesucht ist eine Realisierung zu einer Funktion

$$g_I : D \to \{0,1\} \text{ mit } g_I(a_0(\mathbf{x}), \dots, a_n(\mathbf{x})) = s_I(\mathbf{x}),$$

$$D \subseteq \{0,1\}^{n+1},\ D = \{(1,0,\dots,0), (0,1,0,\dots,0), \dots, (0,\dots,0,1)\}.$$

Zur Bestimmung des Erwartungswertes für die Kosten der Realisierung einer zufällig gewählten totalsymmetrischen Funktion s_I bei vorgegebener Realisierung für die Atome benötigt man noch die minimale Komplexität $C_{B_2}(g_I)$ einer solchen Funktion g_I.

Die minimale Komplexität einer geeigneten Funktion g_I läßt sich unter Zuhilfenahme von Lemma 2.2 bestimmen:

Lemma 3.1 *Sei $s_I = \bigvee_{j \in I} a_j \in B_n$ totalsymmetrisch, $|I| = i$. g_I habe folgende Eigenschaft (*)*

$$g_I : D \to \{0,1\} \textit{ mit } g_I(a_0(\mathbf{x}), \dots, a_n(\mathbf{x})) = s_I(\mathbf{x}),$$

$$D \subseteq \{0,1\}^{n+1},\ D = \{(1,0,\dots,0), (0,1,0,\dots,0), \dots, (0,\dots,0,1)\}.$$

Man kann eine Funktion g_I mit Eigenschaft (*) finden mit

$$C_{B_2}(g_I) = \min(i, n+1-i) - 1.$$

Diese Wahl von g_I ist optimal.

Beweis:

1. Es gibt keine Funktion g_I, die (*) erfüllt und von einem Schaltkreis G realisiert wird mit $C_{B_2}(G) < \min(i, n+1-i) - 1$:
   Sei g_I eine Funktion, die (*) erfüllt.
   Sei G ein optimaler Schaltkreis, der g_I realisiert.
   Die durch G berechnete totale Funktion sei $h \in B_{n+1}$.
   h ist *nicht* gleichzeitig von x_i mit $i \in I$ und von x_j mit $j \notin I$ unabhängig, denn wäre dies der Fall, so wäre

$$\begin{aligned}
1 &= g_I(0,\ldots,0,\underbrace{1}_{i},0,\ldots,\underbrace{0}_{j}\ldots 0)\\
&= h(0,\ldots,0,\underbrace{1}_{i},0,\ldots,\underbrace{0}_{j}\ldots 0)\\
&= h(0,\ldots,\underbrace{0}_{i},\ldots,0,\underbrace{1}_{j},0,\ldots,0)\\
&= g_I(0,\ldots,\underbrace{0}_{i},\ldots,0,\underbrace{1}_{j},0,\ldots,0)\\
&= 0
\end{aligned}$$

   $\Longrightarrow$ h ist mindestens abhängig von allen Variablen x_j mit $j \in I$ oder von allen Variablen x_j mit $j \in \{1, \ldots, n\} \setminus I$
   $\Longrightarrow$ h ist abhängig von mindestens $\min(i, n+1-i)$ Variablen

$$\begin{aligned}
&\overset{\text{Lemma 2.2}}{\Longrightarrow} \quad C_{B_2}(h) \geq \min(i, n+1-i) - 1\\
&\Longrightarrow \quad C_{B_2}(g_I) \geq \min(i, n+1-i) - 1
\end{aligned}$$

2. Es gibt eine Funktion g_I, die (*) erfüllt und einen Schaltkreis G_{opt}, der g_I realisiert mit

$$C_{B_2}(G_{opt}) = \min(i, n+1-i) - 1$$

   <u>1. Fall</u>: $\quad i \leq n+1-i \quad \Leftrightarrow \quad i \leq \frac{n+1}{2}$
   Es gilt $s_I = \bigvee_{i \in I} a_i$.
   Wähle g_I als Disjunktion aller Eingangsvariablen x_j mit $j \in I$.
   $\Longrightarrow$ Man kann g_I durch einen Baum aus $i-1$ or_2–Gattern realisieren.
   <u>2. Fall</u>: $\quad i > n+1-i \quad \Leftrightarrow \quad i > \frac{n+1}{2}$
   Es gilt $s_I = \bigvee_{i \in I} a_i = \overline{\bigvee_{i \notin I} a_i}$.
   Wähle g_I als die Negation einer Disjunktion aller Eingangsvariablen x_j mit $j \notin I$.
   $\Longrightarrow$ Man kann g_I durch einen Baum aus $n-i-1$ or_2–Gattern und einem nor_2–Gatter an der Wurzel realisieren.
   In beiden Fällen gilt für den g_I realisierenden Schaltkreis G_{opt}:

$$C_{B_2}(G_{opt}) = \min(i, n+1-i) - 1$$

□

Die Kosten eines optimalen Schaltkreises zur Berechnung von $s_I = \bigvee_{j \in I} a_j$ mit $|I| = i$ betragen also

$$c_i = \min(i, n+1-i) - 1.$$

Damit ergibt sich für den Erwartungswert E_C der Kosten einer zufällig gewählten totalsymmetrischen Funktion s_I:

$$\begin{aligned}
E_C &= \sum_{i=0}^{n+1} p_i \cdot c_i \\
&= \sum_{i=1}^{n} p_i \cdot c_i{}^{\|} \\
&= \sum_{i=1}^{\lfloor\frac{n+1}{2}\rfloor} \frac{\binom{n+1}{i}}{2^{n+1}} \cdot (i-1) + \sum_{i=\lfloor\frac{n+1}{2}\rfloor+1}^{n} \frac{\binom{n+1}{i}}{2^{n+1}} \cdot (n+1-i-1) \\
&= \frac{1}{2^{n+1}} \cdot \left[\sum_{i=1}^{\lfloor\frac{n+1}{2}\rfloor} \binom{n+1}{i} \cdot (i-1) + \sum_{i=\lfloor\frac{n+1}{2}\rfloor+1}^{n} \binom{n+1}{n+1-i} \cdot (n+1-i-1) \right]
\end{aligned}$$

<u>1. Fall</u>: $n+1$ ungerade

$$\begin{aligned}
E_C &= \frac{1}{2^{n+1}} \cdot \left[\sum_{i=1}^{\frac{n}{2}} \binom{n+1}{i} \cdot (i-1) + \sum_{i=1}^{\frac{n}{2}} \binom{n+1}{i} \cdot (i-1) \right] \\
&= \frac{1}{2^n} \cdot \left[\sum_{i=1}^{\frac{n}{2}} \left(\binom{n+1}{i} \cdot i \right) - \sum_{i=1}^{\frac{n}{2}} \binom{n+1}{i} \right] \\
&= \frac{1}{2^n} \cdot \left[\sum_{i=1}^{\frac{n}{2}} (n+1) \cdot \binom{n}{i-1} - \sum_{i=1}^{\frac{n}{2}} \left(\binom{n}{i-1} + \binom{n}{i} \right) \right] \\
&= \frac{1}{2^n} \cdot \left[n \sum_{i=1}^{\frac{n}{2}} \binom{n}{i-1} - \sum_{i=1}^{\frac{n}{2}} \binom{n}{i} \right] \\
&= \frac{1}{2^n} \cdot \left[n \sum_{i=0}^{\frac{n}{2}-1} \binom{n}{i} - \sum_{i=0}^{\frac{n}{2}-1} \binom{n}{i} - \binom{n}{\frac{n}{2}} + \binom{n}{0} \right] \\
&= \frac{1}{2^n} \cdot \left[(n-1) \sum_{i=0}^{\frac{n}{2}-1} \binom{n}{i} - \binom{n}{\frac{n}{2}} + \binom{n}{0} \right] \\
&= \frac{1}{2^n} \cdot \left[(n-1) \underbrace{\left(\sum_{i=0}^{\frac{n}{2}-1} \binom{n}{i} + \frac{1}{2} \binom{n}{\frac{n}{2}} \right)}_{\frac{1}{2}2^n} - \frac{1}{2}(n-1) \binom{n}{\frac{n}{2}} - \binom{n}{\frac{n}{2}} + 1 \right] \\
&= \frac{n-1}{2} + \frac{1}{2^n} \left[1 - \frac{n+1}{2} \binom{n}{\frac{n}{2}} \right]
\end{aligned}$$

‖Bei $I = \emptyset$ bzw. $I = \{0, \ldots, n\}$ handelt es sich um die konstante 0– bzw. 1–Funktion. Die Konstanten werden mit Kosten 0 angenommen.

$$= \frac{n-1}{2} - O\left(\sqrt{n}\right)^{**}$$

2. Fall: $n+1$ gerade

$$\begin{aligned}
E_C &= \frac{1}{2^{n+1}} \cdot \left[\sum_{i=1}^{\frac{n+1}{2}} \binom{n+1}{i} \cdot (i-1) + \sum_{i=1}^{\frac{n+1}{2}-1} \binom{n+1}{i} \cdot (i-1) \right] \\
&= \frac{1}{2^n} \cdot \left[\sum_{i=1}^{\frac{n+1}{2}} \binom{n+1}{i} \cdot (i-1) \right] - \frac{1}{2^{n+1}} \binom{n+1}{\frac{n+1}{2}} \cdot \left(\frac{n+1}{2} - 1 \right) \\
&= \frac{1}{2^n} \cdot \left[\sum_{i=1}^{\frac{n+1}{2}} \left(\binom{n+1}{i} \cdot i \right) - \sum_{i=1}^{\frac{n+1}{2}} \binom{n+1}{i} \right] \\
&\qquad - \frac{1}{2^{n+1}} \binom{n+1}{\frac{n+1}{2}} \cdot \left(\frac{n+1}{2} - 1 \right) \\
&= \frac{1}{2^n} \cdot \left[(n+1) \sum_{i=1}^{\frac{n+1}{2}} \binom{n}{i-1} - \sum_{i=1}^{\frac{n+1}{2}} \left(\binom{n}{i-1} + \binom{n}{i} \right) \right] \\
&\qquad - \frac{1}{2^{n+1}} \binom{n+1}{\frac{n+1}{2}} \cdot \left(\frac{n+1}{2} - 1 \right) \\
&= \frac{1}{2^n} \cdot \left[n \sum_{i=0}^{\frac{n-1}{2}} \binom{n}{i} - \sum_{i=1}^{\frac{n+1}{2}} \binom{n}{i} \right] - \frac{1}{2^{n+1}} \binom{n+1}{\frac{n+1}{2}} \cdot \left(\frac{n+1}{2} - 1 \right) \\
&= \frac{1}{2^n} \cdot \left[(n-1) \underbrace{\sum_{i=0}^{\frac{n-1}{2}} \binom{n}{i}}_{\frac{1}{2}2^n} + \binom{n}{0} - \binom{n}{\frac{n+1}{2}} \right] \\
&\qquad - \frac{1}{2^{n+1}} \binom{n+1}{\frac{n+1}{2}} \cdot \left(\frac{n+1}{2} - 1 \right) \\
&= \frac{n-1}{2} + \frac{1}{2^n} \cdot \left[1 - \binom{n}{\frac{n+1}{2}} + \frac{1}{2} \binom{n+1}{\frac{n+1}{2}} - \frac{n+1}{4} \binom{n+1}{\frac{n+1}{2}} \right] \\
&= \frac{n-1}{2} + \frac{1}{2^n} \cdot \Bigg[1 - \underbrace{\binom{n}{\frac{n+1}{2}} + \frac{1}{2} \binom{n}{\frac{n+1}{2}} + \frac{1}{2} \binom{n}{\frac{n+1}{2} - 1}}_{=0} \\
&\qquad - \frac{n+1}{4} \binom{n+1}{\frac{n+1}{2}} \Bigg] \\
&= \frac{n-1}{2} + \frac{1}{2^n} \cdot \left[1 - \frac{n+1}{4} \binom{n+1}{\frac{n+1}{2}} \right]
\end{aligned}$$

**Mit Hilfe der Stirling–Formel erhält man: $\binom{n}{\frac{n}{2}} = O\left(\frac{2^n}{\sqrt{n}}\right)$

$$= \frac{n-1}{2} - O\left(\sqrt{n}\right)$$

Anhand der Untersuchungen für die beiden Erzeugendensysteme erkennt man, daß das Erzeugendensystem $b_{m-1}, \ldots, b_0$ mit $(b_{m-1}, \ldots, b_0) = BSUM_n$ verglichen mit den Atomen *im Mittel* zu geringeren Kosten führt. (Im Einzelfall können die Atome jedoch ein günstigeres Erzeugendensystem bilden.)

Zusammenfassung: Abschließend ist festzustellen, daß der Erfolg einer Synthese unter Ausnutzung von G–Symmetrie von folgenden (noch offenen) Punkten stark abhängig ist:

- Anzahl der Stufen und damit Länge der absteigenden Kette von Automorphismengruppen
- Konkrete Wahl der Automorphismengruppen in der absteigenden Kette und damit Wahl der zugehörigen Unteralgebren
- Wahl der Erzeugendensysteme der auftretenden Unteralgebren

Eine Möglichkeit, einen mehrstufigen Schaltkreis mit einer eingeschränkten Wahl der absteigenden Kette von Automorphismengruppen zu finden, liefert das Verfahren von Kim/Dietmeyer aus [KD91], das im nächsten Abschnitt kurz beschrieben wird.

3.3.2 Iterative Ausnutzung teilweiser Symmetrie

Kim/Dietmeyer beschränken sich in [KD91] auf die Ausnutzung von teilweiser Symmetrie in Variablenmengen λ mit $|\lambda| = 2$ oder $|\lambda| = 3$. Der Grund dafür liegt im wesentlichen in der Tatsache, daß solche Symmetrieeigenschaften noch relativ leicht feststellbar sind, während der Aufwand zur Feststellung teilweiser Symmetrie in größeren Variablenmengen λ (zumindest bei partiellen Funktionen, vgl. Abschnitt 3.4.2) schnell wächst. (Ist die Gesamtzahl der Variablen n, so gibt es $\binom{n}{n/2}$ mögliche Variablenmengen λ mit $|\lambda| = n/2$.)

Ist $f \in B_n$ zu realisieren und wird eine teilweise Symmetrie in

$$\lambda = \{x_{i_1}, \ldots, x_{i_k}\} \subseteq \{x_1, \ldots, x_n\}$$

festgestellt ($k \in \{2, 3\}$), so wird in einer 1. Stufe ein bestimmtes Erzeugendensystem der Unteralgebra aller in λ symmetrischen Funktionen berechnet. Das Erzeugendensystem ist gegeben durch

$$w_1^{(j)}(x_{i_1}, \ldots, x_{i_k}), \ldots, w_{j_i}^{(j)}(x_{i_1}, \ldots, x_{i_k}), x_{i_{k+1}}, \ldots, x_{i_n}.$$

Dabei sind $w_1^{(j)},\ldots,w_{j_i}^{(j)}$ Ausgängen sogenannter „Recoderfunktionen“ mit k Eingängen, außerdem gilt $\{x_{i_{k+1}},\ldots,x_{i_n}\} = \{x_1,\ldots,x_n\} \setminus \lambda$.

Die restlichen Stufen müssen dann eine „Restfunktion“ $f' \in B_{n-k+i_j}$ realisieren mit

$$f'(w_1^{(j)}(x_{i_1},\ldots,x_{i_k}),\ldots,w_{j_i}^{(j)}(x_{i_1},\ldots,x_{i_k}),x_{i_{k+1}},\ldots,x_{i_n}) = f(x_1,\ldots,x_n).$$

Für $|\lambda| = 2$ und $|\lambda| = 3$ sind jeweils 2 mögliche Recoderfunktionen vorgesehen: Für $|\lambda| = 2$ ist als 1. Möglichkeit ein Halbaddierer ($W^{(1)}$) vorgesehen, als 2. Möglichkeit eine Funktion $W^{(2)}$ mit 2 Ausgängen, wobei der 1. Ausgang 1 liefert, wenn mindestens ein Eingang 1 ist, und der 2. Ausgang 1 liefert, wenn beide Eingänge 1 sind.

Für $|\lambda| = 3$ ist als 1. Möglichkeit ein Volladdierer ($W^{(3)}$) verwendet, als 2. Möglichkeit eine Funktion $W^{(4)}$ mit 3 Ausgängen. Der 1. Ausgang liefert 1, wenn mindestens ein Eingang 1 ist, der 2. Ausgang liefert 1, wenn mindestens 2 Eingänge 1 sind und der 3. Ausgang liefert 1, wenn alle 3 Eingänge 1 sind.

$W^{(1)}$ – $W^{(4)}$ sind also nach folgender Vorschrift definiert:

$$\begin{array}{rrcl}
W^{(1)}(x,y): & w_1^{(1)}(x,y) & = & xy \\
 & w_2^{(1)}(x,y) & = & x \oplus y \\
W^{(2)}(x,y): & w_1^{(2)}(x,y) & = & x + y \\
 & w_2^{(2)}(x,y) & = & xy \\
W^{(3)}(x,y,z): & w_1^{(3)}(x,y,z) & = & xy + (x \oplus y)z \\
 & w_2^{(3)}(x,y,z) & = & x \oplus y \oplus z \\
W^{(4)}(x,y,z): & w_1^{(4)}(x,y,z) & = & x + y + z \\
 & w_2^{(4)}(x,y,z) & = & xy + xz + yz \\
 & w_3^{(4)}(x,y,z) & = & xyz
\end{array}$$

Der Vorteil, den man bei der Realisierung der „Restfunktion“ f' hat, besteht bei Verwendung von $W^{(3)}$ darin, daß f' einen Eingang weniger hat als f, bei Verwendung von $W^{(1)}, W^{(2)}$ oder $W^{(4)}$ besteht der Vorteil darin, daß f' mehr don't–care–Stellen hat. (Beispielweise können bei $W^{(1)}$ nie beide Ausgänge 0 liefern.)

Man erhält eine mehrstufige Darstellung, indem man das Verfahren nun iterativ auf f' anwendet.

Nachdem Symmetrie in λ festgestellt worden ist, hat man jeweils die Möglichkeit, sich zwischen 2 anwendbaren Recoderfunktionen zu entscheiden. Zur Verringerung der Kosten des zu synthetisierenden Schaltkreises benutzen Kim/Dietmeyer eine Schätzfunktion zur Abschätzung der Schaltkreiskosten unter Verwendung eines bestimmten Recoders und kommen so zu einer Entscheidung, welche der beiden möglichen Recoderfunktionen verwendet werden soll bzw. ob überhaupt eine Recoderfunktion verwendet werden soll, die die vorliegende Symmetrie ausnutzt.

Die hier angegebene Synthesemethode nutzt (disjunkte) Zerlegungen boolescher Funktionen aus. Eine ausführliche Darstellung der Anwendung von Zerlegungen bei der Logiksynthese ist in Kapitel 4 zu finden.

3.4 Feststellung von Symmetrien

In diesem Abschnitt soll ein kurzer Überblick darüber gegeben werden, wie man *Symmetrien in einer Teilmenge der Eingangsvariablen* finden kann. Dabei wird vorausgesetzt, daß die betreffenden Funktionen durch Funktionstabellen oder durch boolesche Polynome gegeben sind. Algorithmen zur Analyse solcher Symmetrieeigenschaften für Funktionen, die durch Funktionsgraphen (ROBDD's) repräsentiert sind, sind in [Möl92] angegeben.

3.4.1 Totale Funktionen

Eine boolesche Funktion $f \in B_n$ ist totalsymmetrisch, wenn f invariant gegenüber sämtlichen Permutationen der n Eingangsvariablen ist. Die Permutationen der n Eingangsvariablen bilden eine Gruppe, die symmetrische Gruppe $\mathbf{S}_n$. Will man testen, ob f totalsymmetrisch ist, so genügt es, die Invarianz für ein beliebiges Erzeugendensystem von $\mathbf{S}_n$ nachzuprüfen. Ein mögliches Erzeugendensystem besteht in dem „zyklischen Shift“ und einer beliebigen Transposition, z. B. aus den Abbildungen $\sigma_{1,2}$ mit $\sigma_{1,2}(\alpha_1, \alpha_2, \ldots, \alpha_n) = (\alpha_2, \alpha_1, \ldots, \alpha_n)$ und σ_{shift} mit $\sigma_{shift}(\alpha_1, \alpha_2, \ldots, \alpha_n) = (\alpha_2, \ldots, \alpha_n, \alpha_1)$. Zum Test, ob f totalsymmetrisch ist, genügt es also festzustellen, ob für alle $(x_1, \ldots, x_n) \in \{0,1\}^n$

$$f(x_1, \ldots, x_n) = f(x_2, x_1, x_3, \ldots, x_n) \quad \text{und}$$

$$f(x_1, \ldots, x_n) = f(x_2, \ldots, x_n, x_1).$$

(Diese Aussage wurde von Arnold/Harrison ([AH63]) und Povarov unabhängig gefunden.)

Ebenso läßt sich teilweise Symmetrie in einer Variablenmenge λ mit 2 Tests feststellen: Es genügt zu testen, ob eine Funktion invariant ist gegenüber einem zyklischen Shift der Variablen in λ und einer Transposition eines beliebigen Variablenpaares aus λ.

Der Vorteil bei der Behandlung von totalen Funktionen gegenüber partiellen Funktionen besteht darin, daß bei totalen Funktionen die Symmetrie in 2 Variablen eine Äquivalenzrelation ist. Ist eine Funktion f symmetrisch in den Variablenpaaren (x_i, x_j) und (x_j, x_k), so ist sie auch symmetrisch in (x_i, x_k) und folglich in der Variablenmenge $\{x_i, x_j, x_k\}$. Die Äquivalenzklassen dieser Relation stellen eine Partition auf den Eingangsvariablen dar. Die einzelnen Äquivalenzklassen sind die größtmöglichen Variablenmengen, in denen f symmetrisch ist. Angenommen, die Symmetrierelation liefert auf den Eingangsvariablen die

Partition $\{\mu_1, \ldots, \mu_k\}$. Dann ist f genau dann in einer Variablenmenge λ symmetrisch, wenn $\lambda \subseteq \mu_i$ für $1 \leq i \leq k$.

Folglich genügen $\binom{n}{2} = 1/2(n^2 - n)$ Symmetrietests, um für alle Teilmengen λ der Eingangsvariablen entscheiden zu können, ob f symmetrisch in λ ist: Man beginnt mit einer Partition der Eingangsvariablen, bei der jede Variable in einer eigenen Menge ist. Nun führt man für sämtliche Variablenpaare einen Symmetrietest durch. Stellt man Symmetrie in (x_i, x_j) fest, so vereinigt man die beiden Mengen der Partition, die x_i und x_j enthalten. (Zur praktischen Durchführung bietet sich hierzu eine Union–Find–Datenstruktur an.) Am Ende erhält man die oben angesprochene Äquivalenzklasseneinteilung, z.B. $\{\mu_1, \ldots, \mu_k\}$. Will man nun entscheiden, ob f symmetrisch in einer beliebigen Teilmenge λ der Eingangsvariablen ist, so genügt es, zu testen, ob $\lambda \subseteq \mu_i$ für ein $i \in \{1, \ldots, k\}$.

3.4.2 Partielle Funktionen

Wie schon in Bemerkung 7 angedeutet, ist es häufig sinnvoll, partielle Funktionen $f \in S(D)$ auch dann als symmetrisch anzusehen, wenn es eine Erweiterung $f' \in S(D')$ von f gibt, die symmetrisch ist. Der Einfachheit halber soll in diesem Abschnitt die Aussage „f symmetrisch in λ" gleichbedeutend sein mit „es gibt eine Erweiterung von f, die symmetrisch in λ ist".

Das Problem, das hier im Gegensatz zu totalen Funktionen auftritt, besteht darin, daß die Symmetrie in 2 Variablen nun keine Äquivalenzrelation mehr darstellt. Betrachte hierzu eine Beispielfunktion f mit folgender Funktionstabelle:

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	0
0	0	1	0
0	1	0	$*$
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

(An der Stelle (001) ist f undefiniert.)
Man sieht leicht:

- f ist symmetrisch in (x_1, x_2)
- f ist symmetrisch in (x_2, x_3)
- f ist *nicht* symmetrisch in (x_1, x_3)

Da die Symmetrie in 2 Variablen keine Äquivalenzrelation darstellt, ist es schwierig, aus Symmetrien in kleineren Teilmengen der Eingangsvariablen Symmetrien in größeren Teilmengen herzuleiten.

Gibt man jedoch eine *einzelne* Variablenmenge λ vor, so ist es auch hier noch relativ einfach festzustellen, ob f symmetrisch ist in λ oder nicht.

$f \in S(D)$, $D \subseteq \{0,1\}^n$ ist genau dann symmetrisch in der Teilmenge λ der Eingangsvariablen, wenn für jede Permutation σ auf den Variablen aus λ gilt:

$$\sigma(ON(f)) \cap OFF(f) = \emptyset.$$

Genau dann, wenn diese Bedingung für alle Permutationen σ auf den Variablen aus λ gilt, gibt es eine Erweiterung von f, die symmetrisch in λ ist.

(Da die Permutationen auf den Variablen aus λ eine Gruppe bilden, kann man die Bedingung auch anders ausdrücken:
$f \in S(D)$, $D \subseteq \{0,1\}^n$ ist genau dann symmetrisch in der Teilmenge λ der Eingangsvariablen, wenn für alle Permutationen σ und τ auf den Variablen aus λ gilt:

$$\sigma(ON(f)) \cap \tau(OFF(f)) = \emptyset. \qquad (*) \quad)$$

In [KD91] ist ein Verfahren angegeben, mit dessen Hilfe Bedingung (*) relativ effizient getestet werden kann, falls die partielle Funktion f durch *on*– und off–Arrays gegeben ist.

Arrays sind spezielle Darstellungen von booleschen Polynomen. Ein Array ist eine Menge von *Kuben*, wobei ein Kubus ein Monom repräsentiert. Ein Kubus ist ein n–Tupel über $\{0, 1, x\}$. Steht an Position i des n–Tupels

- eine 0, so tritt Variable x_i in dem entsprechenden Monom negiert auf,
- eine 1, so tritt Variable x_i in dem entsprechenden Monom nicht–negiert auf,
- ein x, so tritt Variable x_i in dem entsprechenden Monom nicht auf,

(Bsp.: $0x1x1$ steht für $\overline{x_1}x_3x_5$.)
Der Schnitt zweier Kuben ist definiert wie der Schnitt der zugehörigen Monome. Also ist der Schnitt zweier Kuben genau dann leer, wenn an einer festen Position in dem einen Kubus eine 0 vorkommt und in dem anderen eine 1. Der Schnitt zweier Arrays A_1 und A_2 ist folglich die Vereinigung aller paarweisen Schnitte von Kuben aus A_1 mit Kuben aus A_2.

Eine partielle Funktion f kann vollständig durch Angabe ihres ON–Arrays $\boldsymbol{on_f}$ und ihres OFF–Arrays $\boldsymbol{off_f}$ spezifiziert werden. ON–Array und OFF–Array definieren die ON–Menge und die OFF–Menge von f.

Um einen Symmetrietest effizient ausführen zu können, wird in [KD91] eine Normalisierung von Kuben benutzt. Dabei ist die *Normalisierung* $N_\lambda(c)$ eines Kubus c hinsichtlich der Variablenmenge λ ein Kubus, der aus c hervorgeht durch Umordnen der Komponenten, die Variablen in λ entsprechen, nach der Reihenfolge 1, x, 0. N_λ wird erweitert auf Arrays.

(Bsp.: $\begin{array}{l} A = \{01x00, x0111, x0x10\},\ \lambda = \{x_1, x_2, x_3\} \\ \Rightarrow\ N_\lambda(A) = \{1x000, 1x011, xx010\}\ ) \end{array}$

Sei $n^1_\lambda(c)$ die Anzahl von Einsen im λ–Teil von c, analog $n^0_\lambda(c)$ und $n^x_\lambda(c)$. Seien c und d Kuben mit der Eigenschaft, daß an keiner Position, die einer Variablen entspricht, die nicht aus λ ist, bei c eine 0 steht und bei d eine 1 (oder umgekehrt). Dann ist der Schnitt der normalisierten Kuben $N_\lambda(c)$ und $N_\lambda(d)$ genau dann leer, wenn

$$n^1_\lambda(c) + n^x_\lambda(c) < n^1_\lambda(d) \text{ oder}$$

$$n^1_\lambda(d) + n^x_\lambda(d) < n^1_\lambda(c).$$

Wenn man Lemma 3.2 beachtet, so ergibt sich ein einfacher Test auf Symmetrie in λ:

Lemma 3.2 *Sei $P_\lambda(c)$ die Menge aller Kuben, die man aus c erhält, indem man Permutationen auf dem λ–Teil anwendet. Es gilt:*

$$\forall c' \in P_\lambda(c),\ d' \in P_\lambda(d) \text{ gilt } c' \cap d' = \emptyset \Leftrightarrow N_\lambda(c) \cap N_\lambda(d) = \emptyset$$

Beweis:

„ $\Longrightarrow$ “ : klar, da $N_\lambda(c) \in P_\lambda(c)$, $N_\lambda(d) \in P_\lambda(d)$.

„ $\Longleftarrow$ “ : Angenommen, es gibt

$$c' \in P_\lambda(c),\ d' \in P_\lambda(d) \text{ mit } c' \cap d' \neq \emptyset.$$

Dann gibt es keine feste Position, an der in c' eine 0 vorkommt und in d' eine 1 oder umgekehrt.
Sei $w^1_\lambda(c')$ die Anzahl der Einsen im λ–Teil von c', analog $w^0_\lambda(c')$ und $w^x_\lambda(c')$.
Dann gilt:

$$w^1_\lambda(c') + w^x_\lambda(c') \geq w^1_\lambda(d')$$

(Denn zu jeder 1 in d' kommt in c' an der gleichen Position eine 1 oder ein x vor.) Analog:

$$w^1_\lambda(d') + w^x_\lambda(d') \geq w^1_\lambda(c')$$

Es gilt:

$$w^1_\lambda(c') = w^1_\lambda(c) = n^1_\lambda(c)$$

$$w^1_\lambda(d') = w^1_\lambda(d) = n^1_\lambda(d)$$

(Entsprechend für n^0_λ und n^x_λ.) Also gilt auch

$$n^1_\lambda(c) + n^x_\lambda(c) \geq n^1_\lambda(d) \text{ und}$$

$$n^1_\lambda(d) + n^x_\lambda(d) \geq n^1_\lambda(c)$$

und somit

$$N_\lambda(c) \cap N_\lambda(d) \neq \emptyset$$

□

Ist f durch on_f und off_f repräsentiert, so ergibt sich aus Lemma 3.2 ein einfacher Test für Bedingung (*):

Satz 3.3 *Ist $f \in S(D)$ durch on_f und off_f repräsentiert, so ist f symmetrisch in λ genau dann wenn*

$$N_\lambda(on_f) \cap N_\lambda(off_f) = \emptyset$$

Will man Symmetrie bei der Synthese ausnutzen, so sucht man meist nicht nach Symmetrie in einer *bestimmten* Variablenmenge, sondern nach Symmetrie in einer *möglichst großen* Variablenmenge. Die im vorigen Abschnitt angegebene Partition der Eingangsvariablen liefert bei totalen Funktionen in dieser Hinsicht eine hinreichende Antwort.

Bei partiellen Funktionen hat man mit Satz 3.3 einen relativ einfachen Test auf Symmetrie in einer einzelnen Variablenmenge λ. Die Problematik besteht allerdings darin, daß es bei einer festen Mächtigkeit von λ sehr viele Teilmengen der Eingangsvariablen mit dieser Mächtigkeit gibt (bei $|\lambda| = n/2$ z. B. $\binom{n}{n/2} = \Theta(\frac{2^n}{\sqrt{n}})$), so daß der Aufwand für eine solche Suche nach Symmetrie oft unvertretbar hoch werden kann.

Kapitel 4

Zerlegungen

Die Durchführung von disjunkten Zerlegungen bei der Logiksynthese wurde erstmals in Arbeiten von Ashenhurst ([Ash59]), Curtis ([Cur61]) und Karp ([Kar63]) in Betracht gezogen. Im vorliegenden Kapitel wird genauer untersucht, wie man aus Zerlegbarkeitseigenschaften boolescher Funktionen Nutzen im Hinblick auf die Realisierung dieser Funktionen ziehen kann.

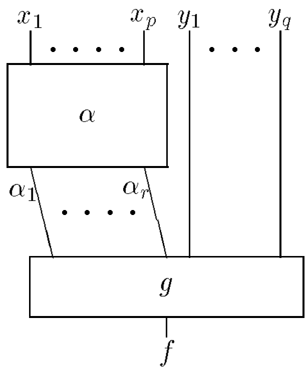
4.1 Zerlegungen von Funktionen mit 1 Ausgang

Zunächst werden 2 Arten von (disjunkten) Zerlegungen definiert: die einseitige und die zweiseitige Zerlegung.

Definition 4.1 (Einseitige Zerlegung) *Eine* **einseitige (disjunkte) Zerlegung** *einer booleschen Funktion $f \in B_n$ mit den Eingangsvariablen $x_1, \ldots, x_p$, $y_1, \ldots, y_q$ $(p + q = n)$ $(p, q > 0)$ hinsichtlich einer Variablenteilmenge $\{x_1, \ldots, x_p\}$ ist eine Darstellung von f in der Form*

$$f(x_1, \ldots, x_p, y_1, \ldots, y_q) = g(\alpha_1(x_1, \ldots, x_p), \ldots, \alpha_r(x_1, \ldots, x_p), y_1, \ldots, y_q)$$

Folgende Abbildung illustriert die obige Definition:

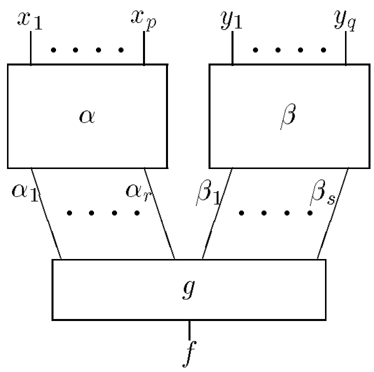


Die Funktionen $\alpha_1, \ldots, \alpha_r$ berechnen also auf den Eingangsvariablen $x_1, \ldots, x_p$ ein „Zwischenergebnis" und die Funktion g berechnet aus diesem Zwischenergebnis und den restlichen Eingangsvariablen $y_1, \ldots, y_q$ den endgültige Funktionswert von f auf den Eingangsvariablen $x_1, \ldots, x_p$, $y_1, \ldots, y_q$.

Definition 4.2 (Zweiseitige Zerlegung)
Eine **zweiseitige (disjunkte) Zerlegung** *einer booleschen Funktion* $f \in B_n$ *mit den Eingangsvariablen* $x_1, \ldots, x_p$, $y_1, \ldots, y_q$ $(p+q = n)$ $(p, q > 0)$ *hinsichtlich einer Variablenaufteilung* $\{\{x_1, \ldots, x_p\}, \{y_1, \ldots, y_q\}\}$ *ist eine Darstellung von* f *in der Form*

$$f(x_1, \ldots, x_p, y_1, \ldots, y_q) = \\ g(\alpha_1(x_1, \ldots, x_p), \ldots, \alpha_r(x_1, \ldots, x_p), \beta_1(y_1, \ldots, y_q), \ldots, \beta_s(y_1, \ldots, y_q))$$

Die zweiseitige Zerlegung wird im folgenden Bild veranschaulicht:



Hier berechnen also die Funktionen $\alpha_1, \ldots, \alpha_r$ auf den Eingangsvariablen $x_1, \ldots,$ x_p ein vorläufiges Ergebnis und die Funktionen $\beta_1, \ldots, \beta_s$ auf den Eingangsvariablen $y_1, \ldots, y_q$ ein vorläufiges Ergebnis. Die Funktion g berechnet aus diesen beiden Zwischenergebnissen den endgültige Funktionswert von f.

Bezeichnung 11 *Die Funktionen* α_i $(1 \leq i \leq r)$ *(und* β_j $(1 \leq j \leq s)$ *im Falle zweiseitiger Zerlegung) werden als* **Zerlegungsfunktionen** *bezeichnet, die Funktion* g *wird als* **Zusammensetzungsfunktion** *bezeichnet.*

Will man Zerlegungen bei der Logiksynthese ausnutzen, so ist es günstig, nach Zerlegungen mit möglichst geringer Anzahl von Zerlegungsfunktionen zu suchen. Dies hat folgende Vorteile:

- Ist die Anzahl der Zerlegungsfunktionen gering, so hat man bei rekursiver Anwendung des Verfahrens weniger Funktionen zu realisieren. Es besteht die Hoffnung, daß die Gesamtkosten des resultierenden Schaltkreises somit möglichst gering bleiben.

- Je weniger Zerlegungsfunktionen bei einer Zerlegung auftreten, desto weniger Eingänge hat die Zusammensetzungsfunktion g. Wenn g wenig Eingänge hat, dann kann man hoffen, daß die Komplexität von g auch gering ist.

- Aus der Netzliste (bzw. dem Ω–Schaltkreis), die die Logiksynthese liefert, wird das Layout eines Schaltkreises bestimmt, der die gesuchte Funktion realisiert. Häufig beobachtet man Schaltungen, bei denen die Fläche des Layouts nicht unbedingt durch die Anzahl der zu realisierenden Gatter bestimmt wird, sondern durch die Anzahl der zu realisierenden globalen Verbindungen. Eine Zerlegung mit wenig Zerlegungsfunktionen legt gerade eine Realisierung mit wenig globalen Verbindungen nahe.

In diesem Sinne sind solche Zerlegungen interessant, bei denen die Anzahl der Zerlegungsfunktionen kleiner ist als die Anzahl der Eingangsvariablen (im Fall zweiseitiger Zerlegungen) bzw. bei denen die Anzahl der Zerlegungsfunktionen kleiner ist als die Anzahl der Eingangsvariablen, von denen die Zerlegungsfunktionen abhängen (im Fall einseitiger Zerlegungen). Man bezeichnet diese Zerlegungen als *nichttrivial*.

Definition 4.3 (Nichttriviale Zerlegung)

- *Eine einseitige Zerlegung*

  $$f(x_1,\ldots,x_p,y_1,\ldots,y_q) = g(\alpha_1(x_1,\ldots,x_p),\ldots,\alpha_r(x_1,\ldots,x_p),y_1,\ldots,y_q)$$

  heißt **nichttrivial**, *wenn* $r < p$.

- *Eine zweiseitige Zerlegung*

  $$\begin{aligned} &f(x_1,\ldots,x_p,y_1,\ldots,y_q) = \\ &\qquad g(\alpha_1(x_1,\ldots,x_p),\ldots,\alpha_r(x_1,\ldots,x_p),\beta_1(y_1,\ldots,y_q),\ldots,\beta_s(y_1,\ldots,y_q)) \end{aligned}$$

  heißt **nichttrivial**, *wenn* $r+s < p+q = n$.

Verwendet man nur nichttriviale Zerlegungen von f und wendet man das Verfahren rekursiv an, um eine Realisierung für die Zerlegungsfunktionen und die Zusammensetzungsfunktion zu finden, so haben alle Funktionen, die auf der nächsten Rekursionsstufe behandelt werden, eine geringere Anzahl an Eingangsvariablen als f.

Bemerkung 9 *Eine einseitige Zerlegung läßt sich auch als eine spezielle zweiseitige Zerlegung betrachten, bei der gerade gilt* $\beta_i(y_1,\ldots,y_q) = y_1$ *für* $1 \le i \le q$. *Allerdings ist die Anzahl dieser Zerlegungsfunktionen* β_i *bei der einseitigen Zerlegung nicht unbedingt minimal.*

Umgekehrt läßt sich jede zweiseitige Zerlegung als eine Folge von 2 einseitigen Zerlegungen betrachten[1].

4.1.1 Minimale Anzahl von Zerlegungsfunktionen

Es ist interessant, Zerlegungen mit einer möglichst geringen Anzahl von Zerlegungsfunktionen zu betrachten. Gesucht ist nun eine Möglichkeit, bei einer vorgegebenen Variablenteilmenge X eine Zerlegung hinsichtlich X mit minimaler Anzahl von Zerlegungsfunktionen zu finden.

Als Hilfsmittel dazu wird der Begriff der *Zerlegungsmatrix* definiert.

