# Ausnutzung von Symmetrien partieller Boolescher Funktionen

Diplomarbeit
vorgelegt von
**Stephan Melchior**

Angefertigt im Fachbereich Informatik der Universität des Saarlandes
nach einem Thema von Prof. Dr. Dr. hc. mult. Günter Hotz

Hiermit versichere ich an Eides statt, daß ich die vorliegende Arbeit selbst verfaßt und keine anderen als die angegebenen Hilfsmittel verwendet habe.

Wallerfangen, Dezember 1997

# Inhaltsverzeichnis

# Einleitung

Boolesche Funktionen, d.h. Funktionen der Form $f : \{0,1\}^n \rightarrow \{0,1\}^m$ finden in vielen Gebieten der Informatik Verwendung. Insbesondere eignen sich Boolesche Funktionen sehr gut dazu, das logische Eingabe-/Ausgabeverhalten kombinatorischer Schaltungen festzulegen.
Der Entwurf kombinatorischer Schaltungen zu gegebenen Booleschen Funktionen ist Aufgabe der Logiksynthese. Dabei sollen die resultierenden Entwürfe möglichst geringe Kosten aufweisen. Als Kostenmaße kommen hierbei z.B. die benötigte Schaltkreisfläche bzw. die Laufzeit der Schaltung in Frage.

Zwar sind im allgemeinen $\theta(\frac{2^n}{n})$ 2-Input-Gatter notwendig, um eine Funktion mit $n$ Eingangsvariablen zu realisieren [Shan49], es zeigt sich aber, daß bei praktischen Problemen meist kleinere Entwürfe möglich sind. Der Grund hierfür liegt darin, daß die in der Praxis zu realisierenden Booleschen Funktionen in der Regel *nicht* durch (für große Eingangsvariablenzahlen riesige) Funktionstabellen mit zufällig bestimmten Belegungen gegeben sind, sondern relativ kurzen und einfachen Bildungsgesetzen genügen (z.B. Addition zweier binärer Zahlen) und daher die Funktionen gewisse Regelmäßigkeiten bzw. Struktureigenschaften aufweisen.
Um zu guten Resultaten zu gelangen, ist es deshalb unbedingt erforderlich, daß solche Struktureigenschaften für die zu realisierende Boolesche Funktion beim Schaltungsentwurf erkannt und ausgenutzt werden.

Eine wichtige Struktureigenschaft Boolescher Funktionen ist die *Symmetrieeigenschaft*. Darunter versteht man z.B. die Invarianz einer Booleschen Funktion gegenüber Permutationen bzw. zusätzlichen Invertierungen bestimmter Eingangsvariablen. Es ist bekannt, daß Funktionen, die invariant gegenüber allen Vertauschungen ihrer $n$ Eingangsvariablen sind (*totalsymmetrische* Funktionen), im worst case mit höchstens linear vielen 2-Input-Gattern realisiert werden können [Weg89].

Bei praktischen Problemen treten häufig *partielle* Boolesche Funktionen auf, d.h. Funktionen, deren Definitionsbereich auf eine Teilmenge $D \subseteq \{0,1\}^n$ beschränkt ist. Die don't care-Menge (d.h. die Menge der unspezifizierten Funktionsstellen) ist bei diesen Funktionen also nicht leer. Solche Funktionen können z.B. daraus resultieren, daß Schaltungen, die diese partiellen Funktionen realisieren sollen, in ein größeres System eingebettet werden und somit gewisse Eingabevektoren für diese Schaltung von vornherein ausgeschlossen werden können.

Durch die Belegung der don't care-Werte ergeben sich verschiedene *Erweiterungen* der partiellen Funktion, die sich hinsichtlich ihrer Komplexität enorm unterscheiden können. Aus diesem Grund ist die Bestimmung einer günstigen Erweiterung besonders wichtig beim Entwurf von Schaltungen für partielle Funktionen.

Tools zur rechnerunterstützten Logiksynthese müssen Boolesche Funktionen im Rechner repräsentieren. Eine in den letzten Jahren sehr häufig verwendete Darstellungsmöglichkeit sind ROBDDs (**r**educed **o**rdered **b**inary **d**ecision **d**iagrams) [Bry86]. Sie zeichnen sich u.a. dadurch aus, daß für viele in der Praxis auftretenden Funktionen kompakte ROBDD-Darstellungen existieren. ROBDDs werden auch beispielsweise bei der Verifikation sequentieller Schaltkreise [BCM+90, CMB89, MWBS88] oder der Generierung von Testmustern [Beck92] erfolgreich eingesetzt. Die Laufzeit und der Speicherverbrauch der entsprechenden Algorithmen wird dabei wesentlich von der Größe der verwendeten ROBDD-Darstellungen bestimmt. Treten in solchen Anwendungen partiellen Funktionen auf, so ist es daher wichtig, daß Erweiterungen verwendet werden, die sich durch möglichst kleine ROBDDs darstellen lassen. Auch hier ergeben sich durch Ausnutzen von Struktureigenschaften (z.B. Symmetrie) meist speicherplatzsparende Darstellungen.

In der vorliegenden Arbeit wird bei der Bestimmung einer Erweiterung für eine partielle Boolesche Funktion deren Symmetrieeigenschaften ausgenutzt, indem nach einer Erweiterung gesucht wird, die in einer möglichst kleinen Partition $\Pi = \{\lambda_1, \ldots \lambda_k\}$ der Eingangsvariablen symmetrisch ist (d.h. die Funktion ist jeweils invariant gegenüber Vertauschungen der Art $(x_i \leftrightarrow x_j)$ mit $x_i, x_j \in \lambda_l$ und $\lambda_l \in \Pi$).
Es zeigt sich, daß dieses Problem bei partiellen Funktionen im Vergleich zu totalen Funktionen wesentlich schwieriger ist.

Die vorliegende Arbeit gliedert sich in folgende Teile:
Im ersten Kapitel werden grundlegende Definitionen und Begriffe angegeben, die für die weitere Arbeit benötigt werden. Dabei werden verschiedene Beschreibungsmöglichkeiten Boolescher Funktionen vorgestellt und deren spezielle Eigenschaften näher untersucht.
Das zweite Kapitel befaßt sich eingehend mit Symmetrien Boolescher Funktionen. Dabei werden anhand von Beispielen die wesentlichen Unterschiede zwischen Symmetrien partieller Funktionen und den Symmetrien totaler Funktionen sowie die daraus resultierenden Probleme aufgezeigt.
Es wird zudem demonstriert, wie Symmetrien vorteilhaft bei der Suche nach kleinen Realisierungen und Darstellungen, speziell für Beschreibungen mittels ROBDDs, genutzt werden können.
Das dritte Kapitel beschäftigt sich ausführlich mit der Bestimmung von speziellen Erweiterungen, die vorgegebene Symmetrieeigenschaften erfüllen.
Im letzten Kapitel wird eine Heuristik angegeben, die zu einer partiellen Booleschen Funktion eine Belegung der don't cares liefert, so daß die resultierende Erweiterung symmetrisch in einer möglichst kleinen Partition der Eingangsva-

riablen ist.
Abschließend werden Ergebnisse der Heuristik zu einigen Benchmarkschaltungen angegeben.

Mein besonderer Dank gilt Herrn Prof. Dr. Günter Hotz für die Vergabe dieses Themas sowie Herrn Dr. Christoph Scholl für die Betreuung der Arbeit.
Allen Mitarbeitern des Lehrstuhls und meinen Kommilitonen sei für das gute Arbeitsklima gedankt.
Ein großer Dank gilt meinen Eltern, die mir mit ihrer Unterstützung dieses Studium überhaupt ermöglicht haben.

# Kapitel 1

# Grundlagen

Boolesche Funktionen bilden den Hauptgegenstand der vorliegenden Arbeit. Deshalb sollen zunächst in diesem Kapitel die Begriffe und Bezeichnungen, die im Zusammenhang mit Booleschen Funktionen und deren Beschreibungsmöglichkeiten verwendet werden, formal eingeführt werden.

## 1.1 Boolesche Funktionen

Man unterscheidet zwischen totalen und partiellen Booleschen Funktionen:

**Definition 1.1 (Totale Boolesche Funktionen)**

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

*ist 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)**

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

*ist die Menge aller* **partiellen Booleschen Funktionen** *bzw. Schaltfunktionen von* $D$ *nach* $\{0,1\}^m$*.*
$D$ *heißt* **Definitionsbereich** *von* $f : D \rightarrow \{0,1\}^m$ *und wird mit* $D(f)$ *bezeichnet.*

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

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

*die Menge aller* **partiellen Booleschen Funktionen** *bzw. Schaltfunktionen von* $D$ *nach* $\{0,1\}$*.*
*(2) Für* $f \in S(D)$ *und* $x \notin D$ *schreibt man auch* $f(x) = *$*.*

*Der Funktionswert $*$ wird auch als* don't care- *Wert bezeichnet.*
*(3)*

$$BP_n := \bigcup_{D \subseteq \{0,1\}^n} S(D)$$

### 1.1.1 Boolesche Operationen

Um aus vorgegebenen Booleschen Funktionen neue Funktionen zu definieren, werden nun verschiedene Boolesche Operationen zur Verfügung gestellt.
Diese Operationen basieren auf Funktionen aus $B_1$ und $B_2$. Jeder Operation $op : S(D) \times S(D) \rightarrow S(D)$ bzw. $op : S(D) \rightarrow S(D)$ ist eindeutig eine Funktion $f_{op} \in B_2$ bzw. $f_{op} \in B_1$ zugeordnet. Im allgemeinen verwendet man die Infixschreibweise, um eine Boolesche Operation anzugeben, d.h. sind $g, h \in S(D)$, so schreibt man statt $op(g,h)$ stets $g\, op\, h$ und meint damit:

$$\forall x \in D : (g\, op\, h)(x) = f_{op}(g(x), h(x))$$

bzw.

$$\forall x \in D : (op\, g)(x) = f_{op}(g(x)).$$

Es werden nun die wichtigsten Booleschen Funktionen aus $B_1 \cup B_2$ mit ihren üblichen Bezeichnungen angegeben. Den daraus resultierenden Booleschen Operationen ordnet man die gleiche Bezeichnungen zu.

**Bezeichnung 3**

- *(Einsfunktion* $\mathbf{1}$*)*
  $\mathbf{1}(x) = 1 \quad \forall x \in \{0,1\}$
- *(Nullfunktion* $\mathbf{0}$*)*
  $\mathbf{0}(x) = 0 \quad \forall x \in \{0,1\}$
- *(Identität $id(x)$)*
  $id(x) = 1 \iff x = 1$
- *(Disjunktion $x \vee y$)*
  $x \vee y := or(x,y) = 1 \iff x = 1$ *oder* $y = 1$
- *(Konjunktion $x \wedge y$)*
  $x \wedge y$ *(bzw. $xy$)* $:= and(x,y) = 1 \iff x = y = 1$
- *(Negation $\neg x$ bzw. $\overline{x}$)*
  $\neg x$ *(bzw. $\overline{x}$)* $:= not(x) = 1 \iff x = 0$
- *(Exor-Funktion $x \oplus y$ )*
  $x \oplus y := exor(x,y) = 1 \iff x \neq y$
- *(Äquivalenz $x \equiv y$ )*
  $x \equiv y := equiv(x,y) = 1 \iff x = y$

Eine weitere (ternäre) Boolesche Operation, die in einem späteren Abschnitt eine wichtige Rolle spielen wird, insbesondere bei der Darstellung Boolescher Funktionen, basiert auf der folgenden Booleschen Funktion:

- *ite ( if-then-else)* $\in B_3$
  $ite(x, y, s) := (\overline{s} \wedge x) \vee (s \wedge y)$

Eine besondere Boolesche Funktion ist die Projektion eines $n-$stelligen binären Vektors auf eine seiner Komponenten:

- $\pi_i^n \in B_n$ mit $\pi_i^n(x_1, \ldots, x_n) = x_i$.

**Bemerkung 1** *Geht aus dem Zusammenhang eindeutig hervor, um welche Projektion es sich handelt, dann schreibt man auch einfach $x_i$ statt $\pi_i^n$.*
*Die Boolesche Funktion $x_i$ wird auch mit $x_i^1$, die Funktion $\overline{x_i}$ mit $x_i^0$ bezeichnet.*

Es ist zudem möglich, Funktionen aus $B_n$ in die Menge $B_k$ $(k \geq n)$ mittels der Abbildung $i_{n,k}$ einzubetten. Dazu definiert man $i_{n,k}(f) = f'$ wie folgt:

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

Im folgenden soll eine Funktion $f \in B_n$ stets mit *allen* Funktionen $i_{n,k}(f)$ identifiziert werden, d.h. falls notwendig, ist implizit die entsprechende Abbildung $i_{n,k}$ zu verwenden. Dies hat zur Folge, daß z.B. $\mathbf{1} \in B_n$ für alle $n \geq 1$.

### 1.1.2 Beschreibung partieller Funktionen durch totale Funktionen

Man kann zu einer partiellen Funktion $f \in S(D)$ drei *totale* Boolesche Funktionen angeben, die $f$ vollständig beschreiben:

**Definition 1.3 ($f_{on}, f_{off}, f_{dc}$ )** *Falls $f \in S(D)$ mit $D \subseteq \{0,1\}^n$, so sind die (totalen) Booleschen Funktionen $f_{on}, f_{off}, f_{dc} \in B_n$ wie folgt definiert:*

- $$f_{on}(x) := \begin{cases} 1 & \text{falls } x \in D \text{ und } f(x) = 1 \\ 0 & \text{sonst} \end{cases}$$

- $$f_{off}(x) := \begin{cases} 1 & \text{falls } x \in D \text{ und } f(x) = 0 \\ 0 & \text{sonst} \end{cases}$$

- $$f_{dc}(x) := \begin{cases} 1 & \text{falls } x \notin D \\ 0 & \text{sonst} \end{cases}$$

Alternativ zu diesen Funktionen können auch spezielle Teilmengen von $\{0,1\}^n$ definiert werden:

**Definition 1.4 (ON-Menge, OFF-Menge, DC-Menge)**
*Sei $f \in S(D)$, $D \subseteq \{0,1\}^n$. Dann ist*

- $DC(f) := \{0,1\}^n \setminus D$ *die* **don't care-Menge** *(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 $f$.*

Man erhält sofort:

**Korollar 1.1** *Ist $f \in S(D)$, $D \subseteq \{0,1\}^n$, dann gilt:*

(1) $\forall x \in \{0,1\}^n : f_{on}(x) + f_{off}(x) + f_{dc}(x) = 1^\dagger$.
(2) *$ON(f), OFF(f), DC(f)$ bilden eine Partition von $\{0,1\}^n$.*

Damit reicht es aus, zur vollständigen Beschreibung einer partiellen Booleschen Funktion $f$ nur *zwei* der Funktionen $f_{on}, f_{off}, f_{dc}$ anzugeben, da die verbleibende Funktion eindeutig durch die beiden anderen bestimmt ist.

## 1.2 Graphen

In der vorliegenden Arbeit werden an vielen Stellen Graphen verwendet, so können z.B. Boolesche Funktionen durch spezielle Graphen beschrieben werden, aber auch Schaltkreise lassen sich durch Graphen formalisieren.
Es sollen daher nun die Begriffe aus der Graphentheorie eingeführt werden, welche in der weiteren Arbeit benötigt werden.

**Definition 1.5 (Ungerichteter Graph)**
*Ein* **ungerichteter Graph** *$G$ ist ein Paar $G = (V, E)$ mit:*

- *$V$ ist eine endliche Menge von Knoten und*
- *$E \subseteq \{M \subseteq V \mid \; |M| = 2\}$ ist die Menge der Kanten.*

*Zwei Knoten $v, w$ heißen* **adjazent** *genau dann, falls $\{v, w\} \in E$.*
*Die Abbildung $degree : V \to \mathbf{N}, degree(v) = |\{w \in V \mid w \text{ ist adjazent zu } v\}|$ ordnet jedem Knoten $v$ einen* **Grad** *$degree(v)$ zu.*
*Eine Knotenmenge $V' \subseteq V$ heißt* **Clique in** *$G$ genau dann, falls alle Knoten in $V'$ adjazent zueinander sind.*

Jeder ungerichtete Graph besitzt auch einen inversen Graphen:

† „+“ bezeichnet die ganzzahlige Addition.

**Definition 1.6 (Inverser Graph)**
*Sei $G = (V, E)$ ein ungerichteter Graph. $\overline{G} = (V, E')$ heißt* **inverser Graph zu** *$G$ genau dann, falls*

$$\forall e = \{v, w\}\ v, w \in V : \ e \in E \iff e \notin E'.$$

Ordnet man den Kanten eine Orientierung zu, so entstehen gerichtete Graphen:

**Definition 1.7 (Gerichteter Graph)**
*Ein* **gerichteter Graph** *$G$ ist ein Paar $G = (V, E)$ mit:*

- *$V$ ist eine endliche Menge von Knoten und*
- *$E \subseteq V \times V$ ist die Menge der Kanten.*

*Ein Knoten $v$ ist* **inzident** *zum Knoten $w$ genau dann, falls $(v, w) \in E$.*
*Ein Knoten $v$ ist* **adjazent** *zum Knoten $w$ genau dann, falls $(w, v) \in E$.*
*Die Abbildungen $Quelle : E \to V$ und $Ziel : E \to V$ ordnen jeder Kante $e \in E$ einen Quellknoten $Quelle(e)$ bzw. Zielknoten $Ziel(e)$ zu.*
*Falls $e = (v, w) \in E$, so ist $Quelle(e) = v$ und $Ziel(e) = w$.*
*Jedem Knoten wird ein Eingangsgrad indeg und ein Ausgangsgrad outdeg zugeordnet:*

$$indeg : V \to \mathbf{N}, indeg(v) = | \{w \in V \mid (w, v) \in E\} | \quad und$$

$$outdeg : V \to \mathbf{N}, outdeg(v) = | \{w \in V \mid (v, w) \in E\} | .$$

*Ein Knoten $v$ mit $indeg(v) = 0$ heißt* **Wurzel** *von $G$.*
*Ein gerichteter Graph $G = (V, E)$ mit $outdeg(v) \in \{0, 2\}$ für alle Knoten $v \in V$ heißt* **binärer Graph**.
*Sollen zusätzlich die zu einem Knoten $v$ inzidenten bzw. adjazenten Knoten eindeutig identifiziert werden, so definiert man zwei partielle injektive Abbildungen $I, O : V \times \mathbf{N} \rightsquigarrow E$.*
*$I(v, i) = e$ mit $Ziel(e) = v$ ist genau dann definiert, falls $1 \le i \le indeg(v)$.*
*$O(v, i) = e$ mit $Quelle(e) = v$ ist genau dann definiert, falls $1 \le i \le outdeg(v)$.*
*Ein gerichteter Graph, der zusätzlich über die Abbildungen $I, O$ verfügt, heißt* **knotenorientiert**.

Verschiedene Kanten des Graphen können zu (gerichteten) Pfaden zusammengefaßt werden:

**Definition 1.8 (Pfad, Zykel, azyklischer Graph)**
*Sei $G = (V, E)$ ein gerichteter Graph. Eine Folge $p = (v_1, \ldots, v_n)$ von Knoten aus $G$ heißt* **Pfad** *(der Länge $n - 1$) in $G$ von $v_1$ nach $v_n$ genau dann, wenn gilt:*

$$v_{i+1} \text{ ist adjazent zu } v_i \quad (1 \le i \le n - 1).$$

*Gilt $v_1 = v_n$, so ist $p$ ein* **Zykel in** *$G$.*
*Ein Graph, der keinen Zykel enthält, heißt* **azyklisch**.

**Bemerkung 2** *Jeder azyklische Graph besitzt mindestens eine Wurzel.*

## 1.3 $\Omega-$Schaltkreise

In der Logiksynthese ergibt sich das Problem, Boolesche Funktionen in Schaltkreise zu überführen, welche diese Funktionen berechnen. Dabei stellt sich oft zusätzlich die Aufgabe, zu einer gegebenen Booleschen Funktion einen möglichst minimalen Schaltkreis zu entwerfen. Diese Anforderung kann sich z.B. auf die Größe oder die Laufzeit der Schaltung beziehen.
In diesem Abschnitt sollen deshalb Schaltkreise formal eingeführt werden und ein Zusammenhang mit den Booleschen Funktionen hergestellt werden.

Jedem Schaltkreis liegt eine Zellenbibliothek zugrunde, in der die Grundgatter abgespeichert sind, die für die Schaltung zur Verfügung stehen. Diese Grundgatter berechnen meist einfache Boolesche Funktionen bzw. realisieren Boolesche Operationen.

**Definition 1.9 (Zellenbibliothek, Grundgatter)**
*Eine endliche Teilmenge $\Omega \subseteq \bigcup_{i \geq 0} B_i$ heißt* **Zellenbibliothek**. *Die einzelnen Elemente in $\Omega$ heißen* **Grundgatter**.

Es eignen sich nicht alle Teilmengen aus $\bigcup_{i \geq 0} B_i$ als Zellenbibliothek, sondern nur solche, mit deren Grundgattern man wirklich auch *alle* Booleschen Funktionen in Form eines Schaltkreises realisieren kann. Derartige Zellenbibliotheken heißen **vollständig**.

**Definition 1.10 ( $\Omega$-Schaltkreis)** *Sei $\Omega$ eine Zellenbibliothek und $I/O = \{EPAD, APAD\}$. Ein* **$\Omega$-Schaltkreis** *mit $n$ Eingängen und $m$ Ausgängen ist ein 4-Tupel*

$$(G, typ, pe, pa)$$

*mit:*

- *G=(V,E) ist ein gerichteter, azyklischer, knotenorientierter Graph.*
  *Ein Knoten wird auch als Gatter bezeichnet, eine Kante auch als (Verbindungs)leitung. Geben die Abbildungen $I, O$ die Knotenorientierung des Graphen an, so gilt:*
  *Ist $I(v,i) = e$ mit $Ziel(e) = v$, so heißt $v$ der i. Input des Knotens $v$.*
  *Ist $O(v,i) = e$ mit $Quelle(e) = v$, so heißt $v$ der i. Output des Knotens $v$.*

- *Die Abbildung $typ : V \to \Omega \cup I/O$ ordnet jedem Knoten einen Typ (also eine Boolesche Funktion oder ein Element aus $I/O$) zu. Dabei muß gelten:*
  $\mid typ^{-1}(EPAD) \mid = n$ *und* $\mid typ^{-1}(APAD) \mid = m$.
  *Falls* $typ(v) = EPAD \Rightarrow indeg(v) = 0$.
  *Falls* $typ(v) = APAD \Rightarrow indeg(v) = 1$ *und* $outdeg(v) = 0$.
  *Falls* $typ(v) \in \Omega \Rightarrow typ(v) \in B_{indeg(v)}$.

- *Alle Knoten $v$ mit $typ(v) \in I/O$ werden numeriert durch die bijektiven Abbildungen*
  $pe : \{1, \ldots, n\} \to \{v \in V \mid typ(v) = EPAD\}$ *und*

*$pa : \{1, \ldots, m\} \rightarrow \{v \in V \mid typ(v) = APAD\}$.*
*$pe(i)$ ($pa(i)$) heißt der i. primäre Eingang (Ausgang) des Schaltkreises.*

Diese Definition der Ω-Schaltkreise ist sehr allgemein, d.h. jeder physikalische Schaltkreis kann durch ein entsprechendes Tupel $(G, typ, pe, pa)$ über einer Zellenbibliothek Ω als Graph dargestellt werden. Bei einer physikalischen Realisierung eines (formalen) Schaltkreises sind im allgemeinen jedoch weitere Einschränkungen nötig. So kann z.B. die Abbildung $outdeg(v)$ für einige Grundgatter, aufgrund elektrotechnischer Erfordernisse, nach oben beschränkt sein (Fanout-Restriktion).

Die beiden folgenden Definitionen stellen eine Zuordnung zwischen der Menge der Booleschen Funktionen sowie der Menge der Ω−Schaltkreise her:

**Definition 1.11 (Durch Ω-Schaltkreis berechnete Boolesche 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 die Leitung $e$ berechnete Funktion definiert durch*

$$f_e = \pi_i^n = x_i.$$

- *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*

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

*Seien nun für $1 \leq i \leq m$, $y_i = I(pa(i), 1)$ und $f_{y_i}$ jeweils die durch $y_i$ berechneten Funktionen.*
*Dann ist die durch den Schaltkreis $S$ berechnete Boolesche Funktion $f_S : \{0,1\}^n \rightarrow \{0,1\}^m$ definiert durch*

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

Wie aus Definition 1.11 hervorgeht, berechnet ein Schaltkreis stets eine *totale* Boolesche Funktion. Oftmals sind allerdings Schaltungen $S$ zu entwerfen, welche in ein größeres System $\tilde{S}$ eingebettet sind. Daher sind die Eingaben dieser Schaltungen auch von den Ausgaben anderer Schaltungen in $\tilde{S}$ abhängig. Es ist somit möglich, bestimmte Eingabevektoren für diese eingebettete Schaltung $S$ a priori auszuschließen. Für das korrekte logische Verhalten von $\tilde{S}$ sind daher nur für eine Teilmenge $D \subseteq \{0,1\}^n$ die Funktionswerte der Funktion $f_S \in B_{n,m}$ von Bedeutung.
Deshalb soll $S$ als Realisierung *aller* Funktionen dienen, die man aus $f_S$ durch Einschränkung des Definitionsbereichs auf $D$ erhält:

**Definition 1.12 (Realisierung einer Booleschen Funktion)**
*Sei $f : D \rightarrow \{0,1\}^m$, $D \subseteq \{0,1\}^n$ , und $S$ ein $\Omega$-Schaltkreis mit $n$ Eingängen und $m$ Ausgängen.*
*Ist $f_S \in B_{n,m}$ die durch $S$ berechnete Boolesche Funktion, dann heißt $S$* **Realisierung von** *$f$, falls $(f_S)|_D = f$.*
*Dabei bezeichnet $(f_S)|_D$ die Einschränkung von $f_S$ auf die Menge $D$.*

Einige Beispiele sollen diese Definitionen verdeutlichen:

**Beispiele**
Ist die Boolesche Funktion $less \in B_4$ gegeben durch
$less(x_1, x_2, x_3, x_4) = 1 \quad :\Longleftrightarrow \; 2x_1 + x_2 < 2x_3 + x_4$, so ist der Schaltkreis aus Abbildung 1.1 eine Realisierung von $f$.
Es gilt $less(x_1, x_2, x_3, x_4) = 1 \iff x_1 < x_3 \vee (x_1 = x_3 \wedge x_2 < x_4)$.
Die Schaltung in Abbildung 1.2 realisiert sowohl die totale Funktion *ite* als auch die partielle Boolesche Funktion $f \in BP_3$ mit $f(x_1, x_2, s) = 1$, falls $x_1 = x_2 = 1$ und $f(x_1, x_2, s) = 0$, falls $x_1 = x_2 = 0$ sowie $f(x_1, x_2, s) = *$ sonst.

### 1.3.1 Kostenmaße und Komplexität

Offensichtlich können für eine Boolesche Funktion mehrere Realisierungen existieren. Dies gilt insbesondere für partielle Funktionen. Um verschiedene Schaltkreisrealisierungen in ihrer Güte miteinander vergleichen zu können, werden Kostenmaße eingeführt und jeder Schaltung gewisse Kosten zugeordnet. Diese Kostenmaße können sich auf unterschiedliche Größen beziehen, z.B. auf die Laufzeit, Anzahl der Grundgatter, verbrauchte Chipfläche, etc.
Ein spezielles Kostenmaß stellt die Komplexität einer Schaltung dar:

**Definition 1.13 (Komplexität eines $\Omega-$Schaltkreises)**
*Sei $S = ((V, E), typ, pe, pa)$ ein $\Omega-$Schaltkreis und $C : \Omega \rightarrow \mathbf{R}_0^+$ eine Abbildung, die jedem Grundgatter $g \in \Omega$ die Kosten $C(g)$ zuordnet. Dann ist*

$$C_\Omega(S) := \sum_{\substack{v \in V \\ typ(v) \in \Omega}} C(v)$$

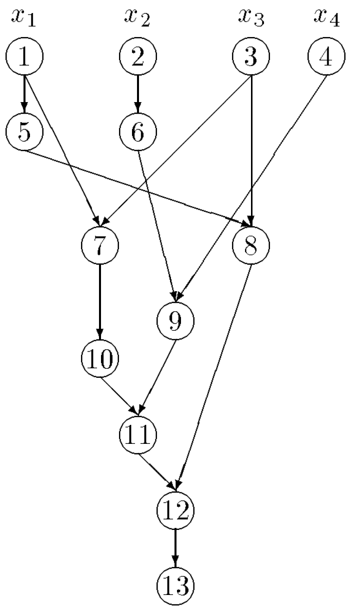
*die* $\Omega-$**Komplexität von** *$S$.*

Zumeist handelt es sich bei der Kostenfunktion $C$ um eine Konstantenfunktion, d.h. jedes Grundgatter besitzt die gleichen Kosten[†].

Oftmals vergleicht man Schaltungen auch hinsichtlich ihrer Tiefe:

**Definition 1.14 (Tiefe eines $\Omega-$Schaltkreises)**
*Sei $S = (G, typ, pe, pa)$ ein $\Omega-$Schaltkreis. Dann ist die* **Tiefe** *von $S$ ($T_\Omega(S)$) die größte Anzahl von Zellen auf einem gerichteten Pfad in $G$ (EPAD- und APAD-Zellen nicht mitgerechnet).*

[†]Im folgenden soll deshalb stets $C(g) = 1$ für alle Gatter $g \in \Omega$ gelten.




| $v$ | $typ(v)$ |
|---|---|
| $v_1$ | EPAD |
| $v_2$ | EPAD |
| $v_3$ | EPAD |
| $v_4$ | EPAD |
| $v_5$ | *not* |
| $v_6$ | *not* |
| $v_7$ | *exor* |
| $v_8$ | *and* |
| $v_9$ | *and* |
| $v_{10}$ | *not* |
| $v_{11}$ | *and* |
| $v_{12}$ | *or* |
| $v_{13}$ | APAD |

| $i$ | $pe(i)$ |
|---|---|
| 1 | $v_1$ |
| 2 | $v_2$ |
| 3 | $v_3$ |
| 4 | $v_4$ |

| $i$ | $pa(i)$ |
|---|---|
| 1 | $v_{13}$ |

Abbildung 1.1: Schaltkreis für $less(x_1, x_2, x_3, x_4)$

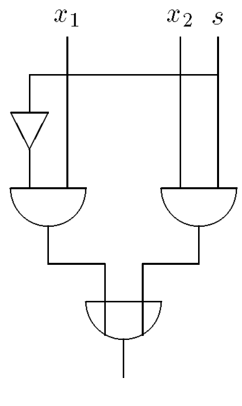



Abbildung 1.2: $B_2$−Schaltkreis für *ite* (vereinfachte Darstellung)

Oft ist nicht nur die Realisierung *einer* Booleschen Funktion erwünscht, sondern Schaltkreise für eine Familie von Funktionen, d.h. eine Menge $\{f_i\}_{i\in\mathbf{N}}$ gesucht, die sich nur in der Anzahl der Eingänge unterscheiden, das „Berechnungs prinzip" aber allen Funktionen aus der Menge gemeinsam ist.

Ein Beispiel hierfür ist die Addition zweier binärer Zahlen, die durch die Funktionen $\{add_i\}_{i\in\mathbf{N}}$ mit
$add_i(x_{i-1},\ldots,x_0,y_{i-1},\ldots,y_0) = (z_i,\ldots,z_0)$ und

$$\sum_{j=0}^{i} z_j 2^j = \sum_{j=0}^{i-1} x_j 2^j + \sum_{j=0}^{i-1} y_j 2^j$$

beschrieben wird.

Die Realisierung dieser Familie führt somit zu einer Menge von Schaltkreisen. Die Komplexität der Schaltkreisfamilie kann daher durch eine Funktion angegeben werden, die von der Anzahl der Eingänge abhängt. Man erhält so asymptotische Aussagen bzgl. der Komplexität dieser Schaltkreise.

Die Komplexität einer Schaltung hängt sowohl von der Wahl der Zellenbibliothek $\Omega$ als auch von der Kostenfunktion $C:\Omega\to\mathbf{R}_0^+$ ab. Jeder $\Omega_1-$Schaltkreis $S_1$ kann in eine $\Omega_2-$Schaltung $S_2$ umgewandelt werden, welche die gleiche Boolesche Funktion berechnet, vorausgesetzt es handelt sich um vollständige Bibliotheken. Dazu wird jedes Gatter in $S_1$ durch einen vorgegebenen $\Omega_2-$Schaltkreis ersetzt, der diese Gatterfunktion realisiert. Wie bereits erwähnt, werden üblicherweise allen Grundgattern die gleichen Kosten zugeordnet (meist 1). In diesem Fall unterscheiden sich die Komplexitäten der beiden Schaltungen $S_1$ und $S_2$ nur durch eine Konstante, die durch die maximale Komplexität unter allen vorgegebenen $\Omega_2-$Schaltungen für die Grundgatter in $\Omega_1$ bestimmt wird[†]. Aus diesem Grund ist es keine wesentliche Einschränkung, bei allen (asymptotischen) Aussagen bzgl. der Komplexität einer Schaltkreisfamilie stets $C_{B_2}$ zu betrachten.

Man ist meist daran interessiert, zu einer Booleschen Funktion die billigste Realisierung bzgl. der Komplexität zu finden. Die Komplexität der billigsten Schaltung soll daher besonders ausgezeichnet werden und dient als Maß für die Komplexität aller Booleschen Funktionen, die durch diese Schaltung realisiert werden:

**Definition 1.15 (Komplexität einer Booleschen Funktion)**
*Ist $f\in BP_{n,m}$, dann heißt*

$$C_\Omega(f) := min\{C_\Omega(S) \mid S \text{ ist eine } \Omega\text{-Realisierung für } f\}$$

**$\Omega-$Komplexität von** $f$.

[†]Zellenbibliotheken sind *endliche* Mengen. Somit existiert immer dieser maximale Wert.

## 1.4 Erweiterungen partieller Boolescher Funktionen

Aus dem vorigen Abschnitt geht hervor, daß ein Schaltkreis gleichzeitig verschiedene partielle Funktionen realisiert. Deshalb sollen Boolesche Funktionen, die durch eine *gemeinsame* Schaltung realisiert werden können, in Relation zueinander gebracht werden. Dies führt zu dem wichtigen Begriff der *Erweiterung* einer Booleschen Funktion:

**Definition 1.16 (Erweiterung einer partiellen Booleschen Funktion)** *Sei $f \in S(D)$, $D \subseteq \{0,1\}^n$. Eine Funktion $g \in S(D')$, $D \subseteq D'$, mit $g|_D = f$ heißt* **Erweiterung der partiellen Booleschen Funktion** *$f$ $(f \preceq g)$.*
*Ist $g \in B_n$, so heißt $g$* **totale Erweiterung von** *$f$.*

**Bezeichnung 4**

$$Erw(f) := \{g \mid f \preceq g\}$$

*heißt* **Erweiterungsmenge von** *$f$.*

Die Erweiterungsmenge einer partiellen Funktion $f \in BP_n$ stellt also die Menge aller Booleschen Funktionen dar, deren Realisierungen auch gleichzeitig eine Realisierung für $f$ sind.
Für die Mächtigkeit dieser Menge gilt offensichtlich:

**Korollar 1.2** *Für $f \in BP_n$ gilt:*

$$\mid Erw(f) \mid = 3^{|DC(f)|}.$$

Das folgende Lemma zeigt, wie sich für zwei Funktionen entscheiden läßt, ob eine der Funktionen eine Erweiterung der anderen bildet:

**Lemma 1.1** *Sei $f \in S(D)$ und $g \in S(D')$ mit $D \subseteq D'$. Dann gilt:*

$$f \preceq g \iff g_{on} \wedge f_{off} = \mathbf{0} \text{ und } g_{off} \wedge f_{on} = \mathbf{0}.$$

**Beweis:** „$\Longrightarrow$"
Aus $g \in Erw(f)$ folgt $ON(f) \subseteq ON(g)$ und $OFF(f) \subseteq OFF(g)$. Da $ON(g) \cap OFF(g) = \emptyset \Rightarrow ON(f) \cap OFF(g) = \emptyset$, d.h. $f_{on} \wedge g_{off} = \mathbf{0}$.
Ebenso gilt $OFF(f) \cap ON(g) = \emptyset$, also $f_{off} \wedge g_{on} = \mathbf{0}$.
„$\Longleftarrow$"
Aus $D \subseteq D'$ folgt $f(x) \neq * \Rightarrow g(x) \neq *$.
Aus $g_{on} \wedge f_{off} = \mathbf{0}$ folgt $f(x) = 0 \Rightarrow g(x) = 0$ und wegen $g_{off} \wedge f_{on} = \mathbf{0}$ gilt $f(x) = 1 \Rightarrow g(x) = 1$.
Das bedeutet $g \in Erw(f)$.

□

Anstelle der Funktionen $f_{on}, f_{off}, f_{dc}$ können auch die entsprechenden Mengen $(ON, OFF, DC)$ betrachtet werden. Aus Lemma 1.1 und dessen Beweis ergibt sich unmittelbar das folgende Korollar:

**Korollar 1.3** *Sei $f \in S(D)$ und $g \in S(D')$ mit $D \subseteq D'$. Dann gilt:*

$$f \preceq g \iff ON(g) \cap OFF(f) = \emptyset \text{ und } OFF(g) \cap ON(f) = \emptyset.$$
$$f \preceq g \iff ON(f) \subseteq ON(g) \text{ und } OFF(f) \subseteq OFF(g).$$

Aus Korollar 1.3 folgt zugleich, daß es sich bei der Erweiterungsrelation $\preceq$ um eine transitive und trivialerweise auch reflexive Relation handelt. Folgendes Korollar hält diese wichtige Aussage fest:

**Korollar 1.4** *Seien $f, g, h \in BP_n$. Dann gilt:*

$$f \preceq g \text{ und } g \preceq h \Rightarrow f \preceq h.$$

Bei der Suche nach einer Realisierung für eine partielle Funktion $f \in BP_n$ erhält man somit Schaltkreise, die (totale) Erweiterungen von $f$ realisieren. Verschiedene Erweiterungen einer partiellen Funktion $f$ können allerdings *extrem* unterschiedliche Komplexitäten besitzen, so daß die Bestimmung einer geeigneten Erweiterung sehr wichtig ist.
Da die Erweiterungsmenge $Erw(f)$ exponentiell mit der Anzahl der unspezifizierten Funktionsstellen wächst, ist es allerdings im allgemeinen nicht praktikabel, *alle* Erweiterungen zu untersuchen, um eine vorgegebene partielle Funktion effizient durch eine Schaltung zu realisieren.
In der vorliegenden Arbeit wird deshalb mittels sukzessiver Belegung der don't care-Werte eine Folge von Erweiterungen $f \preceq f^1 \preceq f^2 \ldots \preceq f^k$ bestimmt, so daß die resultierende Funktion $f^k$ eine möglichst geringe Komplexität besitzt.

## 1.5 Darstellungen Boolescher Funktionen

Um z.B. den Entwurf von Schaltungen für vorgegebene Funktionen zu automatisieren, müssen Boolesche Funktionen im Rechner dargestellt werden. Sehr häufig werden dabei aus vorgegebenen Funktionen neue Funktionen gebildet, z.B. mit Hilfe von Algorithmen für die Booleschen Operationen. Die Laufzeit solcher Algorithmen wird sehr stark dadurch beeinflußt, in welcher Form die Funktionen dargestellt werden.
Um eine geeignete Darstellungsform auszuwählen, untersucht man meist u.a. folgende Eigenschaften:

- Ist die Darstellung für die meisten Funktionen klein?
- Kann die Darstellung des Ergebnisses einer Booleschen Operation im Verhältnis zu den Darstellungen der Operanden in ihrer Größe inakzeptabel, d.h. exponentiell wachsen?
- Ist die Darstellung eindeutig? Dies erleichtert z.B. den Test, ob zwei Darstellungen äquivalent sind, d.h. die gleiche Boolesche Funktion repräsentieren.
- Wie läßt sich die Darstellung in einen Schaltkreis überführen? Wie hängt die Komplexität der Schaltung von der Größe der Darstellung ab?

Es sei darauf hingewiesen, daß nicht alle diese Eigenschaften gleiches Gewicht für die Auswahl der Darstellungsform besitzen müssen. Es kann durchaus vorkommen, daß z.B. die Eindeutigkeit der Darstellung nicht notwendigerweise erfüllt sein muß, da diese für die konkrete Anwendung keinen Vorteil bringt.
Zudem können sich die aufgelisteten Punkte auch oftmals widersprechen. So kann z.B. die Eindeutigkeit einer Darstellung zu größeren Ergebnissen führen als eine mehrdeutige Beschreibungsform.
In den nächsten Abschnitten werden einige bekannte Darstellungsformen vorgestellt und deren spezielle Eigenschaften näher untersucht.

### 1.5.1 Funktionstabellen

Funktionstabellen sind einfach nur eine tabellarische Auflistung aller Funktionsstellen und deren Funktionswerte. Partielle Funktionen können dadurch dargestellt werden, daß Funktionsstellen, die sich nicht im Definitionsbereich befinden, mit dem Funktionswert $*$ markiert werden.

**Bemerkung 3**

- *Alle Funktionen aus $BP_{n,m}$ können als Funktionstabelle (mit $m$ Ergebnisspalten) dargestellt werden.*
- *Funktionstabellen bilden trivialerweise eine eindeutige Darstellung aller Booleschen Funktionen.*
- *Aus der Beschreibung einer Funktion $f \in BP_{n,m}$ in Form einer Funktionstabelle läßt sich eine Realisierung dieser Funktion synthetisieren, deren $B_2$-Komplexität $O(m \cdot n \cdot 2^n)$ beträgt (z.B. disjunktive Normalform (DNF)).*

Ein Vorteil dieser Tabellen liegt u.a. in der Berechnung eines Funktionswertes, die in konstanter Zeit ausgeführt werden kann[†].
Funktionstabellen haben allerdings einen entscheidenden Nachteil: Sie benötigen, solange man keine Reduktion vornimmt, für *alle* Booleschen Funktionen exponentiellen Speicherplatz (bezogen auf die Anzahl der Eingangsvariablen), da alle $2^n$ verschiedenen Funktionsstellen aufgelistet werden müssen. Damit ist auch die Laufzeit der Algorithmen, welche beispielsweise das Ergebnis Boolescher Operationen berechnen, ebenfalls exponentiell in der Größe der Variablenmenge, falls die beteiligten Funktionen durch eine Tabelle dargestellt werden.
Selbst wenn man in der Tabelle nur einen festen Funktionswert (0 oder 1) protokolliert, werden alle kleinen Tabellen durch eine einfache Negierung der dargestellten Funktion sehr groß.
Wegen ihrer inhärent exponentiellen Größe sind Funktionstabellen in der Praxis meist nicht sonderlich geeignet, um Boolesche Funktionen darzustellen.

[†]Man stelle sich die Tabelle durch ein Array implementiert vor.

### 1.5.2 Boolesche Ausdrücke

Eine weitere Möglichkeit, um Boolesche Funktionen darzustellen, bieten Boolesche Ausdrücke oder Boolesche Formeln:

**Definition 1.17 (Boolesche Ausdrücke)** *Sei $X = \{X_1, \ldots, X_n\}$ eine endliche Menge (Variablenmenge), $E = \{0, 1, (, ), \wedge, \vee, \neg\} \cup X$ ein Alphabet und $E^+$ die freie Wörterhalbgruppe über $E$.*
*Dann heißt die kleinste Menge $\mathcal{B}(X) \subseteq E^+$ mit*

- $\{0, 1\} \cup X \subseteq \mathcal{B}(X)$
- $F_1, \ldots, F_i \in \mathcal{B}(X) \Rightarrow (F_1 \vee \ldots \vee F_i) \in \mathcal{B}(X)$
- $F_1, \ldots, F_i \in \mathcal{B}(X) \Rightarrow (F_1 \wedge \ldots \wedge F_i) \in \mathcal{B}(X)$
- $F \in \mathcal{B}(X) \Rightarrow \neg(F) \in \mathcal{B}(X)$

*Menge der* **(vollständig geklammerten) Booleschen Ausdrücke oder Booleschen Formeln über** $X$.

Jedem Booleschen Ausdruck ordnet man mittels einer Kostenfunktion bestimmte Kosten zu:

**Definition 1.18 (Kosten eines Booleschen Ausdrucks)** *Sei $F \in \mathcal{B}(X)$ ein Boolescher Ausdruck. Dann sind die* **Kosten** $C(F)$ **von** $F$ *wie folgt definiert:*

$$\begin{aligned}
C(0) &= C(1) = 0 \\
C(x) &= 0 \quad \forall x \in X \\
C((F_1 \vee \ldots \vee F_i)) &= \sum_{j=1}^{i} C(F_j) + (i-1) \\
C((F_1 \wedge \ldots \wedge F_i)) &= \sum_{j=1}^{i} C(F_j) + (i-1) \\
C(\neg(F)) &= C(F) + 1
\end{aligned}$$

Die Kosten eines Ausdrucks ergeben sich also, indem man die Vorkommen von $\{\wedge, \vee, \neg\}$ summiert.

**Definition 1.19 (Durch einen Ausdruck dargestellte Funktion)**
*Sei $F \in \mathcal{B}(X)$, $|X| = n$ ein Boolescher Ausdruck und $f \in B_n$.*
*Dann heißt $f_F$* **die durch** $F$ **dargestellte Funktion** *genau dann, falls gilt:*

$$\begin{aligned}
F = 0 &\Rightarrow f_F = \mathbf{0} \\
F = 1 &\Rightarrow f_F = \mathbf{1} \\
F = x_i \quad x_i \in X &\Rightarrow f_F = x_i \\
F = (F_1 \wedge \ldots \wedge F_i) &\Rightarrow f_F = f_1 \wedge f_2 \wedge \ldots \wedge f_i \\
F = (F_1 \vee \ldots \vee F_i) &\Rightarrow f_F = f_1 \vee f_2 \vee \ldots \vee f_i \\
F = \neg(G) &\Rightarrow f_F = \overline{g}
\end{aligned}$$

*wobei* $f_j$ *die durch den Booleschen Ausdruck* $F_j$ $(1 \leq j \leq i)$ *dargestellten Funktionen sind und* $g$ *die durch den Booleschen Ausdruck* $G$ *dargestellte Funktion ist.*

In der Definition ist zu beachten, daß $\wedge$ und $\vee$ auf der linken Seite der Formeln jeweils Boolesche Operationen bezeichnen, während es sich auf der rechten Seite lediglich um einfache Textzeichen aus der Menge $E$ handelt.
Aus einem Booleschen Ausdruck läßt sich sehr leicht eine Realisierung der dargestellten Funktion bestimmen:
Da die Booleschen Ausdrücke vollständig geklammert sind, kann jeder Formel ein eindeutiger Syntaxbaum zugeordnet werden. Die Knoten dieses Baums sind mit den entsprechenden Operationen ( $\wedge$ , $\vee$ , $\neg$ ) aus Definition 1.17, die Blätter mit Konstanten oder Variablen indiziert.
Dieser Baum wird in einen entsprechenden Schaltkreis umgewandelt, indem alle inneren Knoten durch Gatter ersetzt werden. Ein Knoten, der seine $i$ Nachfolger disjunktiv miteinander verknüpft, kann durch einen $B_2$-Schaltkreis mit $i-1$ $or$-Gattern ersetzt werden. Entsprechendes gilt für die Konjunktion. Knoten, deren Beschriftung die Negation ist, werden durch ein $not$-Gatter ersetzt. Die Verbindungsleitungen dieses $\Omega$-Schaltkreises ($\Omega = \{\mathbf{0}, \mathbf{1}, and, or, not\}$) entsprechen dann den Kanten des Baums.
Die $\Omega-$Komplexität der so gewonnenen Schaltung ist identisch mit den Kosten des Booleschen Ausdrucks. Allerdings ist die Komplexität natürlich im allgemeinen nicht optimal, d.h. sie muß nicht der Komplexität der dargestellten Booleschen Funktion entsprechen.

**Definition 1.20 (Äquivalente Boolesche Ausdrücke)** *Zwei Boolesche Ausdrücke* $F, G$ *über* $X$ *heißen* **äquivalent** $(F \equiv G)$ *genau dann, falls* $f_F = f_G$.

Die Darstellung Boolescher Funktionen durch Boolesche Ausdrücke ist nicht eindeutig, da es nichttriviale Äquivalenzen gibt:

**Lemma 1.2** *Seien* $F, G, H \in \mathcal{B}(X)$. *Dann gilt:*

- $$F \wedge 0 \equiv 0$$
  $$F \vee 0 \equiv F$$
  $$F \wedge 1 \equiv F$$
  $$F \vee 1 \equiv 1$$
  $$F \vee \overline{F} \equiv 1$$
  $$F \wedge \overline{F} \equiv 0$$

- *Distributivgesetze*
  $$F \wedge (G \vee H) \equiv (F \wedge G) \vee (F \wedge H)$$
  $$F \vee (G \wedge H) \equiv (F \vee G) \wedge (F \vee H)$$

- *De-Morgan Gesetze*

$$\overline{F \wedge G} \equiv \overline{F} \vee \overline{G}$$
$$\overline{F \vee G} \equiv \overline{F} \wedge \overline{G}$$

- *Gesetz der doppelten Negation*

$$\overline{\overline{F}} \equiv F$$

**Beweis:** Die Beweise zu den angegebenen Formeln findet der Leser in jedem Buch zur Booleschen Algebra.

□

#### 1.5.2.1 Boolesche Polynome

Eine in der zweistufigen Logiksynthese häufig verwendete Darstellung Boolescher Funktionen sind Boolesche Polynome. Dies hat u.a. folgende Gründe:

- Boolesche Polynome lassen sich direkt in ein PLA (Programmable Logic Array) überführen. PLAs sind vorgefertigte Chips, die sich durch einfache Manipulationen so programmieren lassen, daß sie spezielle Aufgaben erfüllen. Die interne Struktur solcher PLAs entspricht gerade dem Aufbau der Polynome. Dadurch kann ein Boolesches Polynom unmittelbar zur Programmierung verwendet werden.
  Durch die Vorfertigung können diese Chips relativ billig produziert werden.

- Es ist sehr gut untersucht und verstanden worden, wie man Boolesche Polynome in ihren Kosten minimieren kann. Es gibt effiziente Verfahren, die zu einer Funktion einen minimalen bzw. nahezu minimalen (zweistufigen) Schaltkreis berechnen, der diese Funktion realisiert (z.B. [BHM+84]).

Bevor die Eigenschaften dieser Darstellungsform näher untersucht werden, muß jedoch zunächst auf den Aufbau eines Polynoms eingegangen werden. Boolesche Polynome lassen sich zerlegen in Literale und Monome:

**Definition 1.21 (Literal)** *Sei $X = \{x_1, \ldots, x_n\}$ eine Variablenmenge. Die Booleschen Ausdrücke $x_i$ und $\overline{x_i}$ heißen* **Literale** *über $X$.*
*Anstelle von $x_i$ ($\overline{x_i}$) schreibt man auch $x_i^1$ ($x_i^0$).*

**Definition 1.22 (Boolesches Monom)** *Seien $L_1, \ldots, L_i$ verschiedene Literale über $X$ mit $L_k \neq \overline{L_l}$ für $k, l \in \{1, \ldots, i\}$. Ein Boolescher Ausdruck der Form*

$$m = (L_1 \wedge L_2 \wedge \ \ldots \ \wedge L_i)$$

*heißt* **Boolesches Monom** *über $X$.*
*Ein Boolesches Monom $m = (L_1 \wedge \ \ldots \ \wedge L_n)$ heißt* **vollständig**, *falls jedes $x \in X$ als Literal $x$ oder $\overline{x}$ in $m$ auftaucht.*
*Ein Boolesches Monom $m'$ heißt* **Teilmonom** *von $m$, falls $m'$ aus $m$ durch Streichen einiger Literale hervorgeht.*

**Definition 1.23 (Boolesches Polynom)** *Ein* **Boolesches Polynom** *$p$ über $X$ ist ein Boolescher Ausdruck der Form*

- *$p = 0$ oder*
- *$p = 1$ oder*
- *$p = (m_1 \vee m_2 \vee \ldots \vee m_i)$,*
  *wobei $m_1, \ldots, m_i$ Boolesche Monome über $X$ sind.*

Boolesche Polynome sind damit ebenfalls Boolesche Ausdrücke und somit mittels des skizzierten Verfahrens aus dem vorigen Abschnitt in einen $\Omega-$Schaltkreis $S_p$ mit $\Omega = \{\mathbf{0}, \mathbf{1}, and, or, not\}$ überführbar. Die Kosten eines Polynoms unterscheiden sich in ihrer Definition jedoch häufig von dem Maß aus Definition 1.18. In die Kosten eines Polynoms $p$ geht üblicherweise nur die Anzahl der Elemente $\wedge$ und $\vee$ ein, Negationen werden nicht berücksichtigt. Ein Inverter kann nämlich mit einem nachfolgenden Gatter zu einem Grundgatter aus $R_2 := B_2 \backslash \{exor, equiv\}$ verknüpft werden. Die Kosten eines Polynoms entsprechen demnach der $R_2-$Komplexität von $S_p$.

Schaltfunktionen mit *mehreren* Ausgängen können ebenfalls durch Polynome dargestellt werden:

**Definition 1.24 (Boolesches Polynom mit *mehreren* Ausgängen)**
*Eine Folge $p = (p_1, p_2, \ldots, p_m)(m \geq 1)$ heißt* **Boolesches Polynom mit $m$ Ausgängen** *über $X$ genau dann, falls $p_i$ ein Boolesches Polynom über $X$ ist für $1 \leq i \leq m$.*

Jedes Teilpolynom $p_i$ repräsentiert dabei eine Boolesche Funktion mit einem Ausgang und zwar die $i$. Teilfunktion[†] der Booleschen Funktion $f \in B_{n,m}$ , die durch $p$ dargestellt wird.

**Definition 1.25 (Durch ein Polynom dargestellte Funktion)**
*Sei $p = (p_1, \ldots, p_m)$ $(m \geq 1)$ ein Boolesches Polynom. Dann ist die Funktion $f \in B_{n,m}$ mit $f_i = f_{p_i}$ $(1 \leq i \leq m)$ die* **durch $p$ dargestellte Boolesche Funktion**.

