Diplomarbeit

Heuristische Optimierungen an Spezifikationen von PLA-Tabellen

Andreas Krause

29. September 1995

Universität Hamburg — Fachbereich Informatik

Arbeitsbereich Technische Grundlagen der Informatik

Universität Hamburg — Fachbereich Informatik
Arbeitsbereich Technische Grundlagen der Informatik

Diplomarbeit

Heuristische Optimierungen an Spezifikationen von PLA-Tabellen

Diplomand	Andreas Krause Brehmweg 29, 22527 Hamburg
Matrikelnummer	4 011 204
Beginn	30. September 1994
Abgabe	29. September 1995
Betreuung	Dr. Reinhard Rauscher Technische Grundlagen der Informatik
	Prof. Dr. Rainer Lang Anwendungen der Informatik in Naturwissenschaft und Technik

Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbständig und ohne fremde Hilfe angefertigt zu haben. Es wurde lediglich die angegebene Literatur verwendet.

Hamburg, den 29. September 1995

Zusammenfassung

Beim Chip-Entwurf werden Tabellen häufig als PLA realisiert. Äußerst nachteilig macht sich bei PLAs in (Pseudo-)NMOS-Technologie die hohe statische Stromaufnahme bemerkbar. Oft muß ersatzweise auf eine ROM-Realisierung zurückgegriffen werden.

Allerdings zeigt die Entwurfspraxis, daß Spezifikationen von Tabellen inhärente Unschärfe bzw. verdeckte Optionen aufweisen. Beispielsweise

- geschieht die Diskretisierung von Funktionen zu meist *ad hoc*,
- wird die Skalierung von reellen Werten nahezu willkürlich gewählt,
- erfolgt die Rundung von reellen Werten standardmäßig zur nächstliegenden Ganzzahl,
- werden Bit-Breiten nicht vollständig ausgenutzt, usf.

Der Ansatz der Arbeit besteht darin, bereits die Spezifikation der Tabelle zu beeinflussen und dabei alternative Varianten zu erzeugen. Sie sollen zum einen möglichst PLA-gerecht sein, d. h. eine geringere Stromaufnahme bewirken, und zum anderen nur geringfügig von der ursprünglich beabsichtigten Spezifikation abweichen.

Zur Einordnung der Arbeit in den Kontext der PLA-bezogenen Forschung: Frühere Ansätze, wie z. B. PLA-Faltung, Ausgangs-Partitionierung, Dekodierte-PLAs o. ä. optimieren vordringlich die topologischen Ausmaße im Layout von PLA-Strukturen. Sie sind weniger darauf ausgerichtet, die Stomaufnahme zu minimieren.

Stichworte: ASIC, ESPRESSO, Heuristik, HOPE, Logik-Minimierung, Modifikation, Optimierung, PLA, PLA-Kosten, PLA-Tabelle, PLA-Stromaufnahme, ROM, Rundung, Spezifikation, Unschärfe, Wahrheitstabelle

Inhalt

Abbildungen

Tabellen

Glossar

Abtrennen	Bei einer reellen Zahl können durch *Abtrennen* (engl. *Truncation*) die Nachkommastellen unterdrückt werden.
ASIC	Integrierte Schaltung für kundenspezifische Anwendung (engl. *Application Specific Integrated Circuit*).
ESPRESSO	Ein Logik-Minimierer der *University of California, Berkeley*; er gilt bei der Logik-Minimierung als *de facto* Standard.
Heuristik	Ein Lösungsansatz, dem eine vereinfachende Hypothese zugrunde liegt und der als Hilfsmittel dem Zweck eines besseren Verständnisses dient.
HOPE	Im Rahmen dieser Arbeit implementiertes Entwurfswerkzeug zur Optimierung der Spezifikation von PLA-Tabellen (engl. *Heuristic Optimization on a PLA-Specification to minimize Expense*).
Konversion	Konvertierung eines reellen in einen ganzzahligen Wertebereich.
Logik-Minimierung	Ausgehend von einer Logikfunktion, lautet das Problem *Finde eine minimale Normalform*; es zählt zur Klasse der NP-vollständigen Probleme; siehe auch ESPRESSO.
Logikwerte	Symbol 0 entspricht logisch Null, Symbol 1 logisch Eins und Symbol * bzw. - dem Don't-Care.
LSB	Niederwertigste Bitstelle (engl. *Least Significant Bit*).
Modifikation	Innerhalb der Spezifizierung (*siehe* dort) geschehen Modifikationen und bewirken im allgemeinen eine neue PLA-Tabelle.
MSB	Höchstwertigste Bitstelle (engl. *Most Significant Bit*).
Optimierung	Eine Optimierung findet dadurch statt, daß die durch heuristische Methoden entstandenen Spezifizierung einer strategischen Auslese unterzogen werden. Übrig bleiben nur *günstige* Spezifizierungen, die einer hohe Reduzierung der PLA-Kosten bei gleichzeitig geringen Abweichungen erzielen können.
Option	Andersartige Vorgehensweise, die im Rahmen bestimmter Freiheiten beschritten werden kann.

PLA	Reguläre Schaltungsstruktur zur Realisierung von zweistufigen Schaltnetzen, die beim Chip-Entwurf effektiv als Makrozelle eingesetzt wird; historische Benennung: Programmierbare Logik-Anordnung (engl. *Programmable Logic Array*).
PLA-Tabelle	Darstellung von Logikfunktionen als (Wahrheits-) Tabelle, die direkt zum Entwurf von PLAs verwendet werden kann.
PLA-Kosten	Als dominierendes Kostenkriterium beim PLA erweist sich die Anzahl der (Produkt-) Terme (vgl. Abschnitt 2.2.5.6, S. 32).
ROM	Schaltungsstruktur für einen *Nur Lese Speicher* (engl. *Read Only Memory*).
Rundung	Einer reellen Zahl kann zur Unterdrückung der Nachkommastellen durch Rundung (engl. *Round*) entweder der ganzzahlige Ab- oder Aufrundwert zugeordnet werden.
Schichtenmodell	Modellierung der funktionalen Beziehungen in Kapitel 4, Abschnitt 4.1.4 (S. 60f).
Spezifikation	Ergebnis der Spezifizierung, wie sie Kapitel 3 erläutert wird; bestimmt die Generierung eine PLA-Tabelle.
Spezifizierung	Sequentieller Prozeß, aus dem die Spezifikation resultiert.
Stützstelle	Ein diskreter Argumentenwert, dessen Funktionswert alle Funktionswerte im Bereich der entsprechenden Quantisierungseinheit repräsentiert, auf den sich die Quantisierung der Funktion gewissermaßen *stützt*.
Stufen	Teilschritt im sequentiellen Prozeß der Spezifizierung (*siehe* dort).
Unschärfe	Eine in Kapitel 4, Abschnitt 4.2 (S. 63) vorgestellte Form von „Ungenauigkeiten", die inhärenter Bestandteil prinzipiell jeder Spezifikation ist.
ULSI	Höchstintegrierte Schaltung (engl. *Ultra Large Scale Integration*).
Vielfalt	Synonym für die mögliche Anzahl der verschiedenartigen Spezifizierungen.
VLSI	Hochintegrierte Schaltung (engl. *Very Large Scale Integration*).
Wahrheitstabelle	Tabellarische Repräsentation von aussagenlogischen Zusammenhängen (*siehe* PLA-Tabelle).
[Kra*95]	Synopsis der Literaturverweise mit *als *et al* Hinweis.

Kapitel 1

Einführung

1.1 Einleitung

Die Mikroelektronik wird in der öffentlichen Diskussion zu Recht als eine der zentralen Schlüsseltechnologien unserer Zeit verstanden. Die Märkte sind reich an Absatzmöglichkeiten und bieten einen weiten Raum für zukunftsweisende Innovationen. Maßgebliche Kriterien für die erfolgreiche Vermarktung eines mikroelektronischen Produktes sind das von der Konkurrenz abhebende Leistungsprofil und die ökonomische Bilanzierung.

Die immer anspruchsvoller werdenden Leistungsmerkmale für Mikroelektroniksysteme bedeuten ein ständiges Anwachsen der funktionellen Komplexität [Pos-89]. Zudem führen Fortschritte in der Halbleitertechnologie zu immer höheren Integrationsdichten; in der Fachterminologie wird dies unter dem Begriffen *Hoch-Integration* bzw. *Höchst-Integration* geführt, oder mit den englischen Synonymen VLSI für *Very Large Scale Integration* bzw. ULSI für *Ultra Large Scale Integration*. Als vordringlichste Problemstellungen beim des Schaltungsentwurfes treten nach [Pos-89, S. 148] hervor:

- **Entwicklungszeit:** Die Zeiträume, die typische Entwicklungen derzeit beanspruchen, nehmen etwa proportional mit der Anzahl der integrierten Transistoren zu.

- **Testaufwand:** Die Tests, die an einem gefertigten Mikroelektroniksystem abschließend durchgeführt werden müssen, verursachen einen erheblichen Zeitbedarf, der überproportional mit der Anzahl der integrierten Logikgatter ansteigt.

Zur Beherrschung der angedeuteten Problematik ist der massive Einsatz von rechnergestützten Methoden notwendig. Gegenwärtig werden besondere Anstrengungen unternommen, um die bisherige Gliederung in den drei Disziplinen *Entwurf*, *Fertigung* und *Test*, mit ihren entsprechenden Programmpaketen[1], zu einem integrierten System (Silicon Compiler) [Pos-89, S. 150] zusammenzufassen; einen Überblick gibt beispielsweise auch [deM*87].

[1]Entwurf	*Computer Aided Design* (CAD)
Fertigung	*Computer Aided Manufacturing* (CAM)
Test	*Computer Aided Test* (CAT)

1.1.1 Trends in der Mikroelektronik

Die Fortentwicklung der Entwurfsmethoden für Mikroelektroniksysteme steht immer unter dem Anspruch, dem technologischen Fortschritt rasch folgen zu müssen. Die Schwerpunkte dabei sind aber auch stets von den ökonomisch geprägten Markterfordernissen beeinflußt. Im folgenden wird stellvertretend auf zwei der momentan am Mikroelektronikmarkt herrschenden Trends näher eingegangen, nämlich einerseits die *kundenspezifischen Applikationen* und andererseits die *Energieeinsparung*.

1.1.1.1 Kundenspezifische Applikationen

Allgemein ist durch immer leistungsfähigere Entwurfsumgebungen eine Verringerung der Entwurfszeiten zu verzeichnen, die zur Senkung der Kosten führt. Auch im Bereich der Fertigung von Mikroelektronik-Bausteinen konnten die Produktionskosten entscheidend gesenkt werden.

Ferner hat bei Chip-Herstellern eine Umorientierung stattgefunden: Neben der kostengünstigen Massenfertigung von universell zu programmierenden Standardlösungen bietet die Industrie zunehmend kundenspezifische[2] Lösungen günstig an. Sogar hoch spezialisierte Einzellösungen bzw. einzelne Prototypen können vergleichsweise günstig hergestellt werden.

1.1.1.2 Energieeinsparung

In fast allen Anwendungsbereichen von Mikroelektroniksystemen setzt sich das Bestreben nach Energieeinsparung durch, denn für sogenannte *Low-Power* Schaltkreise öffnen sich neue Märkte, insbesondere dort, wo als Energiequellen Batterien, Akkumulatoren, Solarzellen oder dergleichen zum Einsatz kommen. Als Beispiele für sogenannte *autonome Systeme* seien Handys bzw. Mobil-Telefone, Laptops u. ä. genannt.

Außerdem kann eine Reduzierung des Energiebedarfs zu einer Verringerung der Ausfallwahrscheinlichkeit beitragen, da diese unter dem Einfluß von hoher Verlustwärme steht. Ebenso können Effekte, wie schnelle Alterung und störende Verschiebungen von Arbeitspunkten, auf diese Weise minimiert werden.

1.1.2 Entwurfsengpässe im Wandel

Waren bisher die *technologischen* Bedingungen für die Realisierung von Mikroelektroniksystemen als herausfordernd anzusehen, so ist es mittlerweile die *funktionelle* Komplexität der Systemlösungen; sie wird durch steigende Anfordungen und die erweiterten Möglichkeiten der höheren Integrationsdichte sowie dem hohen Automatisierungsgrad im Entwurfsprozeß begünstigt. Der Schwerpunkt beim Entwurf digitaler Systeme hat sich aufgrund dieser Umstände von der *physikalischen* Ebene hin zur *logischen* Ebene verlagert [Pos-89]. Zur Veranschaulichung der verschiedenen Engpässe im Wandel der Entwurfsschwerpunkte dient Abbildung 1.1 (S. 3), von der nachfolgend zunächst nur die Teilbilder (a) und (b) betrachtet werden.

[2] Gemeint ist hier vor allem das sogenannte ASIC (engl. *Application Specific Integrated Circuit*)

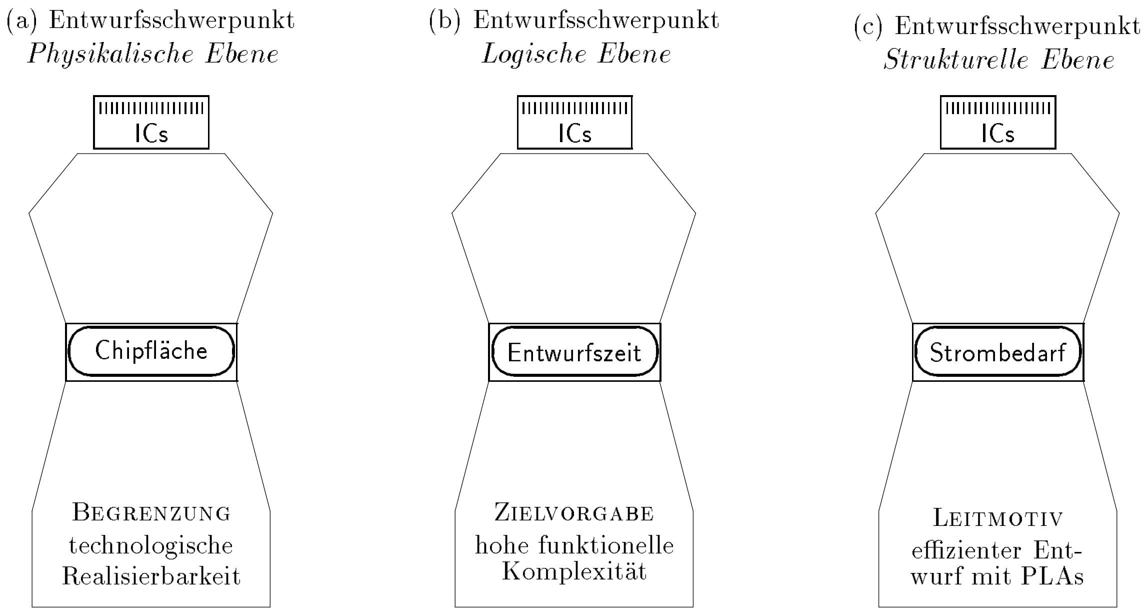

Abbildung 1.1: Engpaß (engl. *Bottleneck*) im Wandel der Entwurfsschwerpunkte

(a) **Entwurfsschwerpunkt: Physikalische Ebene**
Unter schwerpunktmäßiger Betrachtung der *physikalischen* Entwurfsaspekte kristalisiert sich die *technologische* Realisierbarkeit als Begrenzung heraus. Behindert wird der Entwurfsprozeß dabei durch den Engpaß der Chipfläche. Eine Überschreitung der zugelassenen Grenzen führt zu einem erneuten Entwurfsversuch.

(b) **Entwurfsschwerpunkt: Logische Ebene**
Der verlagerte Entwurfsschwerpunkt liegt nun bei den *logischen* Aspekten. Hier ist als Ziel die hohe *funktionelle* Komplexität vorgegeben. Die Kosten sind zum wesentlichen Teil durch die Dauer einer Entwicklung bestimmt, so daß die Entwurfszeit als ein Engpaß wirkt. Es kommt daher entscheidend auf die Verwendung effektiver Methoden an.

(c) **Entwurfsschwerpunkt: Strukturelle Ebene** *siehe* Abschnitt 1.1.3.1 (S. 4).

Lag der Schwerpunkt der Forschungsbemühungen bisherig bei (a) in der Optimierung des Bauteileaufwandes mit der Zielsetzung, möglichst *wenige* und *kleine* Bauteile zu entwerfen, so konzentrieren sie sich nun bei (b) auf die automatische Generierung effizienter Realsierungsstrukturen mit der Zielsetzung, möglichst *einfach* und *schnell* zu generierende Strukturen zu erzeugen, die außerdem auch leicht zu ändern sind. Die nachfolgende Diskussion widmet sich nun der Situation beim heute aktuellen dritten Entwurfsschwerpunkt (c).

1.1.3 Motivation der Aufgabenstellung

PLAs realisieren Logikfunktionen in einer verdichteteren Form als vergleichbare Alternativen und lassen sich einfach automatisch generieren. Sie erweisen sich als sehr effiziente Entwurfsobjekte, deren kompaktes Layout in Full-Custom-Qualität optimiert ist.

1.1.3.1 Engpaß bei PLAs

In der Entwurfspraxis kommt es nicht selten vor, daß ein *ad hoc* entworfenes PLA zu aufwendig ist: Entweder übersteigt dabei die Anzahl der Terme die in der Entwurfsumgebung gesetzte Grenze von z. B. 128 Termen[3], oder, was noch häufiger auftritt, kommt es aufgrund der allgemeinen Bestrebung zur Energieeinsparung vor, daß die zu erwartende Stromaufnahme unakzeptabel hoch ist. Statt einer Realisierung mit der PLA-Struktur kommen dann meistens alternative Lösungen mit Hilfe von ROM-Strukturen zum Einsatz [Rau-92]. Bei derartigen Ersatzlösungen wird dabei ein größerer Flächenbedarf in Kauf genommen, was teilweise erhebliche Einschränkungen im Chip-Layout nach sich zieht.

In Anlehnung an die Teilbilder (a) und (b) in Abbildung 1.1 (S. 3) ist die geschilderte Problematik in Teilbild (c) entsprechend angedeutet: Wird der Schwerpunkt beim Entwurf wie im Kontext dieser Arbeit auf die *strukturelle* Ebene gelenkt, so zeigt sich, daß im Bemühen um einen Entwurf mit *effizienten* Strukturen, wie den *regulären* PLAs, ein Engpaß bei der Stromaufnahme liegt.

1.1.3.2 Ansatz und Zielsetzung der Arbeit

Bei näherem Betrachten typischer *ad hoc* Spezifikationen von PLAs sind verdeckte Optionen bzw. inhärente Unschärfe aufzuzeigen. Dieser Umstand kann insofern ausgenutzt werden, daß geeignete Modifikationen an Spezifikationen zur Senkung der Stromaufnahme führen können. Eigene Untersuchungen zeigten, daß bereits schon geringfügig manipulierte Spezifikationen, die sich im Rahmen der erwähnten Optionen bzw. Unschärfe bewegen, zu einer beachtlichen Senkung der Stromaufnahme auf etwa ein Drittel führen können [Kra-94].

Die vorliegende Arbeit befaßt sich zunächst mit der Realisierung von Logikfunktionen durch PLAs, diskutiert verschiedene Aspekte bei der Spezifikation von Tabellen. Anschließend werden heuristische Methoden vorgestellt, durch deren Einsatz die Spezifikationen optimiert werden können. Das grundsätzliche Ziel ist dabei eine *Minimierung der Stromaufnahme des PLAs bei möglichst geringfügiger Modifikation.* Anhand typischer, teilweise konkret aufgetretener Beispiele aus der Praxis werden die Methode evaluiert, analysiert und schließlich bewertet.

Zusätzlich wird die Implementierung eines Entwurfswerkzeugs mit dem Namen HOPE (Heuristic Optimization on a PLA-Specification to minimize Expense) erörtert, das sich in den Entwurfvorgang integrieren läßt. Es wurde in der Programmiersprache *C* [Ker*-88, Bar-89] unter dem Gesichtspunkt hoher Portabilität geschrieben und kann unabhängig von Einschränkungen bezüglich eines bestimmten Betriebssystems genutzt werden. Mit Hilfe von HOPE können die beschriebenen heuristischen Modifikationen an Spezifikationen vorgenommen und bewertet werden.

[3] Als Beispiel sei die ES2 Entwurfsumgebung SOLO-2030 angeführt [ESS-92]: Im Anhang A sind Angaben zu den dort gesetzten Grenzen in Abbildung A.1 (S. 155) aufgeführt.

1.2 Mathematische Grundlagen

Die Lösung von Problemen mit Hilfe von Mikroelektroniksystemen erfordert Entwurfsprozesse, die aus mehreren Teilschritten bestehen. In den dabei durchlaufenden Abstraktionsebenen geschehen durch Transformationen unterschiedliche Repräsentationen der Problemlösung. Die Spezifikation des gewünschten Systemverhaltens geschieht zunächst in einer *funktionellen* Repräsentation. Entwürfe mit ROMs oder PLAs gestatten eine einfache Übertragung von *funktionellen* Beschreibungen in realisierende Strukturen.

1.2.1 Funktionen zur Problemlösung

Eine Funktion ist eine *injektive Abbildung* zwischen zwei Mengen, und zwar einerseits dem *Definitionsbereich* und andererseits dem *Funktionswertebereich*[4]. Dabei ordnet sie jedem Element des Definitionsbereichs *genau ein* Element des Funktionswertebereichs zu. Beide Mengen können im allgemeinen als Teilmengen übergeordneter Mengen aufgefaßt werden.

1.2.1.1 Totale bzw. partielle (Logik-) Funktionen

Sei $\mathbf{U}$ eine (übergeordnete) Menge und seien $\mathbf{X}$ und $\mathbf{Y}$ Teilmengen von $\mathbf{U}$. Dann ist die Abbildung f wie folgt definiert:

$$f : \mathbf{X} \rightarrow \mathbf{Y} \qquad \text{mit} \quad \mathbf{X}, \mathbf{Y} \subseteq \mathbf{U} \tag{1.1}$$

Ist $\mathbf{X} = \mathbf{U}$, d. h. f ist auf *jedem* Element von $\mathbf{U}$ definiert, dann heißt f *total*. Falls $\mathbf{X} \not\subset \mathbf{U}$ gilt, heißt f *partiell*. Beim Entwurf von Schaltwerken treten überwiegend partielle Funktionen auf. Im Kontext *schaltungstechnischer* bzw. *logischer* Problemlösungen werden die zu diskutierenden Funktionen auch als *Schaltfunktionen* bzw. *Logikfunktionen* bezeichnet. Da die *Boolesche Algebra* zur Formalisierung der logischen Zusammenhänge dient, ist auch der Begriff *Boolesche Funktionen* verwendet.

1.2.2 Logikfunktionen in Binärdarstellung

Die Binärdarstellung von Zahlen basiert auf dem *Dualen Zahlensystem* und den zu Darstellung verwendeten Symbolen 0 und 1; es sei daher die Symbolmenge $\mathbb{B} := \{0, 1\}$ definiert. Für die Zahlendarstellung in $k \in \mathbb{N}$ Binär- bzw. Bitstellen wird $\mathbb{B}^k$ gebildet.

[4] Gebräuchlich sind auch folgende Begriffe für den

- Definitionsbereich: *Vorbereich* oder *Eingangsbereich* (engl. *Domain*).
- Funktionswertebereich: *Nachbereich* oder *Ausgangsbereich* (engl. *Range*).

1.2.2.1 Binärdarstellung natürlicher Zahlen

Zur Kodierung von natürlichen Zahlen in der Binärdarstellung sei folgende Funktion definiert:

$$\mathsf{Bin} : \mathbb{N} \to \mathbb{B}^k \quad \text{mit} \quad k \in \mathbb{N} \tag{1.2}$$

Dabei gibt k die Anzahl der für die Binärdarstellung benötigten Bitstellen an. Umgekehrt läßt sich für eine beliebige natürliche Zahl $a \in \mathbb{N}$, $a \neq 0$ die Anzahl der für die Binärdarstellung benötigten Bitstellen durch folgende Funktion bestimmen:

$$\mathsf{Bits}(a) := \min\Big(\{j \in \mathbb{N} \mid j > \log_2(a)\}\Big) \quad \text{für} \quad a \in \mathbb{N},\ a \neq 0 \tag{1.3}$$

Es wird die nächste, echt größere Ganzzahl zum dualen Logarithmus des Argumentes $a > 0$ bestimmt. Mit der Festlegung von $k := \mathsf{Bits}(a)$ ergibt sich damit gleichzeitig eine Beschränkung für die Menge der binär mit k Bitstellen darstellbaren natürlichen Zahlen:

$$a \in \{0, \ldots, 2^k - 1\} \subseteq \mathbb{N} \tag{1.4}$$

1.2.2.2 Evaluation von Logikfunktionen

Zur binären Darstellung des Eingangsbereiches einer Funktion f seien n Bitstellen nötig, für den Ausgangsbereich entsprechend m Bitstellen. Auf den Schaltungsentwurf bezogen, bei dem in die Logikebene hinein abstrahiert wird, geben n und m die Anzahl der Ein- bzw. Ausgangsvariablen an. In dieser Ebene betrachtet, läßt sich eine Logikfunktion für *jede* Belegung der Eingangsvariablen zu dem Logikwert `1` oder `0` evaluieren. In diesem Sinne bilden derartige Logikfunktionen den Eingangsbereich *total* auf den Ausgangsbereich ab. Sie sind wie folgt definiert:

$$f_{\mathrm{bin}}\Big|_{\mathrm{tot}} : \mathbb{B}^n \to \mathbb{B}^m \tag{1.5}$$

Allerdings sind Logikfunktionen häufig nur unvollständig definiert; dennoch ist es erforderlich, daß bei einer schaltungstechnischen Realisierung für nicht definierte Eingangsbelegungen eine Ausgangsbelegung zugelassen wird. Die gewonnenen Ergebnisse sind jedoch für die Problemlösung nicht relevant. Mit dem Symbol `*`, gebräuchlich sind auch `2` und `-`, als unbestimmten Logikwert[5], der sich entweder zu `0` oder zu `1` evaluieren läßt, stellt eine partiell definierte Logikfunktion nun folgende Abbildung dar:

$$f_{\mathrm{bin}}\Big|_{\mathrm{part}} : \mathbb{B}^n \to \mathbb{D}^m \quad \text{mit} \quad \mathbb{D} := \mathbb{B} \cup \{*\} \tag{1.6}$$

Bei der Evaluation von f für eine undefinierte Eingangsbelegung wird formal der *Don't-Care* Logikwert `*` als Ausgangsbelegung zugeordnet; der tatsächlich in einer schaltungstechnischen Realisierung resultierende Logikwert `0` oder `1` ist für die Problemlösung irrelevant.

[5] Es handelt sich nicht um einen undefinierten Logikwert, sondern lediglich seine tatsächliche Evaluierung ist offenbar irrelevant bzw. „egal“; im Englischen wird er daher auch mit *D̲on't-C̲are* (DC) bezeichnet.

1.2.2.3 Merkmale binärer Tabellen

Bei binären Tabellen wird für alle Indizes und für alle Einträge jeweils eine einheitliche Anzahl der Bitstellen gewünscht: Dabei sollen die größten Indizes bzw. Einträge vollständig darstellbar sein und die kleineren erhalten Vornullen. Hierzu ist die Anzahl der Bitstellen für das jeweilige *Maximum* der Menge zu bestimmen.

$$\begin{aligned} N &:= \max(\mathbf{X}) \\ M &:= \max(\mathbf{Y}) \end{aligned} \tag{1.7}$$

Von beiden Maxima ist nun die Anzahl der Bitstellen zu bestimmen:

$$\begin{aligned} n &:= \mathsf{Bits}(N) \\ m &:= \mathsf{Bits}(M) \end{aligned} \tag{1.8}$$

Damit sind die Ausmaße einer binären Tabelle festgelegt; dieses wird durch das sogenannte *Format* der Tabelle deutlich gemacht: Eine Tabelle mit 2^n Einträgen à m Bit hat ein „$n \times m$ Format“.

Die durch N beschränkte Indexmenge $\mathbf{X}$ hat bei binären Tabellen im allgemeinen die Gestalt $\{0, \ldots, N-1\}$, d. h. sie beginnt bei 0 und reicht fortlaufend bis $N-1$. Vereinfachend kann eine Indexmenge wie folgt ausgedrückt werden:

$$\begin{aligned} \mathbf{I}_N &:= \{0, \ldots, N-1\} \\ &:= \{j \in \mathbb{N} \mid 0 \leq j < N\} \subseteq \mathbb{N} \end{aligned} \tag{1.9}$$

Bei $\mathbf{I}_N$ gibt somit der Index N die *exklusive* obere Schranke der begrenzten Indexmenge an. Tabellen mit einem $n \times m$-Format bilden somit zwischen $\mathbf{X} := \mathbf{I}_n$ und $\mathbf{Y} := \mathbf{I}_m$ ab.

1.2.2.4 Notation bei der Binärdarstellung

Der Binärdarstellung entsprechend wird für jeden Index $\mathbf{x} \in \mathbf{X} := \mathbf{I}_n \subseteq \mathbb{N}$ und jeden Eintrag $\mathbf{y} \in \mathbf{Y} := \mathbf{I}_m \subseteq \mathbb{N}$ der Tabelle bzw. jeden Ein- und Ausgangswert der Logikfunktion folgende Binärnotation[6]verwendet:

$$\begin{aligned} (x_{n-1}, \ldots, x_0) &:= \mathsf{Bin}_n(\mathbf{x}) \quad \text{mit} \quad x_0, \ldots, x_{n-1} \in \mathbb{B} \\ (y_{m-1}, \ldots, y_0) &:= \mathsf{Bin}_m(\mathbf{y}) \quad \text{mit} \quad y_0, \ldots, y_{m-1} \in \mathbb{D} \end{aligned} \tag{1.10}$$

[6] Üblicherweise werden Klammern und Kommata in der Binärdarstellung weggelassen.

1.2.3 Repräsentationsformen für Logikfunktionen

Je nach Abstraktionsebene im Entwurfsprozeß bieten sich verschiedene Formen der Repräsentation für Logikfunktionen an. Neben grafischen Möglichkeiten, wie den *Veitch-Diagrammen* [Gil*80], sind vor allem die *tabellarischen* und die *algebraischen* Darstellungen von Bedeutung.

1.2.3.1 Logikfunktionen in tabellarischer Darstellung

Die binäre Darstellung einer Logikfunktion tabellarischer Form ist insofern eine naheliegende Repräsentation, da aus ihren Zeilen logische Aussagen über das spezifizierte Verhalten formuliert werden können. Wegen dieser aussagenlogischen Darlegungsweise werden die Tabellen auch als *Wahrheitstabellen* bezeichnet.

In dieser Form enthält die Tabelle für jeden definierten Eingangswert den zugeordneten Ausgangswert, d. h. bei partiell definierten Funktionen kommen die Zeilen für undefinierte Argumente nicht vor. In Tabelle 1.1 (S. 8) ist beispielsweise eine Wahrheitstabelle für die drei Eingangsvariablen $x_2, \ldots, x_0$ und die zwei Ausgangsvariablen y_1 und y_0 gegeben.

$\mathbf{x} \in \mathbb{B}^3$			$\mathbf{y} \in \mathbb{B}^2$	
x_2	x_1	x_0	y_1	y_0
0	0	0	1	1
0	0	1	1	0
0	1	0	0	1
0	1	1	0	1
1	0	0	1	0
1	0	1	1	0
1	1	0	1	1
1	1	1	0	1

Tabelle 1.1: Beispiel für eine Wahrheitstabelle

Wahrheitstabellen haben in der Palette der Darstellungsformen einen primären Stellenwert; oft können sie direkt aus elementaren Entwurfsüberlegungen heraus zusammengestellt werden. Ihre direkte Übertragung in Speicherstrukturen, wie beispielsweise dem ROM, gestattet eine einfache Handhabung beim Systementwurf.

1.2.3.2 Logikfunktionen in algebraischer Normalform

Eine Wahrheitstabelle mit mehreren Ausgangsvariablen gibt ein Bündel von Logikfunktionen wieder. Diese sind aus booleschen Ausdrücken zusammengesetzt, die aus beliebigen Kombinationen der Eingangsvariablen in *bejahter* oder *verneinter* Form bestehen. Die so gebildeten Logikfunktionen sind eine *algebraische* Repräsentation eines Problems. Allerdings vermeidet man algebraische Repräsentationsformen beim Schaltungsentwurf, da sie im allgemeinen nicht regulär und daher beim Entwurf unhandlich sind. Eine Ausnahme bildet die *Disjunktive Normalform* (DNF), die eine standardisierte Darstellungsweise ist. Sie kann in einer trivialen Prozedur aus einer Wahrheitstabelle abgeleitet werden.

Disjunktive Normalform (DNF)

Zur Bildung der Disjunktiven Normalform von Logikfunktionen werden für jede Spalte in der Wahrheitstabelle die folgenden Schritte durchgeführt:

1. Zunächst werden bei jeder Logikfunktion *konjunktive* Ausdrücke zusammengestellt: Dazu wird für jede `1` in der Funktionsspalte (Ausgangsvariable) die vollständige Kombination aus *bejahten* oder *verneinten* Eingangsvariablen gebildet. Diese Terme heißen auch *Minterme*. Da jeder Term die jeweilige Funktion impliziert, wird er auch *Implikant* genannt.
2. Anschließend werden die Funktionsgleichungen durch die boolesche Summen über die jeweiligen Minterme gebildet, d. h. alle Minterme einer Funktion müssen dazu *disjunktiv* verknüpft werden.

Eine Logikfunktion in DNF kann direkt auf die Struktur eines 2-stufigen Schaltnetzes abgebildet werden, so daß die DNF eine günstige Ausgangsposition für die Realisierung von Logikfunktionen bietet.

Anhand des vorangegangenen Beispiels der Wahrheitstabelle in Abbildung 1.1 (S. 8) ergeben sich nach dieser Prozedur folgende boolesche Funktionsgleichungen in DNF:

$$\begin{aligned} y_0 &:= \overline{x}_2\,\overline{x}_1\,\overline{x}_0 + \overline{x}_2\,x_1\,\overline{x}_0 + \overline{x}_2\,x_1\,x_0 + x_2\,x_1\,\overline{x}_0 + x_2\,x_1\,x_0 \\ y_1 &:= \overline{x}_2\,\overline{x}_1\,\overline{x}_0 + \overline{x}_2\,\overline{x}_1\,x_0 + x_2\,\overline{x}_1\,\overline{x}_0 + x_2\,\overline{x}_1\,x_0 + x_2\,x_1\,\overline{x}_0 \end{aligned} \tag{1.11}$$

Zwischen Mintermen und Indizes besteht eine direkte Abhängigkeit: Sei N die Anzahl der Einträge einer Wahrheitstabelle und $x \in \mathbf{I}_N$ ein Index für einen Eintrag. Für jede `1` in einer der m Spalten bzw. Logikfunktionen in der x-ten Tabellenzeile wird der entsprechende Minterm anhand des Indexes x gebildet. Aufgrund der Abhängigkeit vom jeweiligen Index der `1`, kann der Minterm durch die binäre Kodierung des Indexes identifiziert werden, d. h. eine `1` im Index an der Bitposition j, wobei $0 \leq j < n$ gilt, deutet auf die bejahte Form des zugehörigen x_j im Minterm hin, umgekehrt weist eine `0` auf die verneinte Form $\overline{x}_j$ hin. Beispielsweise können die Minterme aus Gleichung 1.11 in folgender Weise durch Index-Identifikation dargestellt werden:

$$\begin{aligned} y_0 &: \texttt{000 010 010 110 111} \\ y_1 &: \texttt{000 001 100 101 110} \end{aligned} \tag{1.12}$$

Jede Stelle in einem Term, die mit `1` bzw. `0` notiert ist, stellt eine *bejahte* bzw, *verneinte* Variable, ein sogenanntes *Literal* dar. Kommt eine Variable in einem Term nicht vor, so wird dies durch ein `-` gekennzeichnet; dies bedeutet, daß ein derartiger Term alle Minterme bzw. Indizes überdeckt, bei denen die vorhandenen Literale miteinander übereinstimmen. Hat ein Term wenig Literale, so überdeckt er viele Minterme bzw. Indizes.

1.2.3.3 Komplementäre Logikfunktion

In manchen Fällen ist die Beschreibung bzw. Auswertung einer Logikfunktion in der gegebenen Form zu aufwendig bzw. sehr erschwert. Statt die Logikfunktion f zu evaluieren, bietet sich die komplementäre Logikfunktion $\overline{f}$ an; sie erlaubt häufig eine einfachere Betrachtung und ist folgendermaßen definiert:

$$\overline{f} : \mathbf{U} \setminus \mathbf{X} \rightarrow \mathbf{U} \setminus \mathbf{Y} \tag{1.13}$$

Die Evaluation von $\overline{f}$ hat zur Folge, daß zur Realisierung von f eine zusätzliche Invertierung implementiert werden muß. Bei der Gewinnung der algebraischen Repräsentation von $\overline{f}$ aus der Wahrheitstabelle wird ähnlich wie bei der DNF vorgegangen.

Gewinnung der komplementären Funktion

Im Gegensatz zur Bildung der DNF-Repräsentation einer Logikfunktion f, bei der die Minterme für die `1`en, die sogenannten *Einsterme*, zusammenzustellen, geschieht dies nun für die `0`en, d. h. es werden die *Nullterme* bzw. die *Maxterme* gebildet.

Die Funktionsgleichungen werden nun, analog zur DNF, durch boolesche Summen über den resultierenden Termen gebildet.

Die ersatzweise Betrachtung der komplementären Logikfunktion bietet sich immer dann an, wenn es mehr `1`en als `0`en in der Wahrheitstabelle gibt.

1.2.4 Realisierungsaufwand bei Logikfunktionen

Logikfunktionen werden durch Schaltnetze realisiert. Bei deren Entwurf ist der dafür erforderliche Aufwand von besonderem Interesse. Grundsätzlich bedeutet eine *einfach* gestaltete Logikfunktion auch eine *einfache* Realisierung als Schaltnetz. Um den Aufwand zu minimieren, erscheint der Ansatz, boolesche Funktionen zu vereinfachen, technisch plausibel.

1.2.4.1 Aufwandsaspekte

In [Gil*80] wird ausgeführt, daß sich in einer ersten Näherung zunächst mit der Anzahl der Gattereingänge der Aufwand für eine Realisierung abschätzen läßt. Was andere Aspekte, wie beispielsweise die Schaltgeschwindigkeit oder den Bauteilepreis angeht, so muß in der Regel ein Kompromiß zwischen Geschwindigkeit und Preis gesucht werden. Untersucht man bei gleichen Logikfunktionen die extremen Positionen, so ergeben sich folgende Zusammenhänge:

- **Zweistufige Schaltnetze** sind *teuer*, dafür aber *schnell.*
- **Mehrstufige Schaltnetze** sind *billig* und *langsam.*

Gegenwärtig hat sich die Kostenfrage für Schaltnetze durch die immer höhere Integrationsdichte nahezu erübrigt, so daß im wesentlichen nur noch der Geschwindigkeitsaspekt relevant ist. Daher werden überwiegend nur noch zweistufige Schaltnetze eingesetzt.

1.2.4.2 Logik-Minimierung

Je nach Kontext können noch weitere Gesichtspunkte von Bedeutung sein, z. B. Leistungsaufnahme, Fragen zur Wärmebelastung, Platzbedarf, entwurfsrelevante logische Eigenschaften, Stückzahl etc. Bezogen auf die technische Fragestellung liegt der Hauptaspekt für die weitere Diskussion in der Vereinfachung von Schaltnetzen. Die Logik-Minimierung hat sich als eigene Disziplin ausgebildet und liefert zur Vereinfachung den wesentlichsten Beitrag.

Das grundsätzliche Ziel besteht darin, zu gegebenen Funktionen eine äquivalente und nicht weiter zu vereinfachende Normalform zu finden, die jedoch noch nicht minimal sein muß. Eine solche vereinfachte Normalform ist die DNF (vgl. Abschnitt 1.2.3.2, S. 9). Sie entspricht technisch direkt bereits einem zweistufigen Schaltnetz mit Und- und Oder-Gliedern, dessen Bauteile insgesamt ein Minimum an Eingängen haben [Gil*80].

Allerdings sind Logikfunktionen im allgemeinen *nicht* minimal. Da das Problem *Finden einer minimalen Normalform* zur Klasse der NP-vollständigen Probleme zählt, sind in der Vergangenheit eine Reihe von Verfahren entwickelt worden, um bei der steigenden Problemgröße von dem Aufwand manueller Minimierungen frei zu werden:

- **Karnaugh-Veitch:** Hierbei handelt es sich um ein grafisches Verfahren, das Veitch-Diagramme als Hilfsmittel anwendet und auf eine manuelle Bearbeitung beschränkt ist [Gil*80]. Die Güte des Verfahrens liegt in der individuellen Erfahrung des Anwenders, geeignete Muster in den Diagrammen zu erkennen. Ab einer gewissen Komplexität der Aufgabe (maximal bis etwa 6 Eingangsvariablen) ist dieses Verfahren allerdings nicht mehr empfehlenswert.

- **Quine-McCluskey:** Ein algorithmisches Verfahren, das automatisch bearbeitet werden kann. Es greift zunächst die Menge aller Minterme der Logikfunktion auf und berechnet in einem ersten Schritt die Menge der sogenannte *Primimplikanten.* In einem zweiten Schritt wird nachdem eine Klassifizierung von *wesentlichen* und *unwesentlichen* Primimplikanten erfolgt ist, eine nicht-eindeutige Teilmenge von *unwesentlichen* Primimplikanten zu den *wesentlichen* Primimplikanten hinzugenommen [McC-65, Reu-72, Gil*80]. Der Hauptnachteil bei dieser Methode besteht in dem großen Speicherbedarf.

- **Iterierte Consensus Methode:** Dieser Algorithmus geht im ersten Schritt von einer Logikfunktion in DNF-Darstellung aus und bestimmt durch Iterieren der sogenannten *Consensus Methode* die Menge der Primimplikanten [Lar-75].

Consensus Methode

Diese Methode kombiniert zwei Produktterme miteinander und generiert einen Konsens-Term. Seien dafür $\mathbf{p} := p_{n-1} \dots p_0$ und $\mathbf{q} := q_{n-1} \dots q_0$ zwei n-Bit Produktterme. Ferner sei Dist eine Funktion, die die Anzahl der verschiedenen Literale in $\mathbf{p}$ und $\mathbf{q}$ bestimmt (engl. *Distance*), d. h.

$$\mathsf{Dist}(\mathbf{p},\mathbf{q}) := \left|\{j \mid (p_j, q_j \neq \texttt{-}) \ \mathbf{und}\ (p_j \neq q_j)\right| \tag{1.14}$$

Dann ist der sogenannte Konsens (engl. *Consensus*) wie folgt definiert:

$$\mathsf{Cons}(\mathbf{p},\mathbf{q}) := \begin{cases} \mathbf{p} \equiv \mathbf{q} : & \mathsf{Dist}(\mathbf{p},\mathbf{q}) = 0 \\ \mathbf{p} \cap \mathbf{q} : & \mathsf{Dist}(\mathbf{p},\mathbf{q}) = 1 \\ \emptyset \quad : & \mathsf{Dist}(\mathbf{p},\mathbf{q}) \geq 2 \end{cases} \tag{1.15}$$

Ist die Anzahl der verschiedenen Literale $\mathsf{Dist}(\mathbf{p},\mathbf{q}) \geq 2$, so existiert *kein* Konsens zwischen $\mathbf{p}$ und $\mathbf{q}$, d. h. der Konsens-Term $\mathsf{Cons}(\mathbf{p},\mathbf{q}) := \emptyset$. Für $\mathbf{p} = \mathbf{q}$, also $\mathsf{Dist}(\mathbf{p},\mathbf{q}) = 0$ ist der Konsens-Term gleich $\mathbf{p}$ bzw. gleich $\mathbf{q}$.

Der interessanteste Fall tritt jedoch dann ein, wenn $\mathbf{p}$ und $\mathbf{q}$ sich in nur *genau einem* Literal unterscheiden, d. h. $\mathsf{Dist}(\mathbf{p},\mathbf{q}) = 1$; der Konsens-Term ist dann $\mathsf{Cons}(\mathbf{p},\mathbf{q}) := \mathbf{p} \cap \mathbf{q}$, wobei damit zum einen das Übernehmen der in $\mathbf{p}$ und $\mathbf{q}$ identisch vorkommenden Bitstellen aus $\mathbb{D}$ gemeint ist und zum anderen das nicht identische Literal durch - ersetzt wird.

Der Konsens-Term $\mathsf{Cons}(\mathbf{p},\mathbf{q})$ *überdeckt* damit die beiden, in dieser besonderen Weise verschiedenen Profuktterme $\mathbf{p}$ und $\mathbf{q}$; dieser Art der Überdeckung zweier Produktterme wird auch als *Verschmelzungsregel* bezeichnet. Näheres über die *Consensus Methode* ist in [Giv-70] und [Bra*84, S. 25f] finden.

Ähnlich zum Quine-McCluskey Algorithmus geschieht der zweite Schritt bei der *Iterierte Consensus Methode*, die jedoch schneller als der Quine-McCluskey Algorithmus ist und weniger Speicher benötigt, da nicht alle Minterme vorzuhalten sind [Giv-70, Reu-72].

- **ESPRESSO:** Besonders hervorgetan hat sich in den 80er Jahren die Gruppe um *Brayton, Hachtel* und *Sangiovanni-Vincentelli* [Bra*82b, Bra*84], deren Bemühungen in Gestalt einer Zusammenstellung verschiedener Heuristiken allgemein in Gebrauch sind: Der Logik-Minimierer Espresso (University of California, Berkeley) ist als *de facto* Standard ein wesentlicher Beitrag für den praktischen Entwurf von Schaltnetzen, da er die Anforderungen, die der Ansatz des *Silicon Compilers* stellt, erfüllt, nämlich auch in extremen Spezialfällen Ergebnisse zu liefern. In [Bra*84, Kapitel 7, S. 161] wird in diesem Zusammenhang von zwei Extrembeispielen realer PLAs berichtet, an denen mit Hilfe von Espresso erfolgreich eine Logik-Minimierung durchgeführt werden konnte: Das erste weist 152 Ein- und 382 Ausgangsvariablen auf und das zweite hat 3236 Produktterme. Derartig extreme Beispiele sind für den Kontext dieser Arbeit allerdings irrelevant.

Kapitel 2

Realisierung von Logikfunktionen

Die Berücksichtigung funktionaler Beziehungen in einer Problemlösung ist von zentraler Bedeutung. Dieses schlägt sich bereits im Systementwurf nieder: Systeme, deren Realisierung in VLSI erfolgen soll, werden in einer Folge von hierarchisch gegliederten Stufen entworfen [Pos-89]. Dabei entsteht zunächst eine schematische Darstellung von Subsystemen mit ihren Verknüpfungen, z. B. in Form eines Blockschaltbildes.

Nachdem auf diese Weise eine Systemdefinition erstellt wurde, erfolgt für jedes der Subsysteme eine Auflösung in funktionale Blöcke, die mit Hilfe von logischen Gleichungen, Zustandsdiagrammen oder verbalen Spezifikationen beschrieben werden. Durch mehrere hierarchische Entwurfsschritte entsteht nun die Umsetzung in die Gatterebene.

Da bei der Entwicklung von Schaltungen die Kosten ganz wesentlich von der Entwurfszeit abhängen, bemüht man sich gegenwärtig nicht mehr so sehr um eine Optimierung der Schaltung bezüglich des Bauteileaufwandes. Der Trend geht — mit dem Ziel einer zeitsparenden Automatisierung des Entwurfsprozesses — hin zu *regulären* Strukturen, die schnell durch Parametrisierung auf ein individuelles Schaltungsproblem angepaßt werden können. Im Vergleich zu willkürlich strukturierten Layouts handelt es sich dabei um iterierte Transistor-Strukturen, deren Layout schneller zu erstellen und leichter zu verifizieren ist [Hon*-84].

2.1 Implementierung mit ROM

Der *Nur Lese Speicher* (engl. *Read Only Memory*, ROM) ist eine 2-dimensionale Struktur und stellt die direkte Realisierung einer Wahrheitstabelle dar. Für alle möglichen n-Bit Argumente am Eingang liefert das ROM, entsprechend der Funktionsdefinition, den jeweils zugeordneten m-Bit Funktionswert am Ausgang.

2.1.1 ROM als funktionaler Block

Das ROM besteht als funktionaler Logikblock aus einer Speichermatrix und einem Dekodierer. In Abbildung 2.1 (S. 14) ist ein schematischer Einblick in die interne Struktur gezeigt. Zunächst wird der *Dekodierer* durch die n Eingangssignale des ROMs angesteuert. Dieser Block ermöglicht über die internen Auswahlleitungen die Selektion einer der 2^n Zeilen aus der ROM-Matrix.

Die *ROM-Matrix* enthält in 2^n Zeilen, zu je m-Bit Spalten, die Daten der booleschen Funktion, d. h. für jedes mögliche Argument am Eingang wird genau ein Funktionswert am Ausgang

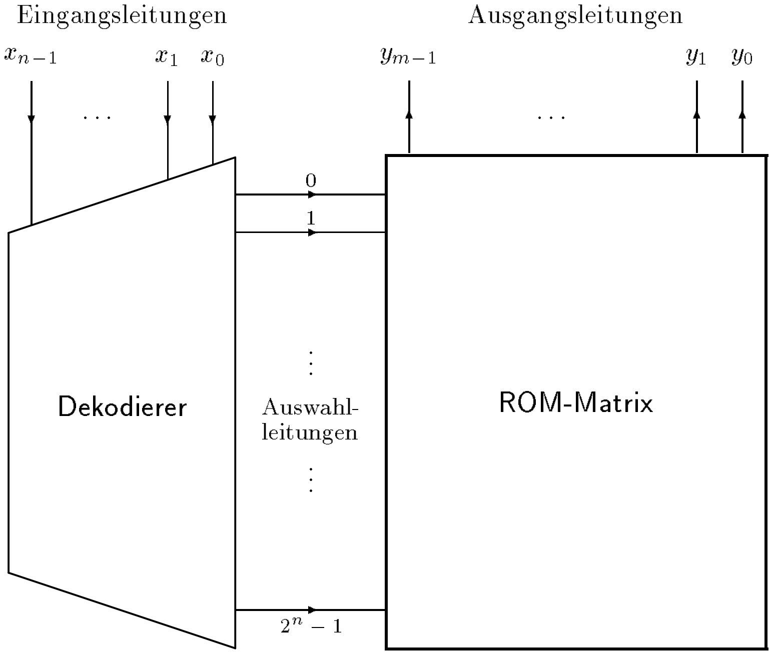

Abbildung 2.1: Interne Struktur eines ROM-Blockes

bereitgestellt. Aus n und m ergibt sich die maximale Anzahl der in der Matrix vorhandenen Knotenpunkte; sie soll im folgenden *Kapazität*[1] (engl. Capacity) genannt und durch $\mathsf{Cap}_{\mathrm{ROM}}$ bezeichnet werden:

$$\mathsf{Cap}_{\mathrm{ROM}} := 2^n \cdot m \tag{2.1}$$

2.1.2 Daten in der ROM-Matrix

Aus den binären Funktionswerten einer Wahrheitstabelle können direkt die Daten für eine ROM-Matrix gewonnen werden. Dieses sei anhand eines einfachen Beispiels veranschaulicht:

Sinus-Viertelwelle

Gegeben sei eine Folge von Werten, die aus der Berechnung[2] der ersten Viertelwelle einer Sinus-Schwingung hervorgeht; der besseren Übersicht wegen sei hier lediglich eine Folge aus 10 Werten betrachtet: 0, 2, 3, 5, 6, 8, 9, 9, 10 und 10. Nach Gleichung 1.1 (S. 5) ist

$$\begin{aligned} \mathbf{X} &:= \mathbf{I}_{10} := \{0, \ldots, 9\} \\ \mathbf{Y} &:= \{0, 2, 3, 5, 6, 8, 9, 10\} \end{aligned}$$

Zur Ermittlung der Anzahl der Bitstellen, die zur Binärdarstellung aller $\mathbf{x} \in \mathbf{X}$ und $\mathbf{y} \in \mathbf{Y}$ benötigt werden, sind nun zunächst die Maxima nach Gleichung 1.7 (S. 7) zu bestimmen: Es ergeben sich $N = 9$ und $M = 10$. Daraus folgt gemäß Gleichung 1.3 für beide Mengen die gleiche Anzahl an Bitstellen von $n = m = 4$ und es gilt: $\mathbf{X}, \mathbf{Y} \subseteq \mathbf{I}_{2^4} = \mathbf{I}_{16} := \{0, \ldots, 15\}$.

[1] Kapazität im Sinne von *Aufnahmefähigkeit*.

[2] Die Berechnung der Werte erfolgte durch Auswertung der Gleichung: $f(\mathbf{x}) := \sin(\frac{\mathbf{x}}{10} \cdot \frac{\pi}{2}) \in \mathbb{R}, \mathbf{x} \in \mathbf{X}$

Durch sukzessives Anwenden der Umwandlungsabbildung Bin aus Gleichung 1.2 (S. 6) erhält man für alle Paare $(\mathbf{x}, \mathbf{y})$ mit $\mathbf{x} \in \mathbf{X}$ und $\mathbf{y} \in \mathbf{Y}$ die Zeilen der gewünschten Wahrheitstabelle in Abbildung 2.2 (a) (S. 15) mit $x_3, \ldots, x_0$ und $y_3, \ldots, y_0$.

Die daraus abgeleitete ROM-Matrix ist in Abbildung 2.2 (b) wiedergegeben. Die beiden logischen Werte 0 und 1 sind in der ROM-Matrix durch die Zeichen ○ und ● entsprechend markiert.

(a) **Folge & Wahrheitstabelle**

Folge		Wahrheitstabelle							
$\mathbf{x}$	$\mathbf{y}$	x_3	x_2	x_1	x_0	y_3	y_2	y_1	y_0
0	0	0	0	0	0	0	0	0	0
1	2	0	0	0	1	0	0	1	0
2	3	0	0	1	0	0	0	1	1
3	5	0	0	1	1	0	1	0	1
4	6	0	1	0	0	0	1	1	0
5	8	0	1	0	1	1	0	0	0
6	9	0	1	1	0	1	0	0	1
7	9	0	1	1	1	1	0	0	1
8	10	1	0	0	0	1	0	1	0
9	10	1	0	0	1	1	0	1	0
10	-	1	0	1	0	*	*	*	*
11	-	1	0	1	1	*	*	*	*
12	-	1	1	0	0	*	*	*	*
13	-	1	1	0	1	*	*	*	*
14	-	1	1	1	0	*	*	*	*
15	-	1	1	1	1	*	*	*	*

(b) **ROM-Matrix**

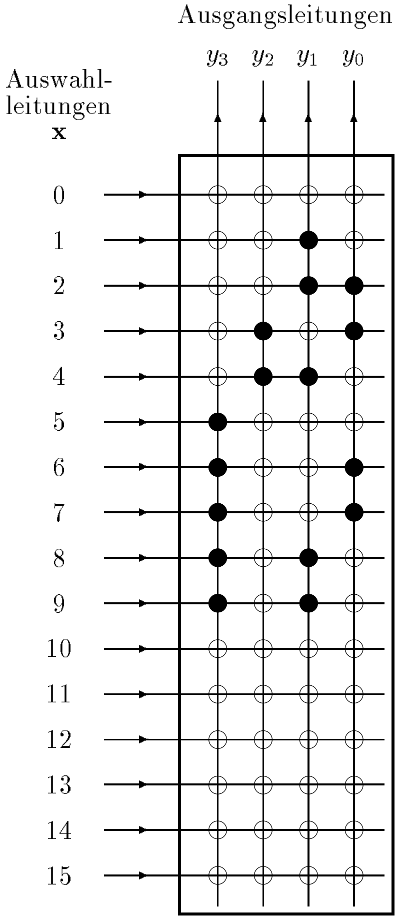

Abbildung 2.2: Beispiel für (a) Wahrheitstabelle und (b) ROM-Matrix

In der Wahrheitstabelle sind 16 1en enthalten; demzufolge sind in der ROM-Matrix nur an 16 von 64 Knotenpunkten mit Kodierungseinheiten besetzt. Setzt man die tatsächlich erreichte Anzahl an besetzten Knotenpunkten (Anzahl der 1en) mit der maximal möglichen Anzahl (Kapazität der ROM-Matrix $\mathsf{Cap}_{\mathrm{ROM}}$) ins Verhältnis, so erhält man den sogenannten *Besetzungsgrad* (engl. *Density*) der Matrix, der durch $\mathsf{Den}_{\mathrm{ROM}}$ bezeichnet werden soll. Im vorliegenden Beispiel entspricht dies einem Wert von 0.25.

$$\mathsf{Den}_{\mathrm{ROM}} := \frac{\#\ \mathtt{1}\text{en}}{\mathsf{Cap}_{\mathrm{ROM}}} \tag{2.2}$$

2.1.3 Knotenpunkte der ROM-Matrix

Zur Repräsentation der binären Daten sind zwei Schaltzustände nötig. Als Schaltglieder werden dabei Transistoren eingesetzt, die an den Knotenpunkten des Matrixgitters angeordnet sind. In Abbildung 2.3 ist das Prinzip-Schaltbild eines einzelnen Knotenpunktes in NMOS-Technik wiedergegeben.

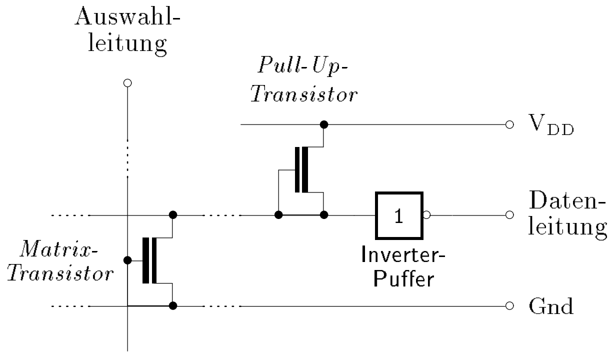

Abbildung 2.3: Prinzip eines Knotenpunktes in der ROM-Matrix

Die Datenleitungen werden durch *Pull-Up-Transistoren* hochohmig an das Potential der Versorgungsspannung V_{DD} gebunden. Auf diese Weise wird am Ausgang des Inverters erreicht, daß für einen unbesetzten Knotenpunkt zunächst der Wert logisch `0` auf der Datenleitung gilt. Für den dazu inversen Wert logisch `1` wird ein am betrachteten Knotenpunkt implementierter *Matrix-Transistor* die Datenleitung auf Massepotential schalten, sobald die betreffende Datenzelle durch ein hohes Potential auf der entsprechenden Auswahlleitung selektiert worden ist.

2.1.4 Kosten und Minimierung bei ROMs

Die Kosten der Realisierung einer ROM-Struktur liegen hauptsächlich im Flächenbedarf, den der ROM-Block im Layout eines Chips einnimmt. Er wird unmittelbar durch die Ausmaße der ROM-Matrix bestimmt; die Abmessungen des Dekodierers machen im allgemeinen nur einen Bruchteil dessen aus und fallen daher nicht ins Gewicht.

Zur Minimierung der Kosten sind einige Verfahren entwickelt worden, die zur Komprimierung der Matrix dienen: So kann man beispielsweise unbenutzte Spalten, also Datenleitungen, eliminieren oder Zeilen, also Adreßleitungen, vertauschen. Manchmal lassen sich individuelle Optimierungen finden, da sich zahlreiche Datensignale gegenseitig ausschließen.

2.1.5 Bewertung der ROM-Implementierung

2.1.5.1 Vorteile ROM

Der grundsätzliche Vorteil der ROMs ist ihre *reguläre* Struktur, die leicht zu modifizieren und zu verifizieren ist. Außerdem ermöglicht sie ein kompaktes Layout und gestattet eine einfache Plazierung des Blockes im Chip-Layout.

Die Layout-Maße bleiben zudem bei Änderungen an den in der Matrix abgelegten Daten konstant, da die eingebrachte Information in höchstens zwei Metallisierungsebenen liegt. Von den insgesamt etwa 15 benötigten Masken können die meisten daher wiederverwendet werden. Die eigentliche Struktur des ROM-Blockes wird dabei nicht verändert [Pos-89].

Der Zugriff auf die Daten erfolgt *wahlfrei* bzw. *direkt*, d. h. unabhängig von der Position einer Speicherzelle ist ein Zugriff auf die gleiche Weise und mit dem gleichen zeitlichen Aufwand möglich. Außerdem ist die Zugriffszeit leicht zu bestimmen, da lediglich der Dekodierer und die ROM-Matrix daraufhin untersucht werden müssen. Sie hängt lediglich von der verwendeten Technologie ab und ist in den meisten Fällen vernachlässigbar klein, so daß ROMs eine hohe Arbeitsgeschwindigkeit aufweisen.

2.1.5.2 Nachteile ROM

Bezüglich des Flächenbedarfs hat die ROM-Struktur einen entscheidenden Nachteil: An den insgesamt $\mathsf{Cap}_{\mathrm{ROM}}$ Knotenpunkten der Matrix ist jeweils der Platz für einen Schalttransistor vorzusehen, obwohl tatsächlich nicht alle Plätze für die Implementierung von Transistoren benötigt werden.

Außerdem wird häufig von Schaltungsproblemen nur ein Teil des Adreßraums von 2^n Matrixzeilen beansprucht. In vielen Fällen ist daher die `1`en-Anzahl wesentlich geringer als die der `0`en, so daß die ROM-Matrix mit tatsächlich eingebrachten Transistoren schwach besetzt ist; im Beispiel war nur ein Besetzungsgrad von 0.25 erzielt worden (vgl. Abschnitt 2.1.2, S. 14 und Abbildung 2.2, S. 15). Wegen der großen Redundanz in der Matrix kommt die ROM-Struktur nicht für die Implementierung von einfachen Logikfunktionen in Frage [Pos-89].

Ein weiterer Nachteil zeigt sich, sobald von seiten des Chip-Layouts eine kompakte und zudem quadratische Proportion wünschenswert erscheint. Der Einfluß des Dekodierers ist hierbei eher gering; am deutlichsten macht sich das Verhältnis der Matrix-Kantenlängen m und n bemerkbar (vgl. Gleichung 2.1, S. 14): Sobald die Anzahl der Ausgänge m klein gegen die Anzahl der Eingänge n ist, ergibt sich eine stark ungleichseitige Proportion des Blockes.