Definition 4.4 (Zerlegungsmatrix) *Die Zerlegungungsmatrix einer (partiellen) Funktion $f \in S(D)$ mit den Eingangsvariablen $x_1, \ldots, x_p$, $y_1, \ldots, y_q$ hinsichtlich der Variablenteilmenge $\{x_1, \ldots, x_p\}$ ist definiert als $(2^p \times 2^q)$–Matrix Z, wobei für alle i, j mit $0 \leq i \leq 2^p - 1$ und $0 \leq j \leq 2^q - 1$ gilt:*

$$Z_{ij} = \begin{cases} f(bin_p(i), bin_q(j)), & falls\ f\ definiert\ an\ dieser\ Stelle \\ *, & falls\ f\ undefiniert\ an\ dieser\ Stelle \end{cases}$$

(Hierbei liefert $bin_p(i)$ ein p–Tupel über $\{0,1\}$, das der Binärdarstellung von i entspricht, $bin_q(j)$ liefert ein q–Tupel über $\{0,1\}$, das eine Binärdarstellung von j darstellt.)

In diesem Abschnitt ist zunächst nur von Zerlegungsmatrizen zu totalen Funktionen die Rede. Für totale Funktionen werden folgende Bezeichnungen definiert:

Bezeichnung 12 *Sei $f \in B_n$ mit den Eingangsvariablen $x_1, \ldots, x_p$, $y_1, \ldots, y_q$. Sei Z die Zerlegungsmatrix von f hinsichtlich der Variablenteilmenge $X := \{x_1, \ldots, x_p\}$.*

Dann wird die Anzahl der verschiedenen Zeilen von Z mit $vz(X, f)$ bezeichnet, die Anzahl der verschiedenen Spalten von Z mit $vs(X, f)$.

Bemerkung 10 *Ist Z Zerlegungsmatrix der (totalen) Funktion f hinsichtlich der Variablenteilmenge X, so liefert die Gleichheit auf den Zeilen von Z eine Äquivalenzklasseneinteilung von $\{0,1\}^{|X|}$, wenn man für $x^{(1)}, x^{(2)} \in \{0,1\}^{|X|}$ definiert: $x^{(1)} \sim x^{(2)}$ genau dann, wenn die Zeilen mit den Indizes $int(x^{(1)})$*[†] *und $int(x^{(2)})$ gleich sind.*

Bevor die minimale Anzahl von Zerlegungsfunktionen bei einer Zerlegung von f hinsichtlich X bestimmt wird, sollen die Begriffe anhand eines Beispiels verdeutlicht werden:

[1] Durch eine Folge von 2 einseitigen Zerlegungen lassen sich aber auch allgemeinere zweiseitige Zerlegungen darstellen, bei denen die beiden Mengen der Variablenaufteilung nicht *disjunkt* sind, sondern einige Variablen gemeinsam enthalten.

[†] $int(x)$ gibt den dezimalen Wert der Binärzahl x an.

x_3	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
x_4	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
y_3	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
y_4	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
$x_1x_2y_1y_2$																
0 0 0 0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1
0 0 0 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0 0 1 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0 0 1 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0 1 0 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0 1 0 1	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1
0 1 1 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0 1 1 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1 0 0 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1 0 0 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1 0 1 0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1
1 0 1 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1 1 0 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1 1 0 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1 1 1 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1 1 1 1	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1

Abbildung 4.1: Zerlegungsmatrix zu vgl_4

Beispiel 1:
Sei die Funktion vgl_4 definiert als

$$vgl_4(x_1, x_2, x_3, x_4, y_1, y_2, y_3, y_4) = \begin{cases} 1, & falls\ (x_1, x_2, x_3, x_4) = (y_1, y_2, y_3, y_4) \\ 0, & falls\ (x_1, x_2, x_3, x_4) \neq (y_1, y_2, y_3, y_4). \end{cases}$$

vgl_4 vergleicht also die Binärzahl gebildet aus den 4 ersten Eingangsvariablen mit der Binärzahl gebildet aus den 4 letzten Eingangsvariablen.

Eine mögliche zweiseitige Zerlegung hinsichtlich $\{\{x_1, x_2, y_1, y_2\}, \{x_3, x_4, y_3, y_4\}\}$ ist

$$vgl_4(x_1, x_2, x_3, x_4, y_1, y_2, y_3, y_4) = and_2(vgl_2(x_1, x_2, y_1, y_2), vgl_2(x_3, x_4, y_3, y_4))$$

$$\text{mit } vgl_2(x_1, x_2, y_1, y_2) = \begin{cases} 1, & falls\ (x_1, x_2) = (y_1, y_2) \\ 0, & falls\ (x_1, x_2) \neq (y_1, y_2) \end{cases}$$

Auf den Variablen $\{x_1, x_2, y_1, y_2\}$ wird also berechnet, ob die ersten beiden Bit der zu vergleichenden Binärzahlen übereinstimmen oder nicht (2 Informationen, kodiert durch 1 Bit der Zerlegungsfunktion), auf den Variablen $\{x_3, x_4, y_3, y_4\}$ wird entsprechend berechnet, ob die beiden letzten Bit der zu vergleichenden Zahlen übereinstimmen. Die Zusammensetzungsfunktion (and_2) setzt die beiden Zwischenergebnisse zum endgültigen Funktionswert zusammen.

Die Anzahl der Informationen, die durch die Zerlegungsfunktionen auf den Variablen der beiden Mengen berechnet werden müssen, kann man auch aus der Zerlegungsmatrix in Bild 4.1 ablesen.

Die Zerlegungsmatrix hat 2 verschiedene Zeilen. Die 0–Zeilen geben jeweils an, daß $(x_1, x_2) \neq (y_1, y_2)$, die anderen Zeilen geben an, daß $(x_1, x_2) = (y_1, y_2)$. (Analog für die beiden verschiedenen Spalten.)

Zur minimalen Abzahl von Zerlegungsfunktionen gilt allgemein:

Satz 4.1 *Sei $f \in B_n$ eine Funktion mit den Eingangsvariablen $x_1, \ldots, x_p, y_1, \ldots, y_q$. Dann gibt es genau dann eine einseitige Zerlegung von f hinsichtlich der Variablenteilmenge $X = \{x_1, \ldots, x_p\}$ der Form*

$$f(x_1, \ldots, x_p, y_1, \ldots, y_q) = g(\alpha_1(x_1, \ldots, x_p), \ldots, \alpha_r(x_1, \ldots, x_p), y_1, \ldots, y_q),$$

wenn

$$r \geq \log(vz(X, f))$$

Beweis:

„$\Longleftarrow$": Sei eine Zerlegung gegeben durch

$$\begin{aligned} &f(x_1, \ldots, x_p, y_1, \ldots, y_q) = \\ &\quad g(\alpha_1(x_1, \ldots, x_p), \ldots, \alpha_r(x_1, \ldots, x_p), y_1, \ldots, y_q) \end{aligned}$$

Dann kann die Funktion

$$\alpha = (\alpha_1, \ldots, \alpha_r) \in B_{p,r}$$

höchstens 2^r verschiedene Funktionswerte annehmen.
Nehme nun an, daß

$$r < \log(vz(X, f), d.h.\ 2^r < vz(X, f).$$

Dann muß es $x^{(1)} = (x_1^{(1)}, \ldots, x_p^{(1)})$ und $x^{(2)} = (x_1^{(2)}, \ldots, x_p^{(2)})$ geben, so daß

$$\alpha(x^{(1)}) = \alpha(x^{(2)}),$$

aber die zugehörigen Zerlegungsmatrixzeilen verschieden sind.
Wenn die zugehörigen Zerlegungsmatrixzeilen verschieden sind, so bedeutet dies, daß es $y^{(0)} = (y_1^{(0)}, \ldots, y_q^{(0)})$ gibt mit

$$f(x_1^{(1)}, \ldots, x_p^{(1)}, y_1^{(0)}, \ldots, y_q^{(0)}) \neq f(x_1^{(2)}, \ldots, x_p^{(2)}, y_1^{(0)}, \ldots, y_q^{(0)}).$$

Trotzdem gilt unabhängig von der Wahl von g:

$$\begin{aligned} &g(\alpha_1(x_1^{(1)}, \ldots, x_p^{(1)}), \ldots, \alpha_r(x_1^{(1)}, \ldots, x_p^{(1)}), y_1^{(0)}, \ldots, y_q^{(0)}) = \\ &\quad g(\alpha_1(x_1^{(2)}, \ldots, x_p^{(2)}), \ldots, \alpha_r(x_1^{(2)}, \ldots, x_p^{(2)}), y_1^{(0)}, \ldots, y_q^{(0)}), \end{aligned}$$

da
$$\alpha(x^{(1)}) = \alpha(x^{(2)}).$$
$\Longrightarrow$ Widerspruch. Es muß also
$$r \geq \log(vz(X, f)$$
gelten.

„$\Longrightarrow$“: Sei
$$r \geq \log(vz(X, f)), d.h.\ 2^r \geq vz(X, f).$$
Dann kann man $\alpha \in B_{p,r}$ definieren, so daß
$$\alpha(x^{(1)}) \neq \alpha(x^{(2)})$$
für alle $x^{(1)}, x^{(2)} \in \{0,1\}^p$, für die die zugehörigen Zerlegungsmatrixzeilen verschieden sind.
Dann kann man die Zusammensetzungsfunktion g definieren nach der Vorschrift (*)
$$g(\alpha_1(x_1, \ldots, x_p), \ldots, \alpha_r(x_1, \ldots, x_p), y_1, \ldots, y_q) = f(x_1, \ldots, x_p, y_1, \ldots, y_q)$$
für alle $(x_1, \ldots, x_p) \in \{0,1\}^p$.
Falls $(a_1, \ldots, a_r)$ nicht im Bild von α auftritt, so ist $g(a_1, \ldots, a_r, y_1, \ldots, y_q)$ undefiniert bzw. frei wählbar.

g ist nach Vorschrift (*) wohldefiniert.
Wäre g nicht wohldefiniert, dann gäbe es $x^{(1)}$ und $x^{(2)}$ aus $\{0,1\}^p$ mit
$$\alpha(x^{(1)}) = \alpha(x^{(2)})$$
und ein $y^{(0)} \in \{0,1\}^q$ mit
$$f(x^{(1)}, y^{(0)}) \neq f(x^{(2)}, y^{(0)}).$$
Dann wären aber die Zerlegungsmatrixzeilen, die zu $x^{(1)}$ und $x^{(2)}$ gehören verschieden und damit nach Definition von α
$$\alpha(x^{(1)}) \neq \alpha(x^{(2)}).$$
$\Longrightarrow$ Widerspruch. g muß wohldefiniert sein.
Man hat also eine Zerlegung gefunden.

□

Anhand des Beweises zu Satz 4.1 ergibt sich folgende Aussage:

Korollar 4.1 *Sei $f \in B_n$ eine Funktion mit den Eingangsvariablen $x_1, \ldots, x_p$, $y_1, \ldots, y_q$, seien für $1 \leq i \leq r$ $\alpha_i \in B_p$ und sei Z die Zerlegungsmatrix von f hinsichtlich $X = \{x_1, \ldots, x_p\}$. Dann gibt es genau dann eine einseitige Zerlegung von f hinsichtlich der Variablenteilmenge X in der Form*

$$f(x_1, \ldots, x_p, y_1, \ldots, y_q) = g(\alpha_1(x_1, \ldots, x_p), \ldots, \alpha_r(x_1, \ldots, x_p), y_1, \ldots, y_q),$$

wenn die Funktion $\alpha = (\alpha_1, \ldots, \alpha_r)$ folgende Eigenschaft hat:
Falls es zu $x^{(1)}, x^{(2)} \in \{0,1\}^p$ ein $y^{(0)} \in \{0,1\}^q$ gibt mit

$$f(x^{(1)}, y^{(0)}) \neq f(x^{(2)}, y^{(0)})$$

(d.h. falls die Zeilen $int(x^{(1)})$ und $int(x^{(2)})$ von Z verschieden sind), dann ist

$$\alpha(x^{(1)}) \neq \alpha(x^{(2)}).$$

Entsprechend gilt für zweiseitige Zerlegungen:

Satz 4.2 *Sei $f \in B_n$ eine Funktion mit den Eingangsvariablen $x_1, \ldots, x_p$, y_1, $\ldots$, y_q. Sei $X = \{x_1, \ldots, x_p\}$ und $Y = \{y_1, \ldots, y_q\}$. Dann gibt es genau dann eine zweiseitige Zerlegung von f hinsichtlich der Variablenaufteilung $\{X, Y\}$ der Form*

$$\begin{aligned} &f(x_1, \ldots, x_p, y_1, \ldots, y_q) = \\ &\quad g(\alpha_1(x_1, \ldots, x_p), \ldots, \alpha_r(x_1, \ldots, x_p), \beta_1(y_1, \ldots, y_q), \ldots, \beta_s(y_1, \ldots, y_q)) \end{aligned}$$

wenn

$$r \geq \log(vz(X,f)) \text{ und } s \geq \log(vs(X,f)).$$

Beweis:

„$\Longleftarrow$“: Nimmt man

$$r < \log(vz(X,f)) \text{ oder } s < \log(vs(X,f))$$

an, so kann man analog zum Beweis des vorhergehenden Satzes einen Widerspruch herleiten.

„$\Longrightarrow$“: Sei

$$r \geq \log(vz(X,f)), d.h.\ 2^r \geq vz(X,f) \text{ und}$$
$$s \geq \log(vs(X,f)), d.h.\ 2^s \geq vs(X,f).$$

Dann kann man $\alpha \in B_{p,r}$ definieren, so daß

$$\alpha(x^{(1)}) \neq \alpha(x^{(2)})$$

für alle $x^{(1)}, x^{(2)} \in \{0,1\}^p$, für die die zugehörigen Zerlegungsmatrixzeilen verschieden sind.
Ebenso kann man $\beta \in B_{q,s}$ definieren, so daß

$$\beta(y^{(1)}) \neq \beta(y^{(2)})$$

für alle $y^{(1)}, y^{(2)} \in \{0,1\}^q$, für die die zugehörigen Zerlegungsmatrixspalten verschieden sind.
Dann kann man die Zusammensetzungsfunktion g definieren nach der Vorschrift (*)

$$g(\alpha_1(x_1,\ldots,x_p),\ldots,\alpha_r(x_1,\ldots,x_p),\beta_1(y_1,\ldots,y_q),\ldots,\beta_s(y_1,\ldots,y_q)) = f(x_1,\ldots,x_p,y_1,\ldots,y_q)$$

für alle $(x_1,\ldots,x_p,y_1,\ldots,y_q) \in \{0,1\}^n$.
Falls $(a_1,\ldots,a_r)$ nicht im Bild von α auftritt oder $(b_1,\ldots,b_s)$ nicht im Bild von β auftritt, so ist $g(a_1,\ldots,a_r,b_1,\ldots,b_s)$ undefiniert bzw. frei wählbar.

g ist nach Vorschrift (*) wohldefiniert.
Angenommen g wäre nicht wohldefiniert, dann gäbe es $(x^{(1)}, y^{(1)})$ und $(x^{(2)}, y^{(2)})$ aus $\{0,1\}^n$ mit

$$\alpha(x^{(1)}) = \alpha(x^{(2)}) \text{ und } \beta(y^{(1)}) = \beta(y^{(2)}) \qquad (**)$$

und

$$f(x^{(1)}, y^{(1)}) \neq f(x^{(2)}, y^{(2)}).$$

D. h. es würde gelten

$$f(x^{(1)}, y^{(1)}) = \epsilon \in \{0,1\} \text{ und}$$

$$f(x^{(2)}, y^{(2)}) = \overline{\epsilon}.$$

Es kann nun *nicht* sein, daß *sowohl* die zu $x^{(1)}$ und $x^{(2)}$ gehörigen Zeilen der Zerlegungsmatrix *als auch* die zu $y^{(1)}$ und $y^{(2)}$ gehörigen Spalten gleich sind:
Angenommen die Zeilen zu $x^{(1)}$ und $x^{(2)}$ sind gleich. Dann gilt

$$f(x^{(2)}, y^{(1)}) = f(x^{(1)}, y^{(1)}) = \epsilon.$$

Wegen

$$f(x^{(2)}, y^{(2)}) = \overline{\epsilon}, \text{ also}$$

$$f(x^{(2)}, y^{(1)}) \neq f(x^{(2)}, y^{(2)}),$$

sind dann die zu $y^{(1)}$ und $y^{(2)}$ gehörigen Spalten ungleich.
Analog folgt aus der Tatsache, daß die Spalten zu $y^{(1)}$ und $y^{(2)}$ gleich sind:

$$f(x^{(1)}, y^{(2)}) = f(x^{(1)}, y^{(1)}) = \epsilon.$$

und somit

$$f(x^{(1)}, y^{(2)}) \neq f(x^{(2)}, y^{(2)}) = \overline{\epsilon}.$$

Also sind die zu $x^{(1)}$ und $x^{(2)}$ gehörigen Zeilen ungleich.

Die zu $x^{(1)}$ und $x^{(2)}$ gehörigen Zeilen oder die zu $y^{(1)}$ und $y^{(2)}$ gehörigen Spalten sind ungleich.

Im ersten Fall gilt nach Definition von α

$$\alpha(x^{(1)}) \neq \alpha(x^{(2)}) \quad \Longrightarrow \text{ Widerspruch zu (**)},$$

im zweiten Fall gilt nach Definition von β

$$\beta(y^{(1)}) \neq \beta(y^{(2)}) \quad \Longrightarrow \text{ Widerspruch zu (**)}.$$

Die Annahme, daß g durch Vorschrift (*) nicht wohldefiniert ist, ist also falsch!

□

Im folgenden wird immer mit Zerlegungen gearbeitet, bei denen die Anzahl der Zerlegungsfunktionen minimal ist. Auch die Arbeiten von Ashenhurst ([Ash59]), Curtis ([Cur61]) und Karp ([Kar63]) benutzen solche Zerlegungen. In [HOI89] (vgl. Abschnitt 4.1.5) werden dagegen Zerlegungen verwendet, bei denen die Anzahl der Zerlegungsfunktionen nicht unbedingt minimal ist.

4.1.2 Auffinden geeigneter Zerlegungen

Wahl einer geeigneten Variablenteilmenge bzw. Variablenaufteilung

Die Anzahl der Zerlegungsfunktionen, die bei einer vorgegebenen Variablenaufteilung (bzw. Variablenteilmenge) mindestens benötigt werden, hängt stark von der gewählten Variablenaufteilung (Variablenteilmenge) ab.

So wird man z.B. bei der Funktion vgl_4 von Beispiel 1 aus einer zweiseitigen Zerlegung hinsichtlich $\{\{x_1, \ldots, x_4\}, \{y_1, \ldots, y_4\}\}$ keinen Nutzen ziehen können. Die beiden zu vergleichenden 4–Bit–Zahlen befinden sich in getrennten Variablenteilmengen. Durch die Zerlegungsfunktionen kann keine sinnvolle Vorberechnung durchgeführt werden, man benötigt 4 Zerlegungsfunktionen auf $\{x_1, \ldots, x_4\}$ und 4 Zerlegungsfunktionen auf $\{y_1, \ldots, y_4\}$. Eine mögliche Wahl der Zerlegungsfunktionen wäre

$$\alpha_i(x_1, \ldots, x_4) = x_i \text{ und } \beta_i(y_1, \ldots, y_4) = y_i$$

für $1 \leq i \leq 4$.

Diesen Sachverhalt kann man auch anhand der Zerlegungsmatrix zu dieser Zerlegung erkennen:

y_1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
y_2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
y_3	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
y_4	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
$x_1 x_2 x_3 x_4$																
0 0 0 0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0 0 0 1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0 0 1 0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0 0 1 1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0 1 0 0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0 1 0 1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0 1 1 0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0 1 1 1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
1 0 0 0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1 0 0 1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
1 0 1 0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
1 0 1 1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
1 1 0 0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
1 1 0 1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
1 1 1 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
1 1 1 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Die Zerlegungungsmatrix ist eine Einheitsmatrix mit 16 Zeilen und 16 Spalten. Also benötigt man 4 Zerlegungsfunktionen auf $\{x_1, \ldots, x_4\}$ und 4 Zerlegungsfunktionen auf $\{y_1, \ldots, y_4\}$.

Es wird also ein Verfahren benötigt, das in der Lage ist, „günstige“ Variablenaufteilungen (bzw. Variablenteilmengen), d.h. Variablenaufteilungen (Variablenteilmengen), die zu einer minimalen Anzahl von Zerlegungsfunktionen führen, zu bestimmen.

Ist bei einseitiger Zerlegung einer Funktion $f \in B_n$ die Mächtigkeit p der Variablenteilmenge, hinsichtlich derer eine Zerlegung durchgeführt werden soll, vorgegeben, so besteht ein mögliches Verfahren darin, für alle $\binom{n}{p}$ p–elementigen Teilmengen X die Anzahl $vz(X, f)$ der Zeilen der zugehörigen Zerlegungsmatrix zu bestimmen. Ausgewählt wird dann die Teilmenge X mit geringstem Wert $vz(X, f)$.

Die Zeilenzahl einer Zerlegungsmatrix hinsichtlich X kann z.B. bestimmt werden durch Sortieren der 2^p Zeilen der Länge 2^q ($q = n - p$) mit einem Sortieralgorithmus wie merge sort und daraufolgendes lineares Durchmustern der Zeilen. Die Laufzeit des Verfahrens wird bestimmt durch das Sortieren und beträgt

$$O(p 2^p 2^q) = O(p 2^n) = O(n 2^n).$$

Dieses Verfahren wird

$$\binom{n}{p} = O\left(\binom{n}{n/2}\right) = O\left(\frac{2^n}{\sqrt{n}}\right)$$

Mal durchgeführt, so daß die Gesamtlaufzeit zur Bestimmung einer geeigneten Variablenteilmenge $O(\sqrt{n} \cdot 2^{2n})$ beträgt. Drückt man die Laufzeit in der Länge der Eingabe $N = 2^n$ (für die Funktionstabelle bzw. die Zerlegungsmatrix) aus, so ergibt sich eine Laufzeit der Größenordnung $O(N^2\sqrt{\log N})$.

Ist bei zweiseitiger Zerlegung einer Funktion $f \in B_n$ die Mächtigkeit p *einer* der beiden Teilmengen der Variablenaufteilung, hinsichtlich derer eine Zerlegung durchgeführt werden soll, vorgegeben, so führt man zur Bestimmung einer Variablenaufteilung mit minimaler Anzahl der Zerlegungsfunktionen ein analoges Verfahren durch. Der einzige Unterschied besteht darin, daß hier zur Bestimmung von Zeilen– und Spaltenanzahl der entsprechenden Zerlegungsmatrizen sowohl Zeilen als auch Spalten sortiert werden. Die Gesamtlaufzeit beträgt auch hier $O(N^2\sqrt{\log N})$.

Nichttriviale Zerlegbarkeit

Aus den Sätzen 4.1 und 4.2 ergibt sich folgendes Korollar:

Korollar 4.2 *$f \in B_n$ ist genau dann nichttrivial hinsichtlich der Variablenteilmenge $X = \{x_1, \ldots, x_p\}$ zerlegbar, wenn*

$$vz(X, f) \leq 2^{p-1}.$$

$f \in B_n$ ist genau dann nichttrivial hinsichtlich der Variablenaufteilung $\{X, Y\}$ $= \{\{x_1, \ldots, x_p\}, \{y_1, \ldots, y_q\}\}$ zerlegbar, wenn

$$vz(X, f) \leq 2^{p-1} \;\; \textit{oder} \;\; vs(X, f) \leq 2^{q-1}.$$

Für die Suche nach geeigneten Zerlegungen wird noch folgendes Lemma formuliert:

Lemma 4.1 *Gibt es keine nichttriviale einseitige Zerlegung von $f \in B_n$ hinsichtlich der Variablenteilmenge $X = \{x_1, \ldots, x_p\}$, so gibt es auch keine nichtriviale einseitige Zerlegung hinsichtlich Teilmengen von X.*
bzw.:
Ist f nichttrivial zerlegbar hinsichtlich $X = \{x_1, \ldots, x_p\}$, so ist f nichttrivial zerlegbar hinsichtlich jeder Obermenge von X.

Beweis:
Sei $f \in B_n$ mit den Eingangsvariablen $x_1, \ldots, x_p$, $y_1, \ldots, y_q$ nichtrivial zerlegbar hinsichtlich $X = \{x_1, \ldots, x_p\}$, d.h.

$$vz(X, f) \leq 2^{p-1}$$

Es genügt nun zu zeigen, daß f hinsichtlich $X' = \{x_1, \ldots, x_p, y_1\}$ nichttrivial zerlegbar ist, d.h. daß

$$vz(X', f) \leq 2^p.$$

Sei Z die Zerlegungsmatrix hinsichtlich X, Z' die Zerlegungsmatrix hinsichtlich X'.

Sind in Z die Zeilen zu $x^{(1)} = (x_1^{(1)}, \ldots, x_p^{(1)})$ und $x^{(2)} = (x_1^{(2)}, \ldots, x_p^{(2)})$ gleich, so gilt

$$f(x^{(1)}, y^{(0)}) = f(x^{(2)}, y^{(0)})$$

für alle $y^{(0)} \in \{0,1\}^q$ und somit

$$f(x^{(1)}, y_1^{(1)}, y^{(0)\prime}) = f(x^{(2)}, y_1^{(1)}, y^{(0)\prime})$$

für alle $y_1^{(1)} \in \{0,1\}$, $y^{(0)\prime} \in \{0,1\}^{q-1}$.
Also sind in Z' die Zeilen zu

$$(x_1^{(1)}, \ldots, x_p^{(1)}, 0) \text{ und } (x_1^{(2)}, \ldots, x_p^{(2)}, 0)$$

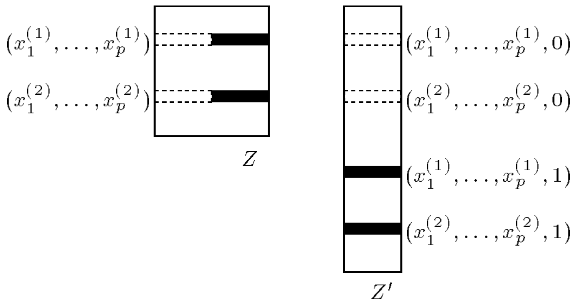
und die Zeilen zu

$$(x_1^{(1)}, \ldots, x_p^{(1)}, 1) \text{ und } (x_1^{(2)}, \ldots, x_p^{(2)}, 1)$$

gleich.
Z' kann also maximal doppelt soviele verschiedene Zeilen haben wie Z, d.h.

$$vz(X', f) \leq 2vz(X, f) \leq 2^p.$$

Der Zusammenhang zwischen den Zerlegungsmatrizen Z und Z' ist in folgender Abbildung dargestellt:



□

Nach Lemma 4.1 ist es leichter, nichttriviale Zerlegungen hinsichtlich großen Variablenteilmengen zu finden als nichttriviale Zerlegungen hinsichtlich kleinen Variablenteilmengen. Das nun folgende Lemma sagt aus, daß sogar jede Funktion $f \in B_n$ $(n \geq 4)$ nichttrivial zerlegbar ist, wenn man die Variablenteilmenge nur groß genug wählt.

Lemma 4.2 *Zu jeder Funktion $f \in B_n$ existiert eine nichttriviale einseitige Zerlegung*

$$f(x_1, \ldots, x_p, y_1, \ldots, y_q) = g(\alpha_1(x_1, \ldots, x_p), \ldots, \alpha_r(x_1, \ldots, x_p), y_1, \ldots, y_q),$$

falls $p \geq 2^q + 1$.

Beweis:

$$p \geq 2^q + 1 \quad \Longleftrightarrow \quad 2^{p-1} \geq 2^{2^q}$$

Die Zeilen der Zerlegungsmatrix hinsichtlich $X = \{x_1, \ldots, x_p\}$ haben Länge 2^q. Es gibt genau 2^{2^q} verschiedene Elemente in $\{0,1\}^{2^q}$.

$\Longrightarrow$ Es kann höchstens 2^{2^q} paarweise verschiedene Zeilen geben.

$\Longrightarrow$ $vz(X, f) \leq 2^{2^q}$

$\Longrightarrow$ $vz(X, f) \leq 2^{2^q} \leq 2^{p-1}$

$\Longrightarrow$ f nichttrivial zerlegbar hinsichtlich X.

□

Korollar 4.3 *Ist $f \in B_n$ mit $n \geq 4$, so gibt es immer eine nichttriviale einseitige Zerlegung.*

Bemerkung 11 *Für Funktionen $f \in B_n$ mit $n \geq 4$ und den Eingangsvariablen $\{x_1, \ldots, x_n\}$ bildet die sogenannte* Shannon–Zerlegung *eine spezielle nichttriviale Zerlegung hinsichtlich einer Variablenteilmenge $\{x_1, \ldots, x_{n-1}\}$ (denn es gilt $n - 1 \geq 2^1 + 1$).*

Die Shannon–Zerlegung von f ist definiert als

$$f(x_1, \ldots, x_n) = g(\alpha_1(x_1, \ldots, x_{n-1}), \alpha_2(x_1, \ldots, x_{n-1}), x_n),$$

wobei

$$\begin{aligned} \alpha_1(x_1, \ldots, x_{n-1}) &= f(x_1, \ldots, x_{n-1}, 0) \\ \alpha_2(x_1, \ldots, x_{n-1}) &= f(x_1, \ldots, x_{n-1}, 1) \\ g(a_1, a_2, x_n) &= a_1 \cdot \overline{x_n} + a_2 \cdot x_n. \end{aligned}$$

Bemerkung 12 *Aus der Definition nichttrivialer Zerlegbarkeit ergibt sich:*

Gibt es zu einer Funktion $f \in B_n$ mit der Menge von Eingangsvariablen $X \cup Y$ eine nichttriviale einseitige Zerlegung hinsichtlich der Variablenteilmenge X, so gibt es auch eine nichttriviale zweiseitige Zerlegung hinsichtlich der Variablenaufteilung $\{X, Y\}$.

Gibt es zu f eine nichttriviale zweiseitige Zerlegung hinsichtlich der Variablenaufteilung $\{X, Y\}$, so gibt es auch eine nichttriviale einseitige Zerlegung hinsichtlich X oder eine nichttriviale einseitige Zerlegung hinsichtlich Y.

Nach Lemma 4.2 existiert also auch zu jeder Funktion f eine nichttriviale zweiseitige Zerlegung, falls die Mächtigkeiten der beiden Mengen der Variablenaufteilung sich nur „ausreichend stark unterscheiden“.

Definition 4.5 (Gleichmächtige Zerlegungen)
Eine zweiseitige Zerlegung von $f \in B_n$ hinsichtlich der Variablenaufteilung $\{\{x_1, \ldots, x_p\}, \{y_1, \ldots, y_q\}\}$ heißt **gleichmächtig**, *falls*

$$p = \lfloor \frac{n}{2} \rfloor,\ q = \lceil \frac{n}{2} \rceil \text{ oder } p = \lceil \frac{n}{2} \rceil,\ q = \lfloor \frac{n}{2} \rfloor$$

Gleichmächtige Zerlegungen sind deshalb interessant, weil die rekursive Ausnutzung gleichmächtiger Zerlegungen mit minimaler Anzahl von Zerlegungsfunktionen im allgemeinen zu baumartigen Realisierungen mit geringer Tiefe führt.

Will man nun eine Logiksynthese unter Ausnutzung gleichmächtiger Zerlegungen durchführen und stellt es sich heraus, daß es keine nichttriviale gleichmächtige Zerlegung gibt, so ist es nach Lemma 4.2 durchaus sinnvoll, einen Zerlegungsschritt einzuschieben, bei dem die Aufteilung der Variablen in zwei verschiedenmächtige Mengen erfolgt. Im nächsten Rekursionsschritt können dann wieder gleichmächtige Zerlegungen möglich sein. Dies soll anhand eines Beispiels verdeutlicht werden:

Beispiel 2:
Gegeben sei eine Funktion $f \in B_5$, die definiert ist durch

$$f(x_0, \ldots, x_4) = \overline{x_0} \cdot (x_1 \oplus x_2) \cdot (x_3 \oplus x_4) \ +\ x_0 \cdot (x_1 + x_3) \cdot (x_2 + x_4)$$

für alle $(x_0, \ldots, x_4) \in \{0,1\}^5$.

Man sieht leicht (z.B. durch Betrachtung aller Zerlegungsmatrizen zu den $\binom{5}{3}$ möglichen gleichmäßigen Variablenaufteilungen), daß es zu f keine nichtriviale gleichmäßige Zerlegung gibt.

Führt man jedoch eine Shannon–Zerlegung durch, so sind die erhaltenen Zerlegungsfunktionen nichtrivial gleichmäßig zerlegbar:

$$f(x_0, \ldots, x_4) = \overline{x_0} \cdot \alpha_1(x_1, \ldots, x_4) \ +\ x_0 \cdot \alpha_2(x_1, \ldots, x_4)$$

mit den Zerlegungsfunktionen α_1 und α_2 mit

$$\alpha_1(x_1, \ldots, x_4) = (x_1 \oplus x_2) \cdot (x_3 \oplus x_4) \text{ und}$$

$$\alpha_2(x_1, \ldots, x_4) = (x_1 + x_3) \cdot (x_2 + x_4)$$

Zu α_1 gibt es eine nichtriviale gleichmäßige Zerlegung hinsichtlich der Variablenaufteilung $\{\{x_1, x_2\}, \{x_3, x_4\}\}$ und zu α_2 gibt es eine nichtriviale gleichmäßige Zerlegung hinsichtlich der Variablenaufteilung $\{\{x_1, x_3\}, \{x_2, x_4\}\}$. Der resultierende Schaltkreis ist in Abbildung 4.2 angegeben.

Verbesserungen des Verfahrens zur Wahl einer geeigneten Variablenaufteilung

Im Abschnitt 4.1.2 wurde ein Verfahren angegeben, das für Zerlegungen einer Funktion $f \in B_n$ in Laufzeit $O(N^2\sqrt{\log N})$ eine geeignete Variablenaufteilung

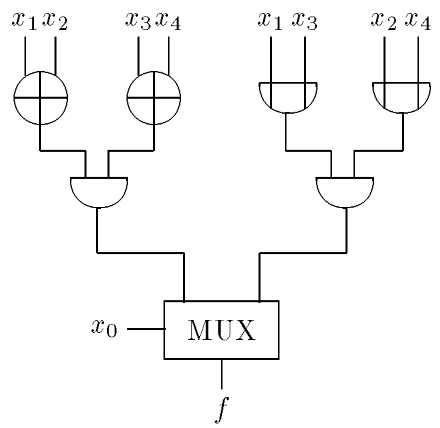


Abbildung 4.2: Schaltkreis zu Beispielfunktion f

bestimmt, wenn die Mächtigkeit einer der beiden Teilmengen der Variablenaufteilung vorgegeben ist. ($N = 2^n$ gibt hierbei die Größe der Funktionstabelle an.)

Da die Größe der Funktionstabellen mit wachsendem n allerdings rasch zunimmt, ist man daran interessiert, schnellere Verfahren zu finden.

Eine Möglichkeit besteht darin, nicht *alle* Variablenaufteilungen durchzuprobieren, sondern ein Verfahren der iterativen Verbesserung anzuwenden. Man beginnt mit einer gewissen Variablenaufteilung $\{X, Y\}$ und bestimmt dazu Zeilen– und Spaltenanzahl der Zerlegungsmatrix. Dann geht man durch Austausch zweier Variablen aus X und Y zu einer neuen Variablenaufteilung über. Denkbar ist in diesem Zusammenhang der Einsatz eines greedy–Algorithmus oder – wenn man die Möglichkeit haben will, aus lokalen Minima wieder herauszukommen – eines Kernighan&Lin–Algorithmus bzw. eines Simulated Annealing–Algorithmus.

Andere Möglichkeiten, die Laufzeit zu verbessern, ergeben sich, wenn man die Funktion f nicht durch ihre Funktionstabelle repräsentiert, sondern durch eine andere Darstellung, die evtl. kürzer ist als die Funktionstabelle.

Darstellung durch Polynome In [HOI92] wird dazu ein Verfahren vorgeschlagen, das auf booleschen Polynomen als Repräsentationsmöglichkeit boolescher Funktionen arbeitet. Das Verfahren berechnet bei Eingabe eines Polynoms zu einer Funktion $f \in B_n$, eines Polynoms zu $\overline{f}$ und einer Variablenteilmenge X die Anzahl $vz(X, f)$ der Zeilen der Zerlegungsmatrix von f hinsichtlich X (ohne die Zerlegungsmatrix explizit aufzustellen).

Sei $f \in B_n$ mit der Eingangsvariablenmenge $X \cup Y$ gegeben. Es soll eine Zerlegung hinsichtlich X durchgeführt werden. f werde durch das Polynom

$$p_f = m_1^{(X)} m_1^{(Y)} + \ldots + m_{k_f}^{(X)} m_{k_f}^{(Y)}$$

repräsentiert. Dabei seien $m_i^{(X)}$ Monome über den Variablen aus X, $m_i^{(Y)}$ Monome über den Variablen aus Y.

$\overline{f}$ werde repräsentiert durch

$$p_{\overline{f}} = l_1^{(X)} l_1^{(Y)} + \ldots + l_{k_{\overline{f}}}^{(X)} l_{k_{\overline{f}}}^{(Y)}$$

Sei $N = \{1, \ldots, k_f\}$ und

$$q = |\{A \mid A = \bigwedge_{i \in I} m_i^{(X)} \wedge \overline{\bigvee_{i \in (N \setminus I)} m_i^{(X)}} \text{ mit } I \subseteq N,\ A \neq 0\}|$$

die Anzahl aller verschiedenen Überlappungen der Monome $m_1^{(X)}, \ldots, m_{k_f}^{(X)}$.

In [HOI92] wird ein rekursives Verfahren zur Berechnung von $vz(X, f)$ angegeben, dessen Rekursionsbaum nach Angaben in [HOI92] höchstens q Blätter hat. Die Laufzeit des Verfahrens ist nach [HOI92] dann durch $O(k_f q)$ gegeben.

Nach einer eigenen Analyse des Verfahrens ist die Anzahl der Blätter des Rekursionsbaumes allerdings nicht durch q beschränkt. Der Rekursionsbaum kann $2^{|X|}$ Blätter haben. Außerdem muß auch die Anzahl $k_{\overline{f}}$ der Monome von $p_{\overline{f}}$ in die Laufzeitberechnung eingehen. Die *worst case*–Laufzeit des in [HOI92] angegebenen Algorithmus beträgt dann $O(k_f k_{\overline{f}} 2^{|X|})$ zur Berechnung von $vz(X, f)$. (Dies ist ein wesentlicher Unterschied, da man leicht Funktionen f finden kann, bei denen die Größe des Minimalpolynoms zu $\overline{f}$ exponentiell in der Größe des Minimalpolynoms zu f ist.)

ROBDD–Darstellungen Eine andere Möglichkeit der Darstellung boolescher Funktionen sind reduzierte Funktionsgraphen (ROBDD's). Für in der Praxis interessante Funktionen ist die ROBDD–Dastellung häufig *wesentlich* kleiner als die Größe der Funktionstabelle.

Sei $F = (G, m)$ ein reduzierter Funktionsgraph über der durch $<_v$ geordneten Menge $X_n = \{x_1, \ldots, x_n\}$ von Eingangsvariablen und sei $f \in B_n$ die durch F definierte Funktion. (Die Ordnung auf X_n sei so definiert, daß $x_i <_v x_j$ genau dann, wenn $i < j$.)

Hier soll kurz eine Idee angegeben werden, die es ermöglichen soll, bei Vorgabe einer Variablenteilmenge $X_p = \{x_1, \ldots, x_p\} \subseteq X_n$ die Anzahl $vz(X_p, f)$ der Zeilen der Zerlegungsmatrix Z von f hinsichtlich X_p zu berechnen, ohne Z explizit aufzustellen.

Die entscheidende Beobachtung hierbei ist, daß bei

$$\epsilon = (\epsilon_1, \ldots, \epsilon_p) \in \{0, 1\}^p$$

die Zeile von Z mit Index $int(\epsilon)$[‡] gerade die Funktionstabelle des Kofaktors $f_{x_1^{\epsilon_1} \ldots x_p^{\epsilon_p}}$ von f darstellt.

[‡] $int(\epsilon)$ gibt den dezimalen Wert der Binärzahl ϵ an.

Die Aufgabe, die Anzahl der verschiedenen Zeilen von Z zu berechnen, ist also gleichbedeutend mit der Aufgabe, die Anzahl der verschiedenen Kofaktoren

$$f_{x_1^{\epsilon_1} \ldots x_p^{\epsilon_p}} \quad \text{mit} \quad (\epsilon_1, \ldots, \epsilon_p) \in \{0,1\}^p$$

zu bestimmen.