Die Realisierung eines Polynoms besteht aus zwei Ebenen. In der ersten Ebene, der $AND$-Ebene, werden alle Monome in gewohnter Weise durch $and-$Gatter realisiert und diese in der $OR$-Ebene durch $or-$Gatter zusammengefaßt.
In einem PLA ist die Größe der $AND-$ und $OR-$Ebene fest vorgegeben. Die Anzahl der Verbindungsleitungen zwischen den beiden Ebenen entspricht gerade der Anzahl der Monome des Polynoms. Sie ist in einem PLA-Baustein beschränkt.
Die Kosten eines Polynoms $p = (p_1, \ldots, p_m)$ ergeben sich allerdings *nicht* durch Aufsummieren der Kosten der einzelnen Teilpolynome, da gleiche Monome $m$, die in verschiedenen Teilpolynomen vorkommen, nur einmal realisiert werden

[†]Unter der $i$. Teilfunktion von $f \in B_{n,m}$ (bzw. $f \in BP_{n,m}$) versteht man die Funktion $f_i \in B_n$ mit $\forall x \in \{0,1\}^n : f_i(x_1, \ldots, x_n) = \pi_i^m(f(x_1, \ldots, x_n))$.

müssen. Der Ausgang des (Teil-)Schaltkreises, der $m$ realisiert, kann nämlich mehrmals abgegriffen werden und so allen Schaltungen der $OR-$Ebene zur Verfügung gestellt werden, die Teilfunktionen berechnen, deren Polynom $p_i$ das Monom $m$ enthält. Es ergibt sich daher:

**Definition 1.26 (Kosten eines Polynoms mit mehreren Ausgängen)** *Sei $p = (p_1, \ldots, p_m)$ $(m \geq 1)$ eine Boolesches Polynom mit $p_i = m^i_{l_1} \vee \ldots \vee m^i_{l_i}$. Sei*

$$\{m_1, \ldots, m_k\} = \bigcup_{i=1}^{m} \cup_{j=1}^{l_i} m^i_j$$

*die Menge, der in $p_1, \ldots, p_m$ vorkommenden Monome. Dann sind die Kosten des Polynoms $p$*

$$C(p) = \sum_{i=1}^{k} C(m_i) + \sum_{i=1}^{m} (l_i - 1).$$

Da es zu einer gegebenen Booleschen Funktion mehrere äquivalente Polynome geben kann und man bestrebt ist, billige Realisierungen in Form von Schaltkreisen zu bestimmen, ist man an dem Polynom interessiert, das unter den äquivalenten Polynomen die geringsten Kosten besitzt.

**Definition 1.27 (Minimalpolynom)** *Sei $f \in B_{n,m}$ eine Boolesche Funktion mit $n$ Eingängen und $m$ Ausgängen und $p$ ein Boolesches Polynom über $X$ mit $m$ Ausgängen. Dann heißt $p$* **Minimalpolynom von** $f$ *genau dann, falls gilt:*

1. $f_p = f$.
2. *Gilt $f_{p'} = f$ für eine weiteres Polynom $p'$, dann ist $C(p) \leq C(p')$.*

**Bemerkung 4**

- *Minimalpolynome sind im allgemeinen* nicht *eindeutig bestimmt.*
- *Es gibt relativ gute Verfahren, um zu einer Booleschen Funktion bzw. dem entsprechenden Polynom $p$ ein Minimalpolynom bzw. ein dazu äquivalentes Polynom zu bestimmen, dessen Kosten sich nicht erheblich von denen des Minimalpolynoms unterscheiden (z.B. [BHM+84]).*

Die beiden nächsten Beispiele zeigen zum einen, daß die Darstellung Boolescher Funktionen mittels Boolescher Polynome nicht für alle Funktionen zu einem befriedigenden Resultat führt und zum anderen, daß schon einfache Boolesche Operationen, wie die Negation, Minimalpolynome *extrem* verteuern können. Dazu benötigt man folgende Begriffe:

**Definition 1.28 (Implikant, Primimplikant)** *Sei $m$ ein Monom über der Variablenmenge $X$ und $f \in B_n$ eine Boolesche Funktion.*
*$m$ heißt* **Implikant** *von $f$, falls gilt:*

$$f_m(\epsilon) = 1 \Rightarrow f(\epsilon) = 1 \quad \forall \epsilon \in \{0,1\}^n .$$

*$m$ heißt* **Primimplikant** *von $f$, falls gilt:*

- *$m$ ist Implikant von $f$ und*
- *kein Teilmonom von $m$ ist Implikant von $f$.*

Es gibt effiziente Verfahren, um alle Primimplikanten einer Booleschen Funktion zu bestimmen [Quine55, Clus56].

Für spezielle Polynome gilt der folgende Satz [Quine53]:

**Satz 1.1** *Ist $p = m_1 \vee \ldots \vee m_l$ ein Polynom, dessen Monome nur positive (nur negative) Literale enthalten und zudem jedes Monom ein Primimplikant von $f_p$ ist, so stellt $p$ das einzige Minimalpolynom von $f_p$ dar.*

Die folgenden beiden Lemma zeigen, daß es nicht immer günstig ist, zur Darstellung Boolescher Funktionen Minimalpolynome zu verwenden:

**Lemma 1.3** *Sei $pair_n \in B_{2n}$ wie folgt definiert:*

$$pair_n(x_1, \ldots, x_{2n}) := (x_1 \vee x_2) \wedge (x_3 \vee x_4) \wedge \ldots \wedge (x_{2n-1} \vee x_{2n}).$$

*Dann besitzt jede Darstellung für $pair_n$ als Boolesches Polynom exponentielle Kosten.*

**Beweis:** Durch mehrmalige Anwendung der De-Morgan Gesetze erhält man für $pair_n$ folgendes Polynom

$$p = \bigvee_{(\epsilon_1, \ldots, \epsilon_n) \in \{0,1\}^n} x_{2-\epsilon_1} \wedge x_{4-\epsilon_2} \wedge \ldots \wedge x_{2n-\epsilon_n}$$

Da jedes Monom in $p$ offensichtlich Primimplikant von $pair_n$ ist, stellt $p$ nach Satz 1.1 das einzige Minimalpolynom von $pair_n$ dar mit Kosten $C(p) \geq 2^n$.

□

Erstaunlicherweise läßt sich die Negation der Funktion kompakter darstellen:

**Lemma 1.4** *Es existiert ein Polynom $p$ zu $\overline{pair_n}$ mit Kosten $C_p \leq 2n - 1$.*

**Beweis:**

$$\begin{array}{rcl}
\overline{pair_n} & = & \overline{(x_1 \vee x_2) \wedge (x_3 \vee x_4) \wedge \ldots \wedge (x_{2n-1} \vee x_{2n})} \\
& \equiv & \overline{(x_1 \vee x_2)} \vee \overline{(x_3 \vee x_4)} \vee \ldots \vee \overline{(x_{2n-1} \vee x_{2n})} \\
& \equiv & (\overline{x_1} \wedge \overline{x_2}) \vee (\overline{x_3} \wedge \overline{x_4}) \vee \ldots \vee (\overline{x_{2n-1}} \wedge \overline{x_{2n}})
\end{array}$$

Das Polynom besitzt $n$ $\wedge$ Gatter sowie $n-1$ $\vee$ Gatter und damit Kosten von $2n-1$.
Es handelt sich nach Satz 1.1 dabei sogar um das (einzige) Minimalpolynom für $\overline{pair_n}$.

□

Das bedeutet, daß jeder Algorithmus für die Negation bei Eingabe des Minimalpolynoms für $\overline{pair_n}$ eine Ausgabe liefern muß, deren Länge exponentiell in der Größe der Eingabe ist.

### 1.5.3 Binary Decision Diagrams

In diesem Abschnitt soll eine weitere Darstellungsform Boolescher Funktionen vorgestellt werden, die in den letzten Jahren sehr breite Verwendung gefunden hat. Es handelt sich hierbei um BDDs [Aks78, Lee59], die im Gegensatz zu Booleschen Polynomen eine mehrstufige Beschreibung einer Funktion ermöglichen. Da die Ergebnisse der vorliegenden Arbeit auch auf die Minimierung von BDDs übertragen werden können, soll diese Datenstruktur ausführlicher untersucht werden.

BDDs basieren auf der **Shannon'schen Zerlegung** Boolescher Funktionen, die folgendermaßen erklärt ist:

$\forall(x_1,\dots,x_n) \in \{0,1\}^n : f(x_1,\dots,x_n) =$

$$x_i \wedge f(x_1,\dots,x_{i-1},1,x_{i+1},\dots,x_n) \vee \overline{x_i} \wedge f(x_1,\dots,x_{i-1},0,x_{i+1},\dots,x_n).$$

Man kann daher eine Funktion in zwei Teilfunktionen zerlegen, die offensichtlich weniger Variablen besitzen als die Ausgangsfunktion. Diese Teilfunktionen aus $B_{n-1}$ werden auch als *Kofaktoren* bezeichnet. Sie sind offensichtlich eindeutig bestimmt. Das Verfahren kann für die beiden Kofaktoren rekursiv fortgesetzt werden, bis nur noch Kofaktoren vorliegen, die Konstanten darstellen. Den sich daraus ergebenden Rekursionsbaum bezeichnet man als Binary Decision Diagram (BDD).

Zusätzlich kann man alle Variablen in eine bestimmte Ordnung bringen:

**Bezeichnung 5** *Eine Menge $X$ ($\mid X \mid = n$), auf der eine lineare Ordnung $<_\pi$ existiert, wird mit $X_\pi$ bezeichnet. $\pi : \{1,\dots,n\} \to X$ ist dabei ein Ordnungsvektor, wobei $\pi[i] = x$ genau dann gilt, falls $i-1$ verschiedene Variablen in $X$ existieren, die kleiner als $x$ (bzgl. $<_\pi$) sind.*

Bezieht man die Ordnung auf der Variablenmenge in die Struktur eines BDDs ein, so erhält man geordnete BDDs [Bry86]:

**Definition 1.29 (Ordered Binary Decision Diagram)** *Sei $X_\pi$ eine geordnete Variablenmenge. Ein* **Ordered Binary Decision Diagram** *(OBDD) über der Menge $X_\pi$ ist ein Tupel $F = (G, index, value, high, low, out)$ mit*

- *$G = (V,E)$ ist ein gerichteter, azyklischer, binärer Graph, wobei*
  - * $V = V_N \cup V_T$.
    $V_N$ ist die Menge der **Nichtterminale**.
    $V_T = \{t_0, t_1\}$ ist die Menge der **Terminale**.
  - * $V_N \cap V_T = \emptyset$.
  - * $E \subseteq (V_N \times V_N) \cup (V_N \times V_T)$.
- *$index : V_N \to X$ ordnet jedem Nichtterminal eine Variable aus $X$ zu.*
- *$value : V_T \to \{0,1\}$ ordnet jedem der beiden Terminale einen Booleschen Wert zu.*
- *$high : V_N \to V$ ordnet jedem Nichtterminal den high Nachfolger zu. Dabei gilt $(v, high(v)) \in E$.*

- *$low : V_N \to V$ ordnet jedem Nichtterminal den low Nachfolger zu. Dabei gilt $(v, low(v)) \in E$.*
- *$out : \{1, \ldots, m\} \to \{v \in V \mid indeg(v) = 0\}$ ist eine bijektive Abbildung, welche alle Wurzeln in $G$ numeriert.*

*Zusätzlich gilt für alle $v \in V_N$:*

$$\begin{aligned} index(v) &<_\pi index(high(v)), \text{falls } high(v) \in V_N \text{ und} \\ index(v) &<_\pi index(low(v)), \text{ falls } low(v) \in V_N. \end{aligned}$$

Dies bedeutet also, daß auf jedem gerichteten Pfad in einem OBDD von einer Wurzel zu einem Terminalknoten die Variablen geordnet auf diesem Pfad vorkommen, d.h. ist $(v_1, \ldots, v_l)$ ein Pfad im OBDD, so gilt stets $index(v_1) <_\pi index(v_2) <_\pi \ldots <_\pi index(v_l)$.
Daher kann auf einem Pfad im OBDD jede Variable *höchstens einmal* auftreten.

**Bezeichnung 6** *Sei $F = (G = (V, E), index, value, high, low, out)$ ein OBDD und $v$ ein Knoten in $F$.*

- *$v \in F \;:\Longleftrightarrow\; v \in V$.*
- *$F_v := (G = (T, E_T), index|_T, value|_T, high|_T, low|_T, out')$ mit*
  *$T = \{w \in F \mid$ es existiert ein Pfad in $G$ von $v$ nach $w\}$ und*
  *$E_T = \{(v_1, v_2) \in E \mid v_1, v_2 \in T\}$,*
  *$out' : \{1\} \to \{v\}$.*
  *$F_v$ heißt* **Unter-OBDD von** *$F$.*
- *Eine Variable $x \in X$ ist in einem OBDD $F$ enthalten genau dann, falls ein Nichtterminal $v$ in $F$ existiert mit $index(v) = x$.*
  *Die Menge der Variablen aus $X$, die in $F$ enthalten sind, bezeichnet man als* **Abhängigkeitsmenge** *$Abh(F)$ von $F$.*

**Bezeichnung 7** *Unter der Größe eines OBDDs $F$ ($|F|$) versteht man die Anzahl der Knoten in $F$, d.h. $|F| := |\{v \in F\}|$.*

Um mittels OBDDs Boolesche Funktionen darzustellen, muß zunächst jedem OBDD eine Funktion zugeordnet werden:

**Definition 1.30 (Durch einen OBDD dargestellte Funktion)**
*Sei $F = (G, index, value, high, low, out)$ ein OBDD über der Menge $X_\pi = \{x_1, \ldots, x_n\}$ und $v \in F$. Die Boolesche Funktion $f_v \in B_n$ heißt* **die vom Knoten $v$ dargestellte Boolesche Funktion** *genau dann, falls gilt:*

- *$v \in V_T$ und $value(v) = 1 \Rightarrow f_v = \mathbf{1}$,*
- *$v \in V_T$ und $value(v) = 0 \Rightarrow f_v = \mathbf{0}$,*

- $v \in V_N$ *und* $index(v) = x_i \Rightarrow f_v = x_i \wedge f_{high(v)} \vee \overline{x_i} \wedge f_{low(v)}$,

*wobei* $f_{high(v)}$ *und* $f_{low(v)}$ *die durch die Knoten* $high(v)$ *und* $low(v)$ *dargestellten Funktionen sind.*
*Besitzt* $F$ $m$ *Wurzelknoten*†*, so ist*

$$f_F := (f_{out(1)}, \ldots, f_{out(m)})$$

*die* **durch den OBDD $F$ dargestellte Boolesche Funktion.**

Die obige Definition legt einen Algorithmus nahe, um für einen OBDD $F$ und einen Knoten $v$ mittels einem bottom-up Durchlauf des Graphen die Funktion $f_{F_v}$ zu berechnen.

Es folgt nun ein Beispiel, um die Datenstruktur ein wenig zu verdeutlichen.

**Beispiel**

Die Realisierung eines Volladdierers ergibt sich aus der Funktion

$$FA: \ \{0,1\}^3 \ \rightarrow \ \{0,1\}^2 \, , (x_1, x_2, x_3) \longmapsto (c, s) \text{ mit}$$

$$x_1 + x_2 + x_3 = 2c + s$$

Dabei ist $s$ das Summenbit und $c$ das Übertragsbit des Ergebnisses. Es ist

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

Die Variablen sollen gemäß ihren Indizes geordnet sein. Dann sind die beiden linken OBDDs in Abbildung 1.3 verschiedene Darstellungen für die Funktion des Übertragbits und der rechte OBDD stellt die Funktion des Summenbits dar.‡

Wie dieses Beispiel des Volladdierers zeigt, gibt es bei der Darstellung einer Booleschen Funktion durch OBDDs keine Eindeutigkeit, auch dann nicht, wenn die Variablenordnung fest bleibt. Eine solche Eindeutigkeit ist aber in vielen Fällen wünschenswert. Will man z.B. vergleichen, ob zwei Darstellungen äquivalent sind, d.h. die gleiche Funktion repräsentieren, so kann dieser Vergleich im Falle von eindeutigen Darstellungen durch einen einfachen Isomorphietest der beiden Graphen ermittelt werden.
Durch die Einführung von reduzierten OBDDs (ROBDDs) [Bry86] erreicht man gerade diese Eindeutigkeit. Die Reduzierung basiert dabei auf den folgenden Beobachtungen:

**Bemerkung 5**
*Die folgenden Manipulationen führen stets zu äquivalenten Darstellungen:*

---

†Da ein OBDD ein azyklischer Graph ist, besitzt $F$ mindestens eine Wurzel.

‡Falls die Kanten eines OBDDs nicht entsprechend beschriftet sind, so führt die linke ausgehende Kante eines Knotens $v$ zu $low(v)$ und die rechte Kante zu $high(v)$.

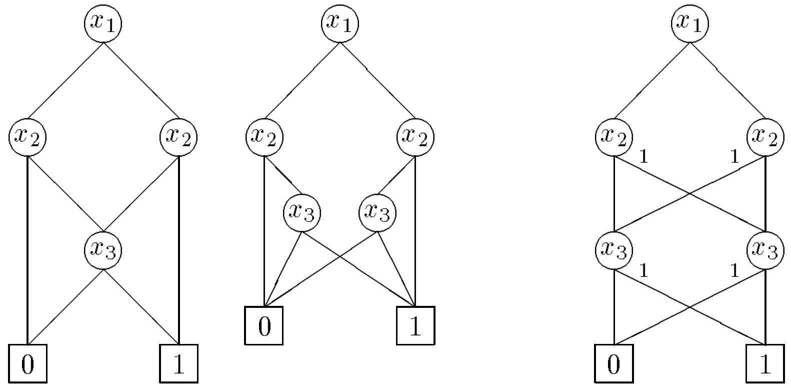



Abbildung 1.3: OBDDs der Teilfunktionen des Volladdierers

- *Ist $v$ ein Knoten in einem* OBDD *$F$ und gilt $high(v) = low(v)$, so kann man $v$ aus $F$ streichen und alle Kanten, die zu $v$ führen, nach $high(v)$ umleiten.*

- *Sind $v_1, v_2$ zwei verschiedene Knoten im* OBDD *$F$, welche die gleiche Funktion repräsentieren, also $f_{v_1} = f_{v_2}$. Dann kann $v_1$ aus $F$ entfernt werden, und alle Kanten, die zu $v_1$ führen, nach $v_2$ umgeleitet werden.*

Erfreulicherweise wird die Größe des OBDDs durch diese Umformungen stets vermindert. Man kann daher diese beiden Eigenschaften ausnutzen, um einen OBDD in seiner Größe zu reduzieren, *ohne* dabei die dargestellte Funktion zu verändern. Dies führt zu reduzierten OBDDs:

**Definition 1.31 (Reduced Ordered Binary Decision Diagrams)**
*Sei $F = (G, index, value, high, low, out)$ ein* OBDD *über $X_\pi$. $F$ heißt* **Reduced Ordered Binary Decision Diagram** (ROBDD)*über $X_\pi$ genau dann, falls gilt:*

- *Es gibt keinen Knoten $v \in F$ mit $high(v) = low(v)$.*

- *Für zwei Knoten $v_1, v_2 \in F$ mit $v_1 \neq v_2$ gilt $f_{v_1} \neq f_{v_2}$.*

ROBDDs bilden eine eindeutige Darstellung Boolescher Funktionen [Bry86]:

**Satz 1.2 (Bryant '86)** *Seien $F$ und $G$ zwei* ROBDD*s über $X_\pi$. Dann gilt:*

$$f_F \neq f_G \iff F \text{ und } G \text{ sind nicht isomorph.}$$

Zusätzlich sind ROBDDs sogar minimale Darstellungen bzgl. einer festen Variablenordnung:

**Lemma 1.5** *Sei $F$ ein* ROBDD *über $X_\pi$ und $f_F$ die durch $F$ dargestellte Funktion. Dann gilt:*

$$|F| = min\{|G| \mid G \text{ ist OBDD-Darstellung über } X_\pi \text{ für } f_F\}.$$

**Beweis:** Da die Darstellung durch ROBDDs eindeutig ist und durch die Reduktion der entsprechende OBDD in seiner Größe stets vermindert wird, kann es keinen kleineren OBDD für $f_F$ als $F$ geben.

□

Die $exor_n$ Funktion liefert einen schönen Vergleich zwischen der zweistufigen Darstellung durch Minimalpolynome und der mehrstufigen ROBDD-Darstellung. Die Funktion ist wie folgt definiert:

$$exor_n(x_1, \ldots, x_n) := x_1 \oplus x_2 \oplus \ldots \oplus x_n.$$

**Lemma 1.6** *Das Minimalpolynom $p$ mit $f_p = exor_n$ besitzt Kosten $C(p) = n2^{n-1} - 1$.*

**Beweis:** Man kann sich leicht überlegen, daß

$$p = \bigvee_{(\sum_{i=1}^{n} \epsilon_i) \; mod \, 2 = 1} x_1^{\epsilon_1} \wedge \ldots \wedge x_n^{\epsilon_n}.$$

das Minmalpolynom für $exor_n$ ist.
Die Kosten dieses Polynoms betragen

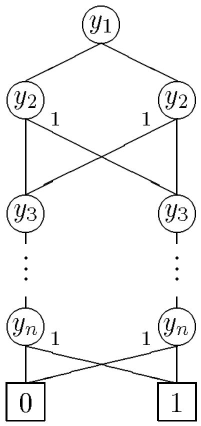
$$C_p = \underbrace{(n-1)}_{\text{Kosten eines Monoms}} \cdot 2^{n-1} + \underbrace{2^{n-1} - 1}_{\vee \; Gatter} = n2^{n-1} - 1$$

□

Eine Darstellung für $exor_n$ durch ROBDDs ist aber in jedem Fall kleiner:

**Lemma 1.7** *Für jeden ROBDD $F$ über $X_\pi$ mit $f_F = exor_n$ ist $|F| = 2n + 1$.*

**Beweis:** Es ist leicht einzusehen, daß der folgende ROBDD $F$ eine Darstellung für $exor_n$ für die Variablenordnung $<_\pi$ ist. Dabei sei $y_i = \pi[i] (1 \leq i \leq n)$.



$F$ besitzt genau $2n + 1$ Knoten.

□

Es zeigt sich, daß die Darstellung Boolescher Funktionen durch ROBDDs für viele in der Praxis auftretende Funktionen zu relativ kompakten ROBDDs führt. Es stellt sich natürlich sofort die Frage, ob man zu *jeder* Booleschen Funktion eine Variablenordnung angeben kann, so daß der entsprechende ROBDD, der die Funktion darstellt, eine polynomielle Größe besitzt. Diese Frage muß leider negativ beantwortet werden: Es gibt Funktionen, für die jeder ROBDD eine exponentielle Größe (bzgl. der Variablenanzahl) besitzt, unabhängig von der Variablenordnung. Die Multiplikation zweier binärer Zahlen ist eine solche Boolesche Funktion [Bry86]. Weitere Beispiele finden sich auch in [Bry91].
In Abschnitt 1.6 wird gezeigt, daß sogar fast alle Booleschen Funktionen einen minimalen ROBDD mit exponentieller Größe besitzen.

In praktischen Implementationen für ROBDDs (z.B. [BRB90]), die neben dieser Datenstruktur auch noch Algorithmen für die wichtigsten Booleschen Operationen anbieten, gibt es oft eine zusätzliche Erweiterungen dieser Struktur, mit der sich Speicherplatz- und erhebliche Laufzeitgewinne erzielen lassen[†].

Wie bei jeder anderen Darstellungsform, so muß es auch bei ROBDDs möglich sein, aus vorgegebenen Funktionen neue Funktionen bzw. neue Darstellungen der Funktionen durch Anwendung Boolescher Operationen zu generieren.
Sind die Booleschen Funktionen durch ROBDDs gegeben, so zeichnet sich für diese Aufgabe die *ite*−Operation ( *if-then-else*) besonders aus. Man kann mit Hilfe dieser Operation (und den beiden Konstantenfunktionen) alle anderen Booleschen Operationen realisieren, welche auf den Funktionen aus $B_1 \cup B_2$ basieren. Dies verdeutlicht die folgende Tabelle:

| Operation | *ite*-Operation | Operation | *ite*-Operation |
|---|---|---|---|
| $\mathbf{0}$ | $ite(\mathbf{0},\mathbf{0},\mathbf{1})$ | $\mathbf{1}$ | $ite(\mathbf{1},\mathbf{1},\mathbf{1})$ |
| $F \wedge G$ | $ite(G,\mathbf{0},F)$ | $F \wedge \overline{G}$ | $ite(\overline{G},\mathbf{0},F)$ |
| $F$ | $ite(F,\mathbf{0},\mathbf{1})$ | $\overline{F} \wedge G$ | $ite(\mathbf{0},G,F)$ |
| $G$ | $ite(G,\mathbf{0},\mathbf{1})$ | $F \oplus G$ | $ite(\overline{G},G,F)$ |
| $F \vee G$ | $ite(\mathbf{1},G,F)$ | $\overline{F \vee G}$ | $ite(\mathbf{0},\overline{G},F)$ |
| $\overline{F \oplus G}$ | $ite(G,\overline{G},F)$ | $\overline{G}$ | $ite(\mathbf{0},\mathbf{1},G)$ |
| $F \vee \overline{G}$ | $ite(\mathbf{1},\overline{G},F)$ | $\overline{F}$ | $ite(\mathbf{0},\mathbf{1},F)$ |
| $\overline{F} \vee G$ | $ite(G,\mathbf{1},F)$ | $\overline{F \wedge G}$ | $ite(\overline{G},\mathbf{1},F)$ |

[†]So wird beim Konzept der *complement edges* die Eigenschaft der ROBDDs genutzt, daß eine Invertierung der dargestellten Funktion lediglich die Vertauschung der beiden Terminalknoten bewirkt. Das Komplement der Funktion, die durch einen Knoten $v$ dargestellt wird, kann daher durch den gleichen Knoten repräsentiert werden, allerdings wird die zu $v$ inzidente Kante als complement edge entsprechend markiert. Um eine kanonische Form der Darstellung zu gewährleisten, darf die Kante zu einem *high*−Nachfolger nicht markiert werden.
Des weiteren wird auch oftmals Hashing verwendet, um Knoten effizient zu speichern. Experimentelle Untersuchungen [BRB90] haben gezeigt, daß der Speicherplatz und die Laufzeit der Algorithmen durch Einführung von complement edges reduziert wird. So kann die Zeit für den Aufbau eines ROBDDs in einigen Fällen sogar halbiert werden.

Die $ite(F,G,H)-$Operation kann in Zeit $O(|F| \cdot |G| \cdot |H|)$ ausgeführt werden [BRB90]. Liegen die Funktionen $F, G, H$ in Form von ROBDDs vor, so besitzt der ROBDD zu $ite(F,G,H)$ ebenfalls höchstens $|F| \cdot |G| \cdot |H|$ Knoten. Da in der obigen Tabelle bei jeder $ite-$Operation mindestens einer der Operanden eine Konstante darstellt und deren ROBDDs lediglich aus dem entsprechenden Terminalknoten bestehen, gilt folgender Satz [Bry86]:

**Satz 1.3 (Bryant '86)** *Sind $F, G$ zwei* ROBDD*s über der Variablenmenge $X_\pi$, dann besitzt jeder* ROBDD *$H$ über $X_\pi$, der die Boolesche Funktion $f_H = f_F \; op \; f_G$ (bzw. $f_H = op \; f_F$) mit $f_{op} \in B_2 \cup B_1$ darstellt, eine Größe von $O(|F| \cdot |G|)$ (bzw. $O(|F|)$) und kann in Zeit $O(|F| \cdot |G|)$ (bzw. $O(|F|)$) berechnet werden.*

Im Gegensatz zur Darstellung durch Boolesche Polynome verhalten sich ROBDDs also in gewissem Sinne „gutartig“ bei der Anwendung Boolescher Operationen.

#### 1.5.3.1 Spezielle Synthese von Schaltkreisen aus ROBDD Darstellungen

Spezielle Darstellungsformen für eine Boolesche Funktion $f$ lassen oftmals besondere Algorithmen zur Logiksynthese zu (vgl. PLA-Realisierung aus Polynomen). Auch ROBDDs lassen sich einfach in einen (mehrstufigen) Schaltkreis umformen. Eine mögliche Realisierung besteht darin, alle Knoten des Graphen durch entsprechende *ite*-Gatter zu ersetzen. Das Beispiel in Abbildung 1.4 zeigt das Ergebnis einer solchen Konstruktion.
Es gibt viele Verfahren, die ausgehend von einem ROBDD, diesen in ein FPGA (Field Programmable Gate Array) überführen und dabei die besonderen Eigenschaften der ROBDD-Struktur nutzen (z.B. [CM92, MNS+90]). FPGA's sind ähnlich wie PLA's bereits vorgefertigte Chips, deren Funktion vom Anwender spezifiziert werden kann. Sie bestehen aus einem zweidimensionalen Feld von gleichartigen Blöcken und Verbindungsleitungen zwischen diesen Blöcken, deren Leitungsverlauf auch programmierbar ist.
Den einzelnen Blöcken können vom Anwender verschiedene Funktionen zugeordnet werden. Die Komplexität der FPGA-Realisierung wird meist entscheidend von der Größe der ROBDD-Darstellung bestimmt.

Aus der unteren Abbildung auf Seite 15 geht hervor, daß jedes $ite-$Gatter durch 3 Gatter† aus der $B_2-$Zellenbibliothek ersetzt werden kann. Demnach gilt:

**Satz 1.4** *Sei $f \in B_{n,m}$ eine Boolesche Funktion und $F$ ein* ROBDD *mit $f = f_F$. Dann gilt:*

$$C_{B_2}(f) \leq 3 \cdot |F|.$$

#### 1.5.3.2 Die Struktur von ROBDDs

Die Struktur von ROBDDs soll in diesem Abschnitt nun näher untersucht werden. Dabei ist man besonders daran interessiert, zu einer Booleschen Funktion die

†Der Inverter und das nachfolgende *and*$-$Gatter können zu *einem* $B_2-$Gatter zusammengefaßt werden.

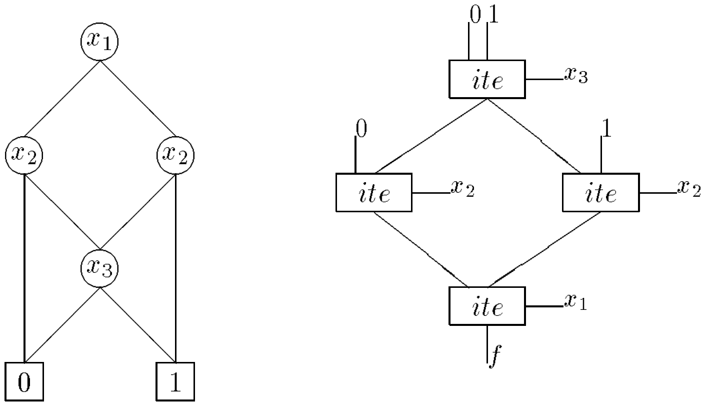



Abbildung 1.4: Beispiel für die Realisierung eines ROBDDs.

Anzahl der Knoten des entsprechenden ROBDDs zu bestimmen.
Jeder Pfad in einem ROBDD $F$, der von einer Wurzel ausgeht, führt zu einem Knoten $v \in F$ und damit zu der Booleschen Funktion, die vom Unter-ROBDD $F_v$ dargestellt wird.
Da es bei jedem Knoten nur zwei alternative Fortsetzungen für einen Pfad gibt, kann man anstelle der im Pfad durchlaufenen Knoten auch einen binären Vektor angeben, der diesen Pfad eindeutig kennzeichnet:

**Definition 1.32 ($\epsilon$-Pfad, $\epsilon$-Knoten)** *Sei $F$ ein ROBDD über $X_\pi$ mit der einzigen Wurzel $w$. Sei $\epsilon = (\epsilon_1, \ldots, \epsilon_l) \in \{0,1\}^l$. Die Folge $(v_1, \ldots, v_l)$ heißt* **$\epsilon$-Pfad** *in $F$ genau dann, falls gilt:*

$$\begin{aligned} v_1 &= w, \\ v_{i+1} &= high(v_i), \text{ falls } index(v_i) = \pi[j] \text{ und } \epsilon_j = 1, \\ v_{i+1} &= low(v_i), \text{ falls } index(v_i) = \pi[j] \text{ und } \epsilon_j = 0,\ 1 \leq i \leq l-1. \end{aligned}$$

*$v_l$ heißt* **$\epsilon$-Knoten** *von $F$.*

**Bemerkung 6**

- *Ein $\epsilon$-Knoten $v \in F$ repräsentiert wiederum einen ROBDD, nämlich $F_v$. Die Abhängigkeitsmenge von $F_v$ ist allerdings gegenüber dem Ursprungsgraphen $F$ kleiner geworden, da die Variablen, die sich auf dem $\epsilon$-Pfad befinden, in $F_v$ nicht mehr vorkommen können. Es gilt demnach $Abh(F_v) \subseteq Abh(F) \backslash \{x_i \in X \mid x_i <_\pi index(v)\}$.*

- *$\epsilon$-Knoten sind eindeutig bestimmt. Zu jedem $\epsilon$ existiert ein entsprechender Knoten in $F$.*

- *Zu jedem Knoten eines ROBDDs gibt es mindestens einen $\epsilon$-Pfad. Verschiedene $\epsilon$-Pfade können zu gleichen Knoten in einem ROBDD führen.*

- *Für $|X_\pi| = n$ führt ein $\epsilon$-Pfad der Form $\epsilon = (\epsilon_1, \ldots, \epsilon_n)$ immer zu einem Terminalknoten.*

Zu Beginn dieses Abschnitts wurde schon der wichtige Begriff des Kofaktors einer Booleschen Funktion erwähnt. Es soll im folgenden ein Zusammenhang zwischen den $\epsilon-$Pfaden in einem ROBDD und Kofaktoren einer Funktion hergeleitet werden. Dazu müssen die Kofaktoren formal definiert werden:

**Definition 1.33 (Kofaktor einer Booleschen Funktion)** *Sei $f \in BP_n$ eine Boolesche Funktion und $X = \{x_1, \ldots, x_n\}$ eine Variablenmenge. Die Boolesche Funktion*
$f_{x_{\varepsilon_1}^{\gamma_1}, \ldots, x_{\varepsilon_k}^{\gamma_k}}(x_1, \ldots, x_n) :=$

$$f(x_1, \ldots, x_{\varepsilon_1 - 1}, \gamma_1, x_{\varepsilon_1 + 1}, \ldots, x_{\varepsilon_k - 1}, \gamma_k, x_{\varepsilon_k + 1}, \ldots, x_n) \in BP_{n-k}$$

*mit $k \leq n$ , $\gamma_i \in \{0,1\}$ , $1 \leq i \leq k$,*
*heißt* **Kofaktor von** *$f$ bzgl. $x_{\varepsilon_1} = \gamma_1, \ldots, x_{\varepsilon_k} = \gamma_k$.*

Ein Kofaktor setzt also die Belegung für bestimmte Variablen auf einen Booleschen Wert fest.

**Bemerkung 7**
*Man schreibt auch $f_{\overline{x_i}}$ für den Kofaktor $f_{x_i^0}$ und $f_{x_i}$ für den Kofaktor $f_{x_i^1}$.*

Mit Hilfe dieser Bezeichnungen kann die Shannon'sche Zerlegung umgeschrieben werden in die Form

$$\forall (x_1, \ldots, x_n) \in \{0,1\}^n :$$
$$f(x_1, \ldots, x_n) = x_i \wedge f_{x_i} \vee \overline{x_i} \wedge f_{\overline{x_i}}.$$

$\epsilon$-Knoten eines ROBDDs $F$ repräsentieren demnach Kofaktoren von $f_F$, die aber spezieller Natur sind, da in einem $\epsilon$-Pfad nicht beliebige Variablen auf einen Booleschen Wert festgesetzt werden können, sondern in einem Pfad der Länge $k$ nur die $k-1$ kleinsten Variablen in der Variablenordnung $<_\pi$. Es gilt:

**Lemma 1.8** *Sei $F$ ein ROBDD über $X_\pi = \{x_1, \ldots, x_n\}$ und $\epsilon = (\epsilon_1, \ldots, \epsilon_k) \in \{0,1\}^k$ , $k \leq n$. Besitzt $F$ nur eine Wurzel und ist $v$ der $\epsilon$-Knoten in $F$, dann gilt:*

$$f_v = (f_F)_{\pi[1]^{\epsilon_1}, \ldots, \pi[k]^{\epsilon_k}}.$$

**Beweis:** Der Beweis wird durch Induktion über die Länge des $\epsilon-$Pfades geführt.
Induktionsanfang: $k = 0$.
Trivial, da in diesem Fall $v$ die Wurzel von $F$ ist.
Induktionsschritt: $k \to k+1$.
Sei $\epsilon' = (\epsilon_1, \ldots, \epsilon_k)$ und $v'$ der $\epsilon'-$Knoten in $F$, dann gilt nach Induktionsvoraussetzung $f_{v'} = (f_F)_{\pi[1]^{\epsilon_1}, \ldots, \pi[k]^{\epsilon_k}}$.
Sei $v$ der $(\epsilon_1, \ldots, \epsilon_{k+1})-$Knoten in $F$.
Man unterscheidet zwei Fälle:
<u>Fall 1:</u> $index(v') = \pi[k+1]$
Aufgrund von Definition 1.30 und 1.32 ist $f_v = (f_{v'})_{\pi[k+1]^{\epsilon_{k+1}}}$, da die Kofaktoren der Shannon'schen Zerlegung eindeutig bestimmt sind.

Aus der Definition der Kofaktoren folgt $f_v = (f_F)_{\pi[1]^{\epsilon_1},\ldots,\pi[k+1]^{\epsilon_{k+1}}}$.
Fall 2: $index(v') > \pi[k+1]$
Dann ist $v' = v$. Somit gilt:

$$f_{v'} = \pi[k+1] \wedge f_v \vee \overline{\pi[k+1]} \wedge f_v.$$

Offensichtlich ist $f_{v'} \in B_{n-k}$ und $f_v \in B_{n-k-1}$.
Da die Kofaktoren der Shannon'schen Zerlegung eindeutig bestimmt sind, gilt $f_v = (f_{v'})_{\pi[k+1]^{\epsilon_{k+1}}}$ und somit auch $f_v = (f_F)_{\pi[1]^{\epsilon_1},\ldots,\pi[k+1]^{\epsilon_{k+1}}}$.

□

Alle Kofaktoren einer Booleschen Funktion werden zu einer Menge zusammengefaßt:

**Definition 1.34 (Kofaktormenge)** *Sei $f \in BP_n$, $X = \{x_1, \ldots, x_n\}$ und $\lambda = \{x_{i_1}, \ldots, x_{i_k}\} \subseteq X$.*

$$Kof_\lambda(f) := \{f_{x_{i_1}^{\epsilon_1},\ldots,x_{i_k}^{\epsilon_k}} \mid (\epsilon_1, \ldots, \epsilon_k) \in \{0,1\}^k\}$$

*heißt* **Kofaktormenge von** $f$ *bezüglich* $\lambda$.

Bevor eine untere Schranke für die Größe eines ROBDDs angegeben wird, soll zunächst die Abhängigkeitsmenge genauer charakterisiert werden.

**Definition 1.35 (Abhängige, unabhängige Variablen)** *Sei $f \in BP_n$ eine Boolesche Funktion und $X = \{x_1, \ldots, x_n\}$ eine Menge von Variablen.*
*$f$ heißt* **unabhängig von** $x_i \in X$ *genau dann, wenn*

$$f_{x_i} = f_{\overline{x_i}}.$$

*Ansonsten heißt $f$* **abhängig von** $x_i$.

Wählt man für die beiden Kofaktoren eine andere Schreibweise, so erhält man:

**Korollar 1.5** *Eine Boolesche Funktion $f \in BP_n$ ist unabhängig von der Variablen $x_i$ genau dann, falls*

$$f(\epsilon_1, \ldots, \epsilon_i, \ldots, \epsilon_n) = f(\epsilon_1, \ldots, \overline{\epsilon_i}, \ldots, \epsilon_n)\ \forall(\epsilon_1, \ldots, \epsilon_n) \in \{0,1\}^n.$$

In einem ROBDD sind alle Knoten stets mit abhängigen Variablen markiert:

**Lemma 1.9** *Sei $f \in B_n$ eine Boolesche Funktion und $F$ ein ROBDD über $X_\pi = \{x_1, \ldots, x_n\}$ mit $f_F = f$. Ist $x_i \in X_\pi$, dann gilt:*

$$f \text{ ist abhängig von } x_i \iff x_i \in Abh(F).$$

**Beweis:** „$\Longrightarrow$“
Ist $f \in B_n$ abhängig von $x_i$, so gilt nach Definition 1.35 $f_{x_i} \neq f_{\overline{x_i}}$. Damit gibt es zwei Funktionsstellen $\epsilon^1 = ({\epsilon^1}_1, \ldots, {\epsilon^1}_n)$, $\epsilon^2 = ({\epsilon^2}_1, \ldots, {\epsilon^2}_n) \in \{0,1\}^n$ mit

$$\begin{array}{rcll} \epsilon^1_j & = & \epsilon^2_j & \text{für } j \neq i \text{ und} \\ \epsilon^1_i & \neq & \epsilon^2_i & \text{und} \\ f(\epsilon^1) & \neq & f(\epsilon^2). & \quad (*) \end{array}$$

Sei $x_i \notin Abh(F)$. Dann gibt es keinen Knoten in $F$, der mit $x_i$ markiert ist. Man betrachte die beiden $\epsilon-$Pfade, welche durch
$(\epsilon^1_{\pi^{-1}[x_1]}, \ldots, \epsilon^1_{\pi^{-1}[x_n]})$ und $(\epsilon^2_{\pi^{-1}[x_1]}, \ldots, \epsilon^2_{\pi^{-1}[x_n]})$ bestimmt werden. Da $x_i$ nicht in $F$ vorkommt, beschreiben beide $\epsilon-$Pfade denselben Pfad in $F$, beginnend mit der Wurzel bis zu einem Terminal. Nach Lemma 1.8 widerspricht dies aber der Annahme (*).
„$\Longleftarrow$“
Ist $x_i \in Abh(F)$, so gibt es einen Knoten $v \in F$ mit $index(v) = x_i$. Dieser Knoten ist durch mindestens einen $\epsilon$-Pfad mittels $\epsilon^1$ zu erreichen. Da $F$ reduziert ist, stellen die beiden Unter-ROBDDs $f_{high(v)}$ und $f_{low(v)}$ verschiedene Funktionen dar. Es gibt daher ein $\epsilon^2$, so daß die entsprechenden $\epsilon-$Pfade in $F_{high(v)}$ bzw. $F_{low(v)}$ zu unterschiedlichen Terminalen führen.
Bildet man nun zum einen den Pfad, der ausgehend von der Wurzel in $F$ zuerst mittels $\epsilon^1$ zu $v$ führt und dann über $high(v)$ mittels $\epsilon^2$ zu einem Terminal, und zum anderen den Pfad, der über $\epsilon^1$ zu $v$ und dann über $low(v)$ mittels $\epsilon^2$ zu einem anderen Terminal führt, so entsprechen den beiden Pfaden nach Lemma 1.8 zwei Funktionsstellen $x^1, x^2 \in \{0,1\}^n$, die sich nur in der $i$. Komponente unterscheiden. Damit gilt $f(x^1) \neq f(x^2)$ und demnach auch $f_{x_i} \neq f_{\overline{x_i}}$, d.h. $f$ ist abhängig von $x_i$.

□

Man erhält somit folgende untere Schranke:

**Korollar 1.6** *Für jeden* ROBDD $F$ *über* $X_\pi$ *gilt:*

$$|F| \geq |\{x_i \in X_\pi \mid f_F \text{ ist abhängig von } x_i\}|.$$

Um eine obere Schranke zu bestimmen, schätzt man zunächst die Anzahl der Knoten bzgl. eines festen Index ab.

**Bezeichnung 8** *Für einen* ROBDD $F = (G, index, value, high, low, out)$ *über* $X_\pi$ *ist*

$$Ind_F(i) := \{v \in F \mid index(v) = \pi[i]\}.$$

**Lemma 1.10** *Sei* $X = \{x_1, \ldots, x_n\}$ *eine Variablenmenge. Es gibt genau* $2^{2^n} - 2^{2^{n-1}}$ *totale Boolesche Funktionen aus* $B_n$*, die von* $x_i \in X$ *abhängen.*

**Beweis:** Es gibt genau $2^{2^n}$ Boolesche Funktionen in $B_n$. Die Funktionen aus $B_n$, welche nicht von $x_i$ abhängen, sind offensichtlich alle Funktionen über

$X \backslash \{x_i\}$. Damit ergibt sich die Behauptung.

□

Damit lassen sich die Mengen $Ind_F(i)$ in ihrer Mächtigkeit abschätzen:

**Lemma 1.11** *Sei $F$ ein* ROBDD *über $X_\pi = \{x_1, \ldots, x_n\}$. Besitzt $F$ genau eine Wurzel, dann gilt:*

$$Ind_F(i) \leq min\{2^{i-1}, 2^{2^{n-i+1}} - 2^{2^{n-i}}\} \qquad (1 \leq i \leq n).$$

**Beweis:** Die Ungleichung $Ind_F(i) \leq 2^{i-1}$ folgt unmittelbar aus der Tatsache, daß jeder Knoten aus $Ind_F(i)$ höchstens zwei Nachfolger aus $Ind_F(i+1)$ besitzen kann und $F$ nur eine Wurzel besitzt.
Da in einem ROBDD alle Knoten einer Menge $Ind_F(i)$ verschiedene Funktionen berechnen, und zwar solche aus $B_{n-i+1}$, die von $\pi[i]$ abhängen, kann es nach Lemma 1.10 höchstens $2^{2^{n-i+1}} - 2^{2^{n-i}}$ solcher Knoten in $F$ geben.

□

Die Größe eines ROBDDs ergibt sich somit aus der Summe $\sum_{i=1}^{n} Ind_F(i)$.
Da die Funktion $2^{i-1}$ streng monoton wachsend und $2^{2^{n-i+1}} - 2^{2^{n-i}}$ streng monoton fallend mit $i$ ist, existiert ein Index $k$ $(1 \leq k \leq n)$ mit der Eigenschaft, daß die maximal mögliche Anzahl der Knoten in $Ind_F(i)$ für $i < k$ mit $i$ zunimmt und für $i > k$ abnimmt, d.h. der ROBDD mit der maximalen Anzahl von Knoten wird mit wachsendem Index $i$ in seiner Breite zunächst zunehmen und nach dem „Wendepunkt" $k$ wiederum stetig schmaler werden.
Mit Hilfe dieser Überlegungen können weitere obere Schranken angegeben werden, die schärfer sind als in Lemma 1.11 (siehe z.B. [HM94]).

#### 1.5.3.3 Einfluß der Variablenordnung auf die Größe eines ROBDDs

Die Eindeutigkeit der ROBDD Darstellung (Satz 1.2) sowie die Minimalität unter allen OBDDs (Lemma 1.5) gilt immer nur unter der Voraussetzung, daß man für alle ROBDDs eine *gemeinsame* Variablenordnung $<_\pi$ festgelegt hat. Ändert man allerdings diese Ordnung, so verändert sich im allgemeinen auch der ROBDD, die dargestellte Boolesche Funktion bleibt natürlich dieselbe.
Wie sehr die Wahl der Variablenordnung die Größe eines ROBDD beeinflussen kann, soll folgendes Beispiel zeigen:

Man betrachte die Funktionenfamilie $double_n \in B_{2n}$ mit

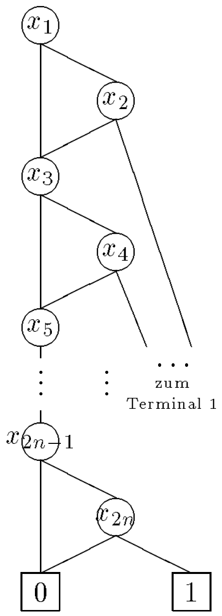
$$double_n := (x_1 \wedge x_2) \vee (x_3 \wedge x_4) \vee \ \ldots \ \vee (x_{2n-1} \wedge x_{2n}).$$

Dann gilt:

**Lemma 1.12** *Sei $X_\pi = \{x_1, \ldots, x_{2n}\}$ eine geordnete Variablenmenge mit $\pi[i] = x_i$ $(1 \leq i \leq 2n)$ und $F$ ein* ROBDD *über $X_\pi$. Ist $f_F = double_n$, dann gilt für die Größe von $F$:*

$$|F| = 2n + 2.$$

**Beweis:** Der ROBDD, welcher die Funktionen $double_n$ repräsentiert, besitzt genau $2n+2$ Knoten:




□

Wenn man die Ordnung auf $X$ allerdings anders wählt, so kann sich die Größe der ROBDDs *extrem* verändern:

**Lemma 1.13** *Sei $X_\pi = \{x_1, \ldots, x_{2n}\}$ eine geordnete Variablenmenge und $F$ ein ROBDD über $X_\pi$ mit*

$$\begin{aligned} x_{2i-1} &= \pi[i] \quad \textit{und} \\ x_{2i} &= \pi[i+n], \quad 1 \leq i \leq n. \end{aligned}$$

*Ist $f_F = double_n$, dann gilt für die Größe von $F$:*

$$|F| \geq 2^n.$$

**Beweis:** Bezeichnen $\epsilon^1 = ({\epsilon^1}_1, \ldots, {\epsilon^1}_n)$ und $\epsilon^2 = ({\epsilon^2}_1, \ldots, {\epsilon^2}_n)$ zwei verschiedene $\epsilon$-Pfade in $F$. Sei $v^1$ der $\epsilon^1$-Knoten und $v^2$ der $\epsilon^2$-Knoten in $F$.
Behauptung: $v^1 \neq v^2$.
Den beiden Knoten entsprechen die beiden Kofaktoren
$double_n(\epsilon^i_1, x_2, \epsilon^i_2, x_4, \ldots, \epsilon^i_n, x_{2n})$, $i \in \{1,2\}$. Da $\epsilon^1$ und $\epsilon^2$ verschiedene Vektoren sind, gibt es ein $k \in \{1, \ldots n\}$ mit $\epsilon^1_k \neq \epsilon^2_k$
O.B.d.A $\epsilon^1_k = 1$. Dann gilt aber

$$\underbrace{double_n(\epsilon^1_1, 0, \ldots, \epsilon^1_k, 1, \ldots, \epsilon^1_n, 0)}_{=1} \neq \underbrace{double_n(\epsilon^2_1, 0, \ldots, \epsilon^2_k, 1, \ldots, \epsilon^2_n, 0)}_{=0}$$

d.h. die beiden Kofaktoren stellen unterschiedliche Funktionen dar, und damit gilt $v^1 \neq v^2$.
Da es $2^n$ verschiedene solcher $\epsilon$-Pfade gibt, die alle zu verschiedenen Knoten führen müssen, besitzt der ROBDD $F$ also mindestens $2^n$ Knoten.

□

Der Bestimmung einer guten Variablenordnung kommt daher eine große Bedeutung zu. Im obigen Beispiel kann durch die entsprechende Wahl der Variablenordnung die Größe der ROBDDs für die Funktionenfamilie linear mit $n$ wachsen (asymptotisch bester Fall), als auch exponentiellen Zuwachs erzielen (asymptotisch schlechtester Fall).
Die Bestimmung der optimalen Variablenordnung für eine Boolesche Funktion $f$, d.h. die Ordnung, für die der entsprechende ROBDD $F$ minimale Größe besitzt, ist *NP*-vollständig [BSW94]. Der beste zur Zeit bekannte Algorithmus hat eine Laufzeit von $O(3^n n^2)$ für eine Boolesche Funktion $f \in B_n$ [FS90].
Aus diesem Grund werden in der Praxis meist heuristische Verfahren für dieses Problem eingesetzt.
Diese Verfahren können in zwei Klassen unterteilt werden: In der ersten Klasse befinden sich Algorithmen, die ausgehend von einer Beschreibung der Booleschen Funktion (z.B. Netzliste), diese analysieren, um so eine möglichst gute Variablenordnung zu generieren (z.B. [MWBS88, FMK91]).
Die andere Klasse bilden dynamische Verfahren, deren Ziel es ist, eine bestehende Variablenordnung zu verbessern (z.B. [Rud93, DBG95]). Dies versetzt die Anwendungsprogramme in die Lage, zu jedem Zeitpunkt (so auch beim Aufbau eines ROBDDs) mittels eines speziellen Aufrufs, die vorliegende Ordnung zu verbessern, um so die Gesamtzahl der ROBDD Knoten zu vermindern. Die entsprechenden Aufrufe können z.B. dann erfolgen, wenn die Anzahl der Knoten eine bestimmte Grenze überschreitet. Das bekannteste dynamische Verfahren ist der $sift$−Algorithmus [Rud93], der sehr gute Ergebnisse liefert. Die Grundstruktur dieser Heuristik basiert auf einer Folge von *Variablenvertauschungen* innerhalb der Ordnung. Eine Vertauschung zweier in der Variablenordnung benachbarter Variablen $x_i$ und $x_j$ verändert lediglich die Knoten im ROBDD, welche mit $x_i$ und $x_j$ beschriftet sind. Dieser Umbau der ROBDD-Struktur kann daher sehr effizient durchgeführt werden. Ist $X_\pi = \{x_1, \ldots, x_n\}$, so bestimmt der $sift$−Algorithmus für jede Variable $x_i$ eine Position in der Variablenordnung, indem $x_i$ mittels einer Folge von benachbarten Vertauschungen $x_i \leftrightarrow x_j$ an alle $n$ möglichen Positionen geschoben und danach dort eingeordnet wird, an der die resultierende ROBDD-Größe am geringsten war. Dies wird für jede Variable aus $X_\pi$ durchgeführt. Nach $O(n^2)$ Schritten hat man so die Position aller Variablen bestimmt.
In [PS95] wurde allerdings gezeigt, daß die Heuristik in dieser Form nicht immer in der Lage ist, auch gute Ergebnisse zu erzielen: Liegt eine Gruppe $\lambda \subseteq X$ von Variablen für gute Variablenordnungen stets benachbart, so tendieren die Variablen aus $\lambda$ dazu, durch $sift$ in ihrer Position nicht verändert zu werden, d.h. sie kehren aufgrund der großen „Anziehungskraft" der Gruppe immer wieder an ihre alte Position zurück. Existieren mehrere solcher Gruppen, so ist es der $sift$−Heuristik unmöglich, die relativen Ordnungen der einzelnen Gruppen

untereinander zu verändern. Abhilfe kann dadurch geschaffen werden, daß mehrere Variablen, die in der Ordnung unmittelbar hintereinander folgen, zu einem *Block* zusammengefaßt werden. Dies veranlaßt *sift* bei der Wahl einer neuen Ordnung, die relativen Positionen der Blockvariablen unverändert zu lassen und lediglich eine Permutation der Blöcke zu suchen. Die Güte dieses Verfahrens hängt natürlich von der Verteilung der Variablen auf die einzelnen Blöcke ab.

## 1.6 Grundsätzliche Betrachtungen zur Komplexität Boolescher Funktionen

Bei allen Beschreibungen Boolescher Funktionen, die bisher betrachtet wurden (Funktionstabellen, Boolesche Polynome, ROBDDs), konnte stets eine Funktion angegeben werden, für die alle Darstellungen exponentielle Kosten (bzgl. der Anzahl der Variablen) aufweisen. Es wird nun gezeigt, daß diese Ergebnisse kein Zufall sind, sondern in der Art der Darstellungen begründet sind.
Um eine generelle Aussage über die Größe einer Darstellung anzugeben, ist der nächste Satz, der auf Shannon [Shan49] zurückgeht, besonders hilfreich:

**Satz 1.5 (Shannon '49)** *Für hinreichend große $n$ gilt für „fast alle“ † Funktionen $f \in B_n$:*

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

Der Satz besagt also, daß nahezu alle Booleschen Funktionen nur durch Schaltkreise mit exponentieller Komplexität realisiert werden können. Andererseits gilt aber auch [Lup58]:

**Satz 1.6 (Lupanov '58)** *Für jede Boolesche Funktion $f \in B_n$ gibt es einen Schaltkreis $S$, der $f$ realisiert mit*

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

Lupanov gibt zudem ein Verfahren an, um zu jeder Booleschen Funktion einen Schaltkreis zu konstruieren, der die Ungleichung im obigen Satz erfüllt (Lupanov'sche $k-s$-Darstellung). In Verbindung mit Satz 1.5 erhält man somit einen optimalen bzw. nahezu optimalen Schaltkreis zu fast jeder Funktion.
In der Praxis liefert diese Konstruktion aber Schaltkreise, deren Komplexität weit vom Optimum entfernt sind, da die Aussage von Shannon *nicht* auf alle Booleschen Funktionen abzielt.
Sehr viele für die Praxis relevanten Funktionen können mit deutlich kleineren Schaltungen realisiert werden.