2.1.5.3 Bewertung ROM

Eine ROM-Implementierung ist daher bei komplexen Funktionen günstig, bei denen alle möglichen Kombinationen am Eingang auftreten können, wie z. B. Steuerprogramme für Controller oder (Mikro-)Programmcode etc. Die ROM-Struktur ist auch dann zu empfehlen, wenn die Daten bei konstantem Umfang häufigen Änderungen unterliegen.

2.2 Implementierung mit PLA

Die *<u>P</u>rogrammierbare <u>L</u>ogik-<u>A</u>nordnung* (engl. *<u>P</u>rogrammable <u>L</u>ogic <u>A</u>rray*, PLA) zählt zu den 2-stufigen Schaltnetzen, die eine direkte Übertragung von 2-stufigen Logikfunktionen darstellen und einfach in *diskreter Logik*, d. h. aus elementaren Logikgattern entworfen werden können.

Zur Reduzierung der Entwurfszeit werden allerdings unter der Verwendung von modernen Entwurfsmethoden, wie z. B. *Semi-Custom-* bzw. *Standard-Cell-Design*, Schaltfunktionen überwiegend mit Hilfe von sogenannten *Makrozellen* implementiert. Dabei handelt es sich um strukturierte Entwurfsbausteine, die einen hohen Grad an Regularität aufweisen, d. h. sie sind neben globalen Entwurfsparametern im wesentlichen durch ihre Funktionalität beschreibbar. Automatische Entwurfswerkzeuge, sogenannte *Modulgeneratoren*, sind aufgrund dieser Eigenschaft in der Lage, anhand der funktionalen Beschreibung einer Makrozelle deren strukturelle Daten bzw. das Layout zu erstellen.

Das PLA stellt nun, gegenüber den speziellen Schaltungsstrukturen aus diskreter Logik, eine reguläre Alternative zur Verfügung, die als Makrozelle im Entwurf mikroelektronischer Systeme mehr Flexibilität und Effizienz bietet. Der Ansatz beim PLA-Baustein besteht in einem matrixartigen Netz aus Logikgattern, welches aus zwei Ebenen (engl. *Planes*) besteht. Die Darstellung von Informationen geschieht durch An- bzw. Abwesenheit von Schaltgliedern bzw. Kontakten an den Knotenpunkten der Matrizen.

Die moderne Realisierung der PLA-Struktur in VLSI gestattet im eigentlichen Sinne keine *Programmierung*[3], d. h. die Funktionalität muß schon zum Entwurfszeitpunkt *fest* vorgegeben sein und das PLA wird bereits ohne die nicht benötigten Schaltglieder entworfen. Die Programmierbarkeit besteht also nur beim Entwurf des PLAs und kommt insofern der festen Verdrahtung bei ROMs oder diskreter Logik gleich.

2.2.1 PLA als funktionaler Block

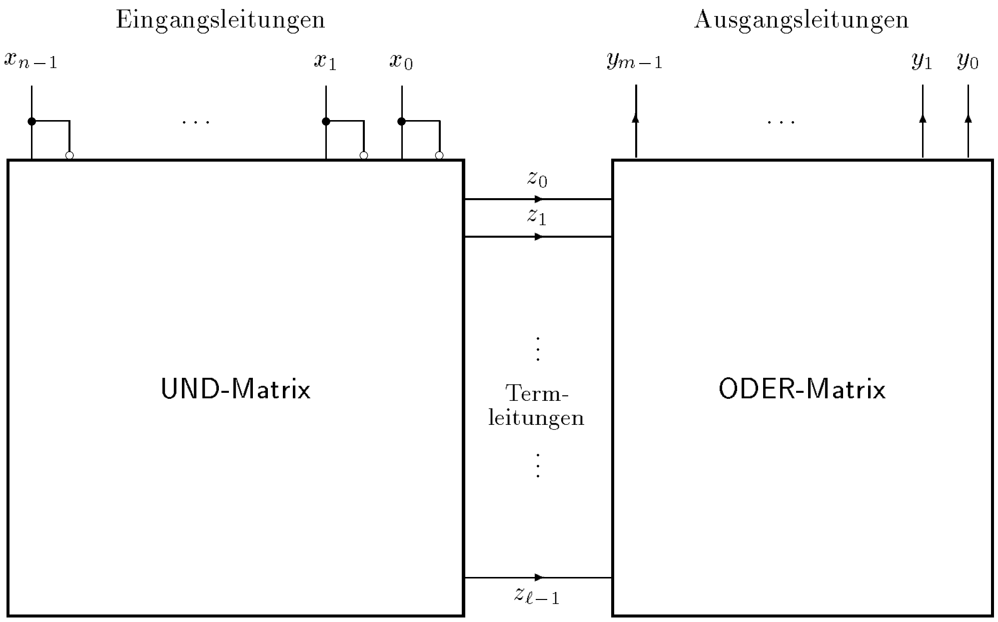

Abbildung 2.4: Interne Struktur eines PLA-Blockes

Ähnlich wie beim ROM wird die Funktionalität beim PLA auch in Matrixform implementiert: Den n-Bit Argumenten am Eingang werden mit Hilfe von zwei intern miteinander verbundenen Matrizen m-Bit Funktionswerte am Ausgang eindeutig zugeordnet. Abbildung 2.4 (S. 18) gibt

[3]**Historische Anmerkung:** Ursprünglich handelte es sich beim PLA um einen einzelnen IC-Baustein, der nach der Fertigung zunächst noch — in der Regel beim Hersteller — programmiert werden mußte. Auf diese Weise lag zunächst ein universeller Logikbaustein vor, der für alle möglichen Logikfunktionen programmierbar und dadurch spezialisierbar war: Es waren dazu zunächst alle Matrixknoten mit Schaltgliedern besetzt. Bei der anschließenden Programmierung wurden die nicht benötigten Schaltglieder irreversibel zerstört.

Seit Ansteigen der Integrationsdichte hat der separate PLA-Baustein seine Bedeutung fast völlig verloren. Insofern ist das Attribut *programmierbar* in der Namensgebung mittlerweile etwas irreführend. Im modernen VLSI-Entwurf zeigt sich allerdings die Eigenständigkeit dieser Struktur dadurch, daß das PLA wie auch das ROM als separate Makrozelle effizient eingesetzt wird.

den schematischen Aufbau eines PLA-Blockes in der Standardform wieder. In Anlehnung an die jeweils durchgeführten Verknüpfungen werden die Matrizen als UND- und ODER-Matrix bzw. -Ebene (engl. *Plane*) bezeichnet; in der englischen Literatur werden sie dementsprechend AND-/OR-Plane oder INPUT-/OUTPUT-Matrix genannt. Die letzte Bezeichnung ist allgemein günstig, da es weitere Varianten der PLA-Struktur gibt, die andere Verknüpfungen als UND/ODER in den Matrizen durchführen. Solche PLA-Varianten werden in Abschnitt 2.2.4 (S. 23ff) diskutiert.

2.2.1.1 UND-Matrix

Die n Eingangsvariablen $x_{n-1}, \ldots, x_0$ und deren Negationen $\overline{x}_{n-1}, \ldots, \overline{x}_0$ werden in der UND-Matrix in beliebigen Kombinationen zu ℓ verschiedenen Konjunktionen verknüpft. Die so gebildeten Ausdrücke werden *Terme* oder *Implikanten* genannt, die darin vorkommenden Variablen *Literale*.

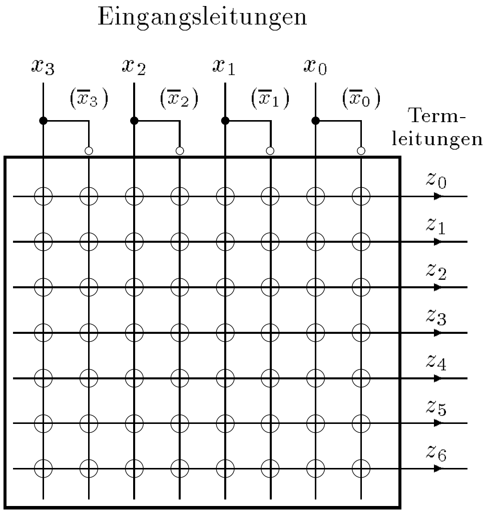

Abbildung 2.5: Struktur einer UND-Matrix

Abbildung 2.5 zeigt eine UND-Matrix für $n = 4$ und $\ell = 7$. Die Anzahl der zur Verfügung stehenden Knotenpunkte, die Kapazität der UND-Matrix ergibt sich aus der Anzahl der Eingangsvariablen n und der Anzahl der Terme ℓ zu:

$$\mathsf{Cap}_{\mathrm{AND}} := 2n \cdot \ell \tag{2.3}$$

Im vorliegenden Fall stehen also $\mathsf{Cap}_{\mathrm{AND}} := 56 = 2 \cdot 4 \cdot 7$ Knotenpunkte in der Matrix zur Verfügung. Bei der Belegung mit Schaltgliedern muß allerdings beachtet werden, daß in einer Matrixzeile offenbar nicht alle Knotenpunkte tatsächlich belegt werden können: In einer Termleitung kann bei einer Variablen x_i nie auch die Position seines Komplementes $\overline{x}_i$ besetzt sein. Jene Termleitung würde dadurch konstant den Pegel für logisch 0 führen.

Das bedeutet, daß für jede der Termleitungen maximal nur die Hälfte der zur Verfügung stehenden Kontenpunkte auch tatsächlich belegt werden kann. Für den Besetzungsgrad der UND-Matrix hat dieses ein Maximum von 0.5 zur Folge [Bac-84, S. 13]; es gilt somit folgende Definition und Abschätzung, wobei die Anzahl der Literale gleich der Anzahl der belegten Knotenpunkte ist:

$$\mathsf{Den}_{\mathrm{AND}} := \frac{\#\ \text{AND-Literals}}{\mathsf{Cap}_{\mathrm{AND}}} < 0.5 \tag{2.4}$$

Andererseits muß in jeder der ℓ Termleitungen zumindest ein Knotenpunkt besetzt sein; ansonsten würde sie nur konstant den Pegel für logisch `1` haben und wäre damit entbehrlich. Der Besetzungsgrad der Und-Matrix ist dadurch nach unten beschränkt:

$$\mathsf{Den}_{\mathrm{AND}} \geq \frac{\ell}{\mathsf{Cap}_{\mathrm{AND}}} \tag{2.5}$$

2.2.1.2 ODER-Matrix

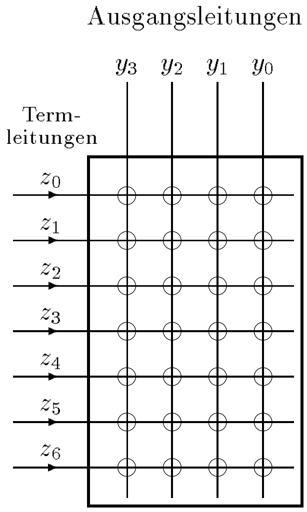

Abbildung 2.6: Struktur einer Oder-Matrix

Abbildung 2.6 (S. 20) zeigt eine Oder-Matrix für $m = 4$ und $\ell = 7$. Die ℓ PLA-Terme werden über die internen Termleitungen $z_0, \ldots, z_{\ell-1}$ an die Oder-Matrix weitergeleitet. Sie verknüpft daraus in beliebigen Kombinationen disjunktiv die m Ausgangssignale $y_{m-1}, \ldots, y_0$. Es ergibt sich die Kapazität der Oder-Matrix aus der Anzahl der Ausgangsvariablen m und der Anzahl der Terme ℓ zu:

$$\mathsf{Cap}_{\mathrm{OR}} := m \cdot \ell \tag{2.6}$$

Im konkreten Fall stehen also $\mathsf{Cap}_{\mathrm{OR}} := 28 = 4 \cdot 7$ Knotenpunkte in der Matrix zur Verfügung. Bei der Besetzung der Oder-Matrix gibt es bezüglich einzelner Ausgangsleitungen keine Einschränkungen [Bac-84, S. 13]. Allerdings werden nie alle Knotenpunkte besetzt sein, da dies m identische Ausgangssignale bedeuten würde. Insofern bleibt der nachfolgend definierte Besetzungsgrad der Oder-Matrix immer kleiner als 1.0:

$$\mathsf{Den}_{\mathrm{OR}} := \frac{\#\ \text{OR-Literals}}{\mathsf{Cap}_{\mathrm{OR}}} < 1.0 \tag{2.7}$$

Ebenso wie bei der UND-Matrix gilt auch für die ODER-Matrix, daß auf jeder Ausgangsleitung mindestens ein Knotenpunkt belegt ist; insofern existiert die untere Schranke:

$$\mathsf{Den}_{\mathrm{OR}} \geq \frac{m}{\mathsf{Cap}_{\mathrm{OR}}} \tag{2.8}$$

Ein Vergleich der Besetzungsgrade von UND- und ODER-Matrix ergibt im allgemeinen eine dichtere Besetzung in der UND-Matrix [Suw-82, S. 63]. Insgesamt ergibt sich damit die Gesamtkapazität des PLAs aus beiden Matrizen mit n, m und ℓ wie folgt:

$$\begin{aligned} \mathsf{Cap}_{\mathrm{PLA}} &:= \mathsf{Cap}_{\mathrm{AND}} + \mathsf{Cap}_{\mathrm{OR}} \\ &:= (m + 2n) \cdot \ell \end{aligned} \tag{2.9}$$

Der Besetzungsgrad ergibt sich aus der Summe der einzelnen Besetzungsgrade $\mathsf{Den}_{\mathrm{AND}}$ und $\mathsf{Den}_{\mathrm{OR}}$:

$$\begin{aligned} \mathsf{Den}_{\mathrm{PLA}} &:= \mathsf{Den}_{\mathrm{AND}} + \mathsf{Den}_{\mathrm{OR}} \\ &:= \frac{\#\text{ AND-Literals} + \#\text{ OR-Literals}}{\mathsf{Cap}_{\mathrm{AND}} + \mathsf{Cap}_{\mathrm{OR}}} \end{aligned} \tag{2.10}$$

2.2.2 Belegung der PLA-Matrizen

Das Beispiel der Sinus-Viertelwelle aus Abschnitt 2.1.2 (S. 14) bietet sich zur Veranschaulichung der Belegung der PLA-Matrizen an: Zunächst müssen die booleschen Gleichungen des Beispiels aufgestellt werden. Dieses geschieht beispielsweise durch Aufstellen der DNF (vgl. Abschnitt 1.2.3.2, S. 9) aus der Wahrheitstabelle und eine anschließende Minimierung der Termmenge. Abbildung 2.7 zeigt das Ergebnis im Kasten *Funktionsgleichungen*.

Funktionsgleichungen

$$\begin{aligned} y_0 &:= x_2 x_1 + \overline{x}_2 x_1 x_0 + \overline{x}_2 x_1 \overline{x}_0 \\ y_1 &:= x_3 + x_2 \overline{x}_1 \overline{x}_0 + \overline{x}_2 x_1 \overline{x}_0 + \overline{x}_2 \overline{x}_1 x_0 \\ y_2 &:= \overline{x}_2 x_1 x_0 + x_2 \overline{x}_1 \overline{x}_0 \\ y_3 &:= x_3 + x_2 x_1 + x_2 x_0 \end{aligned}$$

PLA-Terme (UND-Matrix)

$$\begin{aligned} z_0 &:= \overline{x}_2 \overline{x}_1 x_0 & z_4 &:= x_2 x_0 \\ z_1 &:= \overline{x}_2 x_1 \overline{x}_0 & z_5 &:= x_2 x_1 \\ z_2 &:= x_2 \overline{x}_1 \overline{x}_0 & z_6 &:= x_3 \\ z_3 &:= \overline{x}_2 x_1 x_0 \end{aligned}$$

PLA-Ausgänge (ODER-Matrix)

$$\begin{aligned} y_0 &:= z_5 + z_3 + z_1 \\ y_1 &:= z_6 + z_2 + z_1 + z_0 \\ y_2 &:= z_3 + z_2 \\ y_3 &:= z_6 + z_5 + z_4 \end{aligned}$$

Abbildung 2.7: PLA-Gleichungen zum *Sinus*-Beispiel

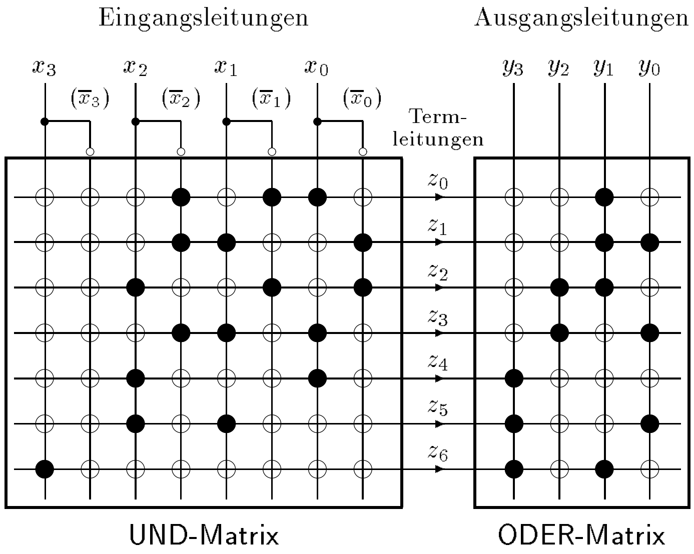

Abbildung 2.8: Beispiel zur Besetzung der PLA-Matrizen

Der Kasten *PLA-Terme* gibt die UND-Gleichungen zur Erzeugung der internen Termsignale $z_0, \ldots, z_6$ wieder. Die dadurch vereinfachten Gleichungen der Ausgangsleitungen sind rechts daneben im Kasten *PLA-Ausgänge* aufgeführt. Mit Hilfe dieser Gleichungen läßt sich nun die Belegung der PLA-Matrizen direkt bestimmen; sie ist in Abbildung 2.8 (S. 22) wiedergegeben.

In der UND-Matrix werden hier 17 von 56 Knotenpunkte besetzt, das entspricht einem Besetzungsgrad von etwa $\mathsf{Den}_{\mathrm{AND}} = 0.30$. Für die ODER-Matrix ergibt sich folgende Bilanz: Von 28 Knotenpunkten sind 12 besetzt, was einen Besetzungsgrad von etwa $\mathsf{Den}_{\mathrm{OR}} = 0.43$ entspricht. Insgesamt hat das PLA damit 29 von 84 Knotenpunkte besetzt und damit einen Besetzungsgrad von insgesamt fast 0.35. Obwohl sich im Vergleich zur ROM-Implementierung für dieses Beispiel mit $\mathsf{Cap}_{\mathrm{PLA}} = 84$ mehr Knotenpunkte ergeben haben als beim ROM mit $\mathsf{Cap}_{\mathrm{ROM}} = 64$, wird eine Verbesserung im Besetzungsgrad von $\mathsf{Den}_{\mathrm{ROM}} = 0.25$ auf $\mathsf{Den}_{\mathrm{PLA}} = 0.35$ erreicht.

2.2.3 Eigenschaften und Bewertung

PLAs sind wie ROMs Festwertspeicher und eignen sich zur Bereitstellung von binär-kodierten Daten jeglicher Art, d. h. es kommen sowohl Logik- als auch Numerik-Daten dafür in Frage. Im Gegensatz zu ROMs braucht jedoch bei PLAs nicht jedem m-Bit Speicherwort in eindeutiger Weise eine Adresse zugeordnet zu sein, und zwischen Adreßlänge n und der „Speicher"-Kapazität des PLAs besteht keine Abhängigkeit.

Bezüglich der Speicherterminologie entspricht der Mechanismus des PLAs dem eines Assoziativspeichers [Gil*80]: Der Zugriff auf Daten einer Speicherzelle geschieht nicht durch eine spezifizierte Adresse, sondern auf alle Speicherzellen gleichzeitig, die dazu eine vorgegebene Inhaltsspezifikation erfüllen. Der Zugriff erfolgt wie beim ROM *wahlfrei* bzw. *direkt*, d. h. bezogen auf die Zugriffszeit wird für alle Daten die gleiche Dauer beansprucht, da der Zugriff unabhängig von Matrixpositionen immer auf die gleiche Weise erfolgt.

Gegenüber dem ROM erlaubt die Repräsentation der logischen Funktionalität bei der PLA-Struktur innerhalb zweier Matrizen eine Verdichtung der Information. Weiter sind neben diesen vergleichenden Aussagen noch die nachfolgend diskutierten Vor- und Nachteile der ROM-Implementierung zu berücksichtigen.

2.2.3.1 Vorteile PLA

Beim PLA handelt es sich durch die Regularität der Struktur um ein einfach zu generierendes Entwurfselement. Es liegt in einer vom Hersteller optimierten Form in Makrozellen-Bibliotheken vor. Die kompakten Abmessungen erleichtern die Plazierung im Chip-Layout. Außerdem ist die Realisierung mit PLAs bei schwach besetzten Tabellen effizienter als beim ROM.

Im Zuge der mikroelektronischen Entwicklung hat die PLA-Struktur zur Senkung der Kosten für die Realisierung von Logikfunktionen maßgeblich beigetragen. Der vorteilhafte Einsatz von PLAs führte zu zahlreichen Forschungsaktivitäten, so daß das PLA heutzutage als gut erforscht gilt.

2.2.3.2 Nachteile PLA

Anders als beim ROM bewirken Änderungen in der Funktionalität beim PLA auch strukturelle Änderungen. Beispielsweise können sich dadurch sogar die Abmessungen des PLA-Blockes ändern, so daß eventuell viele andere Entwurfsschritte wiederholt werden müssen. Im ungünstigsten Fall kann dies eine Neu-Plazierung anderer Funktionsblöcke erforderlich machen.

In der Entwurfspraxis treten nicht selten Größenordnungen der benötigten PLAs auf, die eine hohe statische Stromaufnahme verursachen. Teilweise ist die Realisierung aufgrund überschrittener Grenzwerte gar nicht mehr möglich bzw. die erzielten Werte sind im Licht der allgemeinen Bestrebung zur Senkung des Energiebedarfes nicht akzeptabel.

2.2.3.3 Bewertung PLA

PLAs eignen sich vor allem gut zur Realisierung von Funktionen, deren Berechnung kompliziert ist bzw. die in ihrer Definition wenig „Regularität“ aufweisen. Ebenso günstig sind sie für partielle Funktionen, die für viele Belegungen am Eingang nicht definiert sind.

PLAs und ROMs treten in Konkurrenz zueinander, wenn die Datenmenge eine kritische Größe überschreitet: Bei ROMs kann ein zu geringer Besetzungsgrad störend sein, oder aber der Flächenbedarf im Layout ist unakzeptabel hoch. Bei PLAs können hingegen die Grenzen für maximale Termanzahl bzw. zulässige Stromaufnahme leicht überschritten werden.

2.2.4 Varianten zur Realisierung

Die interne Struktur eines PLA-Blockes, wie sie in den Abbildungen 2.4, 2.5, 2.6 und 2.8 auf symbolischer Ebene dargestellt wird, ist nicht an eine bestimmte Technologie oder bestimmte physikalische Rahmenbedingungen gebunden. Allerdings wird die topologische Anordnung der Komponenten im Layout bereits durch den symbolischen Aufbau definiert, d. h. die Zeilen und Spalten der Matrizen entsprechen direkt horizontalen und vertikalen Elementen auf den Fertigungsmasken [Suw-82, S. 2]. Ebenso entsprechen die besetzten Knotenpunkte den eingebrachten Verbindungen bzw. der Anwesenheit von aktiven Schaltgliedern.

Mit der Zielsetzung einer besseren Flächenausnutzung und einer kurzen Entwicklungszeit mit standardisierten Strukturen sind in der Vergangenheit eine Reihe von strukturellen Realisierungsvarianten vorgestellt worden.

2.2.4.1 Strukturelle Varianten

Die PLA-Struktur in ihrer Standardform mit einer Und- und einer Oder-Matrix gestattet die Implementierung von Logikfunktionen in Gestalt der DNF auf geradezu mechanische Weise. Dieses ist die Basis für die einfache Automatisierung des PLA-Entwurfes. Allerdings geht dieser Vorteil zu Lasten der Layout-Fläche, da beim Ansteigen der Matrizen-Kapazitäten der Besetzungsgrad in der Regel abnimmt [Pos-89]. Die bisherigen Aktivitäten der PLA-bezogenen Forschung gingen deshalb zum einen in Richtung der Flächen-Minimierung, zum anderen in Richtung einer Verbesserung der Entwurfsautomatisierung.

Bei einer Beschränkung auf das Standard-PLA lassen sich Einsparungen in der Fläche nur über die Minimierung der Logikfunktionen erreichen. Dieser Umstand erklärt die teilweise sehr engagierten Bemühungen der bisherigen Forschung in diesem Sektor. Einige strukturelle Abwandlungen von der Standardform des PLAs haben die Steigerung des Besetzungsgrades in den Matrizenzum Ziel. Allerdings werden die Strukturen dadurch meistens noch spezieller auf die jeweilige Funktionalität abgestimmt, so daß diesbezügliche Änderungen zu Lasten der Flexibilität gehen. Eine erste strukturelle Variante bietet das *Dekodierte PLA*:

Dekodiertes PLA

Der Ansatz für die Einführung von Dekodierblöcken innerhalb von PLAs besteht darin, daß sich bei einem geringeren Besetzungsgrad der Matrizen eine Partitionierung in mehrere kleinere Matrizen anbietet. Im allgemeinen dominiert beim Besetzungsgrad die Und-Matrix [Suw-82, S. 63] (vgl. Abschnitt 2.2.1.2, S. 21). Daher werden vor der Und-Matrix Dekodierstufen geschaltet, die eine „Ausblendung“ unbenutzter Teilbereiche der Matrizen erreichen [Pos-89].

Dieses wird durch geschickte Gruppierung der Eingangsvariablen bewirkt, die den Dekodierern zugeführt werden [Fle*75, Sas*79]. Auf diese Weise gelangen weiterhin alle Belegungen der Eingangsvariablen zur Und-Matrix, aber nur die benötigten werden in die Matrix hineingeführt. Während beim Standard-PLA einfache Eingangsvariablen in der Und-Matrix konjunktiv miteinander verknüpft werden, geschehen beim dekodierten PLA nunmehr Disjunktionen von Mintermen.

Die generelle Schwierigkeit liegt in einer günstigen Gruppierung der Eingangsvariablen, um damit die gewünschte „Ausblendung“ zu erzielen. Generell ist das Optimum bei einer großen Anzahl n von Eingangsvariablen schwer zu finden. Deshalb werden meistens nur 2-Bit Dekodierer implementiert. In [Sas-83] sind Ergebnisse für nahezu optimale Eingangszuweisungen für die Dekodierer beschrieben.

Der Effekt der eingangsseitigen Dekodierung liegt darin, daß weniger Termleitungen erforderlich sind und daher weniger Flächenbedarf besteht. Obwohl die zusätzlich eingebrachten Dekodierer etwas Mehrbedarf an Fläche bedeuten und eine zusätzliche Signalverzögerung verursachen (die Inverterstufen bei der Standardform sind allerdings nicht mehr nötig), reduziert sich die Gesamtfläche gegenüber dem Standard-PLA.

Allerdings ist eine weitere Minimierung wesentlich komplexer als bei Standard-PLAs. Es werden dafür heuristisch arbeitende Werkzeuge wie z. B. Mini [Hon*74], McBoole [Dag*85, Dag*86] und Espresso [Bra*82b, Bra*84] verwendet.

Das Standard-PLA kann als Spezialfall eines dekodierten PLAs aufgefaßt werden, bei dem alle Dekodierer nur jeweils einen Eingang aufweisen [Hon*84]. Über den beschriebenen Effekt der „Ausblendung“ unbenutzter Matrixbereiche hinausgehend, verhindert dieser Ansatz nicht das Grundproblem des geringen Besetzungsgrades innerhalb der kleineren Matrizen. Untersuchungen zum Besetzungsgrad in PLA-Matrizen haben gezeigt, daß bei typischen (Standard-)PLAs maximal ein Zehntel der Knotenpunkte nicht durch `0`en und `*` (Don't-Care) besetzt werden [Hac*80]. Der Ansatz der *PLA-Faltung* nutzt nun gerade die Verstreutheit der besetzten Knotenpunkte in den PLA-Matrizen aus und stellt in seiner Komplexität eine neue Dimension für den topologischen Entwurf von PLAs dar [Suw-82].

Gefaltetes PLA

Das Ziel bei der PLA-Faltung liegt in einer kompakten, möglichst quadratischen Anordnung innerhalb der PLA-Matrizen. Sie können in beiden Dimensionen, also spalten- bzw. zeilenorientiert, gefaltet werden, wobei sich je Faltungsdimension maximal eine Halbierung der Fläche ergeben kann.

Der Ansatz der Faltung unterscheidet bei einer Matrix zwischen symbolischer Struktur und physikalischer Anordnung. Damit ist es häufig möglich, durch geschickte Kombination von *zwei* symbolischen Matrixreihen, beide in nur *einer* physikalischen Matrixreihe aufzunehmen. Die beiden symbolischen Reihen müssen disjunkt sein, d. h. in ihnen kommen Variablen nicht identisch vor. Durch eine entsprechende Auftrennung in der physikalischen Reihe bleiben sie voneinander auch schaltungstechnisch unabhängig.

Bei der PLA-Faltung gibt es verschiedene Varianten; bezogen auf die in Abbildung 2.8 (S. 22) gezeigte Anordnung der PLA-Matrizen bedeutet eine

- **Spalten-Faltung** in der Oder-Matrix eine Teilung der Und-Matrix. Durch geschicktes Umordnen der Termleitungen z_k und Falten disjunkter Oder-Spalten wird die Oder-Matrix schmaler, und die PLA-Ausgangsleitungen y_j liegen sich an der Ober- und Unterseite gegenüber.
- Die **Zeilen-Faltung** bewirkt die andere grundsätzliche Variante. Wieder wird durch geeignetes Umsortieren von PLA-Spalten x_i und y_j die Voraussetzung zur Faltung der Termleitungen z_k geschaffen. Da diese dadurch nicht mehr nur nach rechts, sondern auch nach links aus der Und-Matrix herausführen, erfolgt die Aufspaltung der Oder-Matrix in eine linke und eine rechte Teilmatrix. Es entsteht damit eine Aneinanderreihung der Matrizen in Oder-Und-Oder-Form.
- Die **PLA-Segmentierung** stellt eine Kaskadierung der Zeilen-Faltung dar, bei der in einer physikalischen Zeile mehr als nur zwei symbolische Matrixzeilen untergebracht sind. Als Resultat ergibt sich hierdurch eine Aneinanderreihung von Oder-Und-Gruppen [Suw-82].

Das Mischen der ersten beiden Formen, also die Kombination von Spalten- und Zeilen-Faltung, ist sehr effektiv, erfordern allerdings aufwendige Algorithmen [Ega*82]. Branch-and-Bound Methoden der PLA-Faltung werden beispielsweise in [Lew*84, Gra-82] beschrieben.

Die Anwendung der PLA-Faltung ist eine komplexe Problematik, zeigt aber bei der Einsparung von Chipfläche eine große Wirkung. In [Hac*80, Hac*82] sind optimale Algorithmen beschrieben, die sich auch für die Automatisierung eignen.

2.2.4.2 Schaltungstechnische Varianten

Prinzipiell ist die Realisierung einer PLA-Struktur von der Wahl der verwendeten Technologie unabhängig. Trotzdem gibt es Variationen der Grundstruktur, die auf technologische Aspekte Rücksicht nehmen. Während die Und-/Oder-Form, bei der zwei Gattertypen zum Einsatz kommen, gewissermaßen die Basisversion einer PLA-Realisierung darstellt, sind isomorphe Formen aus technologischer Sicht in der Realisierung effizienter. Durch entsprechende Umformungen können die PLA-Gleichungen beispielsweise in die für Nor-/Nor- bzw. Nand-/Nand-Matrizen gerechte Form gebracht werden.

NOR-/NOR-PLA

Zur Repräsentation der binären Daten sind zwei Schaltzustände nötig. Als Schaltglieder werden Transistoren eingesetzt, die an den Knotenpunkten des Matrixgitters angeordnet sind. Das Prinzip-Schaltbild für eine Nmos-Realisierung in Nor-/Nor-Form zeigt Abbildung 2.9. Die Beschreibung der Arbeitsweise ist analog zu den Erläuterungen bei den ROM-Knoten-

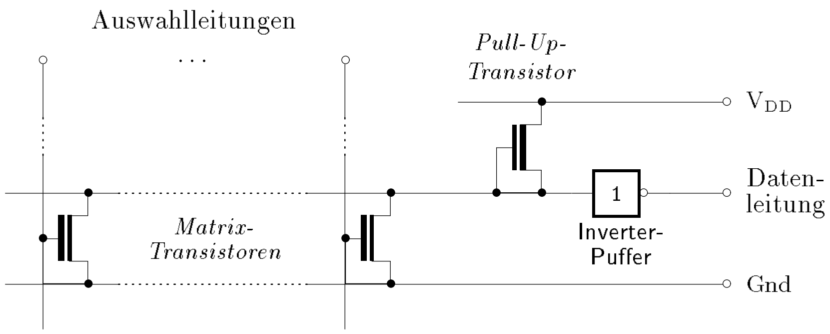

Abbildung 2.9: Prinzip der Knotenpunkte in einem Nor-/Nor-PLA

punkten (vgl. Abschnitt 2.1.3, S. 16): Die Datenleitungen werden durch *Pull-Up-Transistoren* hochohmig an das Potential der Versorgungsspannung V_{DD} gebunden. Auf diese Weise wird am Ausgang des Inverters erreicht, daß für einen unbesetzten Knotenpunkt zunächst der Wert logisch `0` auf der Datenleitung gilt.

Für den logischen Wert `1` wird ein am betrachteten Knotenpunkt implementierter *Matrix-Transistor* die Datenleitung auf Massepotential schalten, sobald die betreffende Datenzelle durch ein hohes Potential auf der entsprechenden Auswahlleitung selektiert worden ist.

NAND-/NAND-PLA

Das Prinzip-Schaltbild zur Nmos-Realisierung der Nand-/Nand-Form zeigt Abbildung 2.10. Für den logischen Wert `1` wird am betrachteten Knotenpunkt ein *Matrix-Transistor* in die

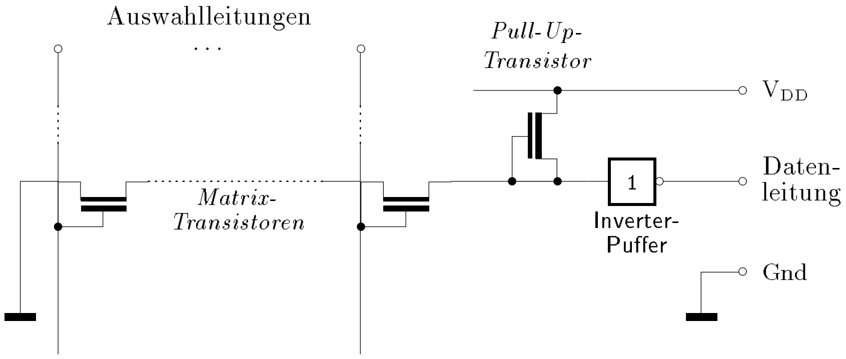

Abbildung 2.10: Prinzip der Knotenpunkte in einem Nand-/Nand-PLA

Reihenschaltung der Datenleitung eingefügt und schaltet das links anstehende Massepotential durch, sobald die betreffende Datenzelle durch ein hohes Potential auf der entsprechenden Auswahlleitung selektiert worden ist.

Der Vergleich zwischen den beiden Strukturen ergibt, daß die Nand-/Nand-Struktur eine kompaktere Verdrahtung als die Nor-/Nor-Struktur ermöglicht; es muß beispielsweise kein Massepotential in die Matrix hineingeführt werden.

Dagegen ist generell bei Strukturen aus Nand-Gattern die Geometrieabhängigkeit des Schalttransistors nachteilig: seine Weite hängt zur Gewährleistung des richtigen Verhältnisses zum Widerstand des Pull-Up-Transistors von der Anzahl der Gattereingänge ab [Pos-89]. Aus diesem Grunde vermeidet man in Nmos generell Nand-Gatter mit mehr als drei Eingängen. Für PLA-Realisierungen bedeutet dies folglich, daß bei größeren Matrixfeldern immer die Nor-Version vorzuziehen ist.

Die gezeigten Schaltbilder geben die Situation für Nmos-Technologie wieder. Bei der Cmos-Technologie sind voll-komplementäre Schaltungen für PLA-Strukturen zu aufwendig, so daß stattdessen auf Pseudo-Nmos zurückgegriffen wird: Beispielsweise müssen für ein Nor-Gatter mit k Eingängen in (Pseudo-)Nmos-Technik $k+1$ Transistoren vorgesehen werden; bei Cmos-Technik dagegen wären es $2 \cdot k$.

2.2.4.3 Funktionelle Variante

Neben den bereits diskutierten Möglichkeiten von strukturellen und schaltungstechnischen Varianten bei der PLA-Implementierung soll noch auf die funktionelle Variante eingegangen werden. Sie wird durch die Realisierung der komplementären Funktionen $\overline{f}$ angeboten (vgl. Abschnitt 1.2.3.3, S. 10). Dieses kann vorteilhaft sein, wenn bei den zu realisierenden Logikfunktionen die Anzahl der Minterme (Einsterme) größer als die der Maxterme (Nullterme) ist. Gegenüber einer Realisierung der nicht-komplementären Funktion f läßt sich in solchen Fällen der Aufwand reduzieren. Ein erster Hinweis auf diese Situation bestünde darin, wenn bei einer hypothetischen ROM-Implementierung der Besetzungsgrad (vgl. Gleichung 2.2, S. 15) größer als 0.5 werden würde, d. h. wenn folgende Abschätzung erfüllt wäre:

$$\begin{array}{rrcl} & \#\,\mathtt{1}\text{en} & > & 0.5 \cdot \mathsf{Cap}_{\text{ROM}} \\ \Longleftrightarrow & \#\,\mathtt{1}\text{en} & > & 2^{-1} \cdot 2^{n} \cdot m \\ \Longleftrightarrow & \#\,\mathtt{1}\text{en} & > & 2^{n-1} \cdot m \end{array} \tag{2.11}$$

Da ausgangsseitig bei einer PLA-Implementierung signalverstärkende Pufferstufen generell notwendig sind, kann davon ausgegangen werden, daß jene gegenüber der Standardform des PLAs auch invertierend ausgelegt werden können. Auf diese Weise ist durch die komplementäre Funktion $\overline{f}$ die nicht negierte Funktion f zu realisieren.

2.2.5 Kostenkriterien

Im folgenden werden Kriterien diskutiert, die zur Einschätzung des Realisierungsaufwandes geeignet erscheinen. Dabei sollen die Beziehungen zwischen den einzelnen Kriterien aufgezeigt und bewertet werden.

2.2.5.1 Anzahl der Terme

Die Größe einer PLA-Struktur wird in erster Linie vom Auftraggeber selbst bestimmt, da bereits die zu realisierende Funktionalität die Wertebereiche und damit das Format (vgl. Gleichungen 1.1 u. 1.8, S. 7) des PLAs festlegt.

Für die n Eingangs- und m Ausgangsvariablen wird eine Wahrheitstabelle aufgestellt, aus der sich die Minterme der DNF (vgl. Abschnitt 1.2.3.2, S. 9) entnehmen lassen. Da für jede `1` in einer der m Funktionsspalten ein solcher Minterm gebildet werden muß, ist die `1`en-Anzahl eine obere Schranke für die Kardinalität ℓ der Termmenge; die `1`en-Anzahl sei im folgenden durch $\ell_{\mathtt{1}}$ bezeichnet. Es gilt somit die Abschätzung:

$$\ell \leq \ell_{\mathtt{1}} \tag{2.12}$$

Mit $\ell_{\mathtt{1}}$ als obere Schranke für die Anzahl ℓ der Terme existiert also ein erster Anhaltspunkt für die entstehenden Realisierungskosten.

Da im allgemeinen für die Minimierung von Termmengen keine eindeutige Lösung existiert, liefern verschiedene Verfahren[4] unterschiedlich „gute“ Resultate ℓ_j ℓ. Unter der Annahme eines Optimums ℓ_{opt} und mehrerer Resultate gilt folgende Beziehung:

$$\ell_{\text{opt}} \leq \ell \leq \ell_1 \tag{2.13}$$

Für jeden Term in der Termmenge wird u. a. eine interne Termleitung z_i, wobei $i \in \{0, \ldots, \ell - 1\}$ im PLA[5] benötigt. Da dies unmittelbar Hardwareaufwand bedeutet, sollte die Anzahl der Terme möglichst gering sein. Sie hängt insofern von der Güte der Logik-Minimierung ab.

2.2.5.2 Anzahl der Literale

Die in einem booleschen Ausdruck zu verknüpfenden Variablen werden als *Literale* bezeichnet. In Bezug auf PLAs ist die Anzahl der Literals gleichbedeutend mit der Anzahl der besetzten Knotenpunkte in den Matrizen. Sie sind allerdings von der Termanzahl ℓ kausal abhängig, da die besetzten Knotenpunkte auf den internen Verbindungsleitungen liegen, die die Terme realisieren.

Insofern existiert für die Anzahl der Literale mit der Kapazität der Matrizen (vgl. Gleichung 2.9, S. 21) eine durch die Termanzahl vorgegebene Beschränkung. Die tatsächliche Anzahl der Literale kann deshalb ins Verhältnis mit ihrem Maximum gebracht werden und wird auf diese Weise zur Bestimmung des Besetzungsgrades der Matrizen herangezogen (vgl. Gleichung 2.10, S. 21).

Man erhält damit für die PLA-Struktur einen Beurteilungsparameter, der bei Vergleichen mit anderen Realisierungsformen, wie z. B. der ROM-Struktur, hilfreich sein kann. Trotzdem konzentrieren sich die Bemühungen um geringe Realisierungskosten weniger auf die Minimierung der Literalanzahl. Sie erlangt erst dann eine Bedeutung, wenn in Vergleichen die Termanzahl einander nahe kommen bzw. übereinstimmen.

2.2.5.3 Kapazität der PLA-Matrix

Die Kapazität der PLA-Matrizen $\mathsf{Cap}_{\text{PLA}}(\ell, m, n)$ ist durch die Formatparameter n und m sowie durch die Termanzahl ℓ bestimmt (vgl. Gleichung 2.9, S. 21). Versteht man die Formatparameter in diesem Kontext zunächst als fest gegebene Entwurfsgrößen, so handelt es sich bei der PLA-Kapazität um eine lineare Funktion in Abhängigkeit von der Termanzahl. Daher sollte die Reduzierung des Realisierungsaufwandes auch bezüglich dieses Kriteriums durch entsprechende Bemühungen bei der Termanzahl erfolgen.

Sofern allerdings die Formatparameter n und m auch als veränderbar interpretiert werden, zeigt sich an diesem Kostenkriterium, daß durch eine Senkung der beiden Parameter (vorzugsweise bei der Anzahl n der Eingänge) leicht eine Reduzierung bei den Kosten erzielt werden kann. Ändern des Formats ist beispielsweise dann anzuraten, wenn in der Wahrheitstabelle identische Spalten vorkommen oder gar Spalten unbenutzt sind, d. h. lediglich aus 0en bestehen.

[4] Als *de facto* Standard wird der Logik-Minimierer Espresso angesehen [Bra*82b, Bra*84].

[5] Die PLA-Struktur vermeidet hierdurch bereits Redundanz, da identische Terme nicht mehrfach gebildet werden.

2.2.5.4 Flächenbedarf

Bei der Betrachtung des Flächenbedarfes sind die in einer PLA-Struktur notwendigen Funktionseinheiten in ihren jeweiligen Flächen-Ausmaßen zu betrachten: Aus dem Blockschaltbild in Abbildung 2.4 (S. 18) geht hervor, daß für jede der n Eingangsleitungen ein Invertierer notwendig ist. Ferner sind signalverstärkende Puffer-Stufen sowohl in den nicht-negierten Zweigen der Eingangsleitungen als auch in den Ausgangsleitungen notwendig. Da deren Ausmaße im Layout gegenüber den beiden Matrizen nur einen geringen Anteil ausmachen, sind sie bei den weiteren Betrachtungen vernachlässigbar.

Der Flächenbedarf des PLAs wird daher maßgeblich von der Kapazität der beiden Matrizen bestimmt. Unter der Annahme, daß die Erhöhung der Anzahl ℓ der Termleitungen im Layout ein vertikales Fortschreiten bewirkt, kann die PLA-Fläche direkt in Gestalt der Layout-Abmessungen gefaßt werden: Sei in vertikaler Orientierung die Höhe eines PLA-Blockes durch $\mathrm{H_{PLA}}$ angegeben; entsprechend in horizontaler Orientierung die Breite durch $\mathrm{W_{PLA}}$. Mit dem Produkt aus Breite $\mathrm{W_{PLA}}$ und Höhe $\mathrm{H_{PLA}}$ einer PLA-Struktur im Layout ergeben sich für die Fläche $\mathrm{A_{PLA}} := \mathrm{W_{PLA}} \cdot \mathrm{H_{PLA}}$ in Einklang zur PLA-Kapazität aus Gleichung 2.9 (S. 21) folgende Proportionalitätsbeziehungen:

$$\begin{array}{lcccl} \text{Breite} & : & \mathrm{W_{PLA}} & \sim & m+n \\ \text{Höhe} & : & \mathrm{H_{PLA}} & \sim & \ell \end{array} \tag{2.14}$$

Erwartungsgemäß sind beide Beziehungen jeweils als linear anzusehen, obwohl bei einer konkreten Realisierung zusätzliche Einflüsse auftreten, die durch die Wahl von und Herstellungstechnologie geprägt werden. Reale Abschätzungen für eine konkrete Entwurfsumgebung sind im Anhang A.2 mit Gleichung A.1 (S. 156) angegeben.

Anders als die bisher diskutierten Kriterien ist der Flächenbedarf von den konkreten Rahmenbedingungen einer Realisierung abhängig; wird jedoch zunächst die Termanzahl vernachlässigt, so sind folgende Aspekte zu berücksichtigen:

- **Entwurf:** Formatparameter n und m.
- **Fertigung:** Herstellungstechnologie (Nmos, Cmos bzw. Pseudo-Nmos o. ä.) mit den Parametern für die Integrationsdichte (z. B. 1μm-Prozeß).

Das kompakte Layout der PLA-Makrozelle hat sich im Vergleich zu anderen Strukturen für die Realisierung von Schaltfunktionen als sehr effizient herausgestellt. Nachdem die Entscheidung für eine PLA-Realisierung gefallen ist, hat in der Entwurfspraxis der relativ geringe Flächenbedarf nur eine untergeordnete Relevanz.

2.2.5.5 Stromaufnahme

Die Strom- bzw. Leistungsaufnahme eines PLAs steht unter dem Einfluß der zur Herstellung verwendeten Technologie und den späteren Einsatzbedingungen gefertigten Schaltung. Die Stromaufnahme bestimmt außerdem den Aufwand, der zur Bereitstellung der elektrischen Energie im Gesamtentwurf erforderlich wird, nämlich:

- Auslegung der Stromversorgungs-Zellen (engl. *Pads*),
- Anzahl der Anschlüsse für die Spannungsversorgung [Rau-95],
- Flächenbedarf der Versorgungsleitungen.

Die Stromaufnahme der PLA-Struktur wird durch die Eigenschaften und die Schalthäufigkeit der aktiven Schaltglieder bestimmt. Dabei fällt auf, daß ständig *alle* implementierten Schaltglieder aktiviert sind, da bei jeder Eingangsbelegung die logischen Verknüpfungen in beiden PLA-Matrizen stattfinden müssen. Anders bei der ROM-Struktur: Dort sorgt ein Dekodierer dafür, daß von den insgesamt 2^n Matrixzeilen in jeweils nur *einer* Zeile die entsprechenden m Schaltglieder aktiviert werden müssen.

Die Anzahl der in den PLA-Matrizen aktiven Schaltglieder entspricht der Anzahl der Literale. Da sie eine kausale Abhängigkeit von der Termanzahl ℓ aufweist (vgl. Abschnitt 2.2.5.2, S. 29), steht auch die Stromaufnahme eines PLAs, neben allgemeinen Entwurfsparametern, überwiegend unter dem Einfluß der Termanzahl.

Der Besetzungsgrad der Matrizen, der sich aus dem Verhältnis von Literalanzahl und Kapazität ergibt (vgl. Gleichung 2.10, S. 21), wird durch die Verknüpfungen, die die Terme repräsentieren, bestimmt. Unter der Annahme einer maximalen Literalanzahl, d. h. die Literalanzahl ist gleich der Kapazität, folgt die totale Besetzung aller Knotenpunkte. Vernachlässigt man die Formatparameter n und m, so ist die Stromaufnahme $\mathrm{I}_{\mathrm{PLA}}$ eines NMOS-PLAs von der Termanzahl ℓ linear abhängig:

$$\mathrm{I}_{\mathrm{PLA}} \sim \ell \tag{2.15}$$

Darüberhinaus besteht für die Stromaufnahme außer der Termanzahl eine Abhängigkeit in dreifacher Hinsicht:

- **Entwurf:** Formatparameter n und m.
- **Fertigung:** Herstellungstechnologie (NMOS, CMOS bzw. Pseudo-NMOS o. ä.) mit den Parametern für die Integrationsdichte (z. B. 1μm-Prozeß).
- **Betrieb:** Versorgungsspannung V_{DD} und Frequenz des Systemtaktes f_{clk}.

Ohne Betrachtung dieser Parameter liefert die Termanzahl einen Anhaltspunkt, der Rückschlüsse auf die zu erwartende Stromaufnahme zuläßt. Reale Abschätzungen für eine konkrete Entwurfsumgebung sind im Anhang A.2 mit Gleichung A.2 (S. 156) angegeben. Wegen des derzeitigen Trends zu autonomen Systemen und Low-Power ICs (vgl. Abschnitt 1.1.1.2, S. 2) ist die Stromaufnahme ein Kriterium mit hoher Relevanz.

2.2.5.6 Überblick und Bewertung

Kriterien für die Kosten eines PLAs dienen der Einschätzung des Realisierungsaufwandes. Sie müssen auf den Hintergrund der Fragestellung abgestimmt sein. Daher sind Flächenbedarf und Stromaufnahme, wie bereits diskutiert, nur unter Einbeziehung technischer Parameter zu betrachten. Im Gegensatz dazu sind die ersten drei Kriterien, Termanzahl, Literalanzahl und Kapazität des PLAs, frei von derartigen Rahmenbedingungen.

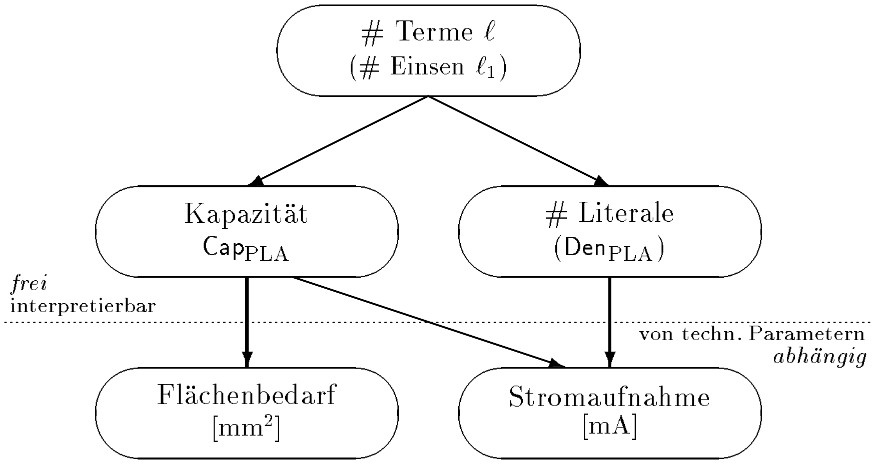

Abbildung 2.11: Beziehungen zwischen PLA-Kostenkriterien

In Abbildung 2.11 (S. 32) werden die Beziehungen zwischen den einzelnen Kostenkriterien grafisch veranschaulicht, wobei unterhalb der gepunkteten Linie die von technischen Parametern abhängenden Kriterien *Flächenbedarf* und *Stromaufnahme* dargestellt sind. Oberhalb stehen diejenigen Kriterien, die im Gegensatz dazu frei interpretiert werden können. In Gestalt der Kanten sind die Beziehungen zwischen den Kostenkriterien untereinander wiedergegeben: Eine Kante weist dabei von einem Kriterium V zum Kriterium W, falls eine Änderung bei V auch eine Änderung bei W zur Folge hat.

Intuitiv einleuchtend ist zunächst, daß eine Änderung der Kapazität zum einen den Flächenbedarf beeinflußt. Zum anderen ist aber auch die Anzahl der Literale davon mitbetroffen, was außerdem eine Änderung der Stromaufnahme bewirkt. Die Kapazität ihrerseits wird durch die Termanzahl bzw. der `1`en-Anzahl beeinflußt. Eine Änderung hier bedeutet aber auch eine Veränderung der Literalanzahl bzw. des Besetzungsgrades der PLA-Matrizen.

In einer Bestandsaufnahme aller diskutierten Kriterien ergibt sich folgende Bewertung bzw. Abwägung der Kostenkriterien gegeneinander:

- **Terme:** Am dominantesten ist die Termanzahl ℓ, da sie offenbar Einfluß auf alle anderen Kriterien ausübt. Sie stellt damit das wichtigste Kostenkriterium dar und ermöglicht eine frei interpretierte Beurteilung der Kosten. Als ein erster Anhaltspunkt kann ihr die, `1`en-Anzahl $\ell_{\mathtt{1}}$ vorausgehen, die sie nach oben beschränkt.

- **Literale:** Die Anzahl der Literale dient lediglich zur Bestimmung des Besetzungsgrades, was innerhalb des PLA-Kontextes nur dann eine Rolle spielt, falls die Diskussion auf strukturelle PLA-Varianten verlagert wird.

- **Kapazität:** Nur eingeschränkt fällt der PLA-Kapazität als Kriterium Bedeutung zu: Bei der Festlegung der Wertebereiche am Eingang (Indizes) und am Ausgang (Funktionswerte) wirkt sich eine günstige Wahl bei der Kapazität sichtbar aus. Als günstig ist eine Wahl anzusehen, bei der die von der Binärdarstellung vorgegebenen Wertebereiche möglichst effektiv ausgeschöpft werden: So ist beispielsweise eine Tabelle günstiger, die $128 = 2^7$ Einträge umfaßt, anstatt nur 100 Einträge zu definieren.

- **Flächenbedarf:** Die Entscheidung für eine PLA-Implementierung wird hauptsächlich dadurch begünstigt, daß sie eine sehr reguläre Struktur darstellt; sie ist damit leicht zu generieren. Außerdem fällt der Flächenbedarf sehr gering aus. Für den weiteren Kontext dieser Arbeit spielt der Flächenbedarf daher als Kostenkriterium keine große Rolle, zumal er nicht unabhängig von technischen Parametern zu interpretieren ist.

- **Stromaufnahme:** Für den Entwurf von Mikroelektroniksystemen weisen gegenwärtige Trends zunehmend in Richtung allgemeiner Energieeinsparung. Begründet wird dieser Umstand durch die gesteigerte Nachfrage nach autonomen Systemen und der wachsenden Verlustwärme-Problematik, die durch die steigende Komplexität der Mikroelektroniksysteme geradezu forciert wird.

  Die Spezifikationen der Chip-Entwürfe sehen demzufolge immer häufiger eine Beschränkung der elektrischen Leistungsaufnahme vor. Bemühungen zur Absenkung der Versorgungsspannung[6] laufen konträr zu einem sicheren Signal-/Rausch-Abstand. Letztlich wird der Aspekt der Stomaufnahme dadurch immer kritischer; vermutlich gehen Anbieter von Entwurfssystemen teilweise deshalb schon dazu über, ihren Entwurfsumgebungen gar keine Werkzeuge zur Generierung von PLAs beizufügen. Der Entwurf von PLA-Strukturen ist daher in der Entwurfspraxis häufig schon aus dem Blickfeld geraten.

  Im Kontext der vorliegenden Arbeit steht zur Motivation der Untersuchungen mehr der kritische Aspekt der Stromaufnahme im Vordergrund. Da sie jedoch in erster Näherung linear von der frei interpretierbaren Termanzahl abhängig ist, dient zur Einschätzung der PLA-Kosten vordergründig die dominierende Termanzahl.

[6] In der Regel sollen die Signalparameter der Schaltkreise TTL-kompatibel sein; wegen des dabei spezifizierten minimalen High-Pegels von 2.5 V kann die Versorgungsspannung kaum weiter als auf etwa 3 V abgesenkt werden.

Kapitel 3

Spezifikation von Tabellen

3.1 Vorbemerkungen

3.1.1 Begriffsbildung: „Spezifikation“

Für den Begriff *Spezifikation* werden in [Tim-82] die beiden Synonyme *Definition* und *Beschreibung* erörtert. Vor dem Hintergrund des *Software-Engineerings* führt *Timm* die Diskussion dort zu der Erkenntnis, daß im allgemeinen der *beschreibende* Aspekt bei einer Spezifikation überwiegt.

Als Folge der Diskussion wird die beschreibende Aktivität als *Spezifizierung* bezeichnet; mit *Spezifikation* wird dementsprechend das resultierende Ergebnis assoziiert. Eine Spezifikation einer Tabelle sollte somit als das Resultat einer Spezifizierung angesehen werden, die als Prozeß je nach Abstraktionsgrad in unterschiedlicher Feinheit aufgelöst werden kann.

3.1.2 Bezug zum PLA-Entwurf

Beim Spezifizieren einer Tabelle steht *nicht* der Entwurf eines PLAs zur Diskussion. Vielmehr bildet die spezifizierte Tabelle die Basis für den PLA-Entwurf, für den prinzipiell alle im Abschnitt 2.2.4 (vgl. S. 23) erwähnten Varianten verwendet werden können. Der eigentliche PLA-Entwurf ist in diesem Kontext nur insofern relevant, da die entstehenden Kosten[1] hier als Bewertungsmaßstab für eine gewählte Tabellen-Spezifikation herangezogen werden. Ferner müssen die beim Spezifizieren vorgenommenen Anpassungen bzw. Festlegungen von den weiteren Stufen in der Systemlösung berücksichtigt werden, damit die beabsichtigte Funktionalität im Gesamtkontext der PLA-Implementierung unverändert[2] erhalten bleibt.

Außerdem soll in diesem Abschnitt gezeigt werden, daß das Spezifizieren einer Tabelle nur in erster Näherung ein statischer *Top-Down* Ablauf von einzelnen, standardmäßig vorzunehmenden Aktionen ist. Im Detail zeigt sich, daß es sich, wie auch beim PLA-Entwurf selbst, um einen Entstehungsprozeß handelt, bei dem Rückkopplungen vorkommen und Entscheidungen zu treffen sind.

[1] Um von der konkreten Wahl einer Realisierungsvariante und einer Entwurfsumgebung unabhängig zu sein, sei hier die *Anzahl der Terme* als relevante Kostengröße interpretiert (vgl. Abschnitt 2.2.5.6, S. 32).

[2] In Kapitel 5 wird genau bei dieser Forderung eine gewisse Toleranz zugelassen.

3.1.3 Abgrenzung des Problemfeldes

Im allgemeinen muß davon ausgegangen werden, daß Modifikationen, wie sie später in Kapitel 5 noch erläutert werden, an einer Spezifikation gravierende Veränderungen verursachen können, die für die meisten Problemlösungen prinzipiell nicht zu akzeptieren sind.

Darunter fallen beispielsweise kombinatorische Systeme, bei denen Abläufe in Steuerwerken mit Hilfe einer PLA-Tabelle implementiert werden. Hierzu zählen u. a. Zustandsübergangsfunktionen für Automaten sowie (Mikro-) Programmcode für Mikroprozessoren oder Controller. Hier würden Modifikationen an einer Tabelle teilweise zu gravierenden Änderungen im Systemverhalten führen, die möglicherweise nicht abschätzbare Auswirkungen zur Folge hätten. Für diese Art von Tabellen werden PLAs ohnehin weniger eingesetzt, vielmehr ist dies ein typisches Anwendungsgebiet von ROM-Strukturen (vgl. Abschnitt 2.1.5.3, S. 17).

Ganz anders verhält es sich bei reellen Funktionen, die in tabellarischer Form dargestellt sind: Bei dieser Art der Lösung eines numerischen Problems treten selbstverständlich Fragen nach der Genauigkeit der verwendeten Operationen auf, so daß immer Abstriche von der ursprünglichen Funktionalität hingenommen werden müssen.

3.1.4 Numerische Probleme

3.1.4.1 Exaktheit von numerischen Lösungen

Grundsätzlich entstehen bei der Lösung von numerischen Problemen mit Hilfe technischer Systeme Fehler aus verschiedenen Quellen, so auch auf dem Wege von der ursprünglichen Funktionalität bis hin zur Problemlösung mit Hilfe einer PLA-Implementierung. Dabei sind auf verschiedenen Ebenen mehrere Formen zur Repräsentation der Funktion notwendig, bei denen zwangsweise Fehler von unterschiedlichem Ausmaß auftreten. Die Spezifikation einer PLA-Tabelle, mag sie auch noch so sorgfältig erarbeitet worden sein, kann in diesem Sinne nie exakt erstellt werden.