Wie schon in Kapitel 1 bemerkt wurde, ist es bei reduzierten Funktionsgraphen leicht zu testen, ob die Kofaktoren

$$f_{x_1^{\epsilon_1} \ldots x_k^{\epsilon_k}} \text{ und } f_{x_1^{\delta_1} \ldots x_k^{\delta_k}}$$

gleich sind. Sie sind genau dann gleich, wenn die durch ϵ bzw. δ erreichbaren Knoten von G identisch sind. $vz(X_p, f)$ ist also gleich der Anzahl der verschiedenen durch p–Tupel $\epsilon \in \{0,1\}^p$ erreichbaren Knoten des reduzierten Funktionsgraphen.

$vz(X_p, f)$ läßt sich demnach mit einem einfachen Algorithmus bestimmen:
Zu Beginn wird der Zähler vz auf 0 gesetzt. Man durchläuft dann G nach einer Tiefensuchstrategie. Abweichend von der üblichen Tiefensuche hört man nicht erst dann auf, in die Tiefe zu laufen, wenn man an einem Blatt ankommt, sondern dann, wenn man an einem Knoten v ankommt, der als Markierung eine Variable x_l mit $l \geq p$ trägt oder 0 bzw. 1 als Markierung trägt. Wurde v vorher noch nicht besucht, so wird vz um 1 erhöht.

Ist dieses Verfahren beendet, so liefert der Stand des Zählers vz die Anzahl der verschiedenen durch p–Tupel $\epsilon \in \{0,1\}^p$ erreichbaren Knoten von G und damit den gesuchten Wert $vz(X_p, f)$. Man kann $vz(X_p, f)$ also mit einer Laufzeit bestimmen, die linear in der Größe von G ist.

Das gerade beschriebene Verfahren funktioniert allerdings nur dann, wenn die Variablenteilmenge X_p gerade aus den p Variablen aus X_n besteht, die hinsichtlich der auf X_n definierten Ordnung $<_v$ die kleinsten sind.

Sucht man jedoch nach einer geeigneten Variablenteilmenge X_p, um eine Zerlegung hinsichtlich X_p durchzuführen, und testet man dazu verschiedene Teilmengen X_p, so enthält X_p natürlich nicht immer gerade die hinsichtlich $<_v$ kleinsten Variablen aus X_n.

Will man das angegebene Verfahren trotzdem anwenden, so muß man zu f einen reduzierten Funktionsgraphen mit einer anderen Variablenordnung bestimmen.[4]

Es würde sich anbieten, dieses Verfahren zur Bestimmung von $vz(X_p, f)$ in Verbindung mit einem Verfahren der iterativen Verbesserung bei der Bestimmung einer geeigneten Variablenteilmenge zu verwenden. Tauscht man eine Variable der aktuellen Teilmenge X_p mit einer Variable aus $X_n \setminus X_p$ aus, so könnte die Bestimmung eines ROBDD mit der geänderten Variablenordnung evtl. durch lokales „Umbauen“ des ursprünglichen ROBDD und anschließendes Reduzieren des erhaltenen OBDD erfolgen.

[4] Die Größe eines Funktionsgraphen kann sich allerdings durch Ändern der Variablenordnung verändern.

4.1.3 Kosten von Realisierungen, die nichttriviale Zerlegbarkeit ausnutzen

Die nichttriviale Zerlegbarkeit einer Funktion $f \in B_n$ ist eine spezielle Struktureigenschaft der Funktion, die gewährleistet, daß die Funktion eine wesentlich geringere Komplexität hat als die durch Shannon festgestellte Mindestkomplexität $\frac{2^n}{n}$, die fast alle zufällig gewählten Funktionen überschreiten.

Lemma 4.3 *Besitzt eine Funktion $f \in B_n$ eine nichttriviale einseitige Zerlegung hinsichtlich der Variablenteilmenge $\{x_1, \ldots, x_p\}$ und ist p konstant, n hinreichend groß, so gibt es eine Realisierung zu f, die diese Zerlegung ausnutzt und eine B_2–Komplexität $< \frac{2^n}{n}$ hat.*

Beweis:
$f \in B_n$ habe eine nichttriviale Zerlegung der Form

$$f(x_1, \ldots, x_p, y_1, \ldots, y_q) = g(\alpha_1(x_1, \ldots, x_p), \ldots, \alpha_r(x_1, \ldots, x_p), y_1, \ldots, y_q)$$

mit $r < p$.

Da p konstant ist, lassen sich die Kosten für die Realisierung von $\alpha_1, \ldots, \alpha_r$ abschätzen durch eine Konstante C.

Nach dem Satz von Lupanow gilt für die B_2–Komplexität von g

$$C_{B_2}(g) \leq \frac{2^{r+q}}{r+q} + o(\frac{2^{r+q}}{r+q}) \leq \frac{2^{p-1+q}}{p-1+q} + o(\frac{2^{p-1+q}}{p-1+q}) = \frac{2^{n-1}}{n-1} + o(\frac{2^{n-1}}{n-1}).$$

Die Gesamtkosten eines Schaltkreises S, der die Zerlegung ausnutzt, lassen sich abschätzen durch

$$C_{B_2}(S) \leq C + \frac{2^{n-1}}{n-1} + o(\frac{2^{n-1}}{n-1}).$$

Ist n genügend groß, so gilt

$$C_{B_2}(S) \leq C + \frac{2^{n-1}}{n-1} + o(\frac{2^{n-1}}{n-1}) < \frac{2^n}{n}.$$

□

Die Aussage des Lemmas ist aber nicht auf konstante p beschränkt:

Lemma 4.4 *Besitzt eine Funktion $f \in B_n$ eine nichttriviale einseitige Zerlegung hinsichtlich der Variablenteilmenge $\{x_1, \ldots, x_p\}$, so gibt es eine Realisierung zu f, die diese Zerlegung ausnutzt und eine B_2–Komplexität $< \frac{2^n}{n}$ hat, falls n und p hinreichend groß sind und*

$$p < (n-1) - \log \frac{n(n-1)}{n-2}.$$

Beweis:
Sei nach Voraussetzung $p < (n-1) - \log \frac{n(n-1)}{n-2}$.
Dann gilt mit genügend kleinem $c > 1$:

$$p \leq (n-1) - \log \frac{cn(n-1)}{(2-c)n-2}.$$

Diese Ungleichung läßt sich äquivalent umformen:

$\iff \quad \underbrace{n-p}_{q} \geq [\log \frac{cn(n-1)}{(2-c)n-2}] + 1$

$\iff \quad 2^{q-1} \geq \frac{cn(n-1)}{(2-c)n-2}$

$\iff \quad cn(n-1) \leq 2^{q-1}(2n-2-cn)$

$\iff \quad c \leq \frac{2^q(n-1)-2^{q-1}cn}{n(n-1)}$

$\iff \quad c(1+\frac{2^{q-1}}{n-1}) \leq \frac{2^q}{n} \qquad (*)$

(Voraussetzung $p < (n-1) - \log \frac{n(n-1)}{n-2}$ wird an späterer Stelle in Form von Ungleichung (*) ausgenutzt.)

Sei eine nichttriviale Zerlegung der Form

$$f(x_1,\ldots,x_p,y_1,\ldots,y_q) = g(\alpha_1(x_1,\ldots,x_p),\ldots,\alpha_r(x_1,\ldots,x_p),y_1,\ldots,y_q)$$

gegeben.

Für die Komplexität der Zerlegungsfunktionen gilt nach dem Satz von Lupanow:

$$C_{B_2}(\alpha_i) \leq \frac{2^p}{p} + o(\frac{2^p}{p})$$

Für die Komplexität der Zusammensetzungsfunktion gilt:

$$C_{B_2}(g) \leq \frac{2^{r+q}}{r+q} + o(\frac{2^{r+q}}{r+q}) \leq \frac{2^{p-1+q}}{p-1+q} + o(\frac{2^{p-1+q}}{p-1+q})$$

Sind p und $n = p+q$ hinreichend groß, so gilt:

$$C_{B_2}(\alpha_i) \leq \frac{2^p}{p} + o(\frac{2^p}{p}) \leq c\frac{2^p}{p}$$

und

$$C_{B_2}(g) \leq \frac{2^{p-1+q}}{p-1+q} + o(\frac{2^{p-1+q}}{p-1+q}) \leq c\frac{2^{p-1+q}}{p-1+q}.$$

Die Gesamtkosten eines Schaltkreises S, der die gegebene Zerlegung ausnutzt, lassen sich dann abschätzen durch

$$\begin{aligned} C_{B_2}(S) &\leq rc\frac{2^p}{p} + c\frac{2^{p-1+q}}{p-1+q} \\ &\leq c\left[(p-1)\frac{2^p}{p} + \frac{2^{p-1+q}}{p-1+q}\right] \end{aligned}$$

$$\begin{aligned}
&= c2^p\left[\frac{p-1}{p}+\frac{2^{q-1}}{n-1}\right] \\
&< 2^p c\left[1+\frac{2^{q-1}}{n-1}\right] \\
&\leq 2^p\frac{2^q}{n} \quad \text{wegen (*)} \\
&= \frac{2^n}{n}
\end{aligned}$$

□

Aufgrund des Shannon'schen Abzählargumentes ist es nach den beiden Lemmata also unwahrscheinlich, daß eine zufällig gewählte Funktion eine solche nichttriviale Zerlegung besitzt. Wie in Kapitel 2 schon angedeutet, hat man es in der Praxis aber nicht mit zufällig gewählten Funktionen zu tun (vgl. auch Abschnitt 4.5).

Auch bei Funktionen mit *ausgezeichneten* Zerlegungseigenschaften muß die Ausnutzung von naheliegenden Zerlegungen nicht notwendigerweise zu optimalen Realisierungen führen.

Dies geht aus einem Satz von W. J. Paul ([Pau76]) hervor. Der Satz wurde formuliert, um zu zeigen, daß die Ausnutzung von Zerlegungen nach Ashenhurst ([Ash59]) nicht notwendigerweise zu optimalen Realisierungen führt. Die von Ashenhurst definierten Zerlegungen sind nichttriviale einseitige Zerlegungen mit genau einer Zerlegungsfunktion.

Satz 4.3 (Paul '76) *Für alle $\epsilon \geq 0$ gibt es ein $d \in \mathbf{N}$, so daß es für hinreichend großes $n \in \mathbf{N}$ eine Funktion $f \in B_n$ gibt mit $C_{B_2}(f) \geq 2^{n^{1/d}}$ und*

$$C_{B_2}(or_2 \circ (f \times f)) \leq (1+\epsilon)C_{B_2}(f).$$

Hierbei definiert $(f \times f)$ eine Funktion aus $B_{2n,2}$ mit

$$(f \times f)(x_1, \ldots, x_n, y_1, \ldots, y_n) = (f(x_1, \ldots, x_n), f(y_1, \ldots y_n))$$

Der Satz von Paul ist auch auf gleichmächtige Zerlegungen anwendbar:

Lemma 4.5 *Es gibt Funktionen $G \in B_{2n}$ mit folgender Eigenschaft:*

Sei $\{X, Y\}$ unter allen Variablenaufteilungen in 2 gleichmächtige Mengen diejenige, zu der es eine gleichmächtige Zerlegung gibt, die eine minimale Anzahl von Zerlegungsfunktionen erfordert.

Zu jeder Realisierung S, die eine gleichmächtige Zerlegung hinsichtlich $\{X, Y\}$ ausnutzt, gibt es eine Realisierung von G, deren Kosten beinahe nur halb so groß sind wie die Kosten von S.

Beweis:
Nach dem Satz von Paul kann man zu einem beliebig kleinen $\epsilon > 0$ $d \in \mathbf{N}$, ein hinreichend großes $m \in \mathbf{N}$ und $f \in B_m$ finden mit $C_{B_2}(f) \geq 2^{m^{1/d}}$ und

$$C_{B_2}(or_2 \circ (f \times f)) \leq (1+\epsilon)C_{B_2}(f).$$

Ist f unabhängig von $(m-n)$ Eingangsvariablen, o.B.d.A. von $x_{n+1}, \ldots, x_m$, so wähle $g \in B_n$ $(n \leq m)$ mit

$$g(x_1, \ldots, x_n) = f(x_1, \ldots, x_m) \text{ für } (x_1, \ldots, x_m) \in \{0,1\}^m$$

Es gilt:

$$C_{B_2}(g) = C_{B_2}(f) \quad \text{und} \quad C_{B_2}(or_2 \circ (f \times f)) = C_{B_2}(or_2 \circ (g \times g))$$

(Denn auch $or_2 \circ (f \times f)$ ist unabhängig von $x_{n+1}, \ldots, x_m$.)

Wähle

$$G = or_2 \circ (g \times g).$$

Es gilt

$$G(x_1, \ldots, x_n, y_1, \ldots, y_n) = g(x_1, \ldots, x_n) \vee g(y_1, \ldots, y_n).$$

G ist abhängig von allen $2n$ Eingangsvariablen.[5] Daher kann es keine zweiseitige Zerlegung mit nur einer Zerlegungsfunktion geben. Die naheliegende Variablenaufteilung $\{X, Y\} = \{\{x_1, \ldots, x_n\}, \{y_1, \ldots, y_n\}\}$ erfordert also die geringste Zahl von Zerlegungsfunktionen, nämlich je eine Zerlegungsfunktion auf jeder der beiden Teilmengen der Variablenaufteilung.

Als Zerlegungsfunktion auf den Variablen aus X kann man entweder g oder $\overline{g}$ wählen. Auch bei der Zerlegungsfunktion auf Y hat man die Wahl zwischen g und $\overline{g}$.

Es gibt also genau 4 mögliche Zerlegungen hinsichtlich der Variablenaufteilung $\{X, Y\}$.

Für die Kosten jeder Realisierung S, die eine der 4 möglichen Zerlegungen ausnutzt, gilt

$$C_{B_2}(S) \geq 2C_{B_2}(g) + 1.$$

Es gilt aber auch

$$C_{B_2}(G) \leq (1+\epsilon)C_{B_2}(g), \text{d.h.}$$

$$C_{B_2}(S) \geq \frac{2}{1+\epsilon} C_{B_2}(G) + 1$$

□

[5] $n > 0$ wegen $C_{B_2}(g) = C_{B_2}(f) \geq 2^{m^{1/d}}$.

4.1.4 Wahl von Zerlegungsfunktionen

Gesucht sei eine einseitige Zerlegung einer booleschen Funktion $f \in B_n$ mit den Eingangsvariablen $x_1, \ldots, x_p, y_1, \ldots, y_q$, hinsichtlich einer Variablenteilmenge $X = \{x_1, \ldots, x_p\}$, also eine Darstellung von f in der Form

$$f(x_1, \ldots, x_p, y_1, \ldots, y_q) = g(\alpha_1(x_1, \ldots, x_p), \ldots, \alpha_r(x_1, \ldots, x_p), y_1, \ldots, y_q).$$

Aus Korollar 4.1 ergibt sich, daß sich alle r–Tupel von Funktionen $(\alpha_1, \ldots, \alpha_r)$ $= \alpha$ als Zerlegungsfunktionen eignen, für die gilt: α ordnet den Indizes verschiedener Zeilen der zugehörigen Zerlegungsmatrix Z verschiedene Funktionswerte zu.

Die Wahl der Zerlegungsfunktionen ist also nicht eindeutig, sondern man hat viele verschiedene Möglichkeiten, geeignete r–Tupel von Zerlegungsfunktionen zu wählen. Ist $r = \log(vz(X, f))$, so gibt es im Bild von α $2^r = vz(X, f)$ verschiedene Werte, die den Indizes verschiedener Zeilen der Zerlegungsmatrix zugeordnet werden. Da es auch genau $2^r = vz(X, f)$ verschiedene Zeilen der Zerlegungsmatrix gibt, handelt es sich um eine eineindeutige Zuordnung zwischen den Funktionswerten von α und Indizes verschiedener Zerlegungsmatrixzeilen. Insgesamt sind $(2^r)!$ solche Zuordnungen möglich, es gibt also $(2^r)!$ verschiedene Möglichkeiten, die Zerlegungsfunktionen $(\alpha_1, \ldots, \alpha_r)$ zu wählen[6]. Ist $r = \lceil \log(vz(X, f)) \rceil$ und $vz(X, f)$ keine Zweierpotenz, gibt es im Bild von $\alpha = (\alpha_1, \ldots, \alpha_r)$ mehr als $vz(X, f)$ Werte. In diesem Fall gibt es noch mehr als $(2^r)!$ verschiedene Möglichkeiten für die Wahl von $(\alpha_1, \ldots, \alpha_r)$, da α auch den Indizes gleicher Zeilen der Zerlegungsmatrix verschiedene Funktionswerte zuordnen kann.

Will man eine Zerlegung hinsichtlich einer Variablenteilmenge X durchführen, so ist man also nicht an eine bestimmte Wahl der Zerlegungsfunktionen gebunden, sondern man hat bei der Auswahl der Zerlegungsfunktionen gewisse Freiheiten. Da sich die Komplexität verschiedener Auswahlen von Zerlegungsfunktionen stark unterscheiden kann, ist es sinnvoll, diese Freiheiten auszunutzen, um möglichst einfache Zerlegungsfunktionen zu finden.

Ein solcher Ansatz findet sich in den Arbeiten von Karp ([Kar63]), während bei Curtis ([Cur61]) die Zerlegungsfunktionen zufällig gewählt wurden. In [Kar63] wird vorgeschlagen, bei der Suche nach „einfachen Zerlegungsfunktionen“ Unteralgebren „einfacher“ Funktionen vorzugeben und zu testen, ob es Zerlegungsfunktionen gibt, die einer solchen Unteralgebra angehören. Als solche Unteralgebren denkbar sind beispielsweise die Unteralgebra aller totalsymmetrischen Funktionen oder eine Unteralgebra von Funktionen, die von einer bestimmten Teilmenge der Eingangsvariablen unabhängig sind.

[6] Da r minimal gewählt ist, sind alle Funktionen $\alpha_1, \ldots, \alpha_r$ paarweise verschieden und können auch nicht durch Invertierung auseinander hervorgehen. Allerdings wurden bei den $(2^r)!$ Möglichkeiten der Wahl von Zerlegungsfunktionen außer einem r-Tupel $(\alpha_1, \ldots, \alpha_r)$ auch sämtliche r–Tupel gezählt, die aus $(\alpha_1, \ldots, \alpha_r)$ durch Vertauschung oder Invertierung der Komponenten hervorgehen. Es gibt zu $(\alpha_1, \ldots, \alpha_r)$ genau $2^r \cdot r!$ solcher r–Tupel. Zählt man solche trivialen Möglichkeiten, verschiedene r–Tupel von Zerlegungsfunktionen zu erhalten, nicht mit, so erhält man $(2^r - 1)!/r!$ verschiedene Wahlmöglichkeiten für die Zerlegungsfunktionen.

Es ist allerdings nicht zu erwarten, daß man bei Vorgabe einer Unteralgabra U von B_p alle Zerlegungsfunktionen α_i aus U wählen kann. Will man aber mindestens einen Teil der Zerlegungsfunktionen aus U wählen, so ist folgende Beobachtung von Bedeutung:

Lemma 4.6 *Sei $f \in B_n$ mit den Eingangsvariablen $x_1, \ldots, x_p$, $y_1, \ldots, y_q$ und sei $r = \lceil \log(vz(X,f)) \rceil$. Dann existiert genau dann eine einseitige Zerlegung hinsichtlich X der Form*

$$f(\mathbf{x}, \mathbf{y}) = g(\alpha'_1(\mathbf{x}), \ldots, \alpha'_h(\mathbf{x}), \alpha_{h+1}(\mathbf{x}), \ldots, \alpha_r(\mathbf{x}), \mathbf{y}),$$

wenn gilt

$$vz(X, f, \alpha') \leq 2^{r-h}.$$

Hierbei ist die Bezeichnung $vz(X, f, \alpha')$ folgendermaßen definiert:

Bezeichnung 13 *Sei Z die Zerlegungsmatrix von f hinsichtlich X und sei A das Bild von $\alpha' = (\alpha'_1, \ldots, \alpha'_h)$. Zu $a \in A$ sei anz_a die Anzahl der* verschiedenen *Zeilen von Z, für die gilt: $\alpha'(x_1, \ldots, x_p) = a$ und $int(x_1, \ldots, x_p)$ ist der Index dieser Zeile. $vz(X, f, \alpha')$ ist dann das Maximum über alle anz_a für $a \in A$.*

Beweis:
Der Beweis erfolgt im wesentlichen analog zum Beweis von Satz 4.1.

„$\Longleftarrow$“: Sei eine Zerlegung gegeben durch

$$f(\mathbf{x}, \mathbf{y}) = g(\alpha'_1(\mathbf{x}), \ldots, \alpha'_h(\mathbf{x}), \alpha_{h+1}(\mathbf{x}), \ldots, \alpha_r(\mathbf{x}), \mathbf{y}).$$

Die Funktion

$$\alpha'' = (\alpha_{h+1}, \ldots, \alpha_r) \in B_{p,r-h}$$

kann genau 2^{r-h} verschiedene Funktionswerte annehmen.
Angenommen

$$vz(X, f, \alpha') > 2^{r-h}.$$

Dann muß es ein $a \in \{0,1\}^h$ im Bild von α', $x^{(1)} = (x_1^{(1)}, \ldots, x_p^{(1)})$ und $x^{(2)} = (x_1^{(2)}, \ldots, x_p^{(2)}) \in \{0,1\}^p$ geben, so daß

$$\alpha'(x^{(1)}) = \alpha'(x^{(2)}) = a,$$

$$\alpha''(x^{(1)}) = \alpha''(x^{(2)}),$$

aber die zu $x^{(1)}$, $x^{(2)}$ gehörigen Zeilen verschieden sind.
Es kann also keine Zerlegung geben mit

$$\alpha'_1, \ldots, \alpha'_h, \alpha_{h+1}, \ldots, \alpha_r$$

als Zerlegungsfunktionen.

„$\Longrightarrow$“: Sei
$$vz(X, f, \alpha') \leq 2^{r-h}.$$

Betrachte ein a aus dem Bild von α'.
Sei $X^a \subseteq \{0,1\}^p$ die Menge aller Urbilder von a.
Dann kann man $\alpha''_a : X^a \to \{0,1\}^{r-h}$ so definieren, daß

$$\alpha''_a(x^{(1)}) \neq \alpha''_a(x^{(2)})$$

für alle $x^{(1)}, x^{(2)} \in X^a$, für die die zugehörigen Zerlegungsmatrixzeilen verschieden sind.
Definiert man auf diese Weise für alle a aus dem Bild von α' eine Funktion α''_a, so kann man diese Funktionen zu einer Funktion α'' auf $\{0,1\}^p$ zusammensetzen. Es gilt dann:

$$(\alpha', \alpha'')(x^{(1)}) \neq (\alpha', \alpha'')(x^{(2)})$$

für alle $x^{(1)}, x^{(2)} \in \{0,1\}^p$, für die die zugehörigen Zerlegungsmatrixzeilen verschieden sind.

□

- Das von Karp vorgeschlagene Verfahren sucht bei Vorgabe einer Unteralgebra U von B_p einzelne Zerlegungsfunktionen α'_1, die dieser Unteralgebra angehören und in Zerlegungen der Form

$$\begin{aligned} &f(x_1, \ldots, x_p, y_1, \ldots, y_q) = \\ &\quad g(\alpha'_1(x_1, \ldots, x_p), \alpha_2(x_1, \ldots, x_p), \ldots, \alpha_r(x_1, \ldots, x_p), y_1, \ldots, y_q) \quad (*) \end{aligned}$$

  verwendbar sind ($r = \lceil \log(vz(X, f)) \rceil$). Nach Lemma 4.6 ist eine Funktion α'_1 genau dann geeignet, wenn $vz(X, f, \alpha'_1) \leq 2^{r-1}$. Die Anzahl der verschiedenen Zeilen der zugehörigen Zerlegungsmatrix Z, deren Indizes durch α'_1 0 bzw.1 zugeordnet wird, darf 2^{r-1} jeweils nicht übersteigen.

- Hat man auf diese Weise 2 Funktionen α'_1 und α'_2 gefunden, so muß es allerdings nicht notwendigerweise eine Zerlegung der Form

$$f(\mathbf{x}, \mathbf{y}) = g(\alpha'_1(\mathbf{x}), \alpha'_2(\mathbf{x}), \alpha_3(\mathbf{x}), \ldots, \alpha_r(\mathbf{x}), \mathbf{y})$$

  geben. Das Kriterium von Lemma 4.6 wird benutzt, um aus den gefundenen Funktionen eine Teilmenge $\{\alpha'_1, \ldots, \alpha'_h\}$ zu bestimmen (h möglichst groß) mit
$$f(\mathbf{x}, \mathbf{y}) = g(\alpha'_1(\mathbf{x}), \ldots, \alpha'_h(\mathbf{x}), \alpha_{h+1}(\mathbf{x}), \ldots, \alpha_r(\mathbf{x}), \mathbf{y}).$$

Das Verfahren von Karp zur Bestimmung von Zerlegungsfunktionen aus einer Unteralgebra U von B_p soll nun etwas genauer beschrieben werden:

Es sei ein beliebiges Erzeugendensystem $E_U = \{e_1, \ldots, e_k\}$ der Unteralgebra U von B_p gegeben. Alle Funktionen aus E_U lassen sich durch die Operationen $\wedge$,

$\vee$ und $^-$ aus den Elementen von E_U erzeugen, d.h. alle Funktionen u aus E_U lassen sich darstellen in der Form

$$u(\mathbf{x}) = h(e_1(\mathbf{x}), \ldots, e_k(\mathbf{x}))\ \forall \mathbf{x} \in \{0,1\}^p$$

mit einer Funktion $h \in B_k$. (Und alle Darstellungen dieser Form liefern eine Funktion aus E_U.)
Folglich ist die Suche nach einer Funktion α_1', die als Zerlegungsfunktion verwendbar ist, gleichbedeutend mit der Suche nach einer Funktion $h \in B_k$.
Bei einer beliebigen Funktion $u \in U$ mit

$$u(\mathbf{x}) = h(e_1(\mathbf{x}), \ldots, e_k(\mathbf{x}))\ \forall \mathbf{x} \in \{0,1\}^p$$

gilt für alle $\mathbf{x}, \mathbf{y} \in \{0,1\}^p$ mit

$$(e_1(\mathbf{x}), \ldots, e_k(\mathbf{x})) = (e_1(\mathbf{y}), \ldots, e_k(\mathbf{y})) = (\epsilon_1, \ldots, \epsilon_k):$$

$u(\mathbf{x}) = h(\epsilon_1, \ldots, \epsilon_k) = u(\mathbf{y})$.
Legt man also den Funktionswert von h auf $\epsilon \in \{0,1\}^k$ auf 0 (bzw. 1) fest, so gilt für alle $\mathbf{x}$ aus X^ϵ, der Menge aller $\mathbf{x} \in \{0,1\}^p$ mit $(e_1(\mathbf{x}), \ldots, e_k(\mathbf{x})) = \epsilon$**: $u(\mathbf{x}) = 0$ (bzw. 1).

Die Gleichheit auf den Zeilen der Zerlegungsmatrix Z liefert eine Äquivalenzklasseneinteilung von $\{0,1\}^p$. Sei diese Äquivalenzklasseneinteilung gegeben durch $\{K_1, \ldots, K_{vz(X,f)}\}$. Nach dem oben Gesagten wird es interessant, zu $\epsilon \in \{0,1\}^k$ die Äquivalenzklassen K_i zu bestimmen, bei denen sich X^ϵ und K_i schneiden. Die Menge

$$EK_\epsilon = \{1 \leq i \leq vz(X,f) \mid K_i \cap X^\epsilon \neq \emptyset\}$$

gibt die Menge der Indizes i von Äquivalenzklassen K_i an, für die gilt: K_i enthält ein $\mathbf{x}$ mit $(e_1, \ldots, e_k)(\mathbf{x}) = \epsilon$. ($|EK_\epsilon|$ gibt also die Anzahl der *verschiedenen* Zeilen von Z an, auf deren Indizes $(e_1, \ldots, e_k)$ den Wert ϵ liefert.)

Sei das Bild von $(e_1, \ldots, e_k)$ auf $\{0,1\}^p$ gegeben durch $\{\epsilon^{(1)}, \ldots, \epsilon^{(l)}\}$. Sei $\alpha_1' \in U$, mit

$$\alpha_1'(\mathbf{x}) = h(e_1(\mathbf{x}), \ldots, e_k(\mathbf{x}))\ \forall \mathbf{x} \in \{0,1\}^p.$$

Dann ist die Anzahl der verschiedenen Zeilen von Z, auf deren Indizes α_1' 0 (bzw. 1) liefert gleich

$$|\bigcup_{h(\epsilon^{(i)})=0} EK_{\epsilon^{(i)}}| \quad (\text{bzw. } |\bigcup_{h(\epsilon^{(i)})=1} EK_{\epsilon^{(i)}}|).$$

Es gibt also genau dann eine Zerlegung der Form (*) mit α_1', wenn

$$|\bigcup_{h(\epsilon^{(i)})=0} EK_{\epsilon^{(i)}}| \leq 2^{r-1}$$

**Hier wird auch klar, warum man ein *beliebiges* Erzeugendensystem von U wählen konnte: Die Mengen X^ϵ, die nicht leer sind, sind für alle Erzeugendensysteme gleich: Es handelt sich um die ON–Mengen der Atome von U.

und

$$|\bigcup_{h(\epsilon^{(i)})=1} EK_{\epsilon^{(i)}}| \leq 2^{r-1}.$$

Hat man eine Funktion α_1' mit

$$\alpha_1'(\mathbf{x}) = h(e_1(\mathbf{x}), \ldots, e_k(\mathbf{x}))\ \forall \mathbf{x} \in \{0,1\}^p$$

gegeben, so daß es eine Zerlegung der Form (*) gibt, so erhält man also durch

$$S_0 = \bigcup_{h(\epsilon^{(i)})=0} EK_{\epsilon^{(i)}} \text{ und } S_1 = \bigcup_{h(\epsilon^{(i)})=1} EK_{\epsilon^{(i)}}$$

zwei Mengen mit Mächtigkeit kleiner gleich 2^{r-1} und es gilt $S_0 \cup S_1 = \{1, \ldots, vz(X,f)\}$.

Umgekehrt findet man zu jedem Paar von Mengen S_0 und S_1 mit

1. $|S_0| \leq 2^{r-1}$, $|S_1| \leq 2^{r-1}$,
2. $S_0 \cup S_1 = \{1, \ldots, vz(X,f)\}$ und
3. $EK_{\epsilon^{(i)}} \subseteq S_0$ oder $EK_{\epsilon^{(i)}} \subseteq S_1\ \forall 1 \leq i \leq l$

eine Funktion h und damit eine Funktion α_1' ($\alpha_1'(\mathbf{x}) = h(e_1(\mathbf{x}), \ldots, e_k(\mathbf{x}))$), so daß

$$\bigcup_{h(\epsilon^{(i)})=0} EK_{\epsilon^{(i)}} \subseteq S_0 \text{ und } \bigcup_{h(\epsilon^{(i)})=1} EK_{\epsilon^{(i)}} \subseteq S_1.$$

Man setzt einfach

$$h(\epsilon^{(i)}) = 0, \text{ falls } EK_{\epsilon^{(i)}} \subseteq S_0 \text{ und } EK_{\epsilon^{(i)}} \not\subseteq S_1,$$
$$h(\epsilon^{(i)}) = 1, \text{ falls } EK_{\epsilon^{(i)}} \subseteq S_1 \text{ und } EK_{\epsilon^{(i)}} \not\subseteq S_0,$$
$$h(\epsilon^{(i)}) \text{ beliebig, falls } EK_{\epsilon^{(i)}} \subseteq S_0 \text{ und } EK_{\epsilon^{(i)}} \subseteq S_1.$$

Folglich gilt

$$|\bigcup_{h(\epsilon^{(i)})=0} EK_{\epsilon^{(i)}}| \leq |S_0| \leq 2^{r-1} \text{ und } |\bigcup_{h(\epsilon^{(i)})=1} EK_{\epsilon^{(i)}}| \leq |S_1| \leq 2^{r-1}.$$

Dann gibt es eine Zerlegung der Form (*) mit α_1'.

Das Verfahren von Karp bestimmt nun Funktionen α_1' aus einer Unteralgebra U, indem es Mengen mit Eigenschaften 1.) bis 3.) berechnet. Insgesamt hat das Verfahren aber folgende Nachteile:

- Es ist darauf angelegt, *alle* möglichen Zerlegungsfunktionen α_1' zu finden. Dies können aber unter Umständen *sehr viele* Funktionen sein. Dadurch kann die Laufzeit des Verfahrens sehr groß werden.

- Das Verfahren sucht *einzelne* Zerlegungsfunktionen, aus denen nach Lemma 4.6 eine Teilmenge $\{\alpha'_1, \ldots, \alpha'_h\}$ bestimmt wird mit

  $$f(\mathbf{x}, \mathbf{y}) = g(\alpha'_1(\mathbf{x}), \ldots, \alpha'_h(\mathbf{x}), \alpha_{h+1}(\mathbf{x}), \ldots, \alpha_r(\mathbf{x}), \mathbf{y}).$$

  Es gibt allerdings (speziell für große r) sehr viele mögliche Auswahlen von Teilmengen gefundener Zerlegungsfunktionen. Eine solche Zusammensetzung der gefundenen Zerlegungsfunktionen ist oft zeitaufwendig. Außerdem ist es möglich, daß es keine größeren Teilmengen der gefundenen Zerlegungsfunktionen aus U gibt, so daß die Funktionen aus solchen Teilmengen *zusammen* in einer Zerlegung (mit minimaler Anzahl von Zerlegungsfunktionen) verwendbar sind.

Um die Laufzeit des Verfahrens einzuschränken, werden in der vorliegenden Arbeit mit einem branch–and–bound–Verfahren in *einem* Schritt h–Tupel von Zerlegungsfunktionen berechnet, die einer vorgegebenen Unteralgebra angehören (zur Implementierung vgl. Abschnitt 4.5). Außerdem begnügt man sich aus Laufzeitgründen mit dem Auffinden eines einzigen solchen h–Tupels von Zerlegungsfunktionen, anstatt nach allen möglichen h–Tupeln zu suchen. Bei der Wahl eines maximalen h, für das ein geeignetes h–Tupel von Zerlegungsfunktionen existiert, wird eine Binärsuche angewandt. Auf eine genauere Beschreibung des Verfahrens wird hier verzichtet, da es analog zu einem Verfahren für Funktionen mit mehreren Ausgängen arbeitet, das in Abschnitt 4.4 angegeben ist.

4.1.5 Faktorisierungsverfahren

An dieser Stelle soll noch kurz auf ein neueres Verfahren ([HOI89]) eingegangen werden, das ebenfalls Zerlegungen ausnutzt.

Gegeben sei eine boolesche Funktion $f \in B_n$ mit den Eingangsvariablen x_1, $\ldots$, x_p, y_1, $\ldots$, y_q. Sei $X = \{x_1, \ldots, x_p\}$ und $Y = \{y_1, \ldots, y_q\}$. Sei Z die Zerlegungsmatrix hinsichtlich der Variablenaufteilung $\{X, Y\}$. Man erhält nun eine Zerlegung hinsichtlich $\{X, Y\}$, indem man mit Hilfe des Gauss–Verfahrens über dem Körper $(\{0, 1\}, \oplus, \wedge)$ eine LDR–Zerlegung von Z berechnet.

Es gilt: $Z = LDR$, wobei L eine untere Dreiecksmatrix, R eine obere Dreiecksmatrix und D eine Diagonalmatrix ist, bei der die ersten $k = rang(Z)$ Hauptdiagonalelemente gleich 1 sind.

Sei $l^{(i)}$ die i. Spalte der Matrix L, $r^{(i)}$ die i. Zeile der Matrix R $(1 \leq i \leq k)$. Dann gilt wegen $Z = LDR = (LD)(DR)$:

$$Z_{ij} = \bigoplus_{1 \leq m \leq k} l_i^{(m)} \wedge r_j^{(m)}$$

Interpretiert man für $1 \leq i \leq k$ $l^{(i)}$ als Funktionstabelle einer Funktion α_i mit Eingangsvariablenmenge X und $r^{(i)}$ als Funktionstabelle einer Funktion β_i mit

Eingangsvariablenmenge Y, so liefert dies eine Zerlegung hinsichtlich $\{X, Y\}$:

$$\begin{aligned}
&f(x_1, \ldots, x_p, y_1, \ldots, y_q) = \\
&= \quad Z_{int(x_1,\ldots,x_p) int(y_1,\ldots,y_q)} \\
&= \quad \bigoplus_{1 \le m \le k} l^{(m)}_{int(x_1,\ldots,x_p)} \wedge r^{(m)}_{int(y_1,\ldots,y_q)} \\
&= \quad \bigoplus_{1 \le m \le k} (\alpha_m(x_1, \ldots, x_p) \wedge \beta_m(y_1, \ldots, y_q)) \\
&= \quad g(\alpha_1(x_1, \ldots, x_p), \ldots, \alpha_k(x_1, \ldots, x_p), \beta_1(y_1, \ldots, y_q), \ldots, \beta_k(y_1, \ldots, y_q)),
\end{aligned}$$

wobei

$$g(a_1, \ldots, a_k, b_1, \ldots, b_k) = \bigoplus_{1 \le m \le k} (a_m \wedge b_m)$$

für alle $(a_1, \ldots, a_k, b_1, \ldots, b_k) \in \{0,1\}^{2k}$.

Die so gefundene Zerlegung ist im allgemeinen keine Zerlegung mit minimaler Anzahl von Zerlegungsfunktionen. Die minimale Anzahl von Zerlegungsfunktionen beträgt nach Lemma 4.2

$$\lceil \log(vz(X, f)) \rceil + \lceil \log(vs(X, f)) \rceil.$$

Der Rang k der Matrix Z ist die maximale Anzahl linear unabhängiger Zeilen bzw. Spalten von Z. Gleiche Zeilen (Spalten) sind linear abhängig. Also gilt

$$\lceil \log(vz(X, f)) \rceil \le k \le vz(X, f) \text{ und}$$

$$\lceil \log(vs(X, f)) \rceil \le k \le vs(X, f)$$

k kann aber *wesentlich* größer als $\lceil \log(vz(X, f)) \rceil$ bzw. $\lceil \log(vs(X, f)) \rceil$ sein. Eigene Tests haben ergeben, daß bei zufällig gewählten Funktionen der Rang der Zerlegungsmatrix bei gleichmächtiger Zerlegung fast immer nahezu maximal ist, d.h. der Rang stimmt fast exakt mit dem Minimum aus Zeilen–und Spaltenzahl von Z überein. Speziell bei gleichmächtigen Zerlegungen mit $p = q = n/2$ und damit einer $2^{n/2} \times 2^{n/2}$–Zerlegungsmatrix ist dann die Anzahl der Eingänge der Zusammensetzungsfunktion g exponentiell in der Anzahl der Eingänge von f.

Häufig ist das Verfahren von Hwang/Owens/Irwin also weit von seinem Ziel „Exploiting Communication Complexity“ entfernt.

4.2 Zusammenhang zwischen Symmetrie und Zerlegung

Wenn man eine Realisierung für eine boolesche Funktion $f \in B_n$ sucht, so kann es vorkommen, daß man schon Vorinformationen über gewisse Symmetrieeigenschaften von f zur Verfügung hat. Im folgenden soll untersucht werden, wie man solche Informationen im Zusammenhang mit Zerlegungen ausnutzen kann.

Ist von einer Funktion $f \in B_n$ beispielsweise bekannt, daß sie teilweise symmetrisch in einer Teilmenge $X = \{x_1, \ldots, x_p\}$ der Eingangsvariablen ist ($p > 2$), d.h. daß sie invariant gegenüber sämtlichen Vertauschungen von Variablen aus X ist, so kann dieses Wissen ausgenutzt werden, indem man eine nichttriviale Zerlegung hinsichtlich X durchführt. Die Zerlegungsmatrix Z hinsichtlich X kann dann höchstens $(p+1)$ verschiedene Zeilen enthalten, denn aufgrund der Invarianz gegenüber Vertauschungen von Variablen aus X ist für den Funktionswert von f an einer Stelle $(\epsilon_1, \ldots, \epsilon_n)$ nur entscheidend, *wieviele* der ϵ_i mit $i \in \{1, \ldots, p\}$ gleich 1 sind, aber nicht *welche*. Also müssen Zeilen mit Indizes i und j, für die die Anzahl der Einsen in $bin_p(i)$ und $bin_p(j)$ gleich ist, identisch sein. Folglich kann man f nichttrivial zerlegen in der Form

$$f(x_1, \ldots, x_p, y_1, \ldots, y_q) = g(\alpha_1(x_1, \ldots, x_p), \ldots, \alpha_r(x_1, \ldots, x_p), y_1, \ldots, y_q)$$

mit $r \leq \lceil \log(p+1) \rceil$.