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

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

Solche Funktionen sind nämlich nicht dadurch gegeben, daß die einzelnen Funktionswerte in einer (riesigen) Funktionstabelle zufällig bestimmt sind, sondern es liegt ihnen meist ein einfaches und kurzes Bildungsgesetz zugrunde (z.B. Addierer). Diese Bildungsgesetze führen zu besonderen *Regelmäßigkeiten* bzw. *Struktureigenschaften* der Funktionen, wie z.B. der Symmetrie. Gerade die Ausnutzung solcher Struktureigenschaften bei der Logiksynthese führt fast immer auch zu Realisierungen, die mit weit weniger als $\frac{2^n}{n}$ Grundgattern auskommen. Daher ist es für Logiksyntheseverfahren sehr wichtig, solche Struktureigenschaften zu erkennen und auszunutzen.

Satz 1.5 hat unmittelbaren Einfluß auf die in diesem Kapitel vorgestellten Beschreibungsmöglichkeiten Boolescher Funktionen:
Zu jedem Booleschen Ausdruck bzw. ROBDD läßt sich eine Schaltung entwerfen, welche die durch den Ausdruck bzw. ROBDD dargestellte Boolesche Funktion realisiert und dabei eine Komplexität besitzt, die sich nur um einen konstanten Faktor $c$ von den Kosten des Booleschen Ausdrucks bzw. der Größe des ROBDDs unterscheidet. Für hinreichend große $n$ besitzt somit nach Satz 1.5 für „fast alle" Booleschen Funktionen $f \in B_n$ jede Darstellung als Boolescher Ausdruck Kosten von mindestens $\frac{2^n}{cn}$ bzw. jeder ROBDD für $f$ mindestens $\frac{2^n}{cn}$ Knoten, unabhängig von der Variablenordnung.
Ähnlich wie beim Schaltungsentwurf, können auch hier durch Ausnutzung bestimmter Struktureigenschaften häufig wesentlich kleinere Darstellungen gefunden werden (vgl. $exor_n$–Funktion).

# Kapitel 2

# Symmetrien Boolescher Funktionen

Dieses Kapitel befaßt sich mit einer besonderen Struktureigenschaft Boolescher Funktionen, der *Symmetrieeigenschaft*. Die Ausnutzung von Symmetrieeigenschaften einer Booleschen Funktion erlaubt den Entwurf von effizienten Algorithmen zur Logiksynthese, welche in der Lage sind, sehr gute Resultate zu generieren (z.B. [Sch96, KD91]). Darüber hinaus verhilft die Kenntnis von symmetrischen Variablenmengen einer Funktion oftmals, den Suchraum bei Problemen, die beim Chipentwurf auftreten, stark einzuschränken, was sich in einem erheblichen Laufzeitgewinn der entsprechenden Algorithmen niederschlägt.

Dieses Kapitel beginnt zunächst mit der Einführung eines sehr allgemeinen Symmetriebegriffs aus [Ho74], welcher später in verschiedenen Formen spezialisiert wird. Dabei werden insbesondere die Symmetrieeigenschaften partieller Boolescher Funktionen betrachtet, die sich, wie anhand von Beispielen gezeigt wird, von denjenigen der totalen Funktionen erheblich unterscheiden.
Des weiteren wird exemplarisch gezeigt, welchen Einfluß Symmetrien auf die Schaltkreisrealisierung und die Darstellung Boolescher Funktionen, speziell auf die Beschreibung durch ROBDDs, besitzen.
Abschließend soll noch kurz auf die Erkennung bzw. Bestimmung von Symmetrien eingegangen werden.

## 2.1 $G-$Symmetrien Boolescher Funktionen

Dieser Abschnitt beschäftigt sich mit einem sehr allgemeinen Symmetriebegriff für Boolesche Funktionen, nämlich der $G-$Symmetrie, wie sie in [Ho74] vorgestellt wird. Da in späteren Abschnitten speziellere Symmetrieformen untersucht werden, die sich allerdings alle als $G-$Symmetrien beschreiben lassen, soll durch die Einführung der $G$-Symmetrie zum einen ein umfassender Überblick über die verschiedenen Symmetriebegriffe dieser Theorie gegeben werden und zum anderen auch eine Möglichkeit, die in dieser Arbeit erzielten Ergebnisse evtl. auf andere Symmetrieformen der Theorie zu übertragen.

Grundlage der $G-$Symmetrie bilden die beiden Mengen von Abbildungen

$$\Sigma_n := \{\sigma_{ij} \mid 1 \leq i < j \leq n\} \text{ und } \Upsilon_n := \{\nu_i \mid i = 1, \ldots, n\}$$

mit

$$\sigma_{ij} : \{0,1\}^n \rightarrow \{0,1\}^n$$

$$\sigma_{ij}((\epsilon_1, \ldots, \epsilon_i, \ldots, \epsilon_j, \ldots, \epsilon_n)) = (\epsilon_1, \ldots, \epsilon_j, \ldots, \epsilon_i, \ldots, \epsilon_n)$$

sowie

$$\nu_i : \{0,1\}^n \rightarrow \{0,1\}^n$$

$$\nu_i((\epsilon_1, \ldots, \epsilon_i, \ldots, \epsilon_n)) = (\epsilon_1, \ldots, \overline{\epsilon_i}, \ldots, \epsilon_n).$$

Die Abbildung $\sigma_{ij}$ vertauscht die $i$. und $j$. Komponente und $\nu_i$ negiert die $i$. Komponente eines Booleschen Vektors.
Die durch die Abbildungen aus $\Sigma_n \cup \Upsilon_n$ erzeugte Gruppe bezeichnet man mit $\mathrm{P_n}$.
Es wird nun der erste allgemeine Symmetriebegriff vorgestellt:

**Definition 2.1 (Starke $G-$Symmetrie)** *Sei $f \in BP_n$ eine partielle Boolesche Funktion und $G \subseteq \mathrm{P_n}$. $f$ heißt* **stark $G-$symmetrisch** *genau dann, falls für alle $\sigma \in G$ gilt:*[†]

$$f(\epsilon) = f(\sigma(\epsilon)) \qquad \forall \epsilon \in \{0,1\}^n .$$

**Bemerkung 8** *O.B.d.A. kann man annehmen, daß es sich bei $G$ um eine Gruppe handelt (Untergruppe von $\mathrm{P_n}$), denn falls für zwei Abbildungen $\sigma_1, \sigma_2 \in G$ $f(\epsilon) = f(\sigma_1(\epsilon)) = f(\sigma_2(\epsilon))$ gilt, dann ist offensichtlich auch $f(\epsilon) = f(\sigma_1(\sigma_2(\epsilon)))$ $= f(\sigma_1^{-1}(\epsilon))$, für alle $\epsilon \in \{0,1\}^n$ .*

*Es reicht daher aus, bei der Überprüfung einer starken $G-$Symmetrie für eine spezielle Gruppe $G$ und eine Boolesche Funktion, nur die Invarianz der Funktionswerte gegenüber einem Erzeugendensystem von $G$ zu zeigen.*

Ist eine Menge von Abbildungen $G \subseteq \mathrm{P_n}$ gegeben, so induziert diese eine Relation $\leftrightarrow_G$ auf $\{0,1\}^n$ , die für zwei Boolesche Vektoren angibt, ob sich diese durch Abbildungen aus $G$ ineinander überführen lassen.

**Bezeichnung 9** *Sei $G \subseteq \mathrm{P_n}$. Für die Relation $\leftrightarrow_G \subseteq \{0,1\}^n \times \{0,1\}^n$ gilt:*

$$\epsilon \leftrightarrow_G \epsilon' :\Longleftrightarrow \exists \sigma \in G \textit{ mit } \sigma(\epsilon) = \epsilon'.$$

Handelt es sich bei $G$ speziell um eine Untergruppe von $\mathrm{P_n}$, so besitzt die Relation $\leftrightarrow_G$ alle Eigenschaften einer Äquivalenzrelation:

**Lemma 2.1** *Die Relation $\leftrightarrow_G$ bildet für eine Gruppe $G \subseteq \mathrm{P_n}$ eine Äquivalenzrelation auf $\{0,1\}^n$ .*

[†]Man beachte, daß hierbei $f(\epsilon) \in \{0,1,*\}$.

**Beweis:** Reflexivität:
Da $G$ eine Gruppe ist, besitzt $G$ auch ein neutrales Element. Dies ist gerade die Identität $id$ mit der Eigenschaft $\forall \epsilon \in \{0,1\}^n : id(\epsilon) = \epsilon$.
Daher gilt auch für alle $\epsilon \in \{0,1\}^n$ $\epsilon \leftrightarrow_G \epsilon$.
Symmetrie:
Da $G$ eine Gruppe ist, gibt es zu jedem Element $\sigma \in G$ ein inverses Element $\sigma^{-1} \in G$ mit $\sigma \circ \sigma^{-1} = id$. Ist $\epsilon \leftrightarrow_G \epsilon'$, d.h. $\exists \sigma \in G : \sigma(\epsilon) = \epsilon'$, so gilt ebenfalls $\sigma^{-1}(\sigma(\epsilon)) = \epsilon = \sigma^{-1}(\epsilon')$ und somit $\epsilon' \leftrightarrow_G \epsilon$.
Transitivität:
Sind $\sigma_1$ und $\sigma_2$ zwei Abbildungen aus $G$, so ist auch $\sigma_2 \circ \sigma_1 \in G$. Damit folgt aus $\sigma_1(\epsilon) = \epsilon'$ und $\sigma_2(\epsilon') = \epsilon''$ auch $\sigma_2(\sigma_1(\epsilon)) = \epsilon''$ und somit $\epsilon \leftrightarrow_G \epsilon''$.

□

Damit wird also jeder Gruppe $G \subseteq \mathrm{P_n}$ eindeutig eine Partition von $\{0,1\}^n$ zugeordnet. Die einzelnen Elemente dieser Partition nennt man auch **Orbits**. Diese werden mit $orbit_G(\epsilon)$ für $\epsilon \in \{0,1\}^n$ bezeichnet, d.h.

$$orbit_G(\epsilon) := \{\epsilon' \in \{0,1\}^n \mid \epsilon \leftrightarrow_G \epsilon'\}.$$

Aus Definition 2.1 und der obigen Bezeichnung ergibt sich unmittelbar:

**Korollar 2.1** *Sei $f \in BP_n$ eine partielle Boolesche Funktion und $G \subseteq \mathrm{P_n}$ eine Gruppe. $f$ ist genau dann stark G-symmetrisch, falls*

$$\mid f(orbit_G(\epsilon)) \mid = 1 \qquad \forall \epsilon \in \{0,1\}^n .$$

Damit bilden die Orbits gerade die maximalen Mengen, für die $f$ den gleichen Funktionswert besitzen muß, damit die Funktion auch stark $G-$symmetrisch ist.
Die durch die Gruppe $G$ induzierten Orbits geben somit einen Aufschluß über die spezielle Struktur der $G-$symmetrischen Funktionen. Die Kenntnis der Orbits ist daher sehr wichtig bei der Bestimmung von besonderen Eigenschaften spezieller $G-$symmetrischer Funktionen.

Zusätzlich zu der bisher definierten starken Symmetrie ist es für partielle Boolesche Funktionen zudem sinnvoll, einen weiteren Symmetriebegriff einzuführen. Das folgende Beispiel soll dazu als Motivation dienen:

**Beispiel**
Sei $f \in BP_2$ wie folgt gegeben:

| $x_1$ | $x_2$ | $f(x_1, x_2)$ |
|---|---|---|
| 0 | 0 | * |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | * |

$f$ ist offensichtlich nicht stark $\{\nu_1, \nu_2\}$−symmetrisch, kann aber mittels $\mathbf{1} \in Erw(f)$ zu einer solchen Funktion erweitert werden. Nach den Ausführungen des vorigen Kapitels führt eine Realisierung dieser Konstantenfunktion auch gleichzeitig zu einer Schaltung für $f$. Daher können spezielle Schaltkreisentwürfe, die sich evtl. durch diese Symmetrie ergeben, für die Erweiterung und damit auch für $f$ ausgenutzt werden. Es ist daher zweckmäßig, auch $f$ in einer besonderen Form als $\{\nu_1, \nu_2\}$−symmetrisch anzusehen.
Dies führt zu dem Begriff der schwachen $G$−Symmetrien:

**Definition 2.2 (Schwache $G$−Symmetrie)** *Sei $f \in BP_n$ eine partielle Boolesche Funktion und $G \subseteq \mathrm{P_n}$. $f$ heißt* **(schwach) $G$−symmetrisch**† *genau dann, falls eine Erweiterung $g \in Erw(f)$ existiert, die stark $G$−symmetrisch ist.*
*Die Funktion $g$ heißt* **($G$−)symmetrische Erweiterung** *von $f$.*

Da die Erweiterungen eine reflexive Relation auf der Menge der Booleschen Funktionen bilden, ist offensichtlich jede stark $G$−symmetrische Funktion auch gleichzeitig schwach $G$−symmetrisch.
Auch schwach $G$-symmetrischen Funktionen lassen sich durch die entsprechenden Orbits charakterisieren:

**Lemma 2.2** *Eine partielle Boolesche Funktion $f \in BP_n$ ist genau dann schwach $G$−symmetrisch für eine Gruppe $G \subseteq \mathrm{P_n}$, falls*

$$\{0,1\} \not\subseteq f(orbit_G(\epsilon)) \qquad \forall \epsilon \in \{0,1\}^n .$$

**Beweis:** „$\Longrightarrow$"
Annahme: $\{0,1\} \subseteq f(orbit_G(\epsilon))$ für ein $\epsilon \in \{0,1\}^n$ .
O.B.d.A. $f(\epsilon) = 1$. In diesem Fall gibt es eine Abbildung $\rho \in G$ mit der Eigenschaft $f(\rho(\epsilon)) = 0$. Da diese Funktionswerte auch für alle Erweiterungen von $f$ nicht verändert werden, kann es keine Erweiterung geben, die stark $G$−symmetrisch ist.
Die Funktion $f$ ist demnach auch nicht schwach $G$−symmetrisch.
„$\Longleftarrow$"
Gilt $\{0,1\} \not\subseteq f(orbit_G(\epsilon))$ für alle $\epsilon \in \{0,1\}^n$ , dann ist die folgende (totale) Erweiterung von $f$ offensichtlich wohldefiniert:

$$f_G(\epsilon) := \begin{cases} f(\epsilon) & \text{falls } f(\epsilon) \neq * \\ 1 & \text{falls } f(\epsilon) = * \wedge 1 \in f(orbit_G(\epsilon)) \\ 0 & \text{falls } f(\epsilon) = * \wedge 0 \in f(orbit_G(\epsilon)) \\ 1 & \text{sonst} \end{cases}$$

Da nach Konstruktion $\mid f_G(orbit_G(\epsilon)) \mid = 1$ für alle $\epsilon \in \{0,1\}^n$ , ist $f_G$ stark $G$−symmetrisch und wegen $f_G \in Erw(f)$ demnach $f$ schwach $G$−symmetrisch.

□

†Nachfolgend wird auf den Zusatz *schwach* meist verzichtet. Dies gilt auch für die in den folgenden Abschnitten vorgestellten speziellen $G$−Symmetrien.

Aus dem Beweis zu diesem Lemma und dabei insbesondere durch die Konstruktion der totalen Erweiterung $f_G$ zu jeder Booleschen Funktion $f$ läßt sich das folgende Korollar ableiten:

**Korollar 2.2** *(1) Eine Boolesche Funktion $f \in BP_n$ ist genau dann $G-$symmetrisch, falls eine totale Erweiterung $g \in Erw(f)$ existiert, die stark $G-$symmetrisch ist.*
*(2) Eine totale Boolesche Funktion $f \in B_n$ ist genau dann (schwach) $G-$symmetrisch, falls $f$ stark $G-$symmetrisch ist.*

## 2.2 Spezielle $G-$Symmetrien

In diesem Abschnitt sollen einige wichtige spezielle $G-$symmetrische Funktionen betrachtet werden, die sich durch eine besondere Wahl der Teilmenge $G \subseteq \mathrm{P_n}$ auszeichnen. Durch die spezielle Wahl von $G$ ergeben sich oftmals besondere Eigenschaften der Relation $\leftrightarrow_G$, welche in entsprechenden Algorithmen effizient ausgenutzt werden können.

### 2.2.1 Vertauschungssymmetrie und Äquivalenzsymmetrie

Die im Zusammenhang mit Booleschen Funktionen, insbesondere in der Logiksynthese, sehr häufig verwendeten Symmetriebegriffe sind die Vertauschungssymmetrie und die Äquivalenzsymmetrie [Hu77].

**Definition 2.3 (Vertauschungssymmetrie und Äquivalenzsymmetrie)**
*Sei $f \in BP_n$ eine partielle Boolesche Funktion, $X = \{x_1, \dots, x_n\}$ eine Variablenmenge und sind $x_i, x_j \in X$ $(i < j)$.*
*$f$ heißt* **(stark) vertauschungssymmetrisch in** $(x_i, x_j)$ *genau dann, falls $f$ (stark) $\{\sigma_{ij}\}-$symmetrisch ist.*
*$f$ heißt* **(stark) äquivalenzsymmetrisch in** $(x_i, x_j)$ *genau dann, falls $f$ (stark) $\{\mu_{ij} := \nu_i \circ \sigma_{ij} \circ \nu_i\}-$symmetrisch ist.*

Aus dieser Definition ergibt sich direkt folgendes Korollar:

**Korollar 2.3**
*(1) Eine partielle Boolesche Funktion $f \in BP_n$ ist genau dann stark vertauschungssymmetrisch in $(x_i, x_j)$ $(i < j)$, falls für alle $\epsilon \in \{0,1\}^n$ gilt:*

$$f(\epsilon_1, \dots, \epsilon_i, \dots, \epsilon_j, \dots, \epsilon_n) = f(\epsilon_1, \dots, \epsilon_j, \dots, \epsilon_i, \dots, \epsilon_n).$$

*(2) Eine partielle Boolesche Funktion $f \in BP_n$ ist genau dann stark äquivalenzsymmetrisch in $(x_i, x_j)$ $(i < j)$, falls für alle $\epsilon \in \{0,1\}^n$ gilt:*

$$f(\epsilon_1, \dots, \epsilon_i, \dots, \epsilon_j, \dots, \epsilon_n) = f(\epsilon_1, \dots, \overline{\epsilon_j}, \dots, \overline{\epsilon_i}, \dots, \epsilon_n).$$

Eine starke Vertauschungssymmetrie in den beiden Variablen $x_i, x_j$ bedeutet demnach, daß die Funktion $f$ invariant gegenüber der Vertauschung der beiden

Variablen $x_i$ und $x_j$ ist. Die Äquivalenzsymmetrie verlangt zusätzlich noch vor der Vertauschung eine Negierung einer der beiden Variablen.
Die neu eingeführten Symmetrieformen können auch durch die Äquivalenz bestimmter Kofaktoren der partiellen Booleschen Funktion beschrieben werden:

**Lemma 2.3**
*(1) Eine partielle Boolesche Funktion $f \in BP_n$ ist genau dann stark vertauschungssymmetrisch in $(x_i, x_j)$, $i < j$, falls $f_{\overline{x_i}x_j} = f_{x_i\overline{x_j}}$.*
*(2) Eine partielle Boolesche Funktion $f \in BP_n$ ist genau dann stark äquivalenzsymmetrisch in $(x_i, x_j)$, $i < j$, falls $f_{\overline{x_i}\,\overline{x_j}} = f_{x_i x_j}$.*

**Beweis:** Zu (1)
„$\Longrightarrow$ "
Da $f$ stark vertauschungssymmetrisch in $(x_i, x_j)$ ist, gilt für zwei Funktionsstellen $\epsilon = (\epsilon_1, \ldots, \epsilon_i = 0, \ldots, \epsilon_j = 1, \ldots, \epsilon_n)$ und $\epsilon' = (\epsilon_1, \ldots, \epsilon_i = 1, \ldots, \epsilon_j = 0, \ldots, \epsilon_n)$ $\sigma_{ij}(\epsilon) = \epsilon'$ und somit $f(\epsilon) = f(\epsilon')$.
Da die beiden Vektoren beliebig gewählt wurden, sind die Kofaktoren $f_{\overline{x_i}x_j}$ und $f_{x_i\overline{x_j}}$ identisch.
„$\Longleftarrow$ "
Es ist zu zeigen, daß für alle $\epsilon \in \{0,1\}^n$ $f(\epsilon) = f(\sigma_{ij}(\epsilon))$ gilt.
Sei $\epsilon \in \{0,1\}^n$ beliebig. In Abhängigkeit von $\epsilon_i, \epsilon_j$ unterscheidet man die folgenden Fälle:
<u>Fall 1:</u> $\epsilon_i = \epsilon_j$
Dann ist $\sigma_{ij}(\epsilon) = \epsilon$ und somit trivialerweise $f(\epsilon) = f(\sigma_{ij}(\epsilon))$.
<u>Fall 2:</u> $\epsilon_i \neq \epsilon_j$
Dann ist $\sigma_{ij}(\epsilon) = (\epsilon_1, \ldots, \epsilon_j, \ldots, \epsilon_i, \ldots, \epsilon_n)$ und aufgrund der Voraussetzung für die beiden Kofaktoren $f(\epsilon) = f(\sigma_{ij}(\epsilon))$.
Damit ist $f$ stark vertauschungssymmetrisch in $(x_i, x_j)$.

Aussage (2) kann analog mit Hilfe der Abbildung $\mu_{ij}$ und den beiden Vektoren $\epsilon = (\epsilon_1, \ldots, \epsilon_i = 0, \ldots, \epsilon_j = 0, \ldots, \epsilon_n)$ und $\epsilon' = (\epsilon_1, \ldots, \epsilon_i = 1, \ldots, \epsilon_j = 1, \ldots, \epsilon_n)$ gezeigt werden.

□

Eine wichtige Eigenschaft der starken Vertauschungssymmetrie ist, daß es sich hierbei um eine Äquivalenzrelation auf der Menge der Variablen handelt:

**Lemma 2.4** *Ist $f \in BP_n$ eine partielle Boolesche Funktion und $X$ eine Variablenmenge, dann bildet die starke Vertauschungssymmetrie für $f$ eine Äquivalenzrelation und die starke Äquivalenzsymmetrie eine reflexive und symmetrische Relation auf $X$.*

**Beweis:**
Reflexivität und Symmetrie:
Dazu identifiziert man eine Vertauschungs- bzw. Äquivalenzsymmetrie in $(x_i, x_i)$ mit der identischen Abbildung $id \in \mathrm{P_n}$ sowie eine entsprechende Symmetrie in $(x_j, x_i)$ $(i < j)$ mit $\sigma_{ij}$ bzw. $\mu_{ij}$.

Transitivität:
Es ist zu zeigen, daß aus einer starken Vertauschungssymmetrie in $(x_i, x_j)$ und $(x_j, x_k)$ auch eine starke Vertauschungssymmetrie in $(x_i, x_k)$ folgt. Dabei kann o.B.d.A. angenommen werden, daß in den beiden Variablenpaaren jeweils verschiedene Variablen vorkommen, da es sich sonst um triviale Fälle handelt. Es gilt:
$f_{x_i x_j \overline{x_k}} \overset{(x_j,x_k)^\dagger}{=} f_{x_i \overline{x_j} x_k} \overset{(x_i,x_j)}{=} f_{\overline{x_i} x_j x_k}$ sowie
$f_{x_i \overline{x_j}\, \overline{x_k}} \overset{(x_i,x_j)}{=} f_{\overline{x_i} x_j \overline{x_k}} \overset{(x_j,x_k)}{=} f_{\overline{x_i}\, \overline{x_j} x_k}$ und damit
$f_{x_i, \overline{x_k}} = f_{\overline{x_i} x_k}$.

Damit ist $f$ stark vertauschungssymmetrisch in $(x_i, x_k)$ und die starke Vertauschungssymmetrie eine Äquivalenzrelation auf $X$.

□

Die starke Äquivalenzsymmetrie bildet keine Äquivalenzrelation, da ihr die Eigenschaft der Transitivität fehlt, wie das folgende Beispiel verdeutlicht:

**Beispiel**
Sei $f \in BP_3$ durch die folgende Funktionstabelle gegeben:

| $x_1$ | $x_2$ | $x_3$ | $f(x_1, x_2, x_3)$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

Die Funktion ist offensichtlich stark äquivalenzsymmetrisch in $(x_1, x_2)$ und $(x_2, x_3)$, aber sie ist nicht stark äquivalenzsymmetrisch in $(x_1, x_3)$, da $f(0,0,0) \neq f(1,0,1)$.

Läßt sich Lemma 2.4 auch auf die entsprechenden schwachen Symmetrien übertragen? Das dies nicht der Fall ist, zeigt das folgende Beispiel.
Es demonstriert einen wichtigen Unterschied zwischen der starken Vertauschungssymmetrie und der schwachen Vertauschungssymmetrie:

**Beispiel**
Sei $f \in BP_3$ durch die folgende Funktionstabelle gegeben:

† Die Tupel über dem Gleichheitszeichen geben diejenigen Symmetriepaare an, welche diese Gleichheit rechtfertigen.

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

Wie man leicht sieht, ist $f$ vertauschungssymmetrisch und äquivalenzsymmetrisch in $(x_1, x_2)$ sowie vertauschungssymmetrisch und äquivalenzsymmetrisch in $(x_2, x_3)$. $f$ ist aber weder vertauschungssymmetrisch in $(x_1, x_3)$, da $f(0,0,1) = 1$ und $f(1,0,0) = 0$, noch äquivalenzsymmetrisch in $(x_1, x_3)$, da $f(0,0,0) = 1$ und $f(1,0,1) = 0$.

Die schwache Vertauschungssymmetrie bildet also im Gegensatz zu starken Vertauschungssymmetrie *keine* Äquivalenzrelation auf der Variablenmenge.

Das folgende Lemma zeigt eine besondere Beziehung zwischen starker Vertauschungssymmetrie und starker Äquivalenzsymmetrie auf. Von der Aussage des Lemmas wird im folgenden sehr oft Gebrauch gemacht werden:

**Lemma 2.5** *Sei $f \in BP_n$ eine partielle Boolesche Funktion, $X$ eine Variablenmenge und $x_i, x_j, x_k \in X$. Ist $f$ stark äquivalenzsymmetrisch in $(x_i, x_j)$, dann gilt:*
*$f$ ist stark äquivalenzsymmetrisch in $(x_i, x_k)$ genau dann, falls $f$ stark vertauschungssymmetrisch in $(x_j, x_k)$ ist.*

**Beweis:** O.B.d.A. kann man annehmen, daß es sich bei den Variablen um paarweise verschiedene Elemente handelt, da sonst die Aussage trivial wird.
Ist $f$ stark äquivalenzsymmetrisch in $(x_i, x_j)$, so gilt $f_{x_i x_j} = f_{\overline{x_i}\,\overline{x_j}}$.
Ist $f$ überdies auch noch stark vertauschungssymmetrisch in $(x_j, x_k)$, so gilt weiter $f_{x_j \overline{x_k}} = f_{\overline{x_j} x_k}$, und demnach ist

$$f_{\overline{x_i}\,\overline{x_j}\,\overline{x_k}} \overset{(x_i,x_j)}{=} f_{x_i x_j \overline{x_k}} \overset{(x_j,x_k)}{=} f_{x_i \overline{x_j} x_k} \text{ und}$$

$$f_{\overline{x_i} x_j \overline{x_k}} \overset{(x_j,x_k)}{=} f_{\overline{x_i}\,\overline{x_j} x_k} \overset{(x_i,x_j)}{=} f_{x_i x_j x_k}.$$

Also ist $f$ stark äquivalenzsymmetrisch in $(x_i, x_k)$.
Ist umgekehrt $f$ stark äquivalenzsymmetrisch in $(x_i, x_j)$ und gleichzeitig stark äquivalenzsymmetrisch in $(x_i, x_k)$, so gilt:

$$f_{x_i x_j \overline{x_k}} \overset{(x_i,x_j)}{=} f_{\overline{x_i}\,\overline{x_j}\,\overline{x_k}} \overset{(x_i,x_k)}{=} f_{x_i \overline{x_j} x_k} \text{ und}$$

$$f_{\overline{x_i}\,\overline{x_j} x_k} \overset{(x_i,x_j)}{=} f_{x_i x_j x_k} \overset{(x_i,x_k)}{=} f_{\overline{x_i} x_j \overline{x_k}}.$$

Also ist $f$ stark vertauschungssymmetrisch in $(x_j, x_k)$.

□

### 2.2.2 Symmetrie

Vertauschungssymmetrie und Äquivalenzsymmetrie sollen nun zu einer neuen $G$–Symmetrie miteinander verknüpft werden. Da der in der folgenden Definition eingeführte Symmetriebegriff für die vorliegende Arbeit von zentraler Bedeutung ist, wird dieser daher auch einfach als *Symmetrie* bezeichnet.

**Definition 2.4 (Symmetrie)** *Sei $f \in BP_n$ eine partielle Boolesche Funktion, $X = \{x_1, \ldots, x_n\}$ eine Variablenmenge, $x_i, x_j \in X$, $\lambda \subseteq X$ und $\Pi = \{\lambda_1, \ldots, \lambda_k\}$ eine Partition von $X$.*
*$f$ heißt* **stark symmetrisch in** *$(x_i, x_j)$ genau dann, wenn $f$ stark vertauschungssymmetrisch oder stark äquivalenzsymmetrisch in $(x_i, x_j)$ ist.*
*$f$ heißt* **(schwach) symmetrisch in** *$(x_i, x_j)$ genau dann, wenn $f$ vertauschungssymmetrisch oder äquivalenzsymmetrisch in $(x_i, x_j)$ ist.*
*Ist $f$ nicht symmetrisch in $(x_i, x_j)$, dann heißt $f$* **asymmetrisch** *in $(x_i, x_j)$.*

*$f$ heißt* **stark symmetrisch in** *$\lambda$ genau dann, wenn $f$ in allen Variablenpaaren aus $\lambda$ stark symmetrisch ist.*
*$f$ heißt* **(schwach) symmetrisch in** *$\lambda$ genau dann, wenn eine Erweiterung von $f$ existiert, die stark symmetrisch in $\lambda$ ist.*
*$\lambda$ bezeichnet man als* **(starke) Symmetriemenge** *von $f$.*

*$f$ heißt* **stark symmetrisch in** *$\Pi$ genau dann, wenn $f$ stark symmetrisch in allen Mengen $\lambda_i \in \Pi$ ist.*
*$f$ heißt* **(schwach) symmetrisch in** *$\Pi$ genau dann, wenn eine Erweiterung von $f$ existiert, die stark symmetrisch in $\Pi$ ist.*
*$\Pi$ bezeichnet man als* **(starke) symmetrische Partition** *von $f$.*

Die Symmetrie bildet somit die disjunktive Verknüpfung der bisher vorgestellten Symmetriebegriffe.
Besonders ausgezeichnet sind Boolesche Funktionen, die in ihrer gesamten Variablenmenge symmetrisch sind:

**Bezeichnung 10** *Eine partielle Boolesche Funktion, die in der Menge ihrer Eingangsvariablen symmetrisch ist, heißt* **totalsymmetrisch***.*

Das nächste Lemma zeigt, daß auch die starke Symmetrie eine Äquivalenzrelation auf der Variablenmenge bildet. Aus diesem Grund ist es sinnvoll, auch symmetrische Mengen bzw. Partitionen zu betrachten, wie es Definition 2.4 zuläßt.

**Lemma 2.6** *Ist $f \in BP_n$ eine partielle Boolesche Funktion und $X$ eine Variablenmenge, dann bildet die starke Symmetrie für $f$ eine Äquivalenzrelation auf $X$.*

**Beweis:**

- Reflexivität und Symmetrie:
  Die beiden Eigenschaften ergeben sich direkt aus der Reflexivität und Symmetrie der entsprechenden starken Symmetrieformen (Vertauschungssymmetrie, Äquivalenzsymmetrie).

- Transitivität
  Seien $x_i, x_j, x_k \in X$ und sei $f$ stark symmetrisch in $(x_i, x_j)$ und in $(x_j, x_k)$.
  Man unterscheidet folgende Fälle:
  Fall 1: Es handelt sich jeweils um Vertauschungssymmetrien.
  Dann ist $f$ ebenfalls stark vertauschungssymmetrisch in $(x_i, x_k)$, da die starke Vertauschungssymmetrie transitiv ist.
  Fall 2: Es handelt sich jeweils um Äquivalenzsymmetrien.
  Dann ist $f$ stark vertauschungssymmetrisch in $(x_i, x_k)$ wegen Lemma 2.5.
  Fall 3: Sei o.B.d.A. $f$ stark äquivalenzsymmetrisch in $(x_i, x_j)$ und stark vertauschungssymmetrisch in $(x_j, x_k)$.
  Dann ist $f$ stark äquivalenzsymmetrisch in $(x_i, x_k)$ wegen Lemma 2.5.

Somit ist $f$ in jedem Fall stark symmetrisch in $(x_i, x_k)$.
Die starke Symmetrie bildet deshalb eine Äquivalenzrelation auf $X$.

□

Es soll nun die Einordnung der Symmetrie als spezielle $G-$Symmetrie unternommen werden. Dazu wird für jede symmetrische Partition $\Pi$ eine Menge $G \subseteq \mathrm{P_n}$ angegeben, welche der Symmetrie in $\Pi$ entspricht. Dadurch lassen sich auch die entsprechenden Orbits bestimmen, die, wie bereits erwähnt, einen genauen Einblick in die Struktur von $G-$symmetrischen Funktionen bieten.
Insbesondere ist man am Aufbau und der Anzahl der Orbits bzgl. der Symmetrie interessiert.
Dazu sind jedoch noch einige wichtige Vorbetrachtungen notwendig:

**Definition 2.5 (Phasenabbildung)**
*Sei $X = \{x_1, \ldots, x_n\}$ eine Variablenmenge. Eine Abbildung $p : X \rightarrow \{0,1\}$ heißt* **Phasenabbildung auf** $X$.
*Gilt $p(x) = 0$ bzw. $p(x) = 1$, so ordnet $p$ der Variablen $x$ eine negative bzw. positive Phase zu.*

Jeder stark symmetrischen Partition der Variablenmenge einer Booleschen Funktion kann eine Phasenabbildung zugeordnet werden. Die Abbildung gibt dann für jedes symmetrische Variablenpaar die Art der vorhandenen Symmetrie (Vertauschungssymmetrie, Äquivalenzsymmetrie) an:

**Definition 2.6 (Induzierte Phasenabbildung)**
*Sei $f \in BP_n$ eine partielle Boolesche Funktion, $X$ eine Variablenmenge und $\Pi = \{\lambda_1, \ldots, \lambda_k\}$ eine Partition von $X$. Ist $f$ stark symmetrisch in $\Pi$, dann heißt die Abbildung $p : X \rightarrow \{0,1\}$ mit $\forall x_i, x_j \in \lambda_l$ $(1 \leq l \leq k)$:*

$$p(x_i) = p(x_j) \iff f \text{ ist stark vertauschungssymmetrisch in } (x_i, x_j)$$

**die von $f$ und $\Pi$ induzierte Phasenabbildung auf $X$.**

Es ist zunächst nicht offensichtlich, ob zu jeder stark symmetrischen Partition überhaupt eine solche Phasenabbildung existiert. Daß dies jedoch der Fall ist, zeigt das nächste Lemma, in dem zu jeder stark symmetrischen Partition eine induzierte Phasenabbildung konstruiert wird:

**Lemma 2.7** *Sei $f \in BP_n$ eine partielle Boolesche Funktion und $X$ eine Variablenmenge. Ist $\Pi = \{\lambda_1, \ldots, \lambda_k\}$ eine stark symmetrische Partition von $f$, dann existiert eine von $f$ und $\Pi$ induzierte Phasenabbildung $p : X \to \{0, 1\}$.*

**Beweis:** Für jede Teilmenge $\lambda_j \in \Pi$ $(1 \leq j \leq k)$ wird die folgende Konstruktion angewandt:

Konstruktion:
Sei $\lambda_j = \{y_1, \ldots, y_l\} \in \Pi$.

Setze $p(y_1) = 1$. (*)

Setze für $2 \leq i \leq l$

$$p(y_i) = \begin{cases} p(y_{i-1}) & \text{falls } f \text{ stark vertauschungssymmetrisch in } (y_{i-1}, y_i) \text{ ist} \\ \overline{p(y_{i-1})} & \text{sonst} \end{cases}$$

Behauptung: Für $i < j$ gilt:
$p(y_i) = p(y_j) \iff f$ ist stark vertauschungssymmetrisch in $(y_i, y_j)$.
Die Behauptung wird mittels Induktion über $j - i$ bewiesen.

Induktionsanfang: $i = j - 1$
Für die Paare $(y_i, y_{i+1})$ folgt die Behauptung direkt aus der Konstruktion der Abbildung $p$.

Induktionsschritt: $j - i > 1$
Sei $j > i + 1$ beliebig.
Man betrachtet $y_{i+1}$.
„$\Longrightarrow$ “
Gilt $p(y_i) = p(y_{i+1}) = p(y_j)$, dann handelt es sich bei $(y_{i+1}, y_j)$ und $(y_i, y_{i+1})$ wegen der Induktionsvoraussetzung jeweils um Vertauschungssymmetrien. Da die starke Vertauschungssymmetrie eine Äquivalenzrelation bildet, ist $f$ auch stark vertauschungssymmetrisch in $(y_i, y_j)$.
Gilt $p(y_i) \neq p(y_{i+1}) \neq p(y_j)$, dann liegen jeweils Äquivalenzsymmetrien vor. Lemma 2.5 besagt in diesem Fall, daß $f$ stark vertauschungssymmetrisch in $(y_i, y_j)$ ist.
Damit ist $f$ in jedem Fall stark vertauschungssymmetrisch in $(y_i, y_j)$.
„$\Longleftarrow$ “
Ist $f$ stark vertauschungssymmetrisch in $(y_i, y_{i+1})$ und $(y_{i+1}, y_j)$, dann gilt nach Induktionsvoraussetzung $p(y_i) = p(y_{i+1}) = p(y_j)$.
Ist $f$ nicht stark vertauschungssymmetrisch, d.h. nur stark äquivalenzsymmetrisch in $(y_i, y_{i+1})$, dann kann $f$ nach Lemma 2.5 auch nicht stark vertauschungssymmetrisch in $(y_{i+1}, y_j)$ sein. Deshalb gilt $p(y_i) \neq p(y_{i+1})$ bzw. $p(y_{i+1}) \neq p(y_j)$,

d.h. $p(y_i) = p(y_j)$.
Damit gilt in jedem Fall $p(y_i) = p(y_j)$.

Führt man die Konstruktion für alle Elemente aus der Partition $\Pi$ durch, dann stellt $p$ eine von $f$ und $\Pi$ induzierte Phasenabbildung auf $X$ dar.

□

Wie aus der Konstruktion im Beweis zu Lemma 2.7 hervorgeht, ist die induzierte Phasenabbildung für eine Boolesche Funktion und eine Partition $\Pi$ nicht eindeutig bestimmt, da in Zeile (*) auch eine andere Zuordnung (d.h. 0 ) für $y_1$ möglich ist. Es ist aber leicht einzusehen, daß alle induzierten Phasenabbildungen gerade dadurch hervorgehen, daß für einige Symmetriemengen aus $\Pi$ die Phasen *aller* Variablen aus diesen Mengen umgekehrt werden.

Für die weiteren Untersuchungen benötigt man noch die folgende Bezeichnung:

**Bezeichnung 11** *Sei $X = \{x_1, \ldots, x_n\}$ eine Variablenmenge und $p$ eine Phasenabbildung auf $X$. Ist $\Pi = \{\lambda_1, \ldots, \lambda_k\}$ eine Partition von $X$, dann ist $G^{sym}_{\Pi,p}$ wie folgt festgelegt:*

$G^{sym}_{\Pi,p} :=$

$$< \bigcup_{\substack{x_i, x_j \in \lambda_l \\ 1 \le l \le k}} \{\rho \in \mathrm{P_n} \mid \rho = \sigma_{ij},\ \textit{falls}\ p(x_i) = p(x_j)\ \textit{und}\ \rho = \mu_{ij},\ \textit{falls}\ p(x_i) \neq p(x_j)\} >^{\dagger}$$

Aus der Konstruktion von $G^{sym}_{\Pi,p}$ und mit Korollar 2.3 folgt offensichtlich:

**Korollar 2.4** *Sei $\rho \in G^{sym}_{\Pi,p} \cap \{\sigma_{ij}, \mu_{ij}\}$ für ein Paar $i, j \in \{1, \ldots, n\}$ und $\epsilon \in \{0,1\}^n$. Dann gilt:*

$$\rho(\epsilon) \neq \epsilon \quad \Longleftrightarrow \quad \begin{array}{ll} \epsilon_i \neq p(x_i) \wedge \epsilon_j = p(x_j) & \textit{oder} \\ \epsilon_i = p(x_i) \wedge \epsilon_j \neq p(x_j) & \end{array}$$

Mit Hilfe dieser Begriffe kann nun die Symmetrie in die Theorie der $G-$symmetrischen Booleschen Funktionen eingeordnet werden:

**Satz 2.1** *Sei $f \in BP_n$ eine partielle Boolesche Funktion, $X$ eine Variablenmenge und $\Pi$ eine Partition von $X$. Dann gilt:*
*$f$ ist (stark) symmetrisch in $\Pi$ genau dann, falls eine Phasenabbildung $p : X \to \{0,1\}$ existiert, so daß $f$ (stark) $G^{sym}_{\Pi,p}-$symmetrisch ist.*

**Beweis:** Sei $\Pi = \{\lambda_1, \ldots, \lambda_k\}$ und $G := G^{sym}_{\Pi,p}$.
„$\Longrightarrow$“
Ist $f$ schwach symmetrisch in $\Pi$, dann gibt es eine Erweiterung $g \in Erw(f)$, die stark symmetrisch in $\Pi$ ist.
Sei $p$ eine von $g$ und $\Pi$ induzierte Phasenabbildung auf $X$.

† Mit $< G >$ wird die von der Menge $G$ mittels Komposition erzeugte Gruppe bezeichnet.

Behauptung: $g$ ist stark $G-$symmetrisch.
Die Menge

$$E := \bigcup_{\substack{x_i,x_j\in\lambda_l\\1\le l\le k}} \{\rho \mid \rho = \sigma_{ij}, \text{ falls } p(x_i) = p(x_j) \text{ und } \rho = \mu_{ij}, \text{ falls } p(x_i) \neq p(x_j)\}$$

bildet ein Erzeugendensystem von $G$.
Ist $\sigma_{ij} \in E$, dann sind $x_i, x_j \in \lambda_l$ für ein $l \in \{1,\ldots,k\}$ und $p(x_i) = p(x_j)$. Da $p$ eine von $g$ und $\Pi$ induzierte Phasenabbildung ist, ist $g$ stark vertauschungssymmetrisch in $(x_i, x_j)$ und $g$ damit invariant gegenüber $\sigma_{ij}$.
Im Fall $\mu_{ij} \in E$ gilt $p(x_i) \neq p(x_j)$. Da die beiden Variablen aus einer starken Symmetriemenge stammen und $g$ nicht stark vertauschungssymmetrisch in $(x_i, x_j)$ ist, muß $g$ stark äquivalenzsymmetrisch in $(x_i, x_j)$ sein und demnach invariant gegenüber $\mu_{ij}$.
Somit ist $g$ stark $G-$symmetrisch und $f$ $G-$symmetrisch.
Ist $f$ bereits stark symmetrisch in $\Pi$, dann gilt $g = f$, und $f$ ist somit stark $G-$symmetrisch.
„$\Longleftarrow$ “
Sei $p : X \rightarrow \{0,1\}$ eine beliebige Phasenabbildung auf $X$ und $f$ $G-$symmetrisch. Dann existiert eine Erweiterung $g \in Erw(f)$, die stark $G-$symmetrisch ist.
Aufgrund der Konstruktion von $G$ ist aber für jedes Variablenpaar $x_i, x_j \in \lambda_l$ $(1 \le l \le k)$ die Boolesche Funktion $g$ invariant gegenüber $\sigma_{ij}$ oder $\mu_{ij}$ und damit stark vertauschungssymmetrisch oder stark äquivalenzsymmetrisch in $(x_i, x_j)$, d.h. $g$ ist stark symmetrisch in $\Pi$.
Somit ist $f$ symmetrisch in $\Pi$.
Ist $f$ bereits stark $G-$symmetrisch, dann gilt $g = f$, und $f$ ist damit stark symmetrisch in $\Pi$.

□

Eine wichtige Folgerung aus dem Beweis ist:

**Korollar 2.5** *Sei $f \in BP_n$ eine partielle Boolesche Funktion, $X$ eine Variablenmenge und $\Pi$ eine Partition von $X$. Ist $f$ stark symmetrisch in $\Pi$ und $p : X \rightarrow \{0,1\}$ eine von $f$ und $\Pi$ induzierte Phasenabbildung auf $X$, dann ist $f$ stark $G^{sym}_{\Pi,p}-$symmetrisch.*

Bevor man sich der Bestimmung der entsprechenden Orbits von $G^{sym}_{\Pi,p}$ zuwenden kann, müssen noch einige wichtige Begriffe eingeführt werden:

**Bezeichnung 12** *Sei $X = \{x_1,\ldots,x_n\}$ eine Variablenmenge und $\lambda \subseteq X$. Dann bezeichnet*

$$Ind(\lambda) := \{i \in \{1,\ldots,n\} \mid x_i \in \lambda\}$$

*die* **Indexmenge von** $\lambda$.

**Definition 2.7 ($\lambda$-Gewicht eines Booleschen Vektors)**
*Sei $X$ $(\mid X \mid = n)$ eine Variablenmenge, $\lambda \subseteq X$ und $p : X \rightarrow \{0,1\}$ eine*

*Phasenabbildung auf $X$. Ist $\epsilon = (\epsilon_1, \ldots, \epsilon_n) \in \{0,1\}^n$ ein Boolescher Vektor, dann ist das $\lambda$* **-Gewicht** *von $\epsilon$ bzgl. $p$ ($|\epsilon|^p_\lambda$) wie folgt definiert:*

$$|\epsilon|^p_\lambda := \sum_{i \in Ind(\lambda)} 1 - |\epsilon_i - p(x_i)| \,.$$

Für jeden Booleschen Vektor ist das $\lambda$-Gewicht nach oben und unten begrenzt:

**Korollar 2.6** *Für alle $\epsilon \in \{0,1\}^n$ gilt:* $0 \leq |\epsilon|^p_\lambda \leq |\lambda|$ .

Mit Hilfe des $\lambda-$Gewichts können alle Booleschen Vektoren in verschiedene Mengen unterteilt werden:

**Definition 2.8 (Gewichtsklassen)**
*Sei $X = \{x_1, \ldots, x_n\}$ eine Variablenmenge und $\Pi = \{\lambda_1, \ldots, \lambda_k\}$ eine Partition von $X$. Für eine Phasenabbildung $p : X \to \{0,1\}$ heißt*

$$C^{\Pi,p}_{w_1,\ldots,w_k} := \{\epsilon \in \{0,1\}^n \mid \; |\epsilon|^p_{\lambda_i} = w_i, 1 \leq i \leq k\}$$

**Gewichtsklasse von** $\Pi$ *bzgl. der Gewichte $(w_1, \ldots w_k)$ und der Abbildung $p$.*

*Für zwei Boolesche Vektoren $\epsilon, \epsilon' \in \{0,1\}^n$ gilt $\epsilon \asymp_{\Pi,p} \epsilon'$ genau dann, falls Gewichte $(w_1, \ldots, w_k)$ existieren mit $\epsilon, \epsilon' \in C^{\Pi,p}_{w_1,\ldots,w_k}$.*

**Bemerkung 9** *Die Relation $\asymp_{\Pi,p}$ ist für eine Partition $\Pi$ und eine Phasenabbildung $p$ eine Äquivalenzrelation auf $\{0,1\}^n$, da die $\lambda-$Gewichte eines Booleschen Vektors durch $p$ eindeutig bestimmt sind. Die Partition, welche sich durch $\asymp_{\Pi,p}$ ergibt, besteht gerade aus den Gewichtsklassen von $\Pi$ bzgl. $p$.*

**Beispiel**
Sei $\Pi = \{\{x_1, \overline{x_2}\}, \{x_3, \overline{x_4}, x_5\}\}^\dagger = \{\lambda_1, \lambda_2\}$.

Das $\lambda_2$-Gewicht von $(1,0,1,0,0)$ ist 2.
$C^{\Pi,p}_{1,2}$ besteht aus allen Vektoren $\epsilon \in \{0,1\}^5$ mit einem $\lambda_1$-Gewicht von 1 und einem $\lambda_2$-Gewicht von 2, d.h.
$C^{\Pi,p}_{1,2} =$

$$\{(0,0,1,0,0),(0,0,0,0,1),(0,0,1,1,1),(1,1,1,0,0),(1,1,0,0,1),(1,1,1,1,1)\}.$$

Das folgende Lemma charakterisiert die Relation $\leftrightarrow_G$ für die Gruppe $G = G^{sym}_{\Pi,p}$ näher:

**Lemma 2.8** *Sei $X$ eine Variablenmenge, $\Pi$ eine Partition von $X$ und $p : X \to \{0,1\}$ eine Phasenabbildung.*
*Ist $G = G^{sym}_{\Pi,p}$, dann gilt für alle $\epsilon, \epsilon' \in \{0,1\}^n$ und $\lambda \in \Pi$:*

$$|\epsilon|^p_\lambda = |\epsilon'|^p_\lambda \;\wedge\; \epsilon_i = \epsilon'_i \text{ für } i \notin Ind(\lambda) \Rightarrow \epsilon \leftrightarrow_G \epsilon'.$$

† Für eine Variable $x_i$ bezeichnet $x_i$ eine positive Phase und $\overline{x_i}$ eine negative Phase für $x_i$.

**Beweis:** Man beweist die Aussage mittels Induktion über $\mid \lambda \mid$.

Induktionsanfang:
Für $\mid \lambda \mid = 1$ ist $\epsilon = \epsilon'$. Die Behauptung gilt, da $G$ eine Gruppe bildet und demnach die Identität $id$ enthält.
Sei $\mid \lambda \mid = 2$ und $\lambda = \{x_i, x_j\}$.
Es können zwei Fälle unterschieden werden:
Fall 1: $\mid \epsilon \mid_\lambda^p \in \{0, 2\}$
Dann ist $\epsilon = \epsilon'$ und somit $\epsilon \leftrightarrow_G \epsilon'$.
Fall 2: $\mid \epsilon \mid_\lambda^p = 1$
Sei o.B.d.A. $\epsilon_i = p(x_i)$ und $\epsilon_j \neq p(x_j)$.
Dann ist entweder $\epsilon = \epsilon'$ oder $\epsilon'_i \neq p(x_i)$ und $\epsilon'_j = p(x_j)$.
Besitzen $x_i$ und $x_j$ die gleiche Phase, dann ist $\sigma_{ij}(\epsilon) = \epsilon'$ und $\sigma_{ij} \in G$. Im anderen Fall ist $\mu_{ij}(\epsilon) = \epsilon'$ und $\mu_{ij} \in G$.
In jedem Fall gilt aber $\epsilon \leftrightarrow_G \epsilon'$.

Induktionsschritt: $\mid \lambda \mid > 2$
Sei $x_i \in \lambda$, $\lambda' = \lambda \backslash \{x_i\}$ und $\Pi' = \Pi \backslash \lambda \bigcup \{\lambda', \{x_i\}\}$, dann können wiederum zwei Fälle unterschieden werden:
Fall 1: $\epsilon_i = \epsilon'_i$
Dann ist $\mid \epsilon \mid_{\lambda'}^p = \mid \epsilon' \mid_{\lambda'}^p$ und nach Induktionsvoraussetzung gilt $\rho(\epsilon) = \epsilon'$ für ein $\rho \in G_{\Pi',p}^{sym} \subseteq G_{\Pi,p}^{sym}$.
Fall 2: $\epsilon_i \neq \epsilon'_i$
Sei o.B.d.A. $\epsilon_i \neq p(x_i), \epsilon'_i = p(x_i)$, dann gilt $\mid \epsilon \mid_{\lambda'}^p = \mid \epsilon' \mid_{\lambda'}^p +1 > 0$. Daher gibt es mindestens ein $j \in Ind(\lambda')$ mit $\epsilon_j = p(x_j)$.
Besitzen $x_i$ und $x_j$ die gleiche Phase, dann sei $\epsilon'' = \sigma_{ij}(\epsilon)$ und im anderen Fall sei $\epsilon'' = \mu_{ij}(\epsilon)$.
In jedem Fall ist aber $\mid \epsilon'' \mid_{\lambda'}^p = \mid \epsilon' \mid_{\lambda'}^p$ und $\epsilon''_i = \epsilon'_i$.
Nach Induktionsvoraussetzung existiert eine weitere Abbildung $\rho' \in G_{\Pi',p}^{sym} \subseteq G_{\Pi,p}^{sym}$ mit $\rho'(\epsilon'') = \epsilon'$. Bildet man $\rho = \rho' \circ \sigma_{ij}$ bzw. $\rho = \rho' \circ \mu_{ij}$, dann gilt $\rho(\epsilon) = \epsilon'$ und $\rho \in G$.