3.1.4.2 Zahlendarstellung mit Stellensystemen

Die Auswertung von Funktionen in digitalen Systemen erfordert eine numerische Repräsentation. Üblicherweise werden dazu Stellensysteme, wie beispielsweise das Dezimal- oder das Dualsystem, verwendet [Bro*81]: Eine positiv rationale Zahl $q \in \mathbb{Q}^+$ ($\mathbb{Q}^+ := \{v \in \mathbb{Q} \mid v \geq 0\}$) kann mit Hilfe einer natürlichen Basiszahl $p \in \mathbb{N}$ mit $p \geq 2$ und einer p-mächtigen Ziffernmenge $\mathbf{T}$ (eine Menge geordneter Symbole) in einer sogenannten p-adischen Darstellung wie folgt repräsentiert werden:

$$\begin{aligned} q \quad &:= \quad t_s p^s + t_{s-1} p^{s-1} + \ldots + t_0 p^0 + \ldots + t_{-r} p^{-r} \\ \text{dabei} \quad &\text{sind} \quad t_s, \ldots, t_0, \ldots, t_{-r} \in \mathbf{T} \end{aligned} \tag{3.1}$$

Dabei ist $s \in \mathbb{N}$ durch $s := \max\Big(\{b \in \mathbb{N} \mid b \leq \log_p(a)\}\Big)$ eindeutig bestimmbar und $r \in \mathbb{N}$ beliebig. Ferner gilt für $t_s \in \mathbf{T}$, daß es nicht die Ziffer Null ist. Die Ziffernfolge $t_s t_{s-1} \ldots t_0, \ldots t_{-r}$ ist das repräsentierende Zahlwort für q, wobei das Komma die Position der Ziffer t_0 in der Symbolreihe markiert.

Da die Darstellung im allgemeinen endlich sein muß, d. h. r muß beschränkt werden, können irrationale Zahlen nicht repräsentiert werden. Das bedeutet, daß eine nicht darzustellende Zahl durch eine darstellbare repräsentiert werden muß; dabei treten unvermeidbare Abweichungen auf (vgl. Abschnitt 3.1.4.4, S. 38). Bei der numerischen Repräsentation von Zahlen, die nicht aus $\mathbb{Z}$ sind, werden der ganzzahlige und der gebrochene Anteil mit Hilfe des Kommas voneinander getrennt. Generell bieten sich zwei alternative Methoden an, die sich in der Behandlung des Kommas voneinander unterschieden:

- **Festkommadarstellung:** Bei der Zahlendarstellung mit festem Komma ist jeder Ziffer $t_k \in \mathbf{T},\ k \in \{b \in \mathbb{Z} \mid -r \leq k \leq s\}$ in der Symbolreihe entsprechend seiner Position zum Komma ein fester Stellenwert mit p^k gegeben. Diese Darstellungsart ist nur dann sinnvoll, wenn sich die Zahlenwerte in einem gewissen Rahmen bewegen, d. h. nicht völlig unbeschränkt sind. Bei der Spezifikation von PLA-Tabellen, wie sie Gegenstand dieser Arbeit ist, kommt ausschließlich diese Darstellung zum Einsatz. Die Tabellen werden im binären Dualsystem dargestellt und erfordern daher eine Beschränkung der Stellenanzahl.

- **Gleitkommadarstellung:** Liegen Zahlenwerte vor, die in ihrer Größenordnung sehr unterschiedlich sind, oder können im Vorwege über die zu repräsentierenden Zahlenwerte keine diesbezüglichen Aussagen gemacht werden, so bietet sich die Gleitkommadarstellung an. Sie erfordert die Aufspaltung eines Zahlenwertes in *Mantisse* und *Exponenten*, was einen erhöhten Darstellungsaufwand bedeutet, zugleich aber leicht einen sehr großen Wertebereich ermöglicht. Zum Einsatz kommt dies beispielsweise bei universellen Rechenwerken [Bro*81]. Für den Kontext dieser Arbeit hat sie keine wesentliche Bedeutung.

3.1.4.3 Kenngrößen einer Zahlendarstellung

Numerische Darstellungen repräsentieren Zahlen im allgemeinen mit endlich vielen Stellen. Der Verlust von signifikanten Stellen ist hier prinzipiell unvermeidbar (vgl. Abschnitt 3.1.4.2, S. 36). Beispielsweise tritt diese Problematik bei der Darstellung von sehr großen Zahlen (Beschränkung von s) oder auch bei irrationalen Zahlen (Beschränkung von r) auf. Als charakteristische Merkmale numerischer Zahlendarstellungen sind vor allem zwei Kenngrößen zu berücksichtigen:

- **Minimale Auflösung:** Mit diesem Begriff sei das Vermögen zur Darstellung von minimalen und von Null verschiedenen Zahlenwerten assoziiert. Damit verknüpft ist die Unterscheidbarkeit von großen Zahlenwerten, d. h. die minimal auflösbare Zahlendifferenz. Die erreichbare Auflösung hängt von der Ausdehnung der Stellen in Richtung niedriger Wertigkeit ab: Sie ist durch die Beschränkung von r in Gleichung 3.1 (S. 36) gegeben. Für ein *großes* r werden *viele* niederwertige Stellen dargestellt; daraus resultiert ein *hohes* Auflösungsvermögen bzw. ein *kleines* absolutes Minimum. Nach einer Beschränkung von r ergibt sich für diese Kenngröße folgender Wert:

$$\text{minimale Auflösung:} \quad \Delta_{\min} := p^{-r} \tag{3.2}$$

- **Absolutes Maximum:** Die numerische Repräsentation mit Hilfe endlich vieler Stellen bedeutet, daß ein absolutes Maximum existiert. Es hängt maßgeblich von der Ausdehnung der Stellen in Richtung höherer Wertigkeit ab: Sie ist im wesentlichen durch die Beschränkung von s in Gleichung 3.1 (S. 36) bestimmt. Von geringerem Ausmaß ist der Einfluß von r, durch welches die niederwertigste Stelle bestimmt wird. Durch Erhöhen von

s können *viele* höherwertige Stellen dargestellt werden, was ein *großes* absolutes Maximum bewirkt. Nach Festlegung von s existiert unabhängig von r für diese Kenngröße folgende Abschätzung:

$$\text{absolutes Maximum:} \quad M < p^{s+1} \tag{3.3}$$

Da beide Kenngrößen den Bereich der darstellbaren Zahlenwerte festlegen, haben sie die Bedeutung von typischen Parametern einer konkreten Zahlendarstellung. Bei ihrer Festlegung wird sowohl für die dezimale als auch die duale Zahlendarstellung die jeweils erforderliche Stellenanzahl beeinflußt.

3.1.4.4 Begrenzung der Auflösung

Die Auflösung in einer p-adischen Zahlendarstellung (vgl. Gleichung 3.1, S. 36) hängt von der Ausdehnung der Stellen in Richtung niedriger Wertigkeit ab (vgl. Gleichung 3.2, S. 37). Das Begrenzen der Auflösung beschränkt deshalb die Anzahl der niederwertigen Stellen: Ausgehend von den höherwertigen Stellen, muß explizit festgelegt werden, bis zu welcher Stelle e ($-r \leq e \leq s$ mit r und s gemäß Gleichung 3.1, S. 36) die Zahlendarstellung erfolgt. Auf diese Weise wird für eine nichtdarstellbare Zahl $q \in \mathbb{R}$ ein sogenannter *Näherungswert* $w_e \in \mathbb{Q}$ erzeugt. Dafür werden nach [Bro*81] grundsätzlich zwei Methoden unterschieden:

- **Abbrechen**: Der Näherungswert $w_e \in \mathbb{Q}$ entsteht durch *Abbrechen* der Darstellung (engl. *truncate*)[3] von $q \in \mathbb{R}$ an der e-ten Stelle. Daher weisen zu Beginn die Darstellungen von q und w_e exakt dieselben Stellen auf. Der resultierende Fehler ergibt sich aus ihrer Differenz und kann wie folgt abgeschätzt werden:

$$\text{Abbruchfehler:} \quad q - w_e < p^{-e} \tag{3.4}$$

- **Runden**: Zur Minimierung der Differenz zwischen $q \in \mathbb{R}^+$ und dem Näherungswert $w_e \in \mathbb{Q}^+$ sind beim sogenannten (Nahe-)*Runden* (engl. *<u>round</u> to nearest integer*) an der e-ten Stelle nach [Bro*81] die folgenden drei Regeln zu berücksichtigen: Sei in q die e-te Ziffer durch $t_e \in \mathbf{T}$ gegeben (vgl. Abschnitt 3.1.4.2, S. 36). Folgt beim Abtrennen der auf t_e folgenden (niederwertigen) Stellen

  (a) eine Ziffer $t_{e-1} \in \{0, 1, 2, 3, 4\}$, so wird t_e unverändert übernommen; dies wird als *Abrunden* bezeichnet.

  (b) eine Ziffer $t_{e-1} \in \{5, 6, 7, 8, 9\}$ und existiert mindestens noch eine weitere niederwertige Ziffer, die nicht Null ist, so wird die e-te Stelle in ihrem Ziffernwert inkrementiert, d. h. t_e wird durch die nächste Ziffer ersetzt, die gemäß der auf $\mathbf{T}$ definierten Ordnung einen höheren Ziffernwert darstellt. Im Falle $t_e = 9$ wird die $0 \in \mathbf{T}$ gewählt und an der nächst höherwertigen Stelle t_{e+1} wird jene Inkrementierung erschöpfend, d. h. bis es nicht mehr notwendig ist, fortgesetzt (analog zum arithmetischen Übertrag). Dieses Vorgehen wird als *Aufrunden* bezeichnet.

  (c) lediglich eine Ziffer $t_{e-1} = 5$ bzw. die Ziffer $t_{e-1} = 5$ und alle weiteren Ziffern $t_{e-k} = 0,\ k > 1$ sind Null, so wird abgerundet (c.1), falls die unveränderte Ziffer t_e *gerade* ist; ansonsten wird aufgerundet (c.2).

[3] Englisch *truncate*: „stutzen, beschneiden, verstümmeln, verkürzen“.

Da je nach angewandter Regel der Näherungswert w_e entweder durch Ab- oder durch Aufrunden erzeugt wird, hat dies im Gegensatz zum Abbrechen die Konsequenz, daß die Stellen der Darstellung von q und w_e im allgemeinen verschieden sein können. Während bei der Erzeugung eines Näherungswertes durch Abbrechen der resultierende Wert w_e nicht größer als die ursprüngliche Zahl q ist, so kann er beim Runden auch größer sein. Die Schranke des resultierenden Fehlers ist maximal nur halb so groß wie der Abbruchfehler:

$$\text{Rundungsfehler:} \quad |q - w_e| \leq \tfrac{1}{2} \cdot p^{-e} \tag{3.5}$$

Grundsätzlich kann das Abtrennen von niederwertigen Stellen ab jeder beliebigen Stelle e (wobei gilt: $-r \leq e \leq s$) mit r und s gemäß Gleichung 3.1 (S. 36) erfolgen. Häufig wird jedoch zur Vereinfachung $e := 0$ gewählt, d. h. es werden Näherungswerte w_0, oder einfacher w, gebildet. Weitere Erläuterungen zum Runden werden später in Abschnitt 3.3.4.1 diskutiert.

3.1.5 Spezifikationsbeispiel: Sinus

Zur Veranschaulichung der nachfolgenden Analyse sei vorweg das bereits erwähnte *Sinus*-Beispiel aus Abschnitt 2.1.2 (vgl. S. 14) aufgegriffen, hier allerdings in einer umfangreicheren Variante. Zu diesem Beispiel wird die Spezifikation einer möglichen PLA-Tabelle vorgestellt. Die Sinus-Funktion erscheint geradezu als typisches Beispiel, da sie in vielen Bereichen angewendet wird, beispielsweise in der Bildverarbeitung beim Rotieren von grafischen Objekten. Sie steht darüberhinaus stellvertretend für transzendente Funktionen, bei denen die Funktionswerte durch Auswerten von Reihenentwicklungen gewonnen werden müssen.

Sinus-Viertelwelle

Es werden 100 Werte der ersten Viertelwelle einer Sinus-Schwingung auf 6 Dezimalstellen generiert. Ausgehend vom Maximalwert 1.0, wird anschließend eine Multiplikation mit dem Faktor 1000.0 durchgeführt. Die Umwandlung in eine direkt binär darstellbare Tabelle findet durch Runden zur nächstliegenden Ganzzahl statt.

In Tabelle 3.1 (S. 40) ist die Spezifizierung in ihren einzelnen Stufen wiedergegeben. Der besseren Übersicht wegen geschieht dies nur auszugsweise, d. h. die Einträge 16 bis 96 sind ausgelassen. Die Festlegung auf eine Anzahl von 100 Werten entstammt der Praxis und soll an dieser Stelle verdeutlichen, daß nicht selten die Werteanzahl ungleich einer 2-er Potenz gewählt wird, da dies im typischen Umfeld der Problemstellung auch nicht als relevant angesehen wird. In der binären Tabelle jedoch hat dies zur Konsequenz, daß einige Einträge ungenutzt bleiben; in diesem Beispiel sind es $2^7 - 100 = 28$ Einträge.

Eine Logik-Minimierung mit Hilfe von Espresso [Bra*82b, Bra*84] ergibt hierfür einen PLA-Kostenaufwand von 93 Termen. Um konkrete Werte für die Stromaufnahme zu errechnen, muß eine Entwurfsumgebung gewählt werden. Im Anhang A sind entsprechende Angaben zur ES2-Entwurfsumgebung Solo-2030 zu finden [ESS-92]: Gemäß Gleichung A.2 (S. 156) liegt die Stromaufnahme für das Beispiel in diesem Fall bei einer Betriebsspannung von 5 V zwischen 28 mA und 39 mA, je nach gewähltem Fertigungsprozeß und Systemtakt (hier beispielsweise im Bereich 20 bis 60 MHz).

i	$\sin(\frac{\pi}{2} \cdot \frac{i}{99})$	$\cdot$ 1000.0	round( )	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	0.000000	0.000	0	0	0	0	0	0	0	0	0	0	0
1	0.015866	15.866	16	0	0	0	0	0	1	0	0	0	0
2	0.031728	31.728	32	0	0	0	0	1	0	0	0	0	0
3	0.047582	47.582	48	0	0	0	0	1	1	0	0	0	0
4	0.063424	63.424	63	0	0	0	0	1	1	1	1	1	1
5	0.079250	79.250	79	0	0	0	1	0	0	1	1	1	1
6	0.095056	95.056	95	0	0	0	1	0	1	1	1	1	1
7	0.110838	110.838	111	0	0	0	1	1	0	1	1	1	1
8	0.126592	126.592	127	0	0	0	1	1	1	1	1	1	1
9	0.142315	142.315	142	0	0	1	0	0	0	1	1	1	0
10	0.158001	158.001	158	0	0	1	0	0	1	1	1	1	0
11	0.173648	173.648	174	0	0	1	0	1	0	1	1	1	0
12	0.189251	189.251	189	0	0	1	0	1	1	1	1	0	1
13	0.204807	204.807	205	0	0	1	1	0	0	1	1	0	1
14	0.220311	220.311	220	0	0	1	1	0	1	1	1	0	0
15	0.235759	235.759	236	0	0	1	1	1	0	1	1	0	0
16	0.251148	251.148	251	0	0	1	1	1	1	1	0	1	1
17	0.266474	266.474	266	0	1	0	0	0	0	1	0	1	0
18	0.281733	281.733	282	0	1	0	0	0	1	1	0	1	0
19	0.296920	296.920	297	0	1	0	0	1	0	1	0	0	1
20	0.312033	312.033	312	0	1	0	0	1	1	1	0	0	0
21	0.327068	327.068	327	0	1	0	1	0	0	0	1	1	1
22	0.342020	342.020	342	0	1	0	1	0	1	0	1	1	0
23	0.356886	356.886	357	0	1	0	1	1	0	0	1	0	1
24	0.371662	371.662	372	0	1	0	1	1	1	0	1	0	0
25	0.386345	386.345	386	0	1	1	0	0	0	0	0	1	0
26	0.400931	400.931	401	0	1	1	0	0	1	0	0	0	1
27	0.415415	415.415	415	0	1	1	0	0	1	1	1	1	1
28	0.429795	429.795	430	0	1	1	0	1	0	1	1	1	0
29	0.444067	444.067	444	0	1	1	0	1	1	1	1	0	0
30	0.458227	458.227	458	0	1	1	1	0	0	1	0	1	0
31	0.472271	472.271	472	0	1	1	1	0	1	1	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
96	0.998867	998.867	999	1	1	1	1	1	0	0	1	1	1
97	0.999497	999.497	999	1	1	1	1	1	0	0	1	1	1
98	0.999874	999.874	1000	1	1	1	1	1	0	1	0	0	0
99	1.000000	1000.000	1000	1	1	1	1	1	0	1	0	0	0
100	-	-	-	*	*	*	*	*	*	*	*	*	*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
127	-	-	-	*	*	*	*	*	*	*	*	*	*

Tabelle 3.1: Beispiel zum Spezifizieren der Sinus-Viertelwelle (Auszug)

3.2 Analyse der Abläufe

3.2.1 Ausgangssituation

In einer Problemlösung kommt eine bestimmte Funktion f als Teillösung vor. Sie soll als PLA realisiert werden. Hierbei treten miteinander konkurrierende Interessen in einen Konflikt, die anhand der polarisierenden Standpunkte eines *Auftraggebers* einerseits, und eines *Entwerfers* andererseits diskutiert werden können: Zum einen werden an das Resultat in Gestalt des PLAs bestimmte Qualitätsansprüche gestellt, die einen entsprechend hohen Aufwand bei der Realisierung hervorrufen. Zum anderen dürfen dabei gesetzte Kostenschranken nicht überschritten werden, bzw. müssen die Kosten so niedrig sein, daß eine PLA-Realisierung überhaupt erst möglich wird (vgl. Abschnitt 2.2.3.2, S. 23).

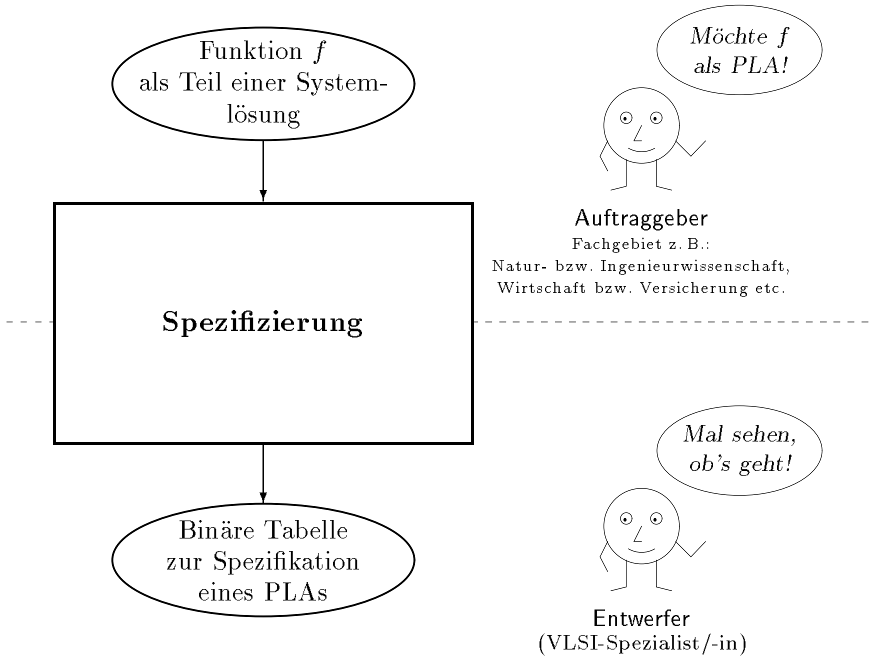

Abbildung 3.1: Ausgangssituation für eine Spezifikation

Die Spezifikation einer Tabelle selbst stellt nun gewissermaßen das Bindeglied zwischen beiden Polen dar. Sie wird entweder komplett nur vom Auftraggeber erstellt oder zusammen und in Absprache mit dem Entwerfer erarbeitet. Anschließend geschieht die Umsetzung zu einem PLA-Entwurf durch den Entwerfer. In Abbildung 3.1 ist diese Situation symbolisch wiedergegeben. Die Symbolik ist dabei an Petrinetze angelehnt: Üblicherweise stellen eckige Objekte (Transitionen) Aktionen dar, während abgerundete Symbole (Stellen) Resultate verkörpern.

Dementsprechend wendet sich der Kontext dieses Abschnittes 3.2 einer *prozedural* orientierten Diskussion zu, d. h. es stehen die inneren Abläufe bei der Spezifizierung im Blickfeld. Den funktionalen Aspekten hingegen widmet sich später im Kapitel 4 der Abschnitt 4.1 (S. 57ff).

3.2.2 „Top-Down“ Ablauf der Spezifizierung

Die am Beispiel der *Sinus-Viertelwelle* in Abschnitt 3.1.5 (S. 39f) demonstrierte Spezifikation deutete bereits eine grobe Gliederung bei der Spezifizierung an. Zunächst handelt es sich um einen abgestuften *Top-Down* Ablauf, dessen Anfangs- und Endpunkt bereits durch die Sichtweisen von Auftraggeber und Entwerfer gegeben sind.

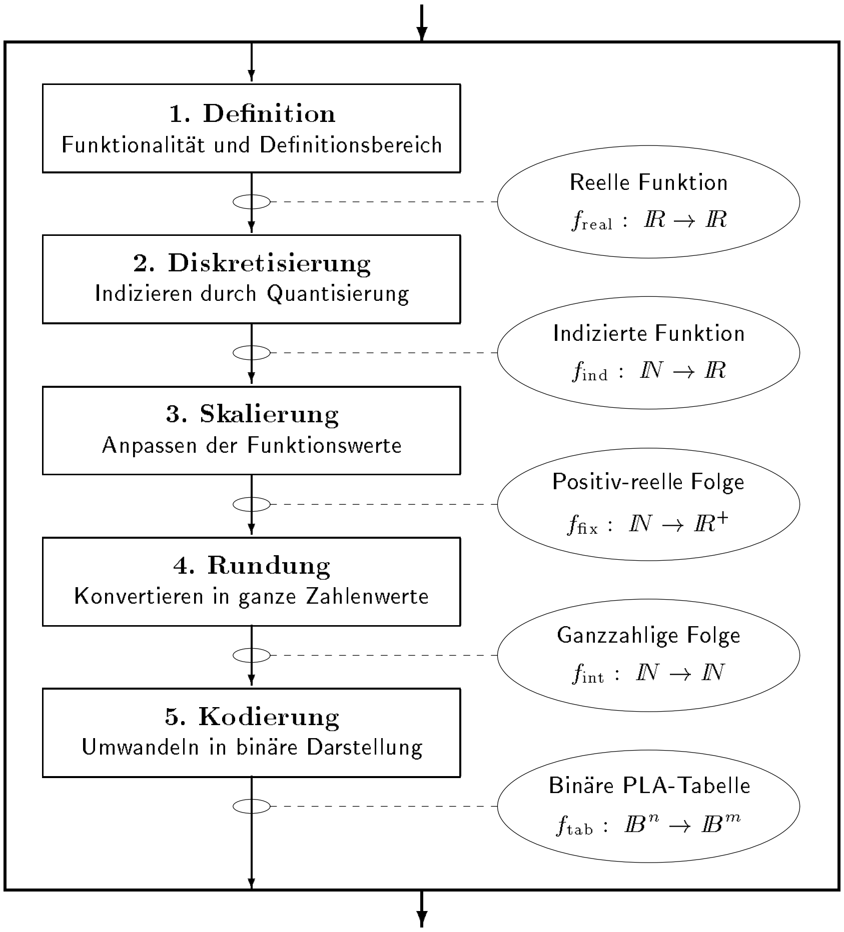

Abbildung 3.2: Spezifizieren in einer 5-stufigen Sequenz

In Abbildung 3.2 (S. 42) wird die interne Gliederung bei der Spezifizierung nun in Gestalt einer 5-stufigen Sequenz veranschaulicht. Die jeweiligen Zwischenresultate sind in Form der Funktionstransformationen angedeutet. In Abschnitt 3.3 ab Seite 43 wird jede Stufe der Sequenz im Detail analysiert.

3.3 Spezifizierung in fünf Stufen

In diesem Abschnitt werden die einzelnen Stufen der Sequenz beim Spezifizieren diskutiert. Die aufgezeigten Ablaufstrukturen und Funktionsbeziehungen dienen später als systematische Grundlagen für die Untersuchung unterschiedlicher Aspekte der Spezifizierung. Außerdem wird jede Stufe am *Sinus*-Beispiel demonstriert.

3.3.1 Stufe 1: Definition

Zu Beginn wird die Festlegung der zu realisierenden Funktionalität benötigt, d. h. es ist zunächst zu bestimmen, welche Funktion im konkreten Fall zur Lösung eines gestellten Problems verwendet werden soll.

3.3.1.1 Reelle Funktion als Abbildung

Im allgemeinen kann von einer reellen Funktion[4] ausgegangen werden. Mit Bestimmung des Definitionsbereiches $\mathbf{D} \subseteq I\!\!R$ und des Wertebereiches $\mathbf{W} \subseteq I\!\!R$ (vgl. Abschnitt 1.2.1.1, S. 5) ergibt sich damit zunächst folgende Abbildung:

$$f_{\text{real}}: \ \mathbf{D} \rightarrow \mathbf{W} \tag{3.6}$$

3.3.1.2 Interner Ablauf bei der Definition

Der interne Ablauf der Definition in dieser Stufe ist trivial: Zunächst wird die Funktion selbst aufgestellt, die ausschließlich nach den Erfordernissen der angestrebten Problemlösung ausgerichtet wird. Weiter ist die Festlegung des Definitionsbereiches von Bedeutung, da auf diese Weise die für die Problemlösung entscheidenden Stellen selektiert werden.

Manche Eigenschaften der definierten Funktion verursachen im weiteren Verlauf der Spezifizierung Einschränkungen und müssen daher bereits in dieser Stufe besonders beachtet werden; dazu zählen z. B. Unstetigkeiten oder Definitionslücken.

3.3.1.3 Definition beim Sinus-Beispiel

Im angedeuteten Beispiel aus Abschnitt 3.1.5 (S. 39) bedeutet diese Stufe die Festlegung, daß „*Die erste Viertelwelle der Sinus-Schwingung*“ als (Teil-)Lösung eines Problems vorgesehen wird.

Die damit gegebene Funktion ist in Abbildung 3.3 (S. 44) grafisch dargestellt. Aus dieser Festlegung ergibt sich für das Beispiel nun folgende formale Definition:

$$f_{\text{real}}(x) := \sin(x) \quad \text{mit} \quad 0 \leq x \leq \frac{\pi}{2} \tag{3.7}$$

[4] Bei komplexen Funktionen ist eine Zerlegung in reellen Komponenten möglich.

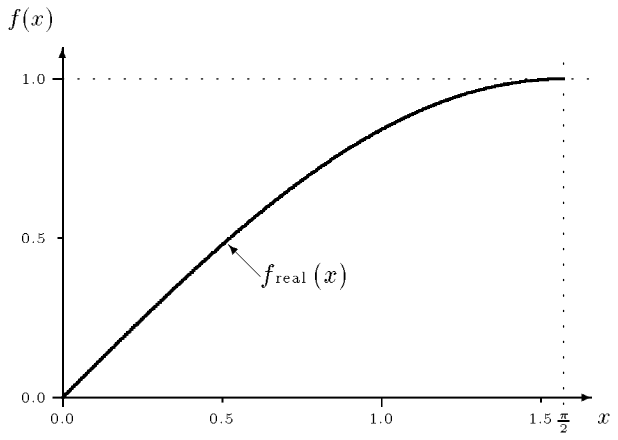

Abbildung 3.3: Beispielfunktion *Sinus-Viertelwelle*

3.3.2 Stufe 2: Diskretisierung

Funktionale Beziehungen, deren Bereiche kontinuierlich interpretiert werden, sind in digital arbeitenden Systemen nicht direkt darstellbar. Zu ihrer Repräsentation müssen kontinuierliche Bereiche durch diskrete Werte nachgebildet werden. Dieses Vorgehen wird *Quantisierung* genannt.

3.3.2.1 Quantisierung von Funktionen

Bei Funktionen führt die Diskretisierung des Definitionsbereiches $\mathbf{D} \subseteq I\!R$, bedingt durch die funktionale Beziehung $f_{\text{real}} : \mathbf{D} \rightarrow \mathbf{W}$, auch zum Diskretisieren des Funktionswertebereiches $\mathbf{W} \subseteq I\!R$. Ausgehend vom Definitionsbereich $\mathbf{D}$, werden sogenannte *Stützstellen* $s_i \in \mathbf{D}$ definiert. Bei der Auswertung der Funktion f_{real} wird dann der i-ten Stützstelle genau ein Funktionswert $f_{\text{real}}(s_i)$ zugeordnet. Werden die Stützstellen *äquidistant* über den Bereich verteilt, d. h. benachbarte Stützstellen haben einen konstanten Abstand zueinander, so wird dieser Abstand auch *Quantisierungseinheit* genannt:

$$\text{Quantisierungseinheit:} \quad \varepsilon := |s_{i+1} - s_i| \quad \text{wobei} \quad i \in I\!N \tag{3.8}$$

Eine Stützstelle s_i repräsentiert damit sämtliche Punkte aus dem Intervall $[s_i - \frac{\varepsilon}{2}, s_i + \frac{\varepsilon}{2}]$. In Abbildung 3.4 (S. 45) wird diese Situation zur Veranschaulichung wiedergegeben.

Offenbar werden bei der Quantisierung zwischen den Stützstellen unendliche Punktmengen vernachlässigt, wodurch zwangsläufig Abweichungen entstehen. Diese nehmen ab, falls durch eine Erhöhung der Stützstellenanzahl $N \in I\!N$ der Abstand ε zwischen ihnen gering gehalten wird. Sofern die Funktion im Bereich der Quantisierungseinheit einen monotonen Verlauf hat, ergibt sich der sogenannte *Quantisierungsfehler* als maximale Abweichung zwischen dem sich an der Stützstelle ergebenden Funktionswert und den durch ihn repräsentierten Funktionswerten. Bei äquidistanter Quantisierung ergibt er sich wie folgt:

$$\text{Quantisierungsfehler:} \quad \delta(s_i) := |f_{\text{real}}(s_i - \tfrac{\varepsilon}{2}) - f_{\text{real}}(s_i + \tfrac{\varepsilon}{2})| \tag{3.9}$$

Der zur Stützstelle s_i zugeordnete Funktionswert $f_{\text{real}}(s_i)$ repräsentiert nun alle Funktionswerte innerhalb des kontinuierlichen Bereiches $\delta(s_i)$; siehe Abbildung 3.4 (S. 45).

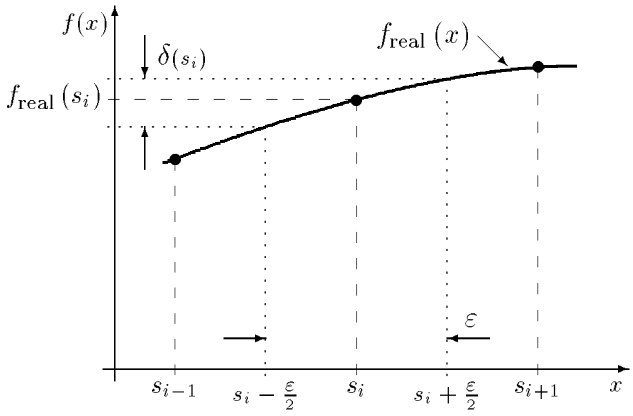

Abbildung 3.4: Quantisierung einer Funktion mit Stützstellen

3.3.2.2 Diskretisierung als Abbildung

Bezüglich der kontinuierlichen Funktion f_{real} aus Abbildung 3.2 (S. 42) geschieht das Gewinnen der Stützstellen dadurch, daß aus der Argumentenmenge einzelne (diskrete) Werte durch eine vorausgehende Abbildung selektiert werden; diese Abbildung muß dabei eine *eineindeutige* Zuordnung zwischen den Elementen einer Indexmenge $\{0, 1, 2, \ldots\} \subseteq \mathbb{N}$ und den reellen Stützstellen durchführen; das bedeutet, sie muß eine *injektive* Abbildung sein. Die entsprechende Abbildung sei *Indizierung* genannt und stellt quasi eine Numerierung der an f_{real} herangeführten Argumentwerte dar; Sie sei durch Ind bezeichnet:

$$\mathsf{Ind} : \mathbb{N} \to \mathbb{R} \tag{3.10}$$

Zu einer injektiven Abbildung existiert eine inverse Abbildung; sie wird auch Umkehrabbildung genannt. Die Umkehrabbildung zu Ind ordnet den einzelnen (diskreten) reellen Argumentwerten für f_{real} die entsprechenden Indizes zu; diese Umkehrabbildung sei im folgenden als *Diskretisierung* bezeichnet.

$$\mathsf{Dis} : \mathbb{R} \to \mathbb{N} \tag{3.11}$$

3.3.2.3 Indizierung der Funktion

Zur Auswertung der Funktionswerte sind nun Ind und f_{real} hintereinander zu evaluieren, d. h. nachdem zunächst durch Ind jedem Index $i \in \mathbb{N}$ eine Stützstelle $s(i) \in \mathbf{D} \subseteq \mathbb{R}$ zugeordnet wird, kommt anschließend f_{real} mit s_i als Argument zur Ausführung und liefert genau einen Funktionswert. Damit liegt nun eine *indizierte* Funktion f_{ind} als endliche, reelle Folge mit N Gliedern vor. Mit $i \in \mathbf{I}_N \subset \mathbb{N}$ (vgl. Abschnitt 1.9, S. 7) ergibt sich nun folgende Situation:

$$\begin{aligned} f_{\text{ind}} &: \mathbb{N} \to \mathbb{R} \quad \text{mit} \\ f_{\text{ind}}(i) &:= f_{\text{real}}\Big(\mathsf{Ind}(i)\Big) \end{aligned} \tag{3.12}$$

Zusammen mit dem Abstand zwischen den Stützstellen wird in dieser Stufe deren Anzahl $N \in I\!N$ festgelegt. Beide stehen miteinander in direkter Wechselwirkung: Eine hohe Anzahl an Stützstellen bedeutet im Mittel einen geringen Abstand zwischen ihnen bzw. umgekehrt. Während viele Stützstellen den Nachteil einer großen Datenmenge haben, bedeuten wenige eine grobe Quantisierung. Die Güte einer Diskretisierung hängt vom Kompromiß zwischen beiden Extremen ab. Da die Anzahl der Stützstellen ein so charakteristisches Merkmal der Eigenschaften einer Diskretisierung darstellt, sei dies künftig durch eine entsprechende Indexschreibweise, wie in Dis_N bzw. Ind_N, vermerkt.

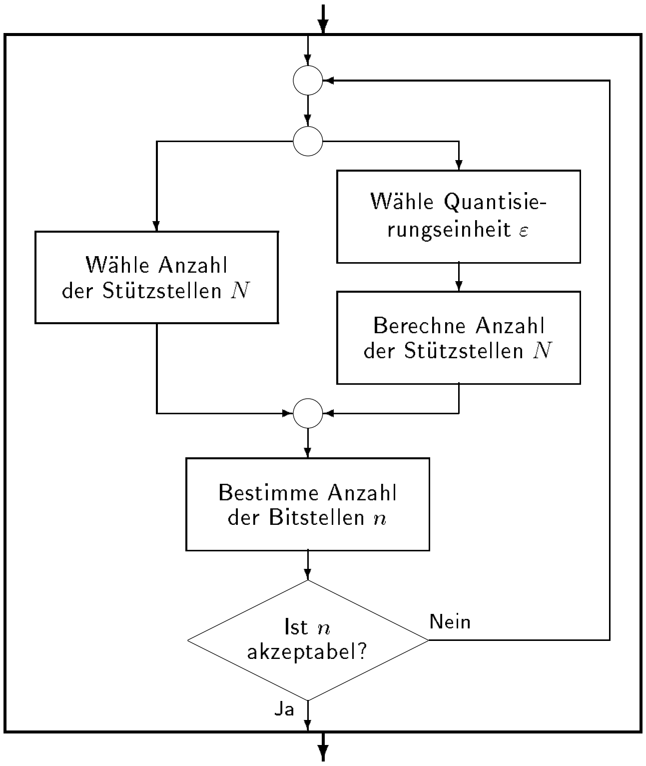

Abbildung 3.5: Ablaufdiagramm zur linearen Diskretisierung

3.3.2.4 Lineare Indizierung

In den meisten Problemlösungsansätzen ist eine Diskretisierung mit äquidistanten Stützstellen erwünscht; die Indizierung kann in diesen Fällen mit einer linearen Funktion vorgenommen werden. Ind_N hat dann folgende Form:

$$\mathsf{Ind}_N(i) := a \cdot i + b \quad \text{mit} \quad a, b \in I\!R \quad \text{und} \quad a \neq 0 \tag{3.13}$$

Durch eine derartige Definition von Ind_N liegt die Umkehrfunktion der Diskretisierung Dis_N vor und mit den begrenzenden Stützstellen $s_0 := b$ und $s_{N-1} := a \cdot (N-1) + b$ ergibt sich das reelle Intervall $[s_0, s_{N-1}]$ als Definitionsbereich der entsprechenden Diskretisierungsfunktion.

3.3.2.5 Interner Ablauf der Diskretisierung

Der generelle Ablauf bei einer linearen Diskretisierung ist in Abbildung 3.5 (S. 46) wiedergegeben. Entweder kann zunächst die Anzahl der äquidistanten Stützstellen definiert werden (linker Zweig), oder alternativ dazu (rechter Zweig) wird zunächst die Quantisierungseinheit festgelegt. Anschließend wird davon die Anzahl der Stützstellen abgeleitet. Schließlich erfolgt nach der Bestimmung der Stellenanzahl für die Zahlendarstellung die Beurteilung, ob die Feinheit der gewählten Quantisierung nicht eine zu hohe Datenmenge bedeutet (vgl. Abschnitt 3.3.2.3, S. 46).

3.3.2.6 Diskretisierung beim Sinus-Beispiel

Im konkreten Fall des *Sinus*-Beispiels erfolgt eine lineare Indizierung mit $N := 100$ Stützstellen, die äquidistant auf den Definitionsbereich $\mathbf{D} := [0, \frac{\pi}{2}]$ verteilt sind. Mit der Quantisierungseinheit $\varepsilon := \frac{\pi}{2}/(N-1)$ ergibt sich folgende Indizierung:

$$
\begin{aligned}
\mathsf{Ind}_N(i) &:= \varepsilon \cdot i \quad \text{mit} \quad i \in \mathbf{I}_N \\
\text{Damit gilt:} \quad f_{\text{ind}}(i) &:= f_{\text{real}}\big(\varepsilon \cdot i\big) = \sin\left(\frac{\pi}{2} \cdot \frac{i}{N-1}\right)
\end{aligned}
\tag{3.14}
$$

Durch diese Indizierung erfolgte eine Diskretisierung der reellen Argumentenmenge. Die daraus resultierende Funktion ist in Abbildung 3.6 (S. 47) grafisch dargestellt. An den Achsen des Koordinatensystems sind neben den Bemaßungen der kontinuierlichen Achsenwerten auch die diskreten Werte angedeutet.

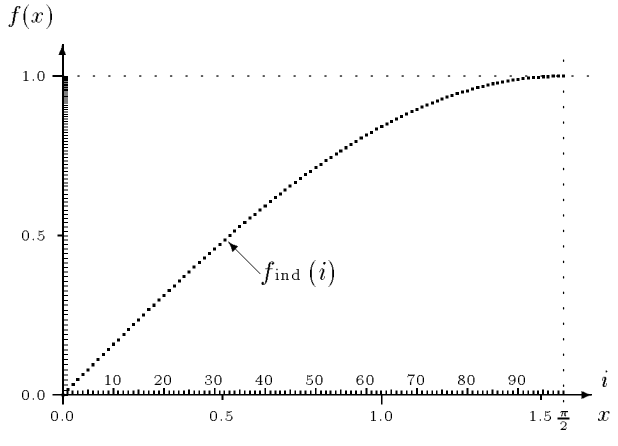

Abbildung 3.6: Diskretisierte *Sinus-Viertelwelle*

3.3.3 Stufe 3: Skalierung

Während bei der vorangegangenen Stufe Veränderungen der Argumentenmenge im Vordergrund standen, geht es hier um den Funktionswertebereich. Der zu Beginn durch die Definition gegebene Wertebereich ist bisher unverändert eine Teilmenge aus ganz $\mathbb{R}$. Demgegenüber erwartet die nachfolgende 4. Stufe einen positiv-reellen[5] Wertebereich, aus dem dann durch Rundung ein ganzzahliger Wertebereich entsteht. Dabei ist ein Verlust an Auflösung unvermeidbar, der allerdings hier durch die *skalare* Anpassungen der Funktionswerte in akzeptablen Grenzen gehalten werden kann. Die dafür eingesetzte Abbildung wird mit *Skalierung* bezeichnet.

3.3.3.1 Skalierung als Abbildung

Diese Abbildung dient zur Vorbereitung der nachfolgenden Abtrennung von Nachkommastellen. Zu diesem Zweck müssen die Funktionswerte skalar angepaßt werden (engl. *scaling*). Diese Abbildung sei deshalb mit Scl bezeichnet und wie folgt definiert:

$$\mathsf{Scl} : I\!R \rightarrow I\!R^{+} \quad \text{wobei} \quad I\!R^{+} := \{q \in I\!R \mid q \geq 0\} \tag{3.15}$$

Die bislang von f_{ind} evaluierten Funktionswerte werden somit durch Scl zunächst in einen positiven Wertebereich abgebildet, so daß anschließend die Auflösungsbegrenzung durchgeführt werden kann.

3.3.3.2 Skalierung in Festkommadarstellung

Zur Vermeidung eines zu großen Aufwandes bei der Repräsentation der Zahlenwerte (vgl. Abschnitt 3.1.4.2, S. 36) ist die Darstellung mit Festkomma (engl. *fixpoint*) am zweckmäßigsten. Die skalierte Funktion sei daher mit f_{fix} bezeichnet und stellt sich in folgender Weise dar:

$$\begin{aligned} f_{\text{fix}} &: I\!N \rightarrow I\!R^{+} \quad \text{wobei} \\ f_{\text{fix}}(i) &:= \mathsf{Scl}\Big(f_{\text{ind}}\,(i)\Big) \quad \text{und} \quad i \in \mathbf{I}_N \end{aligned} \tag{3.16}$$

Zur Begrenzung der Auflösung wird in dieser Stufe der Wertebereich skalar angepaßt. Dadurch wird das Maximum der Funktionswerte bestimmt und damit auch die Ausdehnung der Stellen in höherwertiger Richtung (vgl. Gleichung 3.3, S. 38). Somit ist das Maximum M ein charakteristisches Merkmal der vorgenommenen Skalierung; dies kann durch Hinzufügen eines entsprechenden Indexes, wie in Scl_M, deutlich gemacht werden.

3.3.3.3 Lineare Skalierung

Generell kann eine beliebige Funktion zur Skalierung verwendet werden. In der Regel kommen allerdings nur vergleichsweise triviale Skalierungsfunktionen zum Einsatz: Beispielsweise kann

[5] Zur Sicherstellung, daß bei der Spezifizierung das Vorzeichen unverändert erhalten bleibt, sollte es unabhängig vom skalaren Wert in einer PLA-Realisierung übernommen werden. Auf diese Weise bleibt es von den später noch vorzustellenden Modifikationen unbeeinflußt. Bei Vorzeichen-behafteten Werten bietet es sich deshalb an, das Vorzeichen abzuspalten, da es keine Auswirkung auf die skalare Beschreibung eines Zahlenwertes hat, sondern eher ein Merkmal desselben darstellt, welches durch das PLA in exakter Weise realisiert werden muß. Deshalb sollte es zunächst vor der Kodierung des Zahlenwertes abgespalten werden. Am Ende der Spezifizierung wird es dann wieder angefügt und geht in die PLA-Implementierung mit ein. Alternativ könnte das Vorzeichen auch durch eine nachfolgend erläuterte Verschiebung des Wertebereiches mit Hilfe der Skalierung eliminiert werden.

schon durch Multiplikation mit einem linearen Faktor $c \in I\!R$ die ursprüngliche Auflösung proportional verändert werden. Enthält der ursprüngliche Wertebereich negative Funktionswerte, so bewirkt das Addieren einer Konstanten $d \in I\!R$ die Repräsentation von negativen Zahlenwerten durch positive (vgl. Fußnote 5, S. 48). In dieser Form wird für die Skalierung eine lineare Funktion verwendet:

$$\mathsf{Scl}_M(q) := c \cdot q + d \quad \text{mit} \quad c, d \in I\!R \quad \text{und} \quad c \neq 0 \tag{3.17}$$

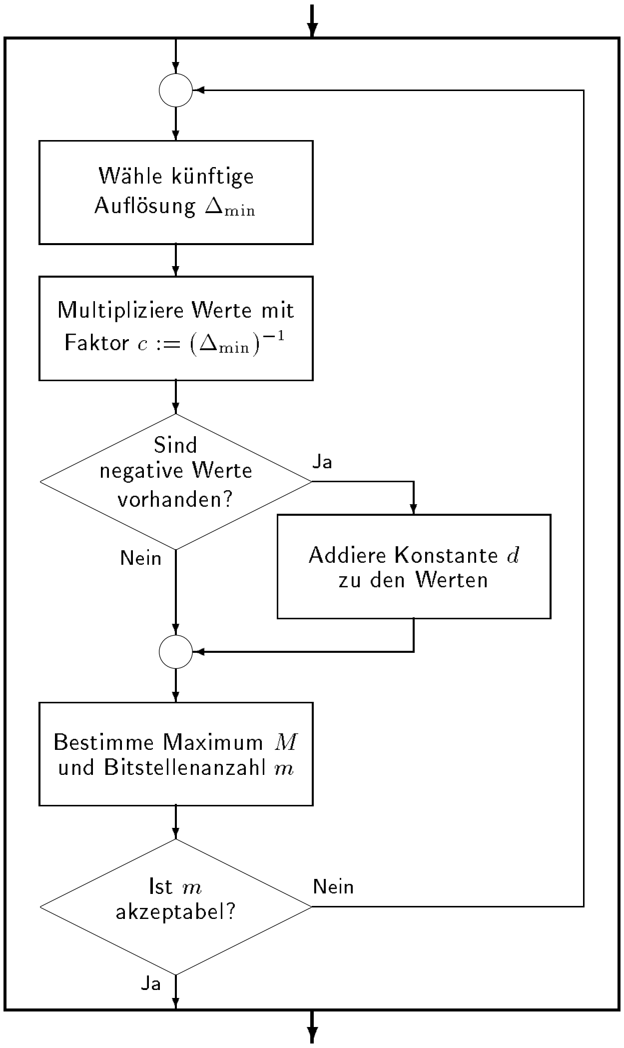

Abbildung 3.7: Ablaufdiagramm zur linearen Skalierung

3.3.3.4 Interner Ablauf der Skalierung

Den internen Ablauf einer linearen Skalierung zeigt Abbildung 3.7 (S. 49). Zunächst ist die künftige Auflösung $\Delta_{\min}$ zu wählen. Daraus ergibt sich der benötigte Linearfaktor $c := (\Delta_{\min})^{-1} \in I\!R$. Sind beim resultierenden Zwischenresultat f_{fix} negative Werte vorhanden, so muß eventuell zu allen Werten noch eine Konstante $d \in I\!R$ addiert werden, die mindestens so groß ist wie das negative Maximum. Mit dem sich schließlich ergebenden Maximum M über alle Funktionwerte kann nun die Anzahl der Bitstellen $m := \mathsf{Bits}(M)$ abgeleitet (vgl. Gleichung 3.3, S. 38) und beurteilt werden.

Eine Alternative zum dargestellten Ablauf besteht etwa in einer umgekehrten Reihenfolge: Zunächst wird die Bitstellenanzahl m gewählt und daraus das darstellbare Maximum $M := 2^m$ gebildet. Sofern keine negativen Funktionswerte (vgl. Fußnote 5, S. 48) vorhanden sind, ergibt sich anschließend aus dem Quotienten von bisherigem Maximum zum künftigen Maximum M die minimale Auflösung:

$$\Delta_{\text{min}} := \frac{\max\left(\{f_{\text{ind}}(i) \mid i \in \mathbf{I}_N\}\right)}{2^m} \tag{3.18}$$

Der benötigte Skalierungsfaktor c läßt sich dann durch $c := (\Delta_{\text{min}})^{-1}$ bestimmen.

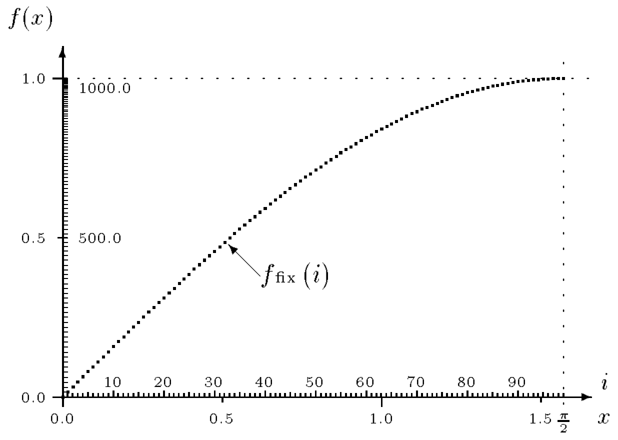

Abbildung 3.8: Skalierte *Sinus-Viertelwelle*

3.3.3.5 Skalierung beim Sinus-Beispiel

Beim konkreten Beispiel der *Sinus-Viertelwelle* liegt mit $0 \leq f_{\text{ind}}(i) \leq 1,\ i \in \mathbf{I}_N$ ein positiv reeller Wertebereich vor. Die zu wählende Auflösung wird mit $\Delta_{\text{min}} := 1\text{E-}3$ als ausreichend[6] akzeptiert. Bei der Skalierung wird nun der Skalierungsfaktor $c := (\Delta_{\text{min}})^{-1}$ gewählt, d. h. es wird mit $c := 1\text{E}3$ linear multipliziert. Da der Wertebereich von f_{ind} keine negativen Funktionswerte enthält und das künftige Maximum M wegen einer möglichst geringen Bitstellenanzahl klein gehalten werden soll, wird der Skalierungssummand $d := 0$ gewählt und bleibt somit ohne verschiebende Wirkung. Es ergibt sich nun mit $M = 1\text{E}3$ folgende Skalierung:

$$\text{Mit} \quad \mathsf{Scl}_{1000}(q) \quad := \quad 1000 \cdot q \quad \text{und mit} \quad q := f_{\text{ind}}(i) \in [0, 1] \tag{3.19}$$

[6] Häufig wird die Auflösung willkürlich in Größenordnungen von Zehnerpotenzen gewählt; vermutlich liegt ein Grund dafür in der Dezimaldarstellung der Funktionswerte. Ferner läßt sich der benötigte Skalierungsfaktor dadurch einfach bestimmen.

$$\text{folgt} \quad f_{\text{fix}}(i) := \mathsf{Scl}_{1000}\Big(f_{\text{ind}}(i)\Big)$$
$$:= 1000 \cdot \sin\left(\frac{\pi}{2} \cdot \frac{i}{N}\right) \quad \text{mit} \quad i \in \mathbf{I}_N$$

Die resultierende Funktion f_{fix} ist in Abbildung 3.8 (S. 50) wiedergegeben.

3.3.4 Stufe 4: Rundung

In dieser Stufe ist die Überführung der Funktionswerte von einem positiv-reellen in einen ganzzahligen Wertebereich durchzuführen. Dies bedeutet die Verringerung des Zahlenvorrates[7], womit zwangsweise eine Begrenzung der Auflösung verbunden ist. Die zu leistende Konvertierung wird hier zunächst mit der Abbildung *Rundung*[8] diskutiert.

3.3.4.1 Rundung als Abbildung

Durch die vorhergehende Skalierung sollte der Wertebereich so angepaßt worden sein, daß sich nun durch die Abtrennung der Nachkommastellen die gewünschte Auflösung (vgl. Abschnitt 3.1.4.4, S. 38) ergibt. Üblicherweise wird die dabei vorgenommene Überführung von einem reellen in einen ganzzahligen Wertebereich mit dem Begriff *Rundung* assoziiert. Im folgenden soll diese konvertierende Abbildung jedoch allgemeiner mit Con (engl. *Conversion*) benannt werden:

$$\mathsf{Con} : I\!R \to Z\!\!Z \tag{3.20}$$

Jedem Argument $q \in I\!R$ ordnet die Abbildung einen Näherungswert $w := \mathsf{Con}(q) \in Z\!\!Z$ zu; in welcher Weise der Näherungswert von dem Argumentenwert abhängt, ist dabei zunächst noch offen.

3.3.4.2 Ganzzahlige Funktion durch Konvertierung

Die durch die Abbildung Con gewonnenen Näherungswerte $w \in Z\!\!Z$ werden in ihrer Zuordnung zu den Argumenten $i \in \mathbf{I}_N$ der Funktion f_{fix} wiederum als Funktionswerte einer ganzzahligen (engl. *Integer*) Funktion aufgefaßt; diese sei mit f_{int} bezeichnet[9] und wie folgt definiert:

$$\begin{aligned} f_{\text{int}} &: I\!N \to Z\!\!Z \quad \text{wobei} \\ f_{\text{int}}(i) &:= \mathsf{Con}\Big(f_{\text{fix}}(i)\Big) \quad \text{mit} \quad i \in \mathbf{I}_N \end{aligned} \tag{3.21}$$

[7] Bekanntlich ist $Z\!\!Z$ *abzählbar unendlich*, während $I\!R$ *überabzählbar* ist.

[8] Der Begriff *Rundung* wird hier als Synonym für das *Abtrennen von Nachkommastellen* interpretiert; dabei ist die Art des Vorgehens zunächst noch offen. Es kann sich beispielsweise um das *Abbrechen* (truncate) oder um das *Runden* (round) handeln (vgl. Abschnitt 3.1.4.4, S. 38f).

[9] Man beachte, daß f_{ind} die indizierte f-Variante und f_{int} die ganzzahlige (engl. *Integer*) f-Variante bezeichnen.

3.3.4.3 Varianten zur Rundung

Die Auswertung von Con kann auf vielfache Weise geschehen; zunächst seien Varianten zur Rundung diskutiert, von denen vier Modi definiert werden (vgl. Abschnitt 3.1.4.4, S. 38f): Jedem Argument $q \in I\!R$ mit der Ziffernfolge $t_s t_{s-1} \dots t_0, \dots t_{-r}$ (vgl. Gleichung 3.1, S. 36) wird bei

- $\mathsf{Trunc} \equiv \langle\rangle$ (Abbrechen, vgl. Fußnote 3, S. 38) die Teilfolge $t_{-1} \dots t_{-r}$ der Nachkommastellen abgetrennt und die daraus resultierende Ganzzahl zugeordnet; es sei dafür definiert:

$$\langle q \rangle := t_s t_{s-1} \dots t_0 \tag{3.22}$$

- Floor[10] $\equiv \lfloor\ \rfloor$ (Abrunden) die nächst *kleinere* Ganzzahl zugeordnet; es sei dafür definiert:

$$\lfloor q \rfloor \quad := \quad \max\Big(\{b \in Z\!\!\!Z \mid b \leq q\}\Big) \tag{3.23}$$

$$\text{bzw. } \mathsf{Floor}(q) \quad := \quad \begin{cases} \mathsf{Trunc}(q) & : \quad q \geq 0 \\ \mathsf{Trunc}(q) - 1 : & \quad q < 0 \end{cases} \tag{3.24}$$

- Ceil[11] $\equiv \lceil\ \rceil$ (Aufrunden) die nächst *größere* Ganzzahl zugeordnet; es sei dafür definiert:

$$\lceil q \rceil \quad := \quad \min\Big(\{b \in Z\!\!\!Z \mid b \geq q\}\Big) \tag{3.25}$$

$$\text{bzw. } \mathsf{Ceil}(q) \quad := \quad \begin{cases} \mathsf{Trunc}(q) & : \quad q \leq 0 \\ \mathsf{Trunc}(q) + 1 : & \quad q > 0 \end{cases} \tag{3.26}$$

- $\mathsf{Round} \equiv []$ ((Nahe-)Runden) die *nächstliegende* Ganzzahl zugeordnet; es sei dafür definiert:

$$\mathsf{Round}(q) \quad := \quad \mathsf{Sgn}(q) \cdot \begin{cases} \mathsf{Floor}(|q|) & : \quad |q| - \mathsf{Trunc}(|q|) < 0.5 \text{ oder} \\ & \quad |q| - \mathsf{Trunc}(|q|) = 0.5 \\ & \quad \text{und } \mathsf{Trunc}(|q|) \text{ ist gerade} \\ \mathsf{Ceil}(|q|) & : \quad |q| - \mathsf{Trunc}(|q|) > 0.5 \text{ oder} \\ & \quad |q| - \mathsf{Trunc}(|q|) = 0.5 \\ & \quad \text{und } \mathsf{Trunc}(|q|) \text{ ist ungerade} \end{cases} \tag{3.27}$$

$$\text{wobei } \mathsf{Sgn}(q) \quad := \quad \begin{cases} +1 : & q > 0 \\ 0 : & q = 0 \\ -1 : & q < 0 \end{cases} \tag{3.28}$$

Zum Vergleich der hier dargestellten Modi sind einige Zahlenbeispiele in Tabelle 3.2 (S. 53) wiedergegeben. Handelt es sich bei den Argumenten lediglich um positiv reelle Zahlen ($q \in I\!R^+$), so weisen Abbrech- (Trunc) und Abrund-Modus (Floor) identische Resultate auf. Da dies für den Kontext der vorliegenden Arbeit im allgemeinen zutrifft, können diese beiden Modi als gegenseitige Synonyme verwendet werden.

[10] Englisch *Floor*: (Fuß-)Boden — im Sinne von „unterhalb gelegen".
[11] Englisch *Ceil*: (Raum-)Decke — im Sinne von „oberhalb gelegen".

Zahlenbeispiel $q \in I\!R$	Trunc $\langle q \rangle$	Floor $\lfloor q \rfloor$	Ceil $\lceil q \rceil$	Round $[q]$	nach Regel
$\pi/2 \approx 1.5714$	1	1	2	2	(b)
$\sqrt{2} \approx 1.4142$	1	1	2	1	(a)
$2/3 \approx 0.6666$	0	0	1	1	(b)
$-2/3 \approx -0.6666$	0	-1	0	-1	(b)
44.5	44	44	45	44	(c.1)
44.5001	44	44	45	45	(b)
45.5	45	45	46	46	(c.2)

Tabelle 3.2: Beispiele zu den vier Rundungsmodi

3.3.4.4 Rundung beim Sinus-Beispiel

Im vorliegenden *Sinus*-Beispiel wurde jeweils zur nächstliegenden Ganzzahl gerundet, d. h. es wurde das standardmäßige (Nahe-)Runden angewendet. Die resultierenden Werte sind in Tabelle 3.1 (S. 40) zahlenmäßig aufgeführt. Die grafische Wiedergabe der Funktion unterscheidet sich wegen der relativ hoch gewählten Anzahl von 1000 Abstufungen auf der vertikalen Achse nicht von der durch Abbildung 3.8 (S. 50) gegebenen.

3.3.4.5 Interner Ablauf bei der Rundung

Bei der Rundung muß zunächst der Modus bestimmt werden. Nachdem dann die Rundung entsprechend durchgeführt worden ist, liegt nun die ganzzahlige Funktion f_{int} vor. Wegen des geradlinig trivialen Ablaufs kann auf eine weitere Veranschaulichung verzichtet werden.

3.3.5 Stufe 5: Kodierung

In der letzten Stufe der Spezifizierung geht es um die Kodierung des ganzzahligen Wertebereiches von f_{int} in das binäre Dualsystem mit $p := 2$ (vgl. Gleichung 3.1, S. 36). Auf diese Weise erhält man die Repräsentation einer Funktion als Wahrheitstabelle (vgl. Abschnitt 1.2.3.1, S. 8), die als Ausgangsbasis für den Entwurf eines PLAs dient.

3.3.5.1 Anzahl der Bitstellen

Die Kodierung geschieht mit der Abbildung Bin_k (vgl. Gleichung 1.2, S. 6), die sowohl auf jedes Argument als auch auf dessen Funktionswert angewandt wird. Dabei ist k die Anzahl der erforderlichen Bitstellen, die bezüglich der

- **Argumente** durch die Diskretisierung in der 2. Stufe bestimmt wurde: Die reellen Elemente der Argumentenmenge werden mit Hilfe der Diskretisierung durch die Elemente einer endlichen Indexmenge $\mathbf{X} \subset I\!N$ identifiziert. Dabei wird die Anzahl der Stützstellen N festgelegt, die die Indexmenge $\mathbf{X}$ begrenzt; die Indexmenge läßt damit auch als $\mathbf{X} := \mathbf{I}_N$ (vgl. Gleichung 1.9, S. 7) schreiben. Nach Gleichung 1.8 (S. 7) ergibt sich die Stellenanzahl zu $n := \mathsf{Bits}(N-1)$, d. h. es ist $k := n$ zu wählen, denn es gilt $(N-1) \leq 2^n$. Die Indizes werden somit durch die Abbildung $\mathsf{Bin}_n : \mathbf{I}_{2^n} \to I\!B^n$ binär kodiert.

- **Funktionswerte** in der 3. Stufe durch die skalaren Anpassungen bestimmt wurde: Die endlich vielen Funktionswerte sind mit Hilfe von Skalierungs- und Rundungsabbildung in einen ganzzahligen Wertebereich $\mathbf{Y} \subseteq \mathbb{Z}$ transformiert worden. Damit existiert insbesondere auch ein Maximum $M := \max(\mathbf{Y})$, so daß sich die Anzahl der für die binäre Kodierung erforderlichen Bitstellen zu $m := \mathsf{Bits}(M)$ (vgl. Gleichung 1.8, S. 7) bestimmen läßt. Es ergibt sich $k := m$, denn es gilt $M \leq 2^m$. Die Funktionswerte werden somit durch die Abbildung $\mathsf{Bin}_m : \mathbf{I}_{2^m} \in \mathbb{B}^m$ binär kodiert.

Im allgemeinen strebt man für die Wahrheitstabelle ein kleines Format an (vgl. Abschnitt 1.2.2.3, S. 7), d. h. das Produkt $n \cdot m$ ist möglichst klein zu halten.