Die Überlegungen sollen nun auf G–symmetrische Funktionen erweitert werden, wobei die betrachteten Gruppen G Untergruppen von $\mathbf{P}_n$ sind, die durch Abbildungen σ_{ik} und ν_i $(1 \leq i < k \leq n)$ erzeugt sind[8].

Sei $f \in B_n$ eine Funktion mit den Eingangsvariablen $\{x_1, \ldots, x_n\}$. Sei X eine Teilmenge der Eingangsvariablen mit Indizes aus $I_X \subseteq \{1, \ldots, n\}$. Sei G eine Untergruppe von $\mathbf{P}_n$, die durch Abbildungen aus $E_\nu \cup E_\sigma$ erzeugt wird. Es gelte

$$E_\nu \subseteq \{\nu_i \mid i \in I_X\} \text{ und } E_\sigma \subseteq \{\sigma_{ik} \mid i, k \in I_X\}.$$

f sei G–symmetrisch.

Zur Abschätzung der minimalen Anzahl von Zerlegungsfunktionen bei einer Zerlegung von f hinsichtlich X wird eine Relation $\sim$ auf den Variablen in X definiert:
$x_i \sim x_j$ genau dann, wenn

- $\sigma_{ij} \in G$
  oder
- $\nu_i, \nu_j \in G$

Lemma 4.7 $\sim$ *ist eine Äquivalenzrelation.*

Beweis:
Der Beweis nutzt im wesentlichen die Gruppeneigenschaft von G aus.

<u>Reflexivität</u>:

klar, wenn man σ_{ii} als Identität definiert.

[8]Zur Definition von σ_{ik} und ν_i vergleiche Kapitel 3. Ist f $\{\sigma_{ik}\}$–symmetrisch, so ist f invariant gegenüber Vertauschung von x_i und x_k, ist f $\{\nu_i\}$–symmetrisch, so ist f unabhängig von x_i. Die Betrachtung von $\{\nu_i\}$–Symmetrie ist beispielsweise bei Funktionen mit mehreren Ausgängen von Interesse, da einzelne Ausgangsfunktionen oft nicht von allen Eingängen abhängen.

Symmetrie:
klar ($\sigma_{ik} = \sigma_{ki}$)

Transitivität:
Sei $x_i \sim x_j$ und $x_j \sim x_k$.

1.Fall: $\sigma_{ij}, \sigma_{jk} \in G$
$\Rightarrow \sigma_{ik} = \sigma_{ij} \circ \sigma_{jk} \circ \sigma_{ij} \in G$

2.Fall: „ν_i, ν_j und $\nu_j, \nu_k \in G$"
klar

3.Fall: $\sigma_{ij} \in G$, $\nu_j, \nu_k \in G$
$\Rightarrow \nu_i = \sigma_{ij} \circ \nu_j \circ \sigma_{ij} \in G$

4.Fall: $\sigma_{jk} \in G$, $\nu_i, \nu_j \in G$
analog zu Fall 3

□

Sei
$$X_U = \{x_i \in X \mid \exists \nu_i \in E_\nu\}.$$
Dann kann man die Äquivalenzklasseneinteilung von X bzgl. $\sim$ erhalten, indem man mit einer Partition beginnt, die aus X_U und Mengen $\{x_i\}$ für alle $x_i \in X \setminus X_U$ besteht. Dann betrachtet man nacheinander alle $\sigma_{ik} \in E_\sigma$ und — falls x_i und x_k in verschiedenen Mengen der Partition sind — vereinigt jeweils die Mengen, die x_i und x_k enthalten. Man erhält die gesuchte Äquivalenzklasseneinteilung
$$P = \{X_1, \ldots, X_k\}.$$
Sei X_1 die Menge, die X_U enthält. Dann ist f von allen Variablen aus X_1 unabhängig[9]. Außerdem ist f symmetrisch in allen Variablenmengen $X_2, \ldots, X_k$. Dann gilt:

Satz 4.4 *Seien f, X, G und die Partition $P = \{X_1, \ldots, X_k\}$ definiert wie oben. Dann ist die Anzahl der Zeilen der Zerlegungsmatrix Z von f hinsichtlich X*
$$vz(X, f) \leq (|X_2| + 1) \cdot (|X_3| + 1) \cdot \ldots \cdot (|X_k| + 1)$$
Folglich gibt es eine Zerlegung von f hinsichtlich X mit höchstens
$$\lceil \log((|X_2| + 1) \cdot \ldots \cdot (|X_k| + 1)) \rceil = \lceil \sum_{i=2}^{k} \log(|X_i| + 1) \rceil$$
Zerlegungsfunktionen.

[9]Ist $x_i \in X_1 \setminus X_U$, so gilt $\nu_i \in G$. Dies läßt sich induktiv zeigen: Gilt die Behauptung im Verlauf des Verfahrens für X_1 und kommt nun x_i zu X_1 neu hinzu wegen $\sigma_{ik} \in G$ mit $x_k \in X_1$, dann gilt $\nu_i = \sigma_{ik} \circ \nu_k \circ \sigma_{ik} \in G$. (Ist f unabhängig von einer Variablen x_k und symmetrisch in (x_i, x_k), so ist f auch unabhängig von x_i.)

Beweis:
Sei für alle $1 \leq i \leq k$
$$I_{X_i} = \{j \mid \exists x_j \in X_i\}$$
die Menge der Indizes von Variablen in X_i.

Zur Bestimmung der Anzahl verschiedener Zeilen von Z genügt es, alle Zeilen mit Indizes
$$j = int(\epsilon_1, \ldots, \epsilon_p),\ (\epsilon_1, \ldots, \epsilon_p) \in \{0,1\}^p$$
zu betrachten, die folgende Eigenschaft (*) haben:

- $\epsilon_i = 0$ für alle $i \in I_{X_1}$
- $\epsilon_i \leq \epsilon_j$ für alle $i, j \in I_{X_l}$, $2 \leq l \leq k$ mit $i < j$
  (d.h. wenn die Indizes der Variablen aus X_l
  $$i_1 < i_2 < \ldots < i_m$$
  sind, dann sind in
  $$\epsilon_{i_1}, \epsilon_{i_2}, \ldots, \epsilon_{i_m}$$
  die Einsen nach rechts sortiert.)

Es gibt genau $(|X_2|+1) \cdot \ldots \cdot (|X_k|+1)$ verschiedene Zeilen mit dieser Eigenschaft. Ist nun eine Zeile mit Index
$$j = int(\delta_1, \ldots, \delta_p),\ (\delta_1, \ldots, \delta_p) \in \{0,1\}^p$$
gegeben, wobei j Eigenschaft (*) nicht erfüllt, so kann man durch Nullsetzen von δ_i mit $i \in I_{X_1}$ und Vertauschen von δ_i mit $i \in I_{X_l}$ $(2 \leq l \leq k)$ einen Index j' konstruieren, der Eigenschaft (*) erfüllt. Da f unabhängig von den Variablen in X_1 ist und invariant gegenüber Vertauschungen der Variablen in X_l $(1 \leq l \leq k)$ ist, ändert sich der Funktionswert durch das Nullsetzen und die angegebenen Vertauschungen nicht. Die Zeilen mit den Indizes j und j' müssen identisch sein.

Die Gesamtzahl verschiedener Zeilen von Z beträgt also höchstens
$$(|X_2|+1) \cdot \ldots \cdot (|X_k|+1).$$
□

Ist eine Funktion $f \in B_n$ invariant gegenüber der Vertauschung zweier Eingangsvariablen x_i und x_j und führt man eine Zerlegung hinsichtlich einer Variablenaufteilung durch, bei der x_i und x_j in der gleichen Menge sind, so wird also schon dadurch die Anzahl der möglichen verschiedenen Zeilen (bzw. Spalten) der Zerlegungsmatrix reduziert. Es ist also wohl eine gute Heuristik, solche Eingangsvariablen bei einer Variablenaufteilung möglichst in die gleiche Menge zu plazieren. Man könnte nun evtl. zur Vermutung kommen, daß es (bei Vorgabe der

x_5	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
x_6	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
x_7	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
x_8	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
$x_1 x_2 x_3 x_4$																
0 0 0 0	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0
0 0 0 1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0
0 0 1 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0 0 1 1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0
0 1 0 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0 1 0 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0 1 1 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0 1 1 1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0
1 0 0 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1 0 0 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1 0 1 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1 0 1 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1 1 0 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1 1 0 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1 1 1 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1 1 1 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Abbildung 4.3: Zerlegungsmatrix bei $n = 4$

Mächtigkeit einer der beiden Variablenteilmengen) unter allen Variablenaufteilungen, die zu einer Zerlegung mit minimaler Anzahl von Zerlegungsfunktionen führen, immer eine Variablenaufteilung gibt, bei der x_i und x_j in der gleichen Menge sind. Dann könnte man bei der Suche nach optimalen Variablenaufteilungen auf alle Variablenaufteilungen verzichten, die x_i und x_j trennen. Dies ist aber leider *nicht* der Fall.

Um dies zu zeigen, wird eine Klasse von Funktionen definiert, die lediglich in *einem* Variablenpaar symmetrisch sind. Alle gleichmäßigen Zerlegungen mit geringster Anzahl von Zerlegungsfunktionen erfolgen aber hinsichtlich einer Variablenaufteilung, die gerade diese beiden Variablen trennt.

Sei $n \geq 3$, dann definiere

$$f(x_1, \ldots, x_n, x_{n+1}, \ldots, x_{2n}) = \begin{cases} 1 & \text{falls } \exists 1 \leq m \leq n \text{ und } \exists n+1 \leq l \leq 2n \text{ mit} \\ & x_i = 0 \text{ für } i \in \{1, \ldots, m\} \cup \{n+1, \ldots, l\} \\ & \text{und } x_i = 1 \text{ sonst.} \\ 0 & \text{sonst} \end{cases}$$

Eine Zerlegungsmatrix von f für $n = 4$ ist in Abbildung 4.3 angegeben.

Es gilt dann folgendes Lemma:

Lemma 4.8

1. *f ist symmetrisch in (x_1, x_{n+1}).*

2. *f ist nicht symmetrisch in x_i, x_j, falls $i \neq j$, $\{i,j\} \neq \{1, n+1\}$.*

3. *$\{\{x_1, \ldots, x_n\}, \{x_{n+1}, \ldots, x_{2n}\}\}$ ist eine gleichmächtige Variablenaufteilung, bei der die Anzahl von Zerlegungsfunktionen minimal wird.*

4. *Bei allen gleichmächtigen Variablenaufteilungen mit minimaler Anzahl von Zerlegungsfunktionen sind x_1 und x_{n+1} nicht in derselben Teilmenge.*

Beweis:

Zu 1.:

Falls $x_1 = 1$ oder $x_{n+1} = 1$, so ist nach Definition der Funktionswert 0. Also gilt

$$f(1, x_2, \ldots, x_n, 0, x_{n+2}, \ldots, x_{2n}) = f(0, x_2, \ldots, x_n, 1, x_{n+2}, \ldots, x_{2n}) = 0$$

für $x_2, \ldots, x_n, x_{n+2}, \ldots, x_{2n} \in \{0, 1\}$. $\Longrightarrow$ f symmetrisch in (x_1, x_{n+1})

Zu 2.:

Betrachte 2 Variablen x_i, x_j mit $\{i,j\} \neq \{1, n+1\}$.

1. Fall: $i, j \in \{1, \ldots, n\}$

Sei o.B.d.A. $i < j$.
Nach Definiton von f gilt

$$f(0, \ldots, \underbrace{0}_{i}, 1, \ldots, \underbrace{1}_{j}, \ldots, \underbrace{1}_{n}, 0, \ldots, 0) = 1,$$

aber

$$f(0, \ldots, \underbrace{1}_{i}, 1, \ldots, 1, \underbrace{0}_{j}, 1, \ldots, \underbrace{1}_{n}, 0, \ldots, 0) = 0.$$

$\Longrightarrow$ f nicht symmetrisch in (x_i, x_j)

2. Fall: $i, j \in \{n+1, \ldots, 2n\}$

Analog zu Fall 1.

3. Fall: $i \in \{1, \ldots, n\}$, $j \in \{n+1, \ldots, 2n\}$

Fall 3.1.: $x_i \neq x_n$, $x_j \neq x_{n+1}$

Nach Definiton von f gilt

$$f(0, \ldots, \underbrace{0}_{i}, \ldots, \underbrace{0}_{n}, \underbrace{0}_{n+1}, 1, \ldots, \underbrace{1}_{j}, \ldots, 1) = 1,$$

aber

$$f(0, \ldots, 0, \underbrace{1}_{i}, 0, \ldots, \underbrace{0}_{n}, \underbrace{0}_{n+1}, 1, \ldots, \underbrace{0}_{j}, 1, \ldots, 1) = 0.$$

$\Longrightarrow$ f nicht symmetrisch in (x_i, x_j)

Fall 3.2.: $x_i \neq x_n$, $x_i \neq x_1$, $x_j = x_{n+1}$

Nach Definiton von f gilt

$$f(0,\ldots,0,\underbrace{1}_{i},\ldots,\underbrace{1}_{n},\underbrace{0}_{n+1},\ldots,0) = 1,$$

aber

$$f(0,\ldots,0,\underbrace{0}_{i},1,\ldots,\underbrace{1}_{n},\underbrace{1}_{n+1},0,\ldots,0) = 0.$$

$\Longrightarrow f$ nicht symmetrisch in (x_i, x_j)

Fall 3.3.: $x_i = x_n$

Nach Definiton von f gilt

$$f(0,1,\ldots,\underbrace{1}_{n},\underbrace{0}_{n+1},\ldots,0) = 1,$$

aber

$$f(0,1,\ldots,1,\underbrace{0}_{n},\underbrace{0}_{n+1},0,\ldots,0,\underbrace{1}_{j},0\ldots,0) = 0.$$

$\Longrightarrow f$ nicht symmetrisch in (x_i, x_j)

Zu 3.:

- Die Zerlegungsmatrix Z von f hinsichtlich

$$\{\{x_1,\ldots,x_n\},\{x_{n+1},\ldots,x_{2n}\}\}$$

hat genau 2 verschiedene Zeilen und 2 verschiedene Spalten.
Zur Anzahl der verschiedenen Zeilen:

1. Fall:
Der Zeilenindex $(x_1,\ldots,x_n)$ ist von der Form $(0,\ldots,0,1,\ldots,1)$ (mit evtl. leerer Folge von Einsen).
Dann gilt: Ein Eintrag dieser Zeile ist genau dann 1, wenn der entsprechende Spaltenindex $(x_{n+1},\ldots,x_{2n})$ ebenfalls von dieser Form ist.

2. Fall:
Der Zeilenindex ist nicht von dieser Form $\Longrightarrow$ alle Einträge der Zeile sind 0.

Z hat also genau 2 verschiedene Zeilen.

Analog folgt, daß f genau 2 verschiedene Spalten hat.

- Es gibt keine Variablenaufteilung $\{X, Y\}$, so daß die zugehörige Zerlegungsmatrix nur eine Zeile bzw. nur eine Spalte hat.

  Offensichtlich ist f von allen Eingangsvariablen abhängig. Hätte die Zerlegungsmatrix hinsichtlich $\{X, Y\}$ nur eine Zeile, so wäre aber f von allen Variablen in X unabhängig, hätte sie nur eine Spalte, so wäre f von den Variablen in Y unabhängig.

$\Longrightarrow$ $\{\{x_1, \ldots, x_n\}, \{x_{n+1}, \ldots, x_{2n}\}\}$ ist eine Variablenaufteilung, bei der die Anzahl von Zerlegungsfunktionen minimal wird.

Zu 4.:

Zeige: Falls bei einer gleichmächtigen Variablenaufteilung $\{X, Y\}$ x_1 und x_{n+1} in derselben Teilmenge sind, dann hat die Zerlegungsmatrix hinsichtlich $\{X, Y\}$ mehr als 2 verschiedene Zeilen und mehr als 2 verschiedene Spalten.

Seien

$$X = \{x_{i_1}, \ldots, x_{i_n}\}, \text{ und } Y = \{x_{i_{n+1}}, \ldots, x_{i_{2n}}\}$$

und o.B.d.A. seien $x_{i_1} = x_1$, $x_{i_2} = x_{n+1}$.

Sei Z die Zerlegungsmatrix von f hinsichtlich $\{X, Y\}$.

- Anzahl der verschiedenen Zeilen von Z:
  Falls $x_1 = 1$ oder $x_{n+1} = 1$, dann $f(x_1, \ldots, x_{2n}) = 0$.
  $\Longrightarrow$ Alle Zeilen mit Indizes

  $$(1, 1, \ldots), (1, 0, \ldots), \text{ bzw. } (0, 1, \ldots)$$

  sind konstant 0.
  Noch zu zeigen: Unter den Zeilen mit den Indizes $(0, 0, x_{i_3}, \ldots, x_{i_n})$ gibt es mindestens 2 verschiedene, die nicht konstant 0 sind.
  Betrachte dazu die Zeilen mit den Indizes $(0, 0, 1, \ldots, 1)$ und $(0, \ldots, 0)$.
  Die Zeilen sind nicht konstant 0, da

  - Bei der Zeile mit Index $(0, \ldots, 0)$ der Eintrag in der Spalte mit Index $(0, \ldots, 0)$ 1 ist.
  - Bei der Zeile mit Index $(0, 0, 1, \ldots, 1)$ der Eintrag in der Spalte mit Index $(1, \ldots, 1)$ 1 ist.

  Es wird nun gezeigt, daß sich die beiden Zeilen auf jeden Fall in ihrem Eintrag in der Spalte mit Index $(0, \ldots, 0)$ oder im Eintrag in der Spalte mit Index $(1, \ldots, 1)$ unterscheiden.

  <u>1. Fall</u>: $Z_{int(0,\ldots,0) int(1,\ldots,1)} = 0$

  Es gilt $Z_{int(0,0,1,\ldots,1) int(1,\ldots,1)} = 1$. Also sind zwei verschiedene Zeilen, die nicht konstant 0 sind gefunden.

  <u>2. Fall</u>: $Z_{int(0,\ldots,0) int(1,\ldots,1)} = 1$

  Dann gilt nach Definition von f:

  $$Y = \{x_{i_{n+1}}, \ldots, x_{i_{2n}}\} = \{x_m, \ldots, x_n, x_l, \ldots, x_{2n}\}$$

für $2 \leq m \leq n$, $n + 2 \leq l \leq 2n$.
Es gilt außerdem $m \geq 3$ oder $l \geq n + 3$, denn wäre $m = 2$ *und* $l = n + 2$, dann wäre

$$n = |\{x_{i_{n+1}}, \ldots, x_{i_{2n}}\}| = n - 1 + n - 1 = 2n - 2.$$

Dies kann aber nicht sein, da $n \geq 3$.
Sei o.B.d.A.: $m \geq 3$
$\Longrightarrow \quad x_2 \in X$, $x_n \in Y$.
$\Longrightarrow \quad Z_{int(0,0,1,\ldots,1)int(0,\ldots,0)} = 0$, da $f(x_1, \ldots, x_{2n}) = 0$ für $x_1 = 0$, $x_2 = 1$, $x_n = 0$.
Es gilt $Z_{int(0,\ldots,0)int(0,\ldots,0)} = 1$. Also sind zwei verschiedene Zeilen, die nicht konstant 0 sind, gefunden.

Insgesamt gibt es also mehr als 2 verschiedene Zeilen der Zerlegungsmatrix.

- Anzahl der verschiedenen Spalten von Z:
Bei der Betrachtung der Zeilenanzahl wurden 2 Zeilen angegeben, die sich auf jeden Fall in ihrem Eintrag in der Spalte mit Index $(0, \ldots, 0)$ oder im Eintrag in der Spalte mit Index $(1, \ldots, 1)$ unterscheiden. Folglich sind diese beiden Spalten verschieden und wegen

$$Z_{int(0,\ldots,0)int(0,\ldots,0)} = Z_{int(0,0,1,\ldots,1)int(1,\ldots,1)} = 1$$

sind sie nicht konstant 0.
Es gibt aber auch Spalten, die konstant 0 sind:

$$\text{Wegen } n \geq 3 \; \exists x_{i_j}, x_{i_k} \in Y \text{ mit } x_{i_j} \neq x_{i_k} \text{ und}$$

$$x_{i_j}, x_{i_k} \in \{x_1, \ldots, x_n\} \text{ oder } x_{i_j}, x_{i_k} \in \{x_{n+1}, \ldots, x_{2n}\}$$

Sei o.B.d.A.: $i_j < i_k$ und $i_j, i_k \in \{1, \ldots, n\}$.
$\Longrightarrow \quad$ Die Spalte mit dem Index

$$(1, \ldots, \underbrace{1}_{i_j}, \ldots, 1, \underbrace{0}_{i_k}, 1, \ldots, 1)$$

ist konstant 0, da

$$f(x_1, \ldots, x_{2n}) = 0,$$

falls $\exists o, p \in \{1, \ldots, n\}, o < p$ mit $x_o = 1$, $x_p = 0$.
$\Longrightarrow \quad$ Insgesamt gibt es also mehr als 2 verschiedene Spalten der Zerlegungsmatrix.

$\square$

4.3 Partielle Funktionen

Bisher wurden nur Zerlegungen totaler Funktionen behandelt. Bei praktischen Problemen treten allerdings häufig partielle Funktionen auf (d.h. Funktionen, deren don't care–Menge nicht leer ist), da die Schaltkreise, die diese partiellen Funktionen realisieren sollen, beispielsweise in ein größeres System eingebettet werden und man daher gewisse Eingabevektoren von vornherein ausschliessen kann oder da im praktischen Problem bei einer Funktion mit mehreren Ausgängen für bestimmte Eingaben der Funktionswert nur an einem Teil der Ausgänge interessant ist.

Auch bei der rekursiven Ausnutzung von Zerlegungen bei der Logiksynthese können partielle Funktionen auftreten. Selbst wenn die ursprünglich zu realisierende Funktion total ist, können bei der rekursiven Behandlung der Zerlegungsfunktionen bzw. der Zusammensetzungsfunktion partielle Funktionen vorkommen. Ist bei einer einseitigen Zerlegung hinsichtlich einer Variablenteilmenge z. B. die Anzahl der Zeilen der zugehörigen Zerlegungsmatrix keine Zweierpotenz, so müssen die Zerlegungsfunktionen nicht alle möglichen Ausgangskombinationen annehmen (auch wenn die Anzahl der Zerlegungsfunktionen minimal gewählt wird). Die Zusammensetzungsfunktion kann dann partiell sein. Es ist weiterhin denkbar, auch Zerlegungsfunktionen als partielle Funktionen zu wählen.

Ein Ω–Schaltkreis, der eine partielle Funktion f realisiert, definiert eine totale Funktion, die eine Erweiterung von f darstellt. Wie in dieser totalen Funktion die bisher undefinierten Funktionswerte von f belegt werden, ist unerheblich. Allerdings haben im allgemeinen verschiedene totale Erweiterungen von f verschiedene Komplexität. Auch die minimale Anzahl von Zerlegungsfunktionen kann bei verschiedenen Erweiterungen von f verschieden sein.

Es ist daher sinnvoll, bei der Logiksynthese die undefinierten Funktionswerte einer partiellen Funktion nicht beliebig festzulegen, sondern nach günstigen Erweiterungen der partiellen Funktion zu suchen.

4.3.1 Einseitige Zerlegungen

Zunächst wird die Behandlung partieller Funktionen bei einseitigen Zerlegungen betrachtet.

Analog zur Gleichheit von Zerlegungsmatrixzeilen bei totalen Funktionen wird hier die Kompatibilität von Zerlegungsmatrixzeilen definiert.

Definition 4.6 (Kompatible Zeilen einer Zerlegungsmatrix)
Sei $f \in S(D)$, $D \subseteq \{0,1\}^n$ eine Funktion mit den Eingangsvariablen $x_1, \ldots, x_p$, $y_1, \ldots, y_q$. Sei Z die Zerlegungsmatrix von f hinsichtlich $\{x_1, \ldots, x_p\}$.
Sind

$$x^{(1)} = (x_1^{(1)}, \ldots, x_p^{(1)}) \textit{ und } x^{(2)} = (x_1^{(2)}, \ldots, x_p^{(2)}) \in \{0,1\}^p$$

und $i = int(x^{(1)})$, $j = int(x^{(2)})$, *so heißen die Zeilen von* Z *mit den Indizes* i *und* j *kompatibel (in Zeichen:* $x^{(1)} \sim x^{(2)}$*), wenn es keinen Spaltenindex* $0 \leq k \leq 2^q - 1$ *gibt mit*

$$(Z_{ik} = 0 \text{ und } Z_{jk} = 1) \quad \text{oder}$$
$$(Z_{ik} = 1 \text{ und } Z_{jk} = 0).$$

(Mit anderen Worten:

$$x^{(1)} \sim x^{(2)} \quad \Leftrightarrow$$
$$\nexists y \in \{0,1\}^q \text{ mit } (x^{(1)}, y), (x^{(2)}, y) \in D \text{ und } f(x^{(1)}, y) \neq f(x^{(2)}, y).)$$

„ $\sim$ “ ist *keine* Äquivalenzrelation auf $\{0,1\}^p$.

Dies zeigt das folgende kleine Beispiel einer Zerlegungsmatrix:

0	*	1
1	0	*
2	0	0
3	0	0

Zeilen 0 und 1 sind kompatibel, Zeilen 1 und 2 sind kompatibel, aber Zeilen 0 und 2 sind *nicht* kompatibel.

Analog zur Anzahl verschiedener Zeilen einer Zerlegungsmatrix bei totalen Funktionen wird hier folgende Bezeichnung eingeführt:

Bezeichnung 14 *Sei* $f \in S(D)$, $D \subseteq \{0,1\}^n$ *mit den Eingangsvariablen* $x_1, \ldots, x_p, y_1, \ldots, y_q$. *Sei* Z *die Zerlegungsmatrix von* f *hinsichtlich der Variablenteilmenge* $X := \{x_1, \ldots, x_p\}$.

Dann wird die minimale Anzahl von Mengen, in die $\{0,1\}^p$ *partitioniert werden kann, so daß je 2 Elemente der gleichen Menge (bzgl.* $\sim$*) kompatibel sind, mit* $vkk(X, f)$ *bezeichnet.*

Es ist nun wie bei totalen Funktionen leicht, ein Kriterium dafür anzugeben, daß zu einer vorgebenen Variablenteilmenge eine Zerlegung mit einer festen Anzahl von Zerlegungsfunktionen existiert. (Ein entsprechender Satz wurde schon von Karp in [Kar63] bewiesen.)

Satz 4.5 *Sei* $f \in S(D)$, $D \subseteq \{0,1\}^n$ *eine Funktion mit den Eingangsvariablen* $x_1, \ldots, x_p, y_1, \ldots, y_q$. *Dann gibt es genau dann eine einseitige Zerlegung von* f *hinsichtlich der Variablenteilmenge* $X = \{x_1, \ldots, x_p\}$, *so daß für alle* $(x_1, \ldots, x_p, y_1, \ldots, y_q)$ *aus* D *gilt:*

$$f(x_1, \ldots, x_p, y_1, \ldots, y_q) = g(\alpha_1(x_1, \ldots, x_p), \ldots, \alpha_r(x_1, \ldots, x_p), y_1, \ldots, y_q),$$

wenn

$$r \geq \log(vkk(X, f))$$

Beweis:
Der Beweis erfolgt genau analog zum Beweis von Satz 4.1. Es ist lediglich jeweils die Gleichheit von Zeilen durch „Kompatibilität von Zeilen“ zu ersetzen.

□

Sind die Zeilen einer Zerlegungsmatrix Z hinsichtlich X so in verschiedene Mengen partitioniert, daß sämtliche Zeilen in einer Menge paarweise kompatibel sind, so kann man die don't care–Stellen („*“) so belegen, daß alle Zeilen in einer Menge gleich sind. (Es gibt in einer Menge ja kein Paar von Zeilen, so daß in der einen Zeile an einer bestimmten Position eine 1 steht, in der anderen Zeile an dieser Position eine 0.) Ist die Anzahl der Mengen in einer solchen Partition minimal, so erhält man auf diese Weise eine Zerlegungsmatrix zu einer totalen Funktion f', wobei $vkk(X, f) = vz(X, f')$.

Sucht man zu einer partiellen Funktion f eine Variablenteilmenge X mit Mächtigkeit p, so daß die Anzahl der Zerlegungsfunktionen, die bei einer einseitigen Zerlegung hinsichtlich X nötig sind, minimal ist, so kann man hier ähnlich wie bei totalen Funktionen für alle p–elementigen Variablenteilmengen X $vkk(X, f)$ bestimmen und dann die Teilmenge X auswählen, für die $vkk(X, f)$ minimal ist.

Das Problem, das dabei auftritt, besteht darin, daß die Aufgabe, bei einer gegebenen Zerlegungsmatrix die minimale Anzahl von Kompatibilitätsklassen $vkk(X, f)$ zu bestimmen, *wesentlich* schwieriger ist als die Bestimmung der Anzahl verschiedener Zeilen der Matrix. Es stellt sich heraus, daß dieses Problem sogar NP–hart ist. Es ist also sehr unwahrscheinlich, daß man einen Algorithmus finden kann, der $vkk(X, f)$ berechnet und dessen Laufzeit polynomiell in der Größe der Zerlegungsmatrix ist.

Folgendes Problem ist zu lösen:

> **Problem der einseitigen Kommunikationsminimierung (EKM)**
>
> *Gegeben* ist eine partielle Funktion $f \in S(D)$, $D \subseteq \{0, 1\}^n$ in den Eingangsvariablen $x_1, \ldots, x_p$, $y_1, \ldots, y_q$ mit der Zerlegungsmatrix Z hinsichtlich $X = \{x_1, \ldots, x_p\}$.
>
> *Gesucht* ist die minimale Anzahl von Zerlegungsfunktionen bei einer einseitigen Zerlegung von f hinsichtlich X, d.h. gesucht ist $\lceil \log(vkk(X, f)) \rceil$.

Es gilt:

Satz 4.6 *Das Problem* **EKM** *ist NP–hart.*

Satz 4.6 wird bewiesen durch Angabe einer Polynomzeittransformation vom NP–vollständigen Problem „Partition into Cliques“ (PC) nach EKM.

Das Problem PC lautet (siehe [GJ79], S. 193):

Partition into Cliques (PC)

Gegeben.: Ein Graph $G = (V, E)$ und eine natürliche Zahl $K \leq |V|$.

Gesucht.: Kann V partitioniert werden in $k \leq K$ disjunkte Mengen $V_1, \ldots, V_k$, so daß für $1 \leq i \leq k$ der Teilgraph, der alle Knoten aus V_i umfaßt, ein vollständiger Graph ist?

Das Problem bleibt NP–vollständig, wenn man die Grenze K auf Zweierpotenzen beschränkt. Also ist auch folgendes Problem NP–vollständig:

Problem PC'

Gegeben.: Ein Graph $G = (V, E)$ und eine natürliche Zahl $K = 2^m$ für $m \in \mathbf{N}$, $K \leq |V|$.

Gesucht.: Kann V partitioniert werden in $k \leq K$ disjunkte Mengen $V_1, \ldots, V_k$, so daß für $1 \leq i \leq k$ der Teilgraph, der alle Knoten aus V_i umfaßt, ein vollständiger Graph ist?

Beweisskizze:
Beweis, daß PC' NP–hart, durch Polynomzeittransformation von PC nach PC': Ist bei einer Instanz des Problems PC die natürliche Zahl K keine Zweierpotenz, so wähle die nächsthöhere Zweierpotenz 2^m und füge zum Graphen G $2^m - K$ zusätzliche Knoten hinzu, die weder Quelle noch Ziel einer Kante sind. Der resultierende Graph hat genau dann eine Partition in $k' \leq 2^m$ vollständige Teilgraphen, wenn der ursprüngliche Graph eine Partition in $k \leq K$ vollständige Teilgraphen hat. □

Nun kann Satz 4.6 bewiesen werden. Die Idee des Beweises besteht darin, daß man die Kompatibilitätsrelation $\sim$ auf Zerlegungsmatrixzeilen als einen Graphen betrachtet. Das Problem, die minimale Anzahl von Kompatibilitätsklassen auf den Zerlegungsmatrixzeilen zu bestimmen, ist äquivalent zum Problem, eine Partition des Graphen in vollständige Teilgraphen zu bestimmen. Die Einzelheiten sind in folgendem Beweis zu finden:

Beweis:
Gegeben sei eine Instanz eines PC'–Problems bestehend aus einem Graphen $G = (V, E)$ und einer natürlichen Zahl $K \leq |V|$ mit $K = 2^m$ für $m \in \mathbf{N}$

Zu G kann in Polynomzeit eine partielle Funktion f bestimmt werden, so daß bei einseitiger Zerlegung von f hinsichtlich einer Variablenteilmenge $X = \{x_1, \ldots, x_p\}$ $vkk(X, f)$ gleich der minimalen Anzahl von vollständigen disjunkten Teilgraphen von G ist.
Zur Konstruktion von f:
Sei p minimal mit $2^p \geq |V|$. Definiere $f : \{0,1\}^{2p} \rightsquigarrow \{0,1\}$ durch Angabe einer Zerlegungsmatrix Z hinsichtlich $X = \{x_1, \ldots, x_p\}$. Nehme dazu der Einfachheit halber o.B.d.A. an, daß $V = \{0, \ldots, |V| - 1\}$.

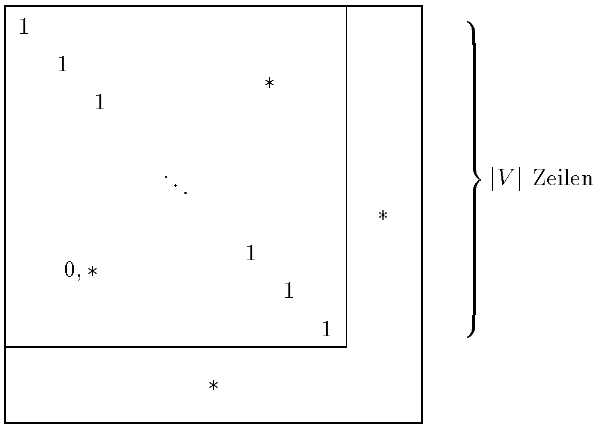
Es gelte

$$Z_{ij} = * \text{ für } i \geq |V| \text{ oder } j \geq |V|.$$

Für alle $0 \leq i, j \leq |V| - 1$ wird definiert:

$$Z_{ij} = \begin{cases} *, \text{ falls } & i < j \\ 1, \text{ falls } & i = j \\ 0, \text{ falls } & i > j \text{ und } \{i,j\} \notin E \\ *, \text{ falls } & i > j \text{ und } \{i,j\} \in E \end{cases}$$

Z hat also folgende Gestalt:



Seien $i, j \in \{0, \ldots, |V| - 1\}$ und sei o.B.d.A. $i > j$.
Für alle Spalten $k \neq j$ gilt:

$$Z_{ik} = Z_{jk} = 0 \text{ oder } Z_{ik} = * \text{ oder } Z_{jk} = *.$$

Es gilt außerdem: $Z_{jj} = 1$.
Also sind Zeilen i und j genau dann inkompatibel (bzw. $bin_p(i) \not\sim bin_p(j)$), wenn

$$Z_{ij} = 0 \overset{\text{Def. von } f}{\Longleftrightarrow} \{i,j\} \notin E$$

bzw.

$$\{i,j\} \in E \iff bin_p(i) \sim bin_p(j) \; \forall i, j \in \{0, \ldots, |V| - 1\}.$$

Also bilden die Knoten $i_1, \ldots i_m$ genau dann einen vollständigen Teilgraphen, wenn die Zeilen mit den Indizes $i_1, \ldots i_m$ paarweise kompatibel sind.

Zeilen mit Indizes $i \geq |V|$ sind zu allen anderen Zeilen kompatibel. Also gilt:

- Falls es eine Partition von $V = \{0, \ldots, |V| - 1\}$ in $k \leq K$ disjunkte Mengen $V_1, \ldots, V_k$ gibt, so daß die durch die Knoten aus V_i $(1 \leq i \leq k)$ induzierten Teilgraphen vollständige Graphen sind, dann ist

  $$\{V_1, \ldots, V_{k-1}, V_k \cup \{|V|, \ldots, 2^p - 1\}\}$$

  eine Partition der Zeilenindizes, bei der die einzelnen Teilmengen nur Indizes kompatibler Zeilen enthalten. Es gilt dann: $vkk(X, f) \leq k \leq K = 2^m$ und es gibt nach Lemma 4.5 eine Zerlegung hinsichtlich X mit $m' \leq m$ Zerlegungsfunktionen.

- Gibt es eine Zerlegung von f hinsichtlich X mit $m' \leq m$ Zerlegungsfunktionen, so gilt nach Lemma 4.5:
$$vkk(X,f) \leq 2^{m'} \leq 2^m = K$$
Es gibt also eine Partition $\{K_1, \ldots, K_k\}$ der Zeilenindizes $\{0, \ldots 2^p - 1\}$ mit $k \leq K$, so daß die Zeilen mit Indizes aus einer der Mengen K_i $(1 \leq i \leq k)$ paarweise kompatibel sind. Dann gibt es auch eine Partition von V in Mengen
$$V_1 = K_1 \cap V, \ldots, v_k = K_k \cap V,$$
so daß die durch die Knoten aus V_i induzierten Teilgraphen vollständige Graphen sind.

Insgesamt gilt: Die in Polynomzeit konstruierbare Funktion f hat genau dann eine Zerlegung hinsichtlich X mit $m' \leq m$ Zerlegungsfunktionen, wenn sich die Knoten von G so partitionieren lassen, daß sich $k \leq K = 2^m$ vollständige Teilgraphen ergeben[1].

□

Anhand des obigen Beweises wird auch ein Verfahren deutlich, wie man für eine Zerlegung von f hinsichtlich X $vkk(X,f)$ berechnen bzw. approximieren kann:

- Bestimme die zugehörige Zerlegungsmatrix.
- Bestimme die Kompatibilitätsrelation $\sim$ auf $\{0,1\}^p$.
- Interpretiere $\sim$ als Kantenmenge eines Graphen mit Knoten aus $\{0,1\}^p$.
- Bestimme $vkk(X,f)$ als die minimale Zahl k, so daß $\{0,1\}^p$ in Mengen $V_1, \ldots, V_k$ partitioniert werden kann, wobei die Teilgraphen aller Knoten in V_i jeweils vollständige Graphen sind. Dies kann mit einer Heuristik zur Lösung des bekannten Problems „Partition into Cliques“ geschehen.

4.3.2 Zweiseitige Zerlegungen

Gemäß Lemma 4.2 kann man bei totalen Funktionen die minimale Anzahl von Zerlegungsfunktionen anhand der Anzahl der verschiedenen Zeilen und verschiedenen Spalten einer Zerlegungsmatrix bestimmen. Definiert man wie auf den Zeilen einer Zerlegungsmatrix einer partiellen Funktion auch auf den Spalten eine Kompatibilitätsrelation, so gilt die analoge Aussage zu Lemma 4.2 allerdings nicht. Bei einer Zerlegung von f hinsichtlich der Variablenaufteilung $\{X, Y\}$ ist die minimale Anzahl der Zerlegungsfunktionen *nicht* durch $\lceil \log(vkk(X,f)) \rceil + \lceil \log(vkk(Y,f)) \rceil$ gegeben. Das folgende kleine Beispiel zeigt den Grund dafür:

Beispiel 3:

[1]Wenn man in Polynomzeit die minimale Anzahl von Zerlegungsfunktionen bei Zerlegung von f hinsichtlich X berechnen könnte, so könnte man natürlich auch berechnen, ob es eine Zerlegung mit $m' \leq m$ Zerlegungsfunktionen gibt.

*	1
0	*

Die beiden Zeilen und die beiden Spalten der Matrix sind jeweils kompatibel. Es gibt aber keine Ersetzung der don't care–Stellen durch Elemente aus $\{0,1\}$, so daß *sowohl* die Zeilen gleich werden *als auch* die Spalten gleich werden.