$\square$

Anhand dieser Vorüberlegungen können die Orbits bzgl. der Symmetriegruppe $G_{\Pi,p}^{sym}$ exakt bestimmt werden:

**Satz 2.2** *Sei $X$ eine Variablenmenge, $\Pi = \{\lambda_1, \ldots, \lambda_k\}$ eine Partition von $X$ und $p : X \rightarrow \{0,1\}$ eine Phasenabbildung auf $X$. Ist $G = G_{\Pi,p}^{sym}$, dann gilt:*

$$orbit_G(\epsilon) = C^{\Pi,p}_{|\epsilon|_{\lambda_1}^p, \ldots, |\epsilon|_{\lambda_k}^p} \quad \forall \epsilon \in \{0,1\}^n .$$

**Beweis:** Der Beweis des Satzes wird durch die beiden Inklusionen $\subseteq$ und $\supseteq$ unterteilt.
„ $\subseteq$ "
Die Menge

$$E := \bigcup_{\substack{x_i, x_j \in \lambda_l \\ 1 \leq l \leq k}} \{\rho \mid \rho = \sigma_{ij}, \text{ falls } p(x_i) = p(x_j) \text{ und } \rho = \mu_{ij}, \text{ falls } p(x_i) \neq p(x_j)\}$$

bildet ein Erzeugendensystem von $G$.
Sei $\epsilon \in \{0,1\}^n$ und $\rho \in E$ beliebig. Es sind zwei Fälle zu unterscheiden:
Fall 1: $\rho = \sigma_{ij}$
Dann sind $x_i, x_j \in \lambda_r$ für ein $r \in \{1,\dots,k\}$ und
$\sigma_{ij}(\epsilon_1,\dots,\epsilon_i,\dots,\epsilon_j,\dots,\epsilon_n) = (\epsilon_1,\dots,\epsilon_j,\dots,\epsilon_i,\dots,\epsilon_n)$.
Somit gilt:

$$\begin{aligned}
&\mid (\epsilon_1,\dots,\epsilon_i,\dots,\epsilon_j,\dots,\epsilon_n) \mid^p_{\lambda_r} \\
&= \sum_{l \in Ind(\lambda_r)} 1- \mid \epsilon_l - p(x_l) \mid \\
&= (\sum_{l \in Ind(\lambda_r)\setminus\{i,j\}} 1- \mid \epsilon_l - p(x_l) \mid) + 1- \mid \epsilon_i - p(x_i) \mid + 1- \mid \epsilon_j - p(x_j) \mid \\
&= (\sum_{l \in Ind(\lambda_r)\setminus\{i,j\}} 1- \mid \epsilon_l - p(x_l) \mid) + 1- \mid \epsilon_j - p(x_i) \mid + 1- \mid \epsilon_i - p(x_j) \mid \\
&\ \text{da } p(x_i) = p(x_j) \\
&= \mid (\epsilon_1,\dots,\epsilon_j,\dots,\epsilon_i,\dots,\epsilon_n) \mid^p_{\lambda_r} .
\end{aligned}$$

Da $\mid \epsilon \mid^p_{\lambda_s} = \mid \sigma_{ij}(\epsilon) \mid^p_{\lambda_s}$ für $s \neq r$, gilt $\epsilon \asymp_{\Pi,p} \sigma_{ij}(\epsilon)$.
Fall 2: $\rho = \mu_{ij}$
Dann sind $x_i, x_j \in \lambda_r$ für ein $r \in \{1,\dots,k\}$ und
$\mu_{ij}(\epsilon_1,\dots,\epsilon_i,\dots,\epsilon_j,\dots,\epsilon_n) = (\epsilon_1,\dots,\overline{\epsilon_j},\dots,\overline{\epsilon_i},\dots,\epsilon_n)$.
Somit gilt:

$$\begin{aligned}
&\mid (\epsilon_1,\dots,\epsilon_i,\dots,\epsilon_j,\dots,\epsilon_n) \mid^p_{\lambda_r} \\
&= \sum_{l \in Ind(\lambda_r)} 1- \mid \epsilon_l - p(x_l) \mid \\
&= (\sum_{l \in Ind(\lambda_r)\setminus\{i,j\}} 1- \mid \epsilon_l - p(x_l) \mid) + 1- \mid \epsilon_i - p(x_i) \mid + 1- \mid \epsilon_j - p(x_j) \mid \\
&= (\sum_{l \in Ind(\lambda_r)\setminus\{i,j\}} 1- \mid \epsilon_l - p(x_l) \mid) + 1- \mid \overline{\epsilon_j} - p(x_i) \mid + 1- \mid \overline{\epsilon_i} - p(x_j) \mid \\
&\ \text{da } p(x_i) = \overline{p(x_j)} \\
&= \mid (\epsilon_1,\dots,\overline{\epsilon_j},\dots,\overline{\epsilon_i},\dots,\epsilon_n) \mid^p_{\lambda_r} .
\end{aligned}$$

Da $\mid \epsilon \mid^p_{\lambda_s} = \mid \mu_{ij}(\epsilon) \mid^p_{\lambda_s}$ für $s \neq r$, gilt $\epsilon \asymp_{\Pi,p} \mu_{ij}(\epsilon)$.
Da $E$ ein Erzeugendensystem von $G$ bildet, ist $orbit_G(\epsilon) \subseteq C^{\Pi,p}_{|\epsilon|^p_{\lambda_1},\dots,|\epsilon|^p_{\lambda_k}}$ gezeigt.
„ $\supseteq$ “
Man konstruiert zu zwei beliebigen Booleschen Vektoren $\epsilon, \epsilon'$ mit $\epsilon \asymp_{\Pi,p} \epsilon'$ eine Abbildung $\rho \in G$, so daß $\rho(\epsilon) = \epsilon'$.
Dazu bildet man für alle $\lambda_l$ $(1 \leq l \leq k)$ mittels der Konstruktion aus dem Beweis zu Lemma 2.8 die Abbildungen $\rho_l \in G$, so daß $\rho_l(\epsilon)_i = \epsilon'_i$ für $i \in Ind(\lambda_l)$. Da die Abbildungen aus $\rho_l$ nur die Komponenten $\epsilon_i$ mit $i \in Ind(\lambda_l)$ bearbeiten und $\Pi$ eine Partition von $X$ bildet, sind diese Abbildungen unabhängig voneinander und können in beliebiger Reihenfolge hintereinander ausgeführt werden.

Ist $\rho = \rho_1 \circ \ldots \circ \rho_k$, dann ist $\rho \in G$ und offensichtlich $\rho(\epsilon) = \epsilon'$.
Somit gilt $orbit_G(\epsilon) \supseteq C^{\Pi,p}_{|\epsilon|^p_{\lambda_1},\ldots,|\epsilon|^p_{\lambda_k}}$.

□

Damit bilden die Gewichtsklassen von $\Pi$ bzgl. $p$ also die entsprechenden Orbits von $G^{sym}_{\Pi,p}$. Somit folgt aus Korollar 2.1 und Lemma 2.2:

**Korollar 2.7** *Sei $f \in BP_n$ eine partielle Boolesche Funktion, $X$ eine Variablenmenge und $\Pi = \{\lambda_1, \ldots, \lambda_k\}$ eine Partition von $X$. Dann gilt:*
*(1) $f$ ist stark symmetrisch in $\Pi$ genau dann, falls eine Phasenabbildung $p : X \rightarrow \{0,1\}$ existiert mit*

$$\forall 0 \le w_i \le |\lambda_i| \quad (1 \le i \le k) \quad |f(C^{\Pi,p}_{w_1,\ldots,w_k})| = 1.$$

*(2) $f$ ist schwach symmetrisch in $\Pi$ genau dann, falls eine Phasenabbildung $p : X \rightarrow \{0,1\}$ existiert mit*

$$\forall 0 \le w_i \le |\lambda_i| \quad (1 \le i \le k) \quad \{0,1\} \not\subseteq f(C^{\Pi,p}_{w_1,\ldots,w_k}).$$

Für die Anzahl der Orbits bzgl. der Symmetrie folgt aus Korollar 2.6:

**Korollar 2.8** *Sei $f \in BP_n$ eine partielle Boolesche Funktion, $X$ eine Variablenmenge und $\Pi = \{\lambda_1, \ldots, \lambda_k\}$ eine stark symmetrische Partition von $X$ für $f$. Dann gibt es genau $\prod_{i=1}^{k}(|\lambda_i| + 1)$ verschiedene Orbits bzgl. $G^{sym}_{\Pi,p}$.*

#### 2.2.2.1 Besondere Eigenschaften der schwachen Symmetrie

Im vorigen Abschnitt wurde schon anhand eines Beispiels der entscheidende Unterschied zwischen der starken und schwachen Symmetrie deutlich gemacht: Die schwache Symmetrie bildet *keine* Äquivalenzrelation auf der Variablenmenge. Daraus ergeben sich weitere Besonderheiten schwacher Symmetrien, die im folgenden Lemma zusammengefaßt sind:

**Lemma 2.9** *Sei $f \in BP_n$ eine partielle Boolesche Funktion und $X$ eine Variablenmenge.*

- *(1) Sei $\lambda \subseteq X$. Ist $f$ schwach symmetrisch in allen Paaren $(x_i, x_j) \in \lambda \times \lambda$, dann ist $f$ nicht notwendigerweise auch schwach symmetrisch in $\lambda$.*
- *(2) Sei $\Pi = \{\lambda_1, \ldots, \lambda_k\}$ eine Partition von $X$. Ist $f$ in allen Mengen $\lambda_i \in \Pi$ schwach symmetrisch, dann ist $f$ nicht notwendigerweise auch schwach symmetrisch in $\Pi$.*

**Beweis:** Zu (1)
Betrachte $f \in BP_5$ mit

$$f(\epsilon) = \begin{cases} 1 & \text{falls } \epsilon = (0,0,1,1,0) \\ * & \text{falls } \epsilon \in \{(0,1,0,1,0),(0,1,1,0,0),(1,0,0,1,0),(1,0,1,0,0)\} \\ 0 & \text{falls } \epsilon = (1,1,0,0,0) \\ 0 & \text{falls } \epsilon = (0,0,0,0,1) \\ 1 & \text{falls } \epsilon \in \{(0,1,0,1,1),(1,0,1,0,1)\} \\ 0 & \text{falls } \epsilon = (1,1,1,1,1) \\ * & \text{sonst} \end{cases}$$

Aufgrund der Konstruktion ist $f$ nicht äquivalenzsymmetrisch in $(x_1, x_3)$ und $(x_2, x_4)$, da $f(0,1,0,1,1) = f(1,0,1,0,1) \neq f(1,1,1,1,1)$.
Offensichtlich ist $f$ aber vertauschungssymmetrisch und damit auch symmetrisch in allen Paaren $(x_i, x_j)$ mit $i, j, \in \{1,2,3,4\}$.
Ist $f'$ eine Erweiterung von $f$, die in $(x_1, x_3)$ stark symmetrisch ist, dann muß $f'(0,1,1,0,0) = 0$ gelten. Ebenso muß für jede Erweiterung $f''$, die in $(x_2, x_4)$ stark symmetrisch ist $f''(0,1,1,0,0) = 1$ gelten.
Es kann daher überhaupt keine Erweiterung von $f$ existieren, die sowohl in $(x_1, x_3)$ als auch in $(x_2, x_4)$ stark symmetrisch ist und daher auch keine Erweiterung, die stark symmetrisch in $\{x_1, x_2, x_3, x_4\}$ ist.
Zu (2)
Wählt man als Partition $\Pi = \{\{x_1, x_3\}, \{x_2, x_4\}\}$ und verwendet die Funktion aus (1), so ist die Behauptung gezeigt.

□

Es ist also im allgemeinen für partielle Boolesche Funktionen nicht möglich, von bekannten Symmetrien in einer kleinen Menge automatisch auf Symmetrien in einer größeren Obermenge zu schließen, falls es sich nicht ausschließlich um starke Symmetrien handelt.
Der folgende Satz besagt jedoch, daß unter gewissen Voraussetzungen, von einer (starken) Symmetrie in $\lambda_1$ und $\lambda_2$ auf eine Symmetrie in $\lambda_1 \cup \lambda_2$ geschlossen werden kann:

**Satz 2.3** *Sei $f \in BP_n$ eine partielle Boolesche Funktion, die stark symmetrisch in der Partition $\Pi$ ist. Sind $[x_i], [x_j]^\dagger \in \Pi$ mit $|\,[x_i]\,| = 1$, und ist $f$ symmetrisch in $(x_i, x_j)$, dann ist $f$ auch symmetrisch in der Partition $\Pi' = \Pi \setminus \{[x_i], [x_j]\} \bigcup \{[x_i] \cup [x_j]\}$.*

**Beweis:** O.B.d.A. $\Pi = \{\lambda_1, \ldots, \lambda_k\}$ und $\lambda_1 = \{x_i\}, \lambda_2 = [x_j]$ und $\lambda_1 \cap \lambda_2 = \emptyset$. Dann ist $\Pi' = \{\lambda_1 \cup \lambda_2, \lambda_3, \ldots, \lambda_k\}$.
Sei $p : X \to \{0,1\}$ eine von $f$ und $\Pi$ induzierte Phasenabbildung auf $X$. Ist $f$ vertauschungssymmetrisch (bzw. äquivalenzsymmetrisch) in $(x_i, x_j)$, dann sei $p(x_i) = p(x_j)$ (bzw. $p(x_i) \neq p(x_j)$) $^\ddagger$.
Es ist also zu zeigen, daß es keine Gewichtsklasse $C^{\Pi',p}_{w_2,\ldots,w_k}$ von $\Pi'$ gibt mit $\{0,1\} \subseteq f(C^{\Pi',p}_{w_2,\ldots,w_k})$.
Man unterscheidet zwei Fälle:
<u>Fall 1:</u> $w_2 \geq 1$
Da $|\,\lambda_1\,| = 1$, kann $C^{\Pi',p}_{w_2,\ldots,w_k}$ als disjunkte Vereinigung zweier Gewichtsklassen von $\Pi$ dargestellt werden:

$$C^{\Pi',p}_{w_2,\ldots,w_k} = C^{\Pi,p}_{0,w_2,\ldots,w_k} \cup C^{\Pi,p}_{1,w_2-1,\ldots,w_k}$$

Da $f$ stark symmetrisch in $\Pi$ ist und $p$ eine induzierte Phasenabbildung, gilt nach Korollar 2.5 und 2.7 $|\,f(C^{\Pi,p}_{0,w_2,\ldots,w_k})\,| = |\,f(C^{\Pi,p}_{1,w_2-1,\ldots,w_k})\,| = 1$.
Ist $\{0,1\} \subseteq f(C^{\Pi',p}_{w_2,\ldots,w_k})$, so kann nur $f(C^{\Pi,p}_{0,w_2,\ldots,w_k}) = b$ und $f(C^{\Pi,p}_{1,w_2-1,\ldots,w_k}) = \overline{b}$

---

$^\dagger$Für eine Partition $\Pi$ ist $[x] = \lambda \in \Pi$ mit $x \in \lambda$.
$^\ddagger$Eine solche induzierte Phasenabbildung existiert immer, da $x_i \notin [x_j]$.

gelten ($b \in \{0,1\}$ ).
Es gibt daher zwei Vektoren $\epsilon \in C^{\Pi,p}_{0,w_2,\ldots,w_k}$ und $\epsilon' \in C^{\Pi,p}_{1,w_2-1,\ldots,w_k}$, so daß $\sigma_{ij}(\epsilon) = \epsilon'$ (bzw. $\mu_{ij}(\epsilon) = \epsilon'$).
Daher kann $f$ nicht vertauschungssymmetrisch (bzw. äquivalenzsymmetrisch) in $(x_i, x_j)$ sein. Dies widerspricht aber der Annahme.

Fall 2: $w_2 = 0$
Da $C^{\Pi',p}_{w_2,\ldots,w_k} = C^{\Pi,p}_{0,w_2,\ldots,w_k}$ und $f$ stark symmetrisch in $\Pi$ ist, gilt $\{0,1\} \not\subseteq f(C^{\Pi',p}_{w_2,\ldots,w_k})$.

□

Es soll hier noch darauf hingewiesen werden, daß die Aussage von Satz 2.3 im allgemeinen falsch wird, falls $|\,[x_i]\,| \neq 1$ oder für die Boolesche Funktion $f$ anstelle der starken Symmetrie nur noch eine schwache Symmetrie in $\Pi$ vorliegt. Dies belegen die folgenden Beispiele:

**Beispiele**
Betrachte folgende Funktion $f \in BP_4$ mit

$$f(\epsilon) = \begin{cases} 0 & \text{falls } \epsilon \in \{(1,1,0,0),(1,1,1,1)\} \\ 1 & \text{falls } \epsilon \in \{(0,0,1,1),(0,0,0,0)\} \\ * & \text{sonst} \end{cases}$$

Wie man leicht sieht, ist $f$ symmetrisch in allen Paaren $(x_i, x_j)$, $i, j \in \{1,2,3,4\}$ und weder in $(x_1, x_2)$ noch in $(x_3, x_4)$ äquivalenzsymmetrisch.
$f$ ist stark (vertauschungs)symmetrisch in $\{x_1, x_2\}$ und $\{x_3, x_4\}$, d.h. stark symmetrisch in der Partition $\Pi = \{\{x_1, x_2\}, \{x_3, x_4\}\}$. $f$ ist aber *nicht* symmetrisch in $\Pi' = \{\{x_1, x_2, x_3, x_4\}\}$, da es keine Phasenabbildung $p$ gibt, so daß $f$ $G^{sym}_{\Pi',p}$−symmetrisch ist:
Da $f$ nicht äquivalenzsymmetrisch in den Paaren $(x_1, x_2)$ und $(x_3, x_4)$ ist, können den beiden Variablen eines Paares nur die gleiche Phase zugeordnet werden. Daher gibt es genau zwei Möglichkeiten:
(1) Alle Variablen besitzen die gleiche Phase.
Dann ist aber $(1,1,0,0) \asymp_{\Pi',p} (0,0,1,1)$.
(2) $p(x_1) \neq p(x_3)$.
Dann ist aber $(0,0,0,0) \asymp_{\Pi',p} (1,1,1,1)$.
Damit ist $f$ in keinem Fall symmetrisch in $\Pi'$.

Die Funktion $f \in BP_3$ mit

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

ist schwach (vertauschungs)symmetrisch in $(x_1, x_2)$ und in $(x_2, x_3)$ und demnach sicher schwach symmetrisch in $\Pi = \{\{x_1, x_2\}, \{x_3\}\}$.
$f$ ist überdies weder in $(x_1, x_2)$ noch in $(x_2, x_3)$ äquivalenzsymmetrisch auch nicht vertauschungssymmetrisch in $(x_1, x_3)$.
Eine Erweiterung von $f$, die stark symmetrisch in $\Pi' = \{\{x_1, x_2, x_3\}\}$ ist, muß daher stark vertauschungssymmetrisch in $(x_1, x_2)$ und $(x_2, x_3)$ sein und somit auch stark vertauschungssymmetrisch in $(x_1, x_3)$. Dies ist aber nicht möglich.
$f$ ist daher *nicht* symmetrisch in $\Pi'$.

### 2.2.3 Multiformsymmetrie

Es besteht auch die Möglichkeit, Vertauschungssymmetrie und Äquivalenzsymmetrie konjunktiv miteinander zu verknüpfen, um zu einer neuen Symmetrieform zu gelangen. Die resultierende Relation erhält die Bezeichnung *Multiformsymmetrie.*

**Definition 2.9 (Multiformsymmetrie)** *Sei $f \in BP_n$ eine partielle Boolesche Funktion, $X = \{x_1, \ldots, x_n\}$ eine Variablenmenge, $x_i, x_j \in X$, $\lambda \subseteq X$ und $\Pi = \{\lambda_1, \ldots, \lambda_k\}$ eine Partition von $X$.*
*$f$ heißt* **stark multiformsymmetrisch in** $(x_i, x_j)$ *genau dann, wenn $f$ stark vertauschungssymmetrisch und stark äquivalenzsymmetrisch in $(x_i, x_j)$ ist.*
*$f$ heißt* **(schwach) multiformsymmetrisch in** $(x_i, x_j)$ *genau dann, wenn $f$ vertauschungssymmetrisch und äquivalenzsymmetrisch in $(x_i, x_j)$ ist.*

*$f$ heißt* **stark multiformsymmetrisch in** $\lambda$ *genau dann, wenn $f$ in allen Variablenpaaren aus $\lambda$ stark multiformsymmetrisch ist.*
*$f$ heißt* **(schwach) multiformsymmetrisch in** $\lambda$ *genau dann, wenn eine Erweiterung von $f$ existiert, die stark multiformsymmetrisch in $\lambda$ ist.*
*$\lambda$ bezeichnet man als* **(starke) multiformsymmetrische Menge** *von $f$.*

*$f$ heißt* **stark multiformsymmetrisch in** $\Pi$ *genau dann, wenn $f$ stark multiformsymmetrisch in allen Mengen $\lambda_i \in \Pi$ ist.*
*$f$ heißt* **(schwach) multiformsymmetrisch in** $\Pi$ *genau dann, wenn eine Erweiterung von $f$ existiert, die stark multiformsymmetrisch in $\Pi$ ist.*
*$\Pi$ bezeichnet man als* **(starke) multiformsymmetrische Partition** *von $f$.*

Bisher konnte jeder $G-$symmetrischen Booleschen Funktion eine Erweiterung zugeordnet werden, die stark $G-$symmetrisch ist. In der obigen Definition ist für die Multiformsymmetrie in einem Variablenpaar aufgrund der konjunktiven Verknüpfung von Vertauschungssymmetrie und Äquivalenzsymmetrie nicht unmittelbar erkennbar, ob jede Boolesche Funktion, die vertauschungssymmetrisch und äquivalenzsymmetrisch in einem Variablenpaar $(x_i, x_j)$ ist, auch eine Erweiterung besitzt, die *gleichzeitig* stark vertauschungssymmetrisch *und* stark äquivalenzsymmetrisch in $(x_i, x_j)$ ist.
Aufgrund der besonderen Eigenschaften der Abbildungen $\mu_{ij}$ und $\sigma_{ij}$ ist dies aber stets gewährleistet:

**Lemma 2.10** *Jede partielle Boolesche Funktion, die schwach multiformsymmetrisch in einem Variablenpaar $(x_i, x_j)$ ist, besitzt eine Erweiterung, welche stark multiformsymmetrisch in $(x_i, x_j)$ ist.*

**Beweis:** Sei $f \in BP_n$ und $X = \{x_1, \ldots, x_n\}$ eine Variablenmenge.
Es ist leicht nachzuprüfen, daß für jeden Vektor $\epsilon \in \{0,1\}^n$ entweder $\sigma_{ij}(\epsilon) \neq \epsilon$ oder $\mu_{ij}(\epsilon) \neq \epsilon$ gilt.
Ist $f$ vertauschungssymmetrisch und äquivalenzsymmetrisch in $(x_i, x_j)$, so kann es keine Funktionsstelle $\epsilon \in \{0,1\}^n$ geben mit

$$\epsilon \in D(f) \text{ und } \sigma_{ij}(\epsilon) \in D(f) \text{ und } f(\epsilon) \neq f(\sigma_{ij}(\epsilon))$$

oder

$$\epsilon \in D(f) \text{ und } \mu_{ij}(\epsilon) \in D(f) \text{ und } f(\epsilon) \neq f(\mu_{ij}(\epsilon)).$$

Aus diesen Gründen ist die Funktion $g \in B_n$, $g \in Erw(f)$ mit

$$g(\epsilon) = \begin{cases} 1 & \text{falls } f(\epsilon) = 1 \\ 0 & \text{falls } f(\epsilon) = 0 \\ f(\mu_{ij}(\epsilon)) & \text{falls } f(\epsilon) = * \text{ und } f(\mu_{ij}(\epsilon)) \neq * \\ f(\sigma_{ij}(\epsilon)) & \text{falls } f(\epsilon) = * \text{ und } f(\sigma_{ij}(\epsilon)) \neq * \\ 1 & \text{sonst} \end{cases}$$

wohldefiniert und darüber hinaus stark vertauschungssymmetrisch und stark äquivalenzsymmetrisch in $(x_i, x_j)$.
Damit ist $g$ auch stark multiformsymmetrisch in $(x_i, x_j)$.

□

Die Multiformsymmetrie bildet in ihrer starken Form ebenfalls eine Äquivalenzrelation auf $X$:

**Lemma 2.11** *Ist $f \in BP_n$ eine partielle Boolesche Funktion und $X$ eine Variablenmenge, dann bildet die starke Multiformsymmetrie für $f$ eine Äquivalenzrelation auf der Variablenmenge.*

**Beweis:**

- Reflexivität und Symmetrie:
  Die beiden Eigenschaften ergeben sich direkt aus der Reflexivität und Symmetrie der entsprechenden starken Symmetrien (Vertauschungssymmetrie, Äquivalenzsymmetrie).

- Transitivität:
  Seien $x_i, x_j, x_k \in X$ und ist $f$ stark multiformsymmetrisch in $(x_i, x_j)$ und $(x_j, x_k)$.
  Da die starke Vertauschungssymmetrie transitiv ist, liegt für $f$ auch eine starke Vertauschungssymmetrie in $(x_i, x_k)$ vor.
  Eine starke Vertauschungssymmetrie in $(x_i, x_j)$ und eine starke Äquivalenzsymmetrie in $(x_j, x_k)$ implizieren für $f$ nach Lemma 2.5 eine starke Äquivalenzsymmetrie in $(x_i, x_k)$.
  Damit ist $f$ auch stark multiformsymmetrisch in $(x_i, x_k)$ und die Relation demnach transitiv.

□

Wie jede Äquivalenzrelation, so zerlegt auch die starke Symmetrie und die starke Multiformsymmetrie eine Variablenmenge in disjunkte stark (multiform)symmetrische Teilmengen. Die Partitionen, die sich dadurch ergeben, werden mit $Part_f^{sym}$ bzw. $Part_f^{mul}$ bezeichnet. Es sollen nun die Elemente dieser beiden Partitionen miteinander verglichen werden.
Zunächst erhält man aus der Definition der Symmetrie und der Multiformsymmetrie:

**Korollar 2.9** *Jede partielle Boolesche Funktion, die (stark) multiformsymmetrisch in einem Variablenpaar ist, ist auch (stark) symmetrisch in diesem Paar.*

Damit ist jedes Element aus $Part_f^{mul}$ zumindest als Teilmenge einer Symmetriemenge aus $Part_f^{sym}$ enthalten.
Das folgende Lemma zeigt, daß in einer starken Symmetriemenge entweder alle Variablenpaare multiformsymmetrisch sind oder keines.

**Lemma 2.12** *Sei $f \in BP_n$ eine partielle Boolesche Funktion, $X = \{x_1, \ldots, x_n\}$ eine Variablenmenge und $\lambda \subseteq X$ eine starke Symmetriemenge von $f$. Sind $x_i, x_j \in \lambda$ verschiedene Variablen, dann gilt:*
*$f$ ist stark multiformsymmetrisch in $\lambda \iff f$ ist stark multiformsymmetrisch in $(x_i, x_j)$.*

**Beweis:** „$\Longrightarrow$ “
Ergibt sich aus der Definition der Multiformsymmetrie.
„$\Longleftarrow$ “
Man zeigt, daß $f$ stark multiformsymmetrisch in $(x_i, x_l)$ für alle $x_l \in \lambda$ ist. Aufgrund der Transitivität der Multiformsymmetrie ist $f$ auch stark multiformsymmetrisch in $\lambda$.
Da $f$ stark symmetrisch in $\lambda$ ist, existiert auch eine Symmetrie in $(x_i, x_l)$. In Abhängigkeit dieser Symmetrie muß unterschieden werden:
<u>Fall 1:</u> $f$ ist stark vertauschungssymmetrisch in $(x_i, x_l)$.
Da $f$ stark vertauschungssymmetrisch in $(x_i, x_j)$ ist, gibt es auch eine starke Vertauschungssymmetrie in $(x_j, x_l)$.
Da $f$ stark äquivalenzsymmetrisch in $(x_i, x_j)$ ist, folgt mit Lemma 2.5, daß $f$ auch stark äquivalenzsymmetrisch in $(x_i, x_l)$ und damit stark multiformsymmetrisch in diesem Paar ist.
<u>Fall 2 :</u> $f$ ist stark äquivalenzsymmetrisch in $(x_i, x_l)$.
Da $f$ stark äquivalenzsymmetrisch in $(x_i, x_j)$ ist, gibt es wegen Lemma 2.5 eine starke Vertauschungssymmetrie in $(x_j, x_l)$. Da $f$ stark vertauschungssymmetrisch in $(x_i, x_j)$ ist, folgt aufgrund der Transitivität der starken Vertauschungssymmetrie auch eine starke Vertauschungssymmetrie in $(x_i, x_l)$.
$f$ ist also stark multiformsymmetrisch in $(x_i, x_l)$.

□

Das Lemma besagt also, daß jede starke Symmetriemenge der Partition $Part_f^{sym}$, die ein stark multiformsymmetrisches Variablenpaar enthält, auch gleichzeitig eine starke multiformsymmetrische Menge ist. Damit läßt sich eine besondere Beziehung der beiden Partitionen $Part_f^{sym}$ und $Part_f^{mul}$ nachweisen:

**Lemma 2.13** *Sei $f \in BP_n$ eine partielle Boolesche Funktion. Dann gilt:*

$$\lambda \in Part_f^{mul} \Rightarrow \lambda \in Part_f^{sym}, \text{ falls } \mid \lambda \mid \geq 2.$$

**Beweis:** Ist $\lambda \in Part_f^{mul}$, dann gilt wegen Korollar 2.9 $\lambda \subseteq \lambda'$ für ein $\lambda' \in Part_f^{sym}$. Da $\mid \lambda \mid \geq 2$, impliziert Lemma 2.12 $\lambda \supseteq \lambda'$ und somit $\lambda = \lambda'$.
□

Bei einer Zerlegung der Variablenmenge in starke Symmetriemengen treten also gerade zwei Typen von Symmetriemengen auf: Zum einen solche Mengen, in denen alle Variablenpaare multiformsymmetrisch sind und zum anderen Symmetriemengen, in denen überhaupt keine multiformsymmetrischen Paare enthalten sind.

Um eine genauere Kenntnis des Aufbaus von multiformsymmetrischen Funktionen zu erhalten, ist man wiederum an den verschiedenen Orbits der entsprechenden $G-$Symmetrie interessiert. Dazu wird eine der Multiformsymmetrie entsprechende Gruppe $G \subseteq \mathrm{P_n}$ angegeben. Dies soll in zwei Schritten geschehen. Zunächst wird eine Abbildung aus der Menge $\mathrm{P_n}$ vorgestellt, die der paarweisen Multiformsymmetrie entspricht:

**Lemma 2.14** *Sei $f \in BP_n$ eine partielle Boolesche Funktion, $X$ eine Variablenmenge und $x_i, x_j$ zwei verschiedene Variablen aus $X$. Dann gilt:*
*$f$ ist (stark) multiformsymmetrisch in $(x_i, x_j)$ genau dann, falls $f$ (stark) $\{\nu_{ij} := \nu_i \circ \nu_j\}-$symmetrisch ist.*

**Beweis:** „$\Longrightarrow$“
Ist $f$ multiformsymmetrisch in $(x_i, x_j)$, dann existiert eine Erweiterung $g \in Erw(f)$, welche stark multiformsymmetrisch in $(x_i, x_j)$ ist.
Sei $\epsilon \in \{0,1\}^n$ beliebig. Man unterscheidet zwei Fälle:
<u>Fall 1:</u> $\epsilon_i = \epsilon_j$
In diesem Fall ist offensichtlich $\mu_{ij}(\epsilon) = \epsilon' = \nu_{ij}(\epsilon)$.
Da $g$ stark multiformsymmetrisch in $(x_i, x_j)$ ist, gilt $g(\epsilon) = g(\epsilon')$.
<u>Fall 2:</u> $\epsilon_i \neq \epsilon_j$
In diesem Fall ist offensichtlich $\sigma_{ij}(\epsilon) = \epsilon' = \nu_{ij}(\epsilon)$.
Da $g$ stark multiformsymmetrisch in $(x_i, x_j)$ ist, gilt $g(\epsilon) = g(\epsilon')$. In jedem Fall ist $g$ invariant gegenüber $\nu_{ij}$ und damit auch stark $\{\nu_{ij}\}-$symmetrisch.
Falls $f$ bereits stark multiformsymmetrisch ist, dann gilt $f = g$, und $f$ ist somit auch stark $\{\nu_{ij}\}-$symmetrisch.
„$\Longleftarrow$“
Ist $f$ $\{\nu_{ij}\}-$symmetrisch, dann existiert eine Erweiterung $g \in Erw(f)$, welche stark $\{\nu_{ij}\}-$symmetrisch ist.

Sei $\epsilon \in \{0,1\}^n$ beliebig. Man unterscheidet zwei Fälle:
Fall 1: $\epsilon_i = \epsilon_j$
In diesem Fall ist $\mu_{ij}(\epsilon) = \nu_{ij}(\epsilon)$ und $\sigma_{ij}(\epsilon) = \epsilon$.
Da $g$ invariant gegenüber der Abbildung $\nu_{ij}$ ist, gilt $g(\epsilon) = g(\sigma_{ij}(\epsilon)) = g(\mu_{ij}(\epsilon))$.
Fall 2: $\epsilon_i \neq \epsilon_j$
In diesem Fall ist $\sigma_{ij}(\epsilon) = \nu_{ij}(\epsilon)$ und $\mu_{ij}(\epsilon) = \epsilon$.
Da $g$ invariant gegenüber der Abbildung $\nu_{ij}$ ist, gilt $g(\epsilon) = g(\mu_{ij}(\epsilon)) = g(\sigma_{ij}(\epsilon))$.
In jedem Fall ist $g$ stark multiformsymmetrisch in $(x_i, x_j)$.
Falls $f$ bereits stark $\{\nu_{ij}\}-$symmetrisch ist, dann gilt $f = g$, und $f$ ist somit auch stark multiformsymmetrisch in $(x_i, x_j)$.

□

Damit ergibt sich:

**Korollar 2.10**
*Eine partielle Boolesche Funktion $f \in BP_n$ ist genau dann stark multiformsymmetrisch in $(x_i, x_j)$, $i < j$, falls für alle $\epsilon \in \{0,1\}^n$ gilt:*

$$f(\epsilon_1, \ldots, \epsilon_i, \ldots, \epsilon_j, \ldots, \epsilon_n) = f(\epsilon_1, \ldots, \overline{\epsilon_i}, \ldots, \overline{\epsilon_j}, \ldots, \epsilon_n).$$

Im nächsten Schritt wird eine Gruppe $G \subseteq \mathrm{P_n}$ angegeben, welche die Multiformsymmetrie auch auf Partitionen der Variablenmenge betrachtet:

**Bezeichnung 13** *Sei $X = \{x_1, \ldots, x_n\}$ eine Variablenmenge und $\Pi = \{\lambda_1, \ldots, \lambda_k\}$ eine Partition von $X$. Dann ist $G_\Pi^{mul}$ wie folgt festgelegt:*

$$G_\Pi^{mul} := < \bigcup_{\substack{x_i, x_j \in \lambda_l \\ 1 \leq l \leq k}} \{\nu_{ij}\} >$$

Die Menge $G_\Pi^{mul}$ ist genau die gesuchte Gruppe für die Multiformsymmetrie:

**Satz 2.4** *Sei $f \in BP_n$ eine partielle Boolesche Funktion, $X$ eine Variablenmenge und $\Pi$ eine Partition von $X$. Dann gilt:*
*$f$ ist (stark) multiformsymmetrisch in $\Pi$ $\iff$ $f$ (stark) $G_\Pi^{mul}-$symmetrisch.*

**Beweis:** Sei $\Pi = \{\lambda_1, \ldots, \lambda_k\}$ und $G := G_\Pi^{mul}$.
„$\Longrightarrow$ "
Ist $f$ symmetrisch in $\Pi$, dann gibt es eine Erweiterung $g \in Erw(f)$, die stark symmetrisch in $\Pi$ ist.
Behauptung: $g$ ist stark $G-$symmetrisch.

Die Menge $E := \bigcup_{\substack{x_i, x_j \in \lambda_l \\ 1 \leq l \leq k}} \{\nu_{ij}\}$ bildet ein Erzeugendensystem von $G$.

Nach Lemma 2.14 ist $g$ invariant gegenüber allen Abbildungen aus $E$ und damit auch stark $G-$symmetrisch und $f$ $G-$symmetrisch.
Ist $f$ bereits stark multiformsymmetrisch in $\Pi$, dann gilt $g = f$, und $f$ ist somit stark $G-$symmetrisch.

„$\Longleftarrow$ “
Ist $f$ $G-$symmetrisch, dann existiert eine Erweiterung $g \in Erw(f)$, die stark $G-$symmetrisch ist.
Wegen $\bigcup\limits_{\substack{x_i,x_j \in \lambda_l \\ 1 \leq l \leq k}} \{\nu_{ij}\} \subseteq G$ ist nach Lemma 2.14 $g$ stark multiformsymmetrisch in allen Variablenpaaren $(x_i, x_j)$ mit $x_i, x_j \in \lambda_l$, $1 \leq l \leq k$.
Damit ist $g$ stark multiformsymmetrisch in $\Pi$ und $f$ multiformsymmetrisch in $\Pi$.
Ist $f$ bereits stark $G-$symmetrisch, dann gilt $g = f$, und $f$ ist somit auch stark multiformsymmetrisch in $\Pi$.

$\square$

Damit ist auch die Multiformsymmetrie vollständig als spezielle $G-$Symmetrie eingeordnet. Es erfolgt nun die Charakterisierung der entsprechenden Orbits. Dazu werden mehrere Gewichtsklassen in einer Menge zusammengefaßt:

**Bezeichnung 14** *Für eine Variablenmenge $X$, eine Phasenabbildung $p$ auf $X$, eine Partition $\Pi = \{\lambda_1, \ldots, \lambda_k\}$ und einen Booleschen Vektor $\epsilon \in \{0,1\}^k$ sei*

$$C^{\Pi,p}_{[\epsilon_1],\ldots,[\epsilon_k]} := \bigcup_{\substack{w_i = \epsilon_i \, mod\, 2 \\ 1 \leq i \leq k}} C^{\Pi,p}_{w_1,\ldots,w_k}.$$

Das folgende Lemma charakterisiert die Relation $\leftrightarrow_G$ für $G = G^{mul}_{\Pi}$ näher:

**Lemma 2.15** *Sei $X$ eine Variablenmenge, $p$ eine Phasenabbildung auf $X$, $\Pi$ eine Partition von $X$. Ist $G = G^{mul}_{\Pi}$, dann gilt für $\epsilon, \epsilon' \in \{0,1\}^n$ und $\lambda \in \Pi$:*

$$\mid \epsilon \mid^p_\lambda \, mod\, 2 = \mid \epsilon' \mid^p_\lambda \, mod\, 2 \wedge \epsilon_i = \epsilon'_i \text{ für } i \notin Ind(\lambda) \Rightarrow \epsilon \leftrightarrow_G \epsilon'.$$

**Beweis:** Man beweist die Aussage mittels Induktion über $\mid \lambda \mid$.

Induktionsanfang:
Für $\mid \lambda \mid = 1$ ist $\epsilon = \epsilon'$. Die Behauptung gilt, da $G$ eine Gruppe bildet und demnach die Identität $id$ enthält.
Sei $\mid \lambda \mid = 2$ und $\lambda = \{x_i, x_j\}$.
Es können zwei Fälle unterschieden werden:
<u>Fall 1:</u> $\mid \epsilon \mid^p_\lambda \, mod\, 2 = 0$, d.h. $\mid \epsilon \mid^p_\lambda, \mid \epsilon' \mid^p_\lambda \in \{0, 2\}$.
Dann ist entweder $\epsilon = \epsilon'$ und damit $\epsilon \leftrightarrow_G \epsilon'$ oder o.B.d.A. $\epsilon_i = p(x_i), \epsilon_j = p(x_j)$ und $\epsilon'_i \neq p(x_i), \epsilon'_j \neq p(x_j)$.
In diesem Fall ist $\nu_{ij}(\epsilon) = \epsilon'$ mit $\nu_{ij} \in G$.
<u>Fall 2:</u> $\mid \epsilon \mid^p_\lambda \, mod\, 2 = 1$, d.h. $\mid \epsilon \mid^p_\lambda = \mid \epsilon' \mid^p_\lambda = 1$.
Dann ist entweder $\epsilon = \epsilon'$ und damit $\epsilon \leftrightarrow_G \epsilon'$ oder o.B.d.A. $\epsilon_i = p(x_i), \epsilon_j \neq p(x_j)$ und $\epsilon'_i \neq p(x_i), \epsilon'_j = p(x_j)$.
In diesem Fall ist $\nu_{ij}(\epsilon) = \epsilon'$ mit $\nu_{ij} \in G$.

Induktionsschritt: $\mid \lambda \mid > 2$
Sei $x_i \in \lambda$, $\lambda' = \lambda \backslash \{x_i\}$ und $\Pi' = \Pi \backslash \lambda \bigcup \{\lambda', \{x_i\}\}$, dann können wiederum zwei Fälle unterschieden werden:

Fall 1: $\epsilon_i = \epsilon'_i$
Unabhängig von $p$ ist $\mid \epsilon \mid^p_{\lambda'} = \mid \epsilon' \mid^p_{\lambda'} \ mod\,2$.
Nach der Induktionsvoraussetzung gibt es ein $\rho \in G^{mul}_{\Pi'} \subseteq G^{mul}_{\Pi}$ mit $\epsilon' = \rho(\epsilon)$, d.h. $\epsilon \leftrightarrow_G \epsilon'$.
Fall 2: $\epsilon_i \neq \epsilon'_i$
Unabhängig von $p$ ist $\mid \epsilon \mid^p_{\lambda'} \ mod\,2 \neq \mid \epsilon' \mid^p_{\lambda'} \ mod\,2$.
Sei $\nu_{ij} \in G$ mit $j \in Ind(\lambda')$ beliebig und $\nu_{ij}(\epsilon) = \epsilon''$. Dann gilt offenbar $\epsilon''_i = \epsilon'_i$ und $\mid \epsilon'' \mid^p_{\lambda'} \ mod\,2 = \mid \epsilon' \mid^p_{\lambda'} \ mod\,2$.
Daher gibt es nach Induktionsvoraussetzung eine weitere Abbildung $\rho' \in G^{mul}_{\Pi'} \subseteq G^{mul}_{\Pi}$ mit $\epsilon' = \rho'(\epsilon'')$. Bildet man $\rho = \rho' \circ \nu_{ij}$, dann gilt $\rho(\epsilon) = \epsilon'$ und $\rho \in G$.
□

Mit Hilfe dieses Lemmas lassen sich nun die Orbits für die Gruppe $G^{mul}_{\Pi}$ exakt bestimmen:

**Satz 2.5** *Sei $X$ eine Variablenmenge, $p$ eine Phasenabbildung auf $X$ und $\Pi = \{\lambda_1, \ldots, \lambda_k\}$ eine Partition von $X$. Ist $G = G^{mul}_{\Pi}$, dann gilt:*

$$orbit_G(\epsilon) = C^{\Pi,p}_{[|\epsilon|^p_{\lambda_1} \ mod\,2],\ldots,[|\epsilon|^p_{\lambda_k} \ mod\,2]}.$$

**Beweis:** Der Beweis des Satzes wird durch die beiden Inklusionen $\subseteq$ und $\supseteq$ unterteilt.
„$\subseteq$"
Die Menge $E := \bigcup_{\substack{x_i, x_j \in \lambda_l \\ 1 \leq l \leq k}} \{\nu_{ij}\}$ bildet ein Erzeugendensystem von $G$.
Sei $\epsilon \in \{0,1\}^n$ beliebig.
Dann sind $x_i, x_j \in \lambda_r$ für ein $r \in \{1, \ldots, k\}$ und
$\nu_{ij}(\epsilon_1, \ldots, \epsilon_i, \ldots, \epsilon_j, \ldots, \epsilon_n) = (\epsilon_1, \ldots, \overline{\epsilon_i}, \ldots, \overline{\epsilon_j}, \ldots, \epsilon_n)$.
Somit gilt:

$$\begin{aligned}
&\mid (\epsilon_1, \ldots, \epsilon_i, \ldots, \epsilon_j, \ldots, \epsilon_n) \mid^p_{\lambda_r} \ mod\,2 \\
&= \Big(\sum_{l \in Ind(\lambda_r)} 1 - \mid \epsilon_l - p(x_l) \mid\Big) \, mod\,2 \\
&= \Big(\Big(\sum_{l \in Ind(\lambda_r) \setminus \{i,j\}} 1 - \mid \epsilon_l - p(x_l) \mid\Big) \, mod\,2 + \\
&\qquad (2 - \mid \epsilon_i - p(x_i) \mid - \mid \epsilon_j - p(x_j) \mid) \, mod\,2\Big) \, mod\,2 \\
&= \Big(\Big(\sum_{l \in Ind(\lambda_r) \setminus \{i,j\}} 1 - \mid \epsilon_l - p(x_l) \mid\Big) \, mod\,2 + \\
&\qquad (2 - \mid \overline{\epsilon_i} - p(x_i) \mid - \mid \overline{\epsilon_j} - p(x_j) \mid) \, mod\,2\Big) \, mod\,2, \\
&\ \text{da } \mid \epsilon_l - p(x_l) \mid = 1 - \mid \overline{\epsilon_l} - p(x_l) \mid \ (l \in \{i,j\}) \text{ und} \\
&\ 2 - (b_1 + b_2) \, mod\,2 = (b_1 + b_2) \, mod\,2 \ (b_1, b_2 \in \{0,1\}) \\
&= \mid (\epsilon_1, \ldots, \overline{\epsilon_i}, \ldots, \overline{\epsilon_j}, \ldots, \epsilon_n) \mid^p_{\lambda_r} \ mod\,2.
\end{aligned}$$

Da $\mid \epsilon \mid^p_{\lambda_s} = \mid \nu_{ij}(\epsilon) \mid^p_{\lambda_s}$ für $s \neq r$, gilt $\nu_{ij}(\epsilon) \in C^{\Pi,p}_{[|\epsilon|^p_{\lambda_1} \ mod\,2],\ldots,[|\epsilon|^p_{\lambda_k} \ mod\,2]}$.
Da $E$ ein Erzeugendensystem von $G$ bildet, ist

$orbit_G(\epsilon) \subseteq C^{\Pi,p}_{[|\epsilon|^p_{\lambda_1} \, mod\, 2],\ldots,[|\epsilon|^p_{\lambda_k} \, mod\, 2]}$ gezeigt.
„$\supseteq$"
Man konstruiert zu zwei beliebigen Booleschen Vektoren $\epsilon, \epsilon'$, welche sich in der gleichen Menge von Gewichtsklassen befinden, eine Abbildung $\rho \in G$, so daß $\rho(\epsilon) = \epsilon'$.
Dazu bildet man für alle $\lambda_l$ $(1 \leq l \leq k)$ mittels der Konstruktion aus dem Beweis zu Lemma 2.15 die Abbildungen $\rho_l \in G$, so daß $\rho_l(\epsilon)_i = \epsilon'_i$ für $i \in Ind(\lambda_l)$. Da die Abbildungen aus $\rho_l$ nur die Komponenten $\epsilon_i$ mit $i \in Ind(\lambda_l)$ bearbeiten und $\Pi$ eine Partition von $X$ bildet, sind diese Abbildungen unabhängig voneinander und können in beliebiger Reihenfolge hintereinander ausgeführt werden. Ist $\rho = \rho_1 \circ \ldots \circ \rho_k$, dann ist $\rho \in G$ und offensichtlich $\rho(\epsilon) = \epsilon'$.
Somit gilt $orbit_G(\epsilon) \supseteq C^{\Pi,p}_{[|\epsilon|^p_{\lambda_1} \, mod\, 2],\ldots,[|\epsilon|^p_{\lambda_k} \, mod\, 2]}$.

□

Da die Gruppe $G^{mul}_{\Pi}$ die Booleschen Vektoren unabhängig von einer Phasenabbildung partitioniert, folgt aus Satz 2.5:

**Korollar 2.11** *Sei $f \in BP_n$ eine partielle Boolesche Funktion, $X$ eine Variablenmenge, $\Pi = \{\lambda_1, \ldots, \lambda_k\}$ eine Partition von $X$. Dann gilt:*
*(1) $f$ ist stark multiformsymmetrisch in $\Pi$ genau dann, falls für alle Phasenabbildungen $p : X \to \{0,1\}$ gilt:*

$$\forall \epsilon \in \{0,1\}^k \qquad | f(C^{\Pi,p}_{[\epsilon_1],\ldots,[\epsilon_k]}) | = 1.$$

*(2) $f$ ist schwach multiformsymmetrisch in $\Pi$ genau dann, falls für alle Phasenabbildungen $p : X \to \{0,1\}$ gilt:*

$$\forall \epsilon \in \{0,1\}^k \qquad \{0,1\} \not\subseteq f(C^{\Pi,p}_{[\epsilon_1],\ldots,[\epsilon_k]}).$$

Für die Anzahl der Orbits bzgl. der Multiformsymmetrie folgt aus Korollar 2.6:

**Korollar 2.12** *Sei $f \in BP_n$ eine partielle Boolesche Funktion, $X$ eine Variablenmenge und $\Pi = \{\lambda_1, \ldots, \lambda_k\}$ eine Partition von $X$. Ist $f$ stark multiformsymmetrisch in $\Pi$, dann gibt es genau $2^k$ verschiedene Orbits bzgl. $G = G^{mul}_{\Pi}$.*

#### 2.2.3.1 Besondere Eigenschaften der schwachen Multiformsymmetrie

Das folgende Lemma stellt ähnlich zu Lemma 2.9 die grundlegenden Unterschiede zwischen der starken und der schwachen Multiformsymmetrie dar:

**Lemma 2.16** *Sei $f \in BP_n$ eine partielle Boolesche Funktion und $X$ eine Variablenmenge.*

- *(1) Seien $x_i, x_j, x_k \in X$ und ist $f$ schwach multiformsymmetrisch in $(x_i, x_j)$ und in $(x_j, x_k)$, dann ist $f$ nicht notwendigerweise schwach multiformsymmetrisch in $(x_i, x_k)$.*

- *(2) Sei $\lambda \subseteq X$. Ist $f$ schwach multiformsymmetrisch in allen Paaren $(x_i, x_j) \in \lambda \times \lambda$, dann ist $f$ nicht notwendigerweise schwach multiformsymmetrisch in $\lambda$.*

- *(3) Sei $\Pi = \{\lambda_1, \ldots, \lambda_k\}$ eine Partition von $X$. Ist $f$ schwach multiformsymmetrisch in allen $\lambda_i \in \Pi$, dann ist $f$ nicht notwendigerweise auch schwach multiformsymmetrisch in $\Pi$.*

**Beweis:** Man gibt für (1) – (3) entsprechende Beispiele an:

zu (1)
Dies wurde schon im Beispiel auf Seite 50 gezeigt.

zu (2)
Sei $f \in BP_4$ wie folgt definiert:

$$f(\epsilon) = \begin{cases} 1 & \text{falls } \epsilon = (0,0,0,1) \\ 0 & \text{falls } \epsilon = (1,1,1,0) \\ * & \text{sonst} \end{cases}$$

$f$ ist offensichtlich multiformsymmetrisch in $(x_i, x_j)$, $i, j \in \{1,2,3,4\}$, aber nicht multiformsymmetrisch in $\{x_1, x_2, x_3, x_4\}$.

zu (3)
Sei $f \in BP_4$ wie folgt definiert:

$$f(\epsilon) = \begin{cases} 1 & \text{falls } \epsilon = (1,1,0,0) \\ 0 & \text{falls } \epsilon = (0,0,1,1) \\ * & \text{sonst} \end{cases}$$

$f$ ist multiformsymmetrisch in den Mengen $\{x_1, x_2\}$ und $\{x_3, x_4\}$. Ist $f'$ eine Erweiterung von $f$, die stark multiformsymmetrisch in $\{x_1, x_2\}$ (bzw. $\{x_3, x_4\}$) ist, dann gilt $f'(0,0,0,0) = 1$ (bzw. $f'(0,0,0,0) = 0$).
Daher ist $f$ nicht multiformsymmetrisch in $\{\{x_1, x_2\}, \{x_3, x_4\}\}$.

$\square$

Bei den Betrachtungen zur Symmetrie hatte sich herausgestellt, daß es nicht immer möglich ist, zwei beliebig große Symmetriemengen zu einer (schwachen) Symmetriemenge zu vereinigen, auch dann nicht, wenn es sich bei den beiden Mengen um starke Symmetriemengen handelt. Lediglich für den Spezialfall, daß mindestens eine der beteiligten Mengen einelementig ist, ließ sich ein Kriterium für den Zusammenschluß angeben (vgl. Satz 2.3).
Im Fall stark multiformsymmetrischer Mengen ist diese besondere Einschränkung allerdings nicht mehr notwendig. Um dies zu zeigen, ist noch eine Vorüberlegung notwendig:

**Lemma 2.17** *Sei $X$ eine Variablenmenge, $p$ eine Phasenabbildung auf $X$ und $\Pi = \{\lambda_1, \ldots, \lambda_k\}$ eine Partition von $X$. Seien weiter $\lambda_r, \lambda_s \in \Pi$ mit $|\lambda_r \cup \lambda_s| \geq 3$ und $x_i \in \lambda_r$ sowie $x_j \in \lambda_s$. Dann gilt für alle $\varepsilon \in \{0,1\}^k$ :*

$$(1) \quad \exists \epsilon \in C^{\Pi,p}_{[\varepsilon_1],\ldots,[\varepsilon_r],\ldots,[\varepsilon_s],\ldots,[\varepsilon_k]} \text{ mit } \sigma_{ij}(\epsilon) \in C^{\Pi,p}_{[\varepsilon_1],\ldots,[\overline{\varepsilon_r}],\ldots,[\overline{\varepsilon_s}],\ldots,[\varepsilon_k]}.$$

$$(2) \quad \exists \epsilon \in C^{\Pi,p}_{[\varepsilon_1],\ldots,[\varepsilon_r],\ldots,[\varepsilon_s],\ldots,[\varepsilon_k]} \text{ mit } \mu_{ij}(\epsilon) \in C^{\Pi,p}_{[\varepsilon_1],\ldots,[\overline{\varepsilon_r}],\ldots,[\overline{\varepsilon_s}],\ldots,[\varepsilon_k]}.$$

**Beweis:** O.B.d.A. $\mid \lambda_r \mid \geq 2$. Dann existiert eine von $x_i$ verschiedene Variable $x_q \in \lambda_r$.
Sei $\epsilon \in C^{\Pi,p}_{[\varepsilon_1],\ldots,[\varepsilon_r],\ldots,[\varepsilon_s],\ldots,[\varepsilon_k]}$ beliebig. Man unterscheidet zwei Fälle:
Fall 1: $\epsilon_i = \epsilon_j$
Dann ist $\mu_{ij}(\epsilon) \in C^{\Pi,p}_{[\varepsilon_1],\ldots,[\overline{\varepsilon_r}],\ldots,[\overline{\varepsilon_s}],\ldots,[\varepsilon_k]}$.
Betrachte nun $\nu_{iq}(\epsilon) = \epsilon'$.
Offensichtlich gilt auch $\epsilon' \in C^{\Pi,p}_{[\varepsilon_1],\ldots,[\varepsilon_r],\ldots,[\varepsilon_s],\ldots,[\varepsilon_k]}$ und $\epsilon'_i \neq \epsilon'_j$.
Daher ist $\sigma_{ij}(\epsilon') \in C^{\Pi,p}_{[\varepsilon_1],\ldots,[\overline{\varepsilon_r}],\ldots,[\overline{\varepsilon_s}],\ldots,[\varepsilon_k]}$.
Fall 2: $\epsilon_i \neq \epsilon_j$
Dann ist $\sigma_{ij}(\epsilon) \in C^{\Pi,p}_{[\varepsilon_1],\ldots,[\overline{\varepsilon_r}],\ldots,[\overline{\varepsilon_s}],\ldots,[\varepsilon_k]}$.
Betrachte nun $\nu_{iq}(\epsilon) = \epsilon'$.
Offensichtlich gilt auch $\epsilon' \in C^{\Pi,p}_{[\varepsilon_1],\ldots,[\varepsilon_r],\ldots,[\varepsilon_s],\ldots,[\varepsilon_k]}$ und $\epsilon'_i = \epsilon'_j$.
Daher ist $\mu_{ij}(\epsilon') \in C^{\Pi,p}_{[\varepsilon_1],\ldots,[\overline{\varepsilon_r}],\ldots,[\overline{\varepsilon_s}],\ldots,[\varepsilon_k]}$.

Damit sind die Aussagen (1) und (2) bewiesen.