3.3.5.2 Kodierte Funktion als Abbildung

Die in den Funktionswerten binär kodierte Variante der Funktion f_{int} sei mit f_{bin} bezeichnet. Als letzte Zwischenvariante der Funktion f bewirkt sie folgende Abbildung:

$$f_{\text{bin}} : \mathbb{N} \to \mathbb{B}^m \tag{3.29}$$

Während die binäre Kodierung der Argumente von f_{bin} durch $\mathsf{Bin}_n(i),\ i \in \mathbf{I}_{2^n}$ gewonnen werden, sind die Funktionswerte von f_{bin} auf die folgende Weise zu evaluieren:

$$f_{\text{bin}}(i) := \mathsf{Bin}_m\Big(f_{\text{int}}(i)\Big) \tag{3.30}$$

Damit liegt nun die komplette, binäre Kodierung von f_{int} in Argument und Funktionswert vor; diese Binärdarstellung entspricht einer Wahrheitstabelle und sei daher mit f_{tab} bezeichnet:

$$f_{\text{tab}}\Big(\mathsf{Bin}_n\,(i)\Big) := \mathsf{Bin}_m\Big(f_{\text{int}}(i)\Big) \tag{3.31}$$

3.3.5.3 Kodierung beim Sinus-Beispiel

Im vorgestellten *Sinus*-Beispiel wurde die Argumentenmenge durch $N := 100$ Stützstellen indiziert, so daß die Indexmenge $\mathbf{I}_{100} := \{0, \ldots, 99\}$ gewählt wurde. Aus $n := \mathsf{Bits}(99)$ ergeben sich für die binäre Darstellung der Indizes $n := 7 = \lceil \log_2(99) \rceil$ Bitstellen. Die reellen Funktionswerte aus dem Intervall [0,1] wurden mit einem Linearfaktor von 1000.0 skaliert, so daß sich nach der Rundung ein ganzzahliges Maximum von $M := 1000$ ergab. Mit $m := \mathsf{Bits}(1000)$ sind nun für die binäre Darstellung der Funktionswerte $m := 10 = \lceil \log_2(1000) \rceil$ Bitstellen erforderlich.

Für die Wahrheitstabelle zum *Sinus*-Beispiel ergibt sich damit ein Format (vgl. Abschnitt 1.2.2.3, S. 7) von „2^7 Einträgen à 10 Bit“ bzw. 7×10. Im Anhang B (S. 157f) ist in Tabelle B.1 (S. 158) die resultierende, tabellarische Repräsentation von f_{int} in binär kodierter Form als f_{tab} aufgeführt. Während in den beiden linken Spalten der Index $i \in \mathbf{I}_{2^n}$ und der Funktionswert $f_{\text{int}}(i) \in \mathbb{Z}$ aufgeführt sind, werden in den beiden rechten Spalten der binär kodierte Index $\mathsf{Bin}_7(i) \in \mathbb{B}^7$ und der ebenfalls binär kodierte Funktionswert $\mathsf{Bin}_{10}\Big(f_{\text{int}}(i)\Big) = f_{\text{bin}}(i) \in \mathbb{B}^{10}$ aufgelistet.

Die Funktion f_{int} ist hier auf $\mathbf{I}_{100} := \{0, \ldots, 99\}$ definiert (vgl. Gleichung 3.21, S. 51). Eine Wahrheitstabelle umfaßt jedoch immer alle zu $\mathbb{B}^n$ gehörenden Argumente aus der Menge $\mathbf{I}_{2^n}$ (vgl. Abschnitt 1.2.3.1, S. 8). Da $\mathbf{I}_{100}$ davon aber nur eine Teilmenge ist ($\mathbf{I}_{100} \subseteq \mathbf{I}_{2^n}$), bleiben in der Wahrheitstabelle die letzten 28 Einträge für $i \in [100, 127]$ unbenutzt. Da f_{int} für $i \geq 100$ nicht definiert ist, handelt es sich bei f_{tab} um eine partiell definierte Logikfunktion (vgl. Gleichung 1.6, S. 6). Aus diesem Grund werden die betreffenden 28 Einträge in der Wahrheitstabelle komplett mit dem unbestimmten Logikwert $*$ aufgefüllt.

3.4 Interpretation der Sequenz

Die Spezifizierung beginnt durch die Definition der zu implementierenden Funktionalität in Stufe 1 der Sequenz. In Abhängigkeit vom jeweiligen Kontext der Problemstellung kann sich anfänglich allerdings eine andere Form der Funktionalität ergeben, als hier mit einer reell-kontinuierlichen Funktion f_{real} angenommen wurde. Die weiteren Zwischenresultate von f zeigen dafür mögliche Einstiegspunkte auf, die optional zur Verfügung stehen. Die 1. Stufe reicht in einem derartigen Fall weiter in die Sequenz hinein, so daß überflüssige Stufen übersprungen werden können.

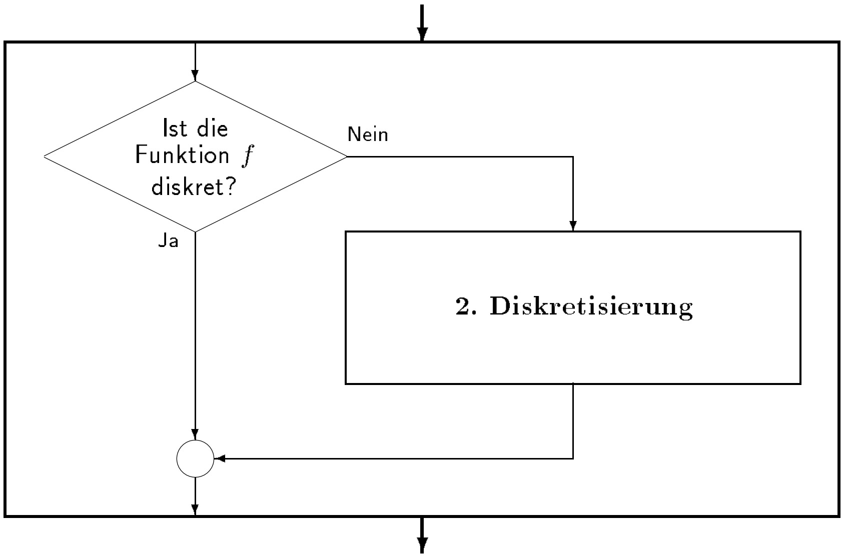

Abbildung 3.9: Umgehung einer Stufe der Sequenz am Beispiel der Diskretisierung

Beispielsweise kann häufig die 2. Stufe entfallen, da bereits aus dem Kontext der Problemstellung eine diskrete Funktion vorliegt. Wie in Abbildung 3.9 gezeigt, erübrigt sich in diesem Fall die Diskretisierung, und bereits f_{ind} bildet den Einstieg in die Spezifizierung.

Kapitel 4

Untersuchung von Spezifikationen

Die Spezifikation einer PLA-Tabelle ist im allgemeinen das Resultat einer 5-stufigen Sequenz, die im Kapitel 3 unter dem Begriff *Spezifizierung* diskutiert wurde. Wie dort bereits in Abschnitt 3.2.1 dargelegt wurde, sind Spezifikationen häufig für eine Realisierung als PLA ungeeignet, da entweder die PLA-Kosten eine unakzeptable Größenordnung erreichen oder aber der PLA-Entwurfsvorgang insgesamt abgebrochen werden muß (vgl. Abschnitt 2.2.3.2, S. 23). Bevor Schritte in Richtung der Modifikation einer Spezifizierung unternommen werden können, ist zunächst eine Relativierung der Spezifizierung, wie sie üblicherweise in der Praxis geschieht, notwendig.

In diesem Kapitel erfolgen eingehende Untersuchungen der Spezifizierung nach möglichen Ansatzpunkten für Modifikationen. Zusätzlich zur Darstellung einer Spezifizierung als 5-stufige Sequenz wie in Kapitel 3 ist ein struktureller Überblick über die Transformationen sinnvoll, die im Verlauf einer Spezifizierung an den f-Varianten vorgenommen werden. Der folgende Abschnitt 4.1 stellt die *funktionale Systematik* der Spezifizierung in Form einer schematischen Struktur dar. Als systematische Grundlage für die weitere Diskussion wird ein Modell vorgestellt, welches das Schema in einer vereinfachenden Weise in nur drei Schichten unterteilt. Auf mögliche Ansatzpunkte für Modifikationen in einer Spezifizierung weisen danach in Abschnitt 4.2 ab Seite 63 Aspekte zur *Unschärfe* und zu *Optionen* hin.

4.1 Funktionale Systematik der Spezifizierung

In den Abschnitten 3.2 und 3.3 erfolgte mit der Diskussion der *5-stufigen Sequenz* eine *prozedural* orientierte Darstellung für das Spezifizieren einer PLA-Tabelle. Eine weitere Perspektive öffnet sich bei einer *funktional* orientierten Betrachtungsweise der Transformationen, die zu den Varianten der Funktion f führen.

4.1.1 Funktionsvarianten bei einer Spezifizierung

Im Rahmen der Spezifizierung treten verschiedene Varianten der Funktion f auf. Sie entstehen als Zwischenresultate der Transformationsschritte, die durch die Abbildungen Dis bzw. Ind, Scl, Con und Bin hervorgerufen werden. Die in Kapitel 3 bereits im Detail, jedoch verstreut diskutierten f-Variante sind in Tabelle 4.1 (S. 58) zusammengestellt.

f-Variante	Erläuterung
$f_{\text{real}}: \; I\!R \to I\!R$	In der 1. Stufe wird die reelle Funktion f_{real} definiert.
$f_{\text{ind}}: \; I\!N \to I\!R$	Anschließend wird in der 2. Stufe durch Diskretisierung eine Transformation zur indizierten Funktion f_{ind} durchgeführt.
$f_{\text{fix}}: \; I\!N \to I\!R^+$	In der 3. Stufe bewirkt die Skalierung eine Anpassung des Wertebereiches, die zum Zwischenresultat f_{fix} führt.
$f_{\text{int}}: \; I\!N \to I\!N$	Durch Konvertierung der Funktionswerte von $I\!R^+$ nach $I\!N$ erzeugt Con in der 4. Stufe die f-Variante f_{int}.
$f_{\text{bin}}: \; I\!N \to I\!B^m$	Die Abbildung Bin_m kodiert in der 5. Stufe f_{int} zu f_{bin}.
$f_{\text{tab}}: \; I\!B^n \to I\!B^m$	Zusammen mit der Kodierung der Indizes durch Bin_n liegt nun die zum PLA-Entwurf verwendbare binäre Variante f_{tab} vor.

Tabelle 4.1: Zusammenstellung der f-Varianten bei einer Spezifizierung

4.1.2 Transformationen in der Spezifizierung

Einen Beitrag zur funktional orientierten Analyse der Spezifizierung liefert die Betrachtung derjenigen Abbildungen, die die Beziehungen zwischen den beteiligten f-Varianten herstellen. In der Darstellung der 5-stufigen Sequenz von Abschnitt 3.3 treten im Verlauf einer Spezifizierung hierbei folgende Abbildungen als Transformationen auf:

- Diskretisierung (vgl. Abschnitt 3.3.2, S. 44f)

$$\begin{array}{rlcccl} & \mathsf{Dis} & : & I\!R & \to & I\!N \quad \text{wobei gilt: } \mathsf{Dis} \equiv \mathsf{Ind}^{-1} \\ \text{z. B.} & \mathsf{Ind}_N & : & \mathbf{I}_N & \to & I\!R \end{array} \tag{4.1}$$

- Skalierung (vgl. Abschnitt 3.3.3, S. 47f)

$$\begin{array}{rlcccl} & \mathsf{Scl} & : & I\!R & \to & I\!R^+ \quad \text{wobei} \quad I\!R^+ := \{q \in I\!R \mid q \geq 0\} \\ \text{z. B.} & \mathsf{Scl}_M & : & I\!R & \to & \{q \in I\!R^+ \mid q \leq M\} \end{array} \tag{4.2}$$

- Konversion beispielsweise mittels Rundung (vgl. Abschnitt 3.3.4.1, S. 51f)

$$\begin{array}{rlcccl} & \mathsf{Con} & : & I\!R & \to & Z\!\!\!Z \\ \text{z. B.} & \mathsf{Con} & : & I\!R^+ & \to & I\!N \end{array} \tag{4.3}$$

- Kodierung (vgl. Abschnitt 3.3.5, S. 53f)

$$\begin{array}{rlcccl} & \mathsf{Bin} & : & I\!N & \to & I\!B^k \quad \text{mit} \quad k \in I\!N \\ \text{z. B.} & \mathsf{Bin}_m & : & \mathbf{I}_{2^m} & \to & I\!B^m \end{array} \tag{4.4}$$

Für jede dieser Abbildungen ist jeweils die Grundform und eine Beispielform angegeben. Bei Dis bzw. Ind und Scl sind die tiefgestellten Größen in der Beispielform angedeutet. Bei Con soll das Beispiel zeigen, daß Con auf positiv-reelle Zahlen angewendet auch nur positiv-ganze, also natürliche Zahlen liefert.

4.1.3 Schema der funktionalen Systematik

4.1.3.1 Grob-Schematisierung der funktionalen Abbildungen

In Tabelle 4.1 (S. 58) sind $f_{\text{real}} : \mathbb{R} \rightarrow \mathbb{R}$ als Anfangsvariante der Funktion f sowie $f_{\text{tab}} : \mathbb{B}^n \rightarrow \mathbb{B}^m$ als Endvariante aufgeführt. In Abbildung 4.1 (S. 59) wird der funktionale Zusammenhang zwischen beiden f-Varianten veranschaulicht: Die schematische Darstellung gibt funktionale Abbildungen als Kanten zwischen kreisförmig angedeuteten Mengen in Gestalt eines gerichteten azyklischen Graphen wieder.

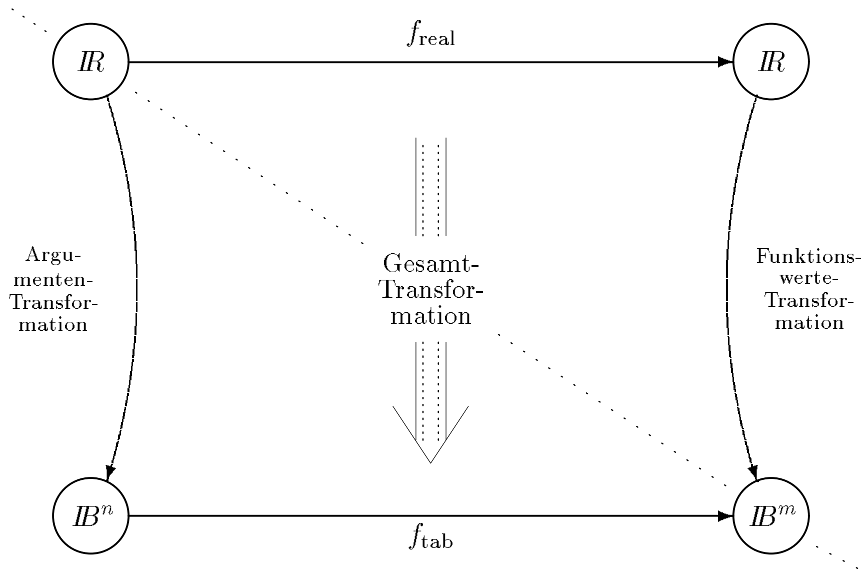

Abbildung 4.1: Schema zur Gesamt-Transformation

Zunächst verläuft oben die Abbildung der Anfangsvariante f_{real} als horizontale Kante von links nach rechts. Am unteren Ende befindet sich der Abbildungsschritt der Endvariante f_{tab}, ebenfalls horizontal von links nach rechts verlaufend. Zwischen beiden f-Varianten sind etliche Transformationen durchzuführen, die insgesamt in der vertikalen Richtung des in der Mitte angedeuteten Pfeils erfolgen.

An der linken bzw. rechten Seite sind die jeweils getrennt stattfindenden Transformationen für die Argumenten- bzw. Funktionswerte durch gekrümmte dünne Kanten vermerkt, was die Notwendigkeit einer weitergehenden Auflösung in die einzelnen Abbildungsschritte suggerieren soll.

Zusätzlich wird anhand der von links-oben nach rechts-unten verlaufenden Diagonale auf die Kommutativität im Graphen hingewiesen: Oberhalb verläuft der Pfad, ausgehend von reell-kontinuierlichen Argumenten, über die Abbildung f_{real} zu den reellen Funktionswerten. Er setzt sich mit den entsprechenden Transformationen zu den binär kodierten m-Bit Funktionswerten in der Wahrheitstabelle fort.

Der unterhalb der Diagonalen verlaufende Pfad führt insgesamt dieselbe Abbildung durch, jedoch läuft er, wiederum ausgehend von reell-kontinuierlichen Argumenten, stattdessen über die erforderlichen Transformationen zu den binär kodierten n-Bit Indizes der Wahrheitstabelle, die ihrerseits den Abbildungsschritt von f_{tab} repräsentiert.

4.1.3.2 Verfeinerung des Schemas mit allen Funktionsvarianten

Eine verfeinerte Veranschaulichung der funktionalen Zusammenhänge zwischen f-Varianten und Transformationen bietet Abbildung 4.2 (S. 60): Es sind darin alle in Abschnitt 3.3 (S. 43) beteiligten Funktions- bzw. Transformations-Abbildungen aufgelöst. Analog zum ersten Schema

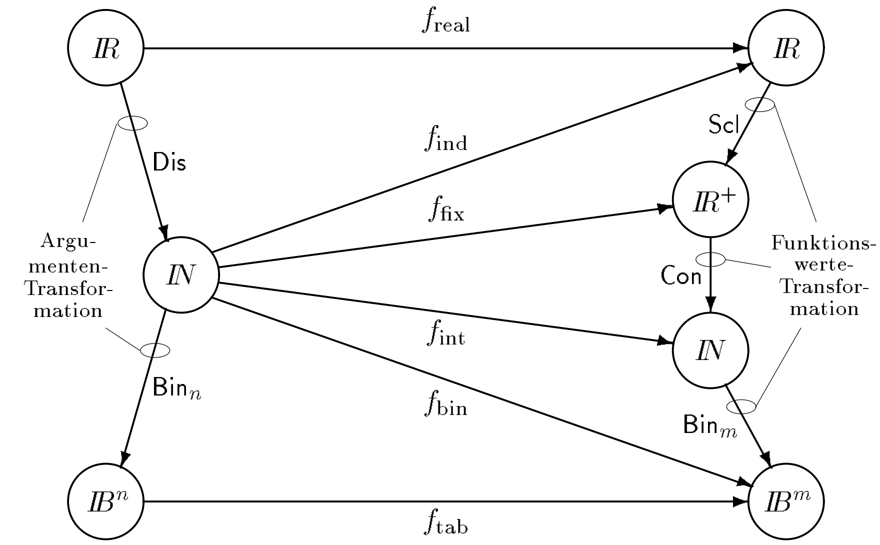

Abbildung 4.2: Schema der funktionalen Systematik

von Abbildung 4.1 (S. 59) werden Mengen in Gestalt von Kreisen angedeutet, zwischen denen gerichtete Kanten funktionale Beziehungen herstellen. Während sich auf der linken Seite die Argumentenmengen befinden, sind die Funktionswertemengen auf der rechten Seite angeordnet. Die von den Argumentenmengen zu den Funktionswertemengen orientierten, funktionalen Abbildungen stellen die Varianten der zu spezifizierenden Funktion f dar, und zwar von oben nach unten in genau der Abfolge, wie sie sich aus der Spezifizierungs-Sequenz (vgl. Abbildung 3.2, S. 42) ergibt; durch entsprechende Beschriftungen in Mitte der Abbildungskanten sind sie als f-Varianten markiert.

Durch die mittige Einengung soll der übergeordnete Aspekt erhalten bleiben, nämlich daß die an der obersten Kante angeordnete Funktionsvariante $f_{\text{real}} : \mathbb{R} \to \mathbb{R}$ im allgemeinen den Einstieg bildet und daß sich an der untersten Kante die Funktionsvariante $f_{\text{tab}} : \mathbb{B}^n \to \mathbb{B}^m$ als abschließendes Resultat ergibt. Dazwischen liegen alle erforderlichen Transformationen innerhalb der Spezifizierungs-Sequenz.

4.1.4 Modell in drei Schichten

Im Vorgriff auf die spätere Diskussion sei im folgenden zur Vereinfachung eine weitere Sichtweise anhand der funktionalen Systematik vorgestellt. Ausgehend von dem Schema der funktionalen Systematik in Abbildung 4.2 (S. 60) wird eine strukturelle Unterteilung anhand der vorliegenden Transformationen vorgenommen. Von größerem Interesse als die Transformationen an den freien Argumentenwerten sind dabei die auf die abhängigen Funktionswerte bezogenen Transformationen: Den Wandel von $\mathbb{R}$ über $\mathbb{R}^+$ und $\mathbb{N}$ nach $\mathbb{B}^m$ bewirken die drei Abbildungen Scl, Con und Bin.

Betrachtet man jede der drei Abbildungen als beliebig substituierbar, beispielsweise durch die später noch vorzustellenden, modifizierten Versionen, so bilden die jeweiligen Transformationsabbildungen drei voneinander disjunkte Sektionen. Sie seien künftig als *Schichten* bezeichnet und werden in Abbildung 4.3 (S. 61) angedeutet. Zusätzlich ist in den Schichten der prinzipielle „Pfad" für Transformationen angedeutet: Dies soll veranschaulichen, daß eine Spezifizierung innerhalb der Numerik-Schicht ihren Ausgangspunkt hat, dann die Konversions-Schicht passieren muß und schließlich in der Binär-Schicht endet.

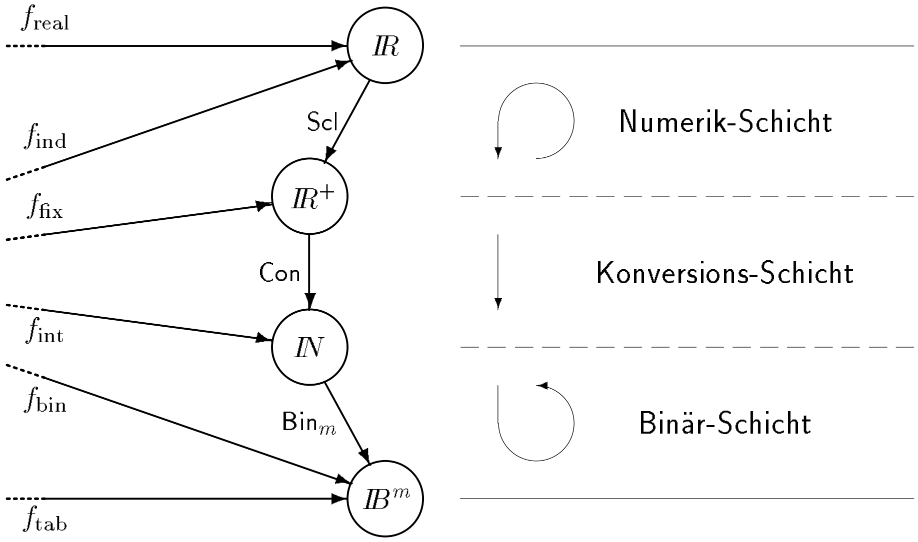

Abbildung 4.3: Modellierung der drei Schichten anhand des Schemas

4.1.4.1 Oben: Die Numerik-Schicht

Oberhalb der Kante von $f_{\text{fix}} : I\!N \rightarrow I\!R^+$ finden sich alle Transformationsabbildungen, die *numerisch* bezogene Ziele verfolgen. Neben der Skalierung mit Scl ist hierbei auch die Diskretisierung durch Dis mit einbezogen (vgl. Abbildung 4.2, S. 60), was allerdings von untergeordneter Bedeutung ist, da sie Veränderungen an den freien Argumentenwerten der Funktion vornehmen.

Zielsetzung der Transformationen in dieser Schicht ist die Repräsentation der Funktionalität in einer Folge von positiv-reellen Zahlen, von denen dann die Abbildungen der nächsten Schicht ausgehen. Die obere Schicht sei im folgenden als *Numerik-Schicht* bezeichnet.

4.1.4.2 Mitte: Die Konversions-Schicht

Die Konvertierung von reellem in ganzzahligen Wertebereich durch die Abbildung Con gestaltet sich bereits mit den in Abschnitt 3.3.4.3 (S. 52) erläuterten drei Modi Floor, Ceil und Round vielfältig. Im weiteren Verlauf dieses Kapitels kommen noch weitere Ansätze zur Diskussion, die ebenso eine Konvertierung der Funktionswerte von $I\!R^+$ nach $I\!N$ bewirken; sie führen jedoch im engeren Sinne keine Rundung durch.

Die am Anfang von Abschnitt 3.3.4.1 (S. 51) mit dem englischen Begriff *Conversion* weiter gefaßte Benennung der Abbildung Con ist insofern gerechtfertigt; für alle Transformationen von

$I\!R^+$ nach $I\!N$ sei im folgenden daher auch der Sammelbegriff *Konversion* gewählt. Im Schema nach Abbildung 4.3 (S. 61) markieren die Kanten der Abbildungen $f_{\text{fix}}: I\!N \to I\!R^+$ und $f_{\text{int}}: I\!N \to I\!N$ diese Schicht, in der die entsprechenden Konvertierungen stattfinden. Die mittlere Schicht sei künftig *Konversions-Schicht* genannt.

4.1.4.3 Unten: Die Binär-Schicht

Unterhalb der Kante von $f_{\text{int}}: I\!N \to I\!N$ seien alle Abbildungsschritte zusammengefaßt, die sich mit der Kodierung von dezimalen Ganzzahlen in *binäre* Dualzahlen befassen bzw. innerhalb der binären Darstellung weiterreichende Modifikationen vornehmen. Im weiteren Verlauf dieses Kapitels werden noch einige Ansätze vorgestellt, die die Existenz dieser Schicht rechtfertigen; sie sei künftig als *Binär-Schicht* bezeichnet.

4.1.5 Interpretation des Schemas

Das Schema in Abbildung 4.2 (S. 60) zeigt, daß die Spezifizierung im allgemeinen bei der obersten, horizontal orientierten Abbildung f_{real} beginnt und sich über die einzelnen Varianten von f fortsetzt. Falls anfänglich bereits eine der auf f_{real} folgenden f-Varianten vorliegt, so verlagert sich der Einstiegspunkt dementsprechend in Richtung der Gesamt-Transformation (vgl. Abschnitt 4.1, S. 59). Das Ziel aller Transformationen besteht schließlich in der untersten, horizontal orientierten Abbildung f_{tab}, die in Form einer Wahrheitstabelle zur Spezifikation eines PLAs dienen kann.

Im Vergleich zur 5-stufigen Sequenz aus Abschnitt 3.3 (S. 43ff) liegt der Einstiegspunkt hier bereits *hinter* der 1. Stufe, in der lediglich die Definition der Funktionalität erfolgt. Bezüglich des Kontextes der weiteren Diskussion wird die Definition der Funktionalität damit als extern betrachtet. Dieses kommt auch der in der Praxis vorherrschenden Situation beim Spezifizieren einer PLA-Tabelle am nächsten. Die weitere Diskussion sei daher auf den Umfang begrenzt, den die drei Schichten des Schemas beschreiben. Als Konsequenz daraus ist bei den noch folgenden Untersuchungen die Betrachtung der Definitionstufe nicht notwendig.

4.2 Unschärfe und Optionen bei der Spezifizierung

Das im Abschnitt 4.1.4 (S. 60) vorgestellte Schichtenmodell zur Spezifizierung bietet nachfolgend eine Basis für die Untersuchung der Transformationen, bezogen auf Aspekte zur *Unschärfe* und *Optionen*. Zuvor sei jedoch auf die Begriffsbildung näher eingegangen.

4.2.1 Begriffsbildung: „Unschärfe“ und „Optionen“

Der Auftraggeber als potentieller Urheber einer Spezifikation (vgl. Abschnitt 3.2.1, S. 41f) erhebt im allgemeinen den Anspruch, daß die von ihm spezifizierte PLA-Tabelle exakt erarbeitet sei und daß sie demzufolge unverändert vom Entwerfer umgesetzt werden müsse.

Allerdings weisen eigene Untersuchungen in [Kra-94] auf Ungenauigkeiten bzw. Wahlmöglichkeiten bei einer *ad hoc* erstellten Spezifikation hin, die offenbar nicht im Bereich der Intensionen des Auftraggebers liegen. Insofern handelt es sich in beiden Fällen um unauffällige Erscheinungen, die den einzelnen Stufen anhaften.

Unschärfe ist inhärent

Der Begriff *Unschärfe* soll in diesem Kontext jene Art von Ungenauigkeiten suggerieren, die bei der standardmäßigen, also unmodifizierten Spezifizierung einer PLA-Tabelle ohnehin vorhanden sind. Das Attribut *inhärent* weist darauf hin, daß von jenen Ungenauigkeiten bereits die elementarsten Operationen betroffen sind; insofern ergibt sich damit das Problem der Unvermeidbarkeit von Unschärfe.

Optionen sind verdeckt

Mit dem Begriff *Optionen* sind diejenigen Wahlmöglichkeiten gemeint, die bei den internen Entscheidungspunkten existieren. Sie werden nicht selten jedoch in geradezu selbstverständlicher Weise mit standardmäßigen Entscheidungen befriedigt und dabei auch nicht näher hinterfragt oder zur Diskussion gestellt.

Beide Begriffe dienen in Kapitel 5 zur Ausrichtung von Methoden, die Modifikationen an einer Spezifizierung vornehmen können. In der Summe werden sie alle von dem Leitmotiv geprägt, daß der Einsparungseffekt bei den PLA-Kosten schwerer wiegt, als die zu akzeptierenden Abweichungen. In der nachfolgenden Diskussion werden Erkenntnisse, bezogen auf die beiden Begriffe, jeweils bei ihrem Auftreten im Kontext, optisch abgesetzt und am Ende des Kapitels in einer Übersicht zusammengestellt.

4.2.2 Untersuchung der Numerik-Schicht

Beim Diskretisieren ergeben sich aus dem Prinzip der Quantisierung, wie es in Abschnitt 3.3.2.1 (S. 44f) erläutert wurde, unvermeidbare Abweichungen zwischen dem Funktionswert an einer Stützstelle und den durch ihn repräsentierten Werten.

4.2.2.1 Fehler beim Quantisieren

Das Ausmaß dieser zwangsweise entstehenden Abweichungen beschreibt der Quantisierungsfehler aus Gleichung 3.9 (S. 44). Anhand des *Sinus*-Beispiels aus Abschnitt 3.1.5 (S. 39) sei die lineare Diskretisierung, wie sie in Abschnitt 3.3.2.6 (S. 47) mit den $N := 100$ äquidistanten Stützstellen $s_i,\ i \in \mathbf{I}_N$ vorgenommen wurde, auf den Quantisierungsfehler hin untersucht. Beispielsweise gibt Tabelle 4.2 (S. 64) die Situation an der 11. Stützstelle $s_{10} := \varepsilon \cdot 10$ wieder: Links stehen Größen, in der Mitte deren Werte und rechts ein Hinweis zur Berechnung. Als erste Zeile ist die Quantisierungseinheit angegeben, als nächstes ist die Stützstelle aufgeführt. In den nächsten beiden Zeilen ist in der Mitte die Aufspaltung des Quantisierungsbereiches, zuerst für die Argumente und darunter für die Funktionswerte, angedeutet. Als letztes ist der resultierende Quantisierungsfehler aufgeführt.

<table>
<tr><td>Quantisierungseinheit</td><td colspan="3">$\varepsilon \approx 0.0158666$</td><td>$\varepsilon := \frac{\pi}{2}/(N-1)$</td></tr>
<tr><td>11. Stützstelle ($i := 10$)</td><td colspan="3">$s_{10} \approx 0.158666$</td><td>$s_i = \mathrm{DIS}_N(i) := \varepsilon \cdot i$</td></tr>
<tr><td>repräsentierte x-Umgebung</td><td>$s_{10} - \varepsilon/2$
≈ 0.150733</td><td>s_{10}
≈ 0.158666</td><td>$s_{10} + \varepsilon/2$
≈ 0.166600</td><td>vgl. Abschnitt 3.3.2.1 (S. 44)</td></tr>
<tr><td>Funktionswerte $f_{\mathrm{real}}(x)$</td><td>≈ 0.150163</td><td>≈ 0.158001</td><td>≈ 0.165830</td><td>vgl. Gleichung 3.7 (S. 43)</td></tr>
<tr><td>Quantisierungsfehler</td><td colspan="3">$\delta(s_{10}) \approx 0.015667$</td><td>vgl. Gleichung 3.9 (S. 44)</td></tr>
</table>

Tabelle 4.2: Quantisierungsfehler am Beispiel der 11. Stützstelle im *Sinus*-Beispiel

Zur Interpretation des geschilderten Beispiels ist anzumerken, daß der Funktionswert $f_{\mathrm{real}}(s_{10}) \approx 0.158001$ (4. Zeile, 3.Spalte in Tabelle 4.2) den Verlauf des Funktionsgraphen im Bereich von etwa $0.150163 \div 0.165830$ (Bereich des Quantisierungsfehlers $\delta(s_{10})$) repräsentiert. Auf den Funktionswert der Stützstelle bezogen, bedeutet dies einen Toleranzbereich von etwa $\pm 5\%$. In Relation zur $N := 100$ erscheint dieser Wert unerwartet hoch, da eine derartige Stützstellenanzahl schon eine relativ hohe Feinheit bzw. Genauigkeit suggeriert. In der Konsequenz aus dem gezeigten Fall ergibt sich die folgende Unschärfe U-1 (S. 64).

Unschärfe U-1: Quantisierungsfehler an jeder Stützstelle

Bei der Diskretisierung durch Quantisieren entsteht an jeder Stützstelle ein individueller Quantisierungsfehler, wie er in Abschnitt 3.9 (S. 44) diskutiert wurde. Dadurch ist der Funktionswert an der jeweiligen Stützstelle mit einer unvermeidbaren Unschärfe behaftet.

4.2.2.2 Anzahl der Stützstellen

In der Regel beeinflussen Details aus der Problemstellung die Anzahl der Stützstellen. Sie sollte einerseits möglichst groß sein, damit die Fehler durch die Quantisierung gering bleiben. Andererseits bewirkt eine Vergrößerung der Anzahl ein Anwachsen der Datenmenge. Nicht selten wird die Anzahl der Stützstellen in Schritten von 10er-Potenzen gewählt. Alternativ dazu ließe sich die folgende Option O-1 (S. 65) anwenden.

Option O-1: Anzahl der Stützstellen als 2er-Potenz
In der Abwägung des Kompromisses zwischen einer großen Datenmenge und kleinen Quantisierungsfehlern wird die Anzahl der Stützstellen festgelegt. Nicht selten wird sie dabei in Schritten von 10er-Potenzen variiert. Die tabellarische Repräsentation von Funktionswerten in einer binären Wahrheitstabelle bedeutet allerdings eine Anzahl in Form einer 2er-Potenz. Im Hinblick auf wenige unbenutzte Einträge in der Wahrheitstabelle sollte die Stützstellenanzahl N in Schritten von 2er-Potenzen gewählt werden, d. h. für $N \in I\!N$ sollte gelten: $\exists\, k \geq 1 : \; N = 2^k, \; k \in I\!N \setminus \{0\}$

Beispiel zur Option O-1

Beim *Sinus*-Beispiel aus Abschnitt 3.1.5 (S. 39) könnte eine Modifikation bei der Festlegung der Stützstellenanzahl N erfolgen: Sie könnte von $N = 100$, entsprechend $7 := \mathsf{Bits}(100)$ Bitstellen, ohne eine Vergrößerung der Wahrheitstabelle auf $N := 128 = 2^7$ erhöht werden. Gegenüber dem vorherigen Zustand bleibt der Formatparameter $n = 7$ hierbei unverändert, die PLA-Tabelle wäre aber vollständig[1] ausgenutzt.

4.2.2.3 Redundanz bei den Stützstellen

Nicht selten besitzt eine zu realisierende Funktion die Eigenschaft der Periodizität, d. h. im Abstand einer Periodenlänge wiederholen sich Funktionswerte. Je nach Festlegung des Definitionsbereiches kann dies Redundanz bedeuten, so daß in manchen Fällen die folgende Option O-2 (S. 65) besteht.

Option O-2: Redundanz aus der Stützstellenfolge entfernen
Eine Periodizität in der Folge der Stützstellen kann zur Symmetrie von Abschnitten der Folge führen. Es können sich so Wiederholungen identischer bzw. umgekehrter Teilfolgen ergeben. Derartige Teilfolgen bedeuten Redundanz und vergrößern unnötig die Datenmenge bei der Realisierung. Eine Abhilfe besteht in entsprechend ausgelegten Auslesemechanismen bei der späteren Beschaltung des PLAs, so daß die entfernten Teilfolgen auf die vorhandenen abgebildet werden.

[1] Auf die PLA-Kosten bezogen muß allerdings angemerkt werden, daß hierdurch im allgemeinen *keine* Reduzierung (Senkung der Termanzahl) erzielt werden kann: Da sich nun in den vorher nicht genutzten 28 Einträgen der Wahrheitstabelle zusätzlich relevante Daten befinden, wird in der Regel die Anzahl der Terme erhöht. Bezüglich der Algorithmen in der Logik-Minimierung sind vollständig mit $*$ (Don't-Care) aufgefüllte Einträge sehr günstig für eine effektive Minimierung.

Beispiel zur Option O-2

Das *Sinus*-Beispiel aus Abschnitt 3.1.5 (S. 39) demonstriert, wie Redundanz durch Spiegelung einer Teilfolge von Funktionswerten vermieden werden kann: Lediglich eine Viertelwelle der *Sinus*-Schwingung soll in der PLA-Tabelle implementiert werden, obwohl möglicherweise die volle Schwingung für die Problemlösung benötigt wird. Die fehlenden Viertelwellen können beim Auslesen des PLAs durch entsprechendes Abbilden auf die vorhandene bereitgestellt werden.

4.2.2.4 Variation des Stützstellenabstandes

In der Regel wird wie in Abschnitt 3.3.2.4 (S. 46) linear diskretisiert, d. h. nachdem die Anzahl der Stützstellen festgelegt ist, werden diese äquidistant auf den Definitionsbereich der Funktion verteilt. Eine Ergänzung kann sich durch die folgende Option O-3 (S. 66) ergeben.

Option O-3: Veränderlicher Abstand der Stützstellen

Im allgemeinen ist in der Problemlösung nicht festgelegt, auf welche Weise die Diskretisierung erfolgen soll: So ist beispielsweise ungeklärt, ob der Abstand zwischen den Stützstellen verändert werden kann oder ob er konstant bleiben soll. Da die Diskretisierung wie alle anderen Anpassungen generell auch in den umliegenden Systemeinheiten berücksichtigt werden muß, kann das Variieren des Abstandes zwischen den Stützstellen den Realisierungsaufwand teilweise erheblich steigern; daher ist diese Option nur in Ausnahmefällen zu empfehlen.

Beispiel zur Option O-3

Beispielsweise bietet sich dies bei Funktionen an, bei denen eine lineare Diskretisierung zu vielen gleichgroßen Funktionswerten führt bzw. die Steigung des Graphen abschnittsweise sehr gering ist. Sofern es von der Systemumgebung zu berücksichtigen wäre, können in solchen Fällen Stützstellen entfallen. Im Vergleich zur linearen Diskretisierung (vgl. Gleichung 3.14, S. 47) beim *Sinus*-Beispiel aus Abschnitt 3.3.2.6 (S. 47) mit $N := 100$ Stützstellen könnte sich beispielsweise folgende nicht-lineare Diskretisierung bzw. Indizierung ergeben: Ab der 68. Stützstelle ($i := 67$) wird die Quantisierungseinheit verdoppelt, d. h. es wird mit dem doppelten Abstand zwischen den Stützstellen diskretisiert, was u. a. die Anzahl der Stützstellen für denselben Definitionsbereich von $[0, \frac{\pi}{2}]$ auf 84 verringert.

In Anlehnung an die in Abbildung 3.6 (S. 47) gezeigte Linear-Diskretisierung gibt Abbildung 4.4 (S. 67) die nicht-linear diskretisierte Sinus-Funktion wieder. Die Aufteilung der zwei Abschnitte $i \in \{0, \ldots, 67\}$ und $i \in \{68, \ldots, 83\}$, die jeweils linear diskretisiert sind, geschah willkürlich nach einer 2/3-Schätzung. Die hohe Konzentration der Werte am oberen Ende der $f(x)$-Achse in Abbildung 3.6 (S. 47) ist bereits durch diese geringfügige Modifikation an der Spezifizierung erkennbar reduziert.

4.2.2.5 Erfassen ausgezeichneter Punkte

Aus der Problemstellung heraus wird der Definitionsbereich in der 1. Stufe der Spezifizierung bestimmt. Je nach den Eigenschaften der zu realisierenden Funktion existieren Punkte im Graphen, die von speziellem Interesse für die Problemlösung sind. Für derartig ausgezeichnete

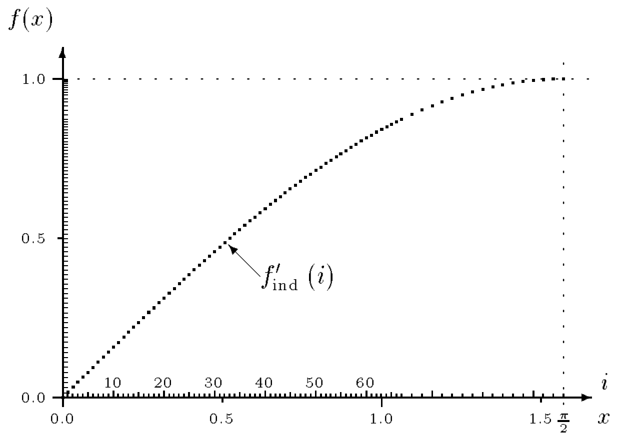

Abbildung 4.4: Nicht-linear diskretisierte *Sinus-Viertelwelle*

Punkte wird dann eine möglichst direkte Erfassung durch Stützstellen angestrebt, wie z. B. bei Nulldurchgängen oder Extremwerten, die nicht selten Randwerte des Definitionsbereiches sind. Eine *ad hoc* angelegte Diskretisierung geschieht üblicherweise so, daß die Randwerte durch Stützstellen direkt erfaßt werden. Für eventuell dazwischenliegende Extremwerte oder Polstellen (asymptotische Unstetigkeiten) ist es schon weitaus problematischer, die direkte Erfassung durch Stützstellen sicherzustellen. Betrachtet man die Festlegung jeder einzelnen Stützstelle auf diesem Hintergrund, so ergibt sich in den meisten Fällen die Tatsache von Unschärfe U-2 (S. 67).

Unschärfe U-2: Stützstellen werden zufällig festgelegt

Bei der Diskretisierung geschieht die Auswahl der Stützstellen in der Regel zufällig. Bestenfalls für speziell ausgezeichnete Funktionswerte wird darauf geachtet, daß sie durch eine entsprechende Diskretisierung direkt durch eine Stützstelle erfaßt werden.

Beispiel zur Unschärfe U-2

Im Graph einer *Sinus*-Funktion kämen als derartig ausgezeichnete Punkte etwa die folgenden in Betracht:

- Nullstellen: $\{x \mid x := \pi \cdot k,\ k \in \mathbb{Z}\}$
- Extremwerte: $\{x \mid x := \frac{\pi}{2} \cdot (2k+1),\ k \in \mathbb{Z}\}$
- ggf. Stellen mit $\frac{1}{2}$ als Funktionswert: $\{x \mid f_{\mathrm{real}}(x) = \frac{1}{2}\}$,
  z. B. $x \in \{\frac{\pi}{6}, \frac{5\pi}{6}, \frac{7\pi}{6}, \frac{11\pi}{6}, \frac{13\pi}{6}, \ldots\}$

Bei dem *Sinus*-Beispiel aus Abschnitt 3.1.5 (S. 39) wurden lediglich die Nullstelle bei $x := 0$ und der Extremwert von $+1$ bei $x := \frac{\pi}{2}$ berücksichtigt.

4.2.2.6 Fehler in der Arithmetik

Die arithmetischen Operationen, die im Zuge der Abbildung Scl an den Funktionswerten durchgeführt werden, sind je nach verwendeter Arithmetik mit Fehlern behaftet. Nach dem IEEE-

Standard[2] existieren mehrere Arithmetikformate, wie sie z. B. in Programmiersprachen wie *C* [Ker*88, Bar-89, Pur-93], aber auch in mathematischen Coprozessoren zum Einsatz kommen; Tabelle 4.3 (S. 68) gibt über Wertebereiche und Genauigkeitsaspekte einen Überblick.

IEEE-Format	C-Datentyp	Wertebereich	Genauigkeit
single real	`float`	10E-38 ÷ 10E+38	32 Bit ⇝ 6-7 Dezimalen
double real	`double`	1.7E-308 ÷ 1.7E+308	64 Bit ⇝ 15 Dezimalen
double extended	`long double`	10E-4932 ÷ 10E+4932	80 Bit ⇝ 19 Dezimalen

Tabelle 4.3: Übersicht der Arithmetikformate nach IEEE

Beispielsweise entspricht dem Arithmetikformat mit 80 Bit (aufgeteilt in zwei Vorzeichen, Mantisse und Exponent) eine Genauigkeit von etwa 19 Dezimalstellen, d. h. die minimale Auflösung liegt bei etwa 1E-19. Aus der Existenz des Arithmetikfehlers folgt die begrenzte Genauigkeit, was durch die folgende Unschärfe U-3 dokumentiert wird.

Unschärfe U-3: Systematische Arithmetikfehler

Bei allen Arithmetikoperationen fließen zwangsläufig numerische Fehler in die resultierenden Werte ein. Da alle Funktionswerte gleichermaßen davon betroffen sind, handelt es sich hierbei um einen systematischen Fehler.

Beispiel zur Unschärfe U-3

Beim *Sinus*-Beispiel wurden die Funktionswerte in der doppelten Genauigkeit (C-Datentyp `long double`, vgl. Tabelle 4.3, S. 68) berechnet. An der Stützstelle für $i = 33$ ergab sich als Argument $\mathsf{Ind}_{100}(33) := \frac{\pi}{6}$. Die Sinusfunktion hat an der Stelle $x = \pi/6$ (entsprechend 30° Winkelgrad) den Wert $\sin(\frac{\pi}{6}) = \frac{1}{2}$, d. h. der Funktionswert an der Stützstelle $i = 33$ ist $f_{\text{ind}}(33) := \frac{1}{2}$. Durch Fehler in der IEEE-gemäßen Arithmetik entstand bei der testweise berechneten Differenz von $(0.5 - f_{\text{ind}}(33))$ aber nicht 0.0 als Ergebnis, sondern ein geringer Wert von etwa $-5\text{E} - 20$. Damit lag hier eine Genauigkeit in der Größenordnung von 1E-19 vor, die dazu führte, daß $f_{\text{ind}}(33) < 0.5$ galt. Die Auswertung des Ausdrucks $\lfloor f_{\text{fix}}(33) \rfloor := 1000 \cdot f_{\text{ind}}(33)$ deckte schließlich mit dem Resultat 499 (gegenüber dem erwarteten Wert 500) den Arithmetikfehler auf.

4.2.2.7 Eigenschaften der Skalierungsabbildung

Der Arithmetikfehler bewegt sich meistens in einer nur geringen Größenordnung, so daß er in der Regel vernachläßigt werden kann. Bei der Skalierung kommt diesbezüglich der Begrenzung der minimalen Auflösung $\Delta_{\min}$ eine entscheidende Bedeutung zu (vgl. Abschnitt 3.3.3.2, S. 48). Sie ist ohnehin im Rahmen einer Spezifizierung erforderlich und sollte so ausgelegt sein, daß die störenden Arithmetikfehler nicht mehr in Erscheinung treten.

Ferner ist grundsätzlich eine Skalierung zu empfehlen, die den Wertebereich der Funktionswerte so manipuliert, daß diese später als binär kodierte Einträge einer Wahrheitstabelle den durch die

[2] IEEE: *Institute of Electrical and Electronic Engineers*; amerikanischer Fachverband, der sich hauptsächlich mit Normungsaufgaben befaßt, die auch international Beachtung finden.

Binärdarstellung bedingten Wertebereich ganz ausnutzen. Das von der Skalierung her festgelegte Maximum M sollte daher ein 2er-Potenzwert sein, d. h. $M := 2^k,\ k \in I\!N$. Mit Gleichung 3.18 (S. 50) wird gezeigt, wie M ausgehend von der gegebenen Anzahl m der Bitstellen bestimmt werden kann.

Sowohl die *Begrenzung der Auflösung* als auch die *Skalierung auf ein 2er-Potenz-Maximum* sollten bei der Bestimmung der Skalierunsabbildung Berücksichtigung finden; beides wird durch die folgende Option O-4 (S. 69) zusammengefaßt.

Option O-4: Bestimmung der Skalierungsabbildung

Die Begrenzung der minimalen Auflösung sollte unter Berücksichtigung von Unschärfe U-3 (Systematische Arithmetikfehler) geschehen, d. h. als Konsequenz daraus sollte die künftige minimale Auflösung $\Delta_{\min}$ durch die Skalierung deutlich größer als der systematische Arithmetikfehler festgelegt werden. Außerdem sollte sich aus der Skalierung als Maximum möglichst ein 2er-Potenzwert ergeben.

Beispiel zur Option O-4

Im Beispiel zur Unschärfe U-3 (S. 68) trat ein Arithmetikfehler in der Größenordnung von 1E-19 auf. Bei der Spezifizierung in Abschnitt 3.1.5 (S. 39) findet Option O-4 insoweit Anwendung, als daß dort mit einem Faktor von 1E3 linear skaliert wird; anschließend werden durch Rundung die Nachkommastellen abgetrennt. Auf diese Weise sind praktisch alle Arithmetikfehler eliminiert.

Von seiten der Problemstellung wird für die Wahl der Skalierungsabbildung in der Regel keine Festlegung getroffen. Dennoch geschieht sie im allgemeinen linear, was allerdings in manchen Fällen ungünstig sein kann: Bei einer sehr stark wachsenden Funktion resultiert beispielsweise ein großes Maximum, was bei der Kodierung der binären Einträge zu einer hohen Anzahl an Bitstellen führt. Für alternative Skalierungsabbildungen besteht deshalb mit der folgenden Option O-5 (S. 69) eine weitere Möglichkeit.

Option O-5: Nicht-lineare Skalierung

Mit Hilfe einer beispielsweise logarithmischen Skalierung kann das resultierende Maximum einer stark wachsenden Funktion deutlich geringer skaliert werden und somit eine Einsparung an Bitstellen erzielt werden. Allerdings muß die logarithmische Skalierung, wie alle im Rahmen einer Spezifizierung vorgenommenen Anpassungen, von der Systemumgebung des daraus entworfenen PLAs berücksichtigt werden. Im allgemeinen erfordert eine logarithmische Skalierung hier mehr Aufwand als eine lineare.

4.2.3 Untersuchung der Konversions-Schicht

Die Transformationsabbildung Con dient zur Konvertierung der Funktionswerte von einem reellen in einen ganzzahligen Wertebereich. Während bei der Rundung durch die Berücksichtigung der Regeln überwiegend logische[3] Operationen im Vordergrund stehen, sind dabei neben ggf. vorhandenen Optionen ebenso auch Aspekte zur Unschärfe relevant.

4.2.3.1 Fehler bei der Rundung

Die Rundung am Komma, wie sie in Abschnitt 3.3.4.1 (S. 51) durch die drei[4] Modi Floor (Abrunden), Ceil (Aufrunden) und Round (Nahe-Runden) beschrieben ist, ordnet der reellen Zahl $q \in I\!R$ entweder die nächstkleinere Ganzzahl $\lfloor q \rfloor \in Z\!\!Z$ oder die nächstgrößere Ganzzahl $\lceil q \rceil \in Z\!\!Z$ zu. Welche der beiden gewählt wird, hängt vom verwendeten Modus ab. Diese unterscheiden sich voneinander durch die jeweils zu berücksichtigende Regel (vgl. Abschnitt 3.3.4.3, S. 52). Den dabei auftretenden Fehler dokumentiert die folgende Unschärfe U-4 (S. 70).

Unschärfe U-4: Rundungsfehler je nach Rundungsmodus
Bei der Rundung wird ein reeller Wertebereich in einen ganzzahligen konvertiert. Für jeden Funktionswert wird dazu ein Näherungswert gewonnen, wobei zwangsweise ein individueller Rundungsfehler (vgl. Abschnitt 3.5, S. 39) entsteht. Dieser hängt vom jeweils gewählten Rundungsmodus (vgl. Abschnitt 3.3.4.3, S. 52) ab.

4.2.3.2 Vergleich der Rundungsmodi

Die beiden Modi Floor und Ceil arbeiten trivial: es wird ohne Rücksicht auf den resultierenden Rundungsfehler für alle Funktionswerte gleichartig verfahren. Anders dagegen beim Round-Modus: hier wird der Rundungsfehler dadurch minimiert, daß zunächst Ab- und Aufrundwert ermittelt werden und anschließend derjenige der beiden zugeordnet wird, bei dem der Rundungsfehler am geringsten ausfällt.

Zur Untersuchung der drei verschiedenen Modi wird eine Folge ungerundeter Werte $q_i \in I\!R^+$ ($i \in \mathbf{I}_N$) dreimal der Rundung mit jeweils einem Modus unterzogen. Anhand der Folge der gerundeten Werte $u_i \in I\!N$ wird der Rundungsfehler mit Hilfe statistischer Kriterien evaluiert, damit Eigenschaften der Modi erkennbar werden und eine Bewertung möglich ist. Im einzelnen sind folgende Kriterien für die Analyse berücksichtigt worden.

- **Häufigkeit von Vergleichsrelationen:**
  Bei Vergleichen zwischen einem gerundeten u_i und einem ungerundeten Wert q_i sind drei Fallunterscheidungen möglich. Dazu können für alle geordneten Paare $(u_i, q_i) \in I\!N \times I\!R^+$, $i \in \mathbf{I}_N$ drei Relationen wie folgt definiert werden:

$$\begin{aligned} &\text{Relation } \mathsf{Lo}: \quad (u_i, q_i) \in \mathsf{Lo} \iff u_i < q_i \\ &\text{Relation } \mathsf{Eq}: \quad (u_i, q_i) \in \mathsf{Eq} \iff u_i = q_i \\ &\text{Relation } \mathsf{Hi}: \quad (u_i, q_i) \in \mathsf{Hi} \iff u_i > q_i \end{aligned} \tag{4.5}$$

[3] So wird z. B. *logisch* geprüft, ob einem reellen Argumentenwert der Ab- oder Aufrundwert zugeordnet wird.

[4] Im Falle von ausschließlich positiven Werten $q_i \in I\!R^+$, wie es durch die vorhergehenden Stufen der Spezifizierung sichergestellt werden soll, kann auf die Darstellung des Trunc-Modus (Abbrechen) verzichtet werden, da Trunc- und Floor-Modus bei positiven Werten identische Resultate liefern (vgl. Tabelle 3.2, S. 53).

Für jeden Rundungsmodus wird die Häufigkeit der drei Relationen durch die Kardinalität der jeweiligen Relationsmenge bestimmt, d. h. es wird die Anzahl der zutreffenden Vergleichsfälle ermittelt.

$$\begin{aligned}
\text{Häufigkeit von } \mathsf{Lo}: \quad |\mathsf{Lo}| &:= \Big|\{(u_i, q_i) \mid u_i < q_i\}\Big| \\
\text{Häufigkeit von } \mathsf{Eq}: \quad |\mathsf{Eq}| &:= \Big|\{(u_i, q_i) \mid u_i = q_i\}\Big| \\
\text{Häufigkeit von } \mathsf{Hi}: \quad |\mathsf{Hi}| &:= \Big|\{(u_i, q_i) \mid u_i > q_i\}\Big|
\end{aligned} \tag{4.6}$$

- **Maximaler Rundungsfehler:**
  Es handelt sich hierbei um das Maximum über die Differenzen zwischen dem ungerundeten Wert q_i und der gerundeten Ganzzahl u_i. Ermittelt wird für jeden Rundungsmodus das

  - Maximum über die Absolut-Differenzen:

  $$\mathrm{M}_{\mathrm{abs}} := \max\left(\Big\{|q_i - u_i| \;\Big|\; i \in \mathbf{I}_N\Big\}\right) \tag{4.7}$$

  - Maximum über die Relativ-Differenzen in Prozent:

  $$\mathrm{M}_{\mathrm{rel}} := \max\left(\Big\{\frac{|q_i - u_i|}{q_i} \;\Big|\; i \in \mathbf{I}_N\Big\}\right) \cdot 100\% \tag{4.8}$$

- **Durchschnittlicher Rundungsfehler:**
  Es wird hierzu der arithmetischer Mittelwert bzw. Durchschnitt (engl. *Average*) über die Differenzen zwischen dem ungerundeten Wert $q_i \in \mathbb{R}^+$ und dem gerundeten Wert $u_i \in \mathbb{N}$ ermittelt. Gebildet wird für jeden Rundungsmodus der

  - Mittelwert über die summierten Absolut-Differenzen:

  $$\mathrm{A}_{\mathrm{abs}} := \frac{1}{N} \cdot \sum_{i=0}^{N-1} |q_i - u_i| \tag{4.9}$$

  - Mittelwert über die *quadratisch* summierten Differenzen:

  $$\mathrm{A}_{\mathrm{sqr}} := \frac{1}{N} \cdot \sqrt{\sum_{i=0}^{N-1} (q_i - u_i)^2} \tag{4.10}$$

  - Mittelwert über die *relativ* summierten Differenzen in Prozent:

  $$\mathrm{A}_{\mathrm{rel}} := \left(\frac{1}{N} \cdot \sum_{i=0}^{N-1} \frac{|q_i - u_i|}{q_i}\right) \cdot 100\% \tag{4.11}$$

In Tabelle 4.4 (S. 72) ist die Anwendung der drei genannten Modi anhand des *Sinus*-Beispiels mit N:=100 in ihren Auswirkungen aufgeführt. Dazu wird jeweils pro Zeile einer der drei Rundungsmodi Floor, Ceil oder Round aufgeführt. Jedes Glied aus der Folge der Funktionswerte $q_i := f_{\mathrm{ind}}(i) := \sin\left(\frac{\pi}{2} \cdot \frac{i}{N-1}\right)$ wird zunächst zu Vergleichen mit dem jeweiligen Rundungswert $u_i := \mathsf{Con}(q_i) \in \mathbb{N}$ herangezogen; anschließend sind der maximale und der durchschnittliche Rundungsfehler vermerkt.

Eine Interpretation der Analyseergebnisse lassen zunächst die Vergleichs-Häufigkeiten beim Round-Modus zu: Die Werte sind nahezu statistisch gleichverteilt, da Ab- und Aufrunden in

Rundungs-Modus	Relation (u_i, q_i)			Maxima		Mittelwerte		
	|Lo|	|Eq|	|Hi|	M_{abs}	M_{rel}	A_{abs}	A_{sqr}	A_{rel}
Floor	97	3	0	0.994	5.46%	0.520	0.061	0.19%
Ceil	0	3	97	0.999	0.99%	0.450	0.055	0.14%
Round	43	3	54	0.497	0.88%	0.216	0.026	0.08%

Tabelle 4.4: Vergleich der Rundungsmodi anhand des *Sinus*-Beispiels

einem Verhältnis von 43:54 nahezu gleich häufig auftreten. Die trivialen Modi Floor und Ceil weisen beide einen maximalen Absolutfehler von $M_{abs} \approx 1$ aus; sie schneiden damit erwartungsgemäß schlechter ab als Round mit $M_{abs} \approx \frac{1}{2}$. Beim maximalen Relativfehler ergab sich für Floor deshalb ein schlechteres Resultat, da zufällig bei den relativ kleinsten Funktionswerten große Fehler auftraten; dieser Fall hätte stattdessen auch auf Ceil zutreffen können, nicht aber für den daraufhin optimierten Round-Modus.

Während die Maxima ein Kriterium bezüglich der gesamten Folge darstellen, steht bei den Mittelwerten die Beurteilung der Konsequenzen bei einzelnen Gliedern, bezogen auf die gesamte Folge, zur Verfügung. So zeigt sich in der Tabelle bei den Mittelwerten A_{abs}, A_{sqr} und A_{rel}, daß die Werte für Floor und Ceil nahezu identisch sind, während Round hier bei allen drei Mittelwerten in etwa den halben Wert aufweist. Zusammenfassend kann festgestellt werden, daß der Round-Modus in dieser Analyse, aber auch im allgemeinen wegen der Minimierung des Rundungsfehlers deutlich besser abschneidet als die anderen beiden Modi Floor und Ceil.

4.2.4 Untersuchung der Binär-Schicht

4.2.4.1 Exaktheit bei der Kodierung

Seien $f_{\text{int}}(i) \in I\!N$ mit $i \in \mathbf{I}_N$ als Funktionswerte gegeben und $M := \max(\{f_{\text{int}}(i) \mid i \in \mathbf{I}_N\})$ das Maximum aller Funktionswerte; dann gilt für jeden Funktionswert: $f_{\text{int}}(i) \in \{0, \ldots, M\}$. Zur binären Kodierung wird zunächst die erforderliche Anzahl der Bitstellen mit $m := \mathsf{Bits}(M)$ ermittelt, d. h. es gilt: $f_{\text{int}}(i) \in \{0, \ldots, M\} \subseteq \mathbf{I}_{2^m}$. Mit Hilfe der Abbildung $\mathsf{Bin}_m : \mathbf{I}_{2^m} \to I\!B^m$ geschieht nun die Binärkodierung der natürlichen Zahlen u_i, wobei die resultierenden Binärzahlen $\mathsf{Bin}_m(u_i) \in I\!B^m$ lediglich 0en und 1en enthalten, da $I\!B := \{\mathtt{0}, \mathtt{1}\}$ ist.