Auch im Fall zweiseitiger Zerlegungen ist das Problem, bei gegebener Variablenaufteilung $\{X,Y\}$ die minimale Anzahl von Zerlegungsfunktionen zu finden, die nötig ist, um eine Zerlegung hinsichtlich $\{X,Y\}$ durchzuführen, NP–hart. Es handelt sich genauer um folgendes Problem:

> **Problem der zweiseitigen Kommunikationsminimierung (ZKM)**
>
> *Gegeben* ist eine partielle Funktion $f \in S(D)$, $D \subseteq \{0,1\}^n$ in den Eingangsvariablen $x_1,\dots,x_p$, $y_1,\dots,y_q$ mit der Zerlegungsmatrix Z hinsichtlich $\{X,Y\} = \{\{x_1,\dots,x_p\}, \{y_1,\dots,y_q\}\}$.
>
> *Gesucht* ist die minimale Anzahl von Zerlegungsfunktionen bei einer zweiseitigen Zerlegung von f hinsichtlich $\{X,Y\}$.

Es gilt:

Satz 4.7 *Das Problem* **ZKM** *ist NP–hart.*

Satz 4.7 wird bewiesen durch Angabe einer Polynomzeittransformation von einem NP–vollständigen Problem **BIP** nach **ZMK**. BIP ist ein Spezialfall des Problems der ganzzahligen Programmierung:

> **Problem BIP**
>
> *Gegeben* ist eine Matrix $A \in \{0,1\}^{m\times n}$ und ein Vektor $b = (\underbrace{1,\dots,1}_{m \text{ Mal}})^T$. *Gesucht* ist eine binäre Lösung $x \in \{0,1\}^n$ von $Ax = b$.

BIP ist NP–vollständig (vergleiche [LP81]).
Auch eine eingeschränkte Form von BIP ist NP–vollständig:

> **Problem BIP'**
>
> *Gegeben* ist eine Matrix $A \in \{0,1\}^{n\times n}$ mit $n = 2^k - 1$, $k \in \mathbf{N}$ und ein Vektor $b = (\underbrace{1,\dots,1}_{n \text{ Mal}})^T$. *Gesucht* ist eine binäre Lösung von $Ax = b$.

<u>Beweisskizze</u>:
Beweis, daß BIP' NP–hart, durch Polynomzeittransformation von BIP nach BIP':

Ist bei einer Instanz des Problems BIP eine Matrix $A \in \{0,1\}^{m \times n}$ gegeben, so wähle k minimal, so daß $2^k - 1 \geq n+1$ und $2^k - 1 \geq m$. Konstruiere zu A eine Matrix A', indem zunächst an A $2^k - 1 - n$ 0-Spalten angehängt werden (also mindestens eine) und dann $2^k - 1 - m$ Zeilen der Form $(0, \dots, 0, 1)$ angehängt werden. b' ergibt sich aus b durch Anhängen von $2^k - 1 - m$ Einsen. Man sieht leicht, daß $Ax = b$ genau dann eine binäre Lösung hat, wenn $A'x' = b'$ eine binäre Lösung hat. □

Nun kann Satz 4.7 bewiesen werden:

Beweis:
Gegeben sei eine Instanz eines BIP'–Problems, d.h. eine Matrix $A \in \{0,1\}^n$ mit $n = 2^k - 1$.

Zu A wird in Polynomzeit die Zerlegungsmatrix Z einer partiellen Funktion f mit $4k + 2$ Eingangsvariablen bestimmt. Es handelt sich um die Zerlegungsmatrix zu einer Zerlegung hinsichtlich einer Variablenaufteilung $\{X, Y\}$ in zwei gleichmächtige Variablenteilmengen.

Z wird durch Bild 4.4 definiert. Die Matrix ist in verschiedene Blöcke eingeteilt, an den Rändern der Matrix ist die Größe der Blöcke angegeben.

Bild 4.5 zeigt diesselbe Matrix. Um leichter argumentieren zu können, werden hier den verschiedenen Blöcken Namen zugeordnet.[2]

Die Gesamtzahl der Zeilen von Z beträgt

$$1 + 2n + n^2 + (n^2 + n + 1) + n = 2(n+1)^2 = 2(2^{2k}) = 2^{2k+1},$$

die Gesamtzahl der Spalten beträgt

$$4n + 3 + [2(n+1)^2 - 4n - 3] = 2(n+1)^2 = 2^{2k+1}.$$

Z ist also Zerlegungsmatrix einer partiellen Funktion f mit $4k + 2$ Eingangsvariablen.

Hilfsbehauptung 1: Z hat bei jeder Ersetzung der * durch 0 oder 1 *mindestens*

- $(5n + 4) - n = 4(n+1) = 2^{k+2}$ verschiedene Spalten
- $(n^2 + 3n + 1) - n = (n+1)^2 = 2^{2k}$ verschiedene Zeilen

[2] Die Matrizen $3S$, 0_S, O_{Z_1}, O_{Z_2}, E_{n+4} sind technische Details. Sie gewährleisten, daß sowohl die Gesamtzahl der Zeilen bzw. Spalten als auch die Minimalzahlen verschiedener Zeilen und Spalten Zweierpotenzen sind. Wichtig ist die Rolle der Matrizen A und A'. A ist die Matrix des ursprünglichen Problems, A' eine Matrix, die nur aus * besteht. Ist $(x_1, \dots, x_n)$ binäre Lösung von $Ax = b$, so erhält man eine Belegung der * durch Einträge aus $\{0,1\}$ mit minimaler Anzahl von Zeilen und Spalten, indem Spalte i von A' mit dem Nullvektor belegt wird, falls $x_i = 0$, bzw. mit der i. Spalte von A, falls $x_i = 1$. Die Matrizen R_i und E_i $(1 \leq i \leq n)$ dienen intuitiv zum „Test, ob in einer bestimmten Zeile von A' nach Ersetzen der * die Summe der Spalteneinträge genau 1 ist".

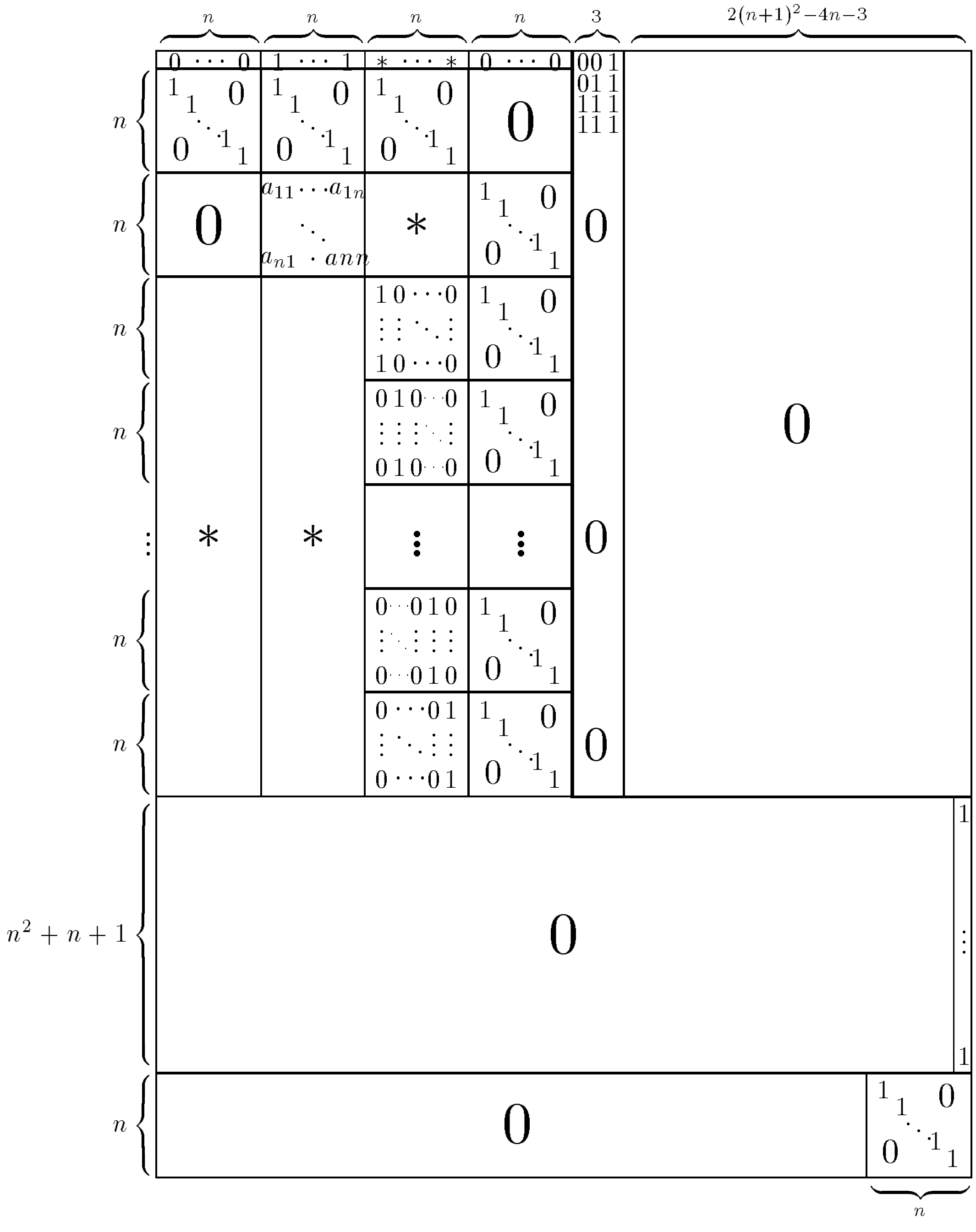


Abbildung 4.4: Zerlegungsmatrix Z

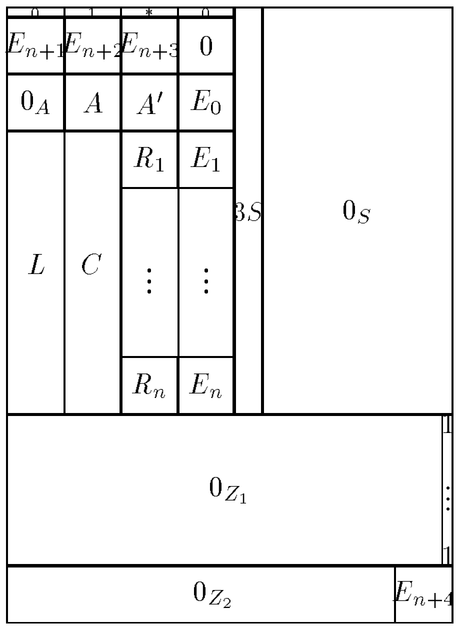


Abbildung 4.5: Zerlegungsmatrix Z

Beweis:

1. Spalten
   - Alle Spalten, die durch E_{n+1} verlaufen, sind untereinander verschieden.
   - Alle Spalten, die durch E_{n+2} verlaufen, sind untereinander verschieden.
   - Alle Spalten, die durch E_0 verlaufen, sind untereinander verschieden.
   - Wegen der 1. Zeile sind alle Spalten durch E_{n+1} verschieden von allen Spalten durch E_{n+2}.
   - Alle Spalten, die durch E_0 verlaufen, sind verschieden von Spalten durch E_{n+1}, E_{n+2}.

   Also muß es in den ersten 4 Blöcken mindestens $3n$ verschiedene Spalten geben.

   - Die 3 Spalten, die durch $3S$ verlaufen, sind untereinander verschieden und verschieden von allen anderen.
   - Die Nullspalte durch 0_S ist verschieden von allen anderen.
   - Alle Spalten, die durch E_{n+4} verlaufen, sind untereinander verschieden und verschieden von allen anderen.

   Insgesamt gibt es also mindestens $3n + 4 + n = 4(n + 1)$ verschiedene Spalten.

(Auch die Spalten durch E_{n+3} sind paarweise verschieden. Die einzige Möglichkeit, Spalten durch Ersetzungen von * durch 0 oder 1 gleich zu machen, besteht bei Spalten durch E_{n+1} mit Spalten durch E_{n+3} bzw. bei Spalten durch E_{n+2} mit Spalten durch E_{n+3}.)

2. Zeilen

   Wie man leicht sieht, sind alle Zeilen, die durch 0_S verlaufen und nicht durch A verlaufen paarweise verschieden. Es handelt sich dabei um $1 + n + n^2$ Zeilen. Die n verschiedenen Zeilen durch 0_{Z_1} und 0_{Z_2} sind verschieden von allen anderen. Insgesamt gibt es also mindesten $n^2 + 2n + 1 = (n+1)^2$ verschiedene Zeilen.

   (Die einzige Möglichkeit, Zeilen durch Ersetzungen von * durch 0 oder 1 gleich zu machen, besteht bei Zeilen durch A'. Sie können mit Zeilen durch $R_1, \ldots, R_n$ durch Ersetzen von * zur Übereinstimmung gebracht werden, jedoch jede Zeile durch A' nur mit maximal einer Zeile durch $R_1, \ldots, R_n$.)

Hilfsbehauptung 2: Falls es eine Ersetzung von * durch Elemente aus $\{0, 1\}$ gibt, so daß die resultierende Matrix $\overline{Z}$ nur $4(n+1)$ verschiedene Spalten und nur $(n+1)^2$ verschiedene Zeilen hat, dann hat $Ax = b$ eine binäre Lösung.

Beweis:

- Alle Spalten, die durch E_{n+3} verlaufen, sind untereinander verschieden. Hat $\overline{Z}$ $4(n+1)$ verschieden Spalten, so müssen nach dem Beweis von Hilfsbehauptung 1 alle Spalten durch $\overline{E_{n+3}}$[‡] mit einer Spalte durch $\overline{E_{n+1}}$ oder durch $\overline{E_{n+2}}$ übereinstimmen. Die i. Spalte von $\overline{A'}$ muß dann entweder gleich der i. Spalte von A oder gleich dem 0–Vektor (der i. Spalte von 0_A) sein (vergleiche darüberliegende Einheitsmatrizen und 1. Zeile!).
- Hat $\overline{Z}$ $(n+1)^2$ verschiedene Zeilen, so muß nach dem Beweis von Hilfsbehauptung 1 jede Zeile durch $\overline{A'}$ mit genau einer Zeile durch $R_1, \ldots, R_n$ übereinstimmen.
  Daraus folgt, daß jede Zeile von $\overline{A'}$ genau eine 1 enthält.

Insgesamt steht also in $\overline{A'}$ eine Kopie von A, bei der allerdings einige Spalten evtl. durch 0–Spalten ersetzt sind und zwar gerade so, daß jede Zeile von $\overline{A'}$ genau eine 1 enthält.
$\Longrightarrow$ Man erhält eine Lösung von $Ax = b$, indem man setzt

$$x_i = \begin{cases} 1, & \text{falls die } i\text{. Spalte von } \overline{A'} \text{ gleich der } i\text{. Spalte von } A \\ 0, & \text{falls die } i\text{. Spalte von } \overline{A'} \text{ gleich der } i\text{. Spalte von } 0_A, \\ & \text{d.h. falls die } i\text{. Spalte von } \overline{A'} \text{ nur Nullen enthält} \end{cases}$$

[‡]Ist B ein Block der Matrix Z, so wird der entsprechende Block in Matrix $\overline{Z}$ mit $\overline{B}$ bezeichnet.

Hilfsbehauptung 3: Falls es eine binäre Lösung von $Ax = b$ gibt, so gibt es eine Ersetzung von * in Z durch Elemente aus $\{0, 1\}$, so daß die resultierende Matrix $\overline{Z}$ nur $4(n + 1)$ verschiedene Spalten und nur $(n + 1)^2$ verschiedene Zeilen hat.

Beweis:
Gegeben sei eine binäre Lösung $x = (x_1, \ldots, x_n)$ von $Ax = b$. Konstruiere daraus nun eine Ersetzung der * in Z, so daß die resultierende Matrix $\overline{Z}$ nur $4(n+1)$ verschiedene Spalten und nur $(n+1)^2$ verschiedene Zeilen hat.

- Falls $x_i = 1$, so ersetze Spalte i in A' durch die i. Spalte von A. Ersetze außerdem die i. Spalte von C durch die Aneinanderreihung der i. Spalten von $R_1, \ldots, R_n$ (*). Weiterhin ersetze den i. * in der 1. Zeile von Z durch 1.
  D.h. bringe insgesamt Spalte $n + i$ und Spalte $2n + i$ von Z zur Übereinstimmung.
- Falls $x_i = 0$, so ersetze Spalte i in A' durch die i. Spalte von 0_A. Ersetze außerdem die i. Spalte von L durch die Aneinanderreihung der i. Spalten von $R_1, \ldots, R_n$. Weiterhin ersetze den i. * in der 1. Zeile von Z durch 0.
  Insgesamt werden also Spalte i und Spalte $2n + i$ von Z zur Übereinstimmung gebracht.

Die Anzahl der verschiedenen Spalten beträgt dann (unabhängig von der Belegung der restlichen *) $4(n + 1)$ (vergleiche Hilfsbehauptung 1).

Da x Lösung von $Ax = (1, \ldots, 1)^T$ ist, enthält nun jede Zeile von $\overline{A'}$ genau eine 1!

Die 1 aus Zeile i von $\overline{A'}$ befinde sich in Spalte j. Dann stimmt diese Zeile mit der i. Zeile von R_j überein. Wähle dann die Ersetzung der * in L und C so, daß die Zeilen von Z, die durch die i. Zeilen von $\overline{A'}$ bzw. R_j verlaufen, gleich werden (*). Allerdings ist durch die Behandlung der Spalten schon ein Teil der * in L bzw. C ersetzt. Es muß also noch gezeigt werden, daß die schon erfolgten Ersetzungen nicht dazu führen können, daß eine Gleichheit der angegebenen Zeilen nicht mehr möglich ist.

Man sieht dies anhand des folgenden Bildes ein:

$$\begin{array}{|ccccc|c}
\hline
 & \vdots & & \vdots & & \\
\cdots & \epsilon_3 & \cdots & \epsilon_2 & \cdots & z_1 \\
 & \vdots & & \vdots & & \\
\cdots & * & \cdots & \epsilon_1 & \cdots & z_2 \\
 & \vdots & & \vdots & & \\
\hline
\multicolumn{1}{c}{} & s_1 & & s_2 & \multicolumn{1}{c}{} &
\end{array}$$

Angenommen Spalte s_1 und Zeile z_2 führen durch L bzw. C und Spalte s_2 und Zeile z_1 führen durch A'. Spalten s_1 und s_2 sollen durch Ersetzung von * zur Übereinstimmung gebracht werden. Der entsprechende * in A' sei schon durch ϵ_2 ersetzt. Dann hat man eine Situation wie in der obigen Abbildung mit ϵ_1, ϵ_2, $\epsilon_3 \in \{0, 1\}$.

Wenn s_1 und s_2 zu Übereinstimmung gebracht werden sollen, so muß der noch verbleibende * durch $\epsilon = \epsilon_1$ ersetzt werden und es muß gelten: $\epsilon_2 = \epsilon_3$.

Entscheidend ist nun, daß man nur dann versucht z_1 und z_2 zur Übereinstimmung zu bringen, wenn gilt: $\epsilon_2 = \epsilon_1$. Dann ist aber auch $\epsilon = \epsilon_1 = \epsilon_2 = \epsilon_3$ und die Ersetzung des * in L bzw. C hat nicht verhindert, daß z_1 und z_2 zur Übereinstimmung gebracht werden können.

Führt man die Ersetzungen vom Typ (*) nun für alle Zeilen durch, die durch die i. Zeile von $\overline{A'}$ verlaufen ($1 \leq i \leq n$), so ergeben sich (unabhängig von der Belegung der restlichen *) insgesamt $(n^2 + 3n + 1) - n = (n + 1)^2$ verschiedene Zeilen von $\overline{Z}$ (vergleiche Hilfsbehauptung 1).

Damit ist Hilfsbehauptung 3 bewiesen.

- Unabhängig von der Ersetzung der * durch 0 oder 1 hat Z, die Zerlegungsmatrix von f hinsichtlich $\{X, Y\}$, nach Hilfsbehauptung 1 mindestens $4(n+1) = 2^{k+2}$ verschiedene Spalten und $(n+1)^2 = 2^{2k}$ verschiedene Zeilen. Die Anzahl der Zerlegungsfunktionen einer Zerlegung von f hinsichtlich $\{X, Y\}$ muß nach Lemma 4.2 also mindestens $(k+2)+2k = 3k+2$ betragen.

  Gibt es andererseits eine Zerlegung von f hinsichtlich $\{X, Y\}$ mit $(k + 2) + 2k = 3k + 2$ Zerlegungsfunktionen, so muß es eine Ersetzung der * in Z geben, so daß die resultierende Matrix höchstens $4(n + 1) = 2^{k+2}$ verschiedene Spalten und $(n + 1)^2 = 2^{2k}$ verschiedene Zeilen hat (wegen Lemma 4.2 und den angegebenen Mindestzahlen für verschiedene Zeilen und Spalten). Die Zahl der verschiedenen Spalten muß also *genau* $4(n + 1)$ betragen, die Zahl der verschiedenen Zeilen *genau* $(n + 1)^2$.

  Dann gilt nach Hilfsbehauptung 2: $Ax = b$ hat eine binäre Lösung.

- Hat $Ax = b$ eine binäre Lösung, so gibt es nach Hilfsbehauptung 3 eine Ersetzung von * in Z durch Elemente aus $\{0, 1\}$, so daß die resultierende Matrix $\overline{Z}$ nur $4(n+1) = 2^{k+2}$ verschiedene Spalten und nur $(n+1)^2 = 2^{2k}$ verschiedene Zeilen hat. Nach Lemma 4.2 existiert dann eine Zerlegung mit $(k + 2) + 2k = 3k + 2$ Zerlegungsfunktionen.

Insgesamt gilt also:
$Ax = b$ hat genau dann eine binäre Lösung, wenn bei einer Zerlegung der in Polynomzeit berechneten Funktion f hinsichtlich der gleichmächtigen Variablenaufteilung $\{X, Y\}$ die minimale Anzahl von Zelegungsfunktionen kleiner oder gleich $3k + 2$ ist. $\square$

Will man zu einer gegebenen partiellen Funktion f und einer Variablenaufteilung $\{X, Y\}$ eine Belegung der don't cares finden, so daß die Anzahl der Zerlegungsfunktionen, die bei einer Zerlegung hinsichtlich $\{X, Y\}$ nötig sind, minimal wird, so ist folgendes Näherungsverfahren möglich:

1. Bestimme bei der Zerlegungsmatrix Z hinsichtlich $\{X, Y\}$ eine Einteilung der Zeilen in Kompatibilitätsklassen $K_1, \ldots, K_z$ hinsichtlich der Kompatibilitätsrelation $\sim$, wobei z minimal ist. (Dies entspricht einer Lösung des Problems „Partition into Cliques".)

2. Gibt es in einer Kompatibilitätsklasse K_i eine Zeile, die in Spalte j eine 1 (bzw. eine 0) hat, so belege bei allen Zeilen aus K_i die j. Spalte mit 1 (bzw. 0). (Wegen der Kompatibilität muß man nur don't care–Stellen ändern (*).) Gibt es bei den Zeilen aus K_i in Spalte j keine 0 bzw. keine 1, so bleiben die entsprechenden don't care–Stellen (*) erhalten. Man erhält so aus Z eine Matrix Z'.

3. Bestimme nun bei Z' eine Einteilung der Spalten in Klassen $I_1, \ldots, I_s$ kompatibler Spalten, wobei s minimal ist.

4. Belege nun bei allen Spalten aus einer Kompatibilitätsklasse I_i die don't cares so, daß die Spalten gleich werden. Man erhält dadurch aus Z' eine Matrix Z''.

Die resultierende Matrix hat z verschiedene Zeilen und s verschiedene Spalten. Für die Korrektheit dieser Aussage ist es wesentlich, festzustellen, daß die Belegung von don't cares in Schritt 4.) die Kompatibilität der Zeilen in K_i $(1 \leq i \leq z)$ nicht zerstört:

Lemma 4.9 *Sind beim obigen Algorithmus 2 Zeilen z_1 und z_2 von Matrix Z in* einer *Kompatibilitätsklasse K_i, so sind auch Zeilen z_1 und z_2 von Matrix Z'' kompatibel.*

Beweis:
<u>Annahme</u>: Zeilen z_1 und z_2 von Z'' sind nicht kompatibel, obwohl z_1 und z_2 in *einer* Kompatibilitätsklasse K_i.
Dann muß es o.B.d.A. s_1 geben, so daß $Z''_{z_1 s_1} = 0$ und $Z''_{z_2 s_1} = 1$.
Außerdem muß gelten: $Z'_{z_1 s_1} = *$ und $Z'_{z_2 s_1} = *$.
(Wegen Definition von Z' in 2.) und da Zeilen z_1 und z_2 von Z kompatibel sind.)
Sei Spalte s_1 in Kompatibilitätsklasse I_j. Dann muß es in I_j Spalten s_2 und s_3 geben mit $Z'_{z_1 s_2} = 0$ und $Z'_{z_2 s_3} = 1$. ($s_2 \neq s_3$, da sonst Zeilen z_1 und z_2 in Z nicht kompatibel.) Es ergibt sich folgendes Bild von Z':

$$
\begin{array}{c c}
\begin{array}{|ccccccc|}
\hline
 & \vdots & & \vdots & & \vdots & \\
\cdots & * & \cdots & 0 & \cdots & \delta & \cdots \\
 & \vdots & & \vdots & & \vdots & \\
\cdots & * & \cdots & \epsilon & \cdots & 1 & \cdots \\
 & \vdots & & \vdots & & \vdots & \\
\hline
\end{array} & \begin{array}{c} \\ z_1 \\ \\ z_2 \\ \\ \end{array} \\
\begin{array}{ccccccc}
 & s_1 & & s_2 & & s_3 &
\end{array} &
\end{array}
$$

- Wäre $\epsilon = 0$, so wären Spalten s_2 und s_3 nicht kompatibel, also könnten s_2 und s_3 nicht gleichzeitig in I_j enthalten sein.
- Wäre $\epsilon = 1$, so wären Zeilen z_1 und z_2 von Z nicht kompatibel.
- $\epsilon = *$ ist nicht möglich, denn wegen der 0 in z_1 wäre $*$ in Schritt 2.) des Algorithmus durch 0 ersetzt worden.

Die dargestellte Situation kann also nicht auftreten. Es ergibt sich ein Widerspruch zur Annahme.

□

Das folgende kleine Beispiel zeigt, daß die Durchführung von Schritt 2.) des Algorithmus entscheidend dafür ist, daß man die Zusicherung von Lemma 4.9 machen kann:

Beispiel 4:
Gegeben ist folgende Zerlegungsmatrix einer Funktion $f : \{0,1\}^3 \leadsto \{0,1\}$:

$$\begin{array}{r|cccc|}
x_2 & 0 & 0 & 1 & 1 \\
x_3 & 0 & 1 & 0 & 1 \\
x_1 & & & & \\
\hline
0 & 1 & * & 0 & 0 \\
1 & * & 0 & 0 & 0 \\
\hline
\end{array}$$

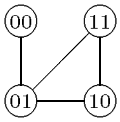
Der Algorithmus würde zunächst feststellen, daß die beiden Zeilen der Zerlegungsmatrix kompatibel sind. Wird dann Schritt 2.) ausgelassen, so kann man eine Partition der Spalten in 2 Mengen paarweise kompatibler Spalten finden. Angenommen Schritt 3.) faßt z.B. die ersten beiden und die letzten beiden Spalten zusammen. Es ergibt sich dann nach Schritt 4.) folgende Matrix Z'', bei der die beiden Zeilen *nicht* kompatibel sind:

$$\begin{array}{r|cccc|}
x_2 & 0 & 0 & 1 & 1 \\
x_3 & 0 & 1 & 0 & 1 \\
x_1 & & & & \\
\hline
0 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 \\
\hline
\end{array}$$

Ebenso sieht man an diesem Beispiel, daß man mit dem Algorithmus, der das Verfahren für einseitige Zerlegungen nacheinander auf die Zeilen und die Spalten anwendet, nur *Näherungslösungen* erhält:

$$\begin{array}{r|cc|}
x_1 & 0 & 1 \\
x_2 x_3 & & \\
\hline
0\ 0 & 1 & * \\
0\ 1 & * & 0 \\
1\ 0 & 0 & 0 \\
1\ 1 & 0 & 0 \\
\hline
\end{array}$$

Die Kompatibilitätsrelation $\sim$ auf den Zeilenindizes aus $\{0,1\}^2$ hat folgendes Aussehen:



Es gibt also verschiedene Möglichkeiten, die Zeilen in 2 Kompatibilitätsklassen zu partitionieren:

- Wählt man $\{\{00, 01\}, \{10, 11\}\}$, dann erhält man folgende Matrix Z'

$x_2 x_3$ \ x_1	0	1
0 0	1	0
0 1	1	0
1 0	0	0
1 1	0	0

  und man erhält 2 verschiedene Spalten.

- Wählt man jedoch $\{\{00\}, \{01, 10, 11\}\}$, dann erhält man folgende Matrix Z'

$x_2 x_3$ \ x_1	0	1
0 0	1	$*$
0 1	0	0
1 0	0	0
1 1	0	0

  und die beiden Spalten sind kompatibel.

4.4 Funktionen mit mehreren Ausgängen

Wie in Kapitel 2 schon angedeutet, beruht die Effizienz guter Realisierungen boolescher Funktionen häufig gerade auf der Tatsache, daß gleiche Teilschaltungen „mehrfach verwendet“ werden. Bei Funktionen mit mehreren Ausgängen werden die einzelnen Ausgangsfunktionen nicht getrennt realisiert, sondern Ergebnisse, die von Teilschaltungen der Realisierung *einer* Ausgangsfunktion berechnet werden, werden bei der Realisierung *anderer* Ausgangsfunktionen mit Vorteil verwendet.

In diesem Abschnitt soll eine Möglichkeit entwickelt werden, auch im Zusammenhang mit Zerlegungen boolescher Funktionen *gleiche* Teilschaltungen bei der Realisierung *mehrerer* Ausgangsfunktionen zu benutzen. Dazu werden Zerlegungsfunktionen berechnet, die gleichzeitig bei der Zerlegung mehrerer Ausgangsfunktionen verwendet werden können (vgl. Abbildung 4.6). Es werden also im Gegensatz zu anderen Lösungen (z.B. [HOI89]) nicht Teilfunktionen miteinander verglichen in der Hoffnung, daß einige dieser Teilfunktionen zufällig gleich sind, sondern es wird darauf hingearbeitet, gleiche Teilfunktionen zu erhalten.

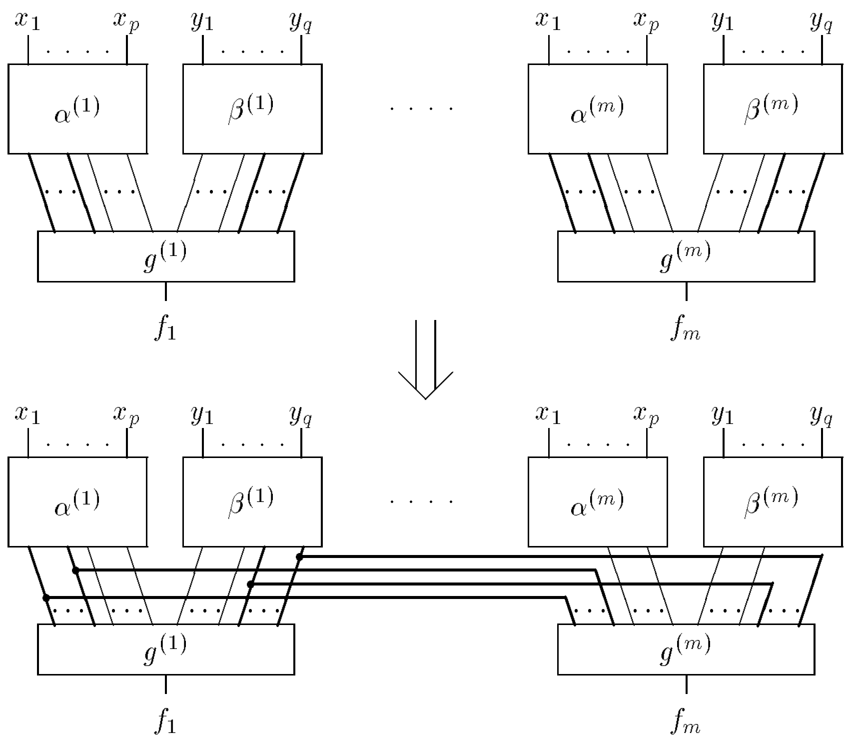


Abbildung 4.6: Zerlegung von Funktionen mit mehreren Ausgängen

Dieses Vorgehen ist nicht nur bei der Bearbeitung von booleschen Funktionen mit *mehreren* Ausgängen von Bedeutung. Auch wenn eine Funktion ursprünglich nur einen Ausgang hat, so treten bei der rekursiven Anwendung des Zerlegungsverfahrens auf ihre Zerlegungsfunktionen im allgemeinen Funktionen mit mehreren Ausgängen auf.

Das vorgeschlagene Verfahren, bei der Zerlegung der Ausgangsfunktionen f_1, $\ldots$, f_m einer Funktion $f \in B_{n,m}$ nach *gemeinsamen* Zerlegungsfunktionen zu suchen, ist nur dann anwendbar, wenn die Zerlegung der Ausgangsfunktionen *hinsichtlich der gleichen Variablenaufteilung* erfolgt. Es ist jedoch nicht vorteilhaft, diese Forderung in jedem Fall aufrechtzuerhalten. Es kann vorkommen, daß die Anzahl der Zerlegungsfunktionen, die bei einer bestimmten Variablenaufteilung notwendig sind, für eine der Ausgangsfunktionen minimal ist, für andere Ausgangsfunktionen aber sehr weit vom Minimum entfernt liegt. Daher werden die Funktionen $f_1, \ldots, f_m$ durch eine Heuristik in *Gruppen* eingeteilt, wobei alle Funktionen aus einer dieser Gruppen hinsichtlich der gleichen Variablenaufteilung zerlegt werden. Die Gruppen werden gerade so gewählt, daß die Anzahl der Zerlegungsfunktionen, die bei Zerlegung hinsichtlich dieser Variablenaufteilung notwendig sind, für keine Funktion aus der Gruppe allzu stark vom Minimum abweicht.

4.4.1 Eine Heuristik zur Wahl einer geeigneten Variablenaufteilung

Die Heuristik, die in diesem Abschnitt angegeben wird, bestimmt zu gegebenen Funktionen $f_1, \ldots, f_m \in B_n$ eine Einteilung der Funktionen in verschiedene Gruppen und zu jeder dieser Gruppen eine Variablenaufteilung in zwei disjunkte Variablenmengen X und Y, wobei $|X|$ vorgegeben ist (z.B. $|X| = \lfloor n/2 \rfloor$). In dem angegebenen Verfahren sind noch Parameter offen, durch deren Wahl die Wirkungsweise der Heuristik beeinflußt werden kann.

Eingabe:
- Eine Menge $F = \{f_1, \ldots, f_m\}$ von Funktionen aus B_n.
- Eine natürliche Zahl p mit $2 \leq p \leq n-2$. Es sollen Variablenaufteilungen[4] $\{X, Y\}$ der n Eingangsvariablen berechnet werden mit $|X| = p$.

Ausgabe: Eine Einteilung der Funktionen in F in Gruppen $G_1, \ldots, G_g$ mit $\cup_{1 \leq i \leq g} G_i = F$ und $G_i \cap G_j = \emptyset$ für $i \neq j$. Zu jeder Gruppe G_i gehört eine Variablenaufteilung A_i. (Die Funktionen aus G_i sollen hinsichtlich A_i zerlegt werden.)

Algorithmus:

1. $g = 0$
2. Für alle Variablenaufteilungen $A = \{X, Y\}$ mit $|X| = p$:
   Für alle $f_i \in F$:
   Berechne $zf_i(A)$, die minimale Anzahl von Zerlegungsfunktionen bei einer Zerlegung von f_i hinsichtlich Variablenaufteilung A
3. Für jede Funktion $f_i \in F$
   Bestimme die Variablenaufteilung A, für die $zf_i(A)$ minimal ist. Setze für dieses A $zfmin_i = zf_i(A)$.
4. Falls es Funktionen f_i in F gibt, so daß $zfmin_i = n$:
   Dann sind diese Funktionen hinsichtlich der betrachteten Variablenaufteilungen nicht nichttrivial zerlegbar.
   $g = g + 1$
   Für jede Funktion $f_i \in F$ mit $zfmin_i = n$:
   $G_1 = G_1 \cup \{f_i\}$
   $F = F \setminus \{f_i\}$

   Für die Funktionen in G_1 wird dann eine Shannon–Zerlegung durchgeführt.
5. Für alle Variablenaufteilungen $A = \{X, Y\}$ mit $|X| = p$:
   Für alle $f_i \in F$:

[4] Hier werden zweiseitige Zerlegungen betrachtet. Analog läßt sich auch ein Algorithmus für einseitige Zerlegungen angeben.

Berechne $abweich_i(A) = zf_i(A) - zfmin_i$, die Abweichung der Anzahl von Zerlegungsfunktionen, die bei Zerlegung von f_i hinsichtlich der Variablenaufteilung A benötigt werden, von der minimalen Anzahl der Zerlegungsfunktionen.

6. Bestimme die Variablenaufteilung A^*, für die

$$\left\| \begin{pmatrix} abweich_{i_1}(A^*) \\ \vdots \\ abweich_{i_k}(A^*) \end{pmatrix} \right\|_l ,$$

minimal ist, wobei $F = \{f_{i_1}, \ldots, f_{i_k}\}$.

7. $g = g + 1$
   $A_g = A^*$
   Für alle $f_i \in F$, für die gilt
   $zf_i(A^*) - zfmin_i < parameter \cdot (n - zfmin_i)$:
   $G_g = G_g \cup \{f_i\}$
   $F = F \setminus \{f_i\}$

   ($parameter$ ist frei wählbar mit $0 < parameter \leq 1$.)
   (Gilt dies für keine Funktion $f_i \in F$:
   Wähle $f_j \in F$, so daß $abweich_j = zf_j(A^*) - zfmin_j$ minimal.
   $G_g = G_g \cup \{f_j\}$
   $F = F \setminus \{f_j\}$)

8. Falls $F \neq \emptyset$, wiederhole die Schritte ab 6., um für die noch verbleibenden Funktionen Gruppen mit gemeinsamer Variablenaufteilung zu finden.

Es folgen einige Erklärungen zum Algorithmus:

zu 4.: Nicht alle Funktionen müssen nichttrivial zerlegbar sein mit der gewünschten Aufteilung der Variablen. Für die Funktionen, die nicht auf diese Weise nichttrivial zerlegbar sind, wird eine „ungleichmäßigere" Variablenaufteilung gewählt (hier die Shannon–Zerlegung). Die Shannon–Zerlegung existiert zu jeder Funktion. Wie in Abschnitt 4.1.2 gezeigt wurde, ist es möglich, daß auf der nächsten Rekursionsstufe die Zerlegungsfunktionen wieder eine nichttriviale Zerlegung hinsichtlich einer gewünschten Variablenaufteilung (z.B. einer gleichmächtigen Variablenaufteilung) besitzen. Man kann bei der Shannon–Zerlegung die Wahl der Zerlegungsvariablen x_i beispielsweise von Zerlegungseigenschaften der Kofaktoren $f|_{x_i}$ und $f|_{\overline{x_i}}$ (d.h. der Zerlegungsfunktionen) abhängig machen.

zu 6.: Betrachtet werden Variablenaufteilungen A und dazu Vektoren, die für die noch zu behandelnden Funktionen gebildet sind aus den Abweichungen der Zerlegungsfunktionsanzahl bei Variablenaufteilung A von der minimalen Anzahl der Zerlegungsfunktionen ($abweich_{i_j}(A) = zf_{i_j}(A) - zfmin_{i_j}$). Ausgewählt wird die Variablenaufteilung A^*, für die die l–Norm dieses

Vektors der Abweichungen minimal ist. Ein entscheidender Faktor ist die Wahl von l: Wählt man als Norm die 1–Norm (Summennorm), so wird die Variablenaufteilung A^* ausgewählt, für die die Summe der Abweichungen minimal ist. (Entscheidend ist die *Summe* der Abweichungen. Es können starke Abweichungen bei einzelnen Funktionen in Kauf genommen werden, wenn nur die Summe der Abweichungen klein ist.) Wählt man das andere Extrem, die ∞–Norm (Maximumnorm), so wird die Variablenaufteilung ausgewählt, für die die größte aller Abweichungen minimal ist. (Entscheidend ist das Maximum aller Abweichungen. Es kann in Kauf genommen werden, daß die Summe der Abweichungen relativ groß ist, wenn nur die Abweichung bei keiner der Funktionen allzu groß ist.)

zu 7.: Nicht alle noch zu behandelnden Funktionen werden hinsichtlich der gefundenen Variablenaufteilung A^* zerlegt, da für einzelne Funktionen f_i die Abweichung $zf_i(A^*) - zfmin_i$ zu groß sein kann. Ob die Abweichung zu groß ist, wird durch Auswertung des Kriteriums

$$zf_i(A^*) - zfmin_i < parameter \cdot (n - zfmin_i)$$

entschieden. Wie diese Entscheidung ausfällt, hängt von der Wahl von *parameter* ab. Ist $parameter = 1$, so wird für alle Funktionen eine Zerlegung hinsichtlich A^* durchgeführt, für die diese Zerlegung nichttrivial ist ($zf_i(A^*) < n$). Ist *parameter* genügend klein (fast 0), so kann nur noch für solche Funktionen eine Zerlegung hinsichtlich A^* durchgeführt werden, für die A^* eine Variablenaufteilung mit minimaler Anzahl von Zerlegungsfunktionen ist ($zf_i(A^*) = zfmin_i$).

zu 8.: Mit den noch verbliebenen Funktionen wird das gleiche Verfahren durchgeführt und eine weitere Gruppe von Funktionen mit einer dazugehörigen Variablenaufteilung abgespalten. Dies geschieht so lange, bis alle Funktionen in Gruppen eingeteilt sind. Das Ergebnis ist beeinflußt durch die Wechselwirkung der gewählten Parameter l und *parameter*.