□

Der folgende Satz besagt, daß sich zwei stark multiformsymmetrische Mengen $\lambda_1, \lambda_2$ stets zu einer multiformsymmetrischen Menge vereinigen lassen, falls eine Multiformsymmetrie in einem Variablenpaar aus $\lambda_1 \times \lambda_2$ existiert:

**Satz 2.6** *Sei $f \in BP_n$ eine partielle Funktion, die stark multiformsymmetrisch in der Partition $\Pi$ ist. Sind $[x_i], [x_j] \in \Pi$, und ist $f$ multiformsymmetrisch in $(x_i, x_j)$, dann ist $f$ auch multiformsymmetrisch in der Partition $\Pi' = \Pi \backslash \{[x_i], [x_j]\} \bigcup \{[x_i] \cup [x_j]\}$.*
*Ist $\mid [x_i] \cup [x_j] \mid \geq 3$, dann gilt die Behauptung auch, falls $f$ symmetrisch in $(x_i, x_j)$ ist.*

**Beweis:** O.B.d.A. $\Pi = \{\lambda_1, \ldots, \lambda_k\}$ und $\lambda_1 = [x_i], \lambda_2 = [x_j]$ mit $\lambda_1 \cap \lambda_2 = \emptyset$.
Dann ist $\Pi' = \{\lambda_1 \cup \lambda_2, \lambda_3, \ldots, \lambda_k\}$.
Sei $p : X \to \{0,1\}$ eine beliebige Phasenabbildung auf $X$.
Es ist also zu zeigen, daß es keine Menge $C^{\Pi',p}_{[\epsilon_2],\ldots,[\epsilon_k]}$ gibt mit $\{0,1\} \subseteq f(C^{\Pi',p}_{[\epsilon_2],\ldots,[\epsilon_k]})$.
$C^{\Pi',p}_{[\epsilon_2],\ldots,[\epsilon_k]}$ kann als disjunkte Vereinigung zweier Mengen von Gewichtsklassen dargestellt werden:

$$C^{\Pi',p}_{[\epsilon_2],[\epsilon_3],\ldots,[\epsilon_k]} = C^{\Pi,p}_{[0],[\epsilon_2],[\epsilon_3],\ldots,[\epsilon_k]} \cup C^{\Pi,p}_{[1],[\overline{\epsilon_2}],[\epsilon_3],\ldots,[\epsilon_k]}$$

Da $f$ stark multiformsymmetrisch in $\Pi$ ist, gilt nach Korollar 2.11
$\mid f(C^{\Pi,p}_{[0],[\epsilon_2],[\epsilon_3],\ldots,[\epsilon_k]}) \mid = \mid f(C^{\Pi,p}_{[1],[\overline{\epsilon_2}],[\epsilon_3],\ldots,[\epsilon_k]}) \mid = 1$.
Ist $\{0,1\} \subseteq f(C^{\Pi'}_{[\epsilon_2],\ldots,[\epsilon_k]})$, so kann nur $f(C^{\Pi,p}_{[0],[\epsilon_2],[\epsilon_3],\ldots,[\epsilon_k]}) = b$ und $f(C^{\Pi,p}_{[1],[\overline{\epsilon_2}],[\epsilon_3],\ldots,[\epsilon_k]}) = \overline{b}$ gelten ($b \in \{0,1\}$ ).
Dann existiert ein Boolescher Vektor $\epsilon \in C^{\Pi,p}_{[0],[\epsilon_2],[\epsilon_3],\ldots,[\epsilon_k]}$ mit $\nu_{ij}(\epsilon) \in C^{\Pi,p}_{[1],[\overline{\epsilon_2}],[\epsilon_3],\ldots,[\epsilon_k]}$.
Daher kann $f$ aber auch nicht multiformsymmetrisch in $(x_i, x_j)$ sein.
Dies ist aber ein Widerspruch zur Annahme.

In Abhängigkeit der Symmetrie für $f$ in $(x_i, x_j)$ unterscheidet man im Fall $\mid [x_i] \cup [x_j] \mid \geq 3$:

Fall 1: $f$ ist vertauschungssymmetrisch in $(x_i, x_j)$.
Da $\mid \lambda_1 \cup \lambda_2 \mid \geq 3$ sind die Voraussetzungen von Lemma 2.17 erfüllt.
Demnach existiert ein Boolescher Vektor $\epsilon \in C^{\Pi,p}_{[0],[\epsilon_2],[\epsilon_3],\ldots,[\epsilon_k]}$ mit $\sigma_{ij}(\epsilon) \in C^{\Pi,p}_{[1],[\overline{\epsilon_2}],[\epsilon_3],\ldots,[\epsilon_k]}$, d.h. $f$ ist nicht vertauschungssymmetrisch in $(x_i, x_j)$. Dies ist ein Widerspruch zur Annahme.

Fall 2: $f$ ist äquivalenzsymmetrisch in $(x_i, x_j)$.
Der Beweis verläuft analog zu Fall 1 mit Hilfe der Abbildung $\mu_{ij}$.

Dies bedeutet $\{0,1\} \not\subseteq f(C^{\Pi',p}_{[\epsilon_2],\ldots,[\epsilon_k]})$, d.h. $f$ ist somit multiformsymmetrisch in $\Pi'$.

□

### 2.2.4 Unabhängigkeit

Im vorigen Kapitel wurde bereits bei der Betrachtung von ROBDDs der Begriff der Unabhängigkeit einer Booleschen Funktion von einer Variablen eingeführt. In diesem Abschnitt soll zunächst gezeigt werden, daß auch die Unabhängigkeit eine besondere Form der $G-$Symmetrie darstellt.

**Definition 2.10 (Unabhängigkeit)**
*Sei $f \in BP_n$ eine partielle Boolesche Funktion, $X$ eine Variablenmenge und $x_i \in X$ eine Variable.*
*$f$ heißt* **(stark) unabhängig von** *$x_i$ genau dann, falls $f$ (stark) $\{\nu_i\}-$symmetrisch ist.*
*Ist $\lambda \subseteq X$ eine Menge von Variablen, dann heißt $f$* **stark unabhängig von** *$\lambda$ genau dann, falls $f$ von allen Variablen aus $\lambda$ stark unabhängig ist.*
*$f$ heißt* **(schwach) unabhängig von** *$\lambda$ genau dann, falls eine Erweiterung von $f$ existiert, die stark unabhängig von $\lambda$ ist.*
*$\lambda$ bezeichnet man als* **(stark) unabhängige Variablenmenge** *von $f$.*

Offensichtlich, ist diese Definition zumindest für die starke Unabhängigkeit äquivalent mit derjenigen aus dem vorigen Kapitel (vgl. Seite 35).
Das folgende Korollar ergibt sich unmittelbar aus der obigen Definition:

**Korollar 2.13** *Sei $f \in BP_n$ eine partielle Boolesche Funktion, $X$ eine Variablenmenge und $\lambda \subseteq X$. Für $G = < \{\nu_i \mid i \in Ind(\lambda)\} >$ gilt:*
*(1) $f$ ist (stark) unabhängig von $\lambda$ genau dann, wenn $f$ (stark) $G-$symmetrisch ist.*
*(2) Für alle $\epsilon \in \{0,1\}^n$ ist $orbit_G(\epsilon) = \{\epsilon' \in \{0,1\}^n \mid \epsilon'_i = \epsilon_i, i \in Ind(X \backslash \lambda)\}$.*

Das folgende Lemma zeigt, daß im allgemeinen nicht alle unabhängigen Variablen auch gleichzeitig eine unabhängige Variablenmenge bilden:

**Lemma 2.18** *Sei $f \in BP_n$ eine partielle Boolesche Funktion, $X$ eine Variablenmenge und $\lambda \subseteq X$. Ist $f$ unabhängig von allen Variablen aus $\lambda$, dann ist $f$ nicht notwendigerweise auch unabhängig von $\lambda$.*

**Beweis:** Betrachte folgende Funktion $f \in BP_2$:

| $x_1$ | $x_2$ | $f(x_1, x_2)$ |
|---|---|---|
| 0 | 0 | * |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | * |

$f$ ist offenbar unabhängig von $x_1$ und $x_2$. Es gibt aber keine Erweiterung von $f$, die von beiden Variablen unabhängig ist, da sich $f$ nicht zu einer Konstantenfunktion erweitern läßt.

□

Da bei der Realisierung einer Booleschen Funktion stark unabhängige Variablen keine Rolle spielen, ist man daher daran interessiert, möglichst große unabhängige Variablenmengen zu bestimmen. Auch für die Darstellung mittels ROBDDs scheint dies von Vorteil zu sein (s. Korollar 1.6).
Lemma 2.18 hat allerdings gezeigt, daß sich dabei ähnliche Probleme ergeben wie bei der Ermittlung von symmetrischen und multiformsymmetrischen Mengen bzw. Partitionen.
Stark unabhängige Mengen lassen sich jedoch jeweils um *eine* zusätzliche unabhängige Variable erweitern:

**Satz 2.7** *Sei $f \in BP_n$ eine partielle Boolesche Funktion, $X$ eine Variablenmenge und $\lambda \subseteq X$. Ist $f$ stark unabhängig von $\lambda$ und unabhängig von $x_i \notin \lambda$, dann ist $f$ auch unabhängig von $\lambda \cup \{x_i\}$.*

**Beweis:** Sei $\lambda' = \lambda \cup \{x_i\}$, $G =< \{\nu_l \mid l \in Ind(\lambda)\} >$ und $G' =< \{\nu_l \mid l \in Ind(\lambda')\} >$.
Es ist zu zeigen, daß $\{0,1\} \not\subseteq f(orbit_{G'}(\epsilon))$ für alle $\epsilon \in \{0,1\}^n$ .
Sei $\epsilon \in \{0,1\}^n$ beliebig.
$orbit_{G'}(\epsilon)$ kann als disjunkte Vereinigung zweier Orbits von $G$ dargestellt werden:

$$orbit_{G'}(\epsilon) = orbit_G(\epsilon) \cup orbit_G(\nu_i(\epsilon))$$

Da $f$ stark unabhängig von $\lambda$ ist, gilt $\mid f(orbit_G(\epsilon)) \mid = \mid f(orbit_G(\nu_i(\epsilon))) \mid = 1$.
Ist $\{0,1\} \subseteq f(orbit_{G'}(\epsilon))$, so kann nur $f(orbit_G(\epsilon)) = b$ und $f(orbit_G(\nu_i(\epsilon))) = \overline{b}$ gelten ($b \in \{0,1\}$ )
Da für alle Vektoren $\epsilon \in \{0,1\}^n$ auch $\epsilon \in orbit_G(\epsilon)$, kann $f$ nicht unabhängig von $x_i$ sein.
Dies ist aber ein Widerspruch zur Annahme. $f$ ist somit unabhängig von $\lambda'$.

□

Alle unabhängigen Variablen bilden auch gleichzeitig eine multiformsymmetrische Variablenmenge:

**Lemma 2.19** *Sei $f \in BP_n$ eine partielle Boolesche Funktion, $X$ eine Variablenmenge und $\lambda \subseteq X$. Ist $f$ (stark) unabhängig von $\lambda$, dann ist $f$ auch (stark) multiformsymmetrisch in $\lambda$.*

**Beweis:** Ist $f$ unabhängig von $\lambda$, dann existiert eine Erweiterung $g \in Erw(f)$, so daß $g$ unabhängig von $\lambda$ ist.
Es ist zu zeigen, daß $g$ stark multiformsymmetrisch in $\lambda$ ist.
Die Funktion $g$ ist invariant gegenüber jeder Abbildung $\nu_i \in \mathrm{P_n}$ mit $i \in Ind(\lambda)$ und somit auch invariant gegenüber den Kompositionen $\nu_{i,j} \in \mathrm{P_n}$ mit $i, j \in Ind(\lambda)$.
Damit ist $g$ stark multiformsymmetrisch in $\lambda$ und $f$ multiformsymmetrisch in $\lambda$.
Ist $f$ stark unabhängig von $\lambda$, dann gilt $f = g$ und $f$ ist somit stark multiformsymmetrisch in $\lambda$.

□

Das abschließende Lemma zeigt, wie sich starke Unabhängigkeiten auch über die starke Symmetrie bestimmen lassen:

**Lemma 2.20** *Sei $f \in BP_n$ eine partielle Boolesche Funktion, $X$ eine Variablenmenge und $x_i, x_j \in X$. Ist $f$ stark unabhängig von $x_i$, und stark symmetrisch in $(x_i, x_j)$, dann ist $f$ auch stark unabhängig von $x_j$.*

**Beweis:** In Abhängigkeit der Symmetrie unterscheidet man:
Fall 1: $f$ ist stark vertauschungssymmetrisch in $(x_i, x_j)$.
Es gilt:

$$f_{x_i x_j} = f_{\overline{x_i} x_j} = f_{x_i \overline{x_j}} \text{ und } f_{\overline{x_i} x_j} = f_{x_i \overline{x_j}} = f_{\overline{x_i}\,\overline{x_j}} \text{ , d.h.}$$

$f$ ist stark unabhängig von $x_j$.
Fall 2: $f$ ist stark äquivalenzsymmetrisch in $(x_i, x_j)$.
Kann analog zu Fall 1 gezeigt werden.

□

## 2.3 Effiziente Schaltkreise und Darstellungen symmetrischer Funktionen

In diesem Abschnitt soll exemplarisch gezeigt werden, wie sich die Eigenschaften spezieller $G-$Symmetrien ausnutzen lassen, um effiziente, d.h. kleine und schnelle Schaltungen sowie kompakte Darstellungen zu erhalten. Dies soll zur Motivation dienen, bei der Realisierung partieller Boolescher Funktionen gezielt nach (stark) $G-$symmetrischen Erweiterungen zu suchen.

Um die Komplexität der entsprechenden Realisierungen zu bestimmen, benötigt man folgendes Lemma aus [Weg89]:

**Lemma 2.21** *Sei $X = \{x_1, \ldots, x_n\}$ eine Variablenmenge und $VM$ die Menge aller vollständigen Monome über $X$. Für eine Teilmenge $M = \{m_1, \ldots, m_l\} \subseteq VM$ sei die Boolesche Funktion $f_M \in B_{n,l}$ durch $f_i = f_{m_i}$ $(1 \leq i \leq l)$ gegeben. Es gilt:*
*(1) $f_{VM}$ kann durch einen $B_2-$Schaltkreis mit $2 \cdot 2^n$ Grundgattern und Tiefe $\lceil \log n \rceil$ realisiert werden.*
*(2) $f_M$ kann durch einen $B_2-$Schaltkreis mit $l + O(2^{n/2})$ Grundgattern und Tiefe $\lceil \log n \rceil$ realisiert werden.*

**Beweis:** Zu(1)
Es werden alle vollständigen Monome über $\{x_1, \ldots x_{\lceil n/2 \rceil}\}$ und alle Monome über $\{x_{\lceil n/2 \rceil + 1}, \ldots, x_n\}$ parallel realisiert. Danach lassen sich alle Teilfunktionen von $f$ parallel mit je einem $\wedge$-Gatter berechnen. Die Aussage über die Tiefe folgt unmittelbar. Die Aussage über die Zahl der Grundgatter folgt für $n \leq 5$ durch Betrachtung der einzelnen Fälle und für $n \geq 6$ induktiv, da

$$2^n + 2 \cdot 2^{\lceil n/2 \rceil} + 2 \cdot 2^{\lfloor n/2 \rfloor} \leq 2^n + 2 \cdot 2^{(n+1)/2} + 2 \cdot 2^{(n-1)/2}.$$
$$= 2^n(1 + 2^{-n/2}(2 \cdot 2^{1/2} + 2 \cdot 2^{-1/2})) \leq 2 \cdot 2^n$$

Zu (2)
Es werden nach (1) alle Monome über $\{x_1, \ldots x_{\lceil n/2 \rceil}\}$ und alle Monome über $\{x_{\lceil n/2 \rceil + 1}, \ldots, x_n\}$ mit $2 \cdot (2^{\lceil n/2 \rceil} + 2^{\lfloor n/2 \rfloor}) = O(2^{n/2})$ Grundgattern realisiert. Danach genügt für jede Teilfunktion von $f_M$ ein weiteres $\wedge$-Gatter.

□

Gemäß Korollar 2.1 sind alle Funktionswerte von $f$ für die Funktionsstellen aus einem Orbit identisch, falls die Funktion stark $G-$symmetrisch ist, d.h. $f(\epsilon)$ ist eindeutig durch $orbit_G(\epsilon)$ bestimmt. Es ist daher sinnvoll, eine Informationsreduktion vorzunehmen. Diese soll durch die totale Boolesche Funktion $f_{orbit_G}$ berechnet werden. Sind alle Orbits bzgl. einer Gruppe $G$ durchnumeriert, dann liefert $f_{orbit_G}(\epsilon)$ gerade die Binärdarstellung von $orbit_G(\epsilon)$ in dieser Aufzählung. Mit Hilfe dieser Funktion können $G-$symmetrische Funktionen effizient realisiert werden:

**Lemma 2.22** *Sei $f \in B_n$ eine totale Boolesche Funktion und $G \subseteq \mathrm{P_n}$ eine Gruppe. Ist $f$ stark $G-$symmetrisch, dann gilt für die Komplexität der Funktion*

$$C_{B_2}(f) \leq C_{B_2}(f_{orbit_G}) + 2 \cdot k + O(\sqrt{k})$$

*wobei $k = \mid \{orbit_G(\epsilon) \mid \epsilon \in \{0,1\}^n\} \mid$ die Anzahl der der Orbits bzgl. $G$ ist. Außerdem existiert eine Realisierung $S$ für $f$ mit $T_{B_2}(S) \leq T_{B_2}(S_G) + \lceil \log k \rceil + \lceil \log \lceil \log k \rceil \rceil$, falls $S_G$ eine Realisierung für $f_{orbit_G}$ ist.*

**Beweis:** Es wird eine Realisierung $S$ für $f$ mit entsprechender Komplexität und Tiefe angegeben.
$S$ besteht aus drei Stufen. In der ersten Stufe wird die Funktion $f_{orbit_G}$ berechnet. Die Schaltung für diese Funktion besitzt genau

$\lceil \log k \rceil$ Ausgänge, um $k$ verschiedene Ergebnisse darzustellen.
In einer zweiten Stufe werden für jeden Orbit mit $f(orbit_G) = 1$ mittels $\wedge$-Gatter die entsprechenden Ausgänge der ersten Stufe zusammengefaßt. Faßt man die Ausgänge von $f_{orbit_G}$ als Variablenmenge auf, so ergibt sich die Aufgabe, entsprechende Monome über dieser Menge zu realisieren.
Nach Lemma 2.21 kann die entsprechende Schaltung mit $k + O(2^{\log k/2}) = k + O(\sqrt{k})$ Grundgattern und Tiefe $\lceil \log \lceil \log k \rceil \rceil$ realisiert werden.
In der dritten Stufe werden alle Ausgänge der zweiten Stufe mittels $\vee$-Gatter in einem balancierten Baum zu einem Ausgang zusammengefaßt. Dieser Schaltkreis besitzt höchstens $k$ Gatter und Tiefe $\lceil \log k \rceil$.
Betrachtet man den so konstruierten Schaltkreis in seiner Größe und Tiefe, so ergibt sich die gewünschte Aussage.

□

Die Komplexität der im vorigen Lemma konstruierten Schaltung hängt hauptsächlich von der Realisierung für $f_{orbit_G}$ ab, die wiederum von $G$ bestimmt wird. Für spezielle Gruppen kann auch eine effiziente Realisierung von $f_{orbit_G}$ angegeben werden. Im folgenden soll dies für totalsymmetrische Funktionen und multiformsymmetrische Funktionen vorgeführt werden.

**Satz 2.8** *Jede totalsymmetrische Funktion $f \in B_n$ kann durch einen Schaltkreis $S$ mit Komplexität*

$$C_{B_2}(S) \leq c \cdot n + O(\sqrt{n}) \qquad \text{für ein } c \in \mathbf{R}$$

*und Tiefe*

$$T_{B_2}(S) \leq c' \cdot \log n + O(\log \log n) \qquad \text{für ein } c' \in \mathbf{R}$$

*realisiert werden.*

**Beweis:** Da $f$ totalsymmetrisch ist, entsprechen die Orbits allen Vektoren mit einem identischen $\lambda-$Gewicht. Dabei ist $\lambda = X$, falls $X = \{x_1, \ldots, x_n\}$ eine $n$ elementige Variablenmenge ist. Offensichtlich gibt es genau $n+1$ verschiedene Orbits.
Für die Berechnung von $f_{orbit_G}$ wird die Funktion $BSUM_n \in B_{n,\lceil \log(n+1) \rceil}$ realisiert. Dabei berechnet $BSUM_n$ aus der Eingabe $(\epsilon_1, \ldots, \epsilon_n)$ die Binärdarstellung von $| \epsilon |_X^p$. Die Abbildung $p$ ist dabei eine feste von $f$ auf $X$ induzierte Phasenabbildung.
Die Realisierung von $BSUM_n$ kann mittels eines Wallace-Tree Addierers [Wall64] und für die beiden letzten Summanden durch einen schnellen Addierer (z.B. Ladner-Fischer Addierer [LF80]) erfolgen. Eine entsprechende Konstruktion mit der Größe $\frac{25}{3}n + O(\log^2 n)$ und Tiefe $3 \log_{3/2} n + O(\log \log n) \leq 2 \log n + O(\log \log n)$ findet der Leser in [Weg89] Seite 127f. Es ist allerdings zu beachten, daß der dort vorgestellte Schaltkreis an die Phasenabbildung $p$ angepaßt werden muß, indem allen Variablen mit negativer Phase ein $not-$Gatter vorgeschaltet wird.
Verwendet man $BSUM_n$ in der Konstruktion von Lemma 2.21, so ergibt sich für den resultierenden Schaltkreis die Behauptung des Lemmas.

□

Dies bedeutet, daß jede totalsymmetrische Funktion durch eine Schaltung mit (asymptotisch) linearer Größe und logarithmischer Tiefe in der Anzahl der Eingangsvariablen realisiert werden kann. Asymptotisch gesehen ist diese Realisierung sogar optimal. Dies zeigt das folgende Lemma:

**Lemma 2.23** *Sei $f \in B_n$ eine totale Boolesche Funktion und $X = \{x_1, \ldots, x_n\}$ eine Variablenmenge. Ist $f$ abhängig von $X$*[†]*, dann gilt für jeden $B_2$-Schaltkreis $S$ mit $f_S = f$*

$$C_{B_2}(S) \geq n - 1 \text{ und}$$

$$T_{B_2}(S) \geq \lceil \log n \rceil.$$

**Beweis:** Sei $S = ((V, E), typ, pe, pa)$ ein optimaler $B_2-$Schaltkreis für $f$.
Da $f$ von allen Variablen aus $X$ abhängt, existiert für jede Variable einen Eingangsknoten $v$ mit $typ(v) = EPAD$ und $outdeg(v) \geq 1$.
Da $S$ optimal ist, gilt für alle Knoten $v$ mit $typ(v) \in B_2$: $outdeg(v) \geq 1$[‡].
Somit gilt:

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

Des weiteren gilt:

- $typ(v) \in B_2 \Rightarrow indeg(v) = 2.$
- $typ(v) = APAD \Rightarrow indeg(v) = 1.$
- $typ(v) = EPAD \Rightarrow indeg(v) = 0.$

Somit erhält man

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

und

$$\begin{aligned} (1), (2) \quad &\Rightarrow \quad 2C_{B_2}(f) + 1 \geq n + C_{B_2}(f) \\ &\Longleftrightarrow \quad C_{B_2}(f) \geq n - 1. \end{aligned}$$

Sei $S = ((V, E), typ, pe, pa)$ nun ein beliebiger $B_2-$Schaltkreis für $f$. Da $f$ von allen Variablen aus $X$ abhängt, existiert ein Ausgangsknoten $v$ mit $typ(v) = APAD$, so daß für alle Eingangsknoten $w$ mit $typ(w) = EPAD$ ein Pfad von $w$ nach $v$ existiert. Wäre dies nicht der Fall, so könnte man die Eingabe mindestens eines Eingangsknotens negieren *ohne* die Funktion dabei zu verändern. Man beweist die Aussage über die Tiefe nun mittels Induktion über $n \geq 1$:

Induktionsanfang: $n = 1$
Aufgrund der vorhergehenden Ausführungen, muß ein Pfad von einem Eingangsknoten zu einem Ausgangsknoten existieren, d.h. $T_{B_2}(S) \geq 0$.

---

[†]D.h. es gibt keine Variable aus $X$ von der $f$ unabhängig ist.

[‡]Sonst könnte man diese Gatter aus der Schaltung entfernen *ohne* die berechnete Funktion zu verändern.

Induktionsschritt: $n > 1$
Sei $Dep(v) := \{w \mid typ(w) = EPAD \wedge \exists$ Pfad in $S$ von $w$ nach $v\}$. Sei *out* ein Ausgangsknoten mit $\mid Dep(out) \mid = n$. Ein solcher Knoten muß existieren. Sei weiter $bottom \in V$ mit $typ(bottom) \in B_2$ und $O(bottom, 1) = out$, $I(bottom, 1) = left$ und $I(bottom, 2) = right$, d.h. *bottom* ist der direkte Vorgängerknoten von *out* in $S$.
Dann gilt:

- $\mid Dep(left) \mid + \mid Dep(right) \mid \geq n$

$\Rightarrow \qquad Min\{\mid Dep(left) \mid, \mid Dep(right) \mid\} \geq \lceil \frac{n}{2} \rceil$.

- Da $\lceil \frac{n}{2} \rceil < n$ für $n > 1$, gilt nach Induktionsvoraussetzung

$T_{B_2}(S) \geq \lceil \log \lceil \frac{n}{2} \rceil \rceil + 1 \geq \lceil \log \frac{n}{2} \rceil + 1 = \lceil \log n \rceil$.

□

Als letztes Beispiel soll die effiziente Realisierung von stark multiformsymmetrischen Funktionen betrachtet werden.

**Lemma 2.24** *Sei $f \in B_n$ eine totale Boolesche Funktion und $X = \{x_1, \ldots, x_n\}$ eine Variablenmenge. Ist $f$ stark multiformsymmetrisch in der Partition $\Pi = \{\lambda_1, \ldots, \lambda_k\}$, dann ist*

$$C_{B_2}(f) \leq \sum_{i=1}^{k} (\mid \lambda_i \mid -1) + 2 \cdot 2^k + O(\sqrt{2^k}).$$

**Beweis:** Die Funktion $f_{orbit_G}$ für $G = G_{\Pi}^{mul}$ wird durch eine Schaltung für $PAR_{\Pi} \in B_{n,k}$ realisiert. Dabei gilt $PAR_{\Pi}(\epsilon_1, \ldots \epsilon_n)_i := \mid \epsilon \mid_{\lambda_i}^{p} \; mod\, 2$. Da $p$ beliebig gewählt werden kann, entscheidet man sich für die Abbildung, die allen Variablen eine positive Phase zuordnet.
Die einzelnen Teilfunktionen$(PAR_{\Pi})_i$ können durch einen balancierten binären Baum aus $exor-$Gattern berechnet werden. Als Eingabe dienen alle Variablen aus $\lambda_i$. Ein solcher Baum besitzt höchstens $\mid \lambda_i \mid -1$ Grundgatter.
Der restliche Teil der Schaltung wird analog zu der in Lemma 2.21 konstruiert. Mit Hilfe der Aussage dieses Lemma folgt auch die Behauptung.

□

Die besonderen Eigenschaften $G-$symmetrischer Boolescher Funktionen können nicht nur beim Chipentwurf, sondern auch bei der Suche nach einer Darstellung für eine Funktion ausgenutzt werden. An dem Beispiel der ROBDDs soll der Einfluß der Symmetrie auf die Größe der Darstellung näher untersucht werden.

Aus dem vorigen Kapitel ist bekannt, daß die Variablenordnung einen entscheidenden Einfluß auf die Größe eines ROBDDs besitzen kann. Deshalb wird im folgenden gezeigt, wie durch die Einbeziehung von Symmetrien die Variablenordnung verbessert werden kann.
Ausgangspunkt der Untersuchungen bildet die Erkenntnis, daß die Vertauschung symmetrischer Variablen innerhalb einer Variablenordnung, die Größe des ROBDDs nicht verändert:

**Lemma 2.25** *Sei $X_\pi$ eine geordnete Variablenmenge, $f \in B_n$ eine totale Boolesche Funktion und $F$ ein* ROBDD *über $X_\pi$ mit $f_F = f$. Ist $f$ stark symmetrisch in $(x_i, x_j)$ und $<_{\pi'}$ die Ordnung auf $X$, welche aus $<_\pi$ durch Vertauschung der beiden Variablen $x_i, x_j$ hervorgeht, dann gilt für die Größe des* ROBDD*s $G$ über $X_{\pi'}$ mit $f_G = f$:*

$$| F |=| G | .$$

**Beweis:** O.B.d.A. $X = \{x_1, \ldots, x_n\}$, $i < j$, und $\pi[l] = x_l$, $1 \leq l \leq n$.
Dann gilt für $\pi'$:

$$\pi'(l) = \begin{cases} x_l & l \notin \{i,j\} \\ x_j & l = i \\ x_i & l = j \end{cases}$$

In Abhängigkeit der Symmetrie in $(x_i, x_j)$ unterscheidet man:
Fall 1: $f$ ist vertauschungssymmetrisch in $(x_i, x_j)$.
Der ROBDD $F$ wird wie folgt verändert: Allen Knoten $v$ mit $index(v) = x_i$ wird $index(v) = x_j$ zugeordnet bzw. $index(v) = x_j$ wird in $index(v) = x_i$ geändert. Man erhält somit einen ROBDD $G$ über $X_{\pi'}$ mit $| F |=| G |$.
Es bleibt daher zu zeigen, daß $f_G = f$ gilt.
Bezeichnet $\epsilon = (\epsilon_1, \ldots, \epsilon_i, \ldots, \epsilon_j, \ldots, \epsilon_n) \in \{0,1\}^n$ einen Pfad in $F$ und $v$ bzw. $w$ den entsprechenden $\epsilon$-Knoten in $F$ bzw. $G$, dann sind $v$ und $w$ Terminalknoten, und nach Lemma 1.8 ist $f_v = f(\epsilon_1, \ldots, \epsilon_i, \ldots, \epsilon_j \ldots \epsilon_n)$.
Da offensichtlich $value(v) = value(w)$ und $f$ stark vertauschungssymmetrisch in $(x_i, x_j)$ ist, gilt $f_w = f(\epsilon_1, \ldots, \epsilon_j, \ldots, \epsilon_i \ldots \epsilon_n)$.
Damit ist nach Lemma 1.8 auch $f_G = f$.
Fall 2: $f$ ist äquivalenzsymmetrisch in $(x_i, x_j)$.
Der ROBDD $F$ wird zunächst wie im ersten Fall zu $G'$ verändert. Anschließend werden die Abbildungen $high$ und $low$ wie folgt modifiziert:

$$high(v) = \begin{cases} high(v) & index(v) \notin \{x_i, x_j\} \\ low(v) & index(v) \in \{x_i, x_j\} \end{cases}$$

und

$$low(v) = \begin{cases} low(v) & index(v) \notin \{x_i, x_j\} \\ high(v) & index(v) \in \{x_i, x_j\} \end{cases}$$

Man erhält somit einen ROBDD $G$ über $X_{\pi'}$ mit $| F |=| G |$.
Es bleibt daher zu zeigen, daß $f_G = f$ gilt.
Bezeichnet $\epsilon = (\epsilon_1, \ldots, \epsilon_i, \ldots, \epsilon_j, \ldots, \epsilon_n) \in \{0,1\}^n$ einen Pfad in $F$ und $v$ bzw. $w$ den entsprechenden $\epsilon$-Knoten in $F$ bzw. $G$,
dann sind $v$ und $w$ Terminalknoten, und nach Lemma 1.8 ist $f_v = f(\epsilon_1, \ldots, \epsilon_i, \ldots, \epsilon_j \ldots \epsilon_n)$. Aufgrund der Konstruktion von $G$ gilt $value(w) = f(\epsilon_1, \ldots, \overline{\epsilon_i}, \ldots, \overline{\epsilon_j}, \ldots, \epsilon_n)$.
Da $f$ äquivalenzsymmetrisch in $(x_i, x_j)$ ist, folgt deshalb $f_w = f(\epsilon_1, \ldots, \epsilon_j, \ldots, \epsilon_i \ldots \epsilon_n)$.
Damit ist nach Lemma 1.8 $f_G = f$.

□

Lemma 1.8 besagt gerade, daß die Anzahl der Knoten eines ROBDDs über $X_\pi$ ($| X_\pi | = n$) durch die Zahl der Kofaktoren der Booleschen Funktion bzgl. der Mengen $\{\pi[1], \pi[2], \ldots, \pi[i]\}$ ($i \leq i \leq n$) bestimmt wird. Es scheint daher im Hinblick auf die resultierende Größe der Darstellung wichtig zu sein, die Variablenordnung so zu wählen, daß die entsprechenden Kofaktormengen möglichst klein sind. Für eine Boolesche Funktion, welche stark symmetrisch in $\lambda$ ist, gilt gerade $Kof_\lambda(f) = | \lambda | + 1$, wie man aus der Anzahl der entsprechenden Orbits folgern kann. Es scheint daher günstig zu sein, symmetrische Variablen in der Variablenordnung besonders nahe beieinander anzuordnen. Dies führt zu *symmetrischen Variablenordnungen* [MMD94]:

**Definition 2.11 (Symmetrische Variablenordnung)** *Sei $f \in BP_n$ eine Boolesche Funktion und $\Pi = \{\lambda_1, \ldots \lambda_k\}$ eine symmetrische Partition für $f$. Eine Variablenordnung $\pi$ heißt* **symmetrische Ordnung für** *$f$ genau dann, falls für jede Symmetriemenge $\lambda_i \in \Pi$ ein $j \in \mathbf{N}$ existiert, so daß*

$$\{\pi[j], \pi[j+1], \ldots, \pi[j + | \lambda_i | - 1]\} = \lambda_i.$$

Experimentelle Untersuchungen (z.B. [MMD94, PSP94]) haben gezeigt, daß für viele Boolesche Funktionen symmetrische Ordnungen sehr gute Ergebnisse im Hinblick auf die resultierende ROBDD-Größe liefern. Insbesondere sind für viele Boolesche Funktionen optimale Ordnungen auch gleichzeitig symmetrische Ordnungen.
Aufgrund von Lemma 2.25 spielt die Position der Variablen innerhalb einer Symmetriemenge für die Größe des resultierenden ROBDDs keine Rolle. Daher kommen den relativen Positionen der einzelnen Symmetriemengen in der Variablenordnung bei der Suche nach einer guten symmetrischen Variablenordnung eine entscheidende Bedeutung zu. Die originäre $sift-$Heuristik, die auf Variablenvertauschungen basiert, ist aus den bekannten Gründen (vgl. Abschnitt 1.5.3.3) nicht in der Lage, dieses Problem zu lösen. Deshalb muß für die Ermittlung einer guten symmetrischen Variablenordnung alle Variablen innerhalb einer Symmetriemenge zu einem Block zusammengefaßt werden und anschließend eine Permutation der Blöcke gesucht werden.
Setzt man für die Permutation der Blöcke die $sift-$Heuristik ein, so wird das entsprechende Verfahren auch als *symmetrisches sifting* ($sym\_sift$) bezeichnet. Untersuchungen in [MMD94, PSP94] zeigen, daß dieses Verfahren eine gute Heuristik ist, um die Variablenordnung zu verbessern.

Mit der Verwendung symmetrischer Variablenordnungen ist es möglich, die Anzahl der Knoten im ROBDD schärfer abzuschätzen als in Lemma 1.11. Für den Fall, daß die $k$ kleinsten Variablen in der Ordnung ein symmetrische Menge bilden, zeigt dies folgendes Lemma aus [Lit97]:

**Lemma 2.26** *Sei $F$ ein ROBDD über der geordneten Variablenmenge $X_\pi = \{x_1, \ldots, x_n\}$ mit $\pi[i] = x_i$, $1 \leq i \leq n$. Besitzt $F$ nur eine Wurzel, und ist $f_F$*

*symmetrisch in* $\lambda = \{x_1, \ldots x_k\}$*, dann gilt:*

| | | |
|---|---|---|
| (1) | $Ind_F(l) \leq min\{l, 2^{(k-l+2)2^{n-k}} - 2^{2^{n-k}}\}$ | *für* $1 \leq l \leq k$ |
| (2) | $Ind_F(l) \leq min\{l, 2^{2^{n-l+1}} - 2^{2^{n-l}}\}$ | *für* $l = k+1$ |
| (3) | $Ind_F(l) \leq min\{2^{l-k-1} \cdot Ind_F(k+1), 2^{2^{n-l+1}} - 2^{2^{n-l}}\}$ | *für* $k+2 \leq l \leq n$ |

**Beweis:** Sei $1 \leq l \leq k$. Da $f$ symmetrisch in $\lambda$ ist, gilt $\mid Kof_{\{x_1,\ldots,x_{l-1}\}}(f_F) \mid \leq l$. Daher gibt es höchstens $l$ Knoten, die mit $x_l$ beschriftet sind, d.h. $Ind_F(l) \leq l$. Jeder der Kofaktoren ist eine Boolesche Funktion über der Variablenmenge $\{x_l, \ldots, x_k, x_{k+1}, \ldots, x_n\}$. Da $f_F$ symmetrisch in $\{x_l, \ldots, x_k\}$ ist, gibt es offensichtlich $2^{(k-l+2)2^{n-k}}$ verschiedene Kofaktoren. Darunter befinden sich allerdings auch Funktionen, die nicht von $x_l$ abhängen, d.h. für die entsprechenden Funktionen gibt es keinen Knoten in $F$, der mit $x_l$ beschriftet ist. Da man nur starke Symmetrien betrachtet, sind diese Funktionen auch unabhängig von $\{x_l, \ldots, x_k\}$. Offensichtlich gibt es genau $2^{2^{n-k}}$ solcher Kofaktoren. Damit ist (1) gezeigt.
Sei $k+2 \leq l \leq n$. Die Bedingung $Ind_F(l) \leq 2^{l-k-1} \cdot Ind_F(k+1)$ folgt aus der Tatsache, daß jedes Nichtterminal höchstens zwei Nachfolger mit einem größeren Index besitzt, so daß $Ind_F(i+1) \leq 2 \cdot Ind_F(i)$ für jedes $i$ gilt. Da es nach Lemma 1.10 höchstens $2^{2^{n-l+1}} - 2^{2^{n-l}}$ verschiedene Funktionen über $\{x_l, \ldots, x_n\}$ gibt, die von $x_l$ abhängen, gilt auch die zweite Ungleichung in (3). Die Aussage (2) ergibt sich durch die bereits bewiesenen Fälle.

□

Ein ähnliches Lemma läßt sich auch für multiformsymmetrische Mengen entwickeln:

**Lemma 2.27** *Sei* $F$ *ein* ROBDD *über der geordneten Variablenmenge* $X_\pi = \{x_1, \ldots, x_n\}$ *mit* $\pi[i] = x_i$, $1 \leq i \leq n$. *Besitzt* $F$ *nur eine Wurzel, und ist* $f_F$ *multiformsymmetrisch in* $\lambda = \{x_1, \ldots x_k\}$*, dann gilt:*

| | | |
|---|---|---|
| (1) | $Ind_F(l) \leq 2$ | *für* $1 \leq l \leq k+1$. |
| (2) | $Ind_F(l) \leq min\{2^{l-k-1} \cdot Ind_F(k+1), 2^{2^{n-l+1}} - 2^{2^{n-l}}\}$ | *für* $k+2 \leq l \leq n$. |

**Beweis:** Aussage (2) ist schon in Lemma 2.26 gezeigt worden.
Die andere Aussage kann analog zu (1) und (2) in Lemma 2.26 bewiesen werden. Da $f_F$ allerdings multiformsymmetrisch in $\lambda$ ist, gilt $\mid Kof_{\{x_1,\ldots,x_{l-1}\}}(f_F) \mid \leq 2$.

□

In [Lit97] wird eine weitergehende Untersuchung symmetrischer Variablenordnungen und deren Einfluß auf die Größe von ROBDDs vorgestellt.

Für den Spezialfall der totalsymmetrischen Funktionen bzw. für Funktionen, die multiformsymmetrisch in der gesamten Variablenmenge sind, ergibt sich mit Hilfe der bisherigen Ergebnisse:

**Lemma 2.28** *Sei $F$ ein ROBDD über $X_\pi = \{x_1, \dots, x_n\}$ und $<_\pi$ eine beliebige Variablenordnung.*
*Besitzt $F$ nur eine Wurzel, dann gilt:*
*(1) Ist $f_F$ totalsymmetrisch, dann gilt $\mid F \mid \leq \frac{n\cdot(n+1)}{2} + 2$.*
*(2) Ist $f_F$ multiformsymmetrisch in $X$, dann gilt $\mid F \mid \leq 2n + 1$.*

**Beweis:** Zu (1)
Da nach Lemma 2.26 $Ind_F(i) \leq i$ für $1 \leq i \leq n$, gilt:

$$\sum_{i=1}^{n} Ind_F(i) = \frac{n \cdot (n+1)}{2}.$$

Für die Größe von $F$ sind noch die beiden Terminalknoten zu addieren.
Zu (2)
Da nach Lemma 2.27 $Ind_F(i) \leq 2$ für $1 \leq i \leq n$ und $F$ nur eine Wurzel besitzt, gilt:

$$\sum_{i=1}^{n} Ind_F(i) = 1 + 2(n-1) = 2n - 1.$$

Auch hier müssen noch die beiden Terminale addiert werden.

□

## 2.4 Bestimmung von Symmetrien

Bisher ging man immer davon aus, daß die Symmetriemengen einer Booleschen Funktion bekannt sind. Dies ist für Funktionen mit wenigen Eingangsvariablen auch relativ leicht festzustellen, evtl. sogar durch Aufstellung der Funktionstabelle. Die meisten für die Praxis interessanten Funktionen besitzen jedoch viele ($n > 32$) Eingangsvariablen, so daß die Funktionstabelle riesig und damit unpraktisch wird.
Man muß demnach andere Darstellungsformen verwenden, um effizient Symmetrien bestimmen zu können. Es hat sich gezeigt, daß die Verwendung von ROBDDs für die Bestimmung von Symmetrien in Variablenpaaren sehr gut geeignet ist.

### 2.4.1 Starke Symmetrien

Aus Korollar 2.3 geht hervor, daß eine totale Boolesche Funktion genau dann vertauschungssymmetrisch in den Variablen $(x_i, x_j)$ ist, falls die beiden Kofaktoren $f_{x_i\overline{x_j}}$ und $f_{\overline{x_i}x_j}$ die gleiche Funktion darstellen. Wird $f$ durch einen ROBDD $F$ repräsentiert, so können die beiden Kofaktoren in Zeit $O(\mid F \mid \cdot \log \mid F \mid)$ konstruiert werden und der Vergleich in linearer[†] Zeit erfolgen [Bry86].

[†]Durch Verwendung einer *unique-table* ([BRB90]) kann der Äquivalenztest in konstanter Zeit berechnet werden.

Ebenso kann eine paarweise Äquivalenzsymmetrie bestimmt werden, es müssen nur die entsprechenden Kofaktoren $f_{x_i x_j}$ und $f_{\overline{x_i}\,\overline{x_j}}$ berechnet werden.
Um alle Symmetriemengen einer totalen Funktion zu bestimmen, können alle $\binom{n}{2}$ Variablenpaare überprüft werden. Da die (starke) Symmetrierelation transitiv ist, muß nicht jedes dieser Paare getestet werden. Ist z.B. die Funktion totalsymmetrisch, so genügen die $n-1$ Symmetrietests bzgl. der Paare $(x_i, x_{i+1})$, $1 \leq i \leq n$.
Für Funktionen mit einer sehr großen Variablenmenge kann das Verfahren des Kofaktorvergleichs schnell ineffizient werden. In [MMW93] wird daher zunächst sehr schnell eine große Anzahl von asymmetrischen Paaren für eine Funktion bestimmt. Die restlichen Paare können danach mit dem bekannten Verfahren mittels Kofaktorvergleich untersucht werden.
Für partielle Boolesche Funktionen, die durch die totalen Funktionen $f_{on}, f_{off}, f_{dc}$ gegeben sind, können die Symmetrien der einzelnen totalen Funktionen gemeinsam betrachtet werden:

**Lemma 2.29** *Eine partielle Boolesche Funktion $f \in BP_n$ ist stark symmetrisch in einem Variablenpaar $(x_i, x_j)$ genau dann, falls zwei der (totalen) Funktionen $f_{on}$, $f_{off}$, $f_{dc}$ symmetrisch in $(x_i, x_j)$ sind.*

**Beweis:** Folgt unmittelbar aus der Eigenschaft, daß jede partielle Boolesche Funktion eindeutig durch zwei der Funktionen $f_{on}$, $f_{off}$ und $f_{dc}$ bestimmt wird.

□

**Lemma 2.30** *Sei $f \in BP_n$ eine partielle Boolesche Funktion. Ist eine der (totalen) Funktionen $f_{on}, f_{off}, f_{dc}$ asymmetrisch in $(x_i, x_j)$, dann ist $f$ nicht stark symmetrisch in $(x_i, x_j)$.*

**Beweis:** Ist eine der Funktionen $f_{on}, f_{off}, f_{dc}$ asymmetrisch in $(x_i, x_j)$, dann existieren Funktionsstellen $\epsilon, \epsilon' \in \{0,1\}^n$ mit $f(\epsilon) \neq f(\sigma_{ij}(\epsilon))$ bzw. $f(\epsilon') \neq f(\mu_{ij}(\epsilon'))$. Dann ist aber auch $f_{x_i\overline{x_j}} \neq f_{\overline{x_i}x_j}$ bzw. $f_{x_i x_j} \neq f_{\overline{x_i}\,\overline{x_j}}$ und $f$ damit nicht stark symmetrisch in $(x_i, x_j)$.

□

Damit kann das Verfahren aus [MMW93] ebenfalls benutzt werden, um starke Symmetrien *partieller* Funktionen aufzufinden. Man kann z.B. die beiden Funktionen $f_{on}$ und $f_{off}$ für jedes Variablenpaar gleichzeitig betrachten. Hat man bereits eine Asymmetrie von $f_{on}$ in $(x_i, x_j)$ festgestellt, so braucht dieses Paar für $f_{off}$ nicht weiter getestet zu werden, da die Funktion $f$ nach Lemma 2.30 dort nicht mehr stark symmetrisch sein kann.

### 2.4.2 Schwache Symmetrien

Für die schwache Vertauschungssymmetrie bzw. Äquivalenzsymmetrie ergibt sich aus Lemma 2.2 unmittelbar folgende Beziehung:

**Korollar 2.14** *Sei $f \in BP_n$ eine partielle Boolesche Funktion und $X$ eine Variablenmenge Sind $x_i, x_j$ zwei verschiedene Variablen aus $X$.*
*$f$ ist vertauschungssymmetrisch in $(x_i, x_j)$ genau dann, falls*

$$\begin{aligned}(f_{on})_{x_i\overline{x_j}} \wedge (f_{off})_{\overline{x_i}x_j} &= \mathbf{0} \quad \textit{und}\\ (f_{off})_{x_i\overline{x_j}} \wedge (f_{on})_{\overline{x_i}x_j} &= \mathbf{0}.\end{aligned}$$

*$f$ ist äquivalenzsymmetrisch in $(x_i, x_j)$ genau dann, falls*

$$\begin{aligned}(f_{on})_{\overline{x_i}\,\overline{x_j}} \wedge (f_{off})_{x_ix_j} &= \mathbf{0} \quad \textit{und}\\ (f_{off})_{\overline{x_i}\,\overline{x_j}} \wedge (f_{on})_{x_ix_j} &= \mathbf{0}.\end{aligned}$$

Da es sich bei der schwachen Symmetrie nicht um eine Äquivalenzrelation handelt, muß für alle $\binom{n}{2}$ Variablenpaare dieser Test durchgeführt werden.
Hat man bereits für eine Funktion starke Symmetrien bestimmt, dann läßt sich die Zahl der Tests reduzieren:

**Lemma 2.31** *Sei $f \in BP_n$ eine partielle Boolesche Funktion, $X$ eine Variablenmenge und $x_i, x_j, x_k \in X$. Ist $f$ stark symmetrisch in $(x_i, x_j)$ und schwach symmetrisch in $(x_j, x_k)$, dann ist $f$ auch schwach symmetrisch in $(x_i, x_k)$.*

**Beweis:** Wählt man die Partition $\Pi = \{\{x_i, x_j\}\} \cup \bigcup_{x \in X \setminus \{x_i, x_j\}} \{\{x\}\}$, so ist $f$ stark symmetrisch in $\Pi$.
Nach Satz 2.3 ist $f$ auch symmetrisch in $\{x_i, x_j, x_k\}$ und damit auch symmetrisch in $(x_i, x_k)$.

□

Zusammenfassend lassen sich schwache Symmetrien zwischen allen Variablen wie folgt berechnen:

- Überprüfung aller $\binom{n}{2}$ Variablenpaare auf starke Symmetrie. Dies kann z.B. mit dem Verfahren aus [MMW93] unter Zuhilfenahme der beiden Funktionen $f_{on}$ und $f_{off}$ erfolgen.
- Für die Paare, bei denen keine starke Symmetrie festgestellt wurde, wird mittels Korollar 2.14 und unter Ausnutzung von Lemma 2.31 getestet, ob eine schwache Symmetrie vorliegt.

Modifiziert man den Algorithmus aus [MMW93], indem man die Ergebnisse für $f_{off}$ und $f_{on}$ zwischenspeichert, so brauchen nicht alle Paare, die dort als asymmetrisch für $f$ erkannt wurden, auf eine schwache Symmetrie getestet zu werden:

**Lemma 2.32** *Sei $f \in BP_n$ eine partielle Funktion. Ist $f_{on}$ oder $f_{off}$ symmetrisch in $(x_i, x_j)$, so ist $f$ schwach symmetrisch in $(x_i, x_j)$.*

**Beweis:** Ist $f_{on}$ vertauschungssymmetrisch in $(x_i, x_j)$, dann gilt für alle $\epsilon \in \{0,1\}^n$ $f(\epsilon) = 1 \Rightarrow f(\sigma_{ij}(\epsilon)) = 1$. Das bedeutet

$$\begin{aligned} (f_{on})_{x_i,\overline{x_j}} \wedge (f_{off})_{\overline{x_i}x_j} &= \mathbf{0} \quad \text{und} \\ (f_{off})_{x_i,\overline{x_j}} \wedge (f_{on})_{\overline{x_i}x_j} &= \mathbf{0}, \end{aligned}$$

d.h. $f$ ist vertauschungssymmetrisch in $(x_i, x_j)$.

Eine entsprechende Argumentation gilt, falls $f_{off}$ vertauschungssymmetrisch in $(x_i, x_j)$ ist .
Ist $f$ äquivalenzsymmetrisch in $(x_i, x_j)$, so sind in den obigen Gleichungen die entsprechenden Kofaktoren einzusetzen.

□

**Bemerkung 10** *Die Umkehrung von Lemma 2.32 gilt i.a. nicht.*

In [KD91] wird ein Verfahren angegeben, mit dessen Hilfe sich auch Symmetrien in größeren Variablenmengen sehr einfach bestimmen lassen. Dazu muß die betrachtete partielle Boolesche Funktion durch *on* und *off* Arrays[†] repräsentiert werden. Die Kombination der berechneten Symmetriemengen zu symmetrischen Partitionen wird in [KD91] allerdings *nicht* untersucht.

### 2.4.3 Bestimmung von unabhängigen Variablen

Abschließend soll noch kurz auf die Bestimmung von unabhängigen Variablen mittels ROBDDs eingegangen werden.
Wie bereits erwähnt wurde, ist eine totale Funktion unabhängig von einer Variablen genau dann, falls es keinen Knoten im ROBDD gibt, der mit dieser Variablen markiert ist.
Dazu kann man mit Hilfe einer Durchmusterung des Graphen alle in einem ROBDD $F$ vorkommenden Variablen bestimmen ($Abh(F)$) und diese Menge von der ursprünglichen Variablenmenge subtrahieren. Bei schwachen Unabhängigkeiten kann dieses Verfahren allerdings nicht eingesetzt werden, da einige Knoten durchaus mit einer unabhängigen Variablen beschriftet sein können.
Aus der Definition der Unabhängigkeit folgert man einfach:

**Korollar 2.15** *Sei $f \in BP_n$ eine partielle Boolesche Funktion, $X$ eine Variablenmenge und $x_i \in X$. Dann ist $f$ unabhängig von $x_i$ genau dann, falls*

$$\begin{aligned} (f_{on})_{x_i} \wedge (f_{off})_{\overline{x_i}} &= \mathbf{0} \quad \textit{und} \\ (f_{off})_{x_i} \wedge (f_{on})_{\overline{x_i}} &= \mathbf{0}. \end{aligned}$$

Damit hat man ein Verfahren gewonnen, um alle unabhängigen Variablen zu ermitteln.
Untersucht man zuerst die Abhängigkeitsmengen der beiden Funktionen $f_{on}$ und $f_{off}$, so können diese Ergebnisse in die Berechnung von schwachen Unabhängigkeiten miteinbezogen werden:

[†]Bei Arrays handelt es sich um eine spezielle Darstellung Boolescher Polynome.

**Lemma 2.33** *Sei $f \in BP_n$ eine partielle Boolesche Funktion. Ist $f_{on}$ oder $f_{off}$ unabhängig von $x_i$, dann ist $f$ schwach unabhängig von $x_i$.*

**Beweis:** Ist $f_{on}$ unabhängig von $x_i$, so gilt für alle $\epsilon \in \{0,1\}^n$ $f(\epsilon) = 1 \Rightarrow f(\nu_i(\epsilon)) = 1$. Das bedeutet

$$\begin{aligned} (f_{on})_{x_i} \wedge (f_{off})_{\overline{x_i}} &= \mathbf{0} \quad \text{und} \\ (f_{off})_{x_i} \wedge (f_{on})_{\overline{x_i}} &= \mathbf{0}, \end{aligned}$$

d.h. $f$ ist unabhängig von $x_i$.
Entsprechendes gilt, falls $f_{off}$ unabhängig von $x_i$ ist.

$\square$

# Kapitel 3

# Symmetrische Erweiterungen partieller Boolescher Funktionen

Dieses Kapitel ist der Berechnung von symmetrischen Erweiterungen zu speziellen Symmetriegruppen $G \subseteq \mathrm{P_n}$ gewidmet.
Während im vorangegangenen Kapitel bereits gezeigt wurde, wie sich symmetrische bzw. multiformsymmetrische Partitionen bestimmen lassen (s. Satz 2.3 bzw. Satz 2.6), soll in diesem Kapitel näher auf die Berechnung der entsprechenden symmetrischen Erweiterungen eingegangen werden.
Dazu werden zunächst spezielle symmetrische Erweiterungen (sog. *notwendige symmetrische Erweiterungen*) eingeführt und deren Eigenschaften untersucht.
Es wird gezeigt, wie sich symmetrische Erweiterungen für größere Mengen bzw. Partitionen durch Folgen von notwendigen symmetrischen Erweiterungen zu den Abbildungen $\sigma_{ij}$ und $\mu_{ij}$ bestimmen lassen.

## 3.1 Notwendige symmetrische Erweiterungen

Wie aus den bisherigen Beispielen hervorgeht, können symmetrische Erweiterungen vorhandene Symmetrien einer partiellen Funktion zerstören. So ist es durchaus möglich, daß für eine partielle Funktion $f$, die symmetrisch in den Paaren $(x_i, x_j)$ und $(x_k, x_l)$ ist, eine symmetrische Erweiterung bezüglich $(x_i, x_j)$ asymmetrisch in $(x_k, x_l)$ wird.
Dies könnte eventuell an der Wahl der symmetrischen Erweiterung liegen, die bis auf die Eigenschaft, daß sie stark symmetrisch in $(x_i, x_j)$ ist, nicht eindeutig bestimmt ist. Deshalb werden nun spezielle symmetrische Erweiterungen eingeführt, die sogenannten *notwendigen symmetrischen Erweiterungen*, die sich gerade dadurch auszeichnen, daß sie unter allen symmetrischen Erweiterungen für eine Gruppe $G \subseteq \mathrm{P_n}$ nur die minimale Anzahl der Funktionsstellen aus der don't care-Menge spezifizieren.