Beispiel zur Kodierung

Im *Sinus*-Beispiel (vgl. Abschnitt 3.3.5.3, S. 54) wird die f-Variante $f_{\text{int}}(i) := \mathsf{Round}(1\text{E}3 \cdot \sin(\frac{\pi}{2} \cdot \frac{i}{N}))$ binär kodiert. Die Anzahl der Argumente wird dort mit $N := 100$ festgelegt, so daß sich als Indexmenge $i \in \mathbf{I}_{100}$, bzw. $i \in \{0, \ldots, 99\}$ ergibt. Wegen $\sin(x) \in [0,1]$, $0 \leq x \leq \frac{\pi}{2}$ ergibt sich das Maximum der Funktionswerte mit $M := 1\text{E}3$, d. h. $f_{\text{int}}(i) \in \{0, \ldots, 1\text{E}3\}$. Die Anzahl m der Bitstellen für die Funktionswerte ist durch $m := \mathsf{Bits}(M)$ zu bestimmen und ergibt hier $m := 10$. Durch Bin wird nun also ganz $\mathbf{I}_{2^{10}} = \mathbf{I}_{1024}$ auf $I\!B^{10}$ abgebildet, was den Funktionswertebereich von f_{int} vollständig umfaßt, denn $\{0, \ldots, 1000\} \subseteq \mathbf{I}_{1024} := \{0, \ldots, 1024 - 1\}$. $I\!B^{10}$ umfaßt damit alle 10-Bit binär kodierbaren Zahlen.

In der beschriebenen Art ist die von Bin vollzogene Abbildung *eineindeutig*, d. h. Bin besitzt die Eigenschaft einer *Injektivität.* Ferner hat Bin auch die Eigenschaft *Surjektivität*, d. h. der tatsächlich von Bin ausgefüllte Bildbereich ist ganz $I\!B^m$. Beide Eigenschaften zusammen bedeuten *Bijektivität.* Die Zuordnung von Bin bedeutet lediglich die Umwandlung der Zahlendarstellung, so daß dabei kein Verlust an Genauigkeit auftritt und insofern *keine* Unschärfe zu verzeichnen ist.

4.2.4.2 Erweiterung der Kodierung

Bezogen auf die Diskussionsergebnisse der vorher untersuchten Numerik- bzw. Konversions-Schicht in den Abschnitten 4.2.2 bzw. 4.2.3, in denen an mehreren Stellen Unschärfe und Optionen auftraten, scheint bei allen Funktionswerten eine *individuelle Toleranz* vorhanden zu sein. In ihren maximalen Ausmaßen kann sie sogar bei der Konvertierung durch Rundung zu verschiedenen Ganzzahlen führen. Trotzdem muß der Kodierung schließlich nur eine einzige Ganzzahl zugeführt werden. Dieses bedeutet eine Einschränkung, die im allgemeinen akzeptiert wird, hier jedoch hinterfragt werden muß.

Ein erster Ansatz zur Abhilfe ergibt sich aus dem Umstand, daß anschließend an die Kodierungsstufe die resultierende Wahrheitstabelle einer Logik-Minimierung unterzogen wird. Aus Gründen der Optimierung läßt die Logik-Minimierung im allgemeinen auch den unbestimmten Logikwert * (Don't-Care) in der zugeführten Wahrheitstabelle zu (vgl. Abschnitt 1.2.2.2, S. 6); Einträge mit * eröffnen den Minimierungsalgorithmen weitere Möglichkeiten, um minimalere Ergebnisse mit Hilfe der *Verschmelzungsregel* bzw. *Consensus Methode* [Giv-70, Bra*84] erreichen zu können (vgl. Abschnitt 1.2.4.2, S. 12). Die auf $I\!D := I\!B \cup \{*\}$ (vgl. Abschnitt 1.2.2.2, S. 6) bezogene Kodierung mit dem unbestimmten Logikwert ergibt eine neue Möglichkeit, die durch die folgende Option O-6 (S. 74) dokumentiert wird.

Option O-6: Kodierung mit Don't-Cares
Bedingt durch Unschärfe kommt es bei den zur Rundung anstehenden Werten zu Toleranzen, in denen mindestens zwei verschiedene Ganzzahlen liegen. Sofern sich diese an nur einer Bitstelle in ihrer Binärkodierung unterscheiden, können beide Zahlen gemeinsam kodiert werden, wobei das ungleiche Bit durch ein ∗ (Don't-Care) kodiert wird.

4.2.5 Überblick und Interpretation der Untersuchung

Bei den vorhergehenden Untersuchungen konnten einige Aspekte zur Unschärfe und Optionen diskutiert werden, die bei den Transformationen in den drei Schichten auftreten können. In Tabelle 4.5 (S. 74) ist eine Zusammenfassung der diskutierten Aspekte zur Unschärfe und den erkannten Optionen wiedergegeben.

Nr.	Gegenstand bzw. Sachverhalt	Abschnitt
U-1	Quantisierungsfehler an jeder Stützstelle	4.2.2.1 (S. 64)
U-2	Stützstellen werden zufällig festgelegt	4.2.2.5 (S. 67)
U-3	Systematische Arithmetikfehler	4.2.2.6 (S. 68)
U-4	Rundungsfehler je nach Rundungsmodus	4.2.3.1 (S. 70)
O-1	Anzahl der Stützstellen als 2er-Potenz	4.2.2.2 (S. 65)
O-2	Redundanz aus der Stützstellenfolge entfernen	4.2.2.3 (S. 65)
O-3	Veränderlicher Abstand der Stützstellen	4.2.2.4 (S. 66)
O-4	Bestimmung der Skalierungsabbildung	4.2.2.7 (S. 69)
O-5	Nicht-lineare Skalierung	4.2.2.7 (S. 69)
O-6	Kodierung mit Don't-Cares	4.2.4.2 (S. 74)

Tabelle 4.5: Übersicht der Aspekte zur Unschärfe und der Optionen

Im folgenden Kapitel 5 werden mit Hilfe der hier aufgeführten Aspekte zur Unschärfe und den Optionen geeignete Ansatzpunkte für Modifikationen aufgezeigt. Es lassen sich schon jetzt die bereits vorliegenden Erkenntnisse in Gestalt der drei folgenden Kernaussagen zusammenfassen:

1. Kernaussage: *Unschärfe ist inhärenter Bestandteil jeder Spezifikation.*

2. Kernaussage: *Jeder Funktionswert ist mit Unschärfe behaftet.*

3. Kernaussage: *Optionen sind verdeckt vorhandene Möglichkeiten für Modifikationen.*

In diesem Sinne weisen die einzelnen Aspekte zur Unschärfe *direkt* auf Ansatzpunkte für Modifikationen hin, während im Gegensatz dazu die Optionen *indirekte* Möglichkeiten andeuten, die in ihrer Gesamtheit als Empfehlungen dienen können. Generell ist bei allen Bemühungen um geschickte Modifikationen in der Regel nur ein Kompromiß zwischen hoher Genauigkeit einerseits und geringen Realisierungskosten andererseits möglich.

Kapitel 5

Heuristische Modifikationen

Auf dem Hintergrund der im vorhergehenden Kapitel 4 dargelegten Problematik bezüglich Unschärfe und Optionen innerhalb von Spezifizierungen besteht nun ein Bedarf nach optimierenden Ansätzen, die grundsätzlich die *Reduzierung der PLA-Kosten* zum Ziel haben. In diesem Kapitel werden daher Modifikationsmethoden erläutert, die in einzelne Stufen der Spezifizierung eingreifen und in der Regel zu abweichenden Spezifikationen führen. Untersuchungen in [Kra-94, Rau-95] zeigen, daß derartig modifizierte Spezifikationen eine teilweise beachtliche Kostenreduktion erzielen können. Diese positive Konsequenz kann im allgemeinen allerdings nicht garantiert werden, da die verwendeten Methoden, wie oft beim VLSI-Entwurf, *heuristisch*[1] arbeiten. Im folgenden geschieht die Diskussion anhand des Schichtenmodells aus Abschnitt 4.1.4 (S. 60f), d. h. grundsätzlich beginnt sie bei der Numerik-Schicht und verläuft dann über die Transformationen innerhalb der Konversions-Schicht zu denen der Binär-Schicht.

Bezüglich der Numerik-Schicht besteht allerdings eine Besonderheit: Im Sinne von Kernaussage 3 in Abschnitt 4.2.5 (S. 74) stehen eine Reihe von indirekt eingreifenden Optionen zur Verfügung. Diese sollen als Satz an Empfehlungen für die Spezifizierung aufgefaßt werden, wodurch grundlegende Modifikationen in den ersten drei Stufen vorgenommen werden können: Beispielsweise werden so die Formatparameter n und m durch Funktionsdefinition, Diskretisierung und Skalierung bestimmt. In der Gesamtheit aller Möglichkeiten, die sich mit Hilfe der gesammelten Optionen entwickeln lassen, wäre der Rahmen dieser Arbeit überschritten. Aus diesem Grunde sollen nun zunächst nur Methoden für Modifikationen innerhalb der Konversions-Schicht und anschließend im Bereich der Binär-Schicht diskutiert werden.

5.1 Methoden für die Konversions-Schicht

Basierend auf dem in Abschnitt 4.1 (S. 57f) diskutierten Schichtenmodell, stehen nun Methoden zur Modifikation an Spezifizierungen im Mittelpunkt dieses Abschnittes. Dabei handelt es sich um heuristische (vgl. Fußnote 1, S. 75) Methoden, d. h. ihnen liegt jeweils eine sogenannte *Arbeitshypothese* zugrunde: Sie dient jeweils zur Konzentration auf ein Merkmal der Spezifizierung bzw. auf einen einzelnen Aspekt der allgemeinen Zielsetzung im Gesamtkontext, nämlich die Reduzierung der PLA-Kosten (vgl. Abschnitt 2.2.5.6, S. 32). In [Rau-92, S. 16] prägt *Rauscher* in diesem Zusammenhang den Begriff *PLA-gerecht* und stellt, in Bezug auf Unschärfe und Optionen, erste Überlegungen zur Modifikation der Rundung an.

[1] Gemäß [Dud*-90, S. 309] bedeutet das sogenannte *heuristische Prinzip*, daß eine *Arbeitshypothese* der Forschung als Hilfsmittel dient, da eine vorläufige Annahme zu einem besseren Verständnis des Sachverhalts verhilft.

5.1.1 Konvertierung durch Rundung

Bei der Rundung entsteht mit dem Rundungsfehler ein Genauigkeitsverlust (vgl. Unschärfe U-4: *Rundungsfehler je nach Rundungsmodus*, S. 70), der von dem angewandten Modus abhängt. Da der Vergleich der Rundungsmodi (vgl. Abschnitt 4.2.3.2, S. 70f) für den Round-Modus den geringsten Fehler ergibt, wird dieser deshalb häufig zur Konvertierung verwendet. Wie bereits diskutiert wurde, weist aber auch dieser Modus inhärente Unschärfe auf. Eine Festlegung für die erforderliche Genauigkeit wird in Problemlösungen jedoch meistens nicht spezifiziert [Rau-92, S. 16].

5.1.1.1 Arbeitshypothese: Alternative Rundungsmodi

Aufgrund der offenbar auch bei der Rundung vorhandenen Unschärfe verliert der Genauigkeitsaspekt beim Round-Modus seine intuitiv zugestandene Relevanz gegenüber den beiden anderen Modi Floor und Ceil. Da in Bezug auf eine PLA-Realisierung ohnehin die Kosten des resultierenden PLAs in den Vordergrund treten, greift hier die generelle Annahme, daß das Zulassen alternativer Konvertierungen zufälligerweise zu einer Reduzierung der PLA-Kosten führen kann. In diesem Sinne fördert die Hinzunahme der beiden Rundungsmodi Floor und Ceil die Ausawahl bei den alternativen Methoden. Die *Arbeitshypothese* lautet dementsprechend allgemein auf alle folgenden Methoden beziehbar:

> Verschiedene Modi einer Methode — hier die Konvertierung durch Rundung — bewirken unterschiedliche PLA-Kosten.

5.1.1.2 Methode: Drei Modi zur Rundung

In Tabelle 3.2 (S. 53f) sind die drei Rundungsmodi Round (vgl. Gleichung 3.27, S. 52), Floor (vgl. Gleichung 3.23, S. 52) und Ceil (vgl. Gleichung 3.25, S. 52) bereits definiert worden. Die beiden zuletzt genannten weisen generell einen größeren Rundungsfehler (vgl. Gleichung 3.5, S. 39) als der Round-Modus auf. Bezüglich der PLA-Kosten bleibt der Prüfung im Einzelfall vorbehalten, ob durch den Einsatz von Floor- und Ceil-Modus zufällig eine Kostenreduzierung resultiert. Aus Gründen der Einheitlichkeit mit den noch folgenden Methoden sei die Methode der Rundung mit dem Kürzel Rnd bezeichnet; sie verfügt über die drei Modi Rnd_Round, Rnd_Floor und Rnd_Ceil.

5.1.1.3 Demonstration der Rundungsmethode

Auf eine Demonstration der Rnd-Methode mit ihren drei Modi kann hier verzichtet werden, da bereits in Tabelle 3.2 (S. 53f) die Wirkungsweise der drei Modi anhand einiger Beispielzahlen demonstriert worden ist. Ferner ist ein Vergleich in Abschnitt 4.2.3.2 (S. 70f) diskutiert worden; in Tabelle 4.4 (S. 72) ist eine Übersicht über die Rundungsfehler aufgeführt.

5.1.2 Konvertierung mit 2er-Potenzwerten

5.1.2.1 Arbeitshypothese: 2er-Potenzwerte

Bezüglich der Logik-Minimierung ist die Anzahl der `1`en ein erster Hinweis auf die Termanzahl (vgl. Gleichung 2.12, S. 28). Eine effektive Minimierung der Termanzahl kann besonders mit den Werten gelingen, die in binärer Darstellung lediglich nur *eine* `1` aufweisen, d. h. es handelt sich dabei um „glatte" 2er-Potenzwerte. Die *Arbeitshypothese* lautet hier somit:

> Einträge in der PLA-Tabelle, die nur eine `1` in ihrer Binärdarstellung aufweisen, führen insgesamt zu geringen PLA-Kosten.

Der Ansatz besteht also darin, nur 2er-Potenzwerte bei der Konvertierung[2] zuzuordnen.

5.1.2.2 Methode: Drei Modi für 2er-Potenzwerte

Die Zuordnung von 2er-Potenzwerten sei als Ansatz durch P2 (engl. *Power-of-2*) identifiziert und vollzieht folgende Abbildung:

$$\mathsf{P2}:\ \mathbb{R}^+ \to \{2^k \mid k \in \mathbb{N}\} \subset \mathbb{N} \tag{5.1}$$

Bei der Konvertierung eines reellen Arguments stehen grundsätzlich zwei 2er-Potenzwerte zur Auswahl: Analog zur Rundung kann der nächstniedrigere, der nächstgrößere sowie von den beiden genannten der nächstliegende 2er-Potenzwertes gewählt werden. Bei einer Zuordnung des höheren 2er-Potenzwert ist zu beachten, daß das Maximum $M := 2^m$, was bei der Skalierung in der Numerik-Schicht festgelegt wird, eine obere Schranke[3] bildet; falls für ein Argument kein höherer 2er-Potenzwert unterhalb der Schranke existiert, muß stattdessen der niedrigere zugeordnet werden. Es ergeben sich somit für diese Methode drei Modi, die wie folgt definiert werden können: Jedem Argument $q \in \mathbb{R}^+$ wird bei

- P2_Low der *nächstniedrigere* 2er-Potenzwert zugeordnet:

$$\mathsf{P2\_Low}(q) := \begin{cases} 0 & : \quad q = 0 \\ 2^{\lfloor \log_2(q) \rfloor} & : \quad \text{sonst.} \end{cases} \tag{5.2}$$

- P2_High der *nächstgrößere* 2er-Potenzwert zugeordnet, falls dabei das Maximum $M := 2^m$ nicht überschritten wird; ansonsten wird der nächstniedrigere 2er-Potenzwert zugeordnet:

$$\mathsf{P2\_High}(q) := \begin{cases} 0 & : \quad q = 0 \\ 2^{\lceil \log_2(q) \rceil} & : \quad q > 0 \quad \text{und} \quad 2^{\lceil \log_2(q) \rceil} < M \\ \mathsf{P2\_Low}(q) & : \quad \text{sonst.} \end{cases} \tag{5.3}$$

[2] Im Gegensatz zur Rundung ordnen Ansätze dieser Art in der Regel keine benachbarte Ganzzahlen zu. Sie können damit auch nicht unter dem Begriff *Rundung* zusammengefaßt werden. Daraus leitet sich die Motivation für den hier entsprechend weiter gefaßten Begriff *Konversion* ab, wodurch über die Art der Zuordnung keine Festlegung gemacht wird.

[3] Diese Überlegung trifft allgemein auf alle Ansätze innerhalb der Konversions-Schicht zu, da größere Ganzzahlen, die das Maximum $M := 2^m$ überschreiten, in der Binär-Schicht nicht zu kodieren sind. Beide Formatparameter n und m (vgl. Abschnitt 1.2.2.3, S. 7) müssen sowohl in der Konversions- als auch in der Binär-Schicht als *konstant* angesehen werden, da auf beide lediglich innerhalb der Numerik-Schicht durch Diskretisierung bzw. Skalierung Einfluß genommen werden kann.

- P2_Near der *nächstliegende* 2er-Potenzwert zugeordnet, falls dabei das Maximum $M := 2^m$ nicht überschritten wird; ansonsten wird der nächstniedrigere 2er-Potenzwert zugeordnet:

$$\begin{aligned} \mathsf{P2\_Near}(q) \quad &:= \quad \begin{cases} 0 & : \quad q = 0 \\ \mathsf{P2\_Low}(q) & : \quad 0 < q < \frac{1}{2} \cdot \Big(\mathsf{P2\_Low}(q) + \mathsf{P2\_High}(q)\Big) \\ \mathsf{P2\_High}(q) & : \quad \text{sonst.} \end{cases} \\ &\neq \quad 2^{[\log_2(q)]} \qquad (!!) \end{aligned} \tag{5.4}$$

5.1.2.3 Demonstration der 2er-Potenz-Methode

Zur Demonstration der P2-Methode sei beispielhaft eine Folge von acht reellen Werten $q_i \in I\!R^+$ mit $i \in \mathbf{I}_8$ gegeben, die mit Hilfe der drei Modi P2_Low, P2_High und Ps_Near nach $I\!N$ konvertiert werden sollen. Die Resultate $v_i, w_i \in \{2^k \mid k \in I\!N\}$ sollen später in eine 7-Bit Binärdarstellung kodiert werden, so daß eine obere Schranke mit dem Maximum $M := 128 = 2^7$ besteht. Die Situation ist in Tabelle 5.1 (S. 78) wiedergegeben.

reelle Folge		P2-Methode				
		$v_i :=$ P2_Low(q_i)		$w_i :=$ P2_High(q_i)		P2_Near$(q_i) :=$
i	q_i	v_i	$\lvert q_i - v_i\rvert$	w_i	$\lvert w_i - q_i\rvert$	v_i oder w_i ?
0	0.0	0	0.0	0	0.0	0
1	2.5	2	0.5	4	1.5	v_i
2	5.7	4	1.7	8	2.3	v_i
3	13.1	8	5.1	16	2.9	w_i
4	22.4	16	6.4	32	9.6	v_i
5	30.9	16	14.9	32	1.1	w_i
6	47.8	32	15.8	64	16.2	v_i
7	69.2	64	5.2	v_i	5.2	v_i

Tabelle 5.1: Demonstration der P2-Methode

Für P2_Low bzw. P2_High sind jeweils der von ihnen zugeordnete Wert u_{L} bzw. u_{H} zusammen mit den hervorgerufenen Differenzen zum reellen Wert $q \in I\!R^+$ aufgeführt. Bei P2_Near wird das Resultat der Auswahl zwischen u_{L} und u_{H} vermerkt; die Differenz ist entsprechend der getroffenen Auswahl. Anzumerken ist der letzte Fall, bei dem sowohl P2_High als auch P2_Near den niedrigeren 2er-Potenzwert zuordnen, da die Beschränkung durch das Maximum $M := 128 = 2^7$ dieses erfordert.

5.1.3 Intervalle in der Konversions-Schicht

Nach der 2. Kernaussage aus Abschnitt 4.2.5 (S. 74) haftet jedem Funktionswert Unschärfe an. Wie bereits in Abschnitt 4.2.4.2 (S. 73) angedeutet wurde, existiert für jeden Funktionswert $f_{\text{ind}}(i) \in I\!R$ offenbar eine *individuelle Toleranz*. Sofern sie nur geringe Ausmaße annimmt, geht dieses innerhalb der Konversions-Schicht unter, da durch die stattfindende *Begrenzung der Auflösung* alle Funktionswerte im Bereich der Toleranz auf nur eine Ganzzahl abgebildet werden. Die Toleranz kann aber auch von größerem Ausmaß sein, so daß sich die Unterschiede zwischen den extremen Funktionswerten innerhalb der Konversions-Schicht derart bemerkbar machen, daß auf *mehrere* Ganzzahlen abgebildet werden kann: aufgrund der Unschärfe kann jeder von ihnen mit gleicher Berechtigung zugeordnet werden.

5.1.3.1 Toleranz bewirkt Mehrdeutigkeit

Auf die f-Variante f_{fix} bezogen, ergibt sich im allgemeinen durch die anzunehmende Toleranz eine *mehrdeutige* Zuordnung: Jedem der $N \in I\!N$ Indizes $i \in \mathbf{I}_N$ kann ein Funktionswert $q_i := f_{\text{fix}}(i) \in I\!R^+$ aus dem Toleranzbereich zugeordnet werden. In der Regel wird es sich um einen kontinuierlichen Toleranzbereich handeln, d. h. es sind unendlich viele Funktionswerte für die Zuordnung zulässig. Mit $s_i, t_i \in I\!R^+$ als Grenzen des Toleranzbereiches ergibt sich für einen zugeordneten Funktionswert $s_i \leq f_{\text{fix}}(i) \leq t_i$. Solange $u_i := \mathsf{Con}(s_i) = \mathsf{Con}(t_i)$ mit beispielsweise $\mathsf{Con} \equiv \mathsf{Round}$ gilt, ist die Toleranz der $f_{\text{fix}}(i)$ in der Konversions-Schicht schon nicht mehr signifikant. Trifft allerdings $\mathsf{Con}(s_i) \neq \mathsf{Con}(t_i)$ zu, dann existieren zwei Ganzzahlen $v_i, w_i \in I\!N$ mit $v_i < w_i$, so daß folgendes gilt: $v_i \leq \mathsf{Con}(s_i) < \mathsf{Con}(t_i) \leq w_i$. Mit v_i und w_i wird ein Intervall beschrieben, aus dem nun jede Ganzzahlen dem Index zugeordnet werden können. Dieses entspricht nicht mehr der *eineindeutigen* Abbildung der f-Variante $f_{\text{int}} : I\!N \to I\!N$.

5.1.3.2 Relationale Zuordnungen bzw. Boolean Relations

In dieser Situation ergibt sich eine *mehrdeutige* Zuordnung, die einem Index $i \in \mathbf{I}_N$ ein Intervall $[v_i, w_i]$ mit $v_i, w_i \in I\!N,\ v_i < w_i$ zuordnet. Damit stehen dem Index i mehrere Funktionswerte aus $[v_i, w_i]$ gegenüber, d. h. der Index i und die Funktionswerte aus $[v_i, w_i]$ *stehen in Relation zueinander*. Die *Relation* r_{int} wird daher wie folgt definiert:

$$r_{\text{int}} \subseteq I\!N \times I\!N \tag{5.5}$$

$$\text{wobei:} \quad (i, u_i) \in r_{\text{int}} \iff u_i \in [v_i, w_i] \tag{5.6}$$

Werden solche Relationen mit in die Binär-Schicht übernommen, so liegen statt Logikfunktionen nunmehr Logikrelationen (engl. *Boolean Relations*) vor, d. h. für Eingangsbelegungen existieren multiple Ausgangsbelegungen. Durch jüngste Forschungsergebnisse wurde dieses Gebiet theoretisch erschlossen [Bra*89a]. In verschiedenen Arbeiten [Bra*89c, Bra*89b, Lin*90a] sind Algorithmen zur Minimierung von derartigen Relationen vorgestellt worden. Da die Ansätze sehr komplex sind und in ihrer Anwendung viel Laufzeit und Speicherplatz benötigt, sind mittlerweile heuristische Verfahren als Alternativen verfügbar [Gho*90, Bra*93].

Ein wesentlicher Grund für die hohe Komplexität solcher Ansätze liegt darin, daß die Herkunft der Daten völlig offen bleibt und die Ansätze daher sehr allgemein gehalten sind, da beliebige

Mehrfachzuordnungen vorkommen können. Für den speziellen Kontext dieser Arbeit, bei dem es sich um eine numerische Herkunft der Daten handelt, treffen auf die Funktion f typische Funktionseigenschaften, wie z. B. Stetigkeit zu, so daß die Mehrfachzuordnungen nicht in einem völlig beliebigen Rahmen stattfinden. Aufgrund der durch Unschärfe bedingten Toleranzen kommt es daher zu wesentlichen Einschränkungen: Für jeden Index ist das zugeordnete ganzzahlige Intervall bezüglich $\mathbb{Z}$ eine lückenlose Teilmenge, d. h. die darin enthaltenen Werte unterliegen damit einer signifikanten Beschränkung und sind somit nicht beliebig zusammengestellt. Ferner lassen die Ansätze die Art der Logik-Realisierung offen, während es in diesem Kontext lediglich um 2-stufige Logik in Gestalt von PLAs geht. Aus diesen Gründen erscheinen derartige Ansätze hier nicht relevant.

5.1.3.3 Arbeitshypothese: Intervalle

Eine systematische Repräsentation von Toleranzen, die innerhalb der Konversions-Schicht signifikant sind, kann durch das Zuordnen von Intervallen zu den Indizes erfolgen. Im allgemeinen sind die Ausmaße der Toleranz für jeden Index individuell, obwohl durch systematisch bedingte Fehler auch einheitlich für alle Indizes Unschärfe auftreten kann (vgl. Unschärfe U-3: *Systematische Arithmetikfehler*, S. 68). Für eine Methode zur Generierung von Intervallen lautet die *Arbeitshypothese* dementsprechend:

> Intervalle können die durch Unschärfe bedingte Toleranz bei den Einträgen repräsentieren.

Als Vereinfachung sollten die Intervalle für alle Einträge generisch erzeugt werden, wie dies später noch erläutert wird. Zur Vervollständigung der Konvertierung wird noch eine weitere Methode benötigt, die in Abschnitt 5.1.4 (S. 83f) erläutert wird.

5.1.3.4 Methode: Generierung von Intervallen

In Kombination mit der im folgenden Abschnitt 5.1.4 (S. 83f) erläuterten Methode zur *Selektion von Werten* besteht mit der Methode zur *Generierung von Intervallen* eine weitere Möglichkeit für die Konvertierung von $\mathbb{R}^+$ nach $\mathbb{N}$. Beide Methoden seien wie folgt umschrieben:

- **Int**: *Generierung von Intervallen* für die reell-positiven Funktionswerte, die der Konversions-Schicht zugeführt werden. Dazu können verschiedene Generierungsmodi entwickelt werden.
- **Sel**: *Selektion von Werten* anhand der durch Int generierten Intervalle; dabei soll die Selektion möglichst *PLA-gerecht* erfolgen. Dafür können ebenfalls unterschiedliche Modi bereitgestellt werden.

Bevor die einzelnen Generierungsmodi erörtert werden, sei die Abbildung der Methode Int zur Generierung von Intervallen wie folgt definiert:

$$\mathsf{Int} \quad : \quad \mathbb{R}^+ \to \mathbb{N} \times \mathbb{N} \tag{5.7}$$

$$\text{wobei } \mathsf{Int}(q) \quad := \quad (v, w) \quad \text{mit} \quad v, w \in \mathbb{N} \quad \text{und} \quad v < w \tag{5.8}$$

Dabei wird das geordnete Paar (v, w) als Unter- bzw. Obergrenze eines geschlossenen Intervalles aufgefaßt, so daß sich durch (v, w) die Menge $\{u \in I\!N \mid v \leq u \leq w\}$ ergibt. Das Resultat der Abbildung Int sei künftig durch die übliche Intervallschreibweise $[v, w]$ bezeichnet.

5.1.3.5 Modus: Intervalle durch Ab- und Aufrundwert

Ein minimales Intervall kann bereits aus dem Ab- und Aufrundwert eines gegebenen Funktionswertes generiert werden. Im Sinne des Begriffes *Rundung* liegt hiermit ein weiterer Rundungsmodus vor, da bei der nachfolgenden Selektion einer Zahl entweder nur der Ab- oder Aufrundwert zugeordnet werden kann. Im Gegensatz zu den bisherigen Rundungsmodi Floor, Ceil und Round kann allerdings durch die Selektion eine *PLA-gerechte* Auswahl erfolgen. Dieser Int-Modus sei mit $\mathsf{Int\_UpDn}$ bezeichnet:

$$\mathsf{Int\_UpDn}(q) := \Big[\lfloor q \rfloor \; , \; \lceil q \rceil\Big] \;=\; \Big\{\lfloor q \rfloor \; , \; \lceil q \rceil\Big\} \tag{5.9}$$

Da für $q \neq 0$ bei diesen Intervallen die Differenz zwischen den Intervallgrenzen immer 1 ist, kann der in Option O-6, : *Kodierung mit Don't-Cares* (S. 74), angedeutete Fall eintreten, daß die Binärdarstellungen der Grenzen sich ggf. nur an einer Bitstelle, dem LSB, unterscheiden. Durch eine Kodierung mit $*$ (Don't-Care) ist dann die Konvertierung vom reellen q zur m-Bit binärkodierten Ganzzahl $\mathsf{Bin}_m(u),\ u \in [v, w]$ bereits vollzogen.

5.1.3.6 Modi: Intervalle durch Toleranzparameter

Im Gegensatz zum $\mathsf{Int\_UpDn}$-Modus können Intervalle auch weiter aufgespannt werden, da die den reellen Werten anhaftende Unschärfe durchaus in höherer Größenordnung auftreten kann, wie beispielsweise der in Abschnitt 4.2.2.1 (S. 64) beschriebene Toleranzbereich von $\pm 5\%$ andeutet. Außerdem ist nicht selten eine Toleranzangabe aus der Problemstellung heraus begründbar [Rau-92, S. 16]. Je mehr ganzzahlige Werte die Intervalle umfassen, desto mehr Möglichkeiten bestehen, eine *PLA-gerechte* Auswahl zu treffen.

Ein Toleranzbereich $[v, w]$ mit $v \leq q \leq w$ muß im allgemeinen nicht symmetrisch um $q \in I\!R^+$ angeordnet sein, d. h. q muß sich nicht unbedingt in der Mitte des Intervalls $[v, w]$ befinden. Zur Festlegung eines Intervalles ist es daher sinnvoll sowohl die Distanz zur unteren $(q - v)$ als auch zur oberen $(w - q)$ Intervallgrenze getrennt anzugeben. Zu beachten ist bei der Generierung der Intervallgrenzen $v, w \in I\!N$, daß zunächst reelle Zwischenwerte entstehen, die noch von $I\!R^+$ nach $I\!N$ konvertiert werden müssen; dies geschieht am sinnvollsten durch entsprechende Rundung mit Floor oder Ceil, jeweils in Richtung der Ausdehnung des Intervalls, d. h. für die untere Grenze wird Floor und für die obere Grenze Ceil eingesetzt. Ferner muß für beide Intervallgrenzen $v, w \in I\!N$ die jeweilige Beschränkung durch den Zielbereich $\mathbf{I}_M := \{0, \ldots, M-1\}$ bei der späteren Kodierung in die m-Bit Binärdarstellung beachtet werden (vgl. Fußnote 3, S. 77), d. h. alle generierten Intervalle müssen innerhalb $[0, M-1]$ liegen.

Die Angabe der jeweiligen Distanzen kann sowohl *absolut* als auch *relativ* bezogen auf q erfolgen, d. h. für die Methode Int können zwei weitere Modi definiert werden, die mit $\mathsf{Int\_Abs}$ bzw. $\mathsf{Int\_Rel}$ bezeichnet seien: Jedem reell-positiven Wert $q \in I\!R^+$ wird bei

- $\mathsf{Int\_Abs}$ ein Intervall zugeordnet, das um je einen *absoluten* Betrag $a \in I\!N$ nach unten bzw. $b \in I\!N$ nach oben um q herum aufgespannt wird:

$$\mathsf{Int\_Abs}\Big|_a^b(q) \;:=\; [v, w] \tag{5.10}$$

$$\text{mit} \quad v \;:=\; \begin{cases} \lfloor q \rfloor - a : & \lfloor q \rfloor \geq a \\ 0 \quad : & \text{sonst.} \end{cases} \tag{5.11}$$

$$\text{und} \quad w \;:=\; \begin{cases} \lceil q \rceil + b : & \lceil q \rceil < M - b \\ M - 1 \; : & \text{sonst.} \end{cases} \tag{5.12}$$

- Int_Rel ein Intervall zugeordnet, das um je einen *relativen* Betrag in Prozent $a \in I\!N$ nach unten bzw. $b \in I\!N$ nach oben um q herum aufgespannt wird:

$$\mathsf{Int\_Rel}\Big|_a^b(q) \;:=\; [v, w] \quad \text{wobei} \quad a, b \leq 100 \text{ in } \% \tag{5.13}$$

$$\text{mit} \quad v \;:=\; \left\lfloor q \cdot (1 - \tfrac{a}{100\%}) \right\rfloor \geq 0 \tag{5.14}$$

$$\text{und} \quad w \;:=\; \begin{cases} \left\lceil q \cdot (1 + \tfrac{b}{100\%}) \right\rceil : & \lceil q \rceil < M - \left\lceil q \cdot (1 + \tfrac{b}{100\%}) \right\rceil \\ M - 1 \quad : & \text{sonst.} \end{cases} \tag{5.15}$$

5.1.3.7 Demonstration der Intervall-Methode

Zur Demonstration der Int-Methode sei eine Folge aus acht reellen Werten $q_i \in I\!R^+$ mit $i \in \mathbf{I}_8$ aus Abschnitt 5.1.2.3 (S. 78) gegeben, die nun mit Hilfe der drei Modi Int_UpDn, Int_Abs und Int_Rel in ganzzahlige Intervalle konvertiert werden sollen. Während für Int_Abs die absoluten Distanzen mit $a := 3$ und $b := 2$ gegeben seien, wird für Int_Rel einheitlich für beide Distanzen $a := b := 5\%$ gewählt. Aus den resultierenden Intervallen kann später mit der Methode Sel eine Selektion von Werten erfolgen, welche anschließend nur noch in eine 7-Bit Binärdarstellung kodiert werden. Daher existiert eine obere Schranke mit dem Maximum $M := 128 = 2^7$. Die Situation nach der Generierung der Intervalle ist in Tabelle 5.2 (S. 82) wiedergegeben.

Folge		Int-Methode		
i	q_i	Int_UpDn(q_i)	Int_Abs$\Big|_3^2(q_i)$	Int_Rel$\Big|_{5\%}^{5\%}(q_i)$
0	0.0	0	[0,2]	0
1	2.5	[2,3]	[0,5]	[2,3]
2	5.7	[5,6]	[2,8]	[5,6]
3	13.1	[13,14]	[10,16]	[12,14]
4	22.4	[22,23]	[19,25]	[21,24]
5	30.9	[30,31]	[27,33]	[29,33]
6	47.8	[47,48]	[44,50]	[45,51]
7	69.2	[69,70]	[66,72]	[65,73]

Tabelle 5.2: Demonstration der Int-Methode

Während bei Int_UpDn und Int_Abs die Intervalle $[v_i, w_i]$, bis auf wenige Ausnahmen, eine konstante Weite $w_i - v_i$ aufweisen, verändert sich diese bei Int_Rel proportional zu q_i. Bemerkenswert ist, daß sich bei $q_0 := 0.0$ lediglich beim Int_Abs-Modus ein Intervall ergibt; bei Int_UpDn und Int_Rel ist das Ergebnis 0. Während es in der Definition von Int_Rel durch die Multiplikation von q_i bewirkt wird, ist dies bei Int_UpDn durch Definition von Floor und Ceil bzw. $\lfloor\ \rfloor$ und $\lceil\ \rceil$ bedingt. Intervalle mit einer Weite $w - v = 1$ können bei der Binärkodierung mit Hilfe von *

(Don't-Care) dargestellt werden; dies tritt insbesondere bei Int_UpDn häufig auf, hier bei den Intervallen [2,3], [22,23] und [30,31].

5.1.4 Konvertierung durch Selektion

5.1.4.1 Arbeitshypothese: Selektion

Wie bereits im vorhergehenden Abschnitt 5.1.3 (S. 79) angedeutet wurde, ist diese Methode zur *Selektion von Werten* in Kombination mit der Methode zur *Generierung von Intervallen* anzuwenden. Als *Arbeitshypothese* steht hierbei folgendes im Vordergrund:

> Die *PLA-gerechte* Selektion von Werten anhand von Intervallen reduziert die PLA-Kosten.

5.1.4.2 Methode: Selektion von Werten

Die Selektion eines ganzzahligen Wertes $u \in I\!N$ aus einem Intervall $[v, w]$ mit $v, w \in I\!N$ und $v < w$, geschieht durch die Abbildung der Methode Sel, die wie folgt definiert ist:

$$\begin{aligned} \mathsf{Sel} &: I\!N \times I\!N \to I\!N \\ \text{wobei} \quad u &:= \mathsf{Sel}(v, w) \end{aligned} \tag{5.16}$$

5.1.4.3 Modi: Selektion anhand von Intervallen

Das Selektieren von Werten $u \in I\!N$ geschieht anhand gegebener Intervalle $[v, w]$ mit $v, w \in I\!N$ und soll dabei möglichst *PLA-gerecht* geschehen, d. h. es sollen nach Möglichkeit solche Werte gewählt werden, die die Termanzahl $\ell \in I\!N$ gering halten. Im folgenden seien aus diesem Bestreben heraus fünf unterschiedliche Modi vorgestellt und definiert:

- Sel_Low selektiert die untere Intervallgrenze (engl. *Low Border*); es sei dafür definiert:

$$\mathsf{Sel\_Low}(v, w) := v \tag{5.17}$$

- Sel_High selektiert die obere Intervallgrenze (engl. *High Border*); es sei dafür definiert:

$$\mathsf{Sel\_High}(v, w) := w \tag{5.18}$$

- Sel_Mean selektiert den Wert in der Mitte des Intervalles, d. h. es wird der gerundete Mittelwert (engl. *Mean Value*) aus den beiden Intervallgrenzen $v < w$ zugeordnet. Eine Besonderheit bilden Intervalle mit der Weite $w - v = 1$, bei denen folgende Fallunterscheidung besteht:

(a) $\exists(k \in I\!N) : (v = 2k)$ und $(w = 2k + 1)$,
d. h. in diesem Fall ist v gerade und w ungerade. Wie bei der Intervallgenerierung durch Int_UpDn bereits angedeutet (vgl. Abschnitt 5.1.3.5, S. 81), ergibt sich eine Kodierung des Intervalls mit Hilfe des unbestimmten Logikwertes * (Don't-Care), da sich in der Binärdarstellung die Differenz als LSB, nämlich $1 = 2^0$, auflöst. Für eine abschließende Logik-Minimierung ist dies von Vorteil, so daß Sel_Mean in diesem Falle kein Ergebnis[4] liefern sollte.

(b) $\exists(k \in I\!N, k \neq 0) : (v = 2k - 1)$ und $(w = 2k)$,
d. h. v ist ungerade und w ist gerade. Da die Weite $w - v = 1$ in der Binärdarstellung von v und w sich nicht als 2^0 auflöst, sind mehr als nur ein Bit unterschiedlich, so daß sich keine Kodierung mit * ergibt. Sel_Mean sollte jedoch immer die obere Grenze zum Ergebnis wählen, da $w = 2k$ gerade ist und damit das LSB 0 ist.

Für Sel_Mean ergibt sich nun insgesamt folgende Definition mit der Besonderheit, daß der Modus „untätig" bleibt, wenn der Fall (a) eintritt:

$$\mathsf{Sel\_Mean}(v, w) := \begin{cases} \text{-} & : \quad w - v = 1 \text{ und (a)} \qquad \text{(vgl. Fußnote 4)} \\ \left[\frac{v+w}{2}\right] & : \quad \text{sonst.} \end{cases} \tag{5.19}$$

Die Motivation zum Runden, anstatt beispielsweise abzuschneiden, ist nicht nur intuitiv, sondern auch durch eine weitere Möglichkeit zur Reduzierung der 1en-Anzahl zu begründen: Beim Abschneiden mit Hilfe von $\langle\ \rangle$ würde es bei statistisch gleichwahrscheinlichen Werten für v und w im LSB gleich häufig zu einer 1 kommen wie zu einer 0. Beim Runden mit [ ] überwiegen die auf 0 gesetzten LSBs aufgrund der Besonderheiten in der Definition der Rundung (vgl. Fall (c) in Abschnitt 3.1.4.4, S. 38): Bezüglich der Summe aus $v + w$ gibt es ein $u \in I\!N$, so daß entweder $v + w = 2u$ oder $v + w = 2u + 1$ gilt. Bei der Halbierung ergibt sich dann entweder u oder $u + \frac{1}{2}$. Im ersten Fall ergibt sich durch das Runden wiederum die Zahl u, die ungerade oder gerade sein kann. Der Logikwert des LSB von u ist gleich häufig 0 oder 1. Im zweiten Fall ergibt die Halbierung jedoch einen Wert, dessen Nachkommaanteil genau 0.5 ist. Das Runden führt nun immer zu einer geraden Zahl und damit zu einer 0 im LSB. Da im allgemeinen von einer gleichen Wahrscheinlichkeit der beiden Fälle ausgegangen werden kann, erscheint die Rundung mit [ ] insgesamt für die Reduzierung der 1en-Anzahl günstig.

- Sel_Min1 selektiert eine Ganzzahl innerhalb des Intervalls, deren Binärkodierung eine minimale Anzahl von 1en aufweist. Zu diesem Zweck sind die Intervallgrenzen v und w in ihrer jeweiligen m-Bit Binärdarstellung $v_{m-1} \ldots v_0 := \mathsf{Bin}_m(v)$ und $w_{m-1} \ldots w_0 := \mathsf{Bin}_m(w)$ (vgl. Gleichung 1.10, S. 7) zu betrachten.

  Ausgehend von den MSBs v_{m-1} und w_{m-1} ist diejenige Bit-Position $j < m$ zu finden, an der sich die beiden Darstellungen erstmals voneinander unterscheiden, d. h. es ist $v_j \neq w_j$. Eine derartige Bit-Position j existiert, da $v < w$ gilt. Dabei ist $v_j = 0$ und $w_j = 1$, und für jede höherwertige Bitstelle an den Positionen i mit $m > i > j$ gilt: $v_i = w_i$. Während nun zur Selektion einer derartigen Ganzzahl die höherwertigen und identischen Bitstellen an den Positionen $m - 1$ bis $j + 1$ unverändert übernommen werden, sind die restlichen Bitstellen von Position j bis 0 durch eine 1 mit nachfolgenden 0en zu ersetzen. Zusammenfassend sind damit zur Selektion eines Wertes $u := \mathsf{Sel\_Min1}(v, w)$ folgende Schritte durchzuführen:

[4] In einer algorithmischen Realisierung von Sel_Mean sollte in diesem Fall das Intervall unverändert übernommen werden.

1. Kodierung von v und w in einer m-Bit Binärdarstellung:
   Es ergeben sich $v_{m-1} \ldots v_0 := \mathsf{Bin}_m(v)$ und $w_{m-1} \ldots w_0 := \mathsf{Bin}_m(w)$.
2. Ermitteln der Bit-Position $j \in \{0, \ldots, m-1\}$, für die gilt: $v_j \neq w_j$
3. Übernahme der höherwertigen und identischen Bitstellen:
   Für alle $i \in \{j+1, \ldots, m-1\}$ ist $u_i := v_i = w_i$.
4. Die j-te Bitstelle in u wird auf $\mathsf{1}$ gesetzt, d. h. $u_j := \mathsf{1}$.
5. Alle niederwertigen Bitstellen werden auf $\mathsf{0}$ gesetzt:
   Für alle $i \in \{0, \ldots, j-1\}$ ist $u_i := \mathsf{0}$.

Es ergibt sich nun in der Binärdarstellung von v, w und u insgesamt folgende Situation:

$$\begin{array}{r|ccccccc} v: & v_{m-1} & \ldots & v_{j+1} & \mathsf{0}_j & v_{j-1} & \ldots & v_0 \\ w: & w_{m-1} & \ldots & w_{j+1} & \mathsf{1}_j & w_{j-1} & \ldots & w_0 \\ \hline u: & u_{m-1} & \ldots & u_{j+1} & \mathsf{1}_j & \mathsf{0}_{j-1} & \ldots & \mathsf{0}_0 \end{array} \tag{5.20}$$

Offenbar gilt $v \leq u \leq w$.

In Tabelle 5.3 (S. 85) wird die Methode $\mathsf{Sel\_Min1}$ anhand zweier Beispiele veranschaulicht: Es sei $m := 8$ und ein Intervall mit $v := 30$ bzw. $v := 94$ und $w := 35$ bzw. $w := 99$ gegeben. Die Methode bestimmt in beiden Fällen j zu 5, da gilt: $v_7 = w_7$ und $v_6 = w_6$, hingegen $v_5 = \mathsf{0} \neq \mathsf{1} = w_5$. Die Ersetzung der Positionen j bis 0 durch eine $\mathsf{1} \ldots \mathsf{0}$-Folge führt zu $\mathsf{0010\,0000}$ bzw. $\mathsf{0110\,0000}$, so daß sich damit als Ergebnis $\mathsf{Sel\_Min1}(30, 35) := 32$ bzw. $\mathsf{Sel\_Min1}(94, 99) := 96$ ergibt.

m:=8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
v:=30	0	0	0	1	1	1	1	0
31	0	0	0	1	1	1	1	1
$\leadsto$ 32	0	0	1	0	0	0	0	0
33	0	0	1	0	0	0	0	1
34	0	0	1	0	0	0	1	0
w:=35	0	0	1	0	0	0	1	1

m:=8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
v:=94	0	1	0	1	1	1	1	0
95	0	1	0	1	1	1	1	1
$\leadsto$ 96	0	1	1	0	0	0	0	0
97	0	1	1	0	0	0	0	1
98	0	1	1	0	0	0	1	0
w:=99	0	1	1	0	0	0	1	1

Tabelle 5.3: Beispiel zur Methode $\mathsf{Sel\_Min1}$

- $\mathsf{Sel\_NP2}$ selektiert einen 2er-Potenzwert, d. h. eine Ganzzahl mit nur einer $\mathsf{1}$ in ihrer binären Darstellung. Sofern kein 2er-Potenzwert innerhalb des Intervalls vorkommt, wird der nächstliegende 2er-Potenzwert (engl. *Nearest-Power-of-2*) selektiert, der sich *außerhalb* des Intervalls befindet. Wie bei $\mathsf{Sel\_Min1}$ sind hierzu die Intervallgrenzen v und w in ihrer jeweiligen m-Bit Binärdarstellung $v_{m-1} \ldots v_0 := \mathsf{Bin}_m(v)$ und $w_{m-1} \ldots w_0 := \mathsf{Bin}_m(w)$ (vgl. Gleichung 1.10, S. 7) zu betrachten.

  Ausgehend von den MSBs v_{m-1} und w_{m-1} ist diejenige Bit-Position $j < m$ zu finden, an der sich erstmals in einer der beiden Darstellungen eine $\mathsf{1}$ befindet, d. h. $v_j = \mathsf{1}$ oder $w_j = \mathsf{1}$. Wegen $v < w$, folgt aus

  - $v_j \neq w_j$: Es sind $v_j = \mathsf{0}$ und $w_j = \mathsf{1}$. Das bedeutet, daß sich der 2er-Potenzwert $\mathsf{0}_{m-1} \ldots \mathsf{0}_{j+1} \mathsf{1}_j \mathsf{0}_{j-1} \ldots \mathsf{0}_0$ *innerhalb* des Intervalls befindet. In diesem Fall wird genauso wie bei der Methode $\mathsf{Sel\_Min1}$ verfahren.

– $v_j = w_j$: Es sind $v_j = 1$ und $W_j = 1$. Das bedeutet, daß sich *innerhalb* des Intervalls *kein* 2er-Potenzwert befindet. Zur Bestimmung des nächstliegenden 2er-Potenzwertes *außerhalb* des Intervalls wird wie folgt verfahren: Der jeweilige Abstand zwischen dem unterhalbliegenden 2er-Potenzwert 2^j und dem oberhalbliegenden 2^{j+1} wird verglichen. Derjenige mit dem geringeren Abstand wird gewählt; bei gleichem Abstand wird willkürlich der obere 2er-Potenzwert gewählt.

m:=8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
32	0	0	1	0	0	0	0	0
⋮				⋮				
v:=50	0	0	1	1	0	0	1	0
51	0	0	1	1	0	0	1	1
52	0	0	1	1	0	1	0	0
53	0	0	1	1	0	1	0	1
54	0	0	1	1	0	1	1	0
w:=55	0	0	1	1	0	1	1	1
⋮				⋮				
64	0	1	0	0	0	0	0	0

Tabelle 5.4: Beispiel zur Methode Sel_NP2

In Tabelle 5.4 (S. 86) wird die Methode Sel_NP2 anhand eines Beispiels veranschaulicht: Wie beim Beispiel von Sel_Min1 in Tabelle 5.3 (S. 85) sei ebenfalls $m := 8$ gegeben; ferner sei ein Intervall $[v, w]$ mit $v := 50$ und $w := 55$ gegeben. Die Methode Sel_NP2 bestimmt j zu 5, da gilt: $v_7 = w_7 = 0$, $v_6 = w_6 = 0$ und $v_5 = w_5 = 1$. Als unterhalbliegender 2er-Potenzwert ergibt sich hier $32 = 2^j$ und als oberhalb gelegener $64 = 2^{j+1}$. Als Abstände ergeben sich nach unten $v - 2^j = 50 - 32 = 18$ und nach oben $2^{j+1} - w = 64 - 55 = 9$, d. h. es wird der oberhalbliegende 2er-Potenzwert $2^6 :=$ Sel_NP2$(50, 55)$ selektiert. Die Selektion eines innerhalb des Intervalls gelegenen 2er-Potenzwertes ist am Beispiel in Tabelle 5.3 (S. 85) entsprechend nachvollziehbar.

Für einen vergleichenden Überblick wird im folgenden Abschnitt 5.1.4.4 (S. 86) das Beispiel zur Demonstration der Intervall-Methode aus Abschnitt 5.1.3.7 (S. 82) aufgegriffen.

5.1.4.4 Demonstration der Selektions-Methode

Die von den drei Modi der Intervall-Methode in Abschnitt 5.1.3.7 (S. 82) erzeugten Intervall-Folgen Int_UpDn, Int_Abs und Int_Rel (vgl. Tabelle 5.2, S. 82) seien hier für die Demonstration der Selektions-Methoden verwendet. Zunächst zeigt Tabelle 5.5 (S. 87) die Selektions-Methode, die anhand der durch Int_UpDn generierten Intervalle einzelne Werte wählt.

Int_UpDn		Selektions-Methode				
i	[v,w]	Sel_Low	Sel_High	Sel_Mean	Sel_Min1	Sel_NP2
0	$[0,0]$	0	0	0	0	0
1	$[2,3]$	2	3	2	$[2,3]$	2
2	$[5,6]$	5	6	6	6	$\underline{4}$
3	$[13,14]$	13	14	14	14	$\underline{16}$
4	$[22,23]$	22	23	22	$[22,23]$	$\underline{16}$
5	$[30,31]$	30	31	30	$[30,31]$	$\underline{32}$
6	$[47,48]$	47	48	48	48	$\underline{32}$
7	$[69,70]$	69	70	70	70	$\underline{64}$

Tabelle 5.5: Demonstration der Selektions-Methode mit Int_UpDn

Bemerkenswert ist beim Sel_Mean-Modus, daß, wie bei der Definition erläutert wurde, in den Fällen einer ungeraden Summe von $v + w$ (was hier immer zutrifft, da die Weite $w - v = 1$ ist) durch das Runden mit $[\,]$ tatsächlich immer gerade Zahlen resultieren. Beim Sel_Min1-Modus sind die Intervalle dort erhalten geblieben, wo sich dies für eine Kodierung mit $*$ (Don't-Care) anbietet. Im übrigen wählt auch Sel_Min1 immer gerade Zahlen aus. Für den Sel_NP2-Modus ergaben sich fast nur 2er-Potenzwerte außerhalb der jeweiligen Intervalle; sie sind unterstrichen. Am kritischsten geschah die Auswahl bei $i = 6$, wo der Abstand des gewählten unteren 2er-Potenzwertes $2^5 = 32$ zu $v := 47$ mit $15 = 47 - 32$ immerhin fast soviel wie beim oberen 2er-Potenzwert $2^6 = 64$ zu $w := 48$ mit $16 = 64 - 48$ beträgt.

Int_Abs		Selektions-Methode				
i	[v,w]	Sel_Low	Sel_High	Sel_Mean	Sel_Min1	Sel_P2
0	$[0,2]$	0	2	1	2	2
1	$[0,5]$	0	5	2	4	4
2	$[2,8]$	2	8	5	8	8
3	$[10,16]$	10	16	13	16	16
4	$[19,25]$	19	25	22	24	$\underline{16}$
5	$[27,33]$	27	33	30	32	32
6	$[44,50]$	44	50	47	48	$\underline{32}$
7	$[66,72]$	66	72	69	72	$\underline{64}$

Tabelle 5.6: Demonstration der Selektions-Methode mit Int_Abs

Etwas anders stellt sich die Kombination der Selektions-Methode mit den Intervallen von Int_Abs in Tabelle 5.6 (S. 87) dar. Im Gegensatz zur vorhergehenden Demonstration resultieren beim Sel_Min1-Modus nicht nur gerade Zahlen, da hier in der Regel die Weite der Intervalle $w - v > 1$ ist; daher sind nicht alle der Summen $v + w$ ungerade, so daß bei der Halbierung aus geraden Summen auch ungerade Summanden hervorgehen.

Bei Sel_Min1 besteht ebenfalls ein Unterschied: Während in der vorherigen Demonstration noch Intervalle beibehalten wurden, um so eine Kodierung mit $*$ zu ermöglichen, muß bei diesen Intervallen eine Zahl mit minimaler 1en-Anzahl gewählt werden. Dabei resultieren ausschließlich Zahlen, die innerhalb der jeweiligen Intervalle liegen. Der Sel_NP2-Modus hingegen muß hier seltener einen 2er-Potenzwert außerhalb (unterstrichen) des Intervalls zuordnen als innerhalb.

In Tabelle 5.7 (S. 88) ist die Situation bei $i := 5$ mit den Intervallgrenzen $v := 29$ und $w :=$

Int_Rel		Selektions-Methode				
i	[v,w]	Sel_Low	Sel_High	Sel_Mean	Sel_Min1	Sel_P2
0	0	0	0	0	0	0
1	$[2,3]$	2	3	2	$[2,3]$	2
2	$[5,6]$	5	6	6	6	$\underline{4}$
3	$[12,14]$	12	14	13	14	$\underline{16}$
4	$[21,24]$	21	24	22	24	$\underline{16}$
5	$[29,33]$	29	33	31	32	32
6	$[45,51]$	45	51	48	48	$\underline{64}$
7	$[65,73]$	65	73	69	72	$\underline{64}$

Tabelle 5.7: Demonstration der Selektions-Methode mit Int_Rel

33 bemerkenswert: Während bei Sel_Min1$(29,33)$ und Sel_NP2$(29,33)$ als Ergebnis der *PLA-gerechte* 2er-Potenzwert 32 selektiert wird, kommt es beim Sel_Mean-Modus zum wesentlich ungünstigeren Resultat 31. Dies zeigt, daß dieser Modus mit der Zuordnung des Mittelwertes allenfalls numerischen Belangen Rechnung trägt; bezüglich der Reduzierung von PLA-Kosten kann es nur zufällig zur günstigen Resultaten kommen.

5.2 Methoden für die Binär-Schicht

In der bisherigen Diskussion sind Optionen bezüglich der Numerik-Schicht und Methoden der Konversions-Schicht erläutert worden. Im folgenden treten Ansätze in den Mittelpunkt, die auf der Menge $\mathbb{B}^m$ bzw. $\mathbb{D}^m$ (vgl. Gleichung 1.6, S. 6) der binär kodierten Funktionswerte arbeiten. Obwohl mit deren Zusammenstellung in Gestalt einer Wahrheitstabelle bereits ein mögliches Resultat einer Spezifizierung vorliegt, das für den Entwurf eines PLAs Verwendung finden kann, werden durch die in den nächsten Abschnitten vorgestellten Binär-Methoden noch weitergehende Modifikationen möglich.

Nach Abschluß der Spezifizierung wird die Wahrheitstabelle stets einer Prozedur zur Logik-Minimierung wie beispielsweise ESPRESSO [Bra*82b, Bra*84] zugeführt. Es ist daher sinnvoll, die Methoden innerhalb der Binär-Schicht auf eine Steigerung der Effektivität bei den Algorithmen der Logik-Minimierung auszurichten: Dabei besteht ein erster Hinweis auf die Anzahl der Terme durch die Anzahl der 1en (vgl. Gleichung 2.12, S. 28). Dieser Zusammenhang sollte allerdings eher in Form der Aussage *Wenig* 1*en führen zu wenigen Termen* interpretiert werden, als in Gestalt der komplementären Aussage *Viele* 1*en führen zu vielen Termen.* So können identische Tabelleneinträge, die viele 1en aufweisen und zunächst zu vielen Mintermen führen (vgl. Abschnitt 1.2.3.2, S. 9), nach der Logik-Minimierung durch sehr wenige Terme überdeckt werden.

5.2.1 Binäre Übertragsbildung

5.2.1.1 Arbeitshypothese: Pseudo-Überträge

Ein effektives Grundprinzip zur Reduzierung der Termanzahl ist die Senkung der 1en-Anzahl in den Einträgen der PLA-Tabelle. Ein typisches Merkmal beim Durchlaufen eines binären Wertebereiches im Dualsystem sind die häufigen Überträge zwischen den Bitstellen, wodurch die 1en-Anzahl in benachbarten Werten starken Schwankungen unterliegt. Beispielsweise führt das Inkrementieren von $31 = 01\,1111$ bzw. das Dekrementieren von $32 = 10\,0000$ zu solchen Schwankungen bei der Anzahl der 1en. Werden nun Werte, die viele 1en aufweisen, um geringe Beträge erhöht bzw. erniedrigt, so können sich benachbarte Substitutionswerte ergeben, die weniger 1en besitzen. In diesem Zusammenhang sei künftig von *Pseudo-Übertragsbildung* gesprochen, da die Überträge künstlich herbeigeführt werden und ggf. über direkt benachbarte Werte hinausgehen. Je nach Erhöhung bzw. Erniedrigung sei dies durch *Pseudo-Inkrementierung* bzw. *Pseudo-Dekrementierung* bezeichnet. Die *Arbeitshypothese* für eine Methode könnte somit folgendermaßen lauten:

Durch geringfügige numerische Änderungen an Einträgen kann eine Senkung der 1en-Anzahl erreicht werden.

5.2.1.2 Methode: Zwei Modi zur Erzeugung von Pseudo-Überträgen

Ein maximaler Effekt wäre bei der Substitution mit 2er-Potenzwerten zu erzielen (vgl. Abschnitt 5.1.2, S. 77f); allerdings müßten dabei hohe Abweichungen hingenommen werden. Eine verfeinerte Möglichkeit besteht in der Substitution durch naheliegende Werte, deren Erzeugung ähnlich geschieht, wie die Übertragsbildung beim Inkrementieren bzw. Dekrementieren eines

Wertes; die beiden Modi der Methode heißen dementsprechend Psd_Inc (engl. *Pseudo Increment*) bzw. Psd_Dec (engl. *Pseudo Decrement*). Für einen Funktionswert $y \in I\!N$ sei durch $\mathsf{Bin}_m(y)$ seine m-Bit Binärkodierung $y_{m-1}\dots y_0$ gegeben. Die Methode mit ihren Modi Psd_Inc und Psd_Dec arbeitet auf binär kodierte Einträge, die auch $*$ enthalten können; es wird somit folgende Abbildung durchgeführt:

$$\mathsf{Psd\_Inc} \text{ bzw. } \mathsf{Psd\_Dec} : \; I\!D^m \to I\!B^m \tag{5.21}$$

Ausgehend vom MSB wird y zunächst nach einer zusammenhängenden Sequenz aus benachbarten Bitstellen $\neq \mathsf{0}$ beim Psd_Inc-Modus bzw. benachbarten Bitstellen $\neq \mathsf{1}$ beim Psd_Dec-Modus durchsucht. Dabei wird die Angabe einer Mindestlänge für die zusammenhängenden Sequenzen gewissermaßen als Schwellenwert benötigt. Dadurch kann die Methode gesteuert werden, ab welcher Länge einer Sequenz die Substitution durchgeführt werden soll. Für beide Modi Psd_Inc und Psd_Dec sei dazu e als Parameter eingeführt, wobei für e gilt:

$$2 \leq e \leq m-1, \tag{5.22}$$

d. h. Längen kleiner als 2 Bit sind nicht sinnvoll und sollten zudem nicht größer als höchstens $m-1$ sein, da sonst für die Substitution innerhalb der m-Bit Darstellung keine Bitstelle für einen binären Übertrag übrig ist. Aus diesen Restriktionen für den Parameter e folgt, daß die Methode erst ab $m \geq 3$ angewendet werden kann. Nachfolgend wird jeder Modus einzeln erläutert:

- **Pseudo-Inkrementierung**
  Zur Beschreibung der Zuordnung, die durch die Abbildung $\mathsf{Psd\_Inc}_e(y_{m-1} \cdots y_0)$ durchgeführt wird, sei eine Bitposition mit $j < m-1$ gegeben, an der, vom MSB ausgehend, die erste Sequenz aus 1en mit der erforderlichen Mindestlänge von e beginnt. An der Position $j+1 \leq m-1$ muß eine 0 existieren, da sonst die 1-Sequenz schon dort beginnen würde und ein beabsichtigter Übertrag in der m-Bit Darstellung nicht mehr zu berücksichtigen wäre.

$$y \;:=\; y_{m-1} \cdots \mathsf{0}_{j+1}\, \overbrace{\mathsf{1}_j \cdots \mathsf{1}_{j-e-1}}^{e\text{-1en}}\, y_{j-e-2} \cdots y_0 \tag{5.23}$$

$$\mathsf{Psd\_Inc}_e(y) \;:=\; y_{m-1} \cdots \mathsf{1}_{j+1}\, \underbrace{\mathsf{0}_j \cdots \mathsf{0}_{j-e-1}\, \mathsf{0}_{j-e-2} \cdots \mathsf{0}_0}_{(j+1)\text{-0en}} \tag{5.24}$$

  Im einzelnen findet die Substitution folgendermaßen statt: An der Position $j+1$ wird die 0 durch eine 1 substituiert und alle folgenden Bits ab Position j bis zum LSB durch 0 ersetzt. Der resultierende Funktionswert hat nun weniger 1en und ist numerisch betrachtet nur geringfügig erhöht worden. Da im Verlauf dieser Methode 1en gesetzt werden, so daß erneut 1-Sequenzen mit einer Länge von e oder mehr entstehen können, läßt sie sich ggf. mehrfach wiederholen, bis sie schließlich erschöpft ist, da kein Übertrag mehr darstellbar ist.