Ähnlich wie bei Funktionen mit einem Ausgang kann man auch hier die Laufzeit durch den Einsatz von Verfahren der iterativen Verbesserung verkürzen. Auch bei der angegebenen Heuristik muß man nicht unbedingt *alle* möglichen Variablenaufteilungen $A = \{X, Y\}$ mit $|X| = p$ betrachten. Man kann beginnen mit einer beliebigen Variablenaufteilung $\{X, Y\}$ und bildet neue Variablenaufteilungen, indem man schrittweise Variablenpaare aus X und Y austauscht. Ergibt sich bei der Anzahl der Zerlegungsfunktionen nach Vertauschen zweier Variablen (oder auch nach mehrmaligem Vertauschen zweier Variablen) für keine der Funktionen $f_1, \ldots, f_m$ mehr eine Verbesserung, so beendet man das Erzeugen neuer Variablenaufteilungen. Die Tabelle aller $zf_i(A)$, mit der im folgenden gearbeitet wird, enthält dann im allgemeinen nur noch die Anzahlen der Zerlegungsfunktionen für einen *Teil* der möglichen Variablenaufteilungen.

Die Wirkung der angegebenen Heuristik soll anhand eines (kleineren) Beispiels demonstriert werden:

Beispiel 5:

Es sei eine Funktion $f = (f_0, f_1) \in B_{4,2}$ gegeben. Es gelte $\forall(x_0, x_1, x_2, x_3) \in \{0,1\}^4$:

$$\begin{aligned} f_0(x_0, x_1, x_2, x_3) &= x_0 \oplus x_1 \oplus x_2 \oplus x_3 \\ f_1(x_0, x_1, x_2, x_3) &= (x_0 \oplus x_2) \cdot (x_1 \oplus x_3) \end{aligned}$$

Es sollen gleichmächtige Variablenaufteilungen verwendet werden.

Für die Anzahlen der Zerlegungsfunktionen gilt:

- Für f_0: $zf_0(A) = 2$ für alle gleichmächtigen Variablenaufteilungen A. (f_0 ist totalsymmetrisch!)
- Für f_1: $zf_1(A) = 2$ für $A = \{\{x_0, x_2\}, \{x_1, x_3\}\}$. Für alle anderen Variablenaufteilungen A' ist $zf_1(A')$ größer.

In Schritt 6. des Algorithmus wird also die Variablenaufteilung $A^* = \{\{x_0, x_2\}, \{x_1, x_3\}\}$ bestimmt. In Schritt 7. werden f_0 und f_1 in die gleiche Gruppe eingeordnet, da bei beiden Funktionen die Abweichung der Zerlegungsfunktionszahl bei Zerlegung hinsichtlich A^* vom Minimum der Zerlegungsfunktionsanzahlen gleich 0 ist.

Die Zerlegungsmatrizen von f_0 und f_1 hinsichtlich A^* sehen folgendermaßen aus:

f_0:

$$\begin{array}{cc|cccc} & x_1 & 0 & 0 & 1 & 1 \\ & x_3 & 0 & 1 & 0 & 1 \\ x_0 & x_2 & & & & \\ \hline 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 \end{array}$$

f_1:

$$\begin{array}{cc|cccc} & x_1 & 0 & 0 & 1 & 1 \\ & x_3 & 0 & 1 & 0 & 1 \\ x_0 & x_2 & & & & \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \end{array}$$

Man erkennt anhand der Zerlegungsmatrizen, daß als Zerlegungsfunktionen auf den Variablen x_0 und x_2 sowohl bei f_0 als auch bei f_1 lediglich

$$\alpha_1(x_0, x_2) = x_0 \oplus x_2 \text{ bzw. } \alpha_1'(x_0, x_2) = \overline{x_0 \oplus x_2}$$

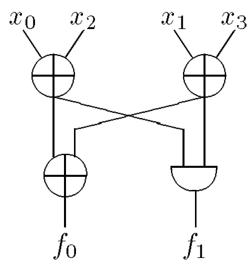
in Frage kommen. Ebenso kommen als Zerlegungsfunktionen auf den Variablen x_1 und x_3 lediglich

$$\beta_1(x_1, x_3) = x_1 \oplus x_3 \text{ bzw. } \beta_1'(x_1, x_3) = \overline{x_1 \oplus x_3}$$

in Frage.

Wählt man die Zerlegungsfunktionen so aus, daß für f_0 und f_1 Zerlegungsfunktionen gemeinsam verwendet werden können (z.B. bei Wahl von α_1 und β_1 sowohl für f_0 als auch für f_1), so ergibt sich folgende Realisierung unter Ausnutzung der Zerlegung[5]:

[5] Näheres zur Wahl von gemeinsamen Zerlegungsfunktionen für den allgemeinen Fall findet man im nächsten Abschnitt.



4.4.2 Bestimmung gemeinsamer Zerlegungsfunktionen

In diesem Abschnitt soll untersucht werden, wie man Zerlegungsfunktionen berechnen kann, die bei der Zerlegung mehrerer Ausgangsfunktionen gemeinsam verwendet werden können. Zunächst wird ein Kriterium angegeben, das sich aus einem entsprechenden Kriterium für Funktionen mit einem Ausgang (siehe Lemma 4.6) ergibt:

Lemma 4.10 *Seien $f_1, \dots, f_m \in B_n$. Seien die Eingangsvariablen von $f_1, \dots, f_m$ $x_1, \dots, x_p$, $y_1, \dots, y_q$ und sei für $1 \le i \le m$ $r_i = \lceil \log(vz(X, f_i)) \rceil$. Dann können $\alpha_1, \dots, \alpha_h$ genau dann alle als Zerlegungsfunktionen in einseitigen Zerlegungen von $f_1, \dots, f_m$ hinsichtlich X verwendet werden, d.h. es gibt genau dann einseitige Zerlegungen von $f_1, \dots, f_m$ hinsichtlich X der Form*

$$\begin{aligned} f_1(\mathbf{x}, \mathbf{y}) &= g^{(1)}(\alpha_1(\mathbf{x}), \dots, \alpha_h(\mathbf{x}), \alpha_{h+1}^{(1)}(\mathbf{x}), \dots, \alpha_{r_1}^{(1)}(\mathbf{x}), \mathbf{y}) \\ &\vdots \\ f_m(\mathbf{x}, \mathbf{y}) &= g^{(m)}(\alpha_1(\mathbf{x}), \dots, \alpha_h(\mathbf{x}), \alpha_{h+1}^{(m)}(\mathbf{x}), \dots, \alpha_{r_m}^{(m)}(\mathbf{x}), \mathbf{y}), \end{aligned}$$

wenn für alle $1 \le i \le m$ gilt:

$$vz(X, f_i, \alpha) \le 2^{r_i - h}.$$

($\alpha = (\alpha_1, \dots, \alpha_h)$.)

Bei der Suche nach Zerlegungsfunktionen tritt häufig die Situation auf, daß für die Zerlegung von Funktionen $f_1, \dots, f_m$ ein Teil der Zerlegungsfunktionen schon bestimmt ist und man möglichst viele der restlichen Zerlegungsfunktionen für $f_1, \dots, f_m$ gemeinsam wählen will. In diesem Zusammenhang ist die Aussage von Lemma 4.11 von Bedeutung, das eine einfache Folgerung aus Lemma 4.10 darstellt. In Lemma 4.11 wird ähnlich wie in Abschnitt 4.1.4 von der Äquivalenzklasseneinteilung Gebrauch gemacht, die durch die Gleichheit auf den Zeilen einer Zerlegungsmatrix induziert wird. Vorher wird noch eine Bezeichnung vereinbart:

Bezeichnung 15 *Sei eine Funktion $f_i \in B_n$ mit den Eingangsvariablen $x_1, \dots,$ x_p, $y_1, \dots, y_q$ gegeben. Sei Z_i die Zerlegungsmatrix von f_i hinsichtlich $X = \{x_1,$*

…, $x_p\}$. Sei weiterhin die durch die Gleichheit von Zerlegungsmatrixzeilen auf $\{0,1\}^p$ induzierte Äquivalenzklasseneinteilung $\{K_1^{(i)}, \ldots, K_{vz(X,f_i)}^{(i)}\}$. Seien

$$\alpha_1^{(i)}, \ldots, \alpha_{k_i}^{(i)}, \alpha_1, \ldots, \alpha_h \in B_p.$$

Für $a \in \{0,1\}^{k_i}$ aus dem Bild von $\alpha^{(i)}$ und $a' \in \{0,1\}^h$ aus dem Bild von α sei $X^{aa'} \subseteq \{0,1\}^p$ die Menge der Urbilder von (a,a') bzgl. $(\alpha^{(i)}, \alpha)$.

Dann wird mit $S_{aa'}^{(i)}$ folgende Menge bezeichnet:

$$S_{aa'}^{(i)} = \{1 \leq j \leq vz(X, f_i) \mid K_j^{(i)} \cap X^{aa'} \neq \emptyset\}$$

Betrachtet man alle binären Zeilenindizes $x \in \{0,1\}^p$ von Z_i, für die$(\alpha^{(i)}, \alpha)(x)$ $= (aa')$ ist und dazu alle Äquivalenzklassen (hinsichtlich Zeilengleichheit) $K_{j_1}^{(i)}$, $\ldots, K_{j_l}^{(i)}$, in die diese binären Zeilenindizes fallen, so umfaßt $S_{aa'}^{(i)}$ gerade die Indizes $j_1, \ldots, j_l$. ($|S_{aa'}^{(i)}|$ gibt also die Anzahl der *verschiedenen* Zeilen von Z_i an, auf deren Indizes $(\alpha^{(i)}, \alpha)$ den Wert (aa') liefert.)

Mit der angegebenen Bezeichnung ergibt sich dann Lemma 4.11:

Lemma 4.11 *Seien $f_1, \ldots, f_m$ Funktionen aus B_n mit den Eingangsvariablen $x_1, \ldots, x_p$, $y_1, \ldots, y_q$. Für $1 \leq i \leq m$ sei Z_i die Zerlegungsmatrix von f_i hinsichtlich $X = \{x_1, \ldots, x_p\}$ und $r_i = \lceil \log(vz(X, f_i)) \rceil$.*
Für alle $1 \leq i \leq m$ seien Zerlegungsfunktionen

$$\alpha_1^{(i)}, \ldots, \alpha_{k_i}^{(i)} \in B_p$$

vorgegeben. Dann sind unter diesen Voraussetzungen

$$\alpha_1, \ldots, \alpha_h \in B_p$$

genau dann gemeinsame Zerlegungsfunktionen von $f_1, \ldots, f_m$, d.h. es gibt genau dann einseitige Zerlegungen von $f_1, \ldots, f_m$ hinsichtlich X der Form

$$\begin{aligned}
&f_1(\mathbf{x}, \mathbf{y}) = \\
&\quad g^{(1)}(\alpha_1^{(1)}(\mathbf{x}), \ldots, \alpha_{k_1}^{(1)}(\mathbf{x}), \alpha_1(\mathbf{x}), \ldots, \alpha_h(\mathbf{x}), \alpha_{k_1+h+1}^{(1)}(\mathbf{x}), \ldots, \alpha_{r_1}^{(1)}(\mathbf{x}), \mathbf{y}) \\
&\ \vdots \\
&f_m(\mathbf{x}, \mathbf{y}) = \\
&\quad g^{(m)}(\alpha_1^{(m)}(\mathbf{x}), \ldots, \alpha_{k_1}^{(m)}(\mathbf{x}), \alpha_1(\mathbf{x}), \ldots, \alpha_h(\mathbf{x}), \alpha_{k_1+h+1}^{(m)}(\mathbf{x}), \ldots, \alpha_{r_m}^{(m)}(\mathbf{x}), \mathbf{y}),
\end{aligned}$$

wenn für alle $1 \leq i \leq m$ gilt:
Ist $a \in \{0,1\}^{k_i}$ aus dem Bild von $\alpha^{(i)} = (\alpha_1^{(i)}, \ldots, \alpha_{k_i}^{(i)})$ und $a' \in \{0,1\}^h$ aus dem Bild von $\alpha = (\alpha_1, \ldots, \alpha_h)$, so gilt:

$$|S_{aa'}^{(i)}| \leq 2^{r_i - k_i - h}.$$

Lemma 4.11 liefert also ein notwendiges und hinreichendes Kriterium für die Existenz gemeinsamer Zerlegungsfunktionen bei der Zerlegung von Funktionen $f_1, \ldots, f_m$, bei denen evtl. schon gewisse Zerlegungsfunktionen fest vorgegeben sind.

Dieses Kriterium wird im Rahmen eines branch–and–bound–Algorithmus ausgenutzt, der gemeinsame Zerlegungsfunktionen $\alpha_1, \ldots, \alpha_h$ berechnet. Der branch–and–bound–Algorithmus baut die Funktionstabelle von $\alpha = (\alpha_1, \ldots, \alpha_h)$ schrittweise auf und testet für die schon aufgestellte Teiltabelle, ob das Kriterium aus Lemma 4.11 evtl. schon verletzt wird. Ist dies nicht der Fall, so wird der Funktionswert der nächsten Zeile der Funktionstabelle festgelegt, ansonsten wird die Belegung für *diese* Zeile (und, falls für diese Zeile schon alle möglichen Funktionswerte getestet wurden, auch für *vorangehende* Zeilen) zurückgenommen. Im folgenden wird das Verfahren genauer beschrieben:

Eingabe:
- Funktionen $f_1, \ldots, f_m$ aus B_n mit den Eingangsvariablen $x_1, \ldots, x_p, y_1, \ldots, y_q$.
- Für $1 \leq i \leq m$: Zerlegungsmatrizen Z_i von f_i hinsichtlich $X = \{x_1, \ldots, x_p\}$ und $r_i = \lceil \log(vz(X, f_i)) \rceil$.
- Für $1 \leq i \leq m$: Sei $\{K_1^{(i)}, \ldots, K_{vz(X,f_i)}^{(i)}\}$ die durch Zeilengleichheit in Z_i auf $\{0,1\}^p$ induzierte Äquivalenzklasseneinteilung. Sei znr_i eine Funktion, die jedem x aus einer Äquivalenzklasse $K_j^{(i)}$ den Index j zuordnet.
- Für $1 \leq i \leq m$: Zerlegungsfunktionen $\alpha_1^{(i)}, \ldots, \alpha_{k_i}^{(i)} \in B_p$.
- Natürliche Zahl h mit $h \leq r_i - k_i$ für alle $1 \leq i \leq m$. (h gibt die Anzahl gemeinsamer Zerlegungsfunktionen an, nach denen gesucht wird.)

Ausgabe:
- $\alpha_1, \ldots, \alpha_h \in B_p$, so daß es einseitige Zerlegungen von $f_1, \ldots, f_m$ hinsichtlich X gibt der Form

$$\begin{aligned}
&f_1(x_1, \ldots, x_p, y_1, \ldots, y_q) = \\
&\qquad g^{(1)}(\alpha_1^{(1)}(\mathbf{x}), \ldots, \alpha_{k_1}^{(1)}(\mathbf{x}), \alpha_1(\mathbf{x}), \ldots, \alpha_h(\mathbf{x}), \\
&\qquad\qquad\qquad \alpha_{k_1+h+1}^{(1)}(\mathbf{x}), \ldots, \alpha_{r_1}^{(1)}(\mathbf{x}), y_1, \ldots, y_q) \\
&\vdots \\
&f_m(x_1, \ldots, x_p, y_1, \ldots, y_q) = \\
&\qquad g^{(m)}(\alpha_1^{(m)}(\mathbf{x}), \ldots, \alpha_{k_m}^{(m)}(\mathbf{x}), \alpha_1(\mathbf{x}), \ldots, \alpha_h(\mathbf{x}), \\
&\qquad\qquad\qquad \alpha_{k_m+h+1}^{(m)}(\mathbf{x}), \ldots, \alpha_{r_m}^{(m)}(\mathbf{x}), y_1, \ldots, y_q),
\end{aligned}$$

  falls es überhaupt h Funktionen aus B_p mit dieser Eigenschaft gibt.
- Sonst: Eine Meldung daß es keine h Funktionen aus B_p mit der angegebenen Eigenschaft gibt.

Algorithmus:

1. Seien für alle $1 \leq i \leq m$, $a \in \{0,1\}^{k_i}$, $a' \in \{0,1\}^h$ die Mengen $S^{(i)}_{aa'}$ leer.
   $\alpha(\epsilon)$ sei undefiniert $\forall \epsilon \in \{0,1\}^p$.
2. Setze $\alpha(0,\ldots,0) = (0,\ldots,0)$.
   $\forall 1 \leq i \leq m$: „$S^{(i)}_{\alpha^{(i)}(0,\ldots,0)(0,\ldots,0)} = S^{(i)}_{\alpha^{(i)}(0,\ldots,0)(0,\ldots,0)} \cup \{znr_i(0,\ldots,0)\}$“
3. $x = 1$
   $fktwert = (0,\ldots,0)$
4. $\alpha(bin_p(x)) = fktwert$
   if $\forall\ 1 \leq i \leq m$
   $|S^{(i)}_{\alpha^{(i)}(bin_p(x))fktwert} \cup \{znr_i(bin_p(x))\}| \leq 2^{r_i - k_i - h}$ **then**
   $\forall 1 \leq i \leq m$: „$S^{(i)}_{\alpha^{(i)}(bin_p(x))fktwert} = S^{(i)}_{\alpha^{(i)}(bin_p(x))fktwert} \cup \{znr_i(bin_p(x))\}$“
   $x = x + 1$
   $fktwert = (0,\ldots,0)$

   else
   while $\alpha(bin_p(x)) = (1,\ldots,1)$ **do**
   $\alpha(bin_p(x)) =$ undefiniert
   $x = x - 1$
   $\forall 1 \leq i \leq m$: „$S^{(i)}_{\alpha^{(i)}(bin_p(x))\alpha(bin_p(x))} = S^{(i)}_{\alpha^{(i)}(bin_p(x))\alpha(bin_p(x))} \setminus \{znr_i(bin_p(x))\}$“

   od
   $fktwert = bin_h(int(\alpha(bin_p(x))) + 1)$
   $\alpha(bin_p(x)) =$ undefiniert

   fi
5. Wenn $x = 2^p$, d.h. wenn die Funktionstabelle von α für alle $x \in \{0,1\}^p$ aufgestellt worden ist, dann:
   Gebe α als Ergebnis aus.
   Wenn $x = 0$, d.h. wenn man am Ausgangspunkt angekommen ist, ohne eine geeignete Funktion α zu finden, dann:
   Gebe aus, daß es kein solches α gibt.
   Sonst:
   Weiter mit Schritt 4.

Anmerkung:
Die Operationen „$\cup$ “ und „$\setminus$“ in den Schritten 2. und 4. wurden der Einfachheit halber bewußt etwas ungenau angegeben. Wird ein Eintrag an der Stelle x in der Funktionstabelle von α wieder gelöscht, so müssen für alle $1 \leq i \leq m$ die Mengen $S^{(i)}_{\alpha^{(i)}(x)\alpha(x)}$ korrigiert werden. Allerdings darf der Äquivalenzklassenindex $znr_i(x)$ nur dann aus $S^{(i)}_{\alpha^{(i)}(x)\alpha(x)}$ entfernt werden, wenn es keine vorangehende

Stelle x' in der Funktionstabelle gibt mit $\alpha^{(i)}(x') = \alpha^{(i)}(x)$, $\alpha(x') = \alpha(x)$ und $znr_i(x) = znr_i(x')$, so daß also $znr_i(x)$ wegen dieser Stelle in $S^{(i)}_{\alpha^{(i)}(x)\alpha(x)}$ bleiben muß. Der Index $znr_i(x)$ darf erst dann endgültig aus der Menge $S^{(i)}_{\alpha^{(i)}(x)\alpha(x)}$ entfernt werden, wenn er genauso oft „entfernt" wurde, wie er „hinzuvereinigt" wurde. Für die Implementierung bietet es sich an, die Mengen $S^{(i)}_{aa'}$ nicht als Bitvektoren, sondern als Vektoren ganzer Zahlen zu repräsentieren. Jedes Mal, wenn ein Index $znr_i(x)$ zu einer Menge hinzugefügt wird, wird die entsprechende Stelle des Vektors um 1 erhöht, beim Wegnehmen um 1 vermindert. Ein Index gilt erst dann nicht in $S^{(i)}_{aa'}$ vorhanden, wenn an der entsprechenden Stelle des Vektors 0 steht.

Einige Punkte des Algorithmus sollen noch kurz erläutert werden:

zu 2.: Der Funktionswert für α an der Stelle $(0, \ldots, 0)$ wird auf $(0, \ldots, 0)$ festgelegt. Falls es eine Funktion α gibt mit den gewünschten Eigenschaften, so gibt es auch eine mit $\alpha(0, \ldots, 0) = (0, \ldots, 0)$. (Denn gibt es allgemein eine Zerlegung mit den Zerlegungsfunktionen $\alpha'_1, \ldots, \alpha'_i, \ldots, \alpha'_r$, so gibt es auch eine mit den Zerlegungsfunktionen $\alpha'_1, \ldots, \overline{\alpha'_i}, \ldots, \alpha'_r$.)

zu 4.: Die Funktionstabelle zu α wird schrittweise aufgebaut. Wird die Bedingung von Lemma 4.11 schon von dem aktuellen Anfangsstück der Funktionstabelle verletzt, so werden alle Fortsetzungen dieser Tabelle die Bedingung verletzen. Also muß „in einen anderen Ast des branch–and–bound–Verfahrens verzweigt werden". Wird die Bedingung durch das Anfangsstück der Funktionstabelle *nicht* verletzt, so versucht man, dieses Anfangsstück fortzusetzen zu einer vollständigen Funktionstabelle von α.
Zu Beginn von Schritt 4. ist ein Anfangsstück der Funktionstabelle von α schon aufgebaut: Die Funktionswerte $\alpha(0, \ldots, 0)$ bis $\alpha(bin_p(x-1))$ sind festgelegt und das aktuelle Anfangsstück der Funktionstabelle verletzt die Bedingung aus Lemma 4.11 noch nicht. Falls $int(fktwert) > 0$, so wurde vorher bereits erfolgslos versucht, $\alpha(bin_p(x))$ mit den Funktionswerten zu belegen, die kleiner als $int(fktwert)$ sind.
Zunächst wird getestet, ob bei einer Belegung $\alpha(bin_p(x)) = fktwert$ die Bedingung aus Lemma 4.11 schon verletzt wird. Ist dies nicht der Fall, so wird bleibt diese Belegung bestehen, für den nächsten Durchlauf von Schritt 4. wird x um 1 erhöht und $fktwert$ auf $(0, \ldots, 0)$ zurückgesetzt.
Wird die Bedingung verletzt, so wird $fktwert$ $(= \alpha(bin_p(x)))$ um 1 erhöht, der Funktionswert von $\alpha(bin_p(x))$ wird wieder auf „undefiniert" gesetzt und im nächsten Durchlauf von Schritt 4. wird eine Belegung von $\alpha(bin_p(x))$ mit dem neuen $fktwert$ versucht. Tritt allerdings der Fall auf, daß $fktwert$ $(= \alpha(bin_p(x)))$ schon gleich $(1, \ldots, 1)$ ist, so wird die nächstkleinere Stelle x' in der Funktionstabelle von α gesucht, für die $\alpha(bin_p(x'))$ noch nicht gleich $(1, \ldots, 1)$ ist. Für x' und alle größeren schon belegten Stellen x der Funktionstabelle wird die Belegung rückgängig gemacht (einschließlich einer Korrektur der Mengen $S^{(i)}_{\alpha^{(i)}(bin_p(x))\alpha(bin_p(x))}$). Nach Abarbeitung der

while–Schleife ist x für den nächsten Durchlauf von Schritt 4. auf dieses x' gesetzt. Damit im nächsten Durchlauf von Schritt 4. eine Belegung von $\alpha(bin_p(x))$ mit dem um 1 erhöhten Wert versucht wird, wird $fktwert$ noch entsprechend belegt.

Aus Laufzeitgründen berechnet das branch–and–bound–Verfahren nur *einen* erfolgreichen Ast, d.h. nur ein h–Tupel von Zerlegungsfunktionen, die das gestellte Problem lösen.

Obwohl häufig nur kleine Teile des „branch–and–bound–Baumes“ durchlaufen werden (nicht erfolgreiche Äste werden schon früh gekappt), ist die worst–case–Laufzeit des Verfahrens exponentiell. Dies ist auch nicht verwunderlich, denn der Algorithmus löst ein NP–hartes Problem.

Folgendes Problem ist NP–hart:

Problem der gemeinsamen Zerlegungsfunktionen (GZF)

Gegeben.: Funktionen $f_1, \ldots, f_m \in B_n$ mit den Eingangsvariablen $x_1, \ldots, x_p$, $y_1, \ldots, y_q$ und eine Variablenteilmenge $X = \{x_1, \ldots, x_p\}$. Die Anzahl der Zerlegungsfunktionen, die bei Zerlegung von f_i hinsichtlich X benötigt werden, betrage für $1 \leq i \leq m$ $r_i = \lceil \log(vz(X, f_i)) \rceil$.
Eine natürliche Zahl h mit $h \leq r_i$ für alle $1 \leq i \leq m$.

Gesucht.: Gibt es Funktionen $\alpha_1, \ldots, \alpha_h \in B_p$, die alle als Zerlegungsfunktionen in einseitigen Zerlegungen von $f_1, \ldots, f_m$ hinsichtlich X verwendet werden können, d.h. gibt es einseitige Zerlegungen von $f_1, \ldots, f_m$ hinsichtlich X der Form

$$\begin{aligned}
&f_1(x_1, \ldots, x_p, y_1, \ldots, y_q) = \\
&\qquad g^{(1)}(\alpha_1(\mathbf{x}), \ldots, \alpha_h(\mathbf{x}), \alpha_{h+1}^{(1)}(\mathbf{x}), \ldots, \alpha_{r_1}^{(1)}(\mathbf{x}), \mathbf{y}) \\
&\quad \vdots \\
&f_m(x_1, \ldots, x_p, y_1, \ldots, y_q) = \\
&\qquad g^{(m)}(\alpha_1(\mathbf{x}), \ldots, \alpha_h(\mathbf{x}), \alpha_{h+1}^{(m)}(\mathbf{x}), \ldots, \alpha_{r_m}^{(m)}(\mathbf{x}), \mathbf{y}),
\end{aligned}$$

Es gilt:

Satz 4.8 *Das Problem* **GZF** *ist NP–hart.*

Satz 4.8 wird bewiesen durch Angabe einer Polynomzeittransformation vom NP–vollständigen Problem 3–PARTITION nach GZF.

Das Problem 3–PARTITION lautet (siehe [GJ79] oder [Meh84b]):

Problem 3–PARTITION (3P)

Gegeben: Gewichte $a_1, \ldots, a_{3n} \in \mathbf{Z}^+$, eine Grenze $B \in \mathbf{Z}^+$, so daß $B/4 < a_i < B/2 \; \forall i$ und $\sum_{i=1}^{3n} a_i = n \cdot B$.

Gesucht: Gibt es eine Partition $S_1, \ldots, S_n$ von $\{1, \ldots, 3n\}$, so daß $\sum_{j \in S_i} a_j = B \; \forall i$?

Satz 4.9 *Das Problem* **3–PARTITION** *ist NP–vollständig im strengen Sinne, d.h es ist auch NP–vollständig, wenn die Zahlen der Eingabe unär kodiert sind.*

Beweis:
Beweis durch Transformation von 3DM, siehe [GJ79], S. 96ff.

Die Polynomzeittransformation zum Beweis, daß GZF NP–hart ist, erfolgt in 2 Stufen. Zunächst wird gezeigt, daß ein Problem 2^l–PARTITION NP–vollständig ist, das folgendermaßen definiert ist:

Problem 2^l–PARTITION

Gegeben: Ganze Zahlen $c_1, \ldots, c_k \in \mathbf{Z}^+$, eine Grenze $B' = 2^b$ mit $b \in \mathbf{N}$ und eine Zahl $M = 2^l$ mit $l \in \mathbf{N}$, so daß $\sum_{i=1}^{k} c_i = M \cdot B' = 2^{b+l}$ $(k > M)$.

Gesucht: Gibt es eine Partition $T_1, \ldots, T_M$ von $\{1, \ldots, k\}$, so daß $\sum_{j \in T_i} c_j = B'$ für $1 \leq i \leq M$?

Satz 4.10 *Das Problem* **2^l–PARTITION** *ist NP–vollständig im strengen Sinne, d.h. es ist auch NP–vollständig, wenn die Zahlen der Eingabe unär kodiert sind.*

Beweis:
Hier wird nur bewiesen, daß 2^l–PARTITION NP–hart ist. Der Beweis erfolgt durch Polynomzeittransformation von 3–PARTITION nach 2^l–PARTITION.

Sei eine Instanz von 3–PARTITION gegeben durch $a_1, \ldots, a_{3n}, B$.
Dazu wird (in Polynomzeit) eine Instanz von 2^l–PARTITION berechnet mit

$$
\begin{aligned}
B' &= 2^{\lceil \log B \rceil + 2} \quad (\text{d.h. } b = \lceil \log B \rceil + 2) \\
k &= 3n + 2^{\lceil \log n \rceil} \\
c_i &= a_i \quad \forall\, 1 \leq i \leq 3n \\
c_i &= B' - B = 2^{\lceil \log B \rceil + 2} - B \quad \forall\, 3n < i \leq 4n \\
c_i &= B' \quad \forall\, 4n < i \leq 3n + 2^{\lceil \log n \rceil} \\
M &= 2^{\lceil \log n \rceil} \quad (\text{d.h. } l = \lceil \log n \rceil)
\end{aligned}
$$

Es gilt dann:

$$\begin{aligned}\sum_{i=1}^{k} c_i &= \sum_{i=1}^{3n} a_i + \sum_{i=3n+1}^{4n} (B'-B) + \sum_{i=4n+1}^{3n+2^{\lceil \log n \rceil}} B' \\ &= n \cdot B + n \cdot (B'-B) + (2^{\lceil \log n \rceil} - n) \cdot B' \\ &= 2^{\lceil \log n \rceil} \cdot B' \\ &= M \cdot B'\end{aligned}$$

Außerdem gilt $k > M$.

Zu zeigen ist:
$\exists$ Partition $S_1, \ldots, S_n$ von $\{1, \ldots, 3n\}$, so daß $\sum_{j \in S_i} a_j = B \ \forall 1 \leq i \leq n$

$$\Longleftrightarrow$$

$\exists$ Partition $T_1, \ldots, T_M$ von $\{1, \ldots, k\}$, so daß $\sum_{j \in T_i} c_j = B' \ \forall 1 \leq i \leq M$

„$\Longrightarrow$“:

Vorauss.: $\exists$ Partition $S_1, \ldots, S_n$ von $\{1, \ldots, 3n\}$, so daß $\sum_{j \in S_i} a_j = B$ $\forall 1 \leq i \leq n$.
Konstruiere $T_1, \ldots, T_M$ wie folgt:

$$\text{Für } 1 \leq i \leq n : T_i = S_i \cup \{3n+i\}$$
$$\text{Für } n < i \leq 2^{\lceil \log n \rceil} : T_i = \{3n+i\}$$

Da $S_1, \ldots, S_n$ eine Partition von $\{1, \ldots, 3n\}$, folgt, daß $T_1, \ldots, T_M$ eine Partition von $\{1, \ldots, k\}$ ist.
Noch z. z.: $\sum_{j \in T_i} c_j = B'$ für $1 \leq i \leq M$.

<u>1. Fall</u>: $1 \leq i \leq n$:

$$\sum_{j \in T_i} c_j = \sum_{j \in S_i} c_j + c_{3n+i} = \sum_{j \in S_i} a_j + (B'-B) = B + B' - B = B'$$

<u>2. Fall</u>: $n < i \leq 2^{\lceil \log n \rceil}$:

$$\sum_{j \in T_i} c_j = c_{3n+j} = B'$$

„$\Longleftarrow$“:

Vorauss.: $\exists$ Partition $T_1, \ldots, T_M$ von $\{1, \ldots, k\}$, so daß $\sum_{j \in T_i} c_j = B'$ $\forall 1 \leq i \leq M$
Dann gilt:

- Es gibt $2^{\lceil \log n \rceil} - n$ Mengen T_i mit $T_i = \{3n + j\}$, wobei $j \in \{n + 1, \ldots, 2^{\lceil \log n \rceil}\}$, da $c_{3n+j} = B'$ und $c_i > 0 \; \forall \, 1 \leq i \leq k$.
Seien o.B.d.A. $T_i = \{3n + i\}$ für $n + 1 \leq i \leq 2^{\lceil \log n \rceil}$.
- Für die restlichen T_i mit $1 \leq i \leq n$ gilt:
Es gibt kein T_i mit $\{i_1, i_2\} \subseteq T_i$ und $3n < i_1, i_2 \leq 4n$, denn sonst gilt:

$$\begin{aligned} c_{i_1} + c_{i_2} &= (B' - B) + (B' - B) \\ &= B' + (B' - 2B) \\ &= B' + (2^{\lceil \log B \rceil + 2} - 2B) \\ &= B' + (4 \cdot 2^{\lceil \log B \rceil} - 2B) \\ &\geq B' + (4B - 2B) \\ &= B' + 2B \\ &> B', \quad \text{da } B > 0. \end{aligned}$$

$\Longrightarrow \; \forall \, 1 \leq i \leq n$ gilt: T_i enthält höchstens ein i_1 mit $3n < i_1 \leq 4n$.
Da aber alle i_1 mit $3n < i_1 \leq 4n$ in einer der Mengen T_i $(1 \leq i \leq n)$ enthalten sein müssen, gilt $\forall \, 1 \leq i \leq n$:
T_i enthält *genau* ein i_1 mit $3n < i_1 \leq 4n$.
Gelte o.B.d.A.: $3n + i \in T_i \quad \forall \, 1 \leq i \leq n$

Wähle nun:

$$S_i = T_i \setminus \{3n + i\} \quad \forall \, 1 \leq i \leq n$$

$$\Longrightarrow \qquad S_i \subseteq \{1, \ldots, 3n\}$$

$S_1, \ldots S_n$ bilden eine Partition von $\{1, \ldots, 3n\}$.
Für $1 \leq i \leq n$ gilt:

$$\sum_{j \in S_i} a_j = \sum_{j \in S_i} c_j = \sum_{j \in T_i} c_j - c_{3n+i} = B' - (B' - B) = B.$$

□

Nachdem nun gezeigt ist, daß 2^l–PARTITION NP–hart ist (auch bei unärer Kodierung der vorkommenden Zahlen), kann Satz 4.8 bewiesen werden:

Beweis:
Beweis durch Polynomtransformation von 2^l–Partition nach GZF:
Sei eine Instanz von 2^l–Partition gegeben durch $c_1, \ldots, c_k \in \mathbf{Z}^+$, $B' = 2^b$, $M = 2^l$ mit $\sum_{i=1}^{k} c_i = M \cdot B' = 2^{b+l}$, $k > M$.
Alle Zahlen sind unär kodiert!
Daraus wird nun in Polynomzeit eine Instanz des Problems GZF berechnet:
Es werden Funktionen f_1 und $f_2 \in B_n$ berechnet mit den Eingangsvariablen $x_1, \ldots, x_p$, $y_1, \ldots, y_p$, $p = l + b + 1$. Die Funktionen werden durch ihre Zerlegungsmatrizen Z_1 und Z_2 hinsichtlich der Variablenteilmenge $X = \{x_1, \ldots, x_p\}$ definiert. Die Zerlegungsmatrizen sind quadratisch mit 2^{l+b+1} Zeilen und 2^{l+b+1} Spalten.

Zur Konstruktion der Zerlegungsmatrizen:
Die Matrizen werden aus k Blöcken konstruiert, entsprechend den k Gewichten $c_1, \ldots, c_k$. (Die Blöcke werden gerade so konstruiert, daß gemeinsame Zerlegungsfunktionen von f_1 und f_2 den Zeilen der einzelnen Blöcke den gleichen Funktionswert zuordnen müssen.)
Bei der Definition von Z_1 und Z_2 wird folgende Kurzschreibweise benutzt: $un(i)$ ist definiert als

$$\underbrace{1\ldots1}_{i \text{ Mal}} \quad \underbrace{0\ldots0}_{2^{l+b+1}-i \text{ Mal}} .$$

Also steht $un(i)$ für eine Zerlegungsmatrixzeile, die mit i Einsen beginnt, und dann nur noch Nullen enthält.
Aufbau von Block Nr. i:

1. Fall: $c_i = 1$

Block i besteht sowohl für Z_1 als auch für Z_2 aus zwei gleichen Zeilen:

Für Z_1: $\boxed{\begin{array}{c} un(\sum_{j=1}^{i-1} c_j) \\ un(\sum_{j=1}^{i-1} c_j) \end{array}}$ Für Z_2: $\boxed{\begin{array}{c} un(\sum_{j=1}^{i-1} c_j) \\ un(\sum_{j=1}^{i-1} c_j) \end{array}}$

2. Fall: $c_i > 1$

Block i besteht für Z_1 und Z_2 aus $2c_i$ Zeilen:

Für Z_1: $\boxed{\begin{array}{c} un(\sum_{j=1}^{i-1} c_j) \\ un(\sum_{j=1}^{i-1} c_j) \\ un(\sum_{j=1}^{i-1} c_j + 1) \\ un(\sum_{j=1}^{i-1} c_j + 1) \\ \vdots \\ un(\sum_{j=1}^{i-1} c_j + (c_i - 2)) \\ un(\sum_{j=1}^{i-1} c_j + (c_i - 1)) \\ un(\sum_{j=1}^{i-1} c_j + (c_i - 1)) \end{array}}$ Für Z_2: $\boxed{\begin{array}{c} un(\sum_{j=1}^{i-1} c_j) \\ un(\sum_{j=1}^{i-1} c_j + 1) \\ un(\sum_{j=1}^{i-1} c_j + 1) \\ un(\sum_{j=1}^{i-1} c_j + 2) \\ \vdots \\ un(\sum_{j=1}^{i-1} c_j + (c_i - 1)) \\ un(\sum_{j=1}^{i-1} c_j + (c_i - 1)) \\ un(\sum_{j=1}^{i-1} c_j) \end{array}}$

Die beiden Zerlegungsmatrizen haben $2\sum_{i=1}^{k} c_i = 2MB' = 2^{l+b+1}$ Zeilen und $2MB'$ Spalten und sind in polynomieller Zeit berechenbar.
Da Zeilen aus verschiedenen Blöcken verschieden sind und bei den Zeilen *eines* Blockes immer je 2 Zeilen paarweise gleich sind, beträgt die Anzahl *verschiedener* Zeilen $\frac{1}{2}(2^{l+b+1}) = 2^{l+b}$.
$\Longrightarrow$ Die minimale Anzahl der Zerlegungsfunktionen bei Zerlegung hinsichtlich X beträgt sowohl bei f_1 als auch bei f_2 $\lceil \log(vz(X, f_1)) \rceil = \lceil \log(vz(X, f_2)) \rceil = l + b$.
Schließlich wird die natürliche Zahl h gewählt als $h = l$, d.h. das Problem besteht darin, ob es Funktionen $\alpha_1, \ldots, \alpha_l \in B_p$ gibt, die alle als gemeinsame Zerlegungsfunktionen in einseitigen Zerlegungen von f_1 und f_2 hinsichtlich X verwendet werden können.
Zu zeigen ist nun: $\exists$ Partition $T_1, \ldots, T_M$ von $\{1, \ldots, k\}$, so daß $\sum_{j \in T_i} c_j = B'$ $\forall 1 \leq i \leq M$

$$\Longleftrightarrow$$

$\exists h = l$ gemeinsame Zerlegungsfunktionen von f_1, f_2.