**Definition 3.1 (Notwendige $G-$symmetrische Erweiterung)**
*Sei $f \in BP_n$ eine partielle Boolesche Funktion und $G \subseteq \mathrm{P_n}$. Ist $f$ $G-$symmetrisch, dann heißt eine $G-$symmetrische Erweiterung $f_G \in Erw(f)$ mit*

$$|DC(f_G)| \geq max\{|DC(g)| \mid g \text{ ist eine } G-\text{symmetrische Erweiterung von } f\}$$

**notwendige $G-$symmetrische Erweiterung von** $f$*.*

**Korollar 3.1** *Eine partielle Boolesche Funktion $f \in BP_n$ ist genau dann stark $G-$symmetrisch, falls $f = f_G$.*

Das folgende Lemma zeigt, daß notwendige symmetrische Erweiterungen *eindeutig* bestimmt sind:

**Lemma 3.1** *Sei $f \in BP_n$ und $G \subseteq \mathrm{P_n}$. Dann ist $f_G$ eindeutig bestimmt, und es gilt für jede $G-$symmetrische Erweiterung $g$ von $f$ stets $g \in Erw(f_G)$.*

**Beweis:** O.B.d.A. $G$ ist eine Untergruppe von $\mathrm{P_n}$.
Sei $h \in BP_n$ wie folgt definiert:

$$h(\epsilon) = \begin{cases} 1 & \text{falls } 1 \in f(orbit_G(\epsilon)) \\ 0 & \text{falls } 0 \in f(orbit_G(\epsilon)) \\ * & \text{sonst} \end{cases}$$

Da $f$ $G-$symmetrisch ist und daher $\{0,1\} \not\subseteq f(orbit_G)$, ist $h$ wohldefiniert. Wegen $\epsilon \in orbit_G(\epsilon)$ ist $h$ auch eine Erweiterung von $f$ und nach Konstruktion auch stark $G-$symmetrisch.
Sei $g \in Erw(f)$ eine weitere $G-$symmetrische Erweiterung von $f$, dann gilt offensichtlich $g \in Erw(h)$, da $g$ stark $G-$symmetrisch ist. Damit ist auch $|DC(h)| \geq |DC(g)|$.
Somit ist $h = f_G$.

□

Damit ist auch die Bezeichnung für $f_G$ als *notwendige* symmetrische Erweiterung gerechtfertigt.

Aus dem Beweis zu Lemma 3.1 ergibt sich:

**Korollar 3.2** *Sei $f \in BP_n$ eine partielle Boolesche Funktion und $G \subseteq \mathrm{P_n}$ eine Gruppe. Gilt für $g \in BP_n$:*

$$g(\epsilon) = b \iff b \in f(orbit_G(\epsilon)) \quad mit\ b \in \{0,1\},$$

*dann ist $g = f_G$.*

Von besonderem Interesse sind die notwendigen Erweiterungen zu den Abbildungen $\sigma_{ij}, \mu_{ij}, \upsilon_{ij}$ und $\upsilon_i$:

**Lemma 3.2** *Sei $f \in BP_n$ eine partielle Boolesche Funktion.*
*Für $\rho \in \{\sigma_{ij}, \mu_{ij}, \upsilon_{ij}, \upsilon_i\}$ $(1 \leq i < j \leq n)$ sei*

$$f_\rho(\epsilon) := \begin{cases} f(\epsilon) & \textit{falls } f(\epsilon) \neq * \\ f(\rho(\epsilon)) & \textit{falls } f(\epsilon) = * \end{cases}$$

*Ist $G =< \{\rho\} >$ und $f$ $G-$symmetrisch, dann gilt $f_\rho = f_G$.*

**Beweis:** Sei $\rho \in \{\sigma_{ij}, \mu_{ij}, \upsilon_{ij}, \upsilon_i\}$ $(1 \leq i < j \leq n)$ beliebig.
Da $G :=< \{\rho\} >= \{\rho, id\}$, ist $orbit_G(\epsilon) = \{\epsilon, \rho(\epsilon)\}$ $\forall \epsilon \in \{0,1\}^n$ .
Da $f$ $G-$symmetrisch ist, d.h. $\{0,1\} \not\subseteq f(orbit_G(\epsilon))$, ergibt sich für $f_\rho$ auch:

$$f_\rho(\epsilon) = \begin{cases} 1 & \text{falls } 1 \in f(orbit_G(\epsilon)) \\ 0 & \text{falls } 0 \in f(orbit_G(\epsilon)) \\ * & \text{sonst} \end{cases}$$

Aus Korollar 3.2 folgt deshalb $f_\rho = f_G$.

□

Aufgrund der besonderen Eigenschaften der Abbildungen $\sigma_{ij}$ und $\mu_{ij}$ läßt sich die notwendige symmetrische Erweiterung $f_{\upsilon_{ij}}$ auch mittels $< \{\sigma_{ij}\} > -$symmetrischer und $< \{\mu_{ij}\} > -$symmetrischer Erweiterungen berechnen:

**Lemma 3.3** *Sei $f \in BP_n$ eine partielle Boolesche Funktion. Ist $f$ multiformsymmetrisch in $(x_i, x_j)$ und sind $g, h \in BP_n$ wie folgt definiert:*

$$\begin{aligned} g &:= f_{\sigma_{ij}} \\ h &:= g_{\mu_{ij}} \end{aligned}$$

*Dann ist $h = f_{\upsilon_{ij}}$.*

**Beweis:** Sei $G =< \{\upsilon_{ij}\} >$. Dann gilt für jeden Vektor $\epsilon \in \{0,1\}^n$ entweder $orbit_G(\epsilon) = \{\mu_{ij}(\epsilon), \epsilon\}$ oder $orbit_G(\epsilon) = \{\sigma_{ij}(\epsilon), \epsilon\}$.
Es ist leicht einzusehen, daß für jeden Booleschen Vektor $\epsilon \in \{0,1\}^n$ entweder $\sigma_{ij}(\epsilon) \neq \epsilon$ oder $\mu_{ij}(\epsilon) \neq \epsilon$ gilt.
Ist $b \in f(orbit_G(\epsilon))$ mit $b \in \{0,1\}$ , dann folgt aus Lemma 3.2 $g(\epsilon) = b$, falls $orbit_G(\epsilon) = \{\sigma_{ij}(\epsilon), \epsilon\}$ oder $h(\epsilon) = b$, falls $orbit_G(\epsilon) = \{\mu_{ij}(\epsilon), \epsilon\}$.
Da $h \in Erw(g)$, ist in jedem Fall $h(\epsilon) = b$.
Ist $f(orbit_G(\epsilon)) = *$, dann ist auch $g(orbit_G(\epsilon)) = h(orbit_G(\epsilon)) = *$.
Das bedeutet insgesamt $h(\epsilon) = b \iff b \in f(orbit_G(\epsilon))$ für $b \in \{0,1\}$. Damit gilt $h = f_G = f_{\upsilon_{ij}}$.

□

Wie aus dem Beweis zu diesem Lemma hervorgeht, ist die Reihenfolge der beiden Symmetrierungen unwichtig.

Es werden nun die Auswirkungen dieser speziellen Symmetrierungen auf alle anderen Symmetrien einer Booleschen Funktion näher untersucht. Die folgenden beiden Lemmata vergleichen die Symmetrien der notwendigen Erweiterungen $f_{\sigma_{ij}}, f_{\mu_{ij}}, f_{\upsilon_i}$ mit denen der Ausgangsfunktion $f$:

**Lemma 3.4** *Sei $f \in BP_n$ eine partielle Boolesche Funktion, $X$ eine Variablenmenge und $x_i, x_j, x_k, x_l, x_q \in X$ paarweise verschiedene Variablen. Ist $f$ stark symmetrisch in $(x_k, x_l)$und stark unabhängig von $x_q$, dann gilt:*
*(1) Ist $f$ vertauschungssymmetrisch in $(x_i, x_j)$, dann ist $f_{\sigma_{ij}}$ stark symmetrisch in $(x_k, x_l)$ und stark unabhängig von $x_q$.*
*(2) Ist $f$ äquivalenzsymmetrisch in $(x_i, x_j)$, dann ist $f_{\mu_{ij}}$ stark symmetrisch in $(x_k, x_l)$ und stark unabhängig von $x_q$.*
*(3) Ist $f$ unabhängig von $x_i$, dann ist $f_{\upsilon_i}$ stark symmetrisch in $(x_k, x_l)$ und stark unabhängig von $x_q$.*

**Beweis:** Zu (1)
In Abhängigkeit von der Symmetrie in $(x_k, x_l)$ unterscheidet man:
Fall 1: $f$ ist stark vertauschungssymmetrisch in $(x_k, x_l)$.
Sei $G =< \{\sigma_{kl}\} >= \{\sigma_{kl}, id\}$. Es ist zu zeigen, daß für alle $\epsilon \in \{0,1\}^n$ gilt $| f_{\sigma_{ij}}(orbit_G(\epsilon)) |= 1$.
Sei $\epsilon \in \{0,1\}^n$ beliebig.
Da $f$ stark vertauschungssymmetrisch in $(x_k, x_l)$ ist, gilt $| f(orbit_G(\epsilon)) |= 1$.
Ist $f(orbit_G(\epsilon)) \in \{0,1\}$, dann gelten diese Belegungen auch für $f_{\sigma_{ij}} \in Erw(f)$.
Sei daher $f(orbit_G(\epsilon)) = *$. Da $orbit_G(\epsilon) = \{\epsilon, \sigma_{kl}(\epsilon)\}$, ist
$f_{\sigma_{ij}}(orbit_G(\epsilon)) = \{f(\sigma_{ij}(\epsilon)), f(\sigma_{ij}(\sigma_{kl}(\epsilon)))\}$.
Da $\sigma_{ij}(\sigma_{kl}(\epsilon)) = \sigma_{kl}(\sigma_{ij}(\epsilon))$ und $f$ stark vertauschungssymmetrisch in $(x_k, x_l)$ ist, gilt $| f_{\sigma_{ij}}(orbit_G(\epsilon)) |= 1$.
Damit ist $f_{\sigma_{ij}}$ stark vertauschungssymmetrisch in $(x_k, x_l)$.

Fall 2: $f$ ist stark äquivalenzsymmetrisch in $(x_k, x_l)$.
Kann analog zu Fall 1 mittels der Gruppe $G =< \{\mu_{kl}\} >= \{\mu_{kl}, id\}$ gezeigt werden.

Sei $G =< \{\upsilon_q\} >= \{\upsilon_q, id\}$. Es ist zu zeigen, daß für alle $\epsilon \in \{0,1\}^n$ gilt $| f_{\sigma_{ij}}(orbit_G(\epsilon)) |= 1$.
Sei $\epsilon \in \{0,1\}^n$ beliebig.
Da $f$ stark unabhängig von $x_q$ ist, gilt $| f(orbit_G(\epsilon)) |= 1$.
Ist $f(orbit_G(\epsilon)) \in \{0,1\}$, dann gelten diese Belegungen auch für $f_{\sigma_{ij}} \in Erw(f)$.
Sei daher $f(orbit_G(\epsilon)) = *$. Da $orbit_G(\epsilon) = \{\epsilon, \upsilon_q(\epsilon)\}$, ist
$f_{\sigma_{ij}}(orbit_G(\epsilon)) = \{f(\sigma_{ij}(\epsilon)), f(\sigma_{ij}(\upsilon_q(\epsilon)))\}$.
Da $\sigma_{ij}(\upsilon_q(\epsilon)) = \upsilon_q(\sigma_{ij}(\epsilon))$ und $f$ stark unabhängig von $x_q$ ist, gilt $| f_{\sigma_{ij}}(orbit_G(\epsilon)) |= 1$.
Damit ist $f_{\sigma_{ij}}$ stark unabhängig von $x_q$.

Die Aussagen (2) und (3) können analog zu (1) mit Hilfe der Abbildungen $\mu_{ij}$ bzw. $\upsilon_i$ bewiesen werden.

□

Das folgende Lemma behandelt die restlichen starken Symmetrien von $f$:

**Lemma 3.5** *Sei $f \in BP_n$ eine partielle Boolesche Funktion, $X$ eine Variablenmenge und $x_i, x_j, x_k \in X$ paarweise verschiedene Variablen. Ist $f$ stark vertauschungssymmetrisch (bzw. äquivalenzsymmetrisch) in $(x_j, x_k)$, dann gilt:*
*(1) Ist $f$ vertauschungssymmetrisch in $(x_i, x_j)$, dann ist $f_{\sigma_{ij}}$ vertauschungssymmetrisch (bzw. äquivalenzsymmetrisch) in $(x_j, x_k)$.*
*(2) Ist $f$ äquivalenzsymmetrisch in $(x_i, x_j)$, dann ist $f_{\mu_{ij}}$ vertauschungssymmetrisch (bzw. äquivalenzsymmetrisch) in $(x_j, x_k)$.*

**Beweis:** Zu (1)
In Abhängigkeit von der Symmetrie in $(x_j, x_k)$ unterscheidet man:
<u>Fall 1:</u> $f$ ist stark vertauschungssymmetrisch in $(x_j, x_k)$.
Sei $G =< \{\sigma_{jk}\} >= \{\sigma_{jk}, id\}$. Es ist zu zeigen, daß für alle $\epsilon \in \{0,1\}^n$ gilt $\{0,1\} \not\subseteq f_{\sigma_{ij}}(orbit_G(\epsilon))$.

Annahme: Es existiert ein $\epsilon \in \{0,1\}^n$ mit $\{0,1\} \subseteq f_{\sigma_{ij}}(orbit_G(\epsilon))$.
Da $orbit_G(\epsilon) = \{\epsilon, \sigma_{jk}(\epsilon)\}$ und $\mid f(orbit_G(\epsilon)) \mid = 1$, muß $f(orbit_G(\epsilon)) = *$ und $\sigma_{jk}(\epsilon) \neq \epsilon$ gelten.
Daher ist $\epsilon_j \neq \epsilon_k$. Wegen $\epsilon_i = \sigma_{jk}(\epsilon)_i$ ist genau ein Vektor aus $orbit_G(\epsilon)$ invariant gegenüber der Abbildung $\sigma_{ij}$.
Aus Lemma 3.2 folgt $* \in f_{\sigma_{ij}}(orbit_G(\epsilon))$.
Da aber $\mid orbit_G(\epsilon) \mid = 2$, ist $\{0,1\} \subseteq f_{\sigma_{ij}}(orbit_G(\epsilon))$ unmöglich.
Dies widerspricht allerdings der Annahme.

<u>Fall 2:</u> $f$ ist stark äquivalenzsymmetrisch in $(x_j, x_k)$.
Kann analog zu Fall 1 mittels der Gruppe $G =< \{\mu_{jk}\} >= \{\mu_{jk}, id\}$ gezeigt werden.

Die Aussage (2) kann analog zu (1) mit Hilfe der Abbildung $\mu_{ij}$ bewiesen werden.

□

Faßt man die beiden Lemmata zusammen, so bedeutet dies, daß durch die Bildung der notwendigen symmetrischen Erweiterungen $f_{\sigma_{ij}}, f_{\mu_{ij}}, f_{v_i}$ alle starken Symmetrien für $f$ zumindest als entsprechende schwache Symmetrien erhalten bleiben.

### 3.1.1 Berechnung notwendiger Erweiterungen

Die Bestimmung der notwendigen symmetrischen Erweiterungen $f_{\sigma_{ij}}, f_{\mu_{ij}}$ und $f_{v_i}$ kann effizient durch die Berechnung von Kofaktoren ausgeführt werden. Die Laufzeit der entsprechenden Algorithmen *make_strongly_symm* und *make_strongly_indep* hängt wesentlich von der Darstellungsform ab, die für die beteiligten Funktionen gewählt wurde.

**Prozedur** *make_strongly_symm*
**Eingabe** Darstellungen $F$ und $X_i$, $X_j$, wobei $f := f_F{}^\dagger, f_{X_i}, f_{X_j} \in BP_n$.

**Ausgabe** Darstellung $F'$ (durch $F'_{on}$,$F'_{off}$) mit

$$f_{F'} = \begin{cases} f_{\sigma_{ij}} & \text{falls } f_{X_i} = x_i \text{ und } f_{X_j} = x_j \\ & \text{oder } f_{X_i} = \overline{x_i} \text{ und } f_{X_j} = \overline{x_j} \\ f_{\mu_{ij}} & \text{falls } f_{X_i} = x_i \text{ und } f_{X_j} = \overline{x_j} \\ & \text{oder } f_{X_i} = \overline{x_i} \text{ und } f_{X_j} = x_j \end{cases}$$

**Algorithmus**

1. if ($\mathit{is\_symm}(F, X_i, X_j)$) = 0 then Ausgabe „Es existiert keine gesuchte Erweiterung“
2. $$\begin{aligned} F'_{on} \;&=\; \overline{X_i}\,\overline{X_j} \wedge cof(F_{on}, \overline{X_i}, \overline{X_j}) \vee (X_i X_j) \wedge cof(F_{on}, X_i, X_j) \vee \\ &\quad (X_i\overline{X_j} \vee \overline{X_i}X_j) \wedge (cof(F_{on}, X_i, \overline{X_j}) \vee cof(F_{on}, \overline{X_i}, X_j)) \end{aligned}$$
3. $$\begin{aligned} F'_{off} \;&=\; \overline{X_i}\,\overline{X_j} \wedge cof(F_{off}, \overline{X_i}, \overline{X_j}) \vee (X_i X_j) \wedge cof(F_{off}, X_i, X_j) \vee \\ &\quad (X_i\overline{X_j} \vee \overline{X_i}X_j) \wedge (cof(F_{off}, X_i, \overline{X_j}) \vee cof(F_{off}, \overline{X_i}, X_j)) \end{aligned}$$

Dabei sind $F_{on}, F_{off}, G_{on}, G_{off}$ die Darstellungen für $f_{on}, f_{off}, g_{on}, g_{off}$.
Die Prozedur *cof* berechnet bei Eingabe von $F, X_i, X_j$ eine Darstellung von $(f_F)_{x_i^{\epsilon_i} x_j^{\epsilon_j}}$, falls $f_{X_i} = x_i^{\epsilon_i}$ und $f_{X_j} = x_j^{\epsilon_j}$.
Die Prozedur *is_symm* testet auf Symmetrie bzw. Äquivalenzsymmetrie in einem Variablenpaar.

Ebenso läßt sich $f_{\upsilon_i}$ berechnen:

**Prozedur** *make_strongly_indep*
**Eingabe** Darstellungen $F$ und $X_i$, wobei $f := f_F, f_{X_i} \in BP_n$.
**Ausgabe** Darstellung $F'$ (durch $F'_{on}$,$F'_{off}$) mit
$f_{F'} = f_{\upsilon_i}$, falls $f_{X_i} = x_i$ oder $f_{X_i} = \overline{x_i}$.
**Algorithmus**

1. if ($\mathit{is\_indep}(F, X_i)$) = 0 then Ausgabe „Es existiert keine gesuchte Erweiterung“
2. $F'_{on} = cof(F_{on}, X_i) \vee cof(F_{on}, \overline{X_i})$
3. $F'_{off} = cof(F_{off}, X_i) \vee cof(F_{off}, \overline{X_i})$

Die Prozeduren *make_strongly_symm* und *make_strongly_indep* belegen für eine partielle Boolesche Funktion $f \in BP_n$ also alle don't care-Werte $\epsilon \in \{0,1\}^n$

[†] Für eine Darstellung $F$ bezeichnet $f_F$ die von $F$ dargestellte Funktion.

mit $f(\rho(\epsilon)) \in \{0,1\}$, wobei $\rho \in \{\sigma_{ij}, \mu_{ij}, \upsilon_{ij}\}$ entsprechend. Die Korrektheit der beiden Prozeduren ergibt sich aus Lemma 3.2.
Eine Darstellung für die notwendige symmetrische Erweiterung $f_{\upsilon_{ij}}$ kann gemäß Lemma 3.3 durch zweimaliges Aufrufen von *make_strongly_symm* berechnet werden:

**Korollar 3.3** *Sei $f \in BP_n$ eine partielle Boolesche Funktion, $x_i, x_j$ zwei Variablen, $F$ eine Darstellung für $f$, $X_i$ eine Darstellung für $x_i$ und $X_j$ eine Darstellung für $x_j$. Ist $F'$ wie folgt definiert:*

$$F' := make\_strongly\_symm(\ make\_strongly\_symm(F, X_i, X_j), X_i, \overline{X_j})).$$

*Dann gilt $f_{F'} = f_{\upsilon_{ij}}$.*

## 3.2 Symmetrierungsfolgen

In diesem Abschnitt soll näher auf die Bestimmung von symmetrischen Erweiterungen bezüglich größerer Symmetriemengen bzw. symmetrischen Partitionen eingegangen werden.
Dabei werden die Ergebnisse aus dem vorigen Kapitel helfen, symmetrische Mengen und Partitionen zu bestimmen (siehe Sätze 2.3 und 2.6).
Es wird gezeigt, wie sich diese Erweiterungen durch Folgen von notwendigen symmetrischen Erweiterungen zu den Abbildungen $\sigma_{ij}$ und $\mu_{ij}$ berechnen lassen.

**Lemma 3.6** *Sei $f \in BP_n$, $\Pi$ eine Partition der Variablenmenge $X$ und $p$ eine Phasenabbildung auf $X$. Sei $[x_i] \in \Pi$, $x_j \in [x_i]$, und ist $f$ $G_{\Pi,p}^{sym}$−symmetrisch. Dann gilt:*
*(1) Ist $p(x_i) = p(x_j)$, dann ist $f_{\sigma_{ij}}$ $G_{\Pi,p}^{sym}$−symmetrisch.*
*(2) Ist $p(x_i) \neq p(x_j)$, dann ist $f_{\mu_{ij}}$ $G_{\Pi,p}^{sym}$−symmetrisch.*

**Beweis:** Zu (1)
Sei $g \in Erw(f)$ und ist $g$ stark $G_{\Pi,p}^{sym}$−symmetrisch, dann ist $g$ auch stark vertauschungssymmetrisch in $(x_i, x_j)$, somit existiert $f_{\sigma_{ij}}$.
Wegen Lemma 3.1 ist $g \in Erw(f_{\sigma_{ij}})$.
Damit ist auch $f_{\sigma_{ij}}$ $G_{\Pi,p}^{sym}$−symmetrisch.

Aussage (2) kann analog gezeigt werden.

□

Zur Bestimmung von symmetrischen Erweiterungen bezüglich einer symmetrischen Partition ist es möglich, *Folgen* von notwendigen symmetrischen Erweiterungen zu den Abbildungen $\sigma_{ij}$ und $\mu_{ij}$ anzugeben:

**Lemma 3.7** *Sei $f \in BP_n$, $\Pi$ eine Partition der Variablenmenge $X$ und $p$ eine Phasenabbildung auf $X$. $f$ ist genau dann $G_{\Pi,p}^{sym}$−symmetrisch, falls ein Folge* *von Symmetrierungen existiert mit*

$$\begin{aligned} f^{(0)} &:= f \\ f^{(1)} &= f^{(0)}_{\rho^1} \\ &\vdots \\ f^{(l)} &= f^{(l-1)}_{\rho^l} \end{aligned}$$

*und* $\rho^k \in G^{sym}_{\Pi,p} \cap \{\sigma_{ij}, \mu_{ij} \mid i,j \in \{1,\ldots,n\}\}$ *für* $1 \leq k \leq l$*, so daß* $f^{(l)}$ *stark* $G^{sym}_{\Pi,p}$*−symmetrisch ist.*

**Beweis:** „$\Longrightarrow$ “
Sei $E := G^{sym}_{\Pi,p} \cap \{\sigma_{ij}, \mu_{ij} \mid i,j \in \{1,\ldots,n\}\}$.
Wegen Lemma 3.6 sind alle Funktionen $f^{(k)} (0 \leq k \leq l)$ $G^{sym}_{\Pi,p}$−symmetrisch, d.h. die Folgeglieder existieren jeweils.
O.B.d.A. kann man annehmen, daß $f^{(k)} \neq f^{(k+1)}$ und damit $DC(f^{(k)}) \supset DC(f^{(k+1)})$ gilt.
Da $|DC(f^{(k)})|$ endlich ist, terminiert die Folge. Die Folge kann daher nicht mehr weitergeführt werden, falls für alle Abbildungen $\rho^k \in E$ gilt $f^{(k)} = f^{(k+1)}$. Dann ist aber $f^{(k)}$ stark $< E >= G^{sym}_{\Pi,p}$−symmetrisch.
„$\Longleftarrow$ “
Existiert diese Folge, so ist $f^{(l)} \in Erw(f)$ und damit ist $f$ $G^{sym}_{\Pi,p}$−symmetrisch.

□

Es ist zu beachten, daß dieses Lemma keine konkrete Symmetrierungsfolge angibt[†], es wird lediglich für jede symmetrische Partition eine solche Folge in ihrer Existenz gesichert.
Mit Hilfe der Ergebnisse aus dem vorigen Kapitel können symmetrische Partitionen und damit entsprechende Symmetrierungsfolgen bestimmt werden.
Bevor eine erste Folge angegeben wird, sind noch einige Vorbetrachtungen und Bezeichnungen notwendig:

**Bezeichnung 15** *Sei $X$ eine Variablenmenge und $p : X \to \{0,1\}$ eine Phasenabbildung auf $X$. Für $x_i \in X$ und $\lambda \subseteq X$ ($i \notin Ind(\lambda)$) ist*

$$E^{i,\lambda}_p := \bigcup_{j \in Ind(\lambda)} \{\sigma_{ij} \mid p(x_i) = p(x_j)\} \cup \{\mu_{ij} \mid p(x_i) \neq p(x_j)\}.$$

Man erhält unmittelbar:

**Korollar 3.4** *Es gilt für alle $\epsilon \in \{0,1\}^n$ und $\rho \in E^{i,\lambda}_p$:*

$$\rho(\epsilon) \neq \epsilon \quad \Longleftrightarrow \quad \begin{aligned} &\epsilon_i \neq p(x_i) \wedge \epsilon_j = p(x_j) \quad \textit{oder} \\ &\epsilon_i = p(x_i) \wedge \epsilon_j \neq p(x_j). \end{aligned}$$

[†]Insbesondere ist die Länge der Folge nicht bestimmt.

**Lemma 3.8** *Sei $X = \{x_1, \ldots, x_n\}$ eine Variablenmenge, $p$ eine Phasenabbildung auf $X$ und $\Pi = \{\lambda_1, \ldots, \lambda_k\}$ eine Partition von $X$ mit $\{x_i\} = \lambda_r \in \Pi$. Dann gilt für alle Gewichte $0 \leq w_l \leq \mid \lambda_l \mid \quad (1 \leq l \leq k)$ mit $w_s > 0$ $(\lambda_s \in \Pi)$:*
*(1) Ist $\epsilon \in C^{\Pi,p}_{w_1,\ldots,w_{r-1},0,w_{r+1},\ldots,w_s,\ldots,w_k}$, dann existiert eine Abbildung $\rho \in E_p^{i,\lambda_s}$, so daß $\rho(\epsilon) \in C^{\Pi,p}_{w_1,\ldots,w_{r-1},1,w_{r+1},\ldots,w_s-1,\ldots,w_k}$.*
*(2) Ist $\epsilon \in C^{\Pi,p}_{w_1,\ldots,w_{r-1},1,w_{r+1},\ldots,w_s-1,\ldots,w_k}$, dann existiert eine Abbildung $\rho \in E_p^{i,\lambda_s}$, so daß $\rho(\epsilon) \in C^{\Pi,p}_{w_1,\ldots,w_{r-1},0,w_{r+1},\ldots,w_s,\ldots,w_k}$.*

**Beweis:** Zu (1)
Sei $\epsilon \in C^{\Pi,p}_{w_1,\ldots,w_{r-1},0,w_{r+1},\ldots,w_s,\ldots,w_k}$ beliebig. Dann ist $\epsilon_i \neq p(x_i)$.
Annahme: Für alle Abbildungen $\rho \in E_p^{i,\lambda_s}$ gilt $\rho(\epsilon) = \epsilon$.
Dann folgt aus Korollar 3.4 für alle $j \in \lambda_s$ $\epsilon_j \neq p(x_j)$.
Damit ist $\mid \epsilon \mid^p_{\lambda_s} = w_s = 0$. Dies ist aber ein Widerspruch zu der Bedingung $w_s > 0$.

Zu (2)
Sei $\epsilon \in C^{\Pi,p}_{w_1,\ldots,w_{r-1},1,w_{r+1},\ldots,w_s-1,\ldots,w_k}$ beliebig. Dann ist $\epsilon_i = p(x_i)$ und $\mid \epsilon \mid^p_{\lambda_s} < \mid \lambda_s \mid$.
Annahme: Für alle Abbildungen $\rho \in E_p^{i,\lambda_s}$ gilt $\rho(\epsilon) = \epsilon$.
Dann folgt aus Korollar 3.4 für alle $j \in \lambda_s$ $\epsilon_j = p(x_j)$.
Damit ist $\mid \epsilon \mid^p_{\lambda_s} = \mid \lambda_s \mid$. Dies ist aber ein Widerspruch zur Bedingung $\mid \epsilon \mid^p_{\lambda_s} < \mid \lambda_s \mid$.

□

Satz 2.3 sagt aus, daß jede stark symmetrische Menge $\lambda$ um eine weitere Variable $x_i$ zu einer symmetrischen Menge $\lambda \cup x_i$ erweitert werden kann, vorausgesetzt es liegt eine Symmetrie zwischen $x_i$ und einer Variablen aus $\lambda$ vor. Der folgende Satz gibt nun eine Symmetrierungsfolge an, mit der sich eine symmetrische Erweiterung für $\lambda \cup x_i$ berechnen läßt:

**Satz 3.1** *Sei $f \in BP$ eine partielle Boolesche Funktion, $X$ eine Variablenmenge, $p$ eine Phasenabbildung auf $X$ und $\Pi$ eine Partition von $X$ mit $\{x_i\}, [x_j] \in \Pi$. Sei $f^{(0)} := f$ und*

$$\begin{aligned} f^{(1)} &= f^{(0)}_{\rho^1} \\ &\vdots \\ f^{(l)} &= f^{(l-1)}_{\rho^l}, \end{aligned}$$

*wobei $\bigcup_{1 \leq r \leq l} \{\rho^r\} = E_p^{i,[x_j]}$ und $\mid [x_j] \mid = l$.*
*Sei $\Pi' = \Pi \setminus \{[x_j], \{x_i\}\} \bigcup \{[x_j] \cup \{x_i\}\}$. Ist $f$ stark $G^{sym}_{\Pi,p}$−symmetrisch, dann gilt:*
*(1) Ist $f$ vertauschungssymmetrisch in $(x_i, x_j)$ und $p(x_i) = p(x_j)$, dann ist $f^{(l)}$ stark $G^{sym}_{\Pi',p}$−symmetrisch.*
*(2) Ist $f$ äquivalenzsymmetrisch in $(x_i, x_j)$ und $p(x_i) \neq p(x_j)$, dann ist $f^{(l)}$*

*stark $G^{sym}_{\Pi',p}$–symmetrisch.*

**Beweis:** Es wird zunächst für beide Aussagen gemeinsam gezeigt, daß die Symmetrierungsfolge überhaupt existiert.
Aus Satz 2.3 folgt[†], daß $f$ $G^{sym}_{\Pi',p}$–symmetrisch ist.
Da $E_p^{i,[x_j]} \subseteq G^{sym}_{\Pi',p}$, sichert Lemma 3.6 die Existenz der Folge.

Nun soll gezeigt werden, daß $f^{(l)}$ stark $G^{sym}_{\Pi',p}$–symmetrisch ist.
O.B.d.A. $\Pi = \{\lambda_1, \ldots, \lambda_k\}$ und $\lambda_1 = \{x_i\}, \lambda_2 = [x_j]$ mit $\lambda_1 \cap \lambda_2 = \emptyset$. Dann ist $\Pi' = \{\lambda_1 \cup \lambda_2, \lambda_3, \ldots, \lambda_k\}$.
Es ist zu zeigen, daß für alle Gewichtsklassen $|\, f(C^{\Pi',p}_{w_2,\ldots,w_k}) \,| = 1$ gilt. Man unterscheidet:
Fall 1: $w_2 \geq 1$.
Da $|\, \lambda_1 \,| = 1$, kann $C^{\Pi',p}_{w_2,\ldots,w_k}$ als disjunkte Vereinigung zweier Gewichtsklassen von $\Pi$ dargestellt werden:

$$C^{\Pi',p}_{w_2,\ldots,w_k} = \underbrace{C^{\Pi,p}_{0,w_2,\ldots,w_k}}_{C_0} \cup \underbrace{C^{\Pi,p}_{1,w_2-1,\ldots,w_k}}_{C_1}$$

Da $f$ stark $G^{sym}_{\Pi,p}$–symmetrisch ist, gilt $|\, f(C_1) \,| = |\, f(C_0) \,| = 1$. Daher müssen die folgenden Fälle betrachtet werden:
Fall 1.1: $f(C_0) \neq * \wedge f(C_1) \neq *$.
Da $f^{(l)} \in Erw(f)$, gelten diese Belegungen auch für $f^{(l)}$.
Fall 1.2: $f(C_0) = * \wedge f(C_1) = *$.
Da für alle Abbildungen $\rho \in E_p^{i,\lambda_2}$ gilt $\rho(C_0)$[‡] $\subseteq C_0 \cup C_1$ bzw. $\rho(C_1) \subseteq C_0 \cup C_1$, werden diese Belegungen auch durch keine notwendige symmetrische Erweiterung aus der Folge verändert.
Fall 1.3: $f(C_0) = * \wedge f(C_1) \neq *$.
Wegen Lemma 3.8 gibt es zu jedem $\epsilon \in C_0$ eine Abbildung $\rho \in E_p^{i,\lambda_2}$ mit $\rho(\epsilon) \in C_1$.
Da $\bigcup\limits_{1 \leq r \leq l} \{\rho^r\} = E_p^{i,\lambda_2}$, gilt $f^{(l)}(C_0) = f(C_1)$ und somit $|\, f^{(l)}(C_0 \cup C_1) \,| = 1$.
Fall 1.4: $f(C_0) \neq * \wedge f(C_1) = *$.
Kann analog zu Fall 1.3 gezeigt werden.

Somit ist in jedem Fall $|\, f^{(l)}(C^{\Pi',p}_{w_2,\ldots,w_k}) \,| = 1$.
Fall 2: $w_2 = 0$.
Dann ist $C^{\Pi',p}_{w_2,\ldots,w_k} = C^{\Pi,p}_{0,w_2,\ldots,w_k}$.
Da $f$ stark $G^{sym}_{\Pi,p}$–symmetrisch ist, gilt $|\, f(C^{\Pi',p}_{w_2,\ldots,w_k}) \,| = 1$.
Für alle $\epsilon \in C^{\Pi,p}_{0,0,w_3\ldots,w_k}$ ist $\epsilon_r \neq p(x_r)$ mit $x_r \in \lambda_1 \cup \lambda_2$. Aufgrund der Konstruktion von $E_p^{i,\lambda_2}$ gilt $\rho(\epsilon) = \epsilon$ für alle $\rho \in E_p^{i,\lambda_2}$ und daher $|\, f^{(l)}(C^{\Pi',p}_{w_2,\ldots,w_k}) \,| = 1$.
□

[†]Man beachte die im Beweis zu Satz 2.3 gestellte Bedingung an $p$.
[‡]Für $C \subseteq \{0,1\}^n$ und $\rho \in \mathrm{P_n}$ ist $\rho(C) = \{\rho(\epsilon) \mid \epsilon \in C\}$.

Es ist zu beachten, daß die in Satz 3.1 angegebene Folge nicht eindeutig ist, da die Abbildungen aus $E_p^{i,[x_j]}$ in beliebiger Reihenfolge zur Symmetrierung angewandt werden können. Lediglich die Länge der Folge ist durch $|[x_j]|$ begrenzt.

Aus dem Satz lassen sich einige wichtige Eigenschaften der Funktionen $f^{(r)}$ ableiten:

**Lemma 3.9** *Seien die Voraussetzungen wie in Satz 3.1 gegeben und* $[x_j] = \{x_{j_1}, \dots, x_{j_l}\}$*, dann gilt:*
*(1) Ist* $\rho^r = \sigma_{ij_r}$*, dann ist* $f^{(s)}$*,* $r \leq s \leq l$ *stark vertauschungssymmetrisch in* $(x_i, x_{j_r})$*.*
*(2) Ist* $\rho^r = \mu_{ij_r}$*, dann ist* $f^{(s)}$*,* $r \leq s \leq l$ *stark äquivalenzsymmetrisch in* $(x_i, x_{j_r})$*.*
*(3) Ist* $f$ *stark multiformsymmetrisch in* $\lambda \in \Pi$ *mit* $\lambda \neq [x_j]$*, dann ist auch* $f^{(l)}$ *stark multiformsymmetrisch in* $\lambda$*.*
*(4)* $f^{(l)} = f_{G^{sym}_{\Pi',p}}$*.*

**Beweis:** Zu (1)
Da $f^{(r)} = f^{(r-1)}_{\sigma_{ij_r}}$, ist $f^{(r)}$ stark vertauschungssymmetrisch in $(x_i, x_{j_r})$.
O.B.d.A. $\Pi = \{\lambda_1, \dots, \lambda_k\}$ und $\lambda_1 = \{x_i\}, \lambda_2 = [x_j]$ mit $\lambda_1 \cap \lambda_2 = \emptyset$. Dann ist $\Pi' = \{\lambda_1 \cup \lambda_2, \lambda_3, \dots, \lambda_k\}$.
Sei $\epsilon \in C^{\Pi',p}_{w_2,\dots,w_k}$ beliebig.
Ist $w_2 = 0$, dann gilt wegen $\sigma_{ij_r}(\epsilon) = \epsilon$ trivialerweise $f^{(s)}(\epsilon) = f^{(s)}(\sigma_{ij_r}(\epsilon))$ für alle $r \leq s \leq l$.
Ist $w_2 > 0$ und $C_0, C_1$ wie im Beweis zu Satz 3.1 definiert, dann gilt offensichtlich $\rho(C_0) \subseteq C_1 \cup C_0$ und $\rho(C_1) \subseteq C_1 \cup C_0$ für alle $\rho \in E_p^{i,[x_j]}$.
Annahme: $f^{(s)}$ $(s > r)$ ist nicht stark vertauschungssymmetrisch in $(x_i, x_{j_r})$.
Da $f$ stark $G^{sym}_{\Pi,p}$–symmetrisch ist und $\{0,1\} \not\subseteq f(C_0 \cup C_1)$, kann nur $f^{(s)}(\epsilon) = b$ und $f^{(s)}(\sigma_{ij_r}(\epsilon)) = *$ für ein $b \in \{0,1\}$ und ein $\epsilon \in C_0 \cup C_1$ gelten. Dann ist aber $f(C_0) = b$ oder $f(C_1) = b$ und somit $f^{(r)}(\sigma_{ij_r}(\epsilon)) = b$. Wegen $f^{(s)} \in Erw(f^{(r)})$ gilt auch $f^{(s)}(\sigma_{ij_r}(\epsilon)) = b$. Dies führt zu einem Widerspruch mit $f^{(s)}(\sigma_{ij_r}(\epsilon)) = *$.

Aussage (2) kann analog zu (1) mittels der Abbildung $\mu_{ij_r}$ gezeigt werden.

Aussage (3) ergibt sich unmittelbar aus Lemma 3.4.

Zu (4)
Wie aus dem Beweis zu Satz 3.1 hervorgeht, gilt für $f^{(l)}$ und $G := G^{sym}_{\Pi',p}$:

$$f^{(l)}(\epsilon) = b \iff b \in f(orbit_G(\epsilon)), \quad b \in \{0,1\}.$$

Das bedeutet aber nach Korollar 3.2 $f^{(l)} = f_G$.

□

Es soll nun versucht werden, gerade die Länge der Symmetrierungsfolge aus Satz 3.1 zu bestimmen, die unbedingt nötig ist, um eine stark $G^{sym}_{\Pi',p}$–symmetrische Erweiterung von $f$ zu erhalten. Dazu untersucht man zunächst für einen festen

Vektor aus $\{0,1\}^n$ , welche Abbildungen aus $E_p^{i,\lambda}$ diesen Vektor in einen anderen Vektor überführen:

**Bezeichnung 16** *Für* $\epsilon \in \{0,1\}^n$ *sei*

$$BRIDGE_p^{i,\lambda}(\epsilon) := \{\rho \in E_p^{i,\lambda} \mid \rho(\epsilon) \neq \epsilon\} \subseteq E_p^{i,\lambda}$$

*und*

$$Ind(BRIDGE_p^{i,\lambda}(\epsilon)) := \{j \in \lambda \mid \sigma_{ij} \text{ oder } \mu_{ij} \in BRIDGE_p^{i,\lambda}(\epsilon)\}.$$

Ähnlich zu Lemma 3.8 gilt nun:

**Lemma 3.10** *Sei* $X = \{x_1, \ldots, x_n\}$ *eine Variablenmenge, p eine Phasenabbildung auf X und* $\Pi = \{\lambda_1, \ldots, \lambda_k\}$ *eine Partition von X mit* $\{x_i\} = \lambda_r \in \Pi$. *Dann gilt für alle Gewichte* $0 \leq w_l \leq \mid \lambda_l \mid \quad (1 \leq l \leq k)$ *mit* $w_s > 0$ $(\lambda_s \in \Pi)$*:*
*(1) Ist* $\epsilon \in C^{\Pi,p}_{w_1,\ldots,w_{r-1},0,w_{r+1},\ldots,w_s,\ldots,w_k}$*, dann existiert für jedes Element* $\epsilon' \in C^{\Pi,p}_{w_1,\ldots,w_{r-1},1,w_{r+1},\ldots,w_s-1,\ldots,w_k}$ *eine Abbildung* $\rho \in BRIDGE_p^{i,\lambda_s}(\epsilon)$*, so daß* $\rho(\epsilon') \in C^{\Pi,p}_{w_1,\ldots,w_{r-1},0,w_{r+1},\ldots,w_s,\ldots,w_k}$.
*(2) Ist* $\epsilon \in C^{\Pi,p}_{w_1,\ldots,w_{r-1},1,w_{r+1},\ldots,w_s-1,\ldots,w_k}$*, dann existiert für jedes Element* $\epsilon' \in C^{\Pi,p}_{w_1,\ldots,w_{r-1},0,w_{r+1},\ldots,w_s,\ldots,w_k}$ *eine Abbildung* $\rho \in BRIDGE_p^{i,\lambda_s}(\epsilon)$*, so daß* $\rho(\epsilon') \in C^{\Pi,p}_{w_1,\ldots,w_{r-1},1,w_{r+1},\ldots,w_s-1,\ldots,w_k}$.

**Beweis:** Zu (1)
Seien $\epsilon \in C^{\Pi,p}_{w_1,\ldots,w_{r-1},0,w_{r+1},\ldots,w_s,\ldots,w_k}$, $\epsilon' \in C^{\Pi,p}_{w_1,\ldots,w_{r-1},1,w_{r+1},\ldots,w_s-1,\ldots,w_k}$ beliebig.
Sei $w = \mid BRIDGE_p^{i,\lambda_s}(\epsilon) \mid$ und $\lambda = \lambda_r \cup \lambda_s$. Dann ist $\mid \epsilon \mid_\lambda^p = \mid \epsilon' \mid_\lambda^p$ und $\epsilon_i \neq p(x_i)$.
Da für alle Abbildungen $\rho \in BRIDGE_p^{i,\lambda_s}(\epsilon)$ $\rho(\epsilon) \neq \epsilon$ gilt und für alle Abbildungen $\rho \in E_p^{i,\lambda_s} \backslash BRIDGE_p^{i,\lambda_s}(\epsilon)$ $\epsilon$ invariant gegenüber $\rho$ ist, folgt aus Korollar 3.4 somit $\epsilon_j = p(x_j)$, $j \in Ind(BRIDGE_p^{i,\lambda_s}(\epsilon))$ und $\epsilon_j \neq p(x_j)$, $j \in Ind(\lambda_s) \backslash Ind(BRIDGE_p^{i,\lambda_s}(\epsilon))$.
Damit ist $\mid \epsilon \mid_\lambda^p = w$.
Annahme: Für alle Abbildungen $\rho \in BRIDGE_p^{i,\lambda_s}(\epsilon)$ gilt $\rho(\epsilon') = \epsilon'$.
Dann folgt aus Korollar 3.4 und $\epsilon'_i = p(x_i)$ für alle Indizes $j \in Ind(BRIDGE_p^{i,\lambda_s}(\epsilon))$ $\epsilon'_j = p(x_j)$.
Damit ist wegen $\epsilon'_i = p(x_i)$ $\mid \epsilon' \mid_\lambda^p \geq w + 1$. Dies ist aber ein Widerspruch zu der Bedingung $\mid \epsilon \mid_\lambda^p = \mid \epsilon' \mid_\lambda^p$.

Zu (2)
Seien $\epsilon \in C^{\Pi,p}_{w_1,\ldots,w_{r-1},1,w_{r+1},\ldots,w_s-1,\ldots,w_k}$, $\epsilon' \in C^{\Pi,p}_{w_1,\ldots,w_{r-1},0,w_{r+1},\ldots,w_s,\ldots,w_k}$ beliebig.
Sei $w = \mid BRIDGE_p^{i,\lambda_s}(\epsilon) \mid$ und $\lambda = \lambda_r \cup \lambda_s$. Dann ist $\mid \epsilon \mid_\lambda^p = \mid \epsilon' \mid_\lambda^p$ und $\epsilon_i = p(x_i)$.
Da für alle Abbildungen $\rho \in BRIDGE_p^{i,\lambda_s}(\epsilon)$ $\rho(\epsilon) \neq \epsilon$ gilt und für alle Abbildungen $\rho \in E_p^{i,\lambda_s} \backslash BRIDGE_p^{i,\lambda_s}(\epsilon)$ $\epsilon$ invariant gegenüber $\rho$ ist, folgt aus Korollar 3.4 somit $\epsilon_j \neq p(x_j)$, $j \in Ind(BRIDGE_p^{i,\lambda_s}(\epsilon))$ und $\epsilon_j = p(x_j)$, $j \in Ind(\lambda_s) \backslash Ind(BRIDGE_p^{i,\lambda_s}(\epsilon))$.
Damit ist $\mid \epsilon \mid_\lambda^p = \mid \lambda \mid - w$.

Annahme: Für alle Abbildungen $\rho \in BRIDGE_p^{i,\lambda_s}(\epsilon)$ gilt $\rho(\epsilon') = \epsilon'$.
Dann folgt aus Korollar 3.4 und $\epsilon'_i \neq p(x_i)$ für alle Indizes $j \in Ind(BRIDGE_p^{i,\lambda_s}(\epsilon))$ $\epsilon'_j \neq p(x_j)$.
Damit ist wegen $\epsilon'_i \neq p(x_i)$ $|\epsilon'|_\lambda^p < |\lambda| - w$. Dies ist aber ein Widerspruch zu der Bedingung $|\epsilon|_\lambda^p = |\epsilon'|_\lambda^p$.

□

**Lemma 3.11** *Seien die Voraussetzungen wie in Satz 3.1 gegeben sowie* $\Pi = \{\lambda_1, \ldots, \lambda_k\}$, $\{x_i\} = \lambda_r$, $[x_j] = \{x_{j_1}, \ldots, x_{jl}\} = \lambda_s$ *und* $\lambda = \lambda_r \cup \lambda_s$.
*Dann gilt für alle Gewichtsklassen mit* $0 \leq w_m \leq |\lambda_m| \quad (1 \leq m \leq k)$, $w_s > 0$ *und*

$$\begin{aligned}
C_0 &:= C^{\Pi,p}_{w_1,\ldots,w_{r-1},0,w_{r+1},\ldots,w_s,\ldots,w_k}, \\
C_1 &:= C^{\Pi,p}_{w_1,\ldots,w_{r-1},1,w_{r+1},\ldots,w_s-1,\ldots,w_k}, \\
C &:= C_0 \cup C_1
\end{aligned}$$

$$\begin{array}{lll}
(1) & f(C_0) \neq * \wedge f(C_1) = * & \Rightarrow \quad |f^{(w_s)}(C)| = 1 \text{ und } * \in f^{(w_s-1)}(C_1). \\
(2) & f(C_0) = * \wedge f(C_1) \neq * & \Rightarrow \quad |f^{(|\lambda|-w_s)}(C)| = 1 \text{ und } * \in f^{(|\lambda|-w_s-1)}(C_0).
\end{array}$$

**Beweis:** O.B.d.A. gelte für die Symmetrierungsfolge in Satz 3.1 $\rho^q \in \{\sigma_{ij_q}, \mu_{ij_q}\}$, $1 \leq q \leq l$.
Zu (1)
Sei $\epsilon \in C = C_0 \cup C_1$ mit $\epsilon_i \neq p(x_i)$, $\epsilon_{j_q} = p(x_{j_q})$, $1 \leq q \leq w_s$ und $\epsilon_{j_q} \neq p(x_{j_q})$ für $w_s < q \leq l$.
Dann ist $\epsilon \in C_0$ und $BRIDGE_p^{i,\lambda_s}(\epsilon) = E_p^{i,\lambda_s} \cap \{\sigma_{ij_q}, \mu_{ij_q} \mid 1 \leq q \leq w_s\}$.
Nach Lemma 3.10 gibt es für jede Funktionsstelle $\epsilon'$ aus $C_1$ eine Abbildung $\rho \in BRIDGE_p^{i,\lambda_s}(\epsilon)$ mit $\rho(\epsilon') \in C_0$.
Damit ist $f^{(w_s)}(C_1) = f(C_0)$ und somit $|f^{(w_s)}(C)| = 1$.

Man betrachte $\epsilon \in C_1$ mit $\epsilon_i = p(x_i)$, $\epsilon_{j_q} = p(x_{j_q})$, $1 \leq q \leq w_s - 1$ und $\epsilon_{j_q} \neq p(x_{j_q})$ für $w_s \leq q \leq l$, dann gilt für alle Abbildungen $\rho \in E_p^{i,\lambda_s} \cap \{\sigma_{ij_q}, \mu_{ij_q} \mid 1 \leq q < w_s\}$ $\rho(\epsilon) = \epsilon$.
Da es sich bei den Symmetrierungen immer um notwendige symmetrische Erweiterungen handelt, folgt $f^{(w_s-1)}(\epsilon) = *$.

Zu (2)
Sei $\epsilon \in C = C_0 \cup C_1$ mit $\epsilon_i = p(x_i)$, $\epsilon_{j_q} \neq p(x_{j_q})$, $1 \leq q \leq |\lambda| - w_s$ und $\epsilon_{j_q} = p(x_{j_q})$ für $|\lambda| - w_s < q \leq l$.
Dann ist $\epsilon \in C_1$ und $BRIDGE_p^{i,\lambda_s}(\epsilon) = E_p^{i,\lambda_s} \cap \{\sigma_{ij_q}, \mu_{ij_q} \mid 1 \leq q \leq |\lambda| - w_s\}$.
Nach Lemma 3.10 gibt es für jede Funktionsstelle $\epsilon'$ aus $C_0$ eine Abbildung $\rho \in BRIDGE_p^{i,\lambda_s}(\epsilon)$ mit $\rho(\epsilon') \in C_1$.
Damit ist $f^{(|\lambda|-w_s)}(C_0) = f(C_1)$ und somit $|f^{(|\lambda|-w_s)}(C)| = 1$.

Man betrachte $\epsilon \in C_0$ mit $\epsilon_i \neq p(x_i)$, $\epsilon_{j_q} \neq p(x_{j_q})$, $1 \leq q \leq |\lambda| - w_s - 1$ und $\epsilon_{j_q} = p(x_{j_q})$ für $|\lambda| - w_s \leq q \leq l$, dann gilt für alle Abbildungen $\rho \in E_p^{i,\lambda_s} \cap \{\sigma_{ij_q}, \mu_{ij_q} \mid 1 \leq q < |\lambda| - w_s\}$ $\rho(\epsilon) = \epsilon$.

Da es sich bei den Symmetrierungen immer um notwendige symmetrische Erweiterungen handelt, folgt $f^{(|\lambda|-w_s-1)}(\epsilon) = *$.

$\square$

**Bezeichnung 17** *Unter den Bedingungen von Lemma 3.11 gilt:*
*Ist $C = C_0 \cup C_1$ eine Gewichtsklasse von $\Pi'$, dann heißt $C$* **0-einseitig für** $f$ *unter $\Pi$ genau dann, falls*

$$f(C_0) \neq * \wedge f(C_1) = *$$

*und* **1-einseitig für** $f$ *unter $\Pi$ genau dann, falls*

$$f(C_0) = * \wedge f(C_1) \neq *.$$

Mit Hilfe dieser Vorüberlegungen kann nun eine obere Schranke für die Symmetrierungsfolge aus Satz 3.1 angegeben werden:

**Satz 3.2** *Seien die Voraussetzungen wie in Satz 3.1 gegeben mit $\Pi' = \{\lambda_1, \ldots, \lambda_{k-1}\}$ und $[x_j] \cup \{x_i\} = \lambda_q \in \Pi'$. Dann gilt:*
*$f^{(r)}$ genau dann stark $G^{sym}_{\Pi',p}$–symmetrisch, falls*
*(1) keine 0-einseitige Gewichtsklasse $C^{\Pi',p}_{w_1,\ldots,w_{k-1}}$ für $f$ unter $\Pi$ existiert mit $w_q > r$ und*
*(2) keine 1-einseitige Gewichtsklasse $C^{\Pi',p}_{w_1,\ldots,w_{k-1}}$ für $f$ unter $\Pi$ existiert mit $|\lambda_q| - w_q > r$.*

**Beweis:** „$\Longrightarrow$"
Existiert eine Gewichtsklasse $C = C^{\Pi',p}_{w_1,\ldots,w_{k-1}}$, die 0-einseitig für $f$ ist mit $w_q > r$ oder die 1-einseitig für $f$ ist mit $|\lambda_q| - w_q > r$, dann ist nach Lemma 3.11 $|f^{(r)}(C)| \neq 1$ und damit $f^{(r)}$ nicht stark $G^{sym}_{\Pi',p}$–symmetrisch.
„$\Longleftarrow$"
Sei $C = C^{\Pi',p}_{w_1,\ldots,w_{k-1}}$ eine beliebige Gewichtsklasse von $\Pi'$.
Ist $w_q = 0$, dann werden die Belegungen für die Funktionsstellen aus $C$ niemals in der Symmetrierungsfolge aus Satz 3.1 verändert. Überdies ist $C$ auch eine Gewichtsklasse von $\Pi$ und $|f^{(r)}(C)| = 1$ für $1 \leq r \leq |[x_j]|$, da $|f(C)| = 1$.
Für $w_q > 0$ impliziert Lemma 3.11 $|f^{(r)}(C)| = 1$.
Damit ist $f^{(r)}$ stark $G^{sym}_{\Pi',p}$–symmetrisch.