$e := 2 \quad m := 8$	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
$y :=$ 55	0	0	1	1	0	1	1	1
$\mathsf{Psd\_Inc}_2$: 64	0	1	0	0	0	0	0	0

$e := 3 \quad m := 8$	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
$y :=$ 55	0	0	1	1	0	1	1	1
$\mathsf{Psd\_Inc}_3$: 56	0	0	1	1	1	0	0	0
$\mathsf{Psd\_Inc}_3$: 64	0	1	0	0	0	0	0	0

Tabelle 5.8: Beispiel zum Psd_Inc-Modus

Anhand eines Beispiels soll die Methode Psd_Inc demonstriert werden: Es sei dabei $m := 8$ und das Argument $y := 55$. In Tabelle 5.8 (S. 91) sind die Fälle $e := 2$ und $e := 3$ aufgeführt. Mit $\mathsf{Bin}_8(55)$ ergibt sich daraus die 8-Bit Kodierung des Argumentes zu 0011 0111 mit 5 1en. Der Parameter e zur Festlegung der Mindestlänge für zu erkennende 1-Sequenzen lautet:

- $e := 2$: Die Methode $\mathsf{Psd\_Inc}_2$ substituiert vom MSB ausgehend die erste 1-Sequenz, die mindestens eine Länge von 2 hat. Dieses trifft ab $j = 5$ mit $y_5 = y_4 = 1$ zu, so daß sich als Ergebnis $64 = 0100\,0000 := \mathsf{Psd\_Inc}_2(0011\,0111)$ ergibt; es hat nur eine 1 und bedeutet gegenüber $y := 55$ eine Abweichung von $+9$. Die Methode $\mathsf{Psd\_Inc}_2$ ist damit ausgeschöpft.
- $e := 3$: Die Methode $\mathsf{Psd\_Inc}_3$ substituiert vom MSB ausgehend die erste 1-Sequenz, die mindestens eine Länge von 3 hat. Dieses trifft zunächst ab $j = 2$ mit $y_2 = y_1 = y_0 = 1$ zu und führt zum Ergebnis $56 = 0011\,1000 := \mathsf{Psd\_Inc}_3(0011\,0111)$; es hat drei 1en und bedeutet gegenüber $y := 55$ eine Abweichung von $+1$. Da durch die Methode $\mathsf{Psd\_Inc}_3$ auch 1en gesetzt werden, kann erneut eine 1-Sequenz in der geforderten Länge entstehen; das Beispiel zeigt einen derartigen Fall. In einer wiederholten Anwendung der Methode $\mathsf{Psd\_Inc}_3$ ergibt sich die folgende Situation: $\mathsf{Psd\_Inc}_3(0011\,1000) := 0100\,0000 = 64$. Das Ergebnis hat nun wie vorher bei $e := 2$ den Wert 64 und hat nur eine 1. Die Methode $\mathsf{Psd\_Inc}_3$ ist damit ausgeschöpft.

- **Pseudo-Dekrementierung**
Zur Beschreibung der Zuordnung, die durch die Abbildung $\mathsf{Psd\_Dec}_e(y_{m-1} \cdots y_0)$ durchgeführt wird, sei eine Bitposition mit $j < m - 1$ gegeben, an der, vom MSB ausgehend, die erste 0-Sequenz mit der erforderlichen Mindestlänge von e beginnt. An der Position $j + 1 \leq m - 1$ muß eine 1 existieren, da sonst die 0-Sequenz schon dort beginnen würde und bei der beabsichtigten Ausdehnung der 0-Sequenz bis zum LSB der Wert 0 entstehen würde.

$$y \;:=\; y_{m-1} \cdots 1_{j+1}\, \overbrace{0_j \cdots 0_{j-e-1}}^{e\text{ -0en}}\, y_{j-e-2} \cdots y_0 \tag{5.25}$$

$$\mathsf{Psd\_Dec}_e(y) \;:=\; y_{m-1} \cdots 1_{j+1}\, \underbrace{0_j \cdots 0_{j-e-1}\, 0_{j-e-2} \cdots 0_0}_{(j+1)\text{ -0en}} \tag{5.26}$$

Im einzelnen findet die Substitution folgendermaßen statt: Ab Position j bis zum LSB werden alle Bitstellen durch 0 ersetzt. Der resultierende Funktionswert hat nun weniger 1en und ist numerisch betrachtet nur geringfügig erniedrigt worden. Durch die erfolgte Substitution ist die Methode bereits erschöpft.

$e := 2 \quad m := 8$	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
$y :=$ 167	1	0	1	0	0	1	1	1
$\mathsf{Psd\_Dec}_2$: 160	1	0	1	0	0	0	0	0

$e := 3 \quad m := 8$	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
$y :=$ 71	0	1	0	0	0	1	1	1
$\mathsf{Psd\_Dec}_3$: 64	0	1	0	0	0	0	0	0

Tabelle 5.9: Beispiel zum Psd_Dec-Modus

Anhand eines Beispiels soll die Methode Psd_Dec demonstriert werden: Es sei dabei $m := 8$. In Tabelle 5.9 (S. 92) sind die Fälle $e := 2$ und $e := 3$ aufgeführt. Mit $\mathsf{Bin}_8(167)$ bzw. $\mathsf{Bin}_8(71)$ ergibt sich die jeweilige 8-Bit Kodierung des Argumentes zu 1010 0111 mit 5 1en bzw. zu 0100 0111 mit 4 1en. Der Parameter e zur Festlegung der Mindestlänge für zu erkennende 0-Sequenzen lautet

- $e := 2$: Die Methode $\mathsf{Psd\_Dec}_2$ substituiert vom MSB ausgehend die erste 0-Sequenz, die mindestens eine Länge von 2 hat. Dieses trifft in $y = 167 =$ 1010 0111 ab $j := 4$ mit $y_4 = y_3 =$ 0 zu und führt zu dem Ergebnis 160 = 1010 0000 := $\mathsf{Psd\_Dec}_2$(1010 0111); es hat nur zwei 1en und bedeutet gegenüber $y := 167$ eine Abweichung von -7.
- $e := 3$: Die Methode $\mathsf{Psd\_Dec}_3$ substituiert vom MSB ausgehend die erste 0-Sequenz, die mindestens eine Länge von 3 hat. Dieses trifft in $y = 71 =$ 0100 0111 auf $j := 5$ mit $y_5 = y_4 = y_3 =$ 0 zu und führt zu dem Ergebnis 64 = 0100 0000 := $\mathsf{Psd\_Dec}_3$(0100 0111); es hat eine 1 und bedeutet gegenüber $y := 71$ eine Abweichung von -7. Die Methode $\mathsf{Psd\_Dec}_3$ ist damit ausgeschöpft.

Die maximale Anzahl der Werte für den Parameter e aus dessen Bereich (vgl. Gleichung 5.22, S. 90) ergibt sich wie folgt: Unter Vernachlässigung der Kaskadierbarkeit von Psd_Inc sind für $m \geq 3$ die eine Anzahl von $m - 2$ Werte für den Parameter e möglich. Mit der Psd-Methode können somit für beide Modi insgesamt bis zu $2 \cdot (m - 2)$ verschiedene Modifikationen vorgenommen werden, die im allgemeinen auch zu unterschiedlichen PLA-Tabellen führen.

5.2.1.3 Demonstration der Methode für Pseudo-Überträge

Binäre Folge			Methode Pseudo-Überträge						
i	$Y_i := [q_i]$	$\mathsf{Bin}_7(Y_i)$	$\mathsf{Psd\_Inc}_2$	$\mathsf{Psd\_Inc}_2$	$\mathsf{Psd\_Inc}_3$	$\mathsf{Psd\_Inc}_4$	$\mathsf{Psd\_Inc}_5$	$\mathsf{Psd\_Dec}_2$	$\mathsf{Psd\_Dec}_3$
0	0	000 0000	000 0000	000 0000	000 0000	000 0000	000 0000	000 0000	000 0000
1	2	000 0010	000 0010	000 0010	000 0010	000 0010	000 0010	000 0010	000 0010
2	6	000 0110	000 1000	000 1000	000 0110	000 0110	000 0110	000 0110	000 0110
3	13	000 1101	0010 000	001 0000	000 1101	000 1101	000 1101	000 1101	000 1101
4	22	001 0110	001 1000	010 0000	001 0110	001 0110	001 0110	001 0110	001 0110
5	31	001 1111	010 0000	010 0000	010 0000	010 0000	010 0000	001 1111	001 1111
6	48	011 0000	100 0000	100 0000	011 0000	011 0000	011 0000	011 0000	011 0000
7	69	100 0101	100 0101	100 0101	100 0101	100 0101	100 0101	100 0000	100 0000

Tabelle 5.10: Demonstration der Psd-Methode

Aus Tabelle 5.1 (S. 78) wird zur Demonstration der beiden Modi Psd_Inc und Psd_Dec die reelle Folge der acht q_i (dort zweite Spalte) übernommen. Sie sind als gerundete Werte in der zweiten Spalte von Tabelle 5.10 (S. 92) wiedergegeben. Daneben ist die binäre 7-Bit Darstellung der gerundeten Werte aufgeführt. Jeweils davon ausgehend werden die Ergebnisse der beiden Modi Psd_Inc und Psd_Dec aufgeführt. Dabei sind sie mit den jeweils nach der Restriktion $2 \leq e \leq m-1$ möglichen Werten für den Parameter e ausgewertet worden. Alle dabei modifizierten Einträge sind durch Unterstreichung markiert. Eine Besonderheit ergibt sich in der fünften Spalte, in der von den Ergebnissen der davon links aufgeführten Auswertung von $\mathsf{Psd\_Inc}_2$ ausgegangen wurde, da der Modus nach der erstmaligen Auswertung noch nicht erschöpft war. Dementsprechend bezieht sich die eine Unterstreichung auf die vorhergehende vierte Spalte und nicht auf die dritte Spalte der gerundeten Werte. Insgesamt ist zu ersehen, daß die Anwendung dieser Methode in ihren parametrisierten Modi eine Verringerung der 1en-Anzahl bewirkt.

5.2.2 Ähnlichkeit von binären Einträgen

5.2.2.1 Arbeitshypothese: Angleichung ähnlicher Einträge

In Wahrheitstabellen können identische Einträge auftreten, was sich allgemein günstig auf die Effektivität der Algorithmen in der Logik-Minimierung auswirkt. Dies gilt insbesondere für die *Consensus Methode* (vgl. Abschnitt 1.2.4.2, S. 12), die im allgemeinen Bestandteil der Algorithmen von Logik-Minimierern ist: Sie senkt die Anzahl der Terme durch das Kombinieren von jeweils zwei Termen zu einem Überdeckungs- bzw. Konsens-Term, der zudem um eine Variable bzw. ein Literal verkürzt ist (vgl. Abschnitt 1.2.4.2, S. 11). Nicht selten sind sich Einträge „ähnlich“, einerseits bezüglich ihrer Binärdarstellung, andererseits aber auch gemäß ihrer numerischen Interpretation. Es ist daher zu prüfen, ob sie durch geringfügige Änderungen einander angeglichen werden können. Durch viele identische Einträge an bestimmten Positionen in der Wahrheitstabelle kommt es zu einer Erhöhung der Anzahl derjenigen Terme, die durch die *Consensus Methode* miteinander kombiniert werden können. Die *Arbeitshypothese* für diese Methode lautet somit:

> Identische Einträge an bestimmten Positionen in der Wahrheitstabelle bewirken eine Reduzierung der PLA-Kosten.

Es bedarf der näheren Erläuterung, was unter dem Begriff *ähnlich* assoziiert werden soll; darauf geht der nächste Abschnitt näher ein. Während die Ähnlichkeit bei der numerischen Interpretation durch Beschränkung der numerischen Differenz zwischen zwei Einträgen erfaßt werden kann, wird eine binäre Ähnlichkeit durch den sogenannten *Hamming-Abstand* quantifizierbar.

Hamming-Abstand

Für den Vergleich von binär dargestellten Zahlen wird der *Hamming-Abstand* ausgewertet, der die Anzahl der unterschiedlichen Bitstellen angibt und wie folgt definiert ist:

$$\mathsf{Ham}: \mathbb{B}^k \times \mathbb{B}^k \to \mathbb{N},\ k \in \mathbb{N} \tag{5.27}$$

Für zwei verschiedene, binär kodierte Zahlen $a, b \in \mathbb{N},\ a \neq b$ mit $a_{k-1} \ldots a_0 := \mathsf{Bin}_k(a)$ und $b_{k-1} \ldots b_0 := \mathsf{Bin}_k(b)$ ergibt sich der Hamming-Abstand folgendermaßen:

$$\mathsf{Ham}(a_{k-1} \ldots a_0\,,\ b_{k-1} \ldots b_0) := \left|\{j \mid a_j \neq b_j,\ \text{wobei } j \in \mathbf{I}_k\}\right| \tag{5.28}$$

Anmerkung: Der Hamming-Abstand ist nicht für binäre Einträge definiert, die mit Hilfe von $*$ (Don't-Care) kodiert wurden.

5.2.2.2 Methode: Ähnliche Einträge

In der Methode Sim (engl. *Similarity*) sollen ähnliche Einträge einander angeglichen werden. Es kommen dafür nicht generell alle Einträge in Betracht, da für die *Consensus Methode* eine Angleichung nur bei bestimmten (vgl. Abschnitt 5.2.2.1, S. 93) Einträgen nützlich ist. Daher müssen zunächst Kandidaten ausgewählt werden, die auf ihre Ähnlichkeit hin untersucht werden sollen. Solche Einträge seien künftig *vergleichbar* genannt. In einem nächsten Schritt können dann *vergleichbare* Einträge auf ihre Ähnlichkeit geprüft werden, d. h. daß zum Attribut *vergleichbar* noch das Attribut *ähnlich* hinzutritt. Beide sollen im folgenden erläutert werden.

Die Funktion f_{int} sei nach Abschnitt 3.3.5 (S. 53f) durch f_{tab} in Gestalt einer Wahrheitstabelle binär repräsentiert. Es seien $a, b \in \mathbf{I}_N$ zwei Indizes aus der Indexmenge und $c, d \in \mathbf{I}_M$ die zugeordneten Funktionswerte, d. h. $c := f_{\text{int}}(a)$ und $d := f_{\text{int}}(b)$.

- **vergleichbar:** Eine Angleichung von Einträgen ist nur bei bestimmten Kandidaten sinnvoll, nämlich solche, die bei der *Consensus Methode* zur Kombination von Termen führen. Da die Terme zunächst als Minterme aus den Indizes einer Wahrheitstabelle gebildet werden, müssen für das Prüfen des Attributes *vergleichbar* die den Einträgen zugeordneten Indizes $a, b \in \mathbf{I}_N$ betrachtet werden: Da es sich um verschiedene Einträge handelt, die miteinander verglichen werden sollen, soll $a < b$ gelten. Es seien $a_{n-1} \ldots a_0 := \mathsf{Bin}_n(a)$ und $b_{n-1} \ldots b_0 := \mathsf{Bin}_n(b)$ ihre n-Bit Binärdarstellungen.

  Als Kandidaten zur Angleichung kommen nur solche Einträge c und d in Frage, deren Indizes a und b einander numerisch und binär „nahe" sind.

  - **binär nahe:** $\mathsf{Ham}(a_{n-1} \ldots a_0\, ,\, b_{n-1} \ldots b_0) = 1$
    Bei der Logik-Minimierung können nur diejenigen Minterme durch die *Consensus Methode* zusammengefaßt werden, die einen Hamming-Abstand von 1 haben.
  - **numerisch nahe:** $|a - b| = 1$
    In der Regel weisen die zu realisierenden Funktionen numerische Eigenschaften, wie z. B. Stetigkeit und Monotonie auf, die durch die Angleichung von weit auseinanderliegenden Einträgen zerstört werden könnten. Nur für nebeneinanderliegende Einträge in der Wahrheitstabelle kann davon ausgegangen werden, daß eine Angleichung nicht zur Zerstörung der vorher erwähnten Funktionseigenschaften führt.

  Zusammenfassend kann festgestellt werden, daß in diesem Sinne binär und numerisch nahe Indizes nur durch benachbarte Zahlen gebildet werden können, von denen a gerade und b ungerade ist: Es seien a und b mit $a < b$ einander numerisch nahe, d. h. es gilt $b - a = 1$. Wegen der binären Nähe sollen sich ihre m-Bit Binärdarstellungen $a_{n-1} \ldots a_0$ und $b_{n-1} \ldots b_0$ in nur einem Bit unterscheiden, dann kann es nur das LSB sein, da es als einzige Bitstelle mit der Wertigkeit von 2^0 die numerische Differenz von 1 darstellen kann; alle anderen Bitstellen würden aufgrund ihrer höheren Wertigkeit größere Differenzen bewirken. Es ergibt sich nun für das Attribut *vergleichbar*, welches sich zwar auf Einträge c und d bezieht, jedoch an deren Indizes a und b geprüft wird, folgende Definition:

  $$\text{Einträge } c \text{ und } d \text{ sind } \textit{vergleichbar} \iff \text{Index } a \textit{ gerade} \text{ und } b \textit{ ungerade} \tag{5.29}$$

  Durch die Kombination der beiden Bedingungen für binäre und numerische Nähe wird die Menge der für Vergleiche heranzuziehenden Einträge klein gehalten. Während die Bedingung für binäre Nähe sich wegen der *Consensus Methode* unmittelbar aus der Arbeitshypothese motiviert, dient die zweite Bedingung für die numerische Nähe eher als eine

Restriktion: Neben der Berücksichtigung von Funktionseigenschaften sorgt sie dafür, daß die Methode nicht durch multiple Angleichungsmöglichkeiten unnötig komplexer wird. Es würde sonst die Ähnlichkeit nicht nur zwischen benachbarten Einträgen geprüft werden, sondern auch zwischen weit auseinanderliegenden, so daß zusätzliche Möglichkeiten für Angleichungen entstehen könnten. In solchen Fällen müßte eine Abwägung stattfinden, für welche Einträge eine Angleichung erfolgen soll. Eigene Tests ließen den höheren Aufwand im Verhältnis zu den erreichten Effekten als nicht gerechtfertigt erscheinen, so daß an dieser Stelle darauf verzichtet wird.

- **ähnlich:** Die den Indizes $a, b \in \mathbf{I}_N$ zugeordneten Einträge $c := f_{\text{int}}(a)$ und $d := f_{\text{int}}(b)$ können bezüglich ihrer Ähnlichkeit einerseits numerisch, andererseits aber auch binär interpretiert werden, so daß zwei Formen der Ähnlichkeit existieren:

  - **numerisch ähnlich:** Zur Bestimmung der numerischen Ähnlichkeit wird die Differenz zwischen den beiden Einträgen gebildet:

  $$\text{numerische Ähnlichkeit:} \quad s := |c - d| \tag{5.30}$$

  - **binär ähnlich:** Die binäre Ähnlichkeit der Einträge bezieht sich auf deren m-Bit Binärdarstellung $c_{m-1} \ldots c_0 := \mathsf{Bin}_m(c)$ und $d_{m-1} \ldots d_0 := \mathsf{Bin}_m(d)$ und läßt sich nun mit Hilfe des Hamming-Abstandes bestimmen:

  $$\text{binäre Ähnlichkeit:} \quad s := \mathsf{Ham}(c_{m-1} \ldots c_0\,,\ d_{m-1} \ldots d_0) \tag{5.31}$$

Für jede der beiden Ähnlichkeitsformen gibt es einen eigenen Modus der Methode: Für die numerische Ähnlichkeit lautet dieser Sim_Num und für die binäre Ähnlichkeit entsprechend Sim_Bin. Beiden Modi ist gemeinsam, daß die Ähnlichkeit gegen eine gegebene Schranke nach oben abgeschätzt wird. Mit $z_{\text{bin}}, z_{\text{num}} \in I\!N$ als der jeweilige Parameter, wobei $z_{\text{bin}} > 0$ und $z_{\text{num}} > 0$ gilt, läßt sich für beide Modi folgendes definieren:

$$\begin{array}{rcl} \text{Einträge} \quad c \text{ und } d & & \text{sind } \textit{ähnlich} \\ & \Longleftrightarrow & s \le z_{\text{bin}} \\ \text{bzw.} & \Longleftrightarrow & s \le z_{\text{num}} \end{array} \tag{5.32}$$

Bei der binären Ähnlichkeit ist z_{bin} nach oben durch die Anzahl m der möglichen Bitstellen begrenzt: $0 < z_{\text{bin}} \le m$. Im Extremfall wären bei $z_{\text{bin}} := m$ sogar Einträge *ähnlich*, bei denen *alle* Bitstellen verschieden sind; dies würde die Angleichung aller *vergleichbaren* Einträge bedeuten. Im Falle von Sim_Bin ergibt sich somit eine maximale Anzahl von m Werten für den Parameter z_{bin}. Dagegen wäre bei Sim_Num für die numerische Ähnlichkeit der Extremfall durch die größte numerische Differenz unter *vergleichbaren* Einträgen gegeben, d. h. es gilt:

$$z_{\text{num}} \le \max\left(\left\{|c_i - d_j| \;\middle|\; i \ne j \quad \text{und} \quad c_i \text{ vergleichbar } d_j\right\}\right) \tag{5.33}$$

Die maximal mögliche Differenz zwischen *vergleichbaren* Einträgen hängt entscheidend von den Eigenschaften der Funktion f ab. Bei der Diskretisierung in der zweiten Stufe der Spezifizierung sollte die Quantisierungseinheit ε so gewählt werden, daß steile Anstiege im Graphen von f durch angemessen viele Stützstelle erfaßt werden. Damit die auftretenden Differenzen zwischen *vergleichbaren* Einträgen nicht zu groß sind, sollte die Quantisierungseinheit ε entsprechend klein sein.

Zusammenfassend kann festgestellt werden, daß als Auswahlkriterium und Voraussetzung für eine Angleichung von Einträgen zunächst zu prüfen ist, ob sie *vergleichbar* sind. Für zwei als *vergleichbar* erkannte Einträge ist dann die Frage zu stellen, wie *ähnlich* sie einander sind, d. h. ob sie die Beschränkung durch den Parameter z erfüllen können. Nun kann derjenige Eintrag, der mehr 1en hat, durch den anderen ersetzt werden, um dadurch die 1en-Anzahl zu minimieren. Bei gleicher 1en-Anzahl unterbleibt eine Angleichung, da in diesen Fällen in Hinsicht auf die 1en-Anzahl keine Verbesserung erzielt werden könnte.

5.2.2.3 Demonstration der Methode für ähnliche Einträge

Mit Hilfe des *Sinus*-Beispiels soll die Methode nachfolgend veranschaulicht werden. Anhand eines Auszuges von 8 Einträgen aus der Tabelle B.1 (S. 158) wird für beide Modi sowohl das Prüfen der binären und numerischen Ähnlichkeit bei vergleichbaren Einträgen als auch die ggf. vollzogene Angleichung von Einträgen demonstriert.

vergleichbar		*ähnlich*					**Angleichung**	
i	$\mathsf{Bin}_7(i)$	$f_{\text{int}}(i)$	$\mathsf{Bin}_{10}(f_{\text{int}}(i))$	1en	Ham()	Diff.	Sim_Bin $z := 3$	Sim_Num $z := 13$
32	001 0000	486	01 1110 0110	6	2	14	-	-
33	001 0001	500	01 1111 0100	6			1en gleich	Diff. $> z$
34	001 0010	514	10 0000 0010	2	3	13	↓	↓
35	001 0011	527	10 0000 1111	5				
36	001 0100	541	10 0001 1101	5	5	13	-	↑
37	001 0101	554	10 0010 1010	4			Ham $> z$	
38	001 0110	567	10 0011 0111	6	5	13	-	↑
39	001 0111	580	10 0100 0100	3			Ham $> z$	

Tabelle 5.11: Demonstration der Sim-Methode

In Tabelle 5.11 (S. 96) sind zunächst von links beginnend die Indizes, nachfolgend die Einträge selbst und schließlich die Resultate der Methode zu betrachten. Vergleichbare Einträge sind durch die horizontalen Linien gruppiert. Als nächstes sind die Einträge in dezimaler und in binärer Form aufgeführt. Weiter ist die Anzahl der 1en, der Hamming-Abstand und die numerische Differenz zwischen den zu vergleichenden Einträgen vermerkt, so daß die Beurteilung der Ähnlichkeit erfolgen kann. In den beiden letzten Spalten werden schließlich die Resultate der Modi Sim_Bin bzw. Sim_Num mit den jeweiligen Parametern $z := 3$ bzw. $z := 13$ wiedergegeben. Beim ersten Vergleichspaar mit den Indizes 32 und 33 kommt es aufgrund der gleichen 1en-Anzahl nicht zur Angleichung. Der Modus Sim_Num ist ebenso davon betroffen wie der Modus Sim_Bin, jedoch war bei Sim_Num bereits die Ähnlichkeitsbedingung verletzt. Ansonsten werden die vorgenommenen Angleichungen in ihrer Zuweisungsrichtung durch die entsprechenden Pfeile angedeutet.

5.2.3 Überdeckung mit Hyper-Termen

Die im folgenden beschriebene Methode basiert auf Kernaussage 1 (S. 74), nach der in jeder Spezifikation inhärente Unschärfe vorhanden ist. Auch die im Rahmen der Logik-Minimierung durch Bilden der DNF (vgl. Abschnitt 1.2.3.2, S. 9) erzeugte Menge der Minterme sowie die daraus minimierte Termmenge trägt grundsätzlich Unschärfe in sich. Werden durch die nachfolgend vorgestellte Methode Modifikationen an der minimierten Termmenge vorgenommen, so entsteht bei einer Evaluation der modifizierten Termmenge eine andere Wahrheitstabelle, da die Überdeckung verändert wurde. Da auf diese Weise modifizierte Wahrheitstabellen erzeugt werden, ist diese Methode auch der Binär-Schicht zuzuordnen.

5.2.3.1 Arbeitshypothese: Zusätzliche Konsens-Terme

In der minimierten Gesamtmenge sind alle Terme enthalten, die für eine vollständige Überdeckung jeder einzelnen der m Logikfunktionen benötigt werden. Würde die Minimierung für jede einzelne Logikfunktion isoliert vorgenommen, so wäre die *Consensus Methode* (vgl. Abschnitt 1.2.4.2, S. 12) im Rahmen der Logik-Minimierung unter den dazugehörigen Termen bereits erschöpfend angewendet worden. Dies ist jedoch nicht der Fall, da stattdessen die Gesamtmenge minimiert wird, d. h. nicht die Minimalität der auf eine einzelne Logikfunktion bezogenen Termmenge ist entscheidend, sondern die der Gesamtmenge der Terme aller Logikfunktionen.

Wird hingegen die Gesamtmenge der Terme betrachtet, so können dabei durchaus Terme vorkommen, die noch durch die *Consensus Methode* zu kombinieren wären, so daß die Termanzahl weiter gesenkt werden kann. Die *Arbeitshypothese* lautet daher:

Eine zusätzliche Anwendung der *Consensus Methode* auf die Gesamtmenge der Terme reduziert die Termanzahl und damit die PLA-Kosten.

Da kombinierbare Terme allerdings zur exakten Überdeckung der Logikfunktionen notwendig sind, führt deren Verlust bzw. Ersetzung durch die Konsens-Terme zu einer Abweichung von der Gesamtfunktion f_{tab}, so daß hierdurch modifizierte Wahrheitstabellen entstehen.

5.2.3.2 Methode: Hyper-Terme

Ausgehend von einer Wahrheitstabelle, die durch die f-Variante $f_{\text{tab}} : I\!B^n \rightarrow I\!B^m$ repräsentiert wird, kann mit Hilfe einer Logik-Minimierung, beispielsweise mit dem Minimierer Espresso [Bra*82b, Bra*84], eine exakte Überdeckung der m Logikfunktionen durch eine minimierte Menge von Produkttermen erzeugt werden. Die Terme seien hier in der gebräuchlichen Schreibweise mit den Symbolen `-01` notiert, in der sie durch überdeckte Minterme identifiziert werden (vgl. Abschnitt 1.2.3.2, S. 9). Das bedeutet, daß $I\!D^n$ mit $I\!D := I\!B \cup \{\texttt{-}\}$ als Menge aller n-stelligen Terme mit $\emptyset$ als Null-Term aufgefaßt werden kann. Die zusätzliche Anwendung der *Consensus Methode* auf alle paarweise kombinierten Terme der bereits minimierten Termmenge $\mathbf{T} \subseteq I\!D^n$ (vgl. Abschnitt 1.2.4.2, S. 12) kann mehrere Konsens-Terme hervorbringen. Sie seien in der Menge $\mathbf{H}$ zusammengefaßt, so daß sich die Situation wie folgt formal darstellen läßt:

$$\begin{array}{rcll} \odot\,\text{paarweise}\ \mathsf{Cons} & : & \mathbf{T} \times \mathbf{T} \rightarrow \mathbf{H} & \textit{Consensus}\ \text{auf minimierte Termmenge,} \\ & \text{wobei} & \mathbf{T}, \mathbf{H} \subseteq I\!D^n & n\text{-stellige Termmengen.} \end{array} \tag{5.34}$$

Für jeden Kombinationsterm $h \in \mathbf{H}$ gibt es zwei Terme $t_i, t_j \in \mathbf{T}$ mit $i, j \in I\!N$ und $i \neq j$, so daß gilt: $h := \mathsf{Cons}(t_i, t_j)$. Da $\mathbf{T}$ ggf. nur Terme enthält, die paarweise disjunkt sind, d. h. wegen $\mathsf{Dist}(t_i, t_j) = 0$ (vgl. Abschnitt 1.2.4.2, S. 12) für alle $i, j \in I\!N$ mit $i \neq j$ ist $\mathsf{Cons}(t_i, t_j) = \emptyset$ immer der Null-Term, kann $\mathbf{H}$ bis auf $\emptyset$ leer sein. Existieren $h \in \mathbf{H}$ mit $h \neq \emptyset$, so handelt es sich um signifikante Ergebnisterme, die ein Literal weniger haben als die jeweils kombinierten Terme $t_i, t_j \in \mathbf{T}$. Solche Terme seien künftig als *Hyper-Terme* bezeichnet, die mindestens zwei Terme aus $\mathbf{T}$, häufig aber noch weitere, überdecken. Als Methode Hyp_Trm oder einfach nur Hyp wird nun die Hinzunahme jeweils eines Hyper-Terms $h \in \mathbf{H}$ zur minimierten Termmenge $\mathbf{T}$ bezeichnet, d. h. es gilt formal:

$$\begin{aligned} \mathsf{Hyp\_Trm} \text{ bzw. } \mathsf{Hyp} \quad &: \quad I\!D^n \to I\!D^n &(5.35)\\ \mathsf{Hyp\_Trm}(h) \quad &:= \quad \mathbf{T} \cup \{h\} \quad \text{wobei: } h \in \mathbf{H} &(5.36)\end{aligned}$$

$\mathsf{Hyp\_Trm}(h) \in I\!D^n$ stellt eine modifizierte Termmenge dar, deren Termanzahl um mindestens 1 verringert ist. Da durch den Hyper-Term h zusätzliche Überdeckungen hervorgerufen werden, treten allerdings Abweichungen an den betroffenen Logikfunktionen auf.

Für eine Abschätzung der Anzahl maximal möglicher Hyper-Terme sei folgende sehr optimistisch angenommene Situation betrachtet: Sei $\mathbf{T}$ eine Termmenge mit ℓ *wesentlichen* Primimplikanten (vgl. Abschnitt 1.2.4.2, S. 11), d. h. diese Terme sind zur Überdeckung der Logikfunktionen zwingend erforderlich. Ferner seien die einzelnen Terme paarweise verschieden und so beschaffen, daß für jedes Paar von ihnen die *Consensus Methode* einen neuen Hyper-Term erzeugt. Da es zur Paarbildung der ℓ Terme $\frac{\ell \cdot (\ell - 1)}{2}$ Kombinationsmöglichkeiten gibt, ist dies die maximale Anzahl der Hyper-Terme.

5.2.3.3 Demonstration der Hyper-Term-Methode

Anhand eines überschaubaren Beispiels soll die Hyp-Methode im folgenden demonstriert werden. Zu diesem Zweck werden hier die ersten drei Spalten von Tabelle 5.10 (S. 92) übernommen: Im linken Teil von Abbildung 5.12 (S. 98) ist u. a. die entsprechende Wahrheitstabelle dargestellt; die Indizes $i \in \mathbf{I}_8$ und Einträge sind hier zunächst in den beiden linken Spalten als Argumente X und Funktionswerte $Y := f_{\text{int}}$ aufgeführt. Die beiden rechten Spalten geben die f-Variante $f_{\text{tab}} : I\!B^3 \to I\!B^7$ in Gestalt der Wahrheitstabelle wieder.

X	Y	$x_2\, x_1\, x_0$	$y_6\, y_5\, y_4\;\, y_3\, y_2\, y_1\, y_0$
0	0	0 0 0	0 0 0 0 0 0 0
1	2	0 0 1	0 0 0 0 0 1 0
2	6	0 1 0	0 0 0 0 1 1 0
3	13	0 1 1	0 0 0 1 1 0 1
4	22	1 0 0	0 0 1 0 1 1 0
5	31	1 0 1	0 0 1 1 1 1 1
6	48	1 1 0	0 1 1 0 0 0 0
7	69	1 1 1	1 0 0 0 1 0 1

$\mathbf{T}$	:	t_1	t_2	t_3	t_4	t_5	t_6	t_7
y_0	:			011		101		111
y_1	:	-01	010		10-			
y_2	:		010	011	10-			111
y_3	:			011		101		
y_4	:				10-		110	
y_5	:						110	
y_6	:							111

Tabelle 5.12: Beispiel zur Methode Hyp mit Wahrheitstabelle und Mintermen

Zunächst sind aus dieser Wahrheitstabelle die Minterme zu generieren (vgl. Abschnitt 1.2.3.2, S. 9) und anschließend zu minimieren. Die resultierenden sieben Terme t_1 bis t_7 bilden die

Termmenge **T** und sind in Abbildung 5.12 (S. 98) auf der rechten Seite entsprechend ihrer Überdeckung bei den Logikfunktionen y_0 bis y_6 dargestellt. Aus dieser Termmenge können nun die insgesamt fünf Hyper-Terme h_1 bis h_5 gebildet werden, die die Termmenge **H** ausmachen; sie ergeben sich wie folgt:

$$
\begin{aligned}
h_1 &:= \texttt{1-1} &= \mathsf{Cons}(t_5, t_7)\\
h_2 &:= \texttt{-11} &= \mathsf{Cons}(t_3, t_7)\\
h_3 &:= \texttt{01-} &= \mathsf{Cons}(t_2, t_3)\\
h_4 &:= \texttt{11-} &= \mathsf{Cons}(t_6, t_7)\\
h_5 &:= \texttt{-10} &= \mathsf{Cons}(t_2, t_6)
\end{aligned}
\tag{5.37}
$$

In der Gesamtmenge der Terme **T** werden nun die kombinierbaren Terme t_2, t_3, t_5, t_6 und t_7 durch den jeweiligen Hyper-Term aus **H** ersetzt, was offenbar zu Abweichungen in der Gesamtfunktion führt. In Tabelle 5.13 (S. 99) ist dafür nach dem Index X und dem unveränderten Eintrag Y zunächst die anfängliche Binärdarstellung aufgeführt. Anschließend sind für den jeweiligen Hyper-Term h_1 bis h_5 die resultierenden Wahrheitstabellen dargestellt, wobei die veränderten Bitstellen unterstrichen sind.

Binäre Folge			Methode Hyper-Terme				
i	$Y_i := [q_i]$	$\mathsf{Bin}_7(Y_i)$	$\mathsf{Hyp\_Trm}(h_1)$	$\mathsf{Hyp\_Trm}(h_2)$	$\mathsf{Hyp\_Trm}(h_3)$	$\mathsf{Hyp\_Trm}(h_4)$	$\mathsf{Hyp\_Trm}(h_5)$
0	0	000 0000	000 0000	0000000	000 0000	000 0000	000 0000
1	2	000 0010	000 0010	0000010	000 0010	000 0010	000 0010
2	6	000 0110	000 0110	0000110	000 $\underline{1}$11$\underline{1}$	000 0110	0$\underline{11}$ 0110
3	13	000 1101	000 1101	$\underline{1}$001101	000 11$\underline{1}$1	000 1101	000 1101
4	22	001 0110	001 0110	0010110	001 0110	001 0110	001 0110
5	31	001 1111	$\underline{1}$01 1111	0011111	001 1111	001 1111	001 1111
6	48	011 0000	011 0000	0110000	011 0000	$\underline{1}$11 0$\underline{1}$0$\underline{1}$	011 0$\underline{11}$0
7	69	100 0101	100 $\underline{1}$101	100$\underline{1}$101	100 0101	1$\underline{11}$ 0101	100 0101

Tabelle 5.13: Demonstration der Methode Hyp

Bemerkenswert ist zunächst, daß bei allen fünf Resultaten von den sieben Termen in **T** jeweils einer eingespart und daher die Termanzahl gesenkt werden konnte. Dies geschieht bei einer gleichzeitigen Erhöhung[5] der 1en-Anzahl, was mit Hilfe der Unterstreichungen bei allen fünf Resultaten zu beobachten ist. Durch das Ersetzen von zwei kombinierbaren Termen in **T** durch einen Hyper-Term $h \in \mathbf{H}$ kommen „größere" Überdeckungen zustande, so daß zusätzliche 1en entstehen. Bei der Betrachtung der numerischen Veränderungen, führt lediglich die $h\!-\!3$-Variante zu einer eventuell akzeptablen Abweichung; bei den anderen sind unschwer Abweichungen in der Größenordnung von einer 2er-Potenz (Faktor 2) bei h_4, sowie drei 2er-Potenzen (Faktor 8) bei h_3 und h_5 zu verzeichnen. In der Regel kommt es beim Einsatz der Hyp-Methode zu Einsparungen bei den PLA-Kosten; geradezu typisch sind hohe numerische Abweichungen.

[5] Insofern wird einmal mehr deutlich, daß eine Senkung bei der 1en-Anzahl nur in einer ersten Annäherung als Hinweis auf eine Senkung der Termanzahl und damit zur Reduzierung von PLA-Kosten zu interpretieren ist.

5.3 Methoden im Überblick

5.3.1 Arbeitshypothesen und Vorgehensweisen

Im vorliegenden Kapitel 5 sind unterschiedliche Methoden zur Modifikation von Spezifikationen vorgestellt worden. Bezogen auf die Transformationen in der Schichten-Einteilung aus Abschnitt 4.1.4 (S. 60f), handelt es sich bei den diskutierten Methoden um alternative bzw. ergänzende Transformationsabbildungen.

Sie basieren jeweils auf dem *heuristischen Prinzip* (vgl. Fußnote 1, S. 75), wobei jede Methode eine individuelle *Arbeitshypothese* voraussetzt, die dem allgemeinen Ziel *Reduzierung der PLA-Kosten* folgt. In Tabelle 5.14 (S. 100) sind alle Methoden, die angenommene *Arbeitshypothese*, die zugrunde gelegten Ansätze und schließlich ein Referenzverweis zusammengetragen.

<table>
<tr><th>Methode</th><th>Arbeitshypothese</th><th>Vorgehensweise</th><th>Abschnitt</th></tr>
<tr><td>Rnd</td><td>Verschiedene Modi bewirken unterschiedliche PLA-Kosten.</td><td>Zulassen aller Rundungsmodi.</td><td>5.1.1</td></tr>
<tr><td>P2</td><td>Einträge mit nur einer 1 führen zu geringen PLA-Kosten.</td><td>Konvertieren nur zu 2er-Potenzwerten.</td><td>5.1.2</td></tr>
<tr><td>Int</td><td>Intervalle repräsentieren die durch Unschärfe bedingten Toleranzen.</td><td rowspan="2">Selektieren von Werten anhand individueller Intervalle.</td><td>5.1.3</td></tr>
<tr><td>Sel</td><td>PLA-gerechte Selektion von Werten reduziert PLA-Kosten.</td><td>5.1.4</td></tr>
<tr><td>Psd</td><td>Numerische Änderung an Einträgen senkt 1en-Anzahl.</td><td>Erzeugen von Pseudo-Überträgen.</td><td>5.2.1</td></tr>
<tr><td>Sim</td><td>Viele identische Einträge bewirken geringere PLA-Kosten.</td><td>Angleichen von ähnlichen Einträgen.</td><td>5.2.2</td></tr>
<tr><td>Hyp</td><td>Zusätzliche Anwendung der Consensus-Methode reduziert PLA-Kosten.</td><td>Erzeugen von nicht-exakten Überdeckungen durch Hyper-Terme.</td><td>5.2.3</td></tr>
</table>

Tabelle 5.14: Übersicht der Methoden mit Arbeitshypothese und Vorgehensweise

5.3.2 Parameter und Komplexität

In den Abschnitten 5.1 (S. 75ff) und 5.2 (S. 89ff) wurden *sieben* Methoden vorgestellt, wobei die Methoden Int und Sel durch ihre enge Kopplung als *eine* feste Kombination zusammengerechnet werden können. Dennoch werden je nach betrachteter Methode durch evtl. erforderliche Parameter und ggf. verschiedene Modi etliche Variationen möglich, so daß ebenso viele Modifikationen an einer Spezifizierung vorgenommen werden können. Tabelle 5.15 (S. 101) faßt diesbezüglich alle Methoden in einer Übersicht zusammen.

Methode	Modi	Parameter (Anzahl der Werte)	Komplexität (Laufzeit)
Rnd	3	*keine*	Besucht alle N Einträge einmalig, d. h. $\mathcal{O}(N)$.
P2	3	*keine*	*dito*
Int	3	Toleranz-Parameter: Int_Abs und Int_Rel: a und b, Int_UpDn: *keine* Parameter.	*dito*
Sel	5	*keine*	*dito*
Psd	2	Länge von 1- bzw. 0-Sequenzen: $2 \leq e \leq m-1$, d. h. $(m-2)$ Werte für e.	Besucht wegen e alle N Einträge $(m-2)$-mal, d. h. $\mathcal{O}(m \cdot N)$
Sim	2	Ähnlichkeitsschranke z: Sim_Bin: $0 < z_{\text{bin}} \leq m$, Sim_Num: $0 < z_{\text{num}} \leq M$, d. h. m bzw. M Werte für z.	Besucht einmalig alle $\frac{N}{2}$ *vergleichbaren* Einträge, d. h. Sim_Bin: $\mathcal{O}(m \cdot N)$ Sim_Num: $\mathcal{O}(M \cdot N)$
Hyp	1	Hyper-Terme h_j maximal $\frac{\ell \cdot (\ell-1)}{2}$ Hyper-Terme, ℓ: Anzahl der Terme	Anzuwenden für jeden Hyper-Term h_j, d. h. $\mathcal{O}(\ell^2)$.

Tabelle 5.15: Übersicht der Methoden mit Parametern und Komplexität

Neben den einzelnen Modi einer Methode, deren jeweilige Anzahl sich in der zweiten Spalte findet, tragen maßgeblich die Parameter zur Vielfalt bei. Sie sind für jede Methode in der dritten Spalte aufgeführt, sofern sie benötigt werden. Weiter sind in der letzten Spalte Angaben zum Aufwand, den die Auswertung einer Methode in Anspruch nimmt. Mit Hilfe des *Landauschen Symbols* $\mathcal{O}$ wird ein *Kriterium zur Bewertung der Effizienz* [Dud*88, S. 425] verwendet, um hier eine Betrachtung zur Komplexität der Methoden anzustellen.

Für eine parametrisierte Methode sind nicht alle Werte als Parameter sinnvoll; wie in der dritten Spalte angedeutet wird, existieren individuelle Einschränkungen, die den entsprechenden Abschnitten der einzelnen Methoden entnommen wurden. Mit dieser Information läßt sich bei den Methoden Psd, Sim und Hyp die Anzahl der möglichen Werte für den Parameter ermitteln, was mit in die Betrachtung der Komplexität für die Laufzeit eingeht.

In der Regel ergibt sich eine *Ordnung* von $\mathcal{O}(N)$, d. h. die Aufwendungen an Laufzeit wachsen proportional mit der Anzahl N der Einträge. Bei Psd und Sim_Bin mit $\mathcal{O}(m \cdot N)$ geht zusätzlich noch die Anzahl m der Ausgangsvariablen ein: Allerdings bleibt m in der Regel klein gegenüber N, da N durchaus um mehrere 10er-Potenzen größer sein kann als m. Anders verhält es sich bei Sim_Num mit der Komplexität $\mathcal{O}(M \cdot N)$; sie fällt um eine logarithmische Größenordnung höher aus als bei Sim_Bin mit $\mathcal{O}(m \cdot N)$, da gilt: $m := \mathsf{Bits}(M)$. Allerdings wird mit M hierbei die maximale Differenz zwischen *vergleichbaren* Einträgen angenommen, was in der Regel äußerst unwahrscheinlich ist. Für Hyp ergibt mit der maximalen Anzahl von $\frac{\ell \cdot (\ell-1)}{2}$ möglichen Hyper-Termen eine *worst-case* Ordnung von $\mathcal{O}(\ell^2)$.

5.3.3 Methoden als Instrumentarium

Die im vorhergehenden Abschnitt 5.3.2 (S. 100) diskutierte Vielfalt aller Methoden, die auf Parameter und Modi beruht, kann als ein *Instrumentarium* zur Modifikation von Spezifizierungen aufgefaßt werden. Allen Methoden ist gemeinsam, daß ihre Resultate im allgemeinen von einer *ad hoc* bzw. konventionell durchgeführten Spezifizierung abweichen. Werden diese modifizierten Spezifizierungen jedoch im Sinne der Diskussion über Unschärfe und Optionen aus Abschnitt 4.2 (S. 63) betrachtet, so tragen sie ebenso wie die *ad hoc* Spezifizierung eine ebenbürtige Berechtigung in sich, für den Entwurf eines PLAs verwendet zu werden.

Ob sich tatsächlich durch ihren Einsatz eine Reduzierung der PLA-Kosten signifikant einstellt, hängt nichtzuletzt vom Zulassen der prinzipbedingten Abweichungen ab. Intuitiv ist bereits zu vermuten, daß zum Erreichen einer großen Kosten-Reduzierung stark von der anfänglichen Spezifizierung abweichende Varianten erforderlich sind. Diese Thematik steht im nachfolgenden Kapitel 6 im Mittelpunkt der Diskussion: Es werden dort Optimierungsschritte aufgezeigt, die mit Hilfe der hier vorgestellten Methoden alternative Wahrheitstabelle erzeugen.

Kapitel 6

Optimierung und Evaluation

Mit den Methoden des vorhergehenden Kapitel 5 liegt ein Instrumentarium zur Modifikation von Spezifikationen vor. Da die einzelnen Methoden gemäß des *heuristischen Prinzips* angelegt sind (vgl. Fußnote 1, S. 75), stehen sie jeweils unter einer individuellen Arbeitshypothese. Diese Vorgehensweise dient zur Ausrichtung jeder Methode auf die allgemeine Zielsetzung im Gesamtkontext, nämlich die *Reduzierung der PLA-Kosten* (vgl. Kapitel 5, S. 75). Um mit dem Einsatz des Instrumentariums insgesamt einen *optimierenden* Effekt erzielen zu können, bedarf es neben dieser übergeordneten Zielsetzung einer unterstützenden Systematik, die im folgenden Abschnitt 6.1 vorgestellt wird. Danach werden im Abschnitt 6.3 (S. 118) exemplarisch Optimierungen vorgeführt, anhand derer eine Beurteilung der Effektivität von Methoden erfolgen kann.

6.1 Systematik zur Modifikation

Die im folgenden vorgestellte Systematik unterstützt den Ablauf der Optimierung dadurch, daß sie einen strukturellen Rahmen schafft, in dem PLA-Tabellen und Spezifikationen als Objekte betrachtet und zu sinnvollen Einheiten zusammengeführt werden. Diese Strukturierung trägt zur Übersichtlichkeit und zur Ausrichtung der Optimierung auf die übergeordnete Zielsetzung bei.

6.1.1 Strukturierung von PLA-relevanten Daten

Gemäß der in Abschnitt 3.2.1 (S. 41) dargestellten Ausgangssituation wird für eine Funktion eine realisierende PLA-Tabelle mit Hilfe einer Spezifikation erzeugt. Die Erzeugung selbst wurde daraufhin als *Spezifizierung* bezeichnet; sie ist für jede verschiedene PLA-Tabelle individuell. Eine PLA-Tabelle hängt damit sowohl von der Funktion als auch von der Spezifizierung ab.

6.1.1.1 Objekte bei der Spezifikation

Zum Zweck der Zusammenführung einer Spezifizierung und der daraus resultierenden PLA-Tabelle, sei eine Kapselung mit Hilfe eines einheitlichen Strukturelementes modelliert. Zunächst sind die an der Erzeugung einer PLA-Tabelle beteiligten Objekte zu betrachten:

1. Objekt: **Funktion**
   Im allgemeinen steht am Anfang der Spezifizierung die Festlegung der zu realisierenden Funktionalität. Dies geschieht durch die Definition einer Funktion f (vgl. Abschnitt 3.3.1, S. 43).

2. Objekt: **Spezifizierung**
   Die Spezifizierung ist eine Sequenz von Transformationsschritten, die z. B. aus den Abbildungen Dis, Scl, Con und Bin (vgl. Abschnitt 3.3, S. 43) besteht.

3. Objekt: **PLA-Tabelle**
   Das Resultat einer Spezifizierung ist die binär kodierte f-Variante $f_{\text{tab}} : \mathbb{B}^n \rightarrow \mathbb{B}^m$, die in Gestalt einer Wahrheitstabelle als Grundlage für den eigentlichen PLA-Entwurf dienen kann.

6.1.1.2 Merkmale einer PLA-Tabelle

Im Zusammenhang mit einer PLA-Tabelle sind des weiteren zwei Merkmale von Bedeutung, insbesondere dann, wenn Vergleiche zwischen mehreren PLA-Tabellen angestellt werden:

1. Merkmal: **Kosten**
   Zur Bestimmung der Kosten einer PLA-Tabelle kann die Termanzahl mit Hilfe eines Logik-Minimierers ermittelt werden.

2. Merkmal: **Abweichung**
   Aufgrund modifizierter Spezifizierungen können unterschiedliche PLA-Tabellen erzeugt werden. Durch Vergleiche zwischen jeweils zwei Tabellen sind Unterschiede feststellbar. Sofern eine der Tabellen als *korrekt* betrachtet wird, kann für die zweite die *Abweichung* ermittelt werden.

6.1.1.3 PLA-Frame als Kapselung der Objekte und Merkmale

Den genannten *Objekten* und *Merkmalen* sei nun ein struktureller Rahmen gegeben, der im folgenden als elementare Einheit aufgefaßt wird: **F** heißt *PLA-Frame* und sei wie folgt definiert:

$$
\begin{array}{rcl}
\mathbf{F} & := & (f, \mathbf{S}, p, e, d) \\
f & : & \text{zu realisierende \underline{F}unktionalität, z. B. } f_{\text{real}} \\
\mathbf{S} & : & \text{\underline{S}pezifizierung, ggf. modifiziert} \\
p & : & \text{binär kodierte \underline{P}LA-Tabelle, z. B. } f_{\text{tab}} \\
c & : & \text{Kosten (engl. \underline{c}osts) von } p\text{; i. a. Termanzahl } \ell \in \mathbb{N} \\
d & : & \text{Abweichung (engl. \underline{d}eviation) von } p\text{; mehrere Kriterien möglich}
\end{array}
\tag{6.1}
$$

Sowohl die Objekte einer Spezifikation aus Abschnitt 6.1.1.1 (S. 103) als auch die Merkmale einer PLA-Tabelle aus Abschnitt 6.1.1.2 (S. 104) werden im folgenden als *Komponenten* eines PLA-Frames betrachtet.

6.1.1.4 Beziehungen zwischen Komponenten

Zwischen den Komponenten eines PLA-Frames bestehen Beziehungen, die im folgenden zunächst aufgezeigt und anschließend untersucht werden:

- p, $\mathbf{S}$ und f:
  Da p aus f durch die Spezifizierung $\mathbf{S}$ hervorgeht, gilt formal: $p := \mathbf{S}(f)$.

- c und p:
  Zur Ermittlung der Kosten $c \in I\!N$ wird die Termanzahl ℓ von p ermittelt; dazu dient ein Logik-Minimierer, wie z. B. ESPRESSO [Bra*82b, Bra*84], der neben einer minimierten Termmenge auch deren Mächtigkeit, also die Anzahl der Terme, ermittelt.

- d und p:
  Die Abweichungen d_1 einer PLA-Tabelle p_1 im Vergleich zu einer anderen PLA-Tabelle p_2 können als Vektor $d_1 \in I\!R \times \cdots \times I\!R$ aufgefaßt werden, so daß für d_1 folgende funktionale Abhängigkeit besteht: $d_1 := f(p_1, p_2)$. Vereinfachend sei folgende Schreibweise zulässig: $d_1(p_2)$ oder auch einfach nur d_1, wenn die Bezugstabelle klar ist. Nähere Erläuterung zu diesem Merkmal, insbesondere zu sinvollen Abweichungskriterien wird auf Abschnitt 6.2.5.3 (S. 113) verwiesen.

Zusammenfassend kann festgestellt werden, daß die Komponenten p, c und d von f und $\mathbf{S}$ abhängige Komponenten sind; sie seien als *nicht-frei* bezeichnet. Dagegen können f und $\mathbf{S}$ als *unabhängige* Komponenten bzw. als *frei* aufgefaßt werden.

6.1.1.5 Individualisierung von PLA-Frames

Ausgehend von einer Funktion f können durch unterschiedliche Spezifizierungen viele verschiedene PLA-Tabellen erzeugt werden, die jeweils zu einem individuellen PLA-Frame führen. Zur Unterscheidung können PLA-Frames und Komponenten durch einen entsprechenden Index gekennzeichnet werden:

$$\mathbf{F}_j := (f, \mathbf{S}_j, p_j, c_j, d_j) \quad \text{mit} \quad j \in I\!N \tag{6.2}$$

6.1.2 Generierung von PLA-Frames

Mit Hilfe der *freien* Komponenten (vgl. Abschnitt 6.1.1.4, S. 105) von PLA-Frames, nämlich der Funktion f und der Spezifizierung $\mathbf{S}_j$, können individuelle PLA-Frames $\mathbf{F}_j := (f, \mathbf{S}_j, p_j, c_j, d_j)$ generiert werden. Da f jedoch für den Ablauf einer Optimierung als konstant anzusehen ist, bleibt lediglich die Spezifizierung $\mathbf{S}_j$ als die bestimmende Komponente übrig; für jede andersartig modifizierte Spezifizierung $\mathbf{S}_j$ entsteht ein neuer individueller PLA-Frame $\mathbf{F}_j$. Die weiteren Komponenten p_j, c_j und d_j hängen von $\mathbf{S}_j$ ab, wobei p_j *direkt* abhängig ist, während c_j und d_j *indirekt* von $\mathbf{S}_j$ abhängen, da sie nur über p_j zu bestimmen sind.

6.1.2.1 PLA-Frame als Referenz durch ad hoc-Spezifizierung

Für die heuristische Optimierung von PLA-Tabellen stehen die Modifikationen an der Spezifizierung einer PLA-Tabelle im Mittelpunkt des Interesses. Die Vorgehensweise des Modifizierens setzt zu Beginn die Existenz einer bereits spezifizierten PLA-Tabelle voraus. Zur Beurteilung der Optimierung sollte es sich um eine *ad hoc*-Spezifikation handeln, die im Sinne von Kapitel 3 konventionell abgelaufen ist; eine typische Demonstration dafür ist die Spezifizierung des *Sinus*-Beispiels in Abschnitt 3.1.5 (S. 39).

Dort geschah insbesondere die Spezifizierung ohne den Einsatz der in Kapitel 5 vorgestellten Methoden und sei deshalb als unmodifizierte Spezifizierung mit $\mathbf{S}_0$ bezeichnet. Da zur Beurteilung der Optimierung die zugehörige PLA-Tabelle p_0 zu Vergleichen herangezogen wird, sei dieser PLA-Frame $\mathbf{F}_0$ als *Referenz* bezeichnet:

$$\text{Referenz:} \quad \mathbf{F}_0 := (f, \mathbf{S}_0, p_0, c_0, d_0) \tag{6.3}$$

Die durch p_0 entstehenden Kosten c_0 werden zur Bestimmung der Kostenreduzierung zu den Kosten anderer PLA-Frames ins Verhältnis gesetzt. Da $\mathbf{F}_0$ im allgemeinen als Referenz dient, wird $d_0 := \emptyset$ angenommen.

6.1.2.2 PLA-Frames durch modifizierte Spezifizierungen

Durch den Einsatz der heuristischen Methoden aus Kapitel 5 (S. 75ff) kann eine Vielzahl von modifizierten Spezifizierungen entstehen. Dabei werden alle modifizierten Spezifizierungen, jeweils ausgehend von der Spezifizierung $\mathbf{S}_0$ der Referenz abgeleitet.

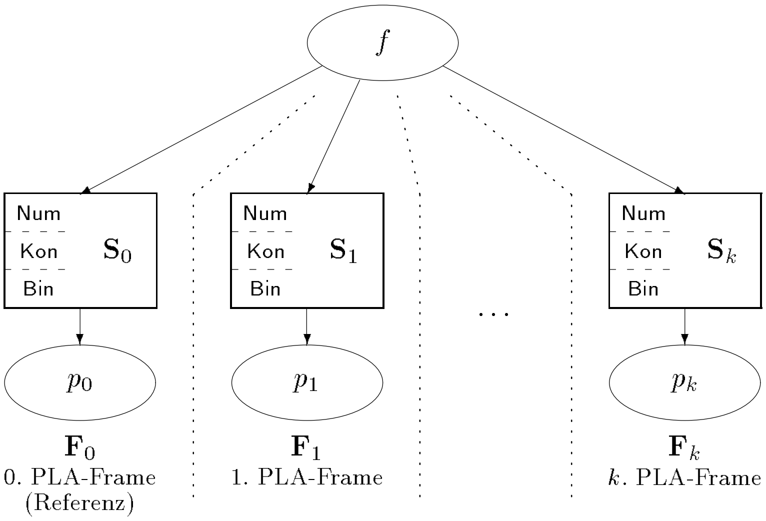

Abbildung 6.1: Mehrere PLA-Frames durch verschiedene Spezifizierungen

In Abbildung 6.1 (S. 106) ist neben der links dargestellten Referenz $\mathbf{F}_0$ die Situation für $k \in I\!N$ ($k \geq 1$) weitere PLA-Frames wiedergegeben: Sowohl die Referenz $\mathbf{F}_0$ wie auch die PLA-Frames $\mathbf{F}_j$ mit $1 \leq j \leq k$ gehen von derselben Funktion f aus. Jede Spezifizierung $\mathbf{S}_j$ stellt jeweils einen individuellen *Top-Down*-Ablauf dar, wie er in Kapitel 3 durch Abbildung 3.1 (S. 41) diskutiert ist. Zusätzlich ist hier die Unterteilung in die drei Schichten, nämlich die Numerik-, die Konversions- und die Binär-Schicht, angedeutet. Jede Spezifizierung $\mathbf{S}_j$ ist zumindest in *einer* der Schichten unterschiedlich modifiziert, so daß sich daraus im allgemeinen auch k unterschiedliche PLA-Tabellen p_j ergeben.

6.2 Optimierung mit PLA-Frames

Der Schwerpunkt der folgenden Diskussion liegt in der Ermittlung einer optimierten Spezifizierung, aus der eine PLA-Tabelle mit möglichst geringen PLA-Kosten hervorgeht. Zu jeder modifizierten Spezifizierung wird dabei der zugehörige PLA-Frame betrachtet: Anhand seiner Merkmal-Komponenten kann die vorgenommene Modifikation in ihren Konsequenzen beurteilt werden.

6.2.1 Begriffsbildung zum „Optimieren"

Um den Kontext für eine Optimierung näher zu bestimmen, folgt zunächst eine kurze Betrachtung zur Begriffsbildung: Unter dem Begriff *optimieren* wird nach [Dud*90, S. 553] das *Ermitteln der günstigsten Lösung für bestimmte Zielstellungen* verstanden. Dazu bedarf es zunächst der Definition einer Aufgabenstellung. Auf verschiedene Weise können Lösungen erzeugt werden, die, in einem übergeordneten Kontext betrachtet, ungünstig in Erscheinung treten. Durch die Festlegung einer bestimmten Zielstellung kann eine Optimierungsprozedur Lösungen hervorbringen, die in diesem Sinne günstig sind.

6.2.2 Problemfelder bei der Optimierung

Auf den Kontext dieser Arbeit übertragen, findet sich in der *Minimierung der PLA-Kosten* die Bestimmung der Zielstellung. Es lassen sich hier drei Problemfelder erkennen, nämlich zunächst die *Aufgabenstellung*, des weiteren die *Lösungserzeugung* und schließlich die *Ergebnisbeurteilung*. Im einzelnen stellen sich die Problemfelder wie folgt dar:

1. *Aufgabenstellung*
   Definition der Funktionalität f und Erstellung eines PLA-Frames $\mathbf{F}_0$ als Referenz. Die Spezifizierung $\mathbf{S}_0$ geschieht *ad hoc* bzw. in konventioneller Weise. Näheres in Abschnitt 6.2.3 (S. 108).