Bevor diese Äquivalenz bewiesen wird, wird noch eine Schreibweise definiert: Die Zerlegungsmatrizen wurden durch k Blöcke definiert. $\{0,1\}^p$ zerfällt in k disjunkte Mengen $block_1, \ldots, block_k$, wenn man definiert:

$$(x_1, \ldots, x_p) \in block_i$$

$$\Longleftrightarrow$$

Die Zeile mit Nummer $int(x_1, \ldots, x_p)$ fällt in Block i.

„$\Longrightarrow$":

Vorauss.: $\exists$ Partition $T_1, \ldots, T_M$ von $\{1, \ldots, k\}$, so daß $\sum_{j \in T_i} c_j = B'$ $\forall 1 \leq i \leq M$.

Es werden nun l gemeinsame Zerlegungsfunktionen $\alpha_1, \ldots, \alpha_l$ von f_1 und f_2 definiert:

Für alle $1 \leq i \leq 2^l$ und alle $j \in T_i$ setze für alle $\mathbf{x} = (x_1, \ldots, x_p) \in block_j$

$$\alpha(\mathbf{x}) = (\alpha_1(\mathbf{x}), \ldots, \alpha_l(\mathbf{x})) = bin_l(i-1)$$

(D.h. falls $T_i = \{i_1, \ldots, i_q\}$, dann erhalten alle Blöcke $block_{i_1}, \ldots block_{i_q}$ den Funktionswert $bin_l(i-1)$.)

Die Anzahl verschiedener Zeilen $anz_{bin_l(i-1)}$ der Zerlegungsmatrizen, denen durch α der Funktionswert $bin_l(i-1)$ zugeordnet wird, beträgt (sowohl für Z_1 als auch für Z_2):

$$\begin{aligned} anz_{bin_l(i-1)} &= \sum_{j \in T_i} (\text{Anzahl verschiedener Zeilen in Block } j) \\ &= \sum_{j \in T_i} c_j \\ &= B' = 2^b \end{aligned}$$

$$\Longrightarrow vz(X, f_1, \alpha) = vz(X, f_2, \alpha) = 2^b.$$

Nach Lemma 4.10 existieren daher einseitige Zerlegungen von f_1 und f_2 hinsichtlich X der Form

$$\begin{aligned} &f_1(x_1, \ldots, x_p, y_1, \ldots, y_p) = \\ &\quad g^{(1)}(\alpha_1(\mathbf{x}), \ldots, \alpha_l(\mathbf{x}), \alpha^{(1)}_{l+1}(\mathbf{x}), \ldots, \alpha^{(1)}_{l+b}(\mathbf{x}), y_1, \ldots, y_p) \\ &f_2(x_1, \ldots, x_p, y_1, \ldots, y_p) = \\ &\quad g^{(2)}(\alpha_1(\mathbf{x}), \ldots, \alpha_l(\mathbf{x}), \alpha^{(2)}_{l+1}(\mathbf{x}), \ldots, \alpha^{(2)}_{l+b}(\mathbf{x}), y_1, \ldots, y_p) \end{aligned}$$

„$\Longleftarrow$“:

Vorauss.: $\exists$ l gemeinsame Zerlegungsfunktionen $\alpha_1, \ldots, \alpha_l$ von f_1 und f_2.

Da Z_1 und Z_2 jeweils 2^{l+b} verschiedene Zeilen haben und die Gesamtzahl der Zerlegungsfunktionen bei der Zerlegung von f_1 bzw. f_2 jeweils $l+b$ beträgt, ist es nicht möglich, daß Zerlegungsfunktionen den Indizes *gleicher* Zeilen von Z_1 bzw. von Z_2 *verschiedene* Zerlegungsfunktionswerte zuordnen.
$\Longrightarrow$ Auch $\alpha_1, \ldots, \alpha_l$ ordnen den Indizes gleicher Zeilen gleiche Werte zu.

<u>Beh.1</u>: $\alpha(x_1, \ldots, x_p) = \alpha(x'_1, \ldots, x'_p)$ für alle $(x_1, \ldots, x_p)$, $(x'_1, \ldots, x'_p) \in block_i$ $(1 \leq i \leq k)$.
Beweis:
$\alpha_1, \ldots, \alpha_l$ sind gemeinsame Zerlegungsfunktionen von f_1 und f_2.
Zeilen 1 und 2 von Block i in Z_1 sind gleich $\Rightarrow$ α liefert auf den Indizes von Zeilen 1 und 2 den gleichen Funktionswert.
Zeilen 2 und 3 von Block i in Z_2 sind gleich $\Rightarrow$ α liefert auf den Indizes von Zeilen 2 und 3 den gleichen Funktionswert.
Betrachtet man weiter Zeile 3 und 4 von Block i in Z_1 usw., so erhält man die Behauptung.

<u>Beh.2</u>: Sowohl bei f_1 als auch bei f_2 ordnet α den Indizes von *genau* $2^b = B'$ verschiedenen Zeilen gleiche Funktionswerte zu.
Beweis:

- Nach Lemma 4.10 ist $vz(X, f_1, \alpha) \leq 2^b$ und $vz(X, f_2, \alpha) \leq 2^b$. Für jeden Funktionswert von α ist also die Zahl der verschiedenen Zeilen, deren Indizes dieser Funktionswert zugeordnet wird, kleiner oder gleich 2^b.
- Es gibt $2^l \cdot 2^b$ verschiedene Zeilen der Zerlegungsmatrizen Z_1 bzw. Z_2 und genau 2^l verschiedene Funktionswerte von α sind möglich. Die Anzahl der verschiedenen Zeilen, deren Indizes der gleiche Funktionswert zugeordnet wird, beträgt also *genau* 2^b.

Aus den Behauptungen 1 und 2 ergibt sich, daß sich die Blöcke der Zerlegungsmatrizen in Gruppen einteilen lassen, wobei die Anzahl der verschiedenen Zeilen in einer solchen Gruppe gerade 2^b beträgt.

Definiere nun für $i = 1, \ldots, 2^l$:

$$T_i = \{j \mid \alpha(\mathbf{x}) = bin_l(i-1) \, \forall \, \mathbf{x} \in block_j\}.$$

$T_1, \ldots, T_M$ bilden eine Partition von $\{1, \ldots, k\}$ wegen Behauptung 1.
Die Anzahl verschiedener Zeilen, deren Indizes durch α der Funktionswert $bin_l(i-1)$ zugeordnet wird, ergibt sich als

$$2^b = B' = \sum_{j \in T_i} (\text{Anzahl verschiedener Zeilen in Block } j) = \sum_{j \in T_i} c_j.$$

Also gilt für die angegebene Partition $T_1, \ldots, T_M$:

$$\sum_{j \in T_i} c_j = B'$$

und $T_1, \ldots, T_M$ stellt eine Lösung der gegebenen Instanz des 2^l–PARTITION–Problems dar. □

Aus dem Beweis ergibt sich folgendes Korollar:

Korollar 4.4 *Das Problem GZF bleibt NP–hart, auch wenn die Anzahl der gegebenen booleschen Funktionen gleich 2 ist.*

Das beschriebene branch–and–bound–Verfahren hat noch 2 Nachteile:

1. Wie schon angesprochen wurde, ist die Laufzeit des Verfahrens im worst case exponentiell.

2. Ist bei einer Zerlegung einer Funktion f_i hinsichtlich $X = \{x_1, \ldots, x_p\}$ die Anzahl der verschiedenen Zeilen der Zerlegungsmatrix $vz(X, f_i)$ keine Zweierpotenz, so können die Zerlegungsfunktionen

   $$\alpha_1, \ldots, \alpha_r \ (r = \lceil vz(X, f_i) \rceil)$$

   mehr als $vz(X, f_i)$ verschiedene Werte annehmen. Es kann daher Zerlegungsfunktionen $(\alpha_1, \ldots, \alpha_r)$ geben, die *verschiedenene* Zerlegungsfunktionswerte auf den Indizes *gleicher* Zeilen liefern. Es kann vorkommen, daß der branch–and–bound–Algorithmus solche Zerlegungsfunktionen berechnet. Der Nachteil besteht dabei darin, daß dadurch evtl. Symmetrieeigenschaften der ursprünglichen Funktion verloren gehen. Wenn die Funktion f_i G–symmetrisch ist, wobei die Automorphismen aus G auf den Variablen aus X operieren, so müssen die berechneten Zerlegungsfunktionen nicht G–symmetrisch sein.
   Beispiel:
   Ist f_i symmetrisch in den Variablen x_1 und x_2, d.h. invariant gegenüber Vertauschung von x_1 und x_2, so müssen alle Zeilen der Zerlegungsmatrix von f_i hinsichtlich X mit Indizes $(0, 1, \epsilon_3, \ldots, \epsilon_p)$ und $(1, 0, \epsilon_3, \ldots, \epsilon_p)$ $(\epsilon_3, \ldots, \epsilon_p \in \{0, 1\})$ gleich sein. Falls die Zerlegungsfunktionen $\alpha_1, \ldots, \alpha_r$ *nicht* auf den Indizes gleicher Zerlegungsmatrixzeilen gleiche Funktionswerte liefern müssen, so kann für ein $(\epsilon_3, \ldots, \epsilon_p) \in \{0, 1\}^{p-2}$ gelten:

   $$\alpha(0, 1, \epsilon_3, \ldots, \epsilon_p) \neq \alpha(1, 0, \epsilon_3, \ldots, \epsilon_p),$$

   so daß also mindestens eine der Zerlegungsfunktionen α_i nicht symmetrisch in x_1 und x_2 ist.

   Wie in den vorangegangenen Kapiteln dargestellt wurde, sind Symmetrieeigenschaften wichtige Eigenschaften boolescher Funktionen, durch die sich Funktionen, die in der Praxis vorkommen, häufig von zufällig gewählten Funktionen unterscheiden. Daher ist es erstrebenswert, vorhandene Symmetrieeigenschaften möglichst zu erhalten.

Um den genannten Nachteilen zu entgehen, erfolgt hier eine Beschränkung in der Wahl der Zerlegungsfunktionen. Es wird nur noch nach solchen Zerlegungsfunktionen gesucht, die den Indizes gleicher Zerlegungsmatrixzeilen auch gleiche Funktionswerte zuordnen. Solche Zerlegungsfunktionen werden im folgenden als *gleichheitserhaltend* bezeichnet.

Definition 4.7 (Gleichheitserhaltende Zerlegungsfunktionen)
Gibt es zu $f \in B_n$ eine Zerlegung hinsichtlich $X = \{x_1, \ldots, x_p\}$ der Form

$$f(x_1, \ldots, x_p, y_1, \ldots, y_q) = g(\alpha_1(x_1, \ldots, x_p), \ldots, \alpha_r(x_1, \ldots, x_p), y_1, \ldots, y_q),$$

so heißen die Zerlegungsfunktionen $\alpha_1, \ldots, \alpha_r$ **gleichheitserhaltend**, *wenn gilt: Falls für alle $(y_1, \ldots, y_q) \in \{0,1\}^q$*

$$f(x_1^{(1)}, \ldots, x_p^{(1)}, y_1, \ldots, y_q) = f(x_1^{(2)}, \ldots, x_p^{(2)}, y_1, \ldots, y_q),$$

dann gilt auch für alle $1 \le i \le r$:

$$\alpha_i(x_1^{(1)}, \ldots, x_p^{(1)}) = \alpha_i(x_1^{(2)}, \ldots, x_p^{(2)}).$$

Aus der Definition von gleichheitserhaltenden Zerlegungsfunktionen ergibt sich direkt die folgende Aussage:

Lemma 4.12 *Eine Funktion $f \in B_n$ besitze eine Zerlegung hinsichtlich $X = \{x_1, \ldots, x_p\}$ der Form*

$$f(x_1, \ldots, x_p, y_1, \ldots, y_q) = g(\alpha_1(x_1, \ldots, x_p), \ldots, \alpha_r(x_1, \ldots, x_p), y_1, \ldots, y_q)$$

mit gleichheitserhaltenden Zerlegungsfunktionen. Sei G eine Gruppe von Automorphismen, die auf den Variablen aus X operieren und sei f G–symmetrisch. Dann sind auch die Zerlegungsfunktionen $\alpha_1, \ldots, \alpha_r$ G–symmetrisch.

Beschränkt man sich also auf die Wahl gleichheitserhaltender Zerlegungsfunktionen, so bleiben Symmetrieeigenschaften erhalten.

Betrachtet man dieses eingeschränkte Problem, so erkennt man, daß man nun auch bei der Laufzeit Gewinne erzielen kann:
Will man gemeinsame, gleichheitserhaltende Zerlegungsfunktionen für die Zerlegung von Funktionen $f_1, \ldots, f_m$ hinsichtlich X ($|X| = p$) bestimmen, so kann man in einer Vorberechnung die Zeilen der zugehörigen Zerlegungsmatrizen in verschiedene Klassen einteilen, von denen man weiß, daß alle gemeinsamen, gleichheitserhaltenden Zerlegungsfunktionen den Indizes dieser Zeilen den gleichen Funktionswert zuordnen müssen.

Die Gleichheit auf Zerlegungsmatrixzeilen liefert für $f_1, \ldots, f_m$ jeweils Äquivalenzklasseneinteilungen auf $\{0,1\}^p$. Seien diese Äquivalenzklasseneinteilungen gegeben durch

$$P_i = \{K_1^{(i)}, \ldots, K_{vz(X,f_i)}^{(i)}\} \;\forall\, 1 \le i \le m.$$

Sind durch $\alpha = (\alpha_1, \ldots, \alpha_h)$ gemeinsame, gleichheitserhaltende Zerlegungsfunktionen gegeben, so muß für $x^{(1)}, x^{(2)} \in \{0,1\}^p$ $\alpha(x^{(1)}) = \alpha(x^{(2)})$ gelten, falls es ein $i \in \{1, \ldots, m\}$ gibt mit

$$x^{(1)}, x^{(2)} \in K_j^{(i)} \text{ für } 1 \leq j \leq vz(X, f_i),$$

d.h. falls die Zeilen $int(x^{(1)})$ und $int(x^{(2)})$ der Zerlegungsmatrix einer Funktion f_i gleich sind (α ist gleichheitserhaltend!).

Definiert man

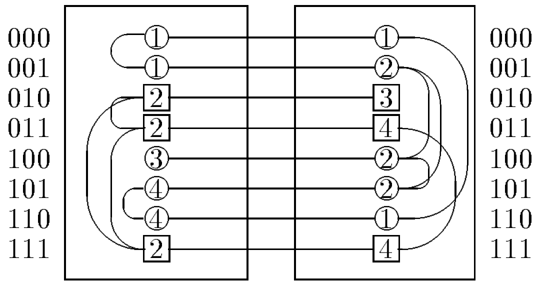
$$x^{(1)} \sim x^{(2)} \;\Leftrightarrow\; \exists 1 \leq i \leq m \text{ mit } x^{(1)}, x^{(2)} \in K_j^{(i)} \text{ für} 1 \leq j \leq vz(X, f_i),$$

so liefert der transitive Abschluß von „$\sim$“ eine Äquivalenzklasseneinteilung $P = \{E_1, \ldots, E_k\}$ auf $\{0,1\}^p$, wobei für alle $x^{(1)}, x^{(2)} \in E_i$ $(1 \leq i \leq k)$ gilt: Alle gleichheitserhaltenden gemeinsamen Zerlegungsfunktionen von $f_1, \ldots, f_m$ liefern auf $x^{(1)}$ und $x^{(2)}$ den gleichen Funktionswert.

Die Bestimmung der Äquivalenzklasseneinteilung P wird an einem Beispiel verdeutlicht:

Beispiel 6:

Im vorliegenden Beispiel werden die Zerlegungsmatrizen von zwei Funktionen f_1 und f_2 betrachtet. Zeilen mit gleichen Nummern stehen für gleiche Zeilen der Zerlegungsmatrizen.



Der Zusammenhang zwischen den einzelnen Zeilen der Zerlegungsmatrizen ist im Bild durch die Kanten eines Graphen ausgedrückt. Zwei Zeilen einer Zerlegungsmatrix werden genau dann verbunden, wenn sie gleich sind. Außerdem werden bei beiden Zerlegungsmatrizen alle Zeilen mit gleichem Index verbunden. Die oben beschriebene Äquivalenzklasseneinteilung erhält man dann durch Bestimmung der Zusammenhangskomponenten dieses Graphen. Es gibt genau zwei Zusammenhangskomponenten, dargestellt durch $\circ$ und $\Box$. Damit ergibt sich für die Äquivalenzklasseneinteilung:

$$P = \{(000), (001), (100), (101), (110)\}, \{(010), (011), (111)\}\}$$

Sind $(\alpha_1, \ldots, \alpha_h) = \alpha$ gemeinsame, gleichheitserhaltende Zerlegungsfunktionen von f_1 und f_2, so ist mit Angabe des Funktionswertes von α auf (000) auch der Funktionswert auf (001), (100), (101) und (110) festgelegt.

Will man für die Zerlegung von Funktionen $f_1, \ldots, f_m$ gemeinsame, gleichheitserhaltende Zerlegungsfunktionen $\alpha_1, \ldots, \alpha_h$ durch ein branch–and–bound–Verfahren bestimmen, so kann man die Eingabe für das Verfahren durch Vorberechnung einer solchen Äquivalenzklasseneinteilung im allgemeinen stark verkleinern. Wie im obigen Beispiel angedeutet, läßt sich die Äquivalenzklasseneinteilung P leicht in linearer Zeit durch Bestimmung der Zusammenhangskomponenten eines (impliziten) Graphen auf den Zerlegungsmatrixzeilen bestimmen.

Ein modifiziertes branch–and–bound–Verfahren baut die Funktionstabelle für $\alpha = (\alpha_1, \ldots, \alpha_h)$ nicht Zeile für Zeile auf, sondern weist den *einzelnen Äquivalenzklassen* E_i aus P Funktionswerte zu. Dadurch wird der Aufwand wesentlich verringert.

Zur Durchführung des modifizierten branch–and–bound–Verfahrens wird für jede Funktion f_i $(1 \leq i \leq m)$ und jede Äquivalenzklasse E_j vorberechnet, wieviele *verschiedene* Zeilen der Zerlegungsmatrix Z_i von f_i einen Index aus E_j haben. Dazu benutzt man die schon oben erwähnte Äquivalenzklasseneinteilung $P_i = \{K_1^{(i)}, \ldots, K_{vz(X,f_i)}^{(i)}\}$, die durch die Gleichheit auf Zeilen von Z_i induziert wird. Man berechnet Mengen

$$ZNR_j^{(i)} = \{l \mid K_l^{(i)} \cap E_j \neq \emptyset\}.$$

Betrachtet man alle Zeilen der Zerlegungsmatrix Z_i mit Index aus E_j (d.h. alle Zeilen „aus der j. Zusammenhangskomponente“), so wird jede *verschiedene* Zeile in $ZNR_j^{(i)}$ durch eine Nummer repräsentiert, nämlich gerade durch den Index der zugehörigen Äquivalenzklasse $K_l^{(i)}$. $|ZNR_j^{(i)}|$ gibt also an, wieviele der Zeilen von Z_i, die einen Index aus E_j haben, verschieden sind.

Für $1 \leq j_1, j_2 \leq k$, $j_1 \neq j_2$, gilt

$$ZNR_{j_1}^{(i)} \cap ZNR_{j_2}^{(i)} = \emptyset,$$

da die Zeilen von Z_i mit Indizes in E_{j_1} und die Zeilen mit Indizes in E_{j_2} verschieden sind. (Denn gleiche Zeilen sind „in der gleichen Zusammenhangskomponente“.)

Das ursprüngliche branch–and–bound–Verfahren berechnet bei Vorgabe von Zerlegungsfunktionen $(\alpha_1^{(i)}, \ldots, \alpha_{r_i}^{(i)}) = \alpha^{(i)}$ für jede der Funktionen f_i gemeinsame Zerlegungsfunktionen $\alpha_1, \ldots, \alpha_h$ und verwaltet dabei Mengen $S_{aa'}^{(i)}$. (Betrachtet man im Verlauf des Verfahrens alle binären Zeilenindizes x von Z_i, für die der Funktionswert von α durch das branch–and–bound–Verfahren bestimmt worden ist und für die $(\alpha^{(i)}, \alpha)(x) = (aa')$ ist, so umfaßt $S_{aa'}^{(i)}$ gerade die Indizes $j_1, \ldots, j_l$ der Äquivalenzklassen (hinsichtlich Zeilengleichheit) $K_{j_1}^{(i)}, \ldots, K_{j_l}^{(i)}$, in die diese binären Zeilenindizes fallen.)
Wird der Funktionswert von α für ein weiteres $x \in \{0,1\}^p$ z.B. auf den Wert a' festgelegt, so wird $S_{\alpha^{(i)}(x)a'}^{(i)}$ aktualisiert durch

$$S_{\alpha^{(i)}(x)a'}^{(i)} = S_{\alpha^{(i)}(x)a'}^{(i)} \cup \{znr_i(x)\}.$$

($znr_i(x) = j$ mit $x \in K_j^{(i)}$.)

Legt man nun in dem modifizierten branch–and–bound–Verfahren die Funktionswerte von α für alle $x \in E_j$ (E_j aus P) beispielsweise auf a' fest, so erfolgt die Aktualisierung von $S^{(i)}_{\alpha^{(i)}(x)a'}$ durch

$$S^{(i)}_{\alpha^{(i)}(x)a'} = S^{(i)}_{\alpha^{(i)}(x)a'} \cup ZNR_j^{(i)} \quad (x \in E_j).$$

(Dabei wird vorausgesetzt, daß auch $\alpha^{(i)}$ gleichheitserhaltend ist, so daß $\alpha^{(i)}(x) = \alpha^{(i)}(y)$ für alle $x, y \in E_j$.)

Hierbei ergibt sich ein weiterer Vorteil des modifizierten Verfahrens:
Zum Test, ob das bisher konstruierte Teilstück der Funktionstabelle von α das Kriterium von Lemma 4.11 evtl. schon verletzt, genügt die Kenntnis der *Mächtigkeiten* der Mengen $S^{(i)}_{aa'}$. Da die Mengen $ZNR^{(i)}_{j_1}$ und $ZNR^{(i)}_{j_2}$ für $j_1 \neq j_2$ disjunkt sind, handelt es sich bei den Mengenvereinigungen

$$S^{(i)}_{\alpha^{(i)}(x)a'} \cup ZNR_j^{(i)}$$

immer um *disjunkte* Vereinigungen. Wenn man sich also ohnehin nur für $|S^{(i)}_{\alpha^{(i)}(x)a'}|$ interessiert, genügt es, die Mächtigkeit von $ZNR_j^{(i)}$ zur Mächtigkeit von $S^{(i)}_{\alpha^{(i)}(x)a'}$ zu addieren. Man kann also die *Verwaltung von Mengen* $S^{(i)}_{aa'}$ durch die *Verwaltung von ganzen Zahlen* ersetzen. Bezeichnet man $|S^{(i)}_{aa'}|$ noch mit $MS^{(i)}_{aa'}$, so erhält man folgenden modifizierten branch–and–bound–Algorithmus:

Eingabe:
- Funktionen $f_1, \ldots, f_m$ aus B_n mit den Eingangsvariablen $x_1, \ldots, x_p, y_1, \ldots, y_q$.
- Für $1 \leq i \leq m$: Zerlegungsmatrizen Z_i von f_i hinsichtlich $X = \{x_1, \ldots, x_p\}$ und $r_i = \lceil \log(vz(X, f_i)) \rceil$.
- Für $1 \leq i \leq m$: Sei $\{K_1^{(i)}, \ldots, K^{(i)}_{vz(X,f_i)}\}$ die durch Zeilengleichheit in Z_i auf $\{0,1\}^p$ induzierte Äquivalenzklasseneinteilung.

- Eine Äquivalenzklasseneinteilung $P = \{E_1, \ldots, E_k\}$ auf $\{0,1\}^p$, wie oben definiert. („Zusammenhangskomponenten")
- Für $1 \leq i \leq m$ und alle $E_j \in P$:
  $ZNR_j^{(i)} = \{l \mid K_l^{(i)} \cap E_j \neq \emptyset\}$
  $MZNR_j^{(i)} = |ZNR_j^{(i)}|$
- Für $1 \leq i \leq m$: *Gleichheitserhaltende* Zerlegungsfunktionen $\alpha_1^{(i)}, \ldots, \alpha_{k_i}^{(i)} \in B_p$.
- Natürliche Zahl h mit $h \leq r_i - k_i$ für alle $1 \leq i \leq m$. (h gibt die Anzahl gemeinsamer gleichheitserhaltender Zerlegungsfunktionen an, nach denen gesucht wird.)

Ausgabe: • $\alpha_1, \ldots, \alpha_h \in B_p$, so daß es einseitige Zerlegungen von $f_1, \ldots, f_m$ hinsichtlich X gibt der Form

$$\begin{aligned}
&f_1(x_1, \ldots, x_p, y_1, \ldots, y_q) = \\
&\quad g^{(1)}(\alpha_1^{(1)}(\mathbf{x}), \ldots, \alpha_{k_1}^{(1)}(\mathbf{x}), \alpha_1(\mathbf{x}), \ldots, \alpha_h(\mathbf{x}), \\
&\qquad\qquad \alpha_{k_1+h+1}^{(1)}(\mathbf{x}), \ldots, \alpha_{r_1}^{(1)}(\mathbf{x}), y_1, \ldots, y_q) \\
&\vdots \\
&f_m(x_1, \ldots, x_p, y_1, \ldots, y_q) = \\
&\quad g^{(m)}(\alpha_1^{(m)}(\mathbf{x}), \ldots, \alpha_{k_m}^{(m)}(\mathbf{x}), \alpha_1(\mathbf{x}), \ldots, \alpha_h(\mathbf{x}), \\
&\qquad\qquad \alpha_{k_m+h+1}^{(m)}(\mathbf{x}), \ldots, \alpha_{r_m}^{(m)}(\mathbf{x}), y_1, \ldots, y_q),
\end{aligned}$$

wobei $\alpha_1, \ldots, \alpha_h$ gleichheitserhaltend sind. (Falls es überhaupt h Funktionen aus B_p mit dieser Eigenschaft gibt.)

- Sonst: Eine Meldung daß es keine h Funktionen aus B_p mit der angegebenen Eigenschaft gibt.

Algorithmus:

Konstruiere zunächst eine „komprimierte Version“ β von α auf den Äquivalenzklassen von P. Es gilt dann für alle $1 \leq j \leq k$:

$$\beta(j) = \epsilon \iff \alpha(x) = \epsilon \text{ für alle } x \in E_j.$$

1. Falls die Anzahl k der Äquivalenzklassen aus P gleich 1 ist:
   Gebe aus, daß es kein solches α gibt, beende den Algorithmus.

2. Für alle $1 \leq i \leq m$, $a \in \{0,1\}^{k_i}$, $a' \in \{0,1\}^h$:
   $MS_{aa'}^{(i)} = 0$
   $\beta(j)$ sei undefiniert für alle $1 \leq j \leq k$.
3. Setze $\beta(1) = (0, \ldots, 0)$. Sei $x \in E_1$.
   $\forall 1 \leq i \leq m$: $MS^{(i)}_{\alpha^{(i)}(x)(0,\ldots,0)} = MS^{(i)}_{\alpha^{(i)}(x)(0,\ldots,0)} + MZNR_1^{(i)}$
   Falls für $1 \leq i \leq m$ $MS^{(i)}_{\alpha^{(i)}(x)(0,\ldots,0)} > 2^{r_i - k_i - h}$:
   Gebe aus, daß es kein solches α gibt, beende den Algorithmus.

4. $j = 2$
   $fktwert = (0, \ldots, 0)$
5. $\beta(j) = fktwert$
   if $\forall\ 1 \leq i \leq m$ **and** $x \in E_j$:
   $MS^{(i)}_{\alpha^{(i)}(x)fktwert} + MZNR_j^{(i)} \leq 2^{r_i - k_i - h}$ **then**
   $\forall 1 \leq i \leq m$: $MS^{(i)}_{\alpha^{(i)}(x)fktwert} = MS^{(i)}_{\alpha^{(i)}(x)fktwert} + MZNR_j^{(i)}$
   $j = j + 1$
   $fktwert = (0, \ldots, 0)$

else
  while $\beta(j) = (1, \ldots, 1)$ **do**
    $\beta(j) =$ undefiniert
    $j = j - 1$
    $\forall\ 1 \leq i \leq m$ **and** $x \in E_j$:
      $MS^{(i)}_{\alpha^{(i)}(x)\beta(j)} = MS^{(i)}_{\alpha^{(i)}(x)\beta(j)} - MZNR^{(i)}_j$

  od
  $fktwert = bin_h(int(\beta(j)) + 1)$
  $\beta(j) =$ undefiniert

fi

6. Wenn $j = k + 1$, d.h. wenn die Funktionstabelle von β komplett aufgestellt worden ist, dann:
   Für alle $x \in \{0, 1\}^p$ setze
   $\alpha(x) = \beta(j)$, wobei $x \in E_j$.
   Gebe α als Ergebnis aus.

   Wenn $j = 1$, d.h. wenn man am Ausgangspunkt angekommen ist, ohne daß eine Funktion β und damit eine geeignete Funktion α gefunden wurde, dann:
   Gebe aus, daß es kein solches α gibt.
   Sonst:
   Weiter mit Schritt 5.

Häufig ergeben sich bei der Vorberechnung schon sehr große Zusammenhangskomponenten, d.h. sehr große Äquivalenzklassen von Elementen aus $\{0, 1\}^p$, denen durch α gleiche Funktionswerte zugeordnet werden müssen. Dadurch ergibt sich eine starke Verringerung der Laufzeit dieses branch–and–bound–Algorithmus gegenüber der ersten Version.

Trotzdem ist die worst–case–Laufzeit exponentiell. Dies ist wohl auch nicht zu ändern, da man durch genauere Betrachtung des Beweises zu Satz 4.8 folgendes Korollar erhält:

Korollar 4.5 *Das Problem GZF bleibt NP–hart, wenn man an die gemeinsamen Zerlegungsfunktionen $\alpha_1, \ldots, \alpha_h$ die zusätzliche Forderung stellt, daß es sich um gleichheitserhaltende Funktionen handeln muß.*

Die Gültigkeit von Korollar 4.5 ergibt sich mit Hilfe des folgenden (trivialen) Lemma:

Lemma 4.13 *Ist*

$$f(x_1, \ldots, x_p, y_1, \ldots, y_q) = g(\alpha_1(x_1, \ldots, x_p), \ldots, \alpha_r(x_1, \ldots, x_p), y_1, \ldots, y_q)$$

eine einseitige Zerlegung einer Funktion f hinsichtlich der Variablenteilmenge X und gilt $vz(X, f) = 2^r$, so sind $\alpha_1, \ldots, \alpha_r$ gleichheitserhaltend.

Die Zerlegungsmatrizen der im Beweis zu Satz 4.8 konstruierten Funktionen f_1 und f_2 haben jeweils insgesamt 2^{l+b+1} Zeilen, davon 2^{l+b} verschiedene. Die nun gestellte Frage lautet, ob von den jeweils $l+b$ Zerlegungsfunktionen l bei f_1 und f_2 identisch gewählt werden können. Nach dem obigen Lemma sind jedoch bei jeder Zerlegung von f_1 und f_2 hinsichtlich der vorgegebenen Variablenteilmenge die $l+b$ Zerlegungsfunktionen gleichheitserhaltend. Es gibt also genau dann l gemeinsame Zerlegungsfunktionen, wenn es l gemeinsame gleichheitserhaltende Zerlegungsfunktionen gibt.

Trotzdem bedeutet die Tatsache, daß das angegebene Problem NP–hart ist, nicht, daß bei praktischen Beispielen von einer Suche nach gemeinsamen (gleichheitserhaltenden) Zerlegungsfunktionen abzuraten ist:

- Exponentielle *worst–case*–Laufzeit muß nicht unbedingt bedeuten, daß die schweren Beispiele auch häufig vorkommen.
- In vielen Fällen ist zu beobachten, daß die Eingabe des branch–and–bound–Verfahrens durch die Vorberechnung der „Zusammenhangskomponenten" schon stark verringert wird.
- Die Laufzeit des Verfahrens kann durch Heuristiken verbessert werden.
  Eine ganz einfache Heuristik besteht beispielsweise darin, die Reihenfolge der Behandlung der verschiedenen Äquivalenzklassen $E_1, \ldots, E_k$ zu verändern, die im obigen Verfahren zufällig gewählt wurde. Das Verfahren verteilt „Gewichte" $MZNR_j^{(i)}$ auf $MS_{aa'}^{(i)}$ ($a \in \{0,1\}^{k_i}$, $a' \in \{0,1\}^h$), wobei die Summe der Gewichte in $MS_{aa'}^{(i)}$ nicht größer als $2^{r_i-k_i-h}$ werden darf. Man kann zum Beispiel versuchen, zuerst die größeren Gewichte und dann erst die kleineren zu verteilen. Die Behandlung der Äquivalenzklassen E_j erfolgt in der Reihenfolge absteigender Werte von $\sum_{i=1}^{m} MZNR_j^{(i)}$.
- Außerdem ist man nicht in jedem Fall gezwungen, einen Algorithmus zur Lösung des *allgemeinen* Problems anzuwenden. Häufig treten Sonderfälle auf, die effizient zu behandeln sind:
  - In praktischen Beispielen treten oft Fälle auf, bei denen schon anhand der Anzahl k der „Zusammenhangskomponenten" oder der Größe der Gewichte $MZNR_j^{(i)}$ klar wird, daß es unmöglich ist, h gemeinsame gleichheitserhaltende Zerlegungsfunktionen zu finden:
    * Gibt für ein $1 \leq i \leq m$ ein Gewicht $MZNR_j^{(i)}$, das größer ist als die Grenze $2^{r_i-k_i-h}$, so ist es von vornherein unmöglich, daß es h gemeinsame gleichheitserhaltende Zerlegungsfunktionen gibt. (Denn α muß der Äquivalenzklasse E_j eine festen Wert zuordnen.)
    * Ist die Anzahl k der „Zusammenhangskomponenten" für ein $i \in \{1, \ldots, m\}$ kleiner oder gleich 2^{h+k_i-1}, so können ebenfalls keine

h weiteren gemeinsamen gleichheitserhaltenden Zerlegungsfunktionen existieren. Denn gäbe es noch h weitere gemeinsame gleichheitserhaltende Zerlegungsfunktionen, so wären $\alpha_1^{(i)}, \ldots, \alpha_{k_i}^{(i)}, \alpha_1, \ldots, \alpha_h$ alle gleichheitserhaltend.
Die Funktion $(\alpha^{(i)}, \alpha)$ könnte also höchstens k und damit nicht mehr als 2^{h+k_i-1} verschiedene Funktionswerte annehmen. (*)
Dies ergibt aber einen Widerspruch zur Minimalität von r_i ($r_i = \lceil \log(vz(X, f_i)) \rceil$). Es gilt nämlich

$$2^{r_i-1} < vz(X, f_i)$$

und damit (wegen (*))

$$2^{r_i-1-(h+k_i-1)} < vz(X, f_i, (\alpha^{(i)}, \alpha))$$

bzw.

$$vz(X, f_i, (\alpha^{(i)}, \alpha)) > 2^{r_i-k_i-h}.$$

Nach Lemma 4.10 gibt es also keine gewünschte Zerlegung mit $\alpha_1, \ldots, \alpha_h$ als zusätzliche gemeinsame gleichheitserhaltende Zerlegungsfunktionen.

* Ein weiterer Sonderfall tritt auf, wenn $h = \lceil \log k \rceil$. (k ist die Anzahl der „Zusammenhangskomponenten".):
Falls für ein $i \in \{1, \ldots, m\}$ dann $k_i > 0$, so kann es keine weiteren h gemeinsamen, gleichheitserhaltenden Zerlegungsfunktionen geben, da wegen

$$h = \lceil \log k \rceil \;\Rightarrow\; h \geq \log k \;\overset{k_i \geq 1}{\Rightarrow}\; h + k_i \geq \log k + 1 \;\Leftrightarrow\; 2^{h+k_i-1} \geq k$$

der vorher angegebene Sonderfall auftritt.
Falls $k_i = 0$ für alle $1 \leq i \leq m$, so läßt sich ebenfalls leicht entscheiden, ob es h gemeinsame gleichheitserhaltende Zerlegungsfunktionen gibt:
Jede Funktion $\alpha = (\alpha_1, \ldots, \alpha_h) \in B_{p,h}$ kann $2^h \geq k$ verschiedene Funktionswerte annehmen. Man kann α also so wählen, daß α auf allen x aus verschiedenen Äquivalenzklassen E_j verschiedene Funktionswerte liefert. Ist dies der Fall, so sind nach Lemma 4.10 $\alpha_1, \ldots, \alpha_h$ genau dann als gemeinsame, gleichheitserhaltende Zerlegungsfunktionen von $f_1, \ldots, f_m$ verwendbar, wenn für alle $1 \leq i \leq m$ und $1 \leq j \leq k$ gilt:

$$MZNR_j^{(i)} = |ZNR_j^{(i)}| \leq 2^{r_i-h} \text{ und damit } vz(X, f_i, \alpha) \leq 2^{r_i-h}.$$

– Hat man 2 Funktionen gegeben, für die noch keine Zerlegungsfunktionen vorgegeben sind ($k_1 = k_2 = 0$) und sucht man genau eine gemeinsame, gleichheitserhaltende Zerlegungsfunktion α_1, so kann man dieses Problem auch durch dynamische Programmierung lösen (vergleichbar mit der Lösung des Knapsack–Problems, siehe z.B. [Meh84b]). Der entsprechende Algorithmus ist in Anhang A angegeben.

Verwendet man die obigen Bezeichnungen, läßt sich die Laufzeit des Verfahrens durch $O(2^{r_1-1} \cdot 2^{r_2-1} \cdot k) = O(vz(X, f_1) \cdot vz(X, f_2) \cdot k) = O(2^{2p} \cdot k) = O(2^{3p})$ abschätzen.
Bei gleichmächtiger Zerlegung mit $p = \lfloor n/2 \rfloor$ ergibt sich $O(2^{1.5n}) = O(N^{1.5})$ als grobe Laufzeitabschätzung. (N ist die Größe der Zerlegungsmatrix.)

Anwendung der Verfahren bei der Suche nach Zerlegungsfunktionen

Hat man Funktionen $f_1, \ldots, f_m$ gegeben, für die durch die Heuristik aus Abschnitt 4.4.1 eine Zerlegung hinsichtlich der gleichen Variablenteilmenge (bzw. Variablenaufteilung) festgelegt worden ist, so versucht man zunächst möglichst viele gemeinsame Zerlegungsfunktionen für $f_1, \ldots, f_m$ zu finden. Hat man eine maximale Anzahl von gemeinsamen Zerlegungsfunktionen gefunden, so betrachtet man immer kleiner werdende Teilmengen der Funktionen $f_1, \ldots, f_m$ und versucht unter der Voraussetzung der Verwendung der schon gefundenen Zerlegungsfunktionen für diese Teilmengen der Funktionen möglichst viele gemeinsame Zerlegungsfunktionen zu finden. Dabei betrachtet man zunächst die Teilmengen mit Mächtigkeit $m-1$, dann die Teilmengen mit Mächtigkeit $m-2$ usw. bis zu Teilmengen mit Mächtigkeit 2.

Um die Laufzeit des Verfahrens einzuschränken, werden einmal gefundene Zerlegungsfunktionen auf jeden Fall verwendet. Eine einmal erfolgte Wahl von Zerlegungsfunktionen wird nicht zurückgenommen.

Allerdings braucht man im Verlauf des Verfahrens nicht sämtliche zwei– oder mehrelementigen Teilmengen der Funktionen $f_1, \ldots, f_m$ zu betrachten. Sind für eine Funktion f_i schon sämtliche Zerlegungsfunktionen gefunden, so brauchen die Teilmengen, die f_i enthalten, natürlich nicht mehr betrachtet zu werden.