$\square$

Dieses Lemma erlaubt die Bestimmung der notwendigen Länge für die Symmetrierungsfolge aus Satz 3.1. Mit den Bezeichnungen aus Satz 3.1 und Satz 3.2 ergibt sich unmittelbar folgendes Verfahren:

- Bestimme
  $P_1 = \{\epsilon \in C \mid C$ ist 1-einseitige Gewichtsklasse für $f$ unter $\Pi\}$ und
  $P_0 = \{\epsilon \in C \mid C$ ist 0-einseitige Gewichtsklasse für $f$ unter $\Pi\}$.

- Bestimme $c_0 = max\{|\epsilon|^p_{\lambda_q} \mid \epsilon \in P_0\}$ und $c_1 = min\{|\epsilon|^p_{\lambda_q} \mid \epsilon \in P_1\}$.
- Bestimme $f^{(max(c_0,|\lambda_q|-c_1))}$.

Da alle Funktionsstellen innerhalb einer Gewichtsklasse von $\Pi'$ das gleiche $\lambda$-Gewicht besitzen für alle $\lambda \in \Pi'$, ist es nicht unbedingt notwendig, $P_0$ bzw. $P_1$ vollständig zu bestimmen. Für die Berechnung von $c_0$ und $c_1$ reicht es daher aus, für jede in Frage kommende Gewichtsklasse, mindestens ein Element zu übernehmen.
Die folgende Prozedur *compute_length* berechnet die notwendige Länge für die Symmetrierungsfolge $f^{(r)}$. Dabei seien die Voraussetzungen und Bezeichnungen wie in Satz 3.1 bzw. Satz 3.2 gegeben.

**Prozedur** *compute_length*
**Eingabe** Darstellungen $F$ und $X_i, X_j$, wobei $f := f_F, f_{X_i}, f_{X_j} \in BP_n$ und $f_{X_i} = x_i^{\epsilon_i}$, $f_{X_j} = x_j^{\epsilon_j}$ mit $\epsilon_i, \epsilon_j \in \{0,1\}$.
**Ausgabe** $min(r) \in \mathbf{N}$, so daß $f^{(r)}$ stark $G^{sym}_{\Pi',p}$−symmetrisch. Dabei ist $p(x_i) = \epsilon_i$ und $p(x_j) = \epsilon_j$.
**Algorithmus**

1. $H_1 = (cof(F_{on}, X_i, \overline{X_j}) \vee cof(F_{off}, X_i, \overline{X_j})) \wedge cof(F_{dc}, \overline{X_i}, X_j)$
2. $H_0 = (cof(F_{on}, \overline{X_i}, X_j) \vee cof(F_{off}, \overline{X_i}, X_j)) \wedge cof(F_{dc}, X_i, \overline{X_j})$
3. $c_1 =$*compute_weight*$(H_1, \lambda_q \backslash \{x_i, x_j\}^\dagger, min) +1$
4. $c_0 =$*compute_weight*$(H_0, \lambda_q \backslash \{x_i, x_j\}, max) +1$
5. **return** $(max(c_0, |\lambda_q| - c_1))$

   Dabei berechnet *compute_weight* $(F', \lambda', min)$ das Minimum und *compute_weight* $(F', \lambda', max)$ das Maximum der Menge $\{|\epsilon|^p_{\lambda'} \mid f_{F'}(\epsilon) = 1\}$.
   Um die Korrektheit des Verfahrens zu gewährleisten, werden im Falle von $f_{F'} = \mathbf{0}$ die Werte $-1$ (bei $max$) bzw. $|\lambda_q| - 1$ (bei $min$) zurückgegeben.

Man kann sich leicht überlegen, daß für jede 1−einseitige Gewichtsklasse $C$ mindestens ein Vektor $\epsilon \in C$ existiert, so daß $f_{H_1}(\epsilon) = 1$. Entsprechendes gilt für alle 0−einseitigen Gewichtsklassen. Damit ergibt sich auch die Korrektheit der Prozedur.
Werden für die Darstellung der Booleschen Funktionen ROBDDs verwendet, so kann die Funktion *compute_weight* mittels Traversierung des Graphen (bottom-up) in linearer Zeit (bezogen auf die Größe des ROBDDs) ausgeführt werden.

Es gibt noch eine weitere Möglichkeit, die Symmetrierungsfolge aus Satz 3.1 evtl. zu verkürzen, *ohne* daß man die exakte Länge der benötigten Folge vorher berechnet:

**Lemma 3.12** *Seien die Voraussetzungen wie in Satz 3.1 gegeben, dann gilt: $f^{(r-1)}$ ist genau dann stark $G^{sym}_{\Pi',p}$−symmetrisch, falls $f^{(r)} = f^{(r-1)}$.*

† Die beiden Funktionen $f_{H_0}$ und $f_{H_1}$ sind (stark) unabhängig von $\{x_i, x_j\}$.

**Beweis:** O.B.d.A $\Pi = \{\lambda_1, \ldots, \lambda_k\}$ und $\lambda_1 = \{x_i\}, \lambda_2 = [x_j] = \{x_{j_1}, \ldots, x_{j_l}\}$ mit $\lambda_1 \cap \lambda_2 = \emptyset$ und $\lambda = \lambda_1 \cup \lambda_2$. Dann ist $\Pi' = \{\lambda, \lambda_3, \ldots, \lambda_k\}$.
Für die Symmetrierungsfolge in Satz 3.1 gelte $\rho^q \in \{\sigma_{ij_q}, \mu_{ij_q}\}$, $1 \leq q \leq l$.
„$\Longrightarrow$ “
Annahme: $f^{(r)} \neq f^{(r-1)}$.
Sei $\rho^r = \sigma_{ij_r} \in E_p^{i,[x_j]} \subseteq G_{\Pi',p}^{sym}$.
Da $f^{(r)} \neq f^{(r-1)}$, kann $f^{(r-1)}$ nicht stark vertauschungssymmetrisch in $(x_i, x_{j_r})$ sein und wegen $p(x_i) = p(x_{j_r})$ daher auch nicht stark $G_{\Pi',p}^{sym}$–symmetrisch.
Entsprechendes gilt für $\rho^r = \mu_{ij_r}$.
„$\Longleftarrow$ “
Annahme: $f^{(r-1)}$ ist nicht stark $G_{\Pi',p}^{sym}$–symmetrisch. Es ist zu zeigen, daß $f^{(r)} \neq f^{(r-1)}$.
Da $f^{(r-1)}$ nicht stark $G_{\Pi',p}^{sym}$–symmetrisch ist, existiert eine Gewichtsklasse[†]

$$C := C_{w_2,\ldots,w_k}^{\Pi',p} = \underbrace{C_{0,w_2,\ldots,w_k}^{\Pi,p}}_{C_0} \cup \underbrace{C_{1,w_2-1,\ldots,w_k}^{\Pi,p}}_{C_1}$$

mit $| f^{(r-1)}(C) | > 1$. Da $f$ stark $G_{\Pi,p}^{sym}$–symmetrisch ist, gilt $| f(C_0) | = | f(C_1) | = 1$. Da $f$ zusätzlich noch symmetrisch in $(x_i, x_j)$ ist, kann daher nur $\{b, *\} \subseteq f^{(r-1)}(C)$ für ein $b \in \{0, 1\}$ gelten.
Man unterscheidet:
<u>Fall 1:</u> $* \in f^{(r-1)}(C_1)$
In diesem Fall ist $C$ 0-einseitig für $f$ unter $\Pi$, da $f$ stark $G_{\Pi,p}^{sym}$–symmetrisch.
Aus Lemma 3.11 folgt $w_2 > r - 1$.
Man betrachte eine Funktionsstelle $\epsilon \in C_1$ mit $\epsilon_i = p(x_i)$, $\epsilon_{j_q} = p(x_{j_q})$, $1 \leq q \leq r - 1$ und $\epsilon_{j_r} \neq p(x_{j_r})$.
Dann ist $\rho^q(\epsilon) = \epsilon$ für $1 \leq q \leq r - 1$ und $\rho^r(\epsilon) \in C_0$.
Da $f^{(r)} = f_{\rho^r}^{(r-1)}$ und $f^{(r-1)}(\epsilon) \neq f^{(r-1)}(\rho^r(\epsilon))$ , ist $f^{(r)} \neq f^{(r-1)}$.

<u>Fall 2:</u> $* \in f^{(r-1)}(C_0)$
In diesem Fall ist $C$ 1-einseitig für $f$ unter $\Pi$, da $f$ stark $G_{\Pi,p}^{sym}$–symmetrisch.
Aus Lemma 3.11 folgt $w_2 < | \lambda | - (r - 1)$.
Man betrachte eine Funktionsstelle $\epsilon \in C_0$ mit $\epsilon_i \neq p(x_i)$, $\epsilon_{j_q} \neq p(x_{j_q})$, $1 \leq q \leq r - 1$ und $\epsilon_{j_r} = p(x_{j_r})$.
Dann ist $\rho^q(\epsilon) = \epsilon$ für $1 \leq q \leq r - 1$ und $\rho^r(\epsilon) \in C_1$.
Da $f^{(r)} = f_{\rho^r}^{(r-1)}$ und $f^{(r-1)}(\epsilon) \neq f^{(r-1)}(\rho^r(\epsilon))$ , ist $f^{(r)} \neq f^{(r-1)}$.

□

Bisher sind nur spezielle Symmetrierungsfolgen der Art $\rho^{(k)} \in E_p^{i,[x_j]} \subseteq G_{\Pi',p}^{sym}$ betrachtet worden. Es stellt sich die Frage, ob man kürzere Folgen erhält, falls man die Abbildungen beliebig aus $G_{\Pi',p}^{sym}$ wählen kann. Das folgende Lemma besagt, daß dies nicht der Fall ist:

[†]Der Fall $w_2 = 0$ braucht nicht betrachtet zu werden, da die entsprechenden Funktionsstellen niemals in der Symmetrierungsfolge verändert werden.

**Lemma 3.13** *Seien die Voraussetzungen von Satz 3.1 gegeben. Ist* $g^{(0)} := f$ *und*

$$\begin{aligned} g^{(1)} &= g^{(0)}_{\psi^1} \\ &\vdots \\ g^{(k)} &= g^{(k-1)}_{\psi^k}, \end{aligned}$$

*wobei* $\psi^r \in G^{sym}_{\Pi',p} \cap \{\sigma_{pq}, \mu_{pq} \mid p, q \in \{1, \ldots, n\}\}$, $1 \leq r \leq k$.
*Dann gilt für alle* $1 \leq r \leq k$: $\exists m \leq r$ *mit* $f^{(m)} = g^{(r)}$.

**Beweis:** Sei $X = \{x_1, \ldots, x_n\}$ und $\lambda = \{x_i\} \cup [x_j]$.
Zu jedem $g^{(r)}$ wird eine äquivalente Folge von $f$ konstruiert, die nicht länger als $r$ ist.
Der Beweis wird mittels Induktion über $r$ geführt:
Induktionsanfang: $r = 0$
Dann ist $g^{(0)} = f^{(0)} = f$.
Induktionsschritt: $r \to r + 1$
Sei $f^{(m)} = g^{(r)}$ und $m \leq r$.
In Abhängigkeit von $\psi^{r+1}$ unterscheidet man:
<u>Fall 1:</u> $\psi^{r+1} \in \{\sigma_{st}, \mu_{st}\}$ mit $x_s, x_t \notin \lambda$.
Dann ist $f^{(m)}$ stark $\{\psi^{r+1}\}$–symmetrisch und daher $f^{(m)} = g^{(r)} = g^{(r+1)}$.
<u>Fall 2:</u> $\psi^{r+1} \in \{\sigma_{st}, \mu_{st}\}$ mit $\{x_s, x_t\} \subseteq \lambda$
<u>Fall 2.1:</u> $x_i \in \{x_s, x_t\}$.
O.B.d.A. $x_s = x_i$.
Dann gibt es für $x_t$ zwei Möglichkeiten. Entweder wurde $x_t$ schon in der Folge für $f^{(m)}$ abgehandelt, d.h. es existiert eine Abbildung $\rho^w \in \{\sigma_{it}, \mu_{it}\}$ für ein $1 \leq w \leq m$. Dann ist aber $\rho^w = \psi^{r+1}$ und nach Lemma 3.9 $f^{(m)} = g^{(r)}$ stark $\{\psi^{r+1}\}$–symmetrisch und somit $g^{(r)} = g^{(r+1)} = f^{(m)}$.
Wurde $x_t$ noch nicht abgehandelt und wählt man $f^{(m+1)} = f^{(m)}_{\psi^{r+1}}$, dann ist $g^{(r+1)} = f^{(m+1)}$.
<u>Fall 2.2:</u> $x_i \notin \{x_s, x_t\}$.
Man muß wiederum unterscheiden:

($\star$) $x_s$ und $x_t$ wurden beide in der Folge für $f^{(m)}$ abgehandelt.
Sei $\rho' \in G^{sym}_{\Pi',p} \cap \{\sigma_{is}, \mu_{is}\}$ und $\rho'' \in G^{sym}_{\Pi',p} \cap \{\sigma_{it}, \mu_{it}\}$. Nach Lemma 3.9 ist $f^{(m)} = g^{(r)}$ stark $\{\rho'\}$–symmetrisch und stark $\{\rho''\}$–symmetrisch und aufgrund der Transitivität der starken Symmetrie somit auch $\{\psi^{r+1}\}$–symmetrisch, d.h. $g^{(r)} = g^{(r+1)} = f^{(m)}$.

($\star$) $x_s$ und $x_t$ wurden beide in der Folge für $f^{(m)}$ nicht abgehandelt.
Da $f$ stark $\{\psi^{r+1}\}$–symmetrisch ist und wegen Lemma 3.4 auch $f^{(m)}$, gilt $g^{(r)} = g^{(r+1)} = f^{(m)}$.

($\star$) Genau eine der Variablen $x_s$ oder $x_t$ wurde in der Folge für $f^{(m)}$ abgehandelt. O.B.d.A. sei das $x_s$.
Sei $\rho' \in G^{sym}_{\Pi',p} \cap \{\sigma_{it}, \mu_{it}\}$, $\rho'' \in G^{sym}_{\Pi',p} \cap \{\sigma_{is}, \mu_{is}\}$ und $f^{(m+1)} = f^{(m)}_{\rho'}$.
Es ist zu zeigen, daß $f^{(m+1)} = g^{(r+1)}$.

$f^{(m+1)}$ ist stark $\{\rho', \rho''\}-$symmetrisch und aufgrund der Transitivität der starken Symmetrie auch stark $\{\psi^{r+1}\}-$symmetrisch.
Da $f^{(m+1)} \in Erw(g^{(r)})$, gilt wegen Lemma 3.1 auch $f^{(m+1)} \in Erw(g^{(r+1)})$.
Da $g^{(r)}$ stark $\{\rho''\}-$symmetrisch ist, gilt dies nach Lemma 3.9 auch für $g^{(r+1)}$.
Aus der zusätzlichen starke Symmetrie bzgl. $\{\psi^{r+1}\}$ folgt für $g^{(r+1)}$ auch eine starke Symmetrie bzgl. $\{\rho'\}$.
Da $g^{(r+1)}$ stark $\{\rho'\}-$symmetrisch ist und $g^{(r+1)} \in Erw(f^{(m)})$, gilt wegen Lemma 3.1 auch $g^{(r+1)} \in Erw(f^{(m+1)})$.
Das bedeutet $g^{(r+1)} = f^{(m+1)}$.

□

Lemma 3.13 besagt also, daß zu jeder Symmetrierungsfolge mittels Abbildungen aus $G^{sym}_{\Pi',p}$ eine dieser Folge äquivalente und evtl. kürzere Folge angegeben werden kann, deren Symmetrierungen nur aus der Menge $E_p^{i,[x_j]}$ bestehen. Da für die Folge von Satz 3.1 die einzelnen Symmetrierungen in ihrer Reihenfolge willkürlich gewählt werden konnten, berechnet *compute_length* tatsächlich die Länge der kürzesten Folge unter allen Symmetrierungsfolgen mit Symmetrierungen mittels Abbildungen aus $G^{sym}_{\Pi',p}$.

Das nächste Beispiel zeigt allerdings, daß unter Umständen noch kürzere Folgen existieren, falls man die Symmetrierungen völlig beliebig wählen darf:

**Beispiel**
Man betrachte die Funktion $f \in BP_6$ mit

$$f(\epsilon) = \begin{cases} * & \text{falls } \epsilon \in \{(0,0,0,1,1,0),(0,0,1,0,1,0)(0,1,0,0,1,0)\} \\ 1 & \text{sonst} \end{cases}$$

Sei $\Pi = \{\{x_1\}, \{x_2, x_3, x_4\}, \{x_5\}, \{x_6\}\}$. Dann ist $f$ stark symmetrisch in $\Pi$, symmetrisch in $\{x_1, x_2\}$ und in $\{x_5, x_6\}$.
Um mittels Satz 3.1 eine symmetrische Erweiterung für die Symmetriemenge $\{x_1\} \cup \{x_2, x_3, x_4\}$ zu bilden, sind genau drei Symmetrierungen notwendig. Bildet man jedoch $f_{\sigma_{56}} = \mathbf{1}$, so erhält man eine Erweiterung, die stark symmetrisch in $\Pi' = \{\{x_1, x_2, x_3, x_4\}, \{x_5\}, \{x_6\}\}$ ist.

### 3.2.1 Multiformsymmetrische Partitionen

In diesem Abschnitt sollen Symmetrierungsfolgen speziell für multiformsymmetrische Partitionen betrachtet werden. Aus dem vorigen Kapitel ist bekannt, daß zwei starke multiformsymmetrische Variablenmengen zu einer gemeinsamen (schwachen) multiformsymmetrischen Mengen vereinigt werden können (vgl. Satz 2.6). Die Größe der beteiligten Mengen spielte dabei keine Rolle, es mußte lediglich eine Multiformsymmetrie zwischen den beiden Mengen bestehen.
Die besonderen Eigenschaften der Multiformsymmetrie ermöglichen es, für die

Berechnung der entsprechenden symmetrischen Erweiterung einer multiformsymmetrischen Partition immer mit einer *konstanten* Folge von Symmetrierungen der Art $f_{\sigma_{ij}}, f_{\mu_{ij}}$ auszukommen.
Folgendes Korollar ergibt sich unmittelbar aus der Definition der Gewichtsklassen und der Abbildung $v_{ij}$:

**Korollar 3.5** *Sei* $\Pi = \{\lambda_1, \ldots, \lambda_k\}$ *eine Partition der Variablenmenge* $X$ *und* $p$ *eine beliebige Phasenabbildung auf* $X$*. Ist* $[x_i] = \lambda_r$ *und* $[x_j] = \lambda_s$*, dann gilt für jeden Vektor* $(\varepsilon_1, \ldots, \varepsilon_k) \in \{0,1\}^k$ *:*

$$\forall \epsilon \in C^{\Pi,p}_{[\varepsilon_1],\ldots,[\varepsilon_r],\ldots,[\varepsilon_s],\ldots,[\varepsilon_k]} \Rightarrow v_{ij}(\epsilon) \in C^{\Pi,p}_{[\varepsilon_1],\ldots,[\overline{\varepsilon_r}],\ldots,[\overline{\varepsilon_s}],\ldots,[\varepsilon_k]}$$

Nun kann die Berechnung der symmetrischen Erweiterung für die multiformsymmetrische Partition aus Satz 2.6 angegeben werden:

**Satz 3.3** *Sei* $f \in BP_n$ *eine partielle Boolesche Funktion und* $X = X_1 \cup X_2$ *eine Variablenmenge mit* $X_1 \cap X_2 = \emptyset$*. Sei* $\Pi_1$ *eine Partition von* $X_1$*,* $\Pi_2$ *eine Partition von* $X_2$ *und* $[x_i], [x_j] \in \Pi_1$*.*
*Ist* $f$ *stark symmetrisch in* $\Pi_2$*, stark multiformsymmetrisch in* $\Pi_1$ *und multiformsymmetrisch in* $(x_i, x_j)$*, dann ist* $f_{v_{ij}}$ *stark symmetrisch in* $\Pi_2$ *und stark multiformsymmetrisch in* $\Pi_1' = \Pi_1 \backslash \{[x_i],[x_j]\} \bigcup \{[x_i] \cup [x_j]\}$*.*

**Beweis:** Aus Lemma 3.4 folgt, daß $g$ stark symmetrisch in $\Pi_2$ ist. Es bleibt daher nur noch zu zeigen, daß $g$ auch stark multiformsymmetrisch in $\Pi_1'$ ist.
O.B.d.A. $\Pi_1 = \{\lambda_1, \ldots, \lambda_k\}$ und $[x_i] = \lambda_1$, $[x_j] = \lambda_2$ mit $\lambda_1 \cap \lambda_2 = \emptyset$. Weiter sei $\Pi_1' = \{\lambda_1 \cup \lambda_2, \lambda_3, \ldots, \lambda_k\}$.
Ist $p$ eine beliebige Phasenabbildung auf $X_1$, dann läßt sich $C^{\Pi_1',p}_{[\epsilon_2],\ldots,[\epsilon_k]}$, als disjunkte Vereinigung zweier Mengen von Gewichtsklassen darstellen:

$$C^{\Pi_1',p}_{[\epsilon_2],[\epsilon_3],\ldots,[\epsilon_k]} = C^{\Pi_1,p}_{[0],[\epsilon_2],[\epsilon_3],\ldots,[\epsilon_k]} \cup C^{\Pi_1,p}_{[1],[\overline{\epsilon_2}],[\epsilon_3],\ldots,[\epsilon_k]}$$

Da $f$ stark multiformsymmetrisch in $\Pi_1$ ist, gilt $| f(C^{\Pi_1,p}_{[1][\overline{\epsilon_2}],[\epsilon_3],\ldots,[\epsilon_k]}) |= 1$ und $| f(C^{\Pi_1,p}_{[0],[\epsilon_2],[\epsilon_3],\ldots,[\epsilon_k]}) |= 1$.
Aus Satz 2.6 folgt $\{0,1\} \not\subseteq f(C^{\Pi_1',p}_{[\epsilon_2],[\epsilon_3],\ldots,[\epsilon_k]})$.
Aufgrund von Korollar 3.5 gilt $| f_{v_{ij}}(C^{\Pi_1',p}_{[\epsilon_2],[\epsilon_3],\ldots,[\epsilon_k]}) |= 1$, d.h. $g$ ist stark multiformsymmetrisch in $\Pi_1'$.

$\square$

Somit reicht die Bestimmung *einer* notwendigen symmetrischen Erweiterung $f_{v_{ij}}$ aus, um zwei stark multiformsymmetrische Mengen zu einer gemeinsamen multiformsymmetrischen Menge zu vereinigen.
Aufgrund von Lemma 3.3 kann $f_{v_{ij}}$ auch durch zweimalige Symmetrierung mittels $\sigma_{ij}$ und $\mu_{ij}$ berechnet werden. Damit erhält man stets eine Folge der Länge 2.

### 3.2.2 Unabhängige Variablenmengen

Auch unabhängige Variablenmengen lassen sich durch eine Folge von Symmetrierungen der Art $f_{\upsilon_i}$ in stark unabhängige Mengen umformen. Dazu reicht es aus, jede Variable aus der Menge nur einmal zur Symmetrierung zu verwenden, wie das folgende Lemma zeigt:

**Lemma 3.14** *Sei $f \in BP_n$ eine partielle Boolesche Funktion, $X$ eine Variablenmenge und $\lambda \subseteq X$ mit $\lambda = \{x_{j_1}, \ldots, x_{j_k}\}$. Sei $f^{(0)} := f$ und*

$$\begin{aligned} f^{(1)} &= f^{(0)}_{\upsilon_{j_1}} \\ &\vdots \\ f^{(k)} &= f^{(k-1)}_{\upsilon_{j_k}}. \end{aligned}$$

*Ist $f$ unabhängig von $\lambda$, dann ist $f^{(k)}$ stark unabhängig von $\lambda$.*

**Beweis:** Man zeigt mittels Induktion über $r \in \{0, \ldots, k\}$:
Behauptung: $f^{(r)}$ ist stark unabhängig von $\{x_{j_1}, \ldots, x_{j_r}\}$ und unabhängig von $\lambda$.

Sei $g \in Erw(f)$ und $g$ stark unabhängig von $\lambda$.
Induktionsanfang $r = 0$:
Es ist $\{x_{j_1}, \ldots, x_{j_r}\} = \emptyset$ und $g \in Erw(f)$.
Induktionsschritt $r - 1 \to r$:
Da $f^{(r)} = f^{(r-1)}_{\upsilon_{j_r}}$, ist $f^{(r)}$ stark unabhängig von $x_{j_r}$. Wegen der Induktionsvoraussetzung und Lemma 3.4 ist $f^{(r)}$ auch stark unabhängig von $\{x_{j_1}, \ldots, x_{j_{r-1}}\}$. Nach der Induktionsvoraussetzung ist $g \in Erw(f^{(r-1)})$. Da $g$ auch stark unabhängig von $x_{j_r}$ ist, gilt aufgrund von Lemma 3.1 $g \in Erw(f^{(r)})$, d.h. $f^{(r)}$ ist unabhängig von $\lambda$.

$\square$

Abschließend soll der Einfluß der symmetrischen Erweiterungen $f_{\upsilon_i}$ auf die stark symmetrischen bzw. stark multiformsymmetrischen Partitionen einer partiellen Booleschen Funktion $f$ untersucht werden.
Eine stark multiformsymmetrische Menge $\lambda$, die zusätzlich eine unabhängige Variable enthält, ist selbst unabhängig. Die symmetrische Erweiterung, welche stark unabhängig von $\lambda$ ist, läßt sich einfach bestimmen, wie folgendes Lemma zeigt:

**Lemma 3.15** *Sei $f \in BP_n$ eine partielle Boolesche Funktion, $X$ eine Variablenmenge, $\lambda \subseteq X$ und $x_i \in \lambda$. Ist $f$ stark multiformsymmetrisch in $\lambda$ und unabhängig von $x_i$, dann ist $f_{\upsilon_i}$ stark unabhängig von $\lambda$.*

**Beweis:** O.B.d.A. $\lambda = \{x_1, \ldots, x_l\}$ und $\Pi = \{\lambda, \{x_{l+1}\}, \ldots \{x_n\}\}$.
Da man multiformsymmetrische Mengen betrachtet, kann die Phasenabbildung beliebig gewählt werden. Sei $p : X \to \{1\}$.

Ist $f$ stark multiformsymmetrisch in $\lambda$, dann ist $f$ auch stark multiformsymmetrisch in der Partition $\Pi$.
Damit gilt für alle $\varepsilon$ $\{0,1\}^k$ mit $k = n - l$

$$|\, f(C^{\Pi,p}_{[1],[\varepsilon_1],\ldots,[\varepsilon_k]})\,| = |\, f(C^{\Pi,p}_{[0],[\varepsilon_1],\ldots,[\varepsilon_k]})\,| = 1.$$

Offensichtlich ist für $G := < \{\nu_t \in \mathrm{P_n} \mid 1 \leq t \leq l\} >$

$$orbit_G(\epsilon) = C^{\Pi,p}_{[1],[\epsilon_{l+1}],\ldots,[\epsilon_n]} \cup C^{\Pi,p}_{[0],[\epsilon_{l+1}],\ldots,[\epsilon_n]},$$

da $|\,\lambda_t\,| = 1$ $(l+1 \leq t \leq n)$.
Wegen $x_i \in \lambda$, gilt für alle $\epsilon \in \{0,1\}^n$

$$\epsilon \in C^{\Pi,p}_{[1],[\epsilon_{l+1}],\ldots,[\epsilon_n]} \quad \Longleftrightarrow \quad \nu_i(\epsilon) \in C^{\Pi,p}_{[0],[\epsilon_{l+1}],\ldots,[\epsilon_n]}.$$

Da $f$ unabhängig von $x_i$ ist, gilt $\{0,1\} \not\subseteq f(orbit_G(\epsilon))$ und daher $|\, f_{\upsilon_i}(orbit_G(\epsilon))\,| = 1$. $f_{\upsilon_i}$ somit stark unabhängig von $\lambda$.

□

Der folgende Satz faßt die Zusammenhänge zwischen den stark symmetrischen und stark multiformsymmetrischen Partitionen einer partiellen Funktion $f$ sowie den entsprechenden Partitionen für $f_{\upsilon_i}$ zusammen:

**Satz 3.4** *Sei $f \in BP_n$ eine partielle Boolesche Funktion und $X = X_1 \cup X_2$ eine Variablenmenge mit $X_1 \cap X_2 = \emptyset$. Sei $\Pi_1$ eine Partition von $X_1$, $\Pi_2$ eine Partition von $X_2$ und $[x_i] \in \Pi_1$, $[x_j] \in \Pi_2$.*
*Ist $f$ stark symmetrisch in $\Pi_1$, stark multiformsymmetrisch in $\Pi_2$ sowie unabhängig von $x_i$ und $x_j$, dann gilt:*
*(1) $f_{\upsilon_i}$ ist stark unabhängig von $x_i$, stark symmetrisch in $\Pi_1' = \Pi_1 \backslash \{[x_i]\} \bigcup \{[x_i] \backslash \{x_i\}, \{x_i\}\}$ und stark multiformsymmetrisch in $\Pi_2$.*
*(2) $f_{\upsilon_j}$ stark unabhängig von $[x_j]$, stark multiformsymmetrisch in $\Pi_2$ und stark symmetrisch in $\Pi_1$.*

**Beweis:** zu (1)
Nach Lemma 3.2 ist $f_{\upsilon_i}$ stark unabhängig von $x_i$.
Aus Lemma 3.4 folgt, daß $f_{\upsilon_i}$ stark symmetrisch in $\Pi_1'$ und stark multiformsymmetrisch in $\Pi_2$ ist.
zu (2)
Nach Lemma 3.2 ist $f_{\upsilon_j}$ stark unabhängig von $x_j$.
Aus Lemma 3.4 folgt, daß $f_{\upsilon_j}$ stark symmetrisch in $\Pi_1$ ist.
Lemma 3.15 besagt, daß $f_{\upsilon_j}$ auch stark unabhängig von $[x_j]$ ist.
Da nach Lemma 2.19 jede stark unabhängige Variablenmenge auch stark multiformsymmetrisch ist, bleibt $f_{\upsilon_j}$ stark multiformsymmetrisch in $\Pi_2$.

□

# Kapitel 4

# Minimale symmetrische Partitionen

In diesem Kapitel wird das Problem untersucht, zu einer gegebenen partiellen Booleschen Funktion eine möglichst kleine Partition zu bestimmen, in der die Funktion symmetrisch ist. Das entsprechende Optimierungsproblem, d.h. die Bestimmung einer kleinsten symmetrischen Partition, ist das Problem *MSP* (Minimale Symmetrische Partition).
Bei der Lösung dieses Problems ergeben sich einige Schwierigkeiten: Zum einen existieren exponentiell viele Teilmengen der Variablenmenge, so daß evtl. sehr viele dieser Mengen auf Symmetrie getestet werden müssen (z.B. mittels des Verfahrens aus [KD91]). Zum anderen haben die vorhergehenden Untersuchungen bereits gezeigt, daß es für partielle Funktionen im allgemeinen nicht möglich ist, von einer Symmetrie in den beiden Mengen $\lambda_1$ und $\lambda_2$ auf die Existenz *einer* Erweiterung zu schließen, welche zugleich stark symmetrisch in $\lambda_1$ und $\lambda_2$ ist.
Das Kapitel beginnt zunächst mit der Bestimmung der Komplexität des Problems *MSP*. Es stellt sich heraus, daß *MSP* *NP*-hart ist und daher in der Praxis Heuristiken verwendet werden müssen, um dieses Problem (näherungsweise) zu lösen. Im zweiten Teil des Kapitels werden eine Heuristik zur Lösung von *MSP* vorgestellt und entsprechende experimentelle Resultate angegeben.

## 4.1 Das Problem *MSP*

Die Bestimmung einer symmetrischen Partition mit wenigen Symmetriemengen bringt u.a. folgende Vorteile:

- Eine Minimierung der Symmetriemengen liefert meist auch wenige *große* Symmetriemengen. Solche große Mengen können die Größe der Darstellung und die Komplexität der Realisierung erheblich reduzieren (vgl. Kapitel 2, insbesondere totalsymmetrische Funktionen).

- Speziell für die Darstellung der symmetrischen Erweiterung durch ROBDDs muß noch eine der symmetrischen Partition entsprechende symmetrische Variablenordnung generiert werden. Dazu werden die Variablen, die sich innerhalb einer Symmetriemenge befinden, zu einem Block zusammengefaßt, d.h. sie können nicht mehr untereinander vertauscht werden, da dies für die Größe des ROBDDs unerheblich ist. Die Suche nach einer guten symmetrischen Ordnung reduziert sich also auf die Permutation der einzelnen Symmetriemengen in der Variablenordnung. Je weniger Mengen vorhanden sind, desto schneller kann der entsprechende Algorithmus (z.B. *sym_sift* [MMD94, PSP94]) eine Lösung finden.

Daher ist man an der Lösung des folgenden Problems interessiert:

Problem: **MSP** **M**inimale **S**ymmetrische **P**artition

*Gegeben*: Partielle Boolesche Funktion $f \in BP_n$.

*Gesucht*: Partition $\Pi$ der Variablenmenge $X = \{x_1, \ldots, x_n\}$, so daß gilt:

- $f$ ist symmetrisch in $\Pi$ und
- für jede symmetrische Partition $\Pi'$ von $f$ gilt $\mid \Pi \mid \leq \mid \Pi' \mid$.

Um die Komplexität dieses Problems zu beweisen, ist es zweckmäßig, alle Symmetrien einer Funktion in einem *Symmetriegraphen* darzustellen:

**Definition 4.1 (Symmetriegraph)** *Sei $f \in BP_n$ eine partielle Boolesche Funktion und $X = \{x_1, \ldots, x_n\}$ eine Variablenmenge. $\mathcal{G}^f_{sym} = (V, E)$ heißt* **Symmetriegraph** *von $f$ genau dann, falls*

- *$V = X$ und*
- *$\{x_i, x_j\} \in E \iff f$ ist symmetrisch in $(x_i, x_j)$.*

Symmetrische Partitionen bilden innerhalb des Symmetriegraphen spezielle Strukturen:

**Lemma 4.1** *Sei $f \in BP_n$ eine partielle Boolesche Funktion, $X$ eine Variablenmenge, $\Pi$ eine Partition von $X$ und $\mathcal{G}^f_{sym}$ der Symmetriegraph von $f$. Ist $f$ symmetrisch in $\Pi$, dann bilden alle Knoten für $\lambda \in \Pi$ eine Clique in $\mathcal{G}^f_{sym}$.*

**Beweis:** Ist $f$ symmetrisch in $\Pi$, dann existiert eine Erweiterung $g \in Erw(f)$, welche stark symmetrisch in $\Pi$ ist.
Damit ist $g$ stark symmetrisch in $\lambda \in \Pi$ und daher auch stark symmetrisch in allen Variablenpaaren aus $\lambda \times \lambda$. Da $g \in Erw(f)$, ist auch $f$ symmetrisch in diesen Variablenpaaren.
Damit sind alle Variablen aus $\lambda$ in $\mathcal{G}^f_{sym}$ adjazent zueinander.

□

Deshalb führt jede symmetrische Partition einer partiellen Booleschen Funktion zu einer Zerlegung des Symmetriegraphen in disjunkte Cliquen.

**Bemerkung 11** *Es ist zu beachten, daß die Umkehrung von Lemma 4.1 im allgemeinen* nicht *gilt, d.h. nicht jede Partition des Symmetriegraphen* $\mathcal{G}^f_{sym}$ *in Cliquen bildet auch gleichzeitig eine symmetrische Partition für* $f$*. Ein Beispiel hierfür bildet die Boolesche Funktion* $f$ *im Beweis zu Lemma 2.9 auf Seite 59:* $\mathcal{G}^f_{sym}$ *besteht gerade aus der Clique* $\{x_1, x_2, x_3, x_4\}$*, aber* $f$ *ist* nicht *symmetrisch in* $\{x_1, x_2, x_3, x_4\}$*.*

Für die Komplexität von *MSP* gilt:

**Satz 4.1** *Das Problem* MSP *ist NP-hart, falls* $f$ *durch* ROBDDs *repräsentiert wird.*

Der Satz wird bewiesen durch Angabe einer Polynomzeittransformation des *NP*-vollständigen Problems „Partition into Cliques“ (PC) nach *MSP*.
Das Problem PC lautet ([GJ79], S.193):

> **Partition into Cliques (PC)**
> *Gegeben:* Ein ungerichteter Graph $G = (V, E)$ und eine natürliche Zahl $K \leq |V|$.
> *Gesucht:* Gibt es eine Partition $Cl = \{cl_1, \ldots, cl_k\}$ von $G$ in Cliquen mit $k \leq K$ ?

**Beweis:** Gegeben sei eine Instanz des Problems PC bestehend aus einem Graphen $G = (V, E), |V| = n$ und einer natürlichen Zahl $K \leq |V|$.
Sei $\epsilon^i := (\underbrace{1, 1, \ldots, 1}_{i-mal}, 0, 0, \ldots, 0, 0) \in \{0,1\}^{n+2}$ , $0 \leq i \leq n+2$.
Zu $G$ bestimmt man die folgende partielle Boolesche Funktion $f \in BP_{n+2}$ :

$$f(\epsilon) = \begin{cases} 1 & \text{falls } \epsilon = \epsilon^i, 0 \leq i \leq n+1 \\ 1 & \text{falls } \epsilon = (0, 0, \ldots, 0, 1) \\ 0 & \text{falls } \sigma_{1i}(\epsilon) = \epsilon^1, 2 \leq i \leq n+1 & (1) \\ 0 & \text{falls } \{v_i, v_j\} \notin E, i < j \wedge \sigma_{i+1\,j+1}(\epsilon) = \epsilon^{i+1} & (2) \\ & \text{für } i, j \in \{1, \ldots, n\} \\ 0 & \text{falls } \mu_{ij}(\epsilon) = (0, 0, \ldots, 0, 1) \text{ für } i, j \in \{1, \ldots, n+1\} & (3) \\ 0 & \text{falls } \mu_{i\,n+2}(\epsilon) = \epsilon^0, 1 \leq i \leq n+1 & (4) \\ * & \text{sonst} \end{cases}$$

Die Funktion ist offensichtlich wohldefiniert.

Die Anzahl der definierten Funktionsstellen besitzt eine polynomielle Größe bezogen auf die Eingabelänge. Deshalb kann die Funktion auch durch ROBDDs in polynomieller Größe dargestellt und ebenfalls in polynomieller Zeit konstruiert werden[†].

Es folgt nun eine genaue Charakterisierung der Symmetrien von $f$.
Sei $X = \{x_1, \ldots, x_{n+2}\}$ eine Menge von Variablen.

[†]Allgemein gilt: Zu jeder Booleschen Funktion $f \in B_n$ mit $|ON(f)| = N$ (bzw. $|OFF(f)| = N$) kann ein ROBDD $F$ mit $f_F = f$ in Zeit $O(nN\log(nN))$ konstruiert werden. Darüber hinaus gilt für die Größe von $F$ ebenfalls $|F| = O(nN\log(nN))$. Eine detaillierte Konstruktion findet der Leser in [Sch96] S.54f.

Behauptung:

- (i) $f$ ist nicht symmetrisch in $(x_i, x_{n+2})$,$1 \leq i \leq n+1$.
- (ii) $f$ ist nicht symmetrisch in $(x_1, x_i)$,$2 \leq i \leq n+1$.
- (iii) $f$ ist nicht äquivalenzsymmetrisch in $(x_i, x_j)$ für $i, j \in \{2, \ldots, n+1\}$.
- (iv) $f$ ist nicht vertauschungssymmetrisch in $(x_{i+1}, x_{j+1})$, falls $\{v_i, v_j\} \notin E$.
- (v) $f$ ist vertauschungssymmetrisch in $(x_{i+1}, x_{j+1})$, falls $\{v_i, v_j\} \in E$.

(i)
Aus (1) folgt $f(0, 0, \ldots, \underbrace{1}_{i}, \ldots, \ldots, 0) = 0$ für alle $2 \leq i \leq n+1$.
Da $f(0, 0, \ldots, 0, 1) = 1$, ist $f$ nicht vertauschungssymmetrisch in $(x_i, x_{n+2})$, $2 \leq i \leq n+1$.
Betrachtet man $\epsilon = \epsilon^2$ und $\epsilon' = \sigma_{1\,n+2}(\epsilon) = \mu_{2\,n+2}(\epsilon^0)$, dann folgt aus (4) $f(\epsilon) = 1$ und $f(\epsilon') = 0$. Daher ist $f$ auch nicht vertauschungssymmetrisch in $(x_1, x_{n+2})$.
Aus (4) folgt, daß $f$ nicht äquivalenzsymmetrisch in $(x_i, x_{n+2})$,$1 \leq i \leq n+1$ ist.
(ii)
Aus (1) folgt, daß $f$ nicht vertauschungssymmetrisch in $(x_1, x_i)$,$2 \leq i \leq n+1$ ist.
Aus (3) folgt, daß $f$ nicht äquivalenzsymmetrisch in $(x_1, x_i)$,$2 \leq i \leq n+1$ ist.
(iii)
Folgt aus (3).
Bevor die beiden Aussagen (iv) und (v) bewiesen werden, zeigt man zunächst folgende Eigenschaft $(\star)$:

Sei $\epsilon \in \{0,1\}^{n+2}$ mit $k = \mid \epsilon \mid^{\dagger} \geq 2$ und $\epsilon_{n+2} = 0$. Dann gilt:
$f(\epsilon) = 0 \iff \exists \sigma_{kj} \in \mathrm{P_{n+2}}$ mit $\sigma_{kj}(\epsilon) = \epsilon^k$, wobei $\{v_{k-1}, v_{j-1}\} \notin E$, $j \in \{2, \ldots, n+1\}$.

Die Aussage „$\Longleftarrow$ “
ergibt sich unmittelbar aus (2). Es bleibt daher noch „$\Longrightarrow$ “
zu zeigen.
Da $\mid \epsilon \mid \geq 2$ und $\epsilon_{n+2} = 0$, kann $\epsilon$ nur aus (2) hervorgegangen sein. Dies beschreibt aber gerade die rechte Seite der Behauptung.
(iv)
Folgt unmittelbar aus $(\star)$ bzw. (2).
(v)
Sei $f$ nicht vertauschungssymmetrisch in $(x_i, x_j)$ $(1 < i < j \leq n+1)$, dann existiert eine Funktionsstelle $\epsilon \in \{0,1\}^{n+2}$ mit $f(\epsilon) = 1$ und $f(\sigma_{ij}(\epsilon)) = 0$.
Man unterscheidet folgende Fälle:

$^{\dagger}\mid \epsilon \mid := \mid \epsilon \mid_X^p$ mit $p(x) = 1 \quad \forall x \in X$.

<u>Fall 1:</u> $\epsilon = (0, 0, \ldots, 0, 1)$.
Dann ist aber $\sigma_{ij}(\epsilon) = \epsilon$.
<u>Fall 2:</u> $\epsilon = \epsilon^k$.
In Abhängigkeit von $k$ wird unterschieden:
<u>Fall 2.1:</u> $k = 0$. Dann ist aber $\sigma_{ij}(\epsilon) = \epsilon$.
<u>Fall 2.2:</u> $k = 1$. Da $1 < i < j$, folgt $\sigma_{ij}(\epsilon) = \epsilon$.
<u>Fall 2.3:</u> $k > 1$.
Ist $k < i$, dann ist $\epsilon_i = \epsilon_j = 0$ und daher $\sigma_{ij}(\epsilon) = \epsilon$.
Ist $j \geq k$, dann ist $\epsilon_i = \epsilon_j = 1$ und daher $\sigma_{ij}(\epsilon) = \epsilon$.
Ist $i < k < j$, dann gilt für $\sigma_{ij}(\epsilon)$ aber $\sigma_{ij}(\epsilon)_i = 0$ und $\sigma_{ij}(\epsilon)_k = 1$. Da $f(\sigma_{ij}(\epsilon)) = 0$ widerspricht dies aber $(\star)$, da keine Abbildung $\sigma_{kl}$ mit $\sigma_{kl}(\epsilon) = \epsilon^k$ existieren kann.
Ist $i = k$, dann ist wegen $(\star)$ nur $\{v_{i-1}, v_{j-1}\} \notin E$ möglich.

Damit bestehen für $f$ nur Vertauschungssymmetrien. Zudem ist $f$ insbesondere asymmetrisch in $(x_1, x_{n+2})$. Damit gilt in jeder symmetrischen Partition von $f$ $\mid [x_1] \mid = \mid [x_{n+2}] \mid = 1$.
Der Symmetriegraph von $f$ ist also bis auf die beiden zusätzlichen Knoten für $x_1$ und $x_{n+2}$ völlig identisch mit $G$.

Es werden nun die Partitionen von $G$ in Cliquen und die symmetrischen Partitionen von $f$ miteinander verglichen:
Sei $Cl = \{cl_1, \ldots, cl_k\}$ eine Partition von $G$ in Cliquen und $\Pi = \{\lambda_1, \ldots, \lambda_k\}$ die Partition von $X$, die aus $Cl$ mittels Ersetzung der Knoten durch die entsprechenden Variablen $(v_i \hat{=} x_{i+1})$ hervorgeht.
Sei $\Pi' = \Pi \cup \{\{x_1\}, \{x_{n+2}\}\}$, d.h. $\lambda_{k+1} = \{x_1\}, \lambda_{k+2} = \{x_{n+2}\}$.

Behauptung: $f$ ist vertauschungssymmetrisch in $\Pi'$.
Dazu sei eine beliebige Menge $C^{\Pi',p}_{w_1,\ldots,w_{k+2}}$ † gegeben. Es bleibt zu zeigen, daß $\{0, 1\} \not\subseteq f(C^{\Pi',p}_{w_1,\ldots,w_{k+2}})$.
Annahme: $\exists \epsilon, \epsilon' \in C^{\Pi',p}_{w_1,\ldots,w_{k+2}}$ mit $f(\epsilon) = 1$ und $f(\epsilon') = 0$.
<u>Fall 1:</u> $w_{k+2} = 0$.
Dann ist $\epsilon = \epsilon^k$ mit $k = \mid \epsilon \mid$.
Ist $\mid \epsilon' \mid = k \geq 2$, dann gibt es nach $(\star)$ eine Abbildung $\sigma_{kj}$ mit $\sigma_{kj}(\epsilon) = \epsilon'$ und $\{v_{k-1}, v_{j-1}\} \notin E$ $(j \in \{2, \ldots, n+1\})$.
Dann ist aber $[v_{k-1}] \neq [v_{j-1}]$ und $f$ nicht symmetrisch in $(x_k, x_j)$.
Daher können $\epsilon$ und $\epsilon'$ nicht in einer Gewichtsklasse liegen. Dies widerspricht aber der Annahme.
Ist $\mid \epsilon \mid \leq 1$, dann kann nur $\epsilon \in \{\epsilon^0, \epsilon^1\}$ gelten, d.h. $w_i = 0, 1 \leq i \leq k$ und $w_{k+2} = 0$.
Damit gilt $\mid C^{\Pi',p}_{w_1,\ldots,w_{k+1},w_{k+2}} \mid = 1$. Daher ist $\{0, 1\} \in f(C^{\Pi',p}_{w_1,\ldots,w_{k+1},w_{k+2}})$ unmöglich.
<u>Fall 2:</u> $w_{k+2} = 1$.
Dann kann nur $\epsilon = (0, 0, \ldots, 0, 1)$ gelten, d.h. $w_i = 0, 1 \leq i \leq k+1$.
Somit gilt $\mid C^{\Pi',p}_{w_1,\ldots,w_{k+1},w_{k+2}} \mid = 1$, daher ist $\{0, 1\} \in f(C^{\Pi',p}_{w_1,\ldots,w_{k+1},w_{k+2}})$ unmöglich.

Umgekehrt gilt:

† Da man nur Vertauschungssymmetrien betrachtet, kann allen Variablen eine positive Phase zugeordnet werden.

Ist $\Pi = \{\lambda_1, \dots, \lambda_k, \{x_1\}, \{x_{n+2}\}\}$ eine symmetrische Partition von $f$ und $Cl = \{cl_1, \dots, cl_k\}$ eine Partition von $G$, die aus $\Pi$ mittels Ersetzung der Variablen durch die entsprechenden Knoten ($v_i \hat{=} x_{i+1}$) hervorgeht.
Aufgrund von Lemma 4.1 und dem Aufbau des Symmetriegraphen von $f$ bilden die Mengen $cl_i$ ($1 \leq i \leq k$) Cliquen in $G$.
Somit gilt folgende Äquivalenz:

$G$ läßt sich genau dann in $k \leq K$ disjunkte Cliquen partitionieren, falls es für $f$ eine symmetrische Partition $\Pi$ der Variablenmenge $X$ mit $| \Pi | \leq K + 2$ gibt. Somit ist *MSP* *NP*-hart.†

□

**Bemerkung 12** *Aus dem Beweis zu Satz 4.1 kann* nicht *abgeleitet werden, daß für alle partiellen Booleschen Funktionen $f$ eine Lösung des Problems* PC *für den Symmetriegraphen ebenfalls eine Lösung von* MSP *für $f$ darstellt.*

### 4.1.1 Eine Heuristik für *MSP*

In diesem Abschnitt wird eine Heuristik für das Problem *MSP* vorgestellt, die auf der Bestimmung einer „Färbung" aller Knoten eines ungerichteten Graphen beruht.

**Definition 4.2 (Färbung der Knoten eines Graphen)** *Sei $G = (V, E)$ ein ungerichteter Graph. Eine* **Färbung der Knoten von** *$G$ ist eine Abbildung $color : V \to \mathbf{N}$ mit $\forall \{v_1, v_2\} \in E : \; color(v_1) \neq color(v_2)$.*

**Bezeichnung 18** $Col(color) := | \{color(v) \mid v \in V\} |$

Der direkte Zusammenhang zwischen der Färbung eines Graphen und dessen Partitionierung in Cliquen ergibt sich aus dem folgenden Lemma:

**Lemma 4.2** *Sei $G = (V, E)$ ein ungerichteter Graph, $\overline{G}$ der zu $G$ inverse Graph und $K$ eine natürliche Zahl. Dann gilt:*
*Es gibt eine Partition von $G$ in $K$ Cliquen, genau dann, falls eine Färbung color von $\overline{G}$ existiert mit $Col(color) = K$.*

**Beweis:** „$\Longrightarrow$ "
Sei $\Pi = \{\lambda_1, \dots, \lambda_k\}$ eine Partitionierung von $G$ in Cliquen.
Man betrachte die Abbildung $color : V \to \mathbf{N}$ mit $color(v) = i \iff v \in \lambda_i$.
Sind $v, v' \in V$ und $color(v) = color(v')$, dann gilt $v, v' \in \lambda$ mit $\lambda \in \Pi$.
Damit sind die beiden Knoten aber nicht adjazent in $\overline{G}$ und somit *color* eine Färbung für $\overline{G}$.
„$\Longleftarrow$ "
Sei $color : V \to \mathbf{N}$ eine Färbung von $\overline{G}$.

†Existiert ein Polynomzeitalgorithmus für *MSP*, dann wäre dieser natürlich auch in der Lage, in Polynomzeit zu bestimmen, ob eine symmetrische Partition $\Pi$ existiert mit $| \Pi | < k$.

Man betrachte die Partition $\Pi = \{\lambda_1, \ldots, \lambda_{Col(color)}\}$ mit $\lambda_i = \{v \in V \mid color(v) = i\}$.
Ist $\lambda \in \Pi$, dann gibt es keine Knoten aus $\lambda$, die in $\overline{G}$ adjazent zueinander sind. Damit bildet $\lambda$ eine Clique in $G$.

□

Der in diesem Abschnitt vorgestellte Algorithmus baut auf einer Heuristik von Brélaz [Bre79] zur Bestimmung einer Färbung eines ungerichteten Graphen auf. Die Grundstruktur der Heuristik basiert auf folgender Schleife:

1. Wähle aus der Menge der ungefärbten Knoten einen Kandidaten $v$ aus.
2. Ordne $v$ eine Farbe zu.
3. Falls noch ungefärbte Knoten existieren, gehe zu 1.

Damit ist die Laufzeit der Heuristik proportional zu der Anzahl der Knoten. Es soll nun näher auf die ersten beiden Punkte eingegangen werden, d.h. auf die Auswahl des Knotens und die Bestimmung einer Farbe.
Zu 1.
Jedem ungefärbten Knoten $v \in G$ wird ein Schlüssel $(k_1, k_2)$ zugeordnet:

- Der primäre Schlüssel $k_1$ gibt die Anzahl der Farben an, die in der Nachbarschaft von $v$ vertreten sind. Diesen Wert bezeichnet man mit $satdeg(v)$. Es gilt somit:

$$satdeg(v) := \mid Col(v) \mid \text{ wobei}$$
$$Col(v) := \{color(w) \mid \{w, v\} \in E \wedge color(w) \neq \text{ undef.}\}.$$

- Der sekundäre Schlüssel $k_2$ gibt die Anzahl der ungefärbten Knoten an, die adjazent zu $v$ sind. Diesen Wert bezeichnet man mit $cdeg(v)$, d.h.

$$cdeg(v) := \mid \{w \mid \{w, v\} \in E \wedge color(w) = \text{ undef.}\} \mid .$$

Zu Beginn der Heuristik gilt für alle Knoten des Graphen folglich $satdeg(v) = 0$ und $cdeg(v) = degree(v)$.
Es wird nun jeweils ein ungefärbter Knoten mit *maximalem* Schlüssel ausgewählt. Dieser Knoten stellt intuitiv den am „schwersten“ zu färbenden Knoten dar, da an ihn die meisten Bedingungen bezüglich der Vergabe einer Farbe geknüpft sind.
Zu 2.
Dem ausgewählten Knoten $v$ wird stets die *minimale* mögliche Farbe zugeordnet, d.h. $min\{c \mid c \notin Col(v)\}$. Die Vergabe erfolgt mit dem Ziel, die Anzahl der verwendeten Farben, d.h. $Col(color)$ möglichst zu minimieren.

Nachdem ein Knoten $v$ gefärbt wurde, müssen die Schlüssel aller ungefärbten Knoten an die neue Situation angepaßt werden, um die Konsistenz der Schlüssel zu gewährleisten. Es ist leicht einzusehen, daß sich lediglich die Werte für die zu $v$ adjazenten Knoten ändern.

Die folgende Prozedur führt diese Änderungen durch, nachdem $v$ eine Farbe zugeordnet wurde:

**change_keys(v)**

<u>**foreach**</u> ($\{w, v\} \in G$ mit $color(w) = undef.$) <u>**do**</u>
  $cdeg(w) := cdeg(w) - 1$
  <u>**if**</u> ($color(v) \notin Col(w)$) <u>**then**</u> (*)
    $satdeg(w) := satdeg(w) + 1$
  <u>**fi**</u>
<u>**od**</u>

Eine Implementierung dieser Heuristik durch Morgenstern [Morg92] besitzt für einen ungerichteten Graphen $G = (V, E)$ eine Laufzeit von $O(|V| + |E|)$. Diese Laufzeit wird durch eine spezielle Speicherung der ungefärbten Knoten erreicht, die kurz erläutert werden soll:

- Ein zweidimensionales Feld *avail_set* ermöglicht es, den Test in Zeile (*) der Prozedur *change_keys* in konstanter Zeit auszuführen.
  $avail\_set[i][j]$ gibt dabei für jeden Knoten $v_i$ an, ob $j \in Col(v_i)$. Die Dimensionen des Feldes sind jeweils durch $|V|$ beschränkt. Zu Beginn der Färbung gilt offenbar für alle Einträge $avail\_set[i][j] = 0$.
  Um eine quadratische Laufzeit beim Initialisieren von *avail_set* zu vermeiden, wurde diese Matrix durch ein Feld der Länge $|V|$ von initialisierungsfreien Bitfeldern[†] (ebenfalls der Länge $|V|$) implementiert. Dadurch kann der benötigte Speicherplatz für *avail_set* in Zeit $O(|V|)$ allokiert werden. Nach dieser Allokation sind gleichzeitig alle Einträge in *avail_set* bereits mit 0 initialisiert. Somit erhält man eine initialisierungsfreie Matrix.