2. *Lösungserzeugung*
   Mit dem in Kapitel 5 vorgestellten Instrumentarium der heuristischen Methoden wird die Spezifizierung $\mathbf{S}_0$ in einer vielfältigen Weise modifiziert, d. h. es wird eine möglichst große Anzahl weiterer PLA-Frames $\mathbf{F}_j$, $j \in I\!N$ mit $j \geq 1$ erzeugt. Näheres dazu in Abschnitt 6.2.4 (S. 108).

3. *Ergebnisbeurteilung*
   Anhand der beiden Merkmal-Komponenten c_j und d_j der erzeugten PLA-Frames $\mathbf{F}_j$ können diese beurteilt werden. Im Sinne des Optimierungszieles wird ein PLA-Frame als *günstigstes* Ergebnis ausgewählt. Näheres in Abschnitt 6.2.5 (S. 113).

Während sich die *Aufgabenstellung* im Verhältnis zu den anderen beiden Problemfeldern bereits relativ konkret abzeichnet, besteht insbesondere bei den beiden zuletzt genannten der Bedarf nach einer verfeinerten Diskussion. In den folgenden Abschnitten sollen daher die drei Problemfelder näher erläutert werden.

6.2.3 Problemfeld: Aufgabenstellung

Neben der definierten Funktionalität in Gestalt der Funktion f mit ihrem Definitionsbereich liegt zumeist eine *ad hoc* spezifizierte PLA-Tabelle als Ausgangsmaterial für eine Optimierung vor. Nachdem die Kosten der PLA-Tabelle durch ihre Termanzahl ermittelt sind, kann ein PLA-Frame $\mathbf{F}_0 := (f, \mathbf{S}_0, p_0, c_0, \emptyset)$ gebildet werden, der hier als Referenz aufgefaßt werden soll.

Auch die so spezifizierte PLA-Tabelle p_0 enthält aufgrund der Untersuchungen in Abschnitt 4.2 (S. 63ff) Unschärfe. Um bei der abschließenden *Ergebnisbeurteilung* die erzeugten PLA-Frames quantitativ einordnen zu können, sollte aus dem Kontext der Aufgabenstellung heraus abgeklärt werden, in welcher Größenordnung eine generelle Toleranz zugelassen werden kann. Zu empfehlen ist hierbei, daß Auftraggeber und Entwerfer (vgl. Abschnitt 3.2.1, S. 41) *gemeinsam* versuchen, die bereits inhärente und nicht vermeidbare Unschärfe abzuschätzen.

Auf diese Weise lassen sich sinnvolle Schranken festlegen, die nicht nur zur Beurteilung von PLA-Frames benötigt werden, sondern auch bei der Erzeugung von PLA-Frames mit Hilfe der kombinierten Methoden Int (Generierung von Intervallen, vgl. Abschnitt 5.1.3, S. 79) und Sel (Selektion von Werten anhand von Intervallen, vgl. Abschnitt 5.1.4, S. 83). Innerhalb der Aufgabenstellung sind somit folgende Aspekte relevant:

1. Definition der Funktion f mit ihrem Definitionsbereich.
2. Erstellung der Referenz $\mathbf{F}_0 := (f, \mathbf{S}_0, p_0, c_0, \emptyset)$.
3. Abschätzung der zulässigen Toleranz für die Funktionswerte von f.

6.2.4 Problemfeld: Lösungserzeugung

Generell geschieht das Erzeugen von verschiedenen Lösungen mit PLA-Frames durch Variieren der *freien* Komponenten f und $\mathbf{S}$ (vgl. Abschnitt 6.1.1.4, S. 105). Da f jedoch im Rahmen einer Optimierung fest gegeben ist, verbleibt allein $\mathbf{S}$ als bestimmende Komponente. In der Regel gehen Modifikationen an Spezifizierungen von der Referenz $\mathbf{F}_0 := (f, \mathbf{S}_0, p_0, c_0, d_0)$ aus.

6.2.4.1 Vielfalt bei Methoden durch Modi und Parameter

Bezüglich der Erzeugung von modifizierten PLA-Frames wird bereits in Abschnitt 6.2.2 (S. 107) bei den einzelnen Methoden von vielfältigen Möglichkeiten für Modifikationen ausgegangen (vgl. Abschnitt 5.3.2, S. 100). Diese beruhen auf den möglichen Modi und Parametern einer Methode:

- **Modi:** Sofern eine Methode mehrere Modi aufweist, kann mit jedem von ihnen eine andersartige Modifikation an einer Spezifizierung vorgenommen werden. Daher entspricht die Anzahl der erzielbaren Modifikationen der jeweiligen Anzahl der Modi.

- **Parameter:** Im allgemeinen erzeugen Methoden, die durch Parameter in ihrer Arbeitsweise gesteuert[1] werden können, für jede Variation der Parameter eine andersartig modifizierte Spezifizierung. Somit sind die möglichen Modifikationen durch die maximale Anzahl der für einen Parameter sinnvollen Werte bestimmt; sie läßt sich häufig nur abschätzen.

In Tabelle 5.15 (S. 101) sind für jede Methode die Anzahl der vorhandenen Modi und die maximale Anzahl der Werte für Parameter zusammengefaßt. In diesem Sinne sei im folgenden der Begriff *Vielfalt* für die Anzahl der verschiedenartigen Spezifizierungen gebraucht und durch $V \in I\!N$ bezeichnet, d. h. in einem konkreten Fall gibt V die Anzahl der verschiedenen Spezifizierungen $\mathbf{F}_0, \ldots, \mathbf{F}_{V-1}$. Beim Kombinieren von Methoden bzw. Modifikationen existieren bezüglich der Vielfalt prinzipiell zwei Möglichkeiten:

$V_{\min}$ **Einzelne Modifikation:**
Eine minimale Vielfalt ergibt sich, falls pro Spezifizierung nur *eine* einzelne Modifikation zugelassen wird. Ansonsten wird wie bei der Referenz-Spezifizierung $\mathbf{S}_0$ verfahren.

$V_{\max}$ **Multiple Modifikationen:**
Werden hingegen bei einer Spezifizierung mehrere Modifikationen gleichzeitig vorgenommen, so ergibt sich daraus eine maximale Vielfalt.

Anhand der drei Schichten wird nachfolgend die jeweils mögliche Vielfalt bei jeder einzelnen Methode diskutiert, wobei $V \in I\!N$ durch einen entsprechenden Indexvermerk an eine Methode oder eine Schicht gebunden wird, wie z. B. in $V_{\mathsf{Rnd}} \in I\!N$. In Abbildung 6.2 (S. 111) ist die Situation bei *Einzelner Modifikation* für eine minimale Vielfalt dargestellt. Durch stärkere Linien ist der Verlauf der Spezifizierung $\mathbf{S}_0$ hervorgehoben: Ausgehend von f verläuft sie über $\mathsf{Dis}(N_0)$ und $\mathsf{Scl}(M_0)$ zu **Rnd_Round**, wo sie nach einer Kodierung durch $\mathsf{Bin}(m)$ als PLA-Tabelle p_0 endet. Zur Klammer-Schreibweise der Parameter bei den Methoden siehe Fußnote 1 (S. 109).

6.2.4.2 Vielfalt in den Schichten durch Modifikationen

In den drei Schichten machen sich Modifikationen in Bezug auf die Vielfalt, also die Anzahl der verschiedenartigen Spezifizierungen, unterschiedlich bemerkbar. In Tabelle 6.1 (S. 110) wird die Abschätzung der bei *Einzelner Modifikation* erzielbaren Vielfalt entwickelt.

Dies geschieht schrittweise, und zwar beginnend im mittleren Block von Tabelle 6.1 mit der Abschätzung der Vielfalt für die einzelnen Methoden. In der nächsten Zeile unterhalb des mittleren Blockes werden die Abschätzungen für die Schichten gebildet, um schließlich in der untersten Zeile die Gesamt-Vielfalt abzuschätzen. Nachfolgend wird anhand der Schichten die Abschätzung der Vielfalt näher erläutert:

- **Modifikationen in der Numerik-Schicht:**
Bereits in Kapitel 5 (S. 75) wird mit Bezug auf Kernaussage 3 aus Abschnitt 4.2.5 (S. 74)

[1] Obwohl bei den Methoden bzw. Transformationsabbildungen bisher mit der Schreibweise $\mathsf{Met}(a)$ einem Argument a durch eine Methode Met ein Ergebniswert zugeordnet wurde, wobei es sich bei a um einen Indexwert oder Eintrag der Wahrheitstabelle handelte, tritt diese Sichtweise hier in den Hintergrund. Eine Methode wird immer auf die gesamte Wahrheitstabelle angewendet, d. h. über alle Argumente iteriert. Entscheidender sind dagegen nun die Parameter einer Methode, die bisher in Index-Schreibweise notiert wurden. Für eine Methode Met mit dem Parameter p sei künftig statt $\mathsf{Met}_p(a)$ auch die Schreibweise $\mathsf{Met}(p)$ ohne die obligatorischen Argumente a zulässig.

<table>
<tr><th colspan="9">Vielfalt der Methoden bei „Einzelner Modifikation“</th></tr>
<tr><td colspan="2">Numerik-</td><td colspan="4">Konversion-</td><td colspan="3">Binär-Schicht</td></tr>
<tr><td>V_{Dis}</td><td>V_{Scl}</td><td>V_{Rnd}</td><td>V_{P2}</td><td>V_{Int}</td><td>V_{Sel}</td><td>V_{Psd}</td><td>V_{Sim}</td><td>V_{Hyp}</td></tr>
<tr><td rowspan="2">≥ 1</td><td rowspan="2">≥ 1</td><td rowspan="2">3</td><td rowspan="2">3</td><td>≥ 1</td><td>5</td><td>≥ 2</td><td rowspan="2">≥ 2</td><td>≥ 0</td></tr>
<tr><td colspan="2">$\geq 5 \ddagger^1$</td><td>$\leq 2 \cdot (m-2)$</td><td>$\leq \frac{\ell \cdot (\ell-1)}{2}$</td></tr>
<tr><td colspan="2">$V_{\mathrm{num}} \geq 1 \ddagger^2$</td><td colspan="4">$V_{\mathrm{kon}} \geq 11 \ddagger^3$</td><td colspan="3">$V_{\mathrm{bin}} \geq 4 \ddagger^4$</td></tr>
<tr><td colspan="9">$V_{\mathrm{min}} \geq 15 \ddagger^5$</td></tr>
</table>

$\ddagger^1$: $V_{\mathrm{Int}} \cdot V_{\mathrm{Sel}}$
$\ddagger^2$: $V_{\mathrm{Dis}} + V_{\mathrm{Scl}} - 1$
$\ddagger^3$: $V_{\mathrm{Rnd}} + V_{\mathrm{P2}} + V_{\mathrm{Int}} \cdot V_{\mathrm{Sel}}$
$\ddagger^4$: $V_{\mathrm{Psd}} + V_{\mathrm{Sim}} + V_{\mathrm{Hyp}}$
$\ddagger^5$: $V_{\mathrm{num}} + V_{\mathrm{kon}} - 1 + V_{\mathrm{bin}}$ (inklusive Referenz)

Tabelle 6.1: Übersicht zur Vielfalt in den Schichten

auf die Besonderheit der Modifikationen innerhalb der Numerik-Schicht eingegangen, da sie nicht durch einzelne Methoden strukturiert betrachtet werden kann. Möglichkeiten zur Modifikation einer Spezifizierung in dieser Schicht werden durch die Optionen O-1 bis O-5 (vgl. Tabelle 4.5, S. 74) empfohlen. Bezüglich der beiden charakteristischen Merkmale bzw. Parameter N und M bestehen folgende Optionen:

- **Modifikation der Anzahl N der Einträge:**
  Die Optionen O-1: *Anzahl der Stützstellen als 2er-Potenz*, O-2: *Redundanz aus der Stützstellenfolge entfernen* und O-3: *Veränderlicher Abstand der Stützstellen* dienen zur Anpassung der Anzahl $N \in \mathbb{N}$ der Einträge in der PLA-Tabelle. N nimmt als charakteristisches Merkmal bei der Diskretisierung Einfluß auf die Abbildung $\mathsf{Dis}(N) : \mathbb{R} \to \mathbf{I}_N \subseteq \mathbb{N}$ (vgl. Abschnitt 3.3.2.3, S. 46).
- **Modifikation des Maximums M der Einträge:**
  Durch die Optionen O-4: *Bestimmung der Skalierungsabbildung* und O-5: *Nichtlineare Skalierung* kann das Maximum M der Einträge einer PLA-Tabelle beeinflußt werden. Durch M als das charakteristische Merkmal bei der Skalierung wird $\mathsf{Scl}(M) : \mathbb{R} \to \mathbb{R}^+$ gesteuert (vgl. Abschnitt 3.3.3.2, S. 48).

Die Vielfalt bezüglich N bzw. M, die durch die Transformationsabbildungen Dis bzw. Scl festgelegt wird, sei mit V_{Dis} bzw. V_{Scl} bezeichnet. Aufgrund der vielen Variationsmöglichkeiten beim Festlegen von N und M sind V_{Dis} und V_{Scl} nur nach unten durch 1 beschränkt. Für die Gesamtvielfalt der Numerik-Schicht V_{num} (vgl. vorletzte Zeile in Tabelle 6.1, S. 110) gilt bei *Einzelner Modifikation* der Ausdruck von $\ddagger^2$: Summe der einzelnen Vielfalt von Dis und Scl, abzüglich einer darin doppelt gezählten Kombination, denn von beiden Abbildungen wird für jede Spezifizierung jeweils eine Variante benötigt.

- **Modifikationen in der Konversions-Schicht:**
  Zunächst ergeben sich für die Methode Rnd und P2 jeweils drei Modifikationen, d. h. $V_{\mathsf{Rnd}} := 3$ und $V_{\mathsf{P2}} := 3$ (vgl. Tabelle 6.1, S. 110). Die Methoden Int und Sel werden kombiniert eingesetzt, wobei auf jeden Intervallmodus jeweils bis zu fünf Selektionsmodi folgen können, d. h. $V_{\mathsf{Sel}} := 5$. Die beiden Intervall-Modi Int_Abs und Int_Rel benötigen Parameter. Die Wahl des Int-Modus und die Festlegung der ggf. erforderlichen Werte für die Parameter bestimmen die durch Intervalle repräsentierte Toleranz bei den Einträgen.

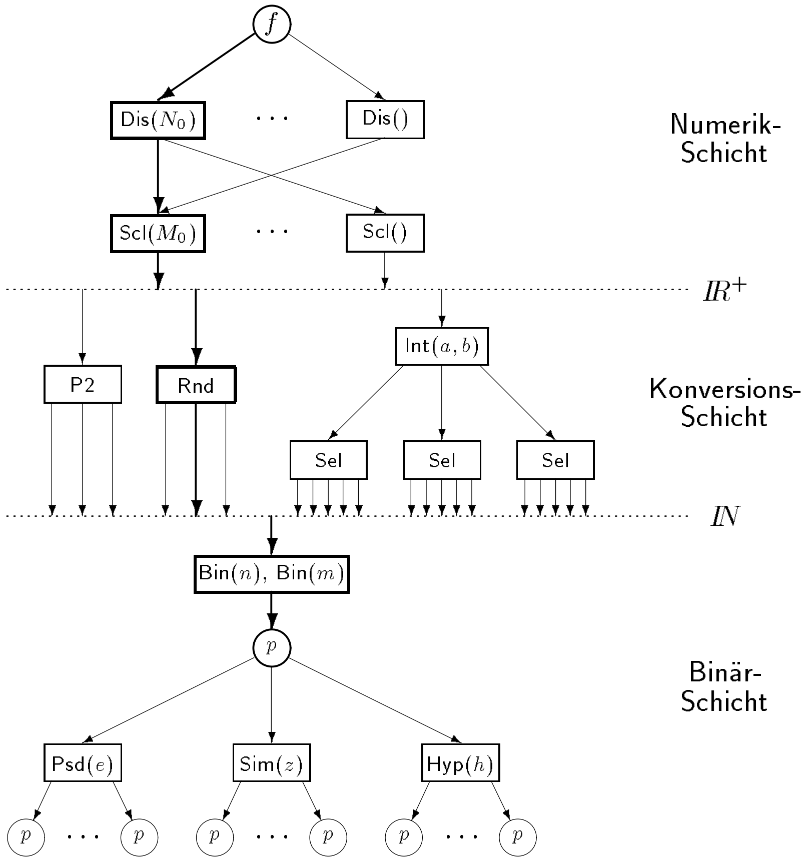

Abbildung 6.2: Überblick zum Einsatz aller Methoden

Wie bereits in Abschnitt 6.2.3 (S. 108) erwähnt, fällt die Bestimmung der durch Unschärfe bedingten Toleranz in den Kontext des Problemfeldes *Aufgabenstellung*. Obwohl sich nach Einschätzung der Unschärfe theoretisch[2] nur *ein* Int-Modus ergeben kann, sollten weitere Toleranzangaben durch hinzugenommene Int-Modi realisiert werden.

Mit $V_{\mathsf{Int}} \geq 1$ ergibt sich in der Konversions-Schicht bei *Einzelner Modifikation* insgesamt eine Vielfalt von $V_{\mathsf{kon}} \geq 11$; siehe Ausdrücke bei $\ddagger^1$ und $\ddagger^3$ (vgl. Tabelle 6.1, S. 110). Da Konversions-Methoden, wie die allgemeine Konversionsabbildung $\mathsf{Con} : I\!R^+ \rightarrow I\!N$, verschiedene Argumenten- und Funktionswertebereiche haben, können sie auch bei *Multipler Modifikation* für jede zugeführte Menge aus $I\!R^+$-Funktionswerte nur in der diskutierten Vielfalt angewendet werden. Durch die jeweilige Binärkodierung entstehen auf diese Weise mindestens 11 PLA-Tabellen, die im allgemeinen auch verschieden sind.

[2]Bei mehrfachen Toleranzangaben beinhaltet die am weitesten reichende die kleineren. Mehrfache Angaben sind daher höchstens zu Testzwecken oder bei entsprechender Unsicherheit über die Einschätzungen sinnvoll, beispielsweise auch dann, wenn Toleranzen in einer nicht-symmetrischen Lage um die Funktionswerte angegeben werden sollen.

Im konkreten Fall können jedoch aus den verschiedenen Spezifizierungen mehrere identische PLA-Tabellen[3] resultieren.

- **Modifikationen in der Binär-Schicht:**
Während die Psd- und Sim-Methode jeweils zwei Modi besitzen, hat die Hyp-Methode nur einen Modus. In Tabelle 5.15 (S. 101) ist die Anzahl der Parameterwerte für die einzelnen Binär-Methoden aufgeführt, die sich an der Anzahl $m \in I\!N$ der Logikfunktionen bzw. Tabellenspalten, weiter an das bei der Skalierung festgelegte Maximum $M \in I\!N$ der Einträge sowie an die durch die Logik-Minimierung ermittelte Termanzahl $\ell \in I\!N$ orientieren. Zusammenfassend stellt sich die Situation wie folgt dar:

  - **Psd-Methode:** Für beide Modi Psd_Inc und Psd_Dec ergibt sich für den Parameter c jeweils eine Werteanzahl von minimal 1 und maximal $(m-2)$, d. h. es gilt für die gesamte Methode $2 \leq V_{\mathsf{Psd}} \leq 2 \cdot (m-2)$.
  - **Sim-Methode:** Die beiden Modi Sim_Bin und Sim_Num interpretieren den Parameter z für die zulässige Ähnlichkeit unterschiedlich. Während sich bei Sim_Bin für den Parameter z_{bin} eine Werteanzahl von minimal 1 und maximal m ergibt, ist sie für Sim_Num minimal ebenfalls 1 und nach oben nur durch das Maximum M aller Funktionswerte beschränkt. Die Vielfalt der Sim-Methode ist daher $V_{\mathsf{Sim}} \geq 2$.
  - **Hyp-Methode:** Die Hyp-Methode arbeitet mit Hyper-Termen, die durch eine zusätzliche Anwendung der *Consensus Methode* auf die minimierte Termmenge erzeugt werden. Im ungünstigsten Fall existieren *keine* Hyper-Terme, so daß die Methode dann auch *keine* Modifikationen vornehmen kann. Für die maximale Anzahl der möglichen Hyper-Terme h kann nur eine Abschätzung nach oben erfolgen. Aus Tabelle 5.15 (S. 101) ist dafür eine maximale Anzahl von $\frac{\ell \cdot (\ell-1)}{2}$ Hyper-Termen zu entnehmen. Damit ergibt sich für die Hyp-Methode eine Vielfalt in folgendem Bereich: $0 \leq V_{\mathsf{Hyp}} \leq \frac{\ell \cdot (\ell-1)}{2}$.

  Für die Binär-Schicht ist die Vielfalt damit bei *Einzelner Modifikation* $V_{\text{bin}} \geq 4$. Sie ergibt sich durch Aufsummieren der Vielfalt bei den einzelnen Binär-Methoden; siehe auch den Ausdruck bei ‡[4] (vgl. Tabelle 6.1, S. 110).

Abschließend kann in der untersten Zeile von Tabelle 6.1 (S. 110) festgestellt werden, daß bei *Einzelner Modifikation* sich unter Einbeziehung aller Schichten eine minimale Gesamt-Vielfalt V_{min} durch Addieren der minimalen Vielfalt von Numerik-, Konversions- und Binär-Schicht ergibt. Allerdings ist dabei eine Spezifizierung bei der Numerik- und der Konversions-Schicht doppelt berücksichtigt, so daß jene zu subtrahieren ist; siehe den Ausdruck bei ‡[5], Tabelle 6.1 (S. 110): Aus $V_{\text{num}} \geq 1$, $V_{\text{kon}} \geq 11$ und $V_{\text{bin}} \geq 4$ ergibt sich somit $V_{\text{min}} \geq 15$.

Die Gesamt-Vielfalt V ist, für *Multiple Modifikationen* (vgl. Abschnitt 6.2.4.1, S. 109) nach oben unbeschränkt, d. h. $V_{\text{max}} \to \infty$. Im allgemeinen entstehen durch eine Vielfalt $V \in I\!N$ von verschiedenartigen Modifikationen $K \in I\!N$ modifizierte Spezifizierungen $\mathbf{S}_j$, wobei gilt: $1 \leq j \leq K \leq V$; selten wird $K = V$ gelten, da trotz verschiedenartiger Modifikationen identische Spezifikationen resultieren können (vgl. Fußnote 3, S. 112).

[3] Beispielsweise wird in einer Spezifizierung $\mathbf{S}_i$ mit dem Round-Modus konvertiert, während in einer weiteren Spezifizierung $\mathbf{S}_j$, $i \neq j$ stattdessen der Floor-Modus eingesetzt wird. Werden der Konversions-Schicht nun zufälligerweise solche reellen Werte zugeführt, bei denen in $\mathbf{S}_i$ der Round-Modus immer nur abrundet, so entsteht dadurch als Ergebnis dieselbe PLA-Tabelle, wie bei $\mathbf{S}_j$ durch den Floor-Modus.

6.2.5 Problemfeld: Ergebnisbeurteilung

Zum Abschluß einer Optimierung ist die Beurteilung der erzeugten Lösungen[4] notwendig, die durch verschiedenartig modifizierte PLA-Frames repräsentiert werden. Vergleiche zwischen PLA-Frames geschehen anhand ihrer Merkmal-Komponenten *Kosten* c und *Abweichungen* d.

6.2.5.1 Beurteilung von PLA-Frames durch Kostenvergleich

Beim Vergleichen zweier PLA-Frames wird grundsätzlich derjenige als *PLA-gerechter* betrachtet, der die geringsten *Kosten* aufweist. Neben dem Kostenvergleich zwischen beliebigen PLA-Frames ist der Vergleich zu den Kosten der Referenz $\mathbf{F}_0$ von besonderer Bedeutung, da sich in der Gegenüberstellung mit den Kosten der *ad hoc* spezifizierten und einer modifiziert spezifizierten PLA-Tabelle die jeweils erzielte Kosten-Reduzierung ergibt.

6.2.5.2 Beurteilung von PLA-Frames durch Abweichungsvergleich

Allerdings können unter den Lösungen durchaus PLA-Frames $\mathbf{F}_i$ und $\mathbf{F}_j$ mit $i \neq j$ auftreten, die unterschiedliche PLA-Tabelle $p_i \neq p_j$ besitzen, jedoch gleiche Kosten $c_i = c_j$ aufweisen. In solchen Fällen ist es notwendig, das zweite Merkmal *Abweichungen* zu betrachten: Jeder PLA-Frame ist verschiedenartig modifiziert und enthält dementsprechend ein individuelles Maß an Unschärfe. Sie läßt sich in ihren Ausmaßen nur durch Vergleiche zwischen PLA-Frames beurteilen. Da prinzipiell keine exakte Lösung unter den PLA-Frames existieren kann (vgl. Abschnitt 3.1.4.1, S. 36), muß ein PLA-Frame ausgewählt werden, von dem angenommen wird, daß in dessen PLA-Tabelle nur in einem geringen Maße Unschärfe enthalten ist. Oft bietet sich hierfür wiederum die *ad hoc* spezifizierte PLA-Tabelle[5] p_0 der Referenz $\mathbf{F}_0$ an.

6.2.5.3 Kriterien zur Bestimmung der Abweichungen

In Abschnitt 6.1.1.4 (S. 105) wurde bereits zur Bezeichnung der Abweichungen zwischen zwei PLA-Frames $\mathbf{F}_i$ und $\mathbf{F}_j$ die Schreibweise für den Abweichungsvektor $d_i(p_j)$ eingeführt. Es handelt sich dabei um den Vergleich der PLA-Tabelle p_i bezogen auf p_j, der jeweils Einträge mit gleichem Index $k \in \mathbf{I}_N$ betrachtet. Dazu können binäre und numerische Kriterien angewendet werden.

- **Binäre Kriterien:**
  Die beiden Einträge der beiden Tabellen p_i und p_j seien $r := p_i(k)$ und $t := p_j(k)$. Sie werden jeweils bitweise miteinander verglichen; dies kann nicht durch den *Hamming-Abstand* (vgl. Abschnitt 5.2.2.1, S. 93) geschehen, da er nicht für $\mathbb{D}$, sondern nur für $\mathbb{B}$ definiert

[4] Ein Vergleich mit dem Optimum ist hier nicht möglich, da die Erzeugung der Lösungen entscheidend von der individuellen Einschätzung der Unschärfe abhängt. Die dadurch bedingten Freiheiten verhindern, daß sich unter den möglichen Lösungen *eine* als Optimum gegenüber allen anderen erweisen kann.

[5] Entsprechend der in Abschnitt 3.2.1 (S. 41) geschilderten Ausgangssituation aus der Praxis, in der sich Auftraggeber und Entwerfer gegenüber stehen, wird in den meisten Fällen vonseiten des Auftraggebers ohnehin die von ihm selbst erstellte PLA-Tabelle für „das Maß aller Dinge“ gehalten; sie ist in der Regel durch eine *ad hoc*-Spezifikation generiert worden. Falls der daraus zusammenstellbare PLA-Frame als Referenz verwendet wird, sind Reduzierungen der Kosten unmittelbar ersichtlich. Dieses fördert die Akzeptanz der Optimierungsbemühungen, bei denen vonseiten des Auftraggebers ja auch Abweichungen zu akzeptieren sind.

ist. In den Einträgen der Wahrheitstabelle können jedoch $*$-Symbole auftreten, so daß eine andere Definition notwendig ist:

Anzahl abweichender Bitstellen in Einträgen
Die Anzahl der voneinander abweichenden bzw. *deviant* Bitstellen in zwei Einträgen mit gleichem Index $k \in \mathbf{I}_N$ sei mit $\mathrm{B_{dev}}(k)$ bezeichnet und für die Einträge $r_{m-1} \ldots r_0$ und $t_{m-1} \ldots t_0$ wie folgt definiert:

$$\mathrm{B_{dev}}(k) := \Big|\{b \mid r_b \neq t_b,\ m > b \geq 0\}\Big| \quad \text{mit} \quad r_b, t_b \in I\!D \tag{6.4}$$

Diese Definition entspricht weitestgehend der Funktion $\mathsf{Dist}(\mathbf{p}, \mathbf{q})$ (vgl. Abschnitt 1.2.4.2, S. 12).

Für den Abweichungsvektor d erweisen sich Aussagen über *alle* Einträge der beiden PLA-Tabellen p_i und p_j als zweckmäßig; beispielsweise seien die beiden folgenden Kriterien definiert:

- Maximum der unterschiedlichen Bitstellen bei allen Einträgen (engl. *entries*). Es sei als $\mathrm{B_{ent}}$ bezeichnet und verwendet über alle Indizes $k \in \mathbf{I}_N$ die Anzahl der unterschiedlichen Bitstellen $\mathrm{B_{dev}}(k)$ (vgl. Gleichung 6.4, S. 114):

$$\mathrm{B_{ent}} := \max\left(\left\{\mathrm{B_{dev}}(k) \;\Big|\; k \in \mathbf{I}_N\right\}\right) \tag{6.5}$$

- Anteil der unterschiedlichen Bitstellen $\mathrm{B_{devt}}(k)$ für alle $k \in \mathbf{I}_N$ bezogen auf die Anzahl N der Einträge der Tabellen in Prozent:

$$\mathrm{B_{tab}} := \frac{1}{N} \cdot \left(\sum_{k=0}^{N-1} \mathrm{B_{dev}}(k)\right) \cdot 100\% \tag{6.6}$$

Während sich durch $\mathrm{B_{tab}}$ generell das Ausmaß der binär interpretierbaren Veränderungen in der Wahrheitstabelle ermessen läßt, gibt $\mathrm{B_{ent}}$ die maximal in den Einträgen vorgenommene Bit-Veränderung an, so daß hierdurch die Wirkung von Binär-Methoden beobachtet werden kann; insbesondere trifft dies auf die Methode $\mathsf{Sim\_Bin}$ zu, bei der $\mathrm{B_{ent}}$ in der Regel mit dem Parameter z_{bin} der Methode einhergeht (vgl. Tabelle 6.5, S. 122).

- **Numerische Kriterien:**
Aufgrund des numerischen Hintergrundes der Daten im Kontext dieser Arbeit gilt im allgemeinen den numerischen Abweichungskriterien ein größeres Interesse als den binären. Analog zu den Gleichungen in Abschnitt 4.2.3.2 (S. 71) zum Vergleich eines reellen Wertes mit einem abgeleiteten Näherungswert, können hier für den Vergleich der Einträge u und t zweier PLA-Tabellen ähnlich definierte Kriterien zum Einsatz kommen:

  - Maximum der Absolut-Differenzen:

$$\mathrm{M_{abs}} := \max\left(\left\{|u_k - t_k| \;\Big|\; k \in \mathbf{I}_N\right\}\right) \tag{6.7}$$

  - Maximum der Relativ-Differenzen in Prozent:

$$\mathrm{M_{rel}} := \max\left(\left\{\frac{|u_k - t_k|}{u_k} \;\Big|\; k \in \mathbf{I}_N\right\}\right) \cdot 100\% \tag{6.8}$$

- Mittelwert über die summierten Absolut-Differenzen:

$$\mathrm{A_{abs}} := \frac{1}{N} \cdot \sum_{k=0}^{N-1} |u_k - t_k| \tag{6.9}$$

- Mittelwert über die summierten Differenz-Quadrate:

$$\mathrm{A_{sqr}} := \frac{1}{N} \cdot \sqrt{\sum_{k=0}^{N-1} (u_k - t_k)^2} \tag{6.10}$$

- Mittelwert über die summierten Relativ-Differenzen in Prozent:

$$\mathrm{A_{rel}} := \left(\frac{1}{N} \cdot \sum_{k=0}^{N-1} \frac{|u_k - t_k|}{u_k} \right) \cdot 100\% \tag{6.11}$$

Je nach dem Charakter der Daten aus der Aufgabenstellung ist die Auswertung nicht für alle Kriterien sinnvoll bzw. erforderlich. Werden beispielsweise aus der Aufgabenstellung, bei der Einschätzung der anzunehmenden Unschärfe, nur *absolut* bemessene Toleranzbereiche angegeben, so sind vor allem auch nur *absolut* ermittelte Abweichungen relevant, weniger dagegen die *relativ* bezogenen. Ebenso sind wegen des *numerischen* Kontexts der Aufgabenstellung die *binären* Kriterien in der Regel weniger relevant.

Aus diesen Gründen ist bei der Definition des PLA-Frames (vgl. Abschnitt 6.1.1.3, S. 104) das Abweichungs-Merkmal nur allgemein als Abweichungsvektor $d \in I\!R \times \cdots \times I\!R$ angegeben. Aus dem Kontext der Problemstellung ergibt sich die Relevanz der einzelnen Kriterien, so daß sich die Dimension des Abweichungsvektors je nach der Anzahl der auszuwertenden Kriterien ergibt.

Um ein Kriterium, wie z. B. $\mathrm{M_{abs}}$, aus einem Abweichungsvektor d_j separat zu betrachten, sei als Bezeichnung folgende Schreibweise zulässig: $\mathrm{M_{abs}}(d_j)$ oder einfach nur $\mathrm{M_{abs}}$, wenn offensichtlich ist, um welche PLA-Tabelle p_j es sich handelt.

In der Regel gilt unter allen genannten Kriterien dem numerischen Paar $\mathrm{M_{abs}}$ und $\mathrm{A_{rel}}$ das größte Interesse: Intuitiv ist die größte Absolut-Abweichung in der gesamten Wahrheitstabelle eine wichtige Kenngröße. Solange sie „klein" ausfällt, können auch für die anderen Kriterien geringe Werte angenommen werden. Erreicht $\mathrm{M_{abs}}$ allerdings größere Werte, so stellt sich die Frage, wie Abweichungen relativiert bei den einzelnen Einträgen auftreten. Das Kriterium $\mathrm{M_{rel}}$, welches die Relativ-Abweichungen über alle Einträge maximiert, ist nicht sehr aussagekräftig, da bei Einträgen mit „kleinen" Werten eine Absolut-Abweichung von 1 bereits zu einem hohen Wert bei der Relativ-Abweichung führt. Dadurch wird der Wert von $\mathrm{M_{rel}}$ leicht dominiert. Als Relativ-Kriterium ist daher der Durchschnitt $\mathrm{A_{rel}}$ über die Relativ-Abweichung aller Einträge vorzuziehen. Unter den binären Kriterien gilt $\mathrm{B_{ent}}$ die größte Aufmerksamkeit, da sich damit die Wirkung der binären Methoden leicht überprüfen lassen.

6.2.6 Strategien zur Optimierung

Anschließend an die *Aufgabenstellung* (vgl. Abschnitt 6.2.3, S. 108) werden PLA-Frames im nächsten Schritt der *Lösungserzeugung* dadurch erzeugt, daß die Spezifizierung $\mathbf{S}_0$ der Referenz $\mathbf{F}_0$ in der Vielfalt der Methoden modifiziert wird. Die resultierenden PLA-Frames sind schließlich der *Ergebnisbeurteilung* zuzuführen.

In Abbildung 6.2 (S. 116) ist dieses Vorgehen dargestellt, wobei im Rahmen der Ergebnisbeurteilung eine Verfeinerung vorgenommen ist.

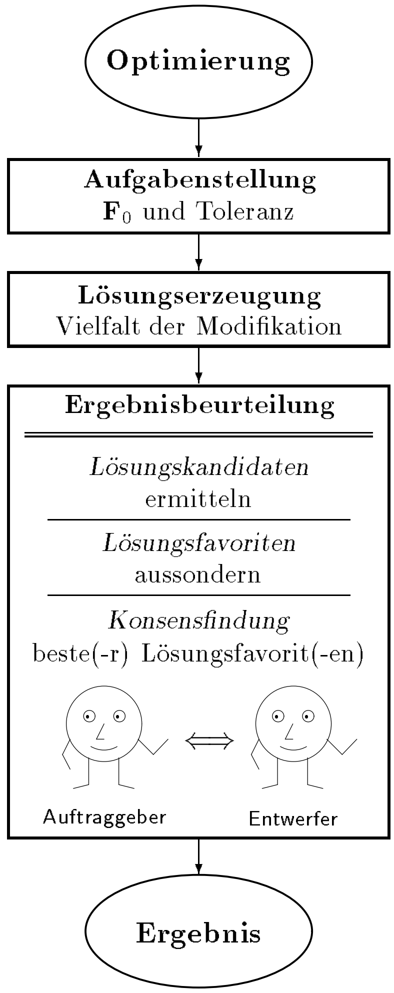

Tabelle 6.2: Vorgehen bei der Optimierung

- **Lösungskandidaten:**
  Zunächst können im ersten Schritt *Lösungskandidaten* ermittelt werden. In diesem Sinne ist ein Lösungskandidat eine *mögliche* Lösung, d. h. unter allen erzeugten PLA-Frames sind diejenigen, die nach bestimmten Kriterien ungünstig erscheinen, fortgelassen.

- **Lösungsfavoriten:**
  An diese erste Auslese schließt sich eine zweite an, die sich nun nur noch auf die Menge der Lösungskandidaten bezieht. Diese zweite Ausleseprozedur findet unter Berücksichtigung eines anderen Kriteriums statt und sondert die sogenannten *Lösungsfavoriten* aus.

Für diese Art einer schrittweisen Auslese können nun je nach den angewendeten Kriterien grundsätzlich zwei Strategien unterschieden werden: Zum einen die *Abweichungs-orientierte* und zum anderen die *Kosten-orientierte*.

6.2.6.1 Abweichungs-orientierte Strategie

Im Rahmen der *Lösungserzeugung* werden bei allen PLA-Frames $\mathbf{F}_j$ die Kosten p_j bestimmt. Als *Lösungskandidaten* treten diejenigen PLA-Frames hervor, die die Kosten c_0 der Referenz $\mathbf{F}_0$ unterschreiten bzw. in Gegenüberstellung mit c_0 eine Reduzierung der Kosten bewirken.

Schließlich werden unter den *Lösungskandidaten* diejenigen PLA-Frames als *Lösungsfavoriten* ausgesondert, bei denen die Abweichungen zur Referenz am geringsten sind, d. h. die *Ergebnisbeurteilung* orientiert sich an den Abweichungen der *Lösungskandidaten*.

6.2.6.2 Kosten-orientierte Strategie

Aus der *Aufgabenstellung* liegt eine Einschätzung der Toleranz für zulässige bzw. akzeptable Abweichungen vor. Von allen zunächst erzeugten PLA-Frames $\mathbf{F}_j$ sind diejenigen hier die *Lösungskandidaten*, deren Abweichungen unterhalb der vorausgesetzten Toleranz liegen. Nur bei den *Lösungskandidaten* werden anschließend die Kosten der PLA-Tabelle p_j bestimmt.

Als *Lösungsfavoriten* werden schließlich diejenigen PLA-Frames ausgesondert, die die geringsten Kosten bzw. in Gegenüberstellung mit der Referenz die größte Reduzierung erzielen können, d. h. *Ergebnisbeurteilung* orientiert sich an den Kosten der *Lösungskandidaten*.

6.2.6.3 Konsensfindung für das Aufwand/Nutzen-Verhältnis

Am Ende einer derartigen Optimierungsprozedur steht also jeweils eine Abweichungs- bzw. Kosten-orientierte Beurteilung der *Lösungskandidaten*, wobei in der Regel mehrere Lösungsfavoriten resultieren. Die Abwägung von „Aufwand“, im Sinne von *zu akzeptierende Abweichungen*, und „Nutzen“, durch *geringere PLA-Kosten*, sollte in einer Konsensfindung idealerweise gemeinsam von Auftraggeber und Entwerfer vorgenommen werden, wie dies bereits in Abbildung 6.2 (S. 116) angedeutet wurde.

6.2.6.4 Bewertung der Strategien

Die beiden alternativen Strategien können gleichermaßen zur Optimierung verwendet werden. Allerdings sind beide nicht frei von Nachteilen:

- **Abweichungs-orientierte Strategie:**
  Für die Abweichungs-orientierte Strategie besteht ein Nachteil darin, daß bei der Ermittlung der *Lösungskandidaten* die Kosten der erzeugten PLA-Frames nur anhand der Referenz bewertet werden. Möglicherweise existiert bereits aus der Grundproblematik einer zu hohen Stromaufnahme (vgl. Abschnitt 2.2.3.2, S. 23) eine noch niedrigere Kostenschranke, die erreicht bzw. unterschritten werden muß.

- **Kosten-orientierte Strategie:**
  Als erstes ist bei der Kosten-orientierten Strategie nachteilig, daß beim Ermitteln der *Lösungskandidaten* eine starre Beschränkung des Toleranzbereiches problematisch sein kann: Hierbei stellt sich die Frage, in welcher Weise bei geringfügigen Verfehlungen des Toleranzbereiches zu verfahren ist. Außerdem sind alle Werte im zulässigen Toleranzbereich gleichwahrscheinlich, d. h. es fehlt an Voraussetzungen, die bei der Selektion von Werten in Richtung auf den ursprünglichen, wenn auch nur fiktiv angenommenen, Wert zielen.

Die Kosten-orientierte Strategie ist insofern von Vorteil, da sie die Laufzeit-intensive Bestimmung der PLA-Kosten zum Ende der Optimierungsprozedur verlagert und somit die Anzahl der erforderlichen Logik-Minimierungen verringert. Statt also wie bei der Abweichungs-orientierten Strategie die Kosten zunächst für *alle* erzeugten PLA-Frames zu bestimmen, geschieht dies bei der Kosten-orientierten Strategie nur für die Teilmenge der *Lösungskandidaten*.

6.3 Evaluation der Methoden

Im folgenden werden die in Kapitel 5 (S. 75ff) vorgestellten Methoden zunächst anhand des *Sinus*-Beispiels, später an weiteren typischen Beispielen, evaluiert. Ihnen allen ist gemeinsam, daß sie aus einem numerischen Kontext kommen und arithmetische Funktionen modellieren, für deren Auswertung aufwendige Operationen durchgeführt werden müssen. Beispielsweise ist das Gewinnen von Näherungswerten mit Hilfe von Reihenentwicklungen sehr Laufzeit-intensiv, so daß sich in solchen Fällen das Ablegen der Werte in PLA-Tabellen anbietet.

6.3.1 Evaluation anhand des Sinus-Beispiels

Wie bereits in Abschnitt 3.1.5 (S. 39) diskutiert, ist die Auswertung der *Sinus*-Funktion für Probleme der Computergrafik geradezu typisch; nicht selten werden bei Echtzeit-Animationen die Werte in nur einen Systemtakt benötigt, z. B. wenn es um die Berechnung von rotierenden Objekten geht. In den folgenden Abschnitten wird anhand dieses Beispiels die Optimierungsprozedur demonstriert, wobei ein besonderer Wert auf die Ausführlichkeit gelegt wurde. Um eine hohe Transparenz zu gewährleisten, werden einzelne Aspekte der Diskussion allerdings vertiefter dargestellt als es in der Praxis nötig wäre. In Abschnitt 6.3.2 (S. 127ff) liegt der Schwerpunkt dann mehr auf der für die Praxis relevanten Vorgehensweise.

6.3.1.1 Aufgabenstellung beim Sinus-Beispiel

Bei diesem Beispiel handelt es sich um die erste Viertelwelle einer *Sinus*-Schwingung, also um die Funktion $f(x) := \sin(x)$ mit dem Definitionsbereich $x \in [0, \frac{\pi}{2}]$. Die Unschärfe für Funktionswerte werden in einer Größenordnung von relativ $\pm 5\%$ willkürlich angenommen, so daß damit ein entsprechender Toleranzbereich vorgegeben ist. Mit Hilfe einer *ad hoc*-Spezifikation wird eine PLA-Tabelle erstellt, von der die Termanzahl bestimmt werden kann. Damit liegt ein PLA-Frame als Referenz vor.

6.3.1.2 Lösungserzeugung beim Sinus-Beispiel

Entsprechend der in Abbildung 6.2 (S. 111) gezeigten Struktur wird nun die *Lösungserzeugung* durchgeführt, d. h. es wird eine minimale Anzahl von PLA-Frames erzeugt, die bei *Einzelner Modifikation* (vgl. Abschnitt 6.2.4.1, S. 109) abgeleitet werden können. Wegen der Besonderheit dieses Beispiels, das die bisherige Diskussion eng begleitet hat, sei bei der Beurteilung der erzeugten PLA-Frames ein maximaler Einblick gewährt. Aus diesem Grunde wird hier die Ermittlung der Kosten vorgezogen und findet nicht erst für die Lösungskandidaten statt, sondern obligatorisch bereits für jeden erzeugten PLA-Frame.

6.3.1.3 Spezifikation der Referenz

Als Referenz in Gestalt des PLA-Frames $\mathbf{F}_0$ wird eine *ad hoc* spezifizierte PLA-Tabelle verwendet. Die Spezifizierung geschieht wie bereits in Abschnitt 3.1.5 (S. 39) erwähnt, d. h. $\mathbf{S}_0$ setzt sich wie folgt zusammen:

$$\mathsf{Dis}(N_0) \rightarrow \mathsf{Scl}(M_0) \rightarrow \mathsf{Rnd\_Round} \tag{6.12}$$

Zunächst wird mit $\mathsf{Dis}(N_0)$ eine lineare Diskretisierung in $N_0 := 100$ Stützstellen vorgenommen. Mit Hilfe von $\mathsf{Scl}(M_0)$ erfolgt eine lineare Skalierung durch Multiplikation mit dem Faktor 1E3; es ergibt sich daraus als Maximum $M_0 := 1\text{E}3$, da für alle $x \in [0, \frac{\pi}{2}]$ gilt: $\sin(x) \leq 1$. Die Konversion durch Rundung mit der Methode $\mathsf{Rnd\_Round}$ schließt die eigentliche Spezifizierung ab; es folgt nur noch die obligatorische Kodierung der Indizes bzw. Einträge in die Binärdarstellung durch $\mathsf{Bin}(n)$ bzw. $\mathsf{Bin}(m)$.

Aus dieser Spezifikation resultierte die PLA-Tabelle p_0, wie sie im Anhang B in Tabelle B.1 (S. 158) dargestellt ist (vgl. Abbildung 6.2, S. 111). Nach einer Logik-Minimierung mit ESPRESSO im Standard-Modus ergibt sich eine Termanzahl $\ell := 93$, so daß das Kosten-Merkmal $c_0 := \ell = 93$ lautet. Es wird angenommen, daß p_0 sehr wenig Unschärfe enthält, so daß p_0 ein Abweichungs-Merkmal von $d_0 := \emptyset$ zugeordnet wird. Gemäß Punkt 3 in Abschnitt 6.2.3 (S. 108) ist eine allgemeine Toleranz für die Funktionswerte aus dem Kontext der Aufgabenstellung abzuleiten: Sie sei hier mit $\pm 5\%$ als Relativ-Toleranz festgelegt.

6.3.1.4 Modifikationen in der Numerik-Schicht

In der Numerik-Schicht wird f z. B. einerseits mit $N_0 := 100$ Einträgen, andererseits entsprechend Option O-1: *Anzahl der Stützstellen als 2er-Potenz* aber auch mit $N_1 := 2^7 = 128$ Einträgen diskretisiert (vgl. Abschnitt 3.1.5, S. 39f und Abschnitt 3.3.2, S. 44f). Ferner sei f bei der linearen Skalierung zum einen mit dem Faktor 1E3 und zum anderen entsprechend Option O-4: *Bestimmung der Skalierungsabbildung* mit $2^{10} = 1024$ multipliziert, so daß sich jeweils ein Maximum von $M_0 := 1000$ bzw. $M_1 := 1024$ ergibt; beide Maxima führen wegen $\mathsf{Bits}(M_0) = \mathsf{Bits}(M_1)$ zu $m := 10$ Bit für die Binärdarstellung der Einträge.

Innerhalb der Numerik-Schicht ergeben sich somit drei verschiedenartig modifizierte Versionen der Spezifizierung, nämlich neben der Kombination von N_0 mit M_0 bei der Referenz noch zusätzlich die beiden Kombinationen von N_1 mit M_0 und von N_0 mit M_1. Damit ergibt sich als Vielfalt $V_{\mathsf{Dis}} := 2$ und $V_{\mathsf{Scl}} := 2$, so daß sich bei *Einzelner Modifikation* (vgl. Abschnitt 6.2.4.1, S. 109) für die Numerik-Schicht insgesamt eine Vielfalt von $V_{\text{num}} := 2 + 2 - 1 = 3$ (vgl. $\ddagger^2$ bei Tabelle 6.1, S. 110) ergibt.

In Tabelle 6.3 (S. 120) sind diese drei resultierenden PLA-Frames $\mathbf{F}_j$ für $0 \leq j \leq 2$ dargestellt, wobei für den Abweichungsvektor d_j die beiden numerischen Kriterien das *Maximum der absoluten Differenzen* M_{abs} (vgl. Gleichung 6.7, S. 114) und der *Mittelwert über die relativ summierten Differenzen in Prozent* A_{rel} (vgl. Gleichung 6.11, S. 115) sowie das binäre Kriterium *Maximum der unterschiedlichen Bitstellen* B_{ent} (vgl. Gleichung 6.5, S. 114) verwendet werden. Sie werden jeweils im Verhältnis zur Referenz $\mathbf{F}_0$ erfaßt.

Numerik-Schicht: 3 Spezifizierungen mit Modifikationen bei Dis() und Scl()	Kosten		Abweichung $d(p_0)$		
	c	r_0	$\mathrm{M_{abs}}$	$\mathrm{A_{rel}}$	$\mathrm{B_{ent}}$
$\mathbf{S}_0$: Dis(N_0) $\to$ Scl(M_0) $\to$ Rnd_Round	93	0%	0	0%	0
$\mathbf{S}_1$: Dis(N_0) $\to$ Scl(M_1) $\to$ Rnd_Round	88	5%	23	1%	7
$\mathbf{S}_2$: Dis(N_1) $\to$ Scl(M_0) $\to$ Rnd_Round	118	-	-	-	-

Tabelle 6.3: PLA-Frames in der Numerik-Schicht beim Sinus-Beispiel

Reduzierung

Für die Kosten c_j ist die sich jeweils ergebende Termanzahl ℓ der Wahrheitstabelle p_j eingetragen; zusätzlich wird jeweils die prozentuale Reduzierung im Verhältnis zu den Kosten c_0 der Referenz mitgeführt. Sie sei für die Kosten c_j mit c_0 als Referenzkosten wie folgt definiert:

$$r_0(c_j) := \begin{cases} \frac{c_0 - c_j}{c_0} \cdot 100\% : & 0 < c_j \leq c_0 \\ - \quad : & \text{sonst} \end{cases} \tag{6.13}$$

Damit ist $r_0(c_j)$ insbesondere nicht negativ. Wenn offensichtlich ist, für welchen PLA-Frame $\mathbf{F}_j$ die Reduzierung $r_0(c_j)$ bestimmt worden ist, sei zur Vereinfachung auch nur r_0 als Bezeichnung zulässig.

Während die Abweichung der Referenz $d_0 := \emptyset$ ist, ergibt sich im Detail für den PLA-Frame $\mathbf{F}_1$ bzw. durch die Spezifizierung $\mathbf{S}_1$ folgende Abweichung gegenüber $\mathbf{F}_0$: Die *maximale Absolut-Differenz* beträgt $\mathrm{M_{abs}}(d_1) := 23$ (vgl. Schreibweise, S. 115), die *mittlere Relativ-Differenz* ist $\mathrm{A_{rel}}(d_1) := 1\%$ und die *maximale Anzahl unterschiedlicher Bitstellen* beträgt $\mathrm{B_{ent}}(d_1) := 7$. Ein Vergleich zwischen der Referenz und $\mathbf{F}_2$ erscheint aufgrund der unterschiedlichen Anzahl an Einträgen $N_0 \neq N_1$ nicht sinnvoll. Die Kosten sind erwartungsgemäß bei $\mathbf{F}_2$ mit $p_2 := 118$ Termen am größten. Überraschend sind dagegen die geringen Kosten bei $\mathbf{F}_1$ von $c_1 := 88$ Termen, da dieser Wert gegenüber der Referenz $\mathbf{F}_0$ mit $c_0 := 93$ Termen bereits eine Reduzierung von $r_0(c_1) := 5\%$ bedeutet.

6.3.1.5 Modifikationen in der Konversions-Schicht

Bei *Einzelner Modifikation* (vgl. Abschnitt 6.2.4.1, S. 109) ergibt sich für die Konversions-Schicht eine Vielfalt von $V_{\mathrm{kon}} \geq 11$; siehe Ausdrücke bei $\ddagger^1$ und $\ddagger^3$ (vgl. Tabelle 6.1, S. 110). Da für den Int_Rel-Modus auch nur ein Wertepaar zur Angabe der Toleranz verwendet wird, kommt es auch im konkreten Beispiel zu einer tatsächlichen Vielfalt von 11 verschiedenartigen Modifikationen. In Tabelle 6.4 (S. 121) sind alle in der Konversions-Schicht erzeugten PLA-Frames $\mathbf{F}_j$ mit $j \in \{0, 3, \ldots, 12\}$ aufgeführt; in den Spalten sind von links nach rechts die Komponenten Spezifizierung $\mathbf{S}_j$, Kosten c_j, Reduzierung $r_0(c_j)$ und Abweichungen $d_j(p_0)$ angeordnet.

In der ersten Zeile befinden sich die Komponenten der mit der Methode Rnd_Round erzeugten Referenz $\mathbf{F}_0$. Anstatt von Rnd_Round sind in den nächsten beiden Zeilen die Methoden Rnd_Floor und Rnd_Ceil eingesetzt; bei den Kosten und der Reduzierung schlägt sich diese Modifikation nicht nieder, wohl aber bei der Abweichung wie bei $\mathrm{M_{abs}}$ und $\mathrm{B_{ent}}$ ersichtlich ist. Erhebliche Veränderungen, sowohl bei den Kosten bzw. der Reduzierung als auch bei der Abweichung, sind

Konversions-Schicht: 11 Spezifizierungen	Kosten		Abweichung $d(p_0)$		
Modifikation mit Konversions-Methoden	c	r_0	M_{abs}	A_{rel}	B_{ent}
$\mathbf{S}_0$: $\cdots \to$ Rnd_Round	93	0%	0	0%	0
$\mathbf{S}_3$: $\cdots \to$ Rnd_Floor	93	0%	1	0%	8
$\mathbf{S}_4$: $\cdots \to$ Rnd_Ceil	93	0%	1	0%	7
$\mathbf{S}_5$: $\cdots \to$ P2_Low	21	77%	488	27%	8
$\mathbf{S}_6$: $\cdots \to$ P2_High	17	82%	488	26%	8
$\mathbf{S}_7$: $\cdots \to$ P2_Near	16	83%	488	22%	8
$\mathbf{S}_8$: $\cdots \to$ Int_Rel(5%, 5%) $\to$ Sel_Low	91	2%	51	3%	9
$\mathbf{S}_9$: $\cdots \to$ Int_Rel(5%, 5%) $\to$ Sel_High	78	16%	49	3%	8
$\mathbf{S}_{10}$: $\cdots \to$ Int_Rel(5%, 5%) $\to$ Sel_Mean	90	3%	14	0%	7
$\mathbf{S}_{11}$: $\cdots \to$ Int_Rel(5%, 5%) $\to$ Sel_Min1	31	67%	44	1%	8
$\mathbf{S}_{12}$: $\cdots \to$ Int_Rel(5%, 5%) $\to$ Sel_NP2	16	83%	488	22%	8

Tabelle 6.4: PLA-Frames in der Konversions-Schicht beim Sinus-Beispiel

im nächsten Zeilenblock bei den Modi der P2-Methode zu beobachten. Das Fazit für P lautet hier: Hohe Reduzierungswerte bei gleichzeitig großer Abweichung.

Im letzten Zeilenblock sind die Intervalle anhand des anfangs festgelegten Toleranzbereiches dimensioniert, d. h. Int_Rel wird mit $\pm 5\%$ parametrisiert. Bei der Betrachtung der Abweichungen $d(p_0)$ bezüglich der Referenz fäll hier der große Wert beim Sel_NP2-Modus auf, dessen resultierende PLA-Tabelle p_{12} mit der durch P2_Near erzeugten PLA-Tabelle p_7 ist. Sie erreichen im Verhältnis zur Referenz eine hohe Reduzierung r_0 bei hoher Abweichung $d(p_0)$; dies bestätigt die intuitive Annahme, daß niedrige Kosten mit hohen Abweichungen einhergehen. Bemerkenswert ist allerdings der durch Sel_Min1 erzeugte PLA-Frame $\mathbf{F}_{11}$, der ähnlich große Abweichungen $d(p_0)$ wie der durch Sel_High erzeugte PLA-Frame $\mathbf{F}_9$ aufweist, dabei jedoch eine mehr als vierfache Reduzierung der Kosten von $r_0(c_{11}) := 67\%$ erreicht.

6.3.1.6 Modifikationen in der Binär-Schicht

Beim *Sinus*-Beispiel erzeugen die Methoden der Binär-Schicht, entsprechend ihrer Vielfalt, die durch die Anzahl der möglichen Modi und sinnvollen Parameter bedingt ist, weitere $V_{\text{bin}} := 60$ PLA-Frames. Diese sind in Tabelle 6.5 (S. 122) zusammengefaßt; dabei wird die Spezifizierung der Referenz $\mathbf{S}_0$ insofern modifiziert, daß sich an $\mathbf{S}_0$ die Anwendung einer der drei Binär-Methoden Psd, Sim oder Hyp anschließt, d. h. sie erzeugt aus der PLA-Tabelle p_0 eine abgwandelte PLA-Tabelle.

- **Psd_Inc und Psd_Dec:**
  Bei der Psd-Methode kommt es für beide Modi Psd_Inc und Psd_Dec jeweils zu einem Bereich von $2 \leq e \leq 7$, so daß sich für beide Modi insgesamt $2 \cdot 6 = 12$ Werte für den ersten Parameter ergeben. Die maximale Anzahl von $2 \cdot (m-2)$ von $2 \cdot (10-2) = 16$ kann hier nicht erreicht werden, da keine 0- bzw. 1-Sequenz die volle Länge von maximal m-Bit erreicht. Aufgrund wiederholter Anwendungen des Psd_Inc-Modus sind noch vier weitere Modifikationen zu verzeichnen, was durch den zweiten Parameter gekennzeichnet wird; er gibt an, um welche der Anwendungen es sich jeweils handelt. Insgesamt ergeben sich für die Psd-Methode $V_{\mathsf{Psd}} := 16$ verschiedenartige Modifikationen.

Binär-Methoden	Kosten		Abwchg. $d(p_0)$		
30 Modifikationen an p_0	c	r_0	M_{abs}	A_{rel}	B_{ent}
S_{13}: → Psd_Inc(2, 1)	65	30%	126	2%	8
S_{14}: → Psd_Inc(2, 2)	60	35%	155	3%	8
S_{15}: → Psd_Inc(2, 3)	61	34%	155	4%	8
S_{16}: → Psd_Inc(2, 4)	60	35%	170	4%	8
S_{17}: → Psd_Inc(3, 1)	77	17%	54	0%	8
S_{18}: → Psd_Inc(3, 2)	78	16%	68	0%	8
S_{19}: → Psd_Inc(4, 1)	87	6%	26	0%	8
S_{20}: → Psd_Inc(5, 1)	89	4%	12	0%	8
S_{21}: → Psd_Inc(6, 1)	90	3%	2	0%	8
S_{22}: → Psd_Inc(7, 1)	90	3%	2	0%	8
S_{23}: → Psd_Dec(2, 1)	62	33%	119	2%	6
S_{24}: → Psd_Dec(3, 1)	83	11%	55	0%	5
S_{25}: → Psd_Dec(4, 1)	86	8%	29	0%	4
S_{26}: → Psd_Dec(5, 1)	88	5%	15	0%	4
S_{27}: → Psd_Dec(6, 1)	89	4%	2	0%	1
S_{28}: → Psd_Dec(7, 1)	89	4%	2	0%	1
S_{29}: → Sim_Bin(1)	86	8%	16	0%	1
S_{30}: → Sim_Bin(2)	86	8%	16	0%	2
S_{31}: → Sim_Bin(3)	78	16%	16	0%	3
S_{32}: → Sim_Bin(4)	71	24%	16	0%	4
S_{33}: → Sim_Bin(5)	67	28%	16	0%	5
S_{34}: → Sim_Bin(6)	64	31%	16	0%	6
S_{35}: → Sim_Bin(7)	63	32%	16	0%	7
S_{36}: → Sim_Num(1)	87	6%	1	0%	1
S_{37}: → Sim_Num(2)	91	2%	2	0%	5
S_{38}: → Sim_Num(3)	89	4%	3	0%	5
S_{39}: → Sim_Num(4)	87	6%	4	0%	5
S_{40}: → Sim_Num(5)	87	6%	5	0%	5
S_{41}: → Sim_Num(6)	84	10%	6	0%	5
S_{42}: → Sim_Num(7)	85	9%	7	0%	5

... Fortsetzung	Kosten		Abwchg. $d(p_0)$		
30 Modifikationen an p_0	c	r_0	M_{abs}	A_{rel}	B_{ent}
S_{43}: → Sim_Num(8)	84	10%	8	0%	5
S_{44}: → Sim_Num(9)	83	11%	9	0%	5
S_{45}: → Sim_Num(10)	82	12%	10	0%	5
S_{46}: → Sim_Num(11)	76	18%	11	0%	5
S_{47}: → Sim_Num(12)	78	16%	12	0%	5
S_{48}: → Sim_Num(13)	71	24%	13	0%	5
S_{49}: → Sim_Num(14)	71	24%	14	0%	6
S_{50}: → Sim_Num(15)	64	31%	15	0%	7
S_{51}: → Sim_Num(16)	64	31%	16	0%	7
S_{52}: → Hyp_Trm(1)	92	1%	256	0%	2
S_{53}: → Hyp_Trm(2)	92	1%	256	0%	2
S_{54}: → Hyp_Trm(3)	92	1%	496	0%	5
S_{55}: → Hyp_Trm(4)	93	0%	129	0%	2
S_{56}: → Hyp_Trm(5)	94	-	33	0%	2
S_{57}: → Hyp_Trm(6)	94	-	16	0%	1
S_{58}: → Hyp_Trm(7)	91	2%	16	0%	1
S_{59}: → Hyp_Trm(8)	93	0%	4	0%	1
S_{60}: → Hyp_Trm(9)	93	0%	140	0%	3
S_{61}: → Hyp_Trm(10)	92	1%	2	0%	1
S_{62}: → Hyp_Trm(11)	92	1%	33	0%	2
S_{63}: → Hyp_Trm(12)	92	1%	41	0%	3
S_{64}: → Hyp_Trm(13)	88	5%	174	1%	5
S_{65}: → Hyp_Trm(14)	91	2%	8	0%	3
S_{66}: → Hyp_Trm(15)	93	0%	16	0%	2
S_{67}: → Hyp_Trm(16)	92	1%	8	0%	1
S_{68}: → Hyp_Trm(17)	95	-	40	0%	2
S_{69}: → Hyp_Trm(18)	91	2%	36	0%	2
S_{70}: → Hyp_Trm(19)	93	0%	32	0%	1
S_{71}: → Hyp_Trm(20)	90	3%	296	1%	4
S_{72}: → Hyp_Trm(21)	92	1%	16	0%	1

Tabelle 6.5: PLA-Frames in der Binär-Schicht beim Sinus-Beispiel

- **Sim_Bin und Sim_Num:**
  Für Sim_Bin ergibt sich beim Parameter für die binäre Ähnlichkeit z_{bin} ein Bereich von $1 \leq z_{\text{bin}} \leq 7$. Bei Sim_Num ist für den Parameter z_{num} ein Bereich von $1 \leq z_{\text{num}} \leq 16$ möglich. Insgesamt ergeben sich somit für die Sim-Methode $V_{\text{Sim}} := 7 + 16 = 23$ verschiedenartige Modifikationen.