Außerdem tritt häufig der Fall auf, daß es zu zwei Funktionen f_{i_1} und f_{i_2} (i_1, $i_2 \in \{1, \ldots, m\}$) überhaupt keine gemeinsamen Zerlegungsfunktionen gibt. Alle Teilmengen von $f_1, \ldots, f_m$, die f_{i_1} und f_{i_2} enthalten, brauchen dann nicht auf gemeinsame Zerlegungsfunktionen untersucht zu werden. Daher wird in einer Vorberechnung zunächst für alle Paare von Zerlegungsfunktionen getestet, ob es überhaupt *eine* gemeinsame Zerlegungsfunktion bei der Zerlegung dieses Paares von Funktionen gibt. (Beschränkt man sich auf gleichheitserhaltende Zerlegungsfunktionen, so kann man dieses Problem effizient durch dynamisches Programmieren lösen.) Ist dies nicht der Fall, so werden von vornherein sämtliche Teilmengen von $f_1, \ldots, f_m$, die dieses Paar enthalten, ausgeschlossen.[6]

Um bei einer Menge von Funktionen $f_1, \ldots, f_m$ die maximale Anzahl von gemeinsamen Zerlegungsfunktionen zu bestimmen, wird für jede Funktion f_i die Anzahl $r_i - k_i$ der noch zu bestimmenden Zerlegungsfunktionen betrachtet. Es können höchstens $h_{max} = \min_{1 \leq i \leq m}(r_i - k_i)$ gemeinsame Zerlegungsfunktionen

[6] Weiterhin ist es möglich, zur Begrenzung der Laufzeit der Suche nach gemeinsamen Zerlegungsfunktionen die Heuristik aus Abschnitt 4.4.1 so abzuändern, daß die Anzahl der *gemeinsam* zu behandelnden Funktionen $f_1, \ldots, f_m$ durch eine frei zu wählende Obergrenze beschränkt wird.

gefunden werden. Die Maximalzahl h der gemeinsamen Zerlegungsfunktionen wird durch Binärsuche für h zwischen 1 und h_{max} bestimmt.

Sind für einige der Funktionen $f_1, \ldots, f_m$ am Ende noch Zerlegungsfunktionen offen, die nicht als gemeinsame Zerlegungsfunktionen gewählt werden können, so werden diese Zerlegungsfunktionen für die Einzelfunktionen bestimmt. Hierbei besteht die Möglichkeit, diese Funktionen „zufällig“ zu wählen (aber gleichheitserhaltend[7]) oder aber Verfahren aus Abschnitt 4.1.4 zur Berechnung von Zerlegungsfunktionen mit bestimmten Eigenschaften zu verwenden.

Zweiseitige Zerlegungen hinsichtlich einer Variablenaufteilung $\{X, Y\}$ können analog behandelt werden. Man kann man die Zerlegungsfunktionen auf X und auf Y nacheinander betrachten.

4.5 Ergebnisse

Die beschriebenen Verfahren zur Zerlegung boolescher Funktionen mit mehreren Ausgängen wurden implementiert und es wurden Schaltkreise für Beispielfunktionen damit entworfen. In diesem Kapitel sollen einige Beispiele angegeben werden.

Das implementierte Verfahren läßt zunächst eine Reihe von Wahlmöglichkeiten offen:

- Es besteht die Möglichkeit, zwischen einseitigen und zweiseitigen Zerlegungen zu wählen.
- Es besteht die Möglichkeit, bei der Variablenaufteilung zu wählen zwischen gleichmächtigen Variablenaufteilungen oder Variablenaufteilungen in zwei Mengen, wobei die Mächtigkeit der einen Menge durch eine Konstante gegeben ist und die andere Menge die restlichen Variablen enthält.
- Bei der Bestimmung von Zerlegungsfunktionen für Funktionen mit einem Ausgang gibt es ebenfalls mehrere Alternativen:
  - Man versucht nach Möglichkeit (total)symmetrische Zerlegungsfunktionen zu finden.
  - Man versucht Zerlegungsfunktionen zu finden, die von möglichst vielen Eingangsvariablen unabhängig sind.
  - Man wählt (zufällig) gleichheitserhaltende Zerlegungsfunktionen.

[7] Gleichheitserhaltende Funktionen haben neben dem Erhalten vorhandener Symmetrieeigenschaften einen weiteren Vorteil: Bei der Zerlegung einer Funktion f mit gleichheitserhaltenden Zerlegungsfunktionen $\alpha_1, \ldots, \alpha_{\lceil \log vz(X,f) \rceil}$ treten genau $vz(X, f)$ verschiedene Funktionswerte von α auf (d.h. eine minimale Anzahl von Funktionswerten). Im Fall, daß die Anzahl der verschiedenen Zeilen der Zerlegungsmatrix keine Zweierpotenz ist, treten also $2^{\lceil \log vz(X,f) \rceil} - vz(X, f)$ Elemente aus $\{0,1\}^{\lceil \log vz(X,f) \rceil}$ nicht im Bild von α auf. Die Zusammensetzungfunktion ist dann partiell, sie hat an den entsprechenden Stellen don't cares. Diese don't cares können bei der Realisierung der Zusammensetzungsfunktion ausgenutzt werden.

Auch Kombinationen dieser Strategien sind möglich, wobei eine Reihenfolge angegeben wird, welche Strategie bevorzugt verwendet wird.

- Bei der Wahl gemeinsamer Zerlegungsfunktionen hat man 3 Wahlmöglichkeiten:
  - Alle möglichen gemeinsamen Zerlegungsfunktionen sind erlaubt.
  - Es werden nur gleichheitserhaltende gemeinsame Zerlegungsfunktionen gewählt.
  - Gleichheitserhaltende Zerlegungsfunktionen werden bevorzugt. Findet man keine gleichheitserhaltenden Zerlegungsfunktionen mehr, so wird noch nach anderen Zerlegungsfunktionen gesucht.
- Treten im Verlauf des Verfahrens z.B. bei Zusammensetzungsfunktionen don't care–Stellen auf, so kann man wählen zwischen
  - einer Behandlung dieser Funktionen als partielle Funktionen, d.h. einer Ausnutzung der dont' cares[8]
  - einer „zufälligen" Belegung der don't cares (z.B. kann die don't care–Menge der OFF–Menge zugeschlagen werden).
- Eventuell können auch Zerlegungen betrachtet werden, die nicht nichttrivial sind. Auch solche Zerlegungen können gegebenenfalls zu günstigen Realisierungen führen. Allerdings hat dann die Zusammensetzungsfunktion ebensoviele Eingänge wie die ursprüngliche Funktion und ihre Komplexität muß nicht geringer sein. Da nicht nichttriviale Zerlegungen daher recht schwer handhabbar sind, wird ihre Verwendung stark eingeschränkt: Nur bei Funktionen mit bis zu 3 Eingängen kann auch die Ausnutzung solcher Zerlegungen versucht werden, falls es dann erstens möglich ist, bei dieser Zerlegung schon vorhandene Zerlegungsfunktionen der nichttrivialen Zerlegung anderer Ausgangsfunktionen zu verwenden *und* die Komplexität der Zusammensetzungsfunktion geringer wird als die Komplexität der ursprünglichen Funktion.

Die im folgenden gezeigten Realisierungen wurden mit einer Festlegung auf zweiseitige Zerlegungen mit Bevorzugung gleichheitserhaltender Zerlegungen erzielt. Bei Funktionen mit einem Ausgang wurden „zufällige" gleichheitserhaltende Zerlegungsfunktionen verwendet, don't cares wurden ausgenutzt (im Sinne von Fußnote 8). Bei allen Beispielen bis auf eines wurde eine gleichmächtige Zerlegung gewählt.

Beim ersten Beispiel handelt es sich um die in Kapitel 2 schon betrachtete $exor$–Funktion mit 8 Eingängen.

Wie in Abbildung 4.7 ersichtlich, handelt es sich bei dem durch den Algorithmus berechneten Schaltkreis um einen balancierten Baum aus 7 $exor_2$–Gattern.

[8] Allerdings ist im bisher implementierten Verfahren eine Ausnutzung von don't cares nur bei Funktionen bis zu 3 Eingängen möglich.

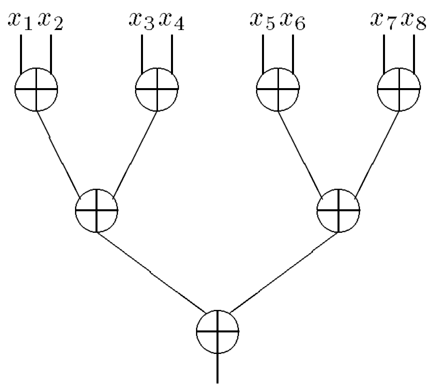


Abbildung 4.7: Realisierung zu $exor_8$

In der ersten Rekursionsstufe wurde eine Variablenaufteilung in die Mengen $\{x_1, \ldots, x_4\}$ und $\{x_5, \ldots, x_8\}$ berechnet. (Auch jede andere Variablenaufteilung wäre geeignet, da die Funktion totalsymmetrisch ist.) Sowohl bei der Zerlegungsfunktion auf $\{x_1, \ldots, x_4\}$ als auch bei der Zerlegungsfunktion auf $\{x_5, \ldots, x_8\}$ handelt es sich um die $exor_4$–Funktion, die Zusammensetzungsfunktion ist die $exor_2$–Funktion. Für die Zerlegungsfunktionen wird rekursiv wiederum eine Zerlegung durchgeführt.

Nach den Ausführungen in Kapitel 2 ist die gefundene Realisierung B_2–optimal.

Das nächste Beispiel ist eine Schwellenfunktion s_4^6, die folgendermaßen definiert ist:

$$s_4^6(x_0, \ldots, x_5) = 1 \iff \sum_{i=0}^{5} x_i \geq 4$$

Der resultierende Schaltkreis ist in Abbildung 4.8 angegeben.

Im ersten Schritt wurde eine Variablenaufteilung in die Mengen $\{x_1, x_2, x_3\}$ und $\{x_4, x_5, x_6\}$ gefunden. (Auch hier wäre jede andere Aufteilung ebenfalls geeignet, da die Funktion totalsymmetrisch ist.) Auf $\{x_1, x_2, x_3\}$ werden 2 Zerlegungsfunktionen c_r und s_r berechnet. Es handelt sich gerade um Übertrags– und Summenbit der binären Addition von x_1, x_2 und x_3. Ebenso sind die Zerlegungsfunktionen c_l und s_l auf den Variablen $\{x_4, x_5, x_6\}$ Übertrags– und Summenbit der binären Addition von x_4, x_5 und x_6. Betrachtet man die gefundene Realisierung für (c_r, s_r), so erkennt man, daß es sich um einen Volladdierer handelt. Der Volladdierer ergibt sich bei der Zerlegung von c_r und s_r hinsichtlich $\{\{x_1\}, \{x_2, x_3\}\}$. Hierbei wird $x_2 \oplus x_3$ bei der Zerlegung von c_r und s_r gemeinsam benutzt. Analog wird (c_l, s_l) durch einen Volladdierer realisiert.

Die Zusammensetzungsfunktion auf der 1. Rekursionsstufe hat 4 Eingänge c_l, s_l, c_r und s_r. Bei der rekursiven Behandlung dieser Funktion wird die Variablenaufteilung $\{\{c_l, c_r\}, \{s_l, s_r\}\}$ berechnet. Auf c_l und c_r werden zwei Zerlegungsfunktionen a_1 und a_2, auf s_l und s_r wird eine Zerlegungsfunktion b_1 berechnet.

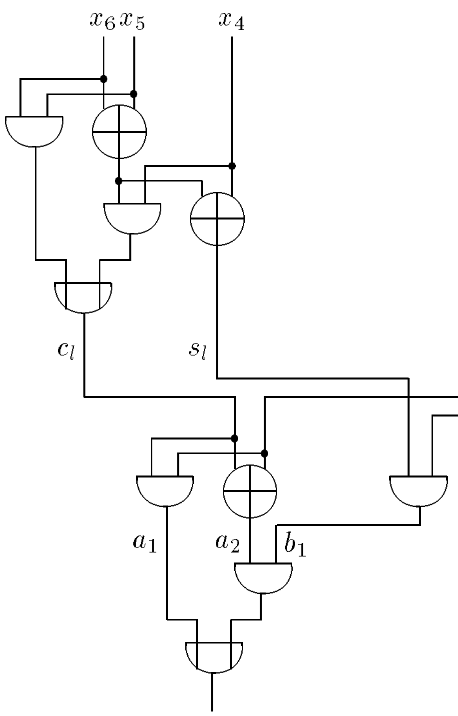
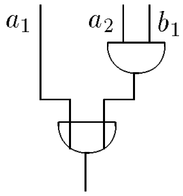


Abbildung 4.8: Realisierung zu s_4^6

Die zugehörige Zusammensetzungsfunktion mit 3 Eingängen a_1, a_2 und b_1 wird wiederum rekursiv zerlegt. An dieser Stelle zeigt sich die Bedeutung der Behandlung von don't cares: Es handelt sich um eine partielle Funktion, da a_1 und a_2 nie gleichzeitig den Wert 1 annehmen können. Würde man die don't care–Menge einfach der OFF–Menge zuschlagen, d.h. die Funktionswerte für $(1,1,0)$ und $(1,1,1)$ auf 0 festlegen, so wäre die resultierende Funktion nicht nichttrivial zerlegbar. Durch Betrachtung der 3 möglichen Zerlegungsmatrizen erkennt man nämlich, daß die Funktion nur dann nichttrivial zerlegbar ist, wenn der Funktionswert für $(1,1,0)$ auf 1 festgelegt wird. Der implementierte Algorithmus legt die Funktionswerte für $(1,1,0)$ und $(1,1,1)$ auf 1 fest und erreicht somit eine nichttriviale Zerlegung mit Aufteilung der Variablen in $\{a_1\}$ und $\{a_2, b_1\}$. Die Funktion wird dann durch folgende einfache Teilschaltung realisiert:



Wählt man als Kostenmaß die R_2–Komplexität, wobei $R_2 = B_2 \setminus \{exor, equiv\}$,

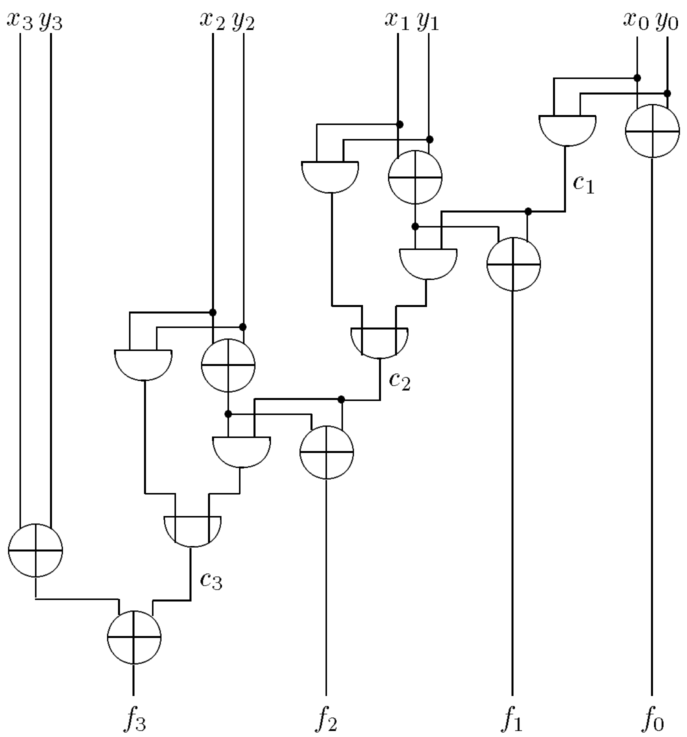


Abbildung 4.9: Realisierung zum 4–Bit–Addierer, Zerlegung nicht gleichmächtig

so ergeben sich als Kosten der berechneten Realisierung S für s_4^6

$$C_{R_2}(S) = 7C_{R_2}(and) + 3C_{R_2}(or) + 5C_{R_2}(exor) = 7 + 3 + 5 \cdot 3 = 25.$$

Realisiert man diese Schwellenfunktion zum Vergleich durch ihr eindeutiges Minimalpolynom, so erhält man

$$P = \bigvee_{0 \leq i_1 \leq \ldots \leq i_4 \leq 5} x_{i_1} \ldots x_{i_4}.$$

Das Minimalpolynom enthält $\binom{6}{4} = 15$ Monome der Länge 4. Die Kosten der Minimalpolynomrealisierung betragen also

$$C_{R_2}(P) = 15C_{R_2}(and_4) + C_{R_2}(or_{15}) = 15 \cdot 3 + 14 = 59.$$

Beim nächsten Beispiel, einem Addierer zweier 4–Bit–Zahlen, wurde nicht die gleichmächtige Zerlegung gewählt, sondern eine Variablenaufteilung, bei der die Mächtigkeit der einen Menge immer 2 beträgt und die andere Menge die restlichen Eingangsvariablen umfaßt. Dann ergibt sich die Realisierung aus Bild 4.9.

Man erkennt, daß es sich bei dem durch das Verfahren berechneten Schaltkreis um einen Carry–Ripple–Addierer handelt.

In der ersten Rekursionsstufe erfolgt eine Zerlegung hinsichtlich der Variablenaufteilung $\{\{x_3, y_3\}, \{x_2, y_2, x_1, y_1, x_0, y_0\}\}$. Da die Funktionen f_0, f_1 und f_2 von x_3 und y_3 unabhängig sind, wird für diese Funktionen auf $\{x_3, y_3\}$ *keine* Zerlegungsfunktion berechnet und die Zerlegungsfunktionen auf $\{x_2, y_2, x_1, y_1, x_0, y_0\}$ sind jeweils die Funktionen f_0, f_1 bzw. f_2 selbst. Bei der Zerlegung von f_3 ergibt sich 1 Zerlegungsfunktion auf $\{x_3, y_3\}$ und 1 Zerlegungsfunktion auf $\{x_2, y_2, x_1, y_1, x_0, y_0\}$, nämlich das Carrybit c_3 der Addition von $x_2x_1x_0$ und $y_2y_1y_0$.

Auf der nächsten Rekursionsstufe werden Zerlegungen zu f_0, f_1, f_2 und c_3 berechnet. Es ergibt sich eine Aufteilung der Variablen in $\{x_2, y_2\}$ und $\{x_1, y_1, x_0, y_0\}$. f_0 und f_1 sind wiederum unabhängig von x_2 y_2, so daß für f_0 und f_1 keine Zerlegungsfunktion auf $\{x_2, y_2\}$ berechnet wird. Die Zerlegungsfunktionen auf $\{x_1, y_1, x_0, y_0\}$ sind f_0 und f_1 selbst. Auf $\{x_2, y_2\}$ werden für c_3 2 Zerlegungsfunktionen, für f_2 1 Zerlegungsfunktion berechnet. Diese Zerlegungsfunktion für f_2 wird von c_3 und f_2 *gemeinsam* verwendet. Die auf $\{x_1, y_1, x_0, y_0\}$ berechnete Zerlegungsfunktion wird ebenfalls von c_3 und f_2 gemeinsam benutzt. Es handelt sich um das Carrybit c_2 der Addition von x_1x_0 und y_1y_0.

Analog werden auf der 3. Rekursionsstufe Zerlegungen für f_0, f_1 und c_2 berechnet, auf der 4. Rekursionsstufe werden Realisierungen für f_0 und c_1 bestimmt. Schließlich erhält man den Carry–Ripple–Addierer aus Abbildung 4.9.

Interessanterweise konnte Redkin (1981) zeigen, daß es keinen Addierer geben kann, der mit weniger Bausteinen auskommt als der Carry–Ripple–Addierer (vgl. [Red81]). Es wurde also ausgehend von der Funktionstabelle des Addierers mit einem automatischen Logiksyntheseverfahren ein Addierer mit minimaler Anzahl von Bausteinen gefunden. Allerdings ist die Laufzeit des Carry–Ripple–Addierers bekanntlich linear in der Anzahl der Eingänge des Addierers, also inakzeptabel hoch. Zur Lösung dieses Problems wird nun eine Realisierung betrachtet, die auf gleichmächtigen Zerlegungen beruht.

Läßt man den Algorithmus mit einem 8–Bit–Addierer als Eingabe und mit der Vorgabe, gleichmächtige Zerlegungen durchzuführen, ablaufen, so ergibt sich die Realisierung in Bild 4.10.

Betrachtet man die Realisierung genauer, so erkennt man, daß sie die gleiche Struktur wie der Conditional–Sum–Addierer hat. (Zum Vergleich ist im darauffolgenden Bild der Conditional–Sum–Addierer mit 16 Eingängen angegeben.)

Auf der ersten Rekursionsstufe wurde die Variablenaufteilung in die beiden Mengen

$$\{x_0, y_0, x_1, y_1, x_2, y_2, x_3, y_3\} \text{ und } \{x_4, y_4, x_5, y_5, x_6, y_6, x_7, y_7\}$$

berechnet. Bei dieser Zerlegung wird die auf $\{x_0, y_0, x_1, y_1, x_2, y_2, x_3, y_3\}$ berechnete Zerlegungsfunktion (das Carry–Bit der Addition von $x_3x_2x_1x_0$ und $y_3y_2y_1y_0$) für die Zerlegung von f_4, f_5, f_6 und f_7 *gemeinsam* benutzt.

Der Unterschied dieser Realisierung zum Conditional–Sum–Addierer liegt einerseits darin, daß grundsätzlich die Zahl der Zerlegungsfunktionen minimal

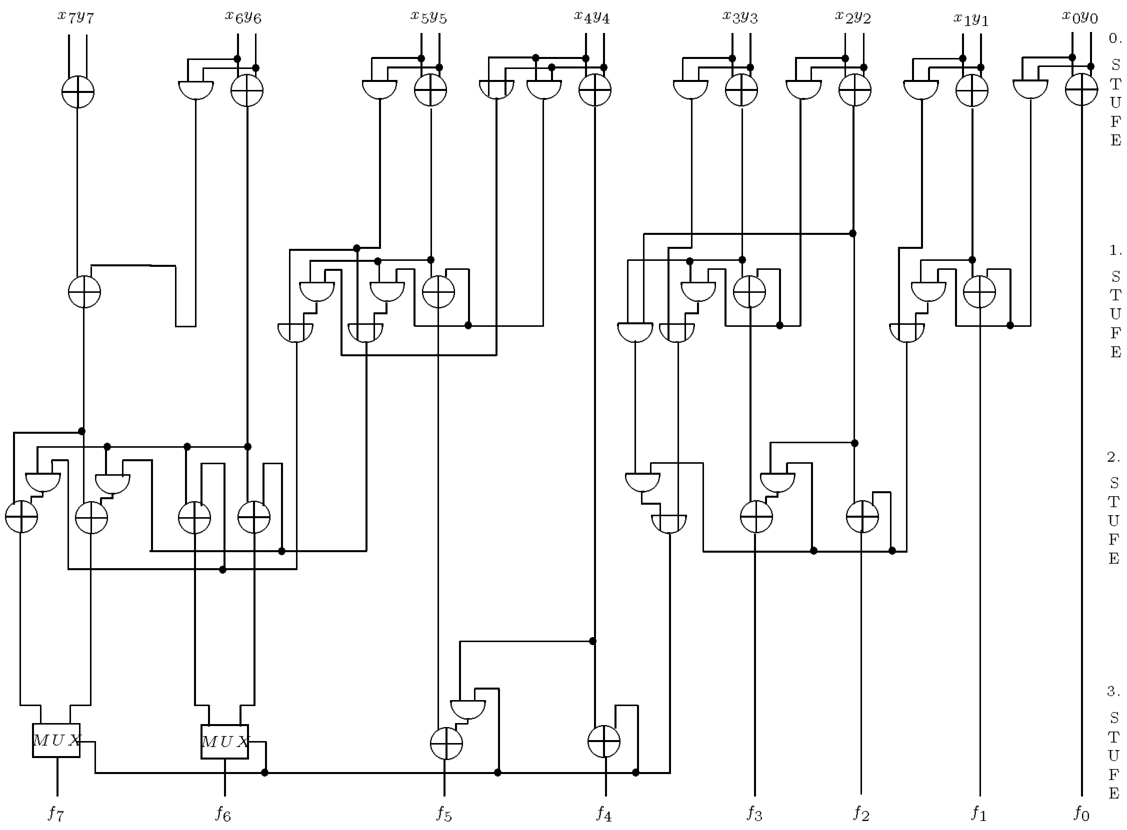


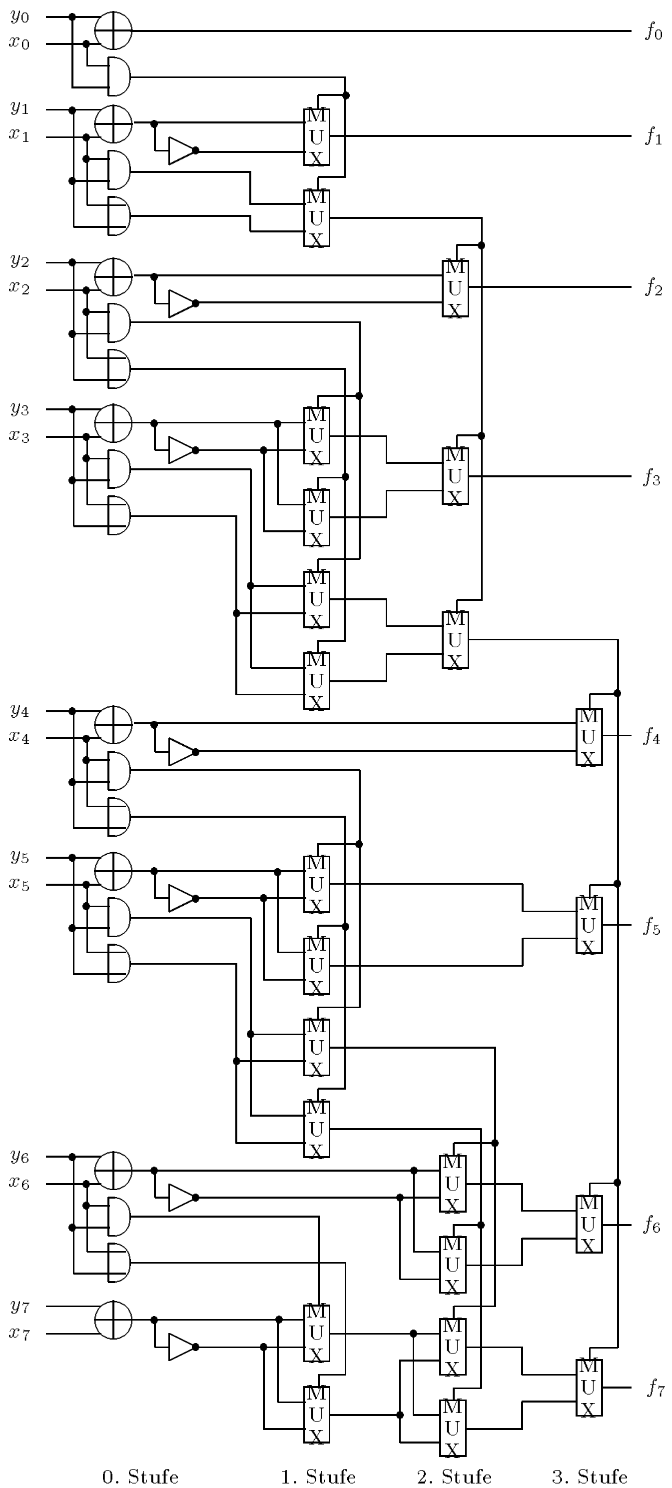
Abbildung 4.10: Realisierung zum 8–Bit–Addierer, Zerlegung gleichmächtig

gewählt wird und andererseits darin, daß nach Möglichkeit immer *gemeinsame* Zerlegungsfunktionen gewählt werden.

Dies führt dazu, daß bei der Zerlegung von f_4 auf der 1. Rekursionsstufe des Algorithmus (im Bild „3. Stufe") nur *eine* Zerlegungsfunktion auf $\{x_4, y_4, x_5, y_5, x_6, y_6, x_7, y_7\}$ berechnet wird, während der Conditional–Sum–Addierer an dieser Stelle 2 Funktionen benutzt, nämlich das letzte Bit der Summe von $x_7x_6x_5x_4$ und $y_7y_6y_5y_4$ und das letzte Bit der Summe von $x_7x_6x_5x_4$, $y_7y_6y_5y_4$ und 1.

Auf der Variablenmenge $\{x_4, y_4, x_5, y_5, x_6, y_6, x_7, y_7\}$ werden auf der 1. Rekursionsstufe für die Funktionen f_5, f_6 und f_7 sowohl bei der Realisierung aus Bild 4.10 als auch beim Conditional–Sum–Addierer 4 Informationen berechnet. Der Conditional–Sum–Addierer kodiert diese 4 Informationen jeweils fest durch die entsprechenden Bits der Summe und der Summe + 1. Bei Funktion f_5 wird in Bild 4.10 allerdings eine andere Kodierung dieser 4 Informationen gewählt. Der Grund für diese Wahl liegt darin, daß in diesem Fall die Zerlegungsfunktion von f_4 auf den Variablen $\{x_4, y_4, x_5, y_5, x_6, y_6, x_7, y_7\}$ von Funktion f_5 „mitverwendet" werden kann.

Diese Unterschiede bewirken, daß der durch den vorliegenden Algorithmus gefundene Addierer eine R_2–Komplexität von 89 hat, während der entsprechende Conditional–Sum–Addierer eine R_2–Komplexität von 106 hat. Die Tiefe ist in beiden Fällen die gleiche (bei einem parametrisierten Entwurf logarithmisch in der Anzahl der Eingänge).



y_0
x_0
y_1
x_1
y_2
x_2
y_3
x_3
y_4
x_4
y_5
x_5
y_6
x_6
y_7
x_7
f_0
f_1
f_2
f_3
f_4
f_5
f_6
f_7
MUX
0. Stufe
1. Stufe
2. Stufe
3. Stufe

Anhang A

Dynamisches Programm

Das folgende dynamische Programm bestimmt zu 2 gegebenen Funktionen eine gemeinsame, gleichheitserhaltende Zerlegungsfunktion. Die Notation orientiert sich an der im Zusammenhang mit dem modifizierten branch–and–bound–Algorithmus aus Kapitel 4 benutzten.

Eingabe:

- Funktionen f_1, f_2 aus B_n mit den Eingangsvariablen $x_1, \ldots, x_p$, $y_1, \ldots, y_q$.
- Zerlegungsmatrizen Z_1 von f_1 und Z_2 von f_2 hinsichtlich $X = \{x_1, \ldots, x_p\}$ und $r_1 = \lceil \log(vz(X, f_1)) \rceil > 0$, $r_2 = \lceil \log(vz(X, f_2)) \rceil > 0$.
- Für $i \in \{1, 2\}$: Sei $\{K_1^{(i)}, \ldots, K_{vz(X,f_i)}^{(i)}\}$ die durch Zeilengleichheit in Z_i auf $\{0,1\}^p$ induzierte Äquivalenzklasseneinteilung.
- Eine Äquivalenzklasseneinteilung $P = \{E_1, \ldots, E_k\}$ auf $\{0,1\}^p$ („Zusammenhangskomponenten").
  Gemeinsame, gleichheitserhaltende Zerlegungsfunktionen müssen allen $x \in E_j$ $(1 \leq j \leq k)$ den gleichen Funktionswert zuordnen.
- Für $i \in \{1, 2\}$ und alle $E_j \in P$:
  $ZNR_j^{(i)} = \{l \mid K_l^{(i)} \cap E_j \neq \emptyset\}$
  $MZNR_j^{(i)} = |ZNR_j^{(i)}|$

Ausgabe:

- $\alpha_1 \in B_p$, so daß es einseitige Zerlegungen von f_1 und f_2 hinsichtlich X gibt der Form

$$\begin{aligned}
&f_1(x_1, \ldots, x_p, y_1, \ldots, y_q) = \\
&\qquad g^{(1)}(\alpha_1(\mathbf{x}), \alpha_2^{(1)}(\mathbf{x}), \ldots, \alpha_{r_1}^{(1)}(\mathbf{x}), y_1, \ldots, y_q) \\
&f_2(x_1, \ldots, x_p, y_1, \ldots, y_q) = \\
&\qquad g^{(2)}(\alpha_1(\mathbf{x}), \alpha_2^{(2)}(\mathbf{x}), \ldots, \alpha_{r_m}^{(m)}(\mathbf{x}), y_1, \ldots, y_q),
\end{aligned}$$

  wobei α_1 gleichheitserhaltend ist. (Falls es überhaupt eine Funktion aus B_p mit dieser Eigenschaft gibt.)

- Sonst: Eine Meldung daß es keine Funktion aus B_p mit der angegebenen Eigenschaft gibt.

Algorithmus:

1. Sei $B[l_1][l_2] = FALSE$, $D[l_1][l_2] = \emptyset$
   $\forall 0 \leq l_1 \leq 2^{r_1 - 1}$, $0 \leq l_2 \leq 2^{r_2 - 1}$.
2. B[0][0] = TRUE
3. **for** $j = 1$ **to** k **do**
   for $l_1 = 2^{r_1-1}$ **downto** $MZNR_j^{(1)}$ **do**
   for $l_2 = 2^{r_2-1}$ **downto** $MZNR_j^{(2)}$ **do**
   if $B[l_1 - MZNR_j^{(1)}][l_2 - MZNR_j^{(2)}] = TRUE$ **then**
   $B[l_1][l_2] = TRUE$
   $D[l_1][l_2] = D[l_1][l_2] \cup j$
   (* Invariante: Es gibt genau dann eine Teilmenge D von $\{1, \ldots, j\}$ mit
   $\sum_{l \in D} MZNR_l^{(1)} = l_1$ und $\sum_{l \in D} MZNR_l^{(2)} = l_2$,
   wenn $B[l_1][l_2] = TRUE$. Dann ist $D = D[l_1][l_2]$ eine mögliche Wahl für D. *)

   fi
   od
   od
   od

4. Wenn es l_1, l_2 gibt mit
   $vz(X, f_1) - 2^{r_1-1} \leq l_1 \leq 2^{r_1-1}$ und $vz(X, f_2) - 2^{r_2-1} \leq l_2 \leq 2^{r_2-1}$
   und $B[l_1][l_2] = TRUE$, dann:
   Setze für alle $j \in D[l_1][l_2]$ und $x \in E_j$
   $\alpha_1(x) = 0$
   und für die restlichen $x \in \{0,1\}^p$
   $\alpha_1(x) = 1$

   Sonst:
   Gebe aus, daß es kein solches α_1 gibt.

Die Korrektheit des Algorithmus ergibt sich aus folgenden Überlegungen:

Mit Lemma 4.10 erkennt man, daß es genau dann eine Zerlegungsfunktion α_1 mit den angegebenen Eigenschaften gibt, wenn es eine Aufteilung von $\{1, \ldots, k\}$ in 2 Mengen

$$D_0 = \{j \mid \alpha_1(x) = 0 \ \forall x \in E_j\} \text{ und } D_1 = \{j \mid \alpha_1(x) = 1 \ \forall x \in E_j\}$$

gibt mit

$$\sum_{l \in D_0} MZNR_l^{(1)} \leq 2^{r_1-1} \text{ und } \sum_{l \in D_1} MZNR_l^{(1)} \leq 2^{r_1-1} \text{ sowie}$$

$$\sum_{l \in D_0} MZNR_l^{(2)} \leq 2^{r_2-1} \text{ und } \sum_{l \in D_1} MZNR_l^{(2)} \leq 2^{r_2-1}.$$

Da

$$\sum_{l=1}^{k} MZNR_l^{(1)} = vz(X, f_1),$$

ist

$$\sum_{l \in D_1} MZNR_l^{(1)} = \sum_{l \in \{1,\ldots k\} \setminus D_0} MZNR_l^{(1)} \leq 2^{r_1-1}$$

äquivalent zu der Aussage

$$\sum_{l \in D_0} MZNR_l^{(1)} \geq vz(x, f_1) - 2^{r_1-1}.$$

Folglich gibt es genau dann eine solche Zerlegungsfunktion α_1, wenn es eine Menge

$$D_0 = \{j \mid \alpha_1(x) = 0 \; \forall x \in E_j\}$$

gibt mit

$$vz(x, f_1) - 2^{r_1-1} \leq \sum_{l \in D_0} MZNR_l^{(1)} \leq 2^{r_1-1} \text{ und}$$

$$vz(x, f_2) - 2^{r_2-1} \leq \sum_{l \in D_0} MZNR_l^{(2)} \leq 2^{r_2-1}.$$

Die Korrektheit des Algorithmus ergibt sich aus dieser Aussage und der Gültigkeit der Invariante aus Schritt 3.

Literaturverzeichnis

[AH63] R. F. Arnold and M. A. Harrison. Algebraic Properties of Symmetric and Partially Symmetric Boolean Functions. IEEE Transact. on Electronic Computers, vol. EC–12, no. 3, pp. 244–251, June 1963.

[Ash59] R. L. Ashenhurst. The Decomposition of Switching Functions. Proceedings of an International Symposium on the Theory of Switching, vol. 28 of Comp. Lab. of Harvard University, pp. 74–116, 1959.

[BHMSV84] R. K. Brayton, G. D. Hachtel, C. T. McMullen and A. Sangiovanni–Vincentelli. Logic Minimization Algorithms for VLSI Synthesis. The Kluwer International Series in Engineering and Computer Science. Kluwer Academic Publishers, 1984.

[Bry86] R. E. Bryant. Graph–Based Algorithms for Boolean Function Manipulation. IEEE Transact. on Computers, 35(8), pp. 677–691, 1986.

[Cur61] H. A. Curtis. A generalized tree circuit. J. Assoc. Comput. Mach. , vol. 8, pp. 484-496, 1961.

[GJ79] M. R. Garey and D. S. Johnson. Computers and Intractability: A Guide to the Theory of **NP**–completeness. New York: Freeman, 1979.

[Hot60] G. Hotz. Zur Reduktionstheorie der booleschen Algebra. Colloquium über Schaltkreis– und Schaltwerk–Theorie, (1960), Herausgeber: Unger, H., Peschel, E., Birkhäuser Verlag, 1961.

[Hot74] G. Hotz. Schaltkreistheorie. Walter de Gruyter, 1974.

[HOI89] T. Hwang, R. M. Owens and M. J. Irwin. Exploiting Communication Complexity for Multilevel Logic Synthesis. IEEE Trans. Computer–Aided Design, vol. 9, pp. 1017–1027, Oct. 1990.

[HOI92] T. Hwang, R. M. Owens and M. J. Irwin. Efficiently Computing Communication Complexity for Multilevel Logic Synthesis. IEEE Trans. Computer–Aided Design, vol. 11, pp. 545–554, May. 1992.

[Kar63] R. M. Karp. Functional Decomposition and Switching Circuit Design. J. Soc. Indust. Appl. Math. , vol. 11, no. 2, pp. 291–335, June 1963.

[KD91] B.–G. Kim and D. L. Dietmeyer. Multilevel Logic Synthesis of Symmetric Switching Functions. IEEE Transact. on CAD, vol. 10, no. 4, pp. 436–446, April 1991.

[Lup58] O. B. Lupanov. A Method of Circuit Synthesis. Izv. VUZ Radioviz 1, pp. 120–140, 1958.

[Meh84b] K. Mehlhorn. Data Structures and Algorithms 2: Graph Algorithms and NP–Completeness. EATCS Monographs on Theoretical Computer Science, Springer–Verlag, Berlin, Heidelberg, New York, Tokio, 1984.

[Möl92] D. Möller. Logiksynthese symmetrischer Schaltfunktionen. Master's thesis, Fachbereich Informatik, Humbold–Universität Berlin.

[Mol90] P. Molitor. Vorlesungsskript Logikminimierung 90/91.

[LP81] H. R. Lewis, C. H. Papademitriou. Elements of the Theory of Computation. Prentice Hall, 1981.

[Pau76] W. J. Paul. Realizing Boolean Functions on Disjoint Sets of Variables. TCS 2, pp. 383–396, 1976.

[Red81] N. P. Redkin. Minimal realization of a binary adder. Probl. Kibern. 38, pp. 181–216, 1981.

[RK62] J. P. Roth and R. M. Karp. Minimization over Boolean Graphs. IBM Journal, vol. 6, no. 2, pp. 227–238, 1962.

[Sha49] C. E. Shannon. The Synthesis of Two–Terminal Switching Circuits. Bell Systems Technical Journal 28, pp. 59–98, 1949.

[Weg87] I. Wegener. The Complexity of Boolean Functions. Wiley–Teubner, 1987.

[Weg89] I. Wegener. Effiziente Algorithmen für grundlegende Funktionen. Teubner Verlag, 1989.