- Die Speicherung der *ungefärbten* Knoten mit ihren zugehörigen Schlüsseln erfolgt in einem zweidimensionalen Feld *heap* von Listen. Dabei stellt $heap[k_1][k_2]$ die Liste aller ungefärbten Knoten aus $V$ dar, welche das Schlüsselpaar $(k_1, k_2)$ besitzen.
  Es ist nicht notwendig, zu Beginn der Heuristik alle $|V|^2$ Listen anzulegen, da zu jedem Zeitpunkt der Heuristik höchstens $|V|$ solcher Listen nicht leer sind. Um auch hier eine quadratische Laufzeit zu vermeiden, wird gewährleistet, daß immer nur solche Listen allokiert sind, welche auch wirklich Einträge enthalten. Aus diesem Grund wird nur dann der benötigte Speicherplatz für eine Liste $heap[k_1][k_2]$ zur Verfügung gestellt, wenn einem ungefärbten Knoten $v$ der Schlüssel $(k_1, k_2)$ zugeordnet wird und $v$ überdies der einzige Knoten mit diesem Schlüssel ist. Sollte während der Prozedur *change_keys* aufgrund der Veränderung der Schlüssel eine Liste in *heap* keine Elemente mehr enthalten, so wird der entsprechende Speicherplatz unmittelbar wieder freigegeben.
  Um einen Überblick über die bereits initialisierten Listen in *heap* zu behalten, wird dies in einer weiteren initialisierungsfreien Matrix vermerkt.

[†] Siehe [Mehl86] S. 269f.

Nachdem alle Knoten gefärbt wurden, existiert somit keine Liste mehr, so daß keine weitere Freigabe von (leeren) Listen notwendig wird.

- Um die Bestimmung eines ungefärbten Knotens mit maximalem Schlüssel effizient zu realisieren, gibt ein Vektor *nonempty* für jeden primären Schlüssel $k_1$ die Anzahl der nichtleeren Listen $heap[k_1][k_2]$ an. Der aktuell größte dieser primären Schlüssel wird dabei in der Variablen *maxprimary* gespeichert.
  Bei der Suche nach einem ungefärbten Knoten, durchläuft man, beginnend mit *maxprimary*, das Feld *nonempty* solange, bis man einen positiven Eintrag für $k_1$ gefunden hat. Anschließend durchsucht man die sekundären Schlüssel unter $k_1$, bis man eine nichtleere Liste $heap[k_1][k_2]$ mit maximalem sekundären Schlüssel erreicht.

Um die Heuristik von Brélaz an das Problem *MSP* anzupassen, bedarf es einer modifizierten Prozedur, um eine Farbe auszuwählen.
*get_color* liefert die kleinste Farbe, die einem Knoten $v_i$ zugeordnet werden kann und zugleich größer als der Parameter *limit* ist:

**get_color(**$v_i$, *limit***)**

$j := limit + 1$
**<u>while</u>** $(avail\_set[i][j]! = 0)$ **<u>do</u>** $j{+}{+}$ **<u>od</u>**
**return**(j)

Abbildung 4.1 zeigt die modifizierte Brélaz Heuristik *compute_min_part* für das Problem *MSP*.

### Korrektheit von *compute_min_part*

Zunächst soll gezeigt werden, daß es sich bei der Partition $\Pi$ stets um eine symmetrische Partition für $g \in Erw(f)$ handelt. Zu Beginn der Heuristik ist $\Pi = \{\{x_1\}, \ldots, \{x_n\}\}$ und damit trivialerweise eine symmetrische Partition für $g$.
Die Existenz der symmetrische Erweiterung $g$ in Zeile (*) folgt unmittelbar aus Satz 2.3, da die Voraussetzungen dieses Satzes von $g$ und den beiden Mengen $[x_i], [x_j]$ erfüllt werden. Die Bestimmung von $g$ in Zeile (*) kann entweder durch die Folge von Satz 3.1 oder durch eine entsprechende optimierte Folge aus dem vorigen Kapitel erfolgen. Nach Beendigung der Heuristik stellt die Partition $\Pi$ der Variablenmenge somit eine symmetrische Partition für $g$ und deshalb auch für $f$ dar. Die Ergebnisfunktion $g$ ist zudem stark symmetrisch in $\Pi$.
Damit ist die Korrektheit vollständig bewiesen.

**Algorithmus *compute_min_part*:**

Berechne Symmetriegraph $\mathcal{G}^f_{sym} = (V, E)$ von $f$ (oder $\overline{\mathcal{G}^f_{sym}} = (V, \overline{E})$).
$\forall 1 \leq k \leq n : color(x_k) := undef.$
$\Pi = \{\{x_1\}, \{x_2\}, \ldots, \{x_n\}\}$
$g := f$
$ungefärbte\_knoten := \{x_1, \ldots, x_n\}$
**while** $(ungefärbte\_knoten \neq \emptyset)$ **do**
    /* $g$ ist stark symmetrisch in $\Pi$ */
    $x_i = choose\_node()$
    /*Wähle Knoten aus *ungefärbte_knoten* gemäß Brélaz Kriterium */
    **while** $(color(x_i) = undef.)$ **do**
        $limit := -1$
        $gewählte\_farbe := get\_color(x_i, limit)$
        $color(x_i) := gewählte\_farbe$
        **if** ($\exists$ bereits gefärbter Knoten $x_j$ mit $color(x_j) = color(x_i)$)
          **then**
            **if** ($g$ ist symmetrisch in $(x_i, x_j)$)
              **then**
                $\Pi := \Pi \setminus \{[x_j], \{x_i\}\} \bigcup \{[x_j] \cup \{x_i\}\}$
                /*Bestimme $g$, so daß $g$ stark symmetrisch in $\Pi$ ist */(*)
              **else**
                $limit := gewählte\_farbe$
                $color(x_i) := undef.$
            **fi**
        **fi**
    **od**
    $ungefärbte\_knoten := ungefärbte\_knoten \setminus \{x_i\}$
**od**

Abbildung 4.1: Algorithmus zur Lösung von *MSP*.

**Laufzeit von** *compute_min_part*

Die Gesamtlaufzeit von *compute_min_part* kann nicht $O(|V| + |\overline{E}|)$ betragen, da zusätzlich zu der symmetrischen Partition noch eine entsprechende Erweiterung berechnet wird. Sieht man allerdings von diesen Operationen ab, sowie von der Zeit, die der Aufbau des (inversen) Symmetriegraphen benötigt, so kann die Partition $\Pi$ in Zeit $O(|V|^2)$ bestimmt werden, falls zur Abspeicherung der Schlüssel die Datenstruktur aus [Morg92] verwendet wird.
*compute_min_part* unterscheidet sich von der ursprünglichen Heuristik dadurch, daß eine Auswahl für *gewählte_farbe* in Zeile 12 verworfen werden muß, falls für $g$ *keine* Symmetrie in $(x_i, x_j)$ vorhanden ist.
Im worst-case kann die Laufzeit der Zeilen 14 – 24 daher mit $O(|V|)$ abgeschätzt werden. Wird während der Heuristik allerdings stets die Wahl für *gewählte_farbe* akzeptiert, dann beträgt die Laufzeit $O(|V| + |\overline{E}|)$.
Die zusätzliche Laufzeit, die für die Bestimmung des Symmetriegraphen und der Erweiterung $g$ notwendig wird, hängt natürlich auch von der Darstellungsform ab, die für die Booleschen Funktionen gewählt wurde.

Falls in Zeile 16 die Funktion $g$ multiformsymmetrisch in $(x_i, x_j)$ ist, bestehen nach Satz 2.3 in Zeile (*) grundsätzlich zwei Möglichkeiten für die Bestimmung der entsprechenden Erweiterung[†]. Man geht dabei folgendermaßen vor:

**Bemerkung 13**

- *Liegt eine starke Symmetrie (Vertauschungssymmetrie oder Äquivalenzsymmetrie) vor, so wird diese genutzt, d.h. eine Neuberechnung von g in Zeile (*) ist nicht notwendig.*

- *Ansonsten wird die Vertauschungssymmetrie bevorzugt.*

#### 4.1.1.1 Behandlung der Multiformsymmetrie

Im bisherigen Algorithmus wurden keine multiformsymmetrischen Mengen bzw. Partitionen betrachtet. Es soll nun die Multiformsymmetrie miteinbezogen werden. Der Algorithmus wird dazu wie folgt abgeändert: Man testet in Zeile 16 zusätzlich, ob $g$ auch multiformsymmetrisch in $(x_i, x_j)$ ist. Ist dies der Fall und bildet $[x_j]$ eine starke multiformsymmetrische Menge für $g$, dann wird in Zeile (*) eine symmetrische Erweiterung $g$ bestimmt, die sowohl stark symmetrisch in $\Pi$ als auch stark multiformsymmetrisch in $[x_j] \cup \{x_i\}$ ist.
Aus Satz 2.6 geht hervor, daß die Bildung *einer* notwendigen symmetrischen Erweiterung bezüglich $v_{ij}$ ausreicht[‡].
Der Test, ob $g$ auch stark multiformsymmetrisch in $[x_j]$ ist, kann durch ein

[†]Solange keine multiformsymmetrische Partitionen betrachtet werden, darf nur genau *eine* Symmetrieform (Vertauschungssymmetrie, Äquivalenzsymmetrie) ausgewählt werden, um die Korrektheit des Verfahrens zu gewährleisten, d.h. es darf nicht mittels $g_{v_{ij}}$ eine stark $(x_i, x_j)$-multiformsymmetrische Erweiterung $g$ bestimmt werden.

[‡]Aus Satz 2.6 geht auch hervor, daß im Fall $|[x_j]| \geq 2$ ein Test auf Symmetrie in $(x_i, x_j)$ für $g$ ausreicht, um festzustellen, ob $g$ multiformsymmetrisch in $[x_j] \cup \{x_i\}$ ist.

zusätzliches Flag, welches für jede (starke) Symmetriemenge von $g$ angibt, ob es sich um eine multiformsymmetrische Menge handelt, in konstanter Zeit erfolgen.

#### 4.1.1.2 Alternativen zur Auswahl der Farbe

Die symmetrische Erweiterung $g$ in Zeile (*) der Heuristik wird durch die Auswahl der Farbe für $x_i$ bestimmt. Im bisher vorgestellten Algorithmus hängt die Vergabe der Farbe lediglich vom Aufbau des Symmetriegraphen $\mathcal{G}^f_{sym}$ (bzw. $\overline{\mathcal{G}^f_{sym}}$) und der aktuellen Färbung ab. Die besonderen Eigenschaften der entsprechenden symmetrischen Erweiterung $g$, welche sich durch die Farbwahl ergeben, haben dagegen keinen Einfluß auf die Auswahl.
Aus diesem Grund sind zwei Alternativen für die Wahl der Farbe implementiert worden:

**Auswahl gemäß der don't cares**
Bei dieser Heuristik wird für den zu färbenden Knoten $x_i$ zunächst die Menge der aufgrund der Symmetrie möglichen Farben für $x_i$ ermittelt und zu jeder Farbe die Größe der don't care-Menge für die symmetrische Erweiterung $g$ bestimmt, die sich durch die entsprechende Färbung von $x_i$ ergibt.
Diejenige Erweiterung, welche die *wenigsten don't care-Werte* spezifiziert, bestimmt die endgültige Farbe für $x_i$, d.h. es wird versucht, eine möglichst große Anzahl von Funktionswerten unspezifiziert zu lassen, um somit im weiteren Verlauf der Färbung möglichst viele Auswahlmöglichkeiten für die Farben zu besitzen.
Die Prozedur *get_color_by_dc* liefert eine Farbe für den Knoten $x_i$ gemäß dieser Strategie:

**get_color_by_dc**($x_i$)

$max\_dc := -1$
$gewählte\_farbe := -1$
$cand(x_i) := \{color(x_j) \mid g$ ist symmetrisch in $(x_i, x_j)$ und $color(x_j) \neq undef.\}$
**<u>foreach</u>** $(farbe \in cand(x_i))$
  **<u>do</u>**
  /* Bestimme Erweiterung $g$, die stark symmetrisch in $\Pi \setminus \{[x_j], \{x_i\}\} \bigcup \{[x_j] \cup \{x_i\}\}$ ist mit $color(x_j) = farbe$ */
  **<u>if</u>** $(\mid DC(g) \mid > max\_dc)$
    **<u>then</u>**
      $gewählte\_farbe = farbe$
      $max\_dc = \mid DC(g) \mid$
  **<u>fi</u>**
**<u>od</u>**
**return**($gewählte\_farbe$ )

**Auswahl gemäß asymmetrischer Paare**
Die entsprechende Prozedur *get_color_by_as* arbeitet analog zu dem obigen Verfahren, allerdings wird hier die Farbe nicht über die don't cares, sondern durch

diejenige symmetrische Erweiterung $g$ bestimmt, welche die *wenigsten asymmetrischen Paare* besitzt. Man will also für $g$ möglichst viele Symmetrien beibehalten, um damit möglichst viele Alternativen für die Auswahl der Farbe zu besitzen.

### 4.1.2 Eine Heuristik für die unabhängigen Variablen

In diesem Abschnitt soll die Behandlung der unabhängigen Variablen einer partiellen Booleschen Funktion näher erörtert werden. Zu einer vorgegebenen partiellen Funktion ist man an der Erweiterung interessiert, die von möglichst wenigen Variablen abhängt, um so kleine Schaltungsentwürfe bzw. kompakte Darstellungen für die partielle Funktion zu erhalten.
Die Prozedur in Abbildung 4.2 erfüllt diese Aufgabe. Die Korrektheit von *compute_min_var* ergibt sich unmittelbar aus Lemma 3.14.
Die Auswahl der Variablen in *choose_var* geschieht nach dem folgenden Prinzip:
Bevor eine neue Variable ausgewählt wird, bestimmt man für alle Variablen $x_i \in \mathit{unabh\ddot{a}ngige\_variablen}$ die Menge $\{x_k \mid g_{v_i}$ ist unabhängig von $x_k\}$. Die Prozedur *choose_var* wählt dann stets die Variable mit der maximalen Menge. Dadurch bleibt die Erweiterung $g$ von möglichst vielen Variablen unabhängig.
Nach Beendigung von *compute_min_var* gibt es offensichtlich keine Variablen mehr, von denen $g$ *nur* schwach unabhängig ist, d.h. alle Unabhängigkeiten in $g$ sind stark.

Man kann die beiden vorgestellten Heuristiken *compute_min_part* und *compute_min_var* auch miteinander kombinieren:
Wird zunächst eine Variablenminimierung mittels *compute_min_var* durchgeführt und danach für das Ergebnis eine minimale symmetrische Partition gesucht, dann ist die entsprechende symmetrische Erweiterung auch stark unabhängig von allen Variablen, die im ersten Schritt „entfernt" wurden.
Umgekehrt kann nach Bildung einer symmetrischen Partition die Zahl der abhängigen Variablen mittels *compute_min_var* minimiert werden. Da alle stark unabhängigen Variablen eine starke multiformsymmetrische Menge bilden, bleibt aufgrund von Satz 3.4 die Anzahl der starken Symmetriemengen für die Funktion gleich oder erhöht sich nur um die Menge der stark unabhängigen Variablen, die aber keinen Einfluß auf die Darstellung bzw. Realisierung der Funktion hat.

### 4.1.3 Funktionen mit mehreren Ausgängen

Bisher wurden ausschließlich partielle Funktionen mit einem Ausgang betrachtet, d.h. Funktionen aus $BP_n$. Oftmals hat man es in der Praxis jedoch mit Funktionen $f \in BP_{n,m}$, $m > 1$ zu tun. Es besteht nun die Möglichkeit, alle Ausgänge getrennt zu betrachten und die entsprechenden symmetrischen Erweiterungen der Teilfunktionen zu einer Erweiterung von $f$ zusammenzufassen. Kleine Schaltungen für Funktionen mit mehreren Ausgängen ergeben sich aber

**Algorithmus *compute_min_var*:**

$X := \{x_1, \ldots, x_n\}$
$stark\_unabhängige\_variablen := \emptyset$
$g := f$
$unabhängige\_variablen := \{x_i \in X \mid g \text{ ist unabhängig von } x_i\}$
**while** $(unabhängige\_variablen \neq \emptyset)$
    **do**
    $x_i = choose\_var(unabhängige\_variablen)$
    $g := g_{v_i}$
    $stark\_unabhängige\_variablen := stark\_unabhängige\_variablen \cup \{x_i\}$
    /* $g$ ist stark unabhängig von *stark_unabhängige_variablen**/
    $unabhängige\_variablen := \{x_i \in X \mid g \text{ ist unabhängig von } x_i\}$
**od**

Abbildung 4.2: Algorithmus zur Behandlung unabhängiger Variablen.

häufig auch dadurch, daß bei der Realisierung für einzelne Teilfunktionen Ergebnisse *mehrfach* genutzt werden können und somit nicht nochmal realisiert werden müssen. Man denke dabei z.B. an die Realisierung eines Booleschen Polynoms mit mehreren Ausgängen. Dort konnten einzelne Monome auf mehrere *or*−Gatter verteilt werden (vgl. Kapitel 1).
Für Funktionen mit mehreren Ausgängen können die bisher vorgestellten Algorithmen verwendet werden. Es muß aber zusätzlich geklärt werden, wann eine Funktion $f \in B_{n,m}$ überhaupt in einem Variablenpaar $(x_i, x_j)$ (schwach) symmetrisch ist. Prinzipiell ergeben sich zwei Möglichkeiten:

- **Konjunktive Verknüpfung**
  In diesem Fall ist die Funktion $f \in BP_{n,m}$ genau dann symmetrisch (multiformsymmetrisch) in $(x_i, x_j)$, falls *alle* Teilfunktionen symmetrisch (multiformsymmetrisch) in $(x_i, x_j)$ sind.
  Es ist hierbei zu beachten, daß aufgrund der Definition der Symmetrie, einzelne Teilfunktionen durchaus verschiedene Symmetrieformen (Vertauschungssymmetrie, Äquivalenzsymmetrie) besitzen können.
  Wendet man *compute_sym_part* auf den so konstruierten Symmetriegraphen an, so erhält man eine *gemeinsame symmetrische Partition* aller Teilfunktionen von $f$. Es existieren aber durchaus verschiedene Phasenabbildungen für einzelne Teilfunktionen.

- **Disjunktive Verknüpfung**
  In diesem Fall ist die Funktion $f \in BP_{n,m}$ genau dann symmetrisch (multiformsymmetrisch) in $(x_i, x_j)$, falls *mindestens eine* der Teilfunktionen symmetrisch (multiformsymmetrisch) in $(x_i, x_j)$ ist.
  Die Heuristik *compute_sym_part* liefert in diesem Fall aber nicht unbedingt eine gemeinsame symmetrische Partition, da nicht immer alle Teilfunktionen von $g$ auch symmetrisch in $\Pi$ sind.

Im folgenden werden stets gemeinsame symmetrische Partitionen betrachtet, d.h. man verwendet die konjunktive Verknüpfung.
Die Prozedur *compute_min_var* kann für Funktionen mit mehreren Ausgängen unterschiedlich ausgeführt werden:
In *compute_min_var_einz* werden alle Teilfunktionen getrennt voneinander bearbeitet, während die Prozedur *compute_min_var_all* nur diejenigen Variablen behandelt, von denen alle Teilfunktionen unabhängig sind.

Hat man für alle Teilfunktionen von $f \in BP_{n,m}$ mittels *compute_min_part* eine *gemeinsame* symmetrische Partition $\Pi$ bzw. eine symmetrische Erweiterung $g \in BP_{n,m}$ bestimmt, so ist es oftmals möglich, für einige Teilfunktionen von $g$ kleinere symmetrische Partitionen zu generieren, indem (starke) Symmetriemengen aus $\Pi$ für diese Funktionen zu einer neuen Symmetriemenge vereinigt werden können. Dadurch lassen sich die entsprechenden Teilfunktionen oftmals noch effizienter realisieren bzw. darstellen. Beim Zusammenschluß solcher Symmetriemengen muß allerdings folgendes beachtet werden:

- Die bisherigen Ergebnisse in den vorangegangenen Kapiteln reichen nicht aus, um ein hinreichendes Kriterium dafür anzugeben, in welchem Fall zwei *beliebig große* starke Symmetriemengen zu einer Symmetriemenge vereinigt werden können.
  Allerdings besagt Satz 2.6 für den Fall, daß es sich um zwei stark multiformsymmetrische Mengen handelt, daß eine Multiformsymmetrie in einem Paar $(x_i, x_j)$ zwischen den beiden Mengen für eine Vereinigung dieser Mengen ausreicht.

- Besonders im Hinblick auf die Darstellung mittels ROBDDs ist zu beachten, daß die Teilfunktionen nicht völlig unabhängig voneinander bearbeitet werden dürfen. Liegt für eine Boolesche Funktion $f \in BP_{n,m}$ z.B. eine (gemeinsame) symmetrische Partition $\Pi = \{\lambda_1, \lambda_2, \lambda_3\}$ vor, so ist es nicht ratsam, für einige Teilfunktionen $\lambda_1, \lambda_2$ und für andere Teilfunktionen $\lambda_1, \lambda_3$ zusammenzuschließen, da man für alle ROBDDs der Teilfunktionen eine *gemeinsame* symmetrische Variablenordnung bzgl. $\Pi$ sucht.

In Abbildung 4.3 ist die entsprechende Prozedur *join* dargestellt.

Die Funktion *choose_sets* liefert ein Paar von starken Symmetriemengen $(\lambda_p, \lambda_q)$, welches $\mid join\_fkt_{p,q} \mid$ maximiert, um so möglichst viele Teilfunktionen gemäß $\Pi$ symmetrisch erweitern zu können.
Wie bereits erwähnt, lassen sich die Mengen $join\_fkt_{p,q}$ a priori mit den bisherigen Mitteln nicht unmittelbar bestimmen. Anstelle dieser Mengen verwendet man daher Obermengen $join\_fkt_{p,q} \subseteq join\_fkt\_approx_{p,q}$, die sich leicht bestimmen lassen[†]. Eine Möglichkeit für $join\_fkt\_approx_{p,q}$ ergibt sich aus dem folgenden Korollar:

[†]Die Auswahl der Mengen in *choose_sets* erfolgt in diesem Fall ebenfalls über $\mid join\_fkt\_approx_{p,q} \mid$.

**Algorithmus *join*:**

/* Gegeben: Stark symmetrische Partition $\Pi$ für $f \in BP_{n,m}$ */<br>
$join\_fkt_{p,q} := \{f_i \mid f_i$ ist symmetrisch in $\Pi \backslash \{\lambda_p, \lambda_q\} \bigcup \{\lambda_p \cup \lambda_q\}\}$<br>
**while** $(\exists \lambda_p, \lambda_q \in \Pi$, so daß $join\_fkt_{p,q} \neq \emptyset)$ **do**<br>
    $(\lambda_p, \lambda_q) := choose\_sets(\Pi)$<br>
    $\Pi := \Pi \backslash \{\lambda_p, \lambda_q\} \bigcup \{\lambda_p \cup \lambda_q\}$<br>
    **foreach** $(f_i \in join\_fkt_{p,q})$ **do**<br>
      /* Mache $f_i$ stark symmetrisch in $\Pi$ */ (*)<br>
    **od**<br>
    $join\_fkt_{p,q} := \{f_i \mid f_i$ ist symmetrisch in $\Pi \backslash \{\lambda_p, \lambda_q\} \bigcup \{\lambda_p \cup \lambda_q\}\}$<br>
**od**

Abbildung 4.3: Algorithmus *join* für Funktionen mit mehreren Ausgängen.

**Korollar 4.1** *Ist $f \in BP_n$ eine partielle Boolesche Funktion und sind $\lambda_p, \lambda_q$ starke Symmetriemengen für $f$. $f$ ist nicht symmetrisch in $\lambda_p \cup \lambda_q$, falls ein asymmetrisches Paar $(x_i, x_j) \in \lambda_p \times \lambda_q$ für $f$ existiert.*

Asymmetrische Paare lassen sich z.B. mit Hilfe des Verfahrens aus [MMW93] sehr schnell bestimmen. Die Menge $join\_fkt\_approx_{p,q}$ besteht somit aus allen Teilfunktionen $f_i$, für die kein asymmetrisches Paar aus $\lambda_p \times \lambda_q$ gefunden wurde.

Ein weitere Schwierigkeit in *join* ist die Bestimmung einer Erweiterung von $f_i$, welche stark symmetrisch in der neuen Partition $\Pi$ ist (Zeile (*) der Prozedur *join*). Unter Verwendung der bisherigen Ergebnisse kann man dabei wie folgt verfahren:
Die Vereinigung der Mengen $\lambda_p$ und $\lambda_q$ wird dadurch gebildet, daß die Symmetriemenge $\lambda_p$ nacheinander um jeweils eine zusätzliche Variable aus $\lambda_q$ erweitert wird. Bei jedem Anschluß einer Variablen aus $\lambda_q$ läßt sich die entsprechende symmetrische Erweiterung mit Hilfe der Symmetrierungsfolgen aus dem vorhergehenden Kapitel leicht berechnen.
Ist $f_i$ symmetrisch in $\Pi \backslash \{\lambda_p, \lambda_q\} \bigcup \{\lambda_p \cup \lambda_q\}$, so besagt Lemma 3.6, daß nach jedem Anschluß einer Variablen aus der Menge $\lambda_q$ an $\lambda_p$ auch für alle restlichen Variablen aus $\lambda_q$ eine Symmetrierungsfolge existiert, so daß das Verfahren nicht zwischendurch abbricht.
Verwendet man anstelle der Mengen $join\_fkt_{p,q}$ lediglich $join\_fkt\_approx_{p,q}$, so kann nicht immer garantiert werden, daß alle Funktionen aus $join\_fkt\_approx_{p,q}$ auch symmetrisch in der Menge $\Pi \backslash \{\lambda_p, \lambda_q\} \bigcup \{\lambda_p \cup \lambda_q\}$ sind. Daher bricht das Verfahren der sukzessiven Erweiterung von $\lambda_p$ irgendwann für eine Variable $x_j \in \lambda_q$ ab, da keine Symmetrien mehr zwischen $x_j$ und den Variablen aus $\lambda_p$ existieren.

Handelt es sich bei den beiden Symmetriemengen $\lambda_p$ und $\lambda_q$ um starke *multiformsymmetrische* Mengen, dann liefert offenbar Satz 2.6 eine hinreichende und notwendige Bedingung für die Vereinigung der beiden Mengen zu einer starken multiformsymmetrischen Menge. Die entsprechende symmetrische Erweiterung wird in diesem Fall mittels Satz 3.3 gebildet.

## 4.2 Anwendung auf die Minimierung von ROBDDs partieller Boolescher Funktionen

In Abschnitt 2.3 wurde schon gezeigt, daß die Verwendung von symmetrischen Variablenordnungen oftmals sehr gute Ergebnisse im Hinblick auf die Größe des resultierenden ROBDDs liefern kann. In vielen Anwendungen, bei denen Boolesche Funktionen intern durch ROBDDs dargestellt werden, hängt die Laufzeit und der Speicherverbrauch unmittelbar mit der Größe der Darstellung zusammen. Treten in diesen Anwendungen partielle Funktionen auf, so ist es sehr wichtig, entsprechende Erweiterungen mit möglichst kleiner Darstellung zu generieren, um so die Verfahren zu beschleunigen. In [Sch96] wird ein Verfahren zur mehrstufigen Logiksynthese vorgestellt, welches ebenfalls mit ROBDDs arbeitet. Dort ist es für die Güte der Resultate sehr wichtig, für partielle Funktionen, die als Zwischenergebnisse auch beim Entwurf totaler Funktionen auftreten können, Erweiterungen dieser Funktionen mit möglichst geringen ROBDD-Darstellungen zu bestimmen.

Um partielle Boolesche Funktionen $f \in BP_n$ mittels ROBDDs eindeutig darzustellen, kann man zwei der (totalen) Funktionen $f_{on}, f_{off}, f_{dc}$ durch die entsprechenden ROBDDs repräsentieren.
Es ist allerdings auch möglich, partielle Funktionen vollständig durch *einen* ROBDD darzustellen. In [CM93] wird dazu die Variablenmenge $X$ um eine zusätzliche Variable $Z \notin X$ erweitert und ein ROBDD für die Funktion

$$f_{ext} = Z \wedge f_{on} \vee \overline{Z} \wedge f_{off}$$

bestimmt.
Für partielle Funktionen mit mehreren Ausgängen kann diese Konstruktion auf alle Teilfunktionen angewandt werden.
Ein solcher ROBDD kann häufig in effizienter Weise heuristisch minimiert werden.
Das folgende Verfahren aus [CCM94] minimiert „schichtweise“ die Anzahl der Knoten eines ROBDDs $F$, falls $F$ die Funktion $f_{ext}$ repräsentiert:
Ist $Z$ die $i$. Variable in der Variablenordnung $<_\pi$, d.h. $\pi[i] = Z$, so wird die Anzahl der Knoten in $F$, die mit $Z$ beschriftet sind (also $ind_F(i)$) dadurch vermindert, indem die Unter-ROBDDs zu den Knoten aus $ind_F(i)$, die selbst wieder partielle Funktionen repräsentieren, durch ROBDDs zu geeigneten (partiellen) Erweiterungen ersetzt werden. Der so entstandene neue ROBDD repräsentiert somit auch eine Erweiterung von $f$.
Um die Anzahl der Knoten mit der Beschriftung $Z$ zu minimieren, wird gezielt nach *gemeinsamen* Erweiterungen für möglichst viele dieser Knoten gesucht.
Da der ROBDD $F$ insgesamt minimiert werden soll, wird in [CCM94] die zusätzliche Variable $Z$ nacheinander an die Positionen $\pi[1], \pi[2], \ldots, \pi[n]$ in der Va-

riablenordnung eingefügt und dabei für jede Position von $Z$ die entsprechenden Knoten mit der Beschriftung $Z$ gemäß dem obigen Verfahren ersetzt.
Als Ergebnis erhält man somit eine ROBDD-Darstellung einer Erweiterung der Ausgangsfunktion $f$ mit geringer Größe.

Ist die Ausgangsfunktion $f$ stark symmetrisch in der Partition $\Pi$, so kann das Verfahren so modifiziert werden, daß die resultierende Erweiterung stark symmetrisch in $\Pi$ bleibt, indem $Z$ nicht alle Positionen in der Variablenmenge durchläuft, sondern stets zwischen zwei starken Symmetriemengen aus $\Pi$ eingeordnet wird [Sch96]:

**Satz 4.2** (**Scholl '96**) *Sei $f \in BP_n$ eine partielle Boolesche Funktion und $\Pi = \{\lambda_1, \ldots, \lambda_k\}$ eine stark symmetrische Partition für $f$. Ist $<_\pi$ eine symmetrische Variablenordnung mit der Eigenschaft, daß alle Variablen aus $\lambda_i$ kleiner sind als die Variablen aus $\lambda_j$, falls $i < j$. Dann liefert das Verfahren aus [CCM94] eine Erweiterung von $f$, die ebenfalls stark symmetrisch in $\Pi$ ist, falls für die Ordnung der zusätzlichen Variable $Z$ stets gilt: $max_i <_\pi Z <_\pi min_{i+1}$, wobei $max_i$ ($min_i$) die größte (kleinste) Variable (bzgl. $<_\pi$) in $\lambda_i$ bezeichnet.*

Um zu einer partiellen Booleschen Funktion $f$ eine Erweiterung mit möglichst kleiner ROBDD-Darstellung zu bestimmen, bietet sich daher folgendes Verfahren an:

- Zunächst bestimmt man für $f$ eine symmetrische Partition $\Pi$ mit möglichst wenigen Symmetriemengen bzw. eine entsprechende Erweiterung $g$, die symmetrisch in $\Pi$ ist.

- Anschließend wird ein ROBDD $G$ für $g_{ext}$ aufgestellt und eine symmetrische Variablenordnung $<_\pi$ gemäß der symmetrischen Partition $\Pi$ erzeugt.

- Nun wird die Größe des ROBDDs $G$ mit Hilfe des Verfahrens aus [CCM94] und unter Beachtung von Satz 4.2 weiter reduziert. Die resultierende Erweiterung bleibt daher stark symmetrisch in $\Pi$.

- Zuletzt optimiert man die symmetrische Variablenordnung $<_\pi$ (z.B. mittels *sym_sift* [PSP94, MMD94]).

## 4.3 Ergebnisse

Die in diesem Kapitel vorgestellten Algorithmen wurden im Rahmen dieser Diplomarbeit durch entsprechende *C++* Programme implementiert. Die Güte der Algorithmen wurde anhand der Ergebnisse bei der Minimierung von ROBDD-Darstellungen für bestimmte Benchmarkschaltungen ermittelt.

Das erste Beispiel betrachtet partielle Multiplizierer. Bei diesen Funktionen wird die Multiplikation zweier $n-$stelliger binärer Zahlen $(a_{n-1}, \ldots, a_0)$ und $(b_{n-1}, \ldots, b_0)$ berechnet, indem die $n^2$ Partialprodukte $p_{ij} = a_i b_j$ $(0 \leq i, j \leq n-1)$ aufsummiert werden. Es gilt:

$partmult_n: \{0,1\}^{n^2} \rightarrow \{0,1\}^{2n}$,
$partmult_n(p_{n-1,n-1}, \ldots, p_{0,n-1}, \ldots, p_{n-1,0}, p_{0,0}) = (r_{2n-1}, \ldots, r_0)$, wobei

$$\sum_{i=0}^{2n-1} r_i \cdot 2^i = \sum_{j=0}^{n-1} \left[ \left( \sum_{i=0}^{n-1} p_{ij} \cdot 2^i \right) \cdot 2^j \right].$$

Das folgende Beispiel zeigt dies exemplarisch für $n = 3$:

$$\begin{array}{ccccccc} & & & & p_{2,0} & p_{1,0} & p_{0,0} \\ + & & & p_{2,1} & p_{1,1} & p_{0,1} & \\ + & & p_{2,2} & p_{1,2} & p_{0,2} & & \\ \hline & r_5 & r_4 & r_3 & r_2 & r_1 & r_0 \end{array}$$

Da die einzelnen Eingaben $p_{ij}$ somit nicht unabhängig voneinander sind, können bestimmte Eingabevektoren schon von vornherein ausgeschlossen werden. Diese bilden die don't care-Menge der partiellen Funktion $partmult_n$.
Die Symmetrien dieser Funktionen ergeben sich u.a. durch die Möglichkeit, alle Partialprodukte, welche *ein gemeinsames* Ergebnisbit bestimmen, zu vertauschen, ohne das Resultat der Addition zu verändern. Im obigen Beispiel sind dies die Mengen $\{p_{2,2}\}, \{p_{2,1}, p_{1,2}\}, \{p_{2,0}, p_{1,1}, p_{0,2}\}, \{p_{1,0}, p_{0,1}\}, \{p_{0,0}\}$.

Die Ergebnisse für verschiedene Multiplizierer zeigt Tabelle 4.4.
Dabei wurden 3 Heuristiken verglichen:

Bei allen Heuristiken wurde zunächst eine symmetrische Partition $\Pi$ ermittelt und eine entsprechende stark symmetrische Erweiterung bestimmt. Anschließend wurden die Variablen gemäß der Partition $\Pi$ auf Blöcke verteilt und mittels *sym_sift* [MMD94, PSP94] eine symmetrische Variablenordnung ermittelt.

Bei $sym\_group_{orig}$ wurden keine don't cares belegt, d.h. die symmetrische Partition $\Pi$ resultiert alleine aus den starken Symmetrien der partiellen Ausgangsfunktion.

Die symmetrische Partition in $sym\_group_{Bre}$ ergibt sich durch die Aufruffolge *compute_min_var_all* $\rightarrow$ *compute_min_part* $\rightarrow$ *compute_min_var_einz,* d.h. vor Bildung der (gemeinsamen) symmetrischen Erweiterung werden zunächst solche Variablen aus dem ROBDD entfernt, von denen alle Teilfunktionen unabhängig sind. Nachdem man eine symmetrische Partition $\Pi$ bzw. eine entsprechende symmetrische Erweiterung bestimmt hat, wird die Anzahl der unabhängigen Variablen für alle Teilfunktionen getrennt minimiert. Die Teilfunktionen bleiben dabei stets stark symmetrisch in $\Pi$.

Bei *sym_cover* wurde der resultierende ROBDD aus $sym\_group_{Bre}$ mit Hilfe der Heuristik aus [CCM94] unter Beibehaltung der starken Symmetrien weiter minimiert, d.h. es wurde die am Ende von Abschnitt 4.2 vorgestellte Verfahren komplett angewendet.

Für jede Heuristik gibt die Spalte *size* jeweils die Größe des ROBDDs der resultierenden Erweiterung an und Spalte *sets* die Anzahl der Elemente (Symmetriemengen) der symmetrischen Partition $\Pi$.
In den Klammern ist jeweils die benötigte CPU Zeit (in sec.) angegeben (gemessen auf einer SPARCstation 20).

| circ | i | o | $sym\_group_{orig}$ | | $sym\_group_{Bre}$ | | | $sym\_cover$ | |
|---|---|---|---|---|---|---|---|---|---|
| | | | size | sets | size | | sets | size | |
| $partmult_3$ | 9 | 6 | 152 | 9 | 35 | (1.2) | 5 | 29 | (1.5) |
| $partmult_4$ | 16 | 8 | 971 | 16 | 222 | (55.4) | 7 | 114 | (57.3) |
| $partmult_5$ | 25 | 10 | 4574 | 25 | 998 | (2072.6) | 9 | 365 | (2094.3) |

Abbildung 4.4: Ergebnisse für partielle Multiplizierer.

| circ | i | o | $sym\_group_{orig}$ | | $sym\_group_{Bre}$ | | | $sym\_cover$ | | $sym\_cover_{join}$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | size | sets | size | | sets | size | | size | |
| 5xp1 | 7 | 10 | 67 | 7 | 66 | (0.1) | 7 | 53 | (0.5) | 56 | (0.9) |
| mux | 21 | 1 | 34 | 14 | 29 | (0.5) | 12 | 29 | (0.8) | 29 | (1.0) |
| b9 | 41 | 21 | 254 | 38 | 185 | (9.7) | 35 | 122 | (13.8) | 137 | (59.6) |
| c8 | 28 | 18 | 156 | 28 | 95 | (1.3) | 23 | 70 | (2.7) | 67 | (11.8) |
| pcler8 | 27 | 17 | 78 | 22 | 73 | (1.4) | 22 | 72 | (2.7) | 70 | (12.4) |
| sao2 | 10 | 4 | 104 | 10 | 104 | (0.2) | 10 | 70 | (0.8) | 69 | (1.3) |
| z4ml | 7 | 4 | 51 | 7 | 32 | (0.1) | 4 | 17 | (0.3) | 17 | (0.3) |
| 9symml | 9 | 1 | 107 | 9 | 25 | (0.4) | 1 | 25 | (0.5) | - | - |
| rd73 | 7 | 3 | 76 | 7 | 34 | (0.3) | 1 | 27 | (0.5) | - | - |
| example2 | 85 | 66 | 491 | 72 | 484 | (61.7) | 72 | 416 | (112.9) | 447 | (662.0) |
| apex7 | 49 | 37 | 805 | 47 | 767 | (30.6) | 45 | 343 | (86.0) | 327 | (276.0) |
| apex6 | 135 | 99 | 1078 | 129 | 1059 | (319.3) | 128 | 617 | (606.5) | 639 | (3445.1) |
| x4 | 94 | 71 | 829 | 90 | 633 | (146.1) | 89 | 485 | (245.0) | 499 | (1414.6) |
| rd84 | 8 | 4 | 140 | 8 | 42 | (0.7) | 1 | 42 | (0.9) | - | - |
| alu2 | 10 | 6 | 192 | 10 | 192 | (0.4) | 10 | 156 | (2.9) | 152 | (5.0) |
| Summe | | | 4462 | | 3810 | | | 2544 | | | |

Abbildung 4.5: Benchmarkschaltkreise mit 40% don't cares.

Wie die Ergebnisse zeigen, lassen sich durch die vorgestellten Verfahren erhebliche Reduzierungen der ROBDD-Größe vornehmen. Hervorzuheben ist auch, daß $compute\_group_{Bre}$ offenbar gerade die symmetrische Partition bestimmt, die man aufgrund der Struktur der partiellen Multiplizierer auch erwartet.
In [Sch96] findet der Leser einen Entwurf des partiellen Multiplizierers für $n = 4$, bei dessen Realisierung u.a. die in dieser Arbeit vorgestellten Algorithmen benutzt wurden.

Tabelle 4.5 zeigt die entsprechenden Ergebnisse für weitere Benchmarkschaltungen. Um zu partiellen Funktionen zu gelangen, wurden die (totalen) Benchmarkschaltungen in eine zweistufige Darstellung (Boolesche Polynome) gebracht und mit einer Wahrscheinlichkeit von 40% die $ON-$Mengen der einzelnen Monome in die don't care-Menge übernommen.

Wie diese Ergebnisse zeigen, erhält man für die Benchmarkschaltungen eine durchschnittliche Reduzierung um 15% bei $sym\_group_{Bre}$ und über 40% bei $sym\_cover$ im Vergleich zu $sym\_group_{orig}$[†]. Gerade für totalsymmetrische Funktionen bzw. solche Funktionen mit wenigen Symmetriemengen ergeben sich erhebliche Verringerungen der ROBDD-Größe.

Tabelle 4.5 ist um eine zusätzliche Spalte $sym\_cover_{join}$ erweitert. Diese Heuristik geht aus $sym\_cover$ hervor, indem zusätzlich *unmittelbar* nach Bildung der

[†]Zählt man die Ergebnisse aus Tabelle 4.4 hinzu, dann ergeben sich sogar durchschnittliche Verringerungen von über 50% bzw. 70%.

| circ | i | o | *sym_group_Bre*$_{orig}$ | | | *sym_group_Bre*$_{dc}$ | | | *sym_group_Bre*$_{as}$ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | size | | sets | size | | sets | size | | sets |
| 5xp1 | 7 | 10 | 35 | (0.2) | 7 | 35 | (0.1) | 7 | 35 | (0.1) | 7 |
| mux | 21 | 1 | 12 | (0.9) | 4 | 11 | (0.9) | 4 | 9 | (3.3) | 4 |
| b9 | 41 | 21 | 64 | (17.0) | 15 | 60 | (18.1) | 15 | 57 | (293.4) | 15 |
| c8 | 28 | 18 | 22 | (4.1) | 8 | 21 | (5.4) | 8 | 21 | (76.4) | 8 |
| pcler8 | 27 | 17 | 29 | (4.4) | 9 | 32 | (5.2) | 8 | 38 | (81.1) | 9 |
| sao2 | 10 | 4 | 57 | (0.4) | 9 | 54 | (0.5) | 9 | 57 | (1.0) | 9 |
| z4ml | 7 | 4 | 27 | (0.2) | 3 | 27 | (0.1) | 3 | 27 | (0.2) | 3 |
| 9symml | 9 | 1 | 21 | (0.3) | 1 | 21 | (0.4) | 1 | 21 | (0.7) | 1 |
| rd73 | 7 | 3 | 33 | (0.2) | 1 | 33 | (0.2) | 1 | 33 | (0.4) | 1 |
| example2 | 85 | 66 | 277 | (135.8) | 39 | 276 | (162.6) | 39 | 270 | (6744.6) | 39 |
| apex7 | 49 | 37 | 369 | (33.9) | 32 | 358 | (35.7) | 32 | 358 | (706.6) | 32 |
| apex6 | 135 | 99 | 552 | (545.1) | 78 | 552 | (853.2) | 78 | 552 | (38031.8) | 78 |
| x4 | 94 | 71 | 320 | (233.6) | 59 | 319 | (284.3) | 59 | 319 | (8750.3) | 59 |
| rd84 | 8 | 4 | 50 | (0.5) | 1 | 50 | (0.8) | 1 | 50 | (1.4) | 1 |
| alu2 | 10 | 6 | 91 | (0.1) | 10 | 91 | (0.1) | 10 | 91 | (0.1) | 10 |
| Summe | | | 1959 | | | 1940 | | | 1938 | | |

Abbildung 4.6: Benchmarkschaltkreise mit 80% don't cares.

symmetrischen Partition die Prozedur *join* aufgerufen wird‡.

Die Laufzeit von *sym_cover*$_{join}$ ist insbesondere für Funktionen mit vielen Eingangsvariablen und großen symmetrischen Partitionen sehr hoch, da dort beim Zusammenschluß zu größeren Symmetriemengen sehr viele Symmetrierungsfolgen gebildet werden müssen.

Um die verschiedenen Alternativen bei der Farbauswahl in *compute_min_part* (s. Abschnitt 4.1.1.2) in ihrer Güte miteinander zu vergleichen, wurde die Wahrscheinlichkeit mit der die $ON-$Mengen der einzelnen Monome in die don't care-Menge übernommen werden auf 80% erhöht und bei der Bestimmung der symmetrischen Erweiterung in *sym_group*$_{Bre}$ *keine* zusätzliche Minimierung der unabhängigen Variablen vorgenommen, d.h. die symmetrische Partition hinsichtlich der die symmetrische Variablenordnung gebildet wird ergibt sich lediglich aus *compute_min_part*. Die entsprechenden Ergebnisse sind in der Abbildung 4.6 dargestellt.

Die Spalte *sym_group_Bre*$_{orig}$ gibt dabei die Größe und Anzahl der Symmetriemengen für die Erweiterung an, die sich ergibt, falls *compute_min_part* wie in Abbildung 4.1 ausgeführt wird.

Die Spalten *sym_group_Bre*$_{dc}$ und *sym_group_Bre*$_{as}$ zeigen die entsprechenden Ergebnisse, welche sich unter Verwendung der beiden Prozeduren *get_color_by_dc* bzw. *get_color_by_as* aus Abschnitt 4.1.1.2 ergeben.

Man erkennt deutlich, daß für diese Benchmarkschaltungen die Auswahl der Farbe gemäß den erzeugten Asymmetrien am besten abschneidet, jedoch insbesondere für Schaltungen mit vielen Eingangsvariablen sehr große Laufzeiten besitzt. Aus diesem Grund scheint *sym_group_Bre*$_{dc}$ die günstigere Alternative zu sein, da sie bessere Resultate als *sym_group_Bre*$_{orig}$ liefert und in etwa die gleiche Laufzeit besitzt.

‡Für totalsymmetrische Erweiterungen wird kein Aufruf von *join* getätigt. Daher sind die entsprechenden Einträge in der Tabelle leer.

## 4.4 Zusammenfassung

In der vorliegenden Arbeit wurde eine Heuristik zur Bestimmung von symmetrischen Erweiterungen partieller Boolescher Funktionen entwickelt. Dabei wurden für die Spezifizierung der don't care-Menge die Symmetrien der Funktion derart ausgenutzt, daß die resultierende Erweiterung in einer möglichst kleinen Partition der Eingangsvariablen stark symmetrisch ist.
Experimentelle Untersuchungen an ROBDD-Darstellungen für Benchmarkschaltungen haben gezeigt, daß sich mit Hilfe der vorgestellten Algorithmen die Größe der Darstellung sehr stark reduzieren läßt.
Für viele ROBDD-basierte Anwendungen (z.B. [CMB89, MNS+90, Sch96]) kann dadurch die Laufzeit, der Speicherverbrauch und die Güte der Schaltungsentwürfe erheblich verbessert werden.

# Literaturverzeichnis

[Aks78] S. B. Akers: Binary Decision Diagrams. IEEE Trans. Comput., vol C-27, pp. 509-516, 1978.

[BCM+90] J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill: Sequential Circuit Verification Using Symbolic Model Checking. Proc. of 27th ACM/IEEE Design Automation Conference, pp. 46-51, 1990.

[Beck92] B. Becker: Synthesis for Testability: Binary Decision Diagrams. Proc. of 9th Annual Symposium on Theoretical Aspects of Computer Science, Lecture Notes in Computer Science vol. 577, Springer-Verlag, pp. 501-512, 1992.

[BHM+84] R.K. Brayton, G.D. Hachtel, C.T. McMullen and A. Sangiovanni-Vicentelli: 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 Transactions on Computers, C-35(8), pp. 677-691, 1986.

[Bry91] R.E. Bryant: On the Complexity of VLSI Implementations and Graph Representations of Boolean Functions with Applications to Integer Multiplication, IEEE Trans. Comput. 40, 2, pp. 205-213, 1991.

[BRB90] K.S. Brace, R.L. Rudell, R.E. Bryant: Efficient Implementation of a BDD Package, Proc. of the 27th IEEE/ACM Design Automat. Conf., pp. 40-45, 1990.

[Bre79] D. Brélaz: New Methods to Color Vertices of a Graph. Communications of the ACM, 22, pp. 251-256, 1979.

[BSW94] B. Bollig, P. Saviky, I. Wegener: On the Improvement of Variable Orderings for OBDDs. IFIP Workshop on Logic and Architecture Synthesis, Grenoble, pp. 71-80, 1994.

[CMB89] O. Coudert, J.-C. Madre, C. Berthet: Verification of Synchronous Sequential Machines Based on Symbolic Execution. In J. Sifkas, editor, Proceedings of the Workshop on Automatic Verification Methods for Finite State Systems, vol.407 of Lecture Notes in Computer Science, Springer-Verlag, pp. 365-373, 1989.

[CCM94] S.C. Chang, D.I. Cheng, M. Marek-Sadowska: Minimizing ROBDD Size of Incompletely Specified Multiple Output Functions, ED & TC, pp. 620-624, 1994.

[Clus56] E.J. McCluskey: Minimization of Boolean Functions. B.S.T.J., vol. 35, No.6, pp. 1417-1444, 1956.

[CM92] S.C. Chang, M. Marek-Sadowska: Technology Mapping via Transformations of Function Graphs. Proc. of the IEEE International Conference on Computer Design, pp. 159-162,1992.

[CM93] S.C.Chang, M. Marek-Sadowska: BDD Representation of Incompletely Specified Functions, IWLS Workshop Notes, 1993.

[DBG95] R. Drechsler, B. Becker, N. Göckel: A Genetic Algorithm for Variable Ordering of OBDDs. International Workshop on Logic Synthesis, Granlibakken (CA), 1995.

[FMK91] M. Fujita, Y. Matsunaga, T. Kakuda: On the Variable Ordering of Binary Decision Diagrams for the Application of Multi-level Logic Synthesis. European Conf. on Design Automation, pp. 50-54, 1991.

[FS90] St. J. Friedman, K.J. Supowit: Finding the Optimal Variable Ordering for Binary Decision Diagrams. IEEE Trans. on CAD, 39 (5), pp. 710-713, 1990.

[GJ79] M.R. Garey, D.S. Johnson: Computers and Intractability: A Guide to the Theory of *NP*-completeness. New York, Freeman, 1979.

[HM94] M. Heap, M. Mercer: Least Upper Bounds on OBDD sizes. IEEE Trans. on Comp., 43(6), pp. 764-767, 1994.

[Ho74] G.Hotz: Schaltkreistheorie, Walter de Gruyter, 1974.

[Hu77] S.L. Hurst: Detection of Symmetries in Combinatorial Functions by Spectral Means. Electronic Circuits and Sysytems, 1(5), pp. 173-180, 1977.

[KD91] B.-G. Kim, D.L. Dietmeyer: Multilevel Logic Synthesis of Boolean Switching Functions. IEEE Trans. on CAD, vol 10(4), pp. 436-446, 1991.

[Lee59] C.Y Lee: Representation of Switching Circuits by Binary Decision Programs. Bell. Syst. Tech. J., vol. 38, pp. 985-999, 1959.

[LF80] R.E. Ladner, M.J. Fischer: Parallel Prefix Computation. Journal of the ACM 27, pp. 831-838, 1980.

[Lit97] L. Heinrich-Litan: Least upper Bounds on the Size of OBDDs of Partially Symmetric Boolean Functions. Diplomarbeit Univ. Halle, 1997.

[Lup58] O.B. Lupanov: A Method of Circuit Synthesis. Izv. VUZ Radioviz 1, pp. 120-140, 1958.

[Mehl86] K. Mehlhorn: Datenstrukturen und effiziente Algorithmen, Bd.1 Sortieren und Suchen, B.G. Teubner, 1986.

[MMD94] D. Möller, P. Molitor, R. Drechsler: Symmetry Based Variable Ordering for ROBDDs. IFIP Workshop on Logic and Architecture Synthesis, Grenoble, pp. 47-53, 1994

[MMW93] D. Möller, J.Mohnke, M.Weber: Detection of Symmetry of Boolean Functions represented by ROBDDs. IEEE Int'l Conf. on CAD, pp. 680-684, 1993.

[MNS+90] R. Murgai, Y. Nishizaki, N. Shenoy, R.K. Brayton A. Sangiovanni-Vincentelli: Logic Synthesis for Programmable Gate Arrays, Proc. Design Automat. Conf., pp. 620-625, 1990.

[Morg92] C. Morgenstern: A New Backtracking Heuristic for Rapidly Four-Coloring Large Planar Graphs. Tech. Report CoSc-1992-2, Texas Christian University, Fort Worth, 1992.

[MWBS88] S. Malik, A.R. Wang, R.K. Brayton, A. Sangiovanni-Vincentelli: Logic Verification using Binary Decision Diagrams in a Logic Synthesis Environment. Proc. of the IEEE Int'l Conf. on CAD, pp. 6-9, 1988.

[PS95] S. Panda, F. Somezi: Who are the Variables in Your Neighborhood. IEEE Int'l Conf. on CAD, pp. 74-77, 1995.

[PSP94] S. Panda, F. Somenzi, B. Plessier: Symmetry Detection and Dynamic Variable Ordering of Decision Diagrams. IEEE Int'l Conf. on CAD, pp. 628-631,1994.

[Quine53] W.V. Quine: Two Theorems about Truth Functions. Soc. Math. Mex. 10, pp. 64-70, 1953.

[Quine55] W.V. Quine: A Way to Simplify Truth Functions. American Mathematical Monthly, vol. LXII, pp. 627-631, 1955.

[Rud93] R. Rudell: Dynamic Variable Ordering for Ordered Binary Decision Diagrams. IEEE Int'l Conf. on CAD, pp. 42-47, 1993.

[Shan49] C.E. Shannon: The Synthesis of Two-Terminal Switching Circuits. Bell Systems Technical Journal 28, pp. 59-98, 1949.

[Sch96] C. Scholl: Mehrstufige Logiksynthese unter Ausnutzung funktionaler Eigenschaften. Dissertation Univ. Saarbrücken, 1996.

[Wall64] C.S. Wallace: A Suggestion for a fast Multiplier. IEEE Trans. on Comp. 13, pp. 14-17, 1964.

[Weg87] I. Wegener: The Complexity of Boolean Functions. Wiley-Teubner, 1987.

[Weg89] I. Wegener: Effiziente Algorithmen für grundlegende Funktionen. B.G. Teubner, Stuttgart, 1989.