- **Hyp_Trm:**
  Für die Hyp_Trm-Methode werden die Terme von der PLA-Tabelle p_0 der Referenz $\mathbf{F}_0$ verwendet; aus der mit ESPRESSO (Standard-Modus) minimierten Termmenge ergibt sich eine Anzahl von $\ell := 93$ Termen, d. h. $c_0 := \ell = 93$. Durch die zusätzliche Anwendung der *Consensus Methode* entstehen 21 Hyper-Terme, so daß die Methode Hyp $V_{\text{Hyp}} := 21$ verschiedenartige Modifikationen erzeugt.

Bei *Einzelner Modifikation* (vgl. Abschnitt 6.2.4.1, S. 109) ergibt sich für die Binär-Schicht $V_{\text{bin}} \geq 4$; siehe den Ausdruck bei $\ddagger^4$ (vgl. Tabelle 6.1, S. 110). Durch Addition der Vielfalt der einzelnen binären Methoden, ergibt sich insgesamt eine Vielfalt von $V_{\text{bin}} := 16 + 23 + 21 = 60$ verschiedenartigen Modifikationen.

In der Binär-Schicht werden Modifikationen an der Referenz $\mathbf{S}_0$ vorgenommen: Beim Analysieren der Resultate der Hyp-Methode, sind nur geringfügig unterschiedliche Kosten gegenüber der Referenz zu verzeichnen, so daß diese Ergebnisse keine weitere Beachtung finden. Dagegen bringen die Psd- und Sim-Methode Resultate hervor, die in Zusammenhang mit ihren Parametern interpretiert werden können: So führen niedrige Parameterwerte bei Psd_Inc und Psd_Dec zu geringen Kosten bei gleichzeitig hohen Abweichungen.

Umgekehrt verhalten sich die Kosten bei Sim_Bin, die bei hohen Parameterwerten niedrig sind; die Abweichungen, die hier durch Angleichung von Einträgen verursacht werden, sind nur bei binärer Interpretation zu erkennen und bleiben numerisch gesehen mit $\mathrm{M_{abs}} := 16$ konstant: Wenn es zu Angleichungen kommt, geschehen bei steigendem Parameter z auch immer dieselben Veränderungen, wie sie schon für kleinere z Parameter vorkamen. Bei Sim_Num werden die Kosten bei steigendem Parameterwert in ihrer Tendenz geringer, wobei die Termanzahl allerdings nicht monoton abnimmt. Bedingt durch das Prinzip dieser Methode, sind die numerischen Abweichungen exakt durch den Parameter steuerbar.

6.3.1.7 Anzahl der PLA-Frames

In den einzelnen Schichten sind zusammenfassend aus den vorhergehenden Abschnitten 6.3.1.4, 6.3.1.5 und 6.3.1.6 in folgender Vielfalt PLA-Frames erzeugt worden:

- Numerik-Schicht: $V_{\mathrm{num}} := 3$
- Konversions-Schicht: $V_{\mathrm{kon}} := 11$
- Binär-Schicht: $V_{\mathrm{bin}} := 60$

Inklusive der unmodifizierten Spezifizierung $\mathbf{S}_0$ kommt es damit für alle Schichten insgesamt zu $V := V_{\mathrm{num}} + V_{\mathrm{kon}} - 1 + V_{\mathrm{bin}} := 3 + 11 - 1 + 60 = 73$ verschiedenartigen Spezifizierungen.

6.3.1.8 Lösungskandidaten beim Sinus-Beispiel

Abweichend vom strategischen Vorgehen, wie es in Abschnitt 6.2.6 (S. 115) beschrieben ist, liegen bei diesem Beispiel bereits für alle PLA-Frames die ermittelten Kosten vor (vgl. Abschnitt 6.3.1.2, S. 118). Bei der Kosten-orientierten Strategie geschieht nun die Bestimmung von *Lösungskandidaten* und *Lösungsfavoriten* wie folgt: Als *Lösungskandidaten* werden diejenigen PLA-Frames ermittelt, die die vorgegebene Toleranz in ihrer Abweichung[6] unterschreiten. Aus der Aufgabenstellung ist dafür eine Relativ-Toleranz von $\pm 5\%$ gegeben.

In Tabelle 6.6 (S. 124) sind diejenigen PLA-Frames aufgeführt, bei denen alle Einträge *innerhalb* der relativen Toleranz von $\pm 5\%$ liegen; wegen dieser Auslese fehlen insbesondere die durch Sel_NP2 und P2-Modi erzeugten PLA-Frames, da sie auch *außerhalb* der Intervalle liegende Werte selektieren. Für alle Lösungskandidaten gilt, daß sich in binärer Interpretation beim Abweichungs-Merkmal $\mathrm{B_{ent}}$ von Null verschiedene Werte ergeben können; bezüglich numerischer Interpretation weisen sie bei den Merkmalen $\mathrm{M_{abs}}$ und $\mathrm{A_{rel}}$ allerdings *keine* Abweichung auf. Aus diesem Grunde sind die Spalten für Abweichungen in den Tabellen der Lösungskandidaten unterdrückt worden.

[6]Hinweis: Die in den drei Tabellen 6.3 (S. 120), 6.4 (S. 121) und 6.5 (S. 122) aufgeführte Abweichung $d(p_0)$ ist jeweils auf die PLA-Tabelle p_0 der Referenz bezogen. An dieser Stelle ist jedoch die auf das Toleranz-Intervall bezogene Abweichung gemeint!

Lösungskandidaten 20 PLA-Frames bei ±5% Relativ-Toleranz	Kosten c	r_0
$\mathbf{S}_{11}$: $\cdots \to$ Int_Rel(5%,5%) $\to$ Sel_Min1	31	67%
$\mathbf{S}_{48}$: $\mathbf{S}_0 \to$ Sim_Num(13)	71	24%
$\mathbf{S}_{49}$: $\mathbf{S}_0 \to$ Sim_Num(14)	71	24%
$\mathbf{S}_{46}$: $\mathbf{S}_0 \to$ Sim_Num(11)	76	18%
$\mathbf{S}_9$: $\cdots \to$ Int_Rel(5%,5%) $\to$ Sel_High	78	16%
$\mathbf{S}_{47}$: $\mathbf{S}_0 \to$ Sim_Num(12)	78	16%
$\mathbf{S}_{45}$: $\mathbf{S}_0 \to$ Sim_Num(10)	82	12%
$\mathbf{S}_{44}$: $\mathbf{S}_0 \to$ Sim_Num(9)	83	11%
$\mathbf{S}_{41}$: $\mathbf{S}_0 \to$ Sim_Num(6)	84	10%
$\mathbf{S}_{43}$: $\mathbf{S}_0 \to$ Sim_Num(8)	84	10%
$\mathbf{S}_{42}$: $\mathbf{S}_0 \to$ Sim_Num(7)	85	9%
$\mathbf{S}_{25}$: $\mathbf{S}_0 \to$ Psd_Dec(4,1)	86	8%
$\mathbf{S}_{19}$: $\mathbf{S}_0 \to$ Psd_Inc(4,1)	87	6%
$\mathbf{S}_{36}$: $\mathbf{S}_0 \to$ Sim_Num(1)	87	6%
$\mathbf{S}_{39}$: $\mathbf{S}_0 \to$ Sim_Num(4)	87	6%
$\mathbf{S}_{40}$: $\mathbf{S}_0 \to$ Sim_Num(5)	87	6%
$\mathbf{S}_{26}$: $\mathbf{S}_0 \to$ Psd_Dec(5,1)	88	5%
$\mathbf{S}_{20}$: $\mathbf{S}_0 \to$ Psd_Inc(5,1)	89	4%
$\mathbf{S}_{27}$: $\mathbf{S}_0 \to$ Psd_Dec(6,1)	89	4%
$\mathbf{S}_{22}$: $\mathbf{S}_0 \to$ Psd_Dec(7,1)	89	4%

Lösungskandidaten 19 PLA-Frames (Fortsetzung)	Kosten c	r_0
$\mathbf{S}_{38}$: $\mathbf{S}_0 \to$ Sim_Num(3)	89	4%
$\mathbf{S}_{10}$: $\cdots \to$ Int_Rel(5%,5%) $\to$ Sel_Mean	90	3%
$\mathbf{S}_{21}$: $\mathbf{S}_0 \to$ Psd_Inc(6,1)	90	3%
$\mathbf{S}_{22}$: $\mathbf{S}_0 \to$ Psd_Inc(7,1)	90	3%
$\mathbf{S}_8$: $\cdots \to$ Int_Rel(5%,5%) $\to$ Sel_Low	91	2%
$\mathbf{S}_{37}$: $\mathbf{S}_0 \to$ Sim_Num(2)	91	2%
$\mathbf{S}_{58}$: $\mathbf{S}_0 \to$ Hyp_Trm(7)	91	2%
$\mathbf{S}_{65}$: $\mathbf{S}_0 \to$ Hyp_Trm(14)	91	2%
$\mathbf{S}_{69}$: $\mathbf{S}_0 \to$ Hyp_Trm(18)	91	2%
$\mathbf{S}_{61}$: $\mathbf{S}_0 \to$ Hyp_Trm(10)	92	1%
$\mathbf{S}_{62}$: $\mathbf{S}_0 \to$ Hyp_Trm(11)	92	1%
$\mathbf{S}_{63}$: $\mathbf{S}_0 \to$ Hyp_Trm(12)	92	1%
$\mathbf{S}_{67}$: $\mathbf{S}_0 \to$ Hyp_Trm(16)	92	1%
$\mathbf{S}_{72}$: $\mathbf{S}_0 \to$ Hyp_Trm(21)	92	1%
$\mathbf{S}_0$: $\cdots \to$ Rnd_Round	93	0%
$\mathbf{S}_3$: $\cdots \to$ Rnd_Floor	93	0%
$\mathbf{S}_4$: $\cdots \to$ Rnd_Ceil	93	0%
$\mathbf{S}_{59}$: $\mathbf{S}_0 \to$ Hyp_Trm(8)	93	0%
$\mathbf{S}_{57}$: $\mathbf{S}_0 \to$ Hyp_Trm(6)	94	-

Tabelle 6.6: Lösungskandidaten des Sinus-Beispiels bei relativer Toleranz

6.3.1.9 Lösungsfavoriten beim Sinus-Beispiel

Aus der Menge der 39 Lösungskandidaten von Tabelle 6.6 (S. 124), die bei einer relativen Toleranz von $\pm 5\%$ ermittelt wurden, sind nun abschließend Kosten-orientiert die Lösungsfavoriten auszusondern: Für diese Auslese ist die Festlegung einer Mindest-Reduzierung gegenüber der Referenz $\mathbf{F}_0$ zweckmäßig. Im konkreten Fall sei für die Lösungskandidaten $\mathbf{F}_j$ eine minimale Reduzierung von $r_0(c_j) := 10\%$ gegenüber $c_0 := 93$ Termen gefordert, so daß von ihnen diejenigen PLA-Frames die neue Teilmenge der Lösungsfavoriten bilden, die weniger als $\frac{100\%-10\%}{100\%} \cdot 93 = 84$ Terme erzielen. Es ergeben sich auf diese Weise 10 Lösungsfavoriten, die in Tabelle 6.7 (S. 124) zusammengestellt sind.

Lösungsfavoriten 14 PLA-Frames bei 10% Mindest-Reduzierung	Kosten c	$r_0(c)$
$\mathbf{S}_{11}$: $\cdots \to$ Int_Rel(5%,5%) $\to$ Sel_Min1	31	67%
$\mathbf{S}_{48}$: $\mathbf{S}_0 \to$ Sim_Num(13)	71	24%
$\mathbf{S}_{49}$: $\mathbf{S}_0 \to$ Sim_Num(14)	71	24%
$\mathbf{S}_{46}$: $\mathbf{S}_0 \to$ Sim_Num(11)	76	18%
$\mathbf{S}_9$: $\cdots \to$ Int_Rel(5%,5%) $\to$ Sel_High	78	16%
$\mathbf{S}_{47}$: $\mathbf{S}_0 \to$ Sim_Num(12)	78	16%
$\mathbf{S}_{45}$: $\mathbf{S}_0 \to$ Sim_Num(10)	82	12%
$\mathbf{S}_{44}$: $\mathbf{S}_0 \to$ Sim_Num(9)	83	11%
$\mathbf{S}_{41}$: $\mathbf{S}_0 \to$ Sim_Num(6)	84	10%
$\mathbf{S}_{43}$: $\mathbf{S}_0 \to$ Sim_Num(8)	84	10%

Tabelle 6.7: Lösungsfavoriten des Sinus-Beispiels bei einer Mindest-Reduzierung von 10%

6.3.1.10 Ergebnisbeurteilung beim Sinus-Beispiel

Die PLA-Frames mit höheren Werten bei der Reduzierung verursachen innerhalb des Toleranzbereiches zwar größere Abweichungen, sind aber dennoch interessant: Die höchste Reduzierung erzielt der PLA-Frame $\mathbf{F}_{11}$ mit $r_0(c_{11}) := 67\%$, bei dem in der Spezifizierung $\mathbf{S}_{11}$ eine Modifikation gegenüber $\mathbf{S}_0$ mit den Methoden Int_Rel$(5\%, 5\%)$ und Sel_Min1 vorgenommen wird. Die PLA-Kosten sind damit sogar auf ein Drittel des bei $\mathbf{F}_0$ *ad hoc* erreichten Wertes reduzierbar. Im Vergleich zum *ad hoc* spezifizierten Graphen in Abbildung 3.8 (S. 50) ist der aus PLA-Tabelle p_{11} resultierende Graph in Abbildung 6.3 (S. 125) dargestellt.

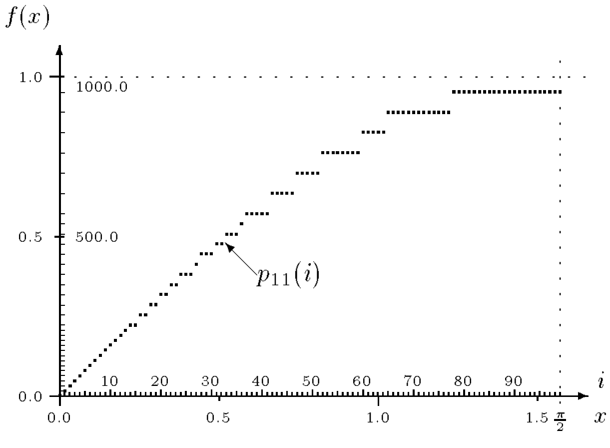

Abbildung 6.3: Graph zu p_{11} durch Int_Rel(5%,5%) $\rightarrow$ Sel_Min1, Reduzierung: $r_0(c_{11}) := 67\%$

Die Betrachtung des p_{11}-Graphen zeigt anschaulich das Ausmaß der dort vorgenommenen Abweichungen, obwohl sie noch innerhalb der $\pm 5\%$-igen Toleranz liegen. Eine etwas geringere Reduzierung mit $r_0(c_{49}) := 24\%$ erzielt der PLA-Frame $\mathbf{F}_{49}$, dessen Graph in Abbildung 6.4 (S. 125) dargestellt ist.

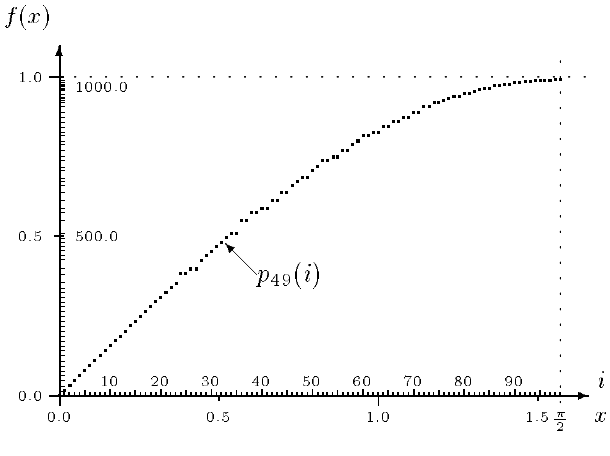

Abbildung 6.4: Graph zu p_{49} durch Sim_Num(14), Reduzierung: $r_0(c_{49}) := 24\%$

Bei der Spezifizierung $\mathbf{S}_{49}$ wurde die Methode Sim_Num(14) auf die PLA-Tabelle p_0 angewendet. Die Methode zielt darauf ab, *vergleichbare* Einträge einander anzugleichen; das ist im Graphen gut zu erkennen. Außerdem zeigt sich hier, wie sinnvoll die restriktive Definition des Begriffes *vergleichbar* in Abschnitt 5.2.2.2 (S. 95) ist, bei der nur benachbarte Einträge für Angleichungen zugelassen werden, deren Indizes in der PLA-Tabelle nebeneinander liegen. Beim PLA-Frame $\mathbf{F}_{49}$ bedeutet der Parameter $z_{\text{num}} := 14$ für die Methode Sim_Num, daß *vergleichbare* Einträge eine numerische Differenz von bis zu 14 haben dürfen, um einander angeglichen zu werden.

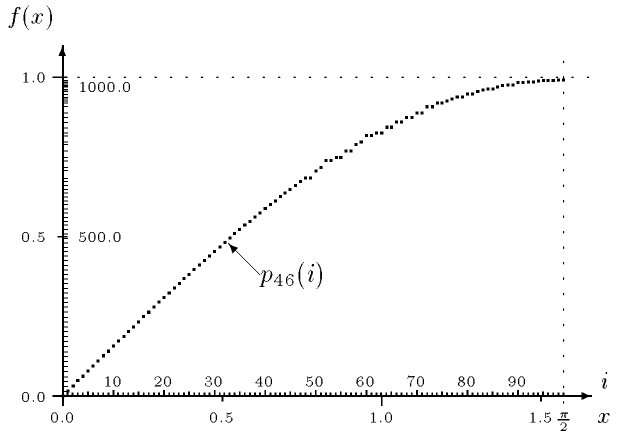

Abbildung 6.5: Graph zu p_{46} durch Sim_Num(11), Reduzierung: $r_0(c_{46}) := 18\%$

Eine verringerte Differenz liegt bei dem PLA-Frame $\mathbf{F}_{46}$ vor, bei dem die Methode Sim_Num(11) auf die PLA-Tabelle p_0 angewendet wird. Der Graph dazu in Abbildung 6.5 (S. 126) zeigt dies deutlich, da erst ab dem 50. Index ($i := 49$) modifizierte Einträge auftreten, denn erst ab dort ist das anfänglich stark monotone Anwachsen des Graphen soweit abgeschwächt, daß zwischen *vergleichbaren* Einträgen die numerische Differenz nur noch höchstens $z_{\text{num}} := 11$ beträgt. Die hierbei erzielte Reduzierung von $r_0(c_{46}) := 18\%$ ist immer noch ein beachtlicher Effekt.

6.3.2 Evaluation an weiteren Beispielen

Im folgenden wird die Evaluation der Methoden an weiteren Beispielen im Überblick dargestellt, d. h. anstatt die PLA-Frames im Detail aufzuführen, werden jeweils nur die durch eine Kosten-orientierte Optimierensstrategie ausgewiesenen fünf besten Lösungsfavoriten präsentiert. Durch eine übergeordnete Sichtweise verlagert sich der Schwerpunkt der Betrachtungen weg von den PLA-Frames und hin zu den Methoden, die zur Modifikation eingesetzt werden.

Generell wird die Vielfalt bei den Methoden auf *Einzelne Modifikation* beschränkt (vgl. Abschnitt 6.2.4.1, S. 109), so daß eine modifizierte Spezifizierung durch die dabei eingesetzte Methode identifiziert werden kann, da hierbei keine weiteren Modifikationen zulässig sind. Auf *Multiple Modifikationen* wurde zugunsten der Übersichtlichkeit verzichtet, zumal sich auch ohne mehrfache Kombination von Methoden bereits brauchbare Ergebnisse erzielen lassen.

6.3.2.1 Erläuterungen zu Begriffen und Variablen

In diesem Abschnitt sind zunächst Erläuterungen zu Begriffen und Variablen wiederholt, wie sie in den Beispielen einheitlich verwendet werden.

p_0 **PLA-Tabelle**
Sie ist *ad hoc* spezifiziert worden; insbesondere geschieht die Konvertierung in der Regel mit Hilfe der Rnd_Round-Methode.

c_0 **Kosten**
Mit c_0 sind die PLA-Kosten der PLA-Tabelle p_0 gemeint, die in der Regel durch den Logik-Minimierer ESPRESSO [Bra*82b, Bra*84] in dessen *Normal*-Modus ermittelt worden sind. Sie dienen im allgemeinen als Bezugsgröße, wenn bei modifizierten Spezifizierungen die Reduzierung der Kosten diskutiert wird.

r_0 **Kosten-Reduzierung**
Berechnungsgröße, die die PLA-Kosten c einer PLA-Tabelle p in Bezug zu den Kosten c_0 von p_0 bringt (vgl. Gleichung 6.13, S. 120).

N **Anzahl der Einträge**

M **Maximalwert**
Maximum der Einträge bzw. Funktionswerte.

n **Anzahl der Eingangsvariablen des PLAs**
Sie ist die Bitbreite der Indizes mit $n := \mathsf{Bits}(N)$; oft, aber nicht zwingend, gilt $N := 2^n$.

m **Anzahl der Ausgangsvariablen des PLAs**
Sie ist die Bitbreite für Einträge mit $m := \mathsf{Bits}(M)$; oft gilt $M := 2^m - 1$.

6.3.2.2 Definition der Beispiele

`sine` **Sinus (1. Viertelwelle):**
Dieses Beispiel ist dem bereits ausführlich diskutierten *Sinus*-Beispiel angelehnt. Nähere Erläuterungen zum Hintergrund sind daher dem Abschnitt 3.1.5 (S. 39f) zu entnehmen. Die PLA-Tabelle hat ein 7×10-Format (vgl. Abschnitt 1.2.2.3 (S. 7) und umfaßt 2^7 Einträge à 10-Bit, d. h. im Gegensatz zu dem bisherigen *Sinus*-Beispiel mit $N = 100$ und $M = 1000$ handelt es sich hier um $N := 128$ Einträge im Wertebereich von 0 bis $M := 2^{10} - 1 = 1023$. Die Berechung der reellen Werte geschieht gemäß folgender Gleichung:

$$f(i) := \sin\left(\frac{\pi}{2} \cdot \frac{i}{N-1}\right) \quad \text{mit} \quad x \in [0, \tfrac{\pi}{2}] \tag{6.14}$$

Die Logik-Minimierung mit ESPRESSO ergibt eine Termanzahl von $\ell := 118$, so daß sich damit die Kosten der Referenz zu $c_0 := \ell = 118$ ergeben.

`root` **Quadratwurzel:**
Dieses Beispiel stammt aus dem Softwarepaket zum Logik-Minimierer ESPRESSO [Bra*84, Tabelle 7.1 (S. 165)] und stellt die Berechnung der Quadratwurzel nach folgender Gleichung dar:

$$f(x) := \sqrt{x} \quad \text{mit} \quad x \geq 0 \tag{6.15}$$

Die Werte liegen dem Softwarepaket ESPRESSO-II C (*C*-Version #2.3) [Esp-88] als *ad hoc* spezifizierte PLA-Tabelle im 8×5-Format vor. Als Argumente sind also 8-Bit kodierte Werte zugelassen, d. h. $n := 8$ und $N := 2^n$. Mit den Argumenten aus $x \in \mathbf{I}_N := \{0, \ldots, 255\}$ erreicht die monoton sehr langsam wachsende Wurzelfunktion f maximal einen Funktionswert von $f(255) := 15.9687\ldots$ Bei einer Rundung dieses Maximalwertes ergibt sich $\mathsf{Rnd\_Round}\big(f(255)\big) := 16$ als ganzzahliges Maximum, so daß wegen $m := \mathsf{Bits}(16) = 5$ eine 5-Bit Kodierung erforderlich ist; hier wäre eine Skalierung der Originalfunktion empfehlenswert, um in einer nur 4-Bit umfassenden Darstellung zu kodieren. Die Logik-Minimierung mit ESPRESSO[7] ergibt eine Termanzahl von $\ell := 58$, so daß sich damit die Kosten der Referenz zu $c_0 := \ell = 58$ ergeben.

`expo` **Exponentialfunktion:**
Mit der transzendenten Exponentialfunktion liegt hier, wie im Falle der trigonometrischen Sinusfunktion, ebenso ein typisches Beispiel für die Realisierung mit einem PLA vor, da ihre Auswertung sowohl bei der Laufzeit als auch bezüglich des Speicherplatzes zu den komplexeren Problemen zählt. In diesem Beispiel mit $n := 6$ wurden die $N := 64 = 2^n$ Funktionswerte durch Auswertung der folgenden Definition gewonnen:

$$f(x) := \exp\left(\frac{x}{N} \cdot \log(a)\right) \quad \text{mit} \quad x \geq 0 \quad \text{und} \quad a \in I\!N \tag{6.16}$$

Da die Exponentialfunktion sehr stark wächst, ist die Einflußnahme auf das gewünschte Maximum bereits im Argument sinnvoll: Mit $a \in I\!N$ kann das gewünschte Maximum

[7] In [Bra*84, Tabelle 7.1, S. 165] werden für dieses Beispiel 57 Terme angegeben. Dieses Ergebnis wurde im sogenannten *Strong*-Modus von ESPRESSO ermittelt; er stellt die effektivste, zugleich aber auch die Laufzeitintensivste Betriebsart des Minimierers dar. Näheres dazu in Anhang D im Abschnitt D.3.1 (S. 175).

festgelegt werden. Für die Binärdarstellung wurde $m := 16$ gewählt, so daß sich als Maximum $a := 65535 = 2^{16} - 1$ ergibt. Die PLA-Tabelle hat damit ein 6×16-Format. Als PLA-Kosten c_0 ergibt eine ESPRESSO-Minimierung für dieses Beispiel eine Termanzahl von $\ell := 58$.

`loga` **Logarithmus:**
Die Logarithmusfunktion ist die Umkehrfunktion der Exponentialfunktion; ähnlich zur Wurzelfunktion (siehe Beispiel `root`) wächst sie monoton nur sehr langsam, so daß sich der Wertevorrat ebenfalls auf nur wenige Werte beschränkt. Damit eine bessere Auflösung in der Darstellung möglich wird, soll im Gegensatz zum Beispiel `root` an dieser Stelle eine Skalierung durch Multiplikation mit einem Faktor vorgenommen werden.

$$f(x) := a \cdot \log_e(x) \quad \text{mit} \quad x > 0 \quad \text{und} \quad a \in I\!N \tag{6.17}$$

Für $N := 128$ Einträge wird die Funktion f mit einem Faktor $a := 13$ evaluiert. Als Maximum der monoton wachsenden Funktion ergibt sich wegen $2^6 = 64 > a \cdot \log(N-1) = 62.974\ldots$ eine 6-Bit Binärdarstellung. Mit $n := 7 = \mathsf{Bits}(N-1)$ folgt damit für die PLA-Tabelle ein 7×6-Format. Die Logik-Minimierung mit ESPRESSO ergibt eine Termanzahl von $\ell := 59$, so daß sich damit die Kosten der Referenz zu $c_0 := \ell = 59$ ergeben.

`hbol` **Hyperbel:**
Die Hyperbelfunktion (engl. *hyperbolic function*) stellt auf einfachste Weise die bei Reihenentwicklungen auftretende Problematik der Näherungswerten dar, da es sich nur um *eine* Division handelt. In diesem Beispiel werden $N := 64$ Quotienten nach folgender Definition berechnet:

$$f(x) := a \cdot \frac{1}{x} \quad \text{mit} \quad x > 0,\ a \in I\!N \tag{6.18}$$

Der Skalierungsfaktor sei durch $a := 100$ gewählt, so daß sich mit $n := \mathsf{Bits}(N-1) = 6$ und $m := \mathsf{Bits}(a) = 7$ für die PLA-Tabelle ein 6×7-Format ergibt.

6.3.2.3 Optimierung bei den Beispielen

Bei den Beispielen geschieht die Optimierung gemäß der Kosten-orientierten Strategie (vgl. Abschnitt 6.2.6.2, S. 117), d. h. zunächst wird die Menge der Lösungskandidaten anhand der jeweiligen Abweichung ermittelt. Anschließend werden davon die Lösungsfavoriten gemäß ihrer Kosten bzw. der jeweils erzielten Reduzierung ausgesondert.

PLA-Beispiel			**Lösungsfavoriten bei Rundungs-Toleranz**						
Name	Format $n \times m$	Terme c_0	Top-Five Heuristiken Methode	Matches	Kosten c	r_0	Abweichung $d(p_0)$ M_{abs}	A_{rel}	B_{ent}
`sine`	7×10	118	Sel_Mean	126	109	8%	1	0%	7
			Sel_Min1	69	110	7%	1	0%	7
			Rnd_Ceil	128	111	6%	1	0%	7
			Sel_High	126	114	3%	1	0%	7
			Rnd_Floor	128	116	2%	1	0%	6
`root`	8×5	58	Sel_Min1	128	35	40%	1	5%	5
			Rnd_Floor	256	38	34%	1	5%	5
			Sel_Low	241	38	34%	1	5%	5
			Sel_Mean	240	38	34%	1	5%	5
			Sim_Bin(5)	127	42	28%	1	1%	5
`expo`	6×16	58	Sel_Mean	62	56	3%	1	5%	4
			Sel_Min1	30	56	3%	1	5%	4
			Rnd_Floor	64	57	2%	1	3%	5
			Sim_Num(1)	6	57	2%	1	1%	1
			Sel_Low	62	57	2%	1	3%	5
`loga`	7×6	59	Sel_Min1	60	45	24%	1	1%	6
			Sel_Mean	127	46	22%	1	1%	6
			Rnd_Ceil	127	56	5%	1	1%	6
			Sel_High	127	57	3%	1	1%	6
`hbol`	6×7	24	Sel_Min1	29	18	25%	1	8%	4
			Sel_Mean	56	19	21%	1	8%	4
			Rnd_Ceil	63	22	8%	1	13%	4
			Sim_Num(1)	28	22	8%	1	1%	3
			Sel_High	56	22	8%	1	13%	4

Tabelle 6.8: Optimierungsergebnisse bezüglich einer Toleranz durch Ab- und Aufrundwert

Im Gegensatz zum *Sinus*-Beispiel, das in Abschnitt 6.3.1.3 (S. 119ff) sehr ausführlich diskutiert wurde, seien hier drei Toleranzbereiche für die Generierung von Intervallen festgelegt: Neben einer Absolut- und Relativ-Toleranz von ± 2 bzw. $\pm 5\%$ kommt hier auch noch diejenige aus Ab- und Aufrundwert hinzu, so daß die Intervalle mit den drei Methoden Int_UpDn, Int_Abs(2,2) und Int_Rel(5%,5%) erzeugt werden können. Die Sel-Methoden stehen in Kombination mit der jeweils Intervall-erzeugenden Int-Methode, d. h. werden die Intervalle beispielsweise mit Int_Abs generiert, so wird die Sel-Methode auch nur darauf angewendet, und nicht zusätzlich auf die anderen Int-Modi Int_UpDn und Int_Rel.

6.3.2.4 Darstellung der Ergebnisse

In den drei Tabellen 6.8, 6.9 und 6.10 (S. 130ff) sind die Ergebnisse wie folgt zusammengestellt: In den Titelzeilen befindet sich jeweils die Vorgabe der Toleranz, die zur Auswahl des besten Lösungskandidaten herangezogen wurde. Auf der linken Seite der Tabellen ist zunächst das Beispiel durch seinen *Namen* und das *Format* der PLA-Tabelle p_0 identifiziert. Für die Referenz, die aus der *ad hoc* spezifizierten PLA-Tabelle p_0 abgeleitet wird, ist ferner die Anzahl der *Terme* als Kosten c_0 angegeben.

PLA-Beispiel			**Lösungsfavorit bei Absolut-Toleranz: ±2**						
Name	Format	Terme	Top-Five Heuristiken		Kosten		Abweichung $d(p_0)$		
	$n \times m$	c_0	Methode	Matches	c	r_0	M_{abs}	A_{rel}	B_{ent}
`sine`	7×10	118	Sel_Min1	125	104	12%	3	0%	8
			Sel_Mean	128	109	8%	1	0%	7
			Rnd_Ceil	128	114	3%	1	0%	7
			Rnd_Floor	128	117	1%	1	0%	6
			Sel_High	128	117	1%	3	1%	8
`root`	8×5	58	Sel_Min1	255	19	67%	3	14%	5
			Rnd_Floor	256	38	34%	1	5%	5
			Sel_Low	256	39	33%	3	28%	4
			Sel_Mean	256	39	33%	1	6%	5
			Sim_Bin(5)	127	42	28%	1	1%	5
`expo`	6×16	58	Sel_Mean	64	56	3%	1	6%	4
			Sim_Num(3)	9	56	3%	2	1%	1
			Sel_Low	64	56	3%	3	18%	7
			Sel_Min1	63	56	3%	3	19%	7
			Rnd_Floor	64	57	2%	1	3%	5
`loga`	7×6	59	Sel_Min1	117	26	56%	3	3%	6
			Sim_Num(3)	62	42	29%	3	0%	5
			Sim_Num(2)	61	43	27%	2	0%	5
			Sim_Num(1)	59	44	25%	1	0%	5
			Sel_Mean	128	50	15%	1	1%	6
`hbol`	6×7	24	Sel_Min1	63	13	46%	3	48%	4
			Sel_Mean	64	19	21%	1	8%	4
			Sim_Num(3)	29	21	13%	3	1%	5
			Rnd_Ceil	63	22	8%	1	13%	4
			Sim_Num(1)	28	22	8%	1	1%	3

Tabelle 6.9: Optimierungsergebnisse bezüglich einer Absolut-Toleranz von ±2

Im rechten Bereich der Tabellen sind für jedes Beispiel die fünf besten Methoden (Top-Five Heuristiken) aufgeführt, wobei in der Spalte *Matches* die Anzahl der Einträge steht, die durch die Methode verändert werden konnten. Als nächstes folgt die Termanzahl der durch die jeweilige Methode modifizierten PLA-Tabelle, anschließend steht die Reduzierung bezogen auf die Kosten c_0. Zu beachten ist, daß die Abweichung $d(p_0)$ sich jeweils auf die PLA-Tabelle p_0 der Referenz bezieht; sie ist damit irrelevant für die Auswahl der Lösungsfavoriten, die ja entsprechend der jeweiligen Toleranz-Vorgabe geschieht, und dient nur dem eventuell ergänzend anstellbaren Vergleich mit der Referenz.

6.3.2.5 Interpretation der Ergebnisse

Bezüglich der verschiedenen Toleranzbereiche besteht die intuitive Vermutung, daß die größten Reduzierungseffekte bei der Relativ-Toleranz erzielt werden können, da sie bei großen Funktionswerten breitere Intervalle erzeugt als die anderen beiden Toleranzarten. Durch relative Toleranz existieren damit für jeden Funktionswert quantitativ mehr zulässige Werte, so daß insbesondere die Selektionsmethoden Sel effektiver Werte auswählen können.

In Tabelle 6.11 (S. 132) sind für jedes Beispiel jeweils die Methoden mit der größten Reduzierung im Überblick zusammengefaßt. Der beschriebene Trend bestätigt sich tatsächlich für die Beispiele `sine` und `expo`, die beide einen großen Wertebereich mit einem Maximum in der Größenordnung von 1E3 bzw. 1E4 aufweisen und insofern von diesem Effekt profitieren können. Bei den anderen Beispielen, die deutlich geringere Wertebereiche mit einem Maximum in der Größenordnung von 1E2 aufweisen, läßt sich dieser Trend dagegen nicht bestätigen.

PLA-Beispiel			Lösungsfavorit bei Relativ-Toleranz: ±5%						
Name	Format	Terme	Top-Five Heuristiken		Kosten		Abweichung $d(p_0)$		
	$n \times m$	c_0	Methode	Matches	c	r_0	M_{abs}	A_{rel}	B_{ent}
`sine`	7×10	118	Sel_Min1	126	55	53%	50	2%	9
			Sim_Num(11)	46	89	25%	11	0%	9
			Sim_Num(10)	40	95	19%	10	0%	7
			Sim_Num(8)	29	100	15%	8	0%	6
			Sim_Num(7)	25	101	14%	7	0%	6
`root`	8×5	58	Sel_Min1	228	33	43%	2	7%	5
			Rnd_Floor	256	38	34%	1	5%	5
			Sel_Low	255	41	29%	2	11%	5
			Sim_Bin(5)	127	42	28%	1	1%	5
			Sim_Bin(4)	126	46	21%	1	33%	4
`expo`	6×16	58	Sel_Min1	63	51	12%	2387	8%	11
			Rnd_Floor	64	57	2%	1	3%	5
			Psd_Dec(4,1)	6	57	2%	7	0%	3
			Psd_Dec(5,1)	4	57	2%	7	0%	3
			Psd_Dec(6,1)	2	57	2%	7	0%	3
`loga`	7×6	59	Sel_Min1	117	26	56%	4	3%	6
			Sim_Num(2)	61	43	27%	2	0%	5
			Sim_Num(1)	59	44	25%	1	0%	5
			Sim_Bin(2)	56	51	14%	1	3%	1
			Sel_High	127	51	14%	4	5%	5
`hbol`	6×7	24	Sel_Min1	49	15	38%	4	8%	4
			Sel_Low	64	16	33%	5	23%	5
			Sel_Mean	64	19	21%	1	4%	3
			Rnd_Ceil	63	22	8%	1	13%	4
			Sim_Num(1)	28	22	8%	1	1%	3

Tabelle 6.10: Optimierungsergebnisse bezüglich einer Relativ-Toleranz von ±5%

Neben den Betrachtungen, die sich auf die Toleranzbereiche konzentrieren, steht die Beurteilung der Reduzierungeffekte bei den einzelnen Methoden an, die im folgenden diskutiert wird:

- **Methode: Sel**
  Beim Betrachten der besten Lösungsfavoriten in Tabelle 6.11 (S. 132) fällt die permanente Dominanz der Sel-Methode auf: Der Modus Sel_Min1 erzielt, bis auf wenige Ausnahmen, die maximale Reduzierung der Kosten, wobei der Spitzenwert von $r_0 = 67\%$ beim Beispiel `root` unter einer Absolut-Toleranz von ±2 erreicht wird.

  Aber auch der Sel_Mean-Modus ist in den drei Tabellen 6.8, 6.9 und 6.10 (S. 130ff) häufig unter den besten fünf Methoden zu finden; dieses ist aufgrund symmetrischer Intervalle insofern überraschend, als das Sel_Mean die arithmetischen Mittelwerte in den Intervallen

Beispiel		Toleranzbereich					
Name	Format	Rundungs-Toleranz		Absolut-Toleranz: ±2		Relativ-Toleranz: ±5%	
`sine`	7×10	Sel_Mean	$r_0 := 8\%$	Sel_Min1	$r_0 := 12\%$	Sel_Min1	$r_0 := 53\%$
`root`	8×5	Sel_Min1	$r_0 := 40\%$	Sel_Min1	$r_0 := 67\%$	Sel_Min1	$r_0 := 43\%$
`expo`	6×16	Sel_Mean	$r_0 := 3\%$	Sel_Mean	$r_0 := 3\%$	Sel_Min1	$r_0 := 12\%$
`loga`	7×6	Sel_Min1	$r_0 := 24\%$	Sel_Min1	$r_0 := 56\%$	Sel_Min1	$r_0 := 56\%$
`hbol`	6×7	Sel_Min1	$r_0 := 25\%$	Sel_Min1	$r_0 := 46\%$	Sel_Min1	$r_0 := 38\%$

Tabelle 6.11: Beste Lösungsfavoriten der Beispiele je nach Toleranzbereich

selektiert und damit näher an den Werten der Referenz liegt als Sel_Min1. Das läßt sich beim sine-Beispiel in Tabelle 6.9 (S. 131) erkennen, bei dem Sel_Mean mit einer etwas geringeren Reduzierung nach Sel_Min1 an zweiter Stelle steht und dabei gleichzeitig weniger stark abweicht: Bezogen auf die Werte der Referenz, die durch Rundung mit Rnd_Round aus den reellen Werten gewonnen wurden, weicht Sel_Mean bei der maximalen Absolut-Abweichung M_{abs} in einem geringeren Maße ab als Sel_Min1 und liegt damit durchweg näher an den gerundeten Funktionswerten.

- **Methode:** Sim
  Beachtlich ist auch das Abschneiden der Sim-Methode: Sie taucht in den drei Tabellen 6.8, 6.9 und 6.10 (S. 130ff) an mehreren Stellen vereinzelt auf, dominiert aber insbesondere beim loga-Beispiel unter Absolut- bzw. Relativ-Toleranz und beim sine-Beispiel unter Relativ-Toleranz. Sie erzielt durchweg eine immerhin so hohe Reduzierung, daß sie bezüglich der Effektivität an zweiter Position nach der Sel-Methode einzuordnen ist. Durch dieses Abschneiden der Sim-Methode erscheint die vergleichsweise aufwendige Definition im Rückblick lohnend.

- **Methode:** Rnd
  Bemerkenswert ist das Auftreten der Rundungsmodi Rnd_Floor und Rnd_Ceil, die nicht selten in den drei Tabellen 6.8, 6.9 und 6.10 (S. 130ff) unter den besten fünf Heuristiken vorkommen; sie erlangen zwar nicht Spitzenpositionen, weichen dafür aber auch nicht stark von der Referenz ab, die jeweils durch Rnd_Round gewonnen wurde. Das gute Abschneiden von Rnd_Floor beim root-Beispiel ist auf den geringen Wertebereich mit einem Maximalwert kleiner als $2^5 - 1 = 31$ (tatsächlich aber nur bis 15) zu begründen.

- **Methoden:** P2, Psd und Hyp
  Die Resultate dieser Methoden sind unter der Einschränkung, daß die Abweichung innerhalb der Toleranzbereiche liegen muß, nicht relevant. Während für P2 und auch für Sel_NP2 bei großer Abweichung die Kosten durchweg sehr klein sind, ist dies bei für Hyp nur vereinzelt der Fall. Die Psd-Methode tritt nur mit geringem Stellenwert beim expo-Beispiel unter 5%-iger Relativ-Toleranz auf und erreicht dort nur geringe Werte für die Reduzierung, die auch von Rnd_Floor erzielt werden.

Bezüglich der Beispiele ist der kleine Wertebereich bei root anzumahnen; dieses Beispiel ist unverändert aus dem Softwarepaket zu Espresso übernommen worden. Gemäß der *Modifikation des Maximum M der Einträge* (S. 110) wäre eine Skalierung hier zu empfehlen. In der unveränderten Situation verursacht jegliche Modifikation, ob geringfügig oder umfassend, in einem derart kleinen Wertebereich durchweg beträchtliche Abweichungen. Es kommen dementsprechend auch die hohen Werte bei der Reduzierung zustande. Außerdem weist root mit 58 Termen relativ hohe Kosten bei der Referenz auf, die leicht durch die Heuristiken um etwa ein Drittel reduziert werden können. Die Referenzkosten von 58 Termen beim root-Beispiel sollten im Verhältnis zur selben Termanzahl beim expo-Beispiel betrachtet werden, bei dem aber mit einem Maximum von $2^{16} - 1 = 65535$ einen viel größerer Wertebereich vorliegt; den 16 Logikfunktionen dort stehen nur 5 beim root-Beispiel gegenüber.

6.3.3 Bewertung der Evaluation

Die gezeigte Evaluation der Methoden an den Beispielen läßt insgesamt den Schluß zu, daß unter leichten Modifikationen an der Spezifizierung einer PLA-Tabelle teilweise erhebliche Reduzierungen bei den PLA-Kosten möglich sind. Dabei zeichnet sich insbesondere aus, wenn das Ausmaß der Unschärfe, die im Rahmen einer Spezifikation unvermeidbar auftritt, in Gestalt eines Toleranzbereichs angegeben werden kann. Auf diese Weise wird für jeden Eintrag der PLA-Tabelle eine größere Anzahl von zulässigen Werten erreicht.

Mit diesen neu hinzu gewonnenen Freiheiten sind vor allem heuristisch arbeitende Selektionsmethoden in der Lage, enorme Reduzierungen der Kosten zu erzielen. Aber auch ohne die Möglichkeit, Toleranzbereiche für die Unschärfe angeben zu können, sind effektive Methoden gefunden worden, die sich mit der Ähnlichkeit von Einträgen und deren mögliche Angleichung widmen; sie unterstützen damit die Effektivität der im Rahmen eines PLA-Entwurfs obligatorischen Logik-Minimierung.

Mit Hilfe der diskutierten Optimierungen an Spezifizierungen von PLA-Tabellen sind Lösungen erzielbar, die insbesondere von der individuellen Beurteilung der Unschärfe in den vorliegenden Daten abhängt. Dieser Umstand läßt allerdings eine Bewertung des gezeigten Vorgehens in einem übergeordneten Kontext nicht zu: Insbesondere ist daher die Nähe der erzielbaren Lösungen zu einem Optimum nicht zu fassen. Es bedürfte dafür der Festlegung der zulässigen Abweichungen, die eine Lösung aufweisen darf; dies läuft aber konträr zur Erkenntnis der Unschärfe, die zwar unvermeidbar vorhanden ist, jedoch nicht exakt zu quantifizieren ist.

Kapitel 7

Implementierung eines Werkzeugs

Die Aufgabenstellung zu dieser Diplomarbeit beinhaltete neben den zu erarbeitenden Heuristiken und Methoden auch die Erstellung eines Werkzeugs für die Entwurfspraxis von PLAs, da die hohen Werte bei der statischen Stromaufnahme in konkreten Entwurfsprojekten zu massiven Schwierigkeiten führten bzw. die Realisierung mit Hilfe von PLA-Strukturen gänzlich unmöglich machte.

Im folgenden wird zunächst ein Anforderungsprofil diskutiert, das sich an den Bedürfnissen der Entwurfspraxis orientiert. Im weiteren Verlauf dieses Kapitels wird dann das entstandene System HOPE und seine Entstehung erläutert.

7.1 Konzeption des Werkzeugs

7.1.1 Anforderungsprofil

Wie *Rauscher* in [Rau-95] bemerkt, sollte das Werkzeug ein Ergebnis liefern, daß etwa folgender Aussage entspricht: *Wenn die Spezifizierung in der mit Hilfe des Werkzeugs empfohlenen Weise modifiziert wird, wäre eine Reduzierung der Kosten um r% gegenüber der Originalspezifikation möglich.* Mit diesem Gedanken als Leitmotiv lassen sich bezüglich des Werkzeugs drei Grundanforderungen ableiten: Das Werkzeug soll zur *Erfassung, Modifikation* und *Analyse* von PLA-bezogenen Daten dienen. Dabei handelt es sich um PLA-Tabellen oder um Daten, aus denen eine PLA-Tabelle abgeleitet werden kann. Im einzelnen treten folgende Aspekte hervor:

- **Erfassung:**
  Das Werkzeug dient zunächst zur Erfassung von Daten, aus denen eine PLA-Tabelle abgeleitet werden kann. Beispielsweise kommen hierbei Dateien mit reellen Werten in Frage; aber auch Dateien, die bereits eine PLA-Tabelle beinhalten, sollten als Eingabe möglich sein.

- **Modifikation:**
  Zentraler Aspekt des Werkzeugs ist die Modifikation der Spezifizierung, d. h. die Spezifikation einer PLA-Tabellen ist zu manipulieren. Zu diesem Zweck sollte das Werkzeug eine Sammlung von Methoden anbieten, mit deren Hilfe derartige Modifikationen vorgenommen werden können.

- **Analyse:**
  Die Analyse der PLA-Tabellen beinhaltet zunächst das Ermitteln der PLA-Kosten in Gestalt der Termanzahl. Dies sollte nach Möglichkeit durch eine effiziente Logik-Minimierung wie beispielsweise Espresso [Bra*82b, Bra*84] erfolgen. Beim Vergleich mehrerer PLA-Tabellen soll die eventuell bewirkte Reduzierung bestimmt werden können.

7.1.2 Datenkonzept

Da in dem Vorgang der Optimierung ggf. sehr viele Varianten für eine PLA-Tabelle entstehen können (vgl. Abschnitt 6.2.4.2, S. 109), sollte der Anwender des Werkzeugs im Überblick unterstützt werden. Es bietet sich daher eine Strukturierung der Daten an, die sich an die im letzten Kapitel 6 vorgestellten Datenkapselung mit Hilfe der PLA-Frames anlehnt (vgl. Abschnitt 6.1.1.3, S. 104).

Da die beim PLA-Frame als Ganzes in einer Komponente erfaßte Spezifizierung **S** (vgl. Gleichung 6.1, S. 104) modifiziert werden soll, ist sie in elementare Operationen zu zerlegen. Das bedeutet, daß die durch eine Operation erzeugten Zwischenresultate für jeden Schritt innerhalb der Spezifizierung erfaßt werden müssen. Um Redundanz in der Datenhaltung zu verhindern, sollten gleiche Operationen auf identischen Zwischenresultaten dadurch vermieden werden, daß bei modifizierten Spezifizierungen identische Schritte mehrfach verwendet werden können.

Für jeden konkreten Fall einer zu spezifizierenden PLA-Tabelle sollten alle im Rahmen der Optimierung mit dem Werkzeug bewirkten Schritte zusammengefaßt und archivierbar gemacht werden können. Dadurch können unnötige Wiederholungen im Vorgehen vermieden und auf bisher erreichte Ergebnisse leicht zurückgegriffen werden.

7.1.3 Benutzerschnittstelle

Da jede konkrete PLA-Spezifikation mit der inhärenten Unschärfe individuelle Merkmale in sich trägt, ist auch die Optimierung in einer indviduellen Weise vorzunehmen. Das Werkzeug sollte zu diesem Zweck eine *interaktive Benutzerschnittstelle* aufweisen, die im Dialog mit dem Benutzer Mittel zur Manipulation des Geschehens anbietet. Die Interaktivität begründet sich nicht zuletzt aus dem relativ hohen menschlichen Anteil bei der Konsensfindung im Rahmen der diskutierten Optimierungsstrategien.

Die Präsentation des in Abschnitt 5.3.3 (S. 102f) vorgestellten Instrumentariums der Methoden sollte in einer Baukasten-ähnlichen Weise geschehen, d. h. der Benutzer soll in freier Auswahl Methoden einsetzen und miteinander kombinieren können. Dabei sollte das Werkzeug durch eine strukturierte Darbeitung den Überblick über die Methode erleichtern und deren sinnvolle Benutzung durch entsprechende Kontrollmechansimen, wie z. B. eine Plausibilitätsprüfung der Eingaben, unterstützen. Um eine maximale Transparenz in der Benutzerführung zu erreichen, eignen sich u. a. folgende Gestaltungsmittel:

- **Menüs:** Durch Menüs können die ausführbaren Aktionen dem Benutzer transparent dargeboten werden.

- **Dialoge:** Mit Hilfe von Dialogen sind Interaktionen einheitlich gestaltbar, so daß der Benutzer nach Ablauf einer kurzen Einarbeitungsphase aufgrund wiederkehrender Abläufe Sicherheit in der Bedienung des Werkzeugs erlangen kann.

- **Meldungen:** Durch Meldungen können dem Benutzer Informationen zum Arbeitsablauf bzw. über den Datenbestand gegeben werden; es können drei drei Meldungstypen unterschieden werden:

  1. Ausgaben
  2. Bestätigungen mit minimalem Ja/Nein-Dialog
  3. Fehlermeldungen

Als Hilfestellung bei der Dokumentation der Ergebnisse ist die Protokollierung der Aktivitäten und Meldungen sinnvoll. Ferner sollte das Werkzeug eine möglichst tolerante Fehlerbehandlung haben, um maximale Freiheiten für die Benutzung zu offerieren.

7.1.4 Integrationsanforderungen

Zur Integration des Werkzeugs in den Vorgang des PLA-Entwurfs sind noch einige weitere Aspekte zu berücksichtigen, die nachfolgend diskutiert werden:

- **Standardisierung:**
  Speziell im Analyseteil des Werkzeugs sollte ein allgemein anerkannter Logik-Minimierer integriert sein; empfehlenswert ist hierfür der Logik-Minimierer ESPRESSO von der *University of California, Berkeley* [Bra*-82b, Bra*-84], der als *de facto* Standard gilt.

- **Datenaustausch:**
  Wünschenswert wäre die Unterstützung von gängigen und wenig komplexen Dateiformaten. Derartige Dateien sollten in gängigen Editierwerkzeugen manipulierbar sein. Ebenso ist die Ausgabe von (PLA-) Tabellen in äquivalenten Datei-Formaten zu ermöglichen.

- **Universalität:**
  Das Entwurfswerkzeug soll unter einer Vielzahl von verschiedenen Betriebsbedingungen einsetzbar sein. Dafür ist die Unabhängigkeit von speziellen Entwurfsumgebungen und Rechner-Plattformen bzw. Betriebssystemen wichtig.

- **Portabilität:**
  Damit einhergehend ist der Aspekt der Portabilität zu sehen, d. h. die Implementierung des Werkzeugs sollte keine systemspezifisch zu realisierenden Funktionen aufweisen, sondern möglichst nur solche Dienste anbieten, die leicht auf unterschiedlichen Zielsystemen, also insbesondere auch auf Personal-Computer, zu implementieren sind. Zur Programmierung des Werkzeugs bietet sich daher beispielsweise die Programmiersprache *C* an [Ker*-88, Bar-89].

- **Tabellengenerierung:**
  Für den Bereich der Datenerfassung sollte neben der Möglichkeit des Daten-Imports auch die Möglichkeit bestehen, Daten entsprechend numerischer Standardfunktionen zu generieren. Wünschenswert wäre das Manipulieren von Generierungsparametern.

- **Geschwindigkeit:**
  Bezüglich des strategischen Vorgehens bei der Optimierung, wurde in Abschnitt 6.2.6.3 (S. 117) auf die Beteiligung von Auftraggeber und Entwerfer an der Auslese aus der Menge der Lösungsfavoriten eingegangen. Da für einen Benutzer die Akzeptanz eines Werkzeugs bei steigenden Anwortzeiten rasch abnimmt, sollte das Werkzeug möglichst schnell arbeiten.

7.2 Implementierung von HOPE

In diesem Abschnitt wird die Implementierung eines im Rahmen dieser Diplomarbeit erstellten Systems mit dem Namen Hope als Entwurfswerkzeug erläutert. Die Namensgebung leitet sich aus dem generellen Anliegen ab, nämlich *Heuristic Optimization on a PLA-Specification to Minimize Expense*. Die nachfolgende Beschreibung des Systems widmet sich zunächst den Datenstrukturen und danach der in den Modulen implementierten Funktionalität; da die Aufgabenstellung zu dieser Arbeit jedoch nicht die Implementierung eines Werkzeugs zum Schwerpunkt hatte, erfolgt die Beschreibung nur oberflächlich, um den vorgegebenen Rahmen nicht zu sprengen.

7.2.1 Datenstrukturen in HOPE

In Abschnitt 7.1.2 (S. 136) ist bezüglich einer Strukturierung der anfallenden Daten auf die PLA-Frames hingewiesen worden (vgl. Abschnitt 6.1.1.3, S. 104). Ein PLA-Frame $\mathbf{F}_j$ kapselt PLA-relevante Daten und faßt dabei die Spezifizierung $\mathbf{S}_j$ komplett als eine Komponente auf. Bei der Implementierung des Werkzeugs ist hier eine Verfeinerung notwendig.

7.2.1.1 Operationen und Resultate

Eine Spezifizierung wird in der Systematik von Hope als eine Sequenz aus elementaren Operationen betrachtet. Nach jeder Operation entsteht ein Resultat, das durch die vorausgegangene Sequenz von durchgeführten Operationen bestimmt ist. Für die Reproduzierbarkeit der Resultate genügt es daher, lediglich die Operationen zu erfassen. In der Systematik von Hope ist dafür das sogenannte `Object` definiert, welches in Tabelle 7.1 (S. 138) mit seiner internen Struktur aufgeführt ist.

Hope-Struktur `Object`			
Komponente	Bedeutung	Komponente	Bedeutung
`int id`	Identification Number	`int relid`	Id Relate-Object
`int flags`	Type and Attribute	`int idev`	Int-Deviation
`int orgid`	Id Origin-Object	`long ldev`	Long-Deviation
`int tool`	Tool-Identifier	`char *name`	Filename (optional)
`int ipar1`	Int-Parameter-1	`int terms`	Costs: Terms
`int ipar2`	Int-Parameter-2	`int lits`	Costs: Literals
`int ipar3`	Int-Parameter-3	`int ones`	Count: # of 1-Bits
`long lpar`	Long-Parameter	`int dcs`	Count: # of DC-Bits

Tabelle 7.1: Erfassung einer Operation in Hope als Strukturelement `Object`

Die Datenstruktur `Object` wird zunächst durch die Komponente `id` identifiziert. Neben der Kodierung einiger Eigenschaften des Objekts in den `flags`, nimmt die Komponente `orgid` die Verkettung mit anderen Objekten vor, wobei in `orgid` die Identifikationsnummer `id` des Vorgängerobjekts abgelegt ist; damit ist der Ursprung (engl. *Origin*) der Objektdaten bei jedem Objekt bekannt. Zum (Re-)Produzieren der Resultate ist auf das Vorgängerobjekt die durch `tool` bezeichnete Operation anzuwenden, wobei eventuell benötigte Parameter in den Komponenten `ipar1`, `ipar2`, `ipar3` und `lpar` abgelegt sind.

Mit den Informationen zu dem Vorgängerobjekt und der Operation lassen sich die resultierenden Daten jederzeit reproduzieren; dabei handelt es sich entweder[1] um *numerische* Tabellen mit reellen Einträgen oder um *binäre* Tabellen. Entsprechend wird dies als Typ des Objektes in den `flags` vermerkt. Merkmale von *binären* Tabellen können in den weiteren Komponenten `terms`, `lits`, `ones`, `dcs` abgelegt sein. Insbesondere ist dies für die Anzahl der Terme und Literale von Vorteil, da deren Bestimmung hohe Laufzeiten verursachen kann. Die beiden Merkmale `idev` und `ldev` nehmen Angaben zu Abweichungen zwischen der *binären* Tabelle des aktuellen Objektes in Bezug zu den Daten des durch `relid` angegebenen Objektes auf. Schließlich enthält die Komponente `name` einen optionalen Bezeichner für das Objekt, der erforderlich wird, wenn das Objekt einer externen Datei zugeordnet wird.

7.2.1.2 Spezifizierungen in HOPE

Anstatt in Hope sämtliche Resultate ständig vorzuhalten, werden in den Objekten nur die *Operationen* und die *Merkmale* der resultieren Daten erfaßt. Durch die Verkettung von Objekten, also Instanzen der Datenstruktur `Object`, entstehen komplette Spezifizierungen, so daß anhand der zurückverfolgbaren Sequenz der Operationen die PLA-Tabelle dynamisch reproduziert werden kann. Die Verkettung mehrerer Objekte bewirkt eine Baum-ähnliche Struktur. In Abbildung 7.1 (S. 139) ist dies zur Veranschaulichung dargestellt.

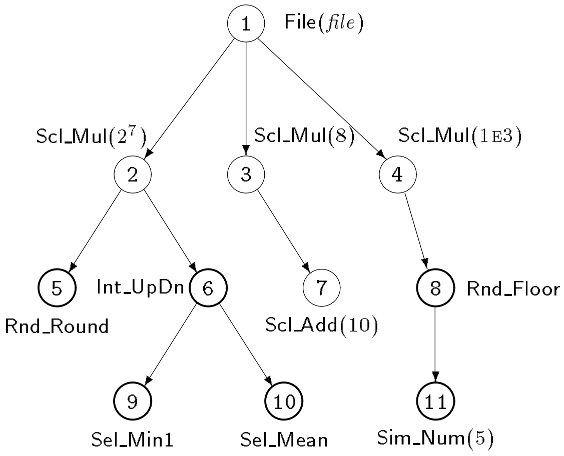

Abbildung 7.1: Baum-Struktur eines `Sets` durch `Object`-Verkettung in Hope

Die Verkettung von Objekten beginnt beim Anfangsobjekt, durch das beispielsweise extern erzeugte Werte als Datei nach Hope importiert werden. Durch Operationen können nun schrittweise Nachfolgeobjekte erzeugt werden, wobei für jede individuelle Kombination aus einer konkreten *Operation* und einem bestimmten *Objekt* jeweils ein neues Nachfolgeobjekt erzeugt wird. Gemäß der Unterscheidung von *numerischen* und *binären* Tabellen, existieren zwei Typen von

[1]Gemäß der beiden Trennlinien zwischen den drei Schichten in dem Modell aus Abschnitt 4.1.4 (S. 60) existieren grundsätzlich zwei darzustellende Wertebereich, nämlich $\mathbb{R}^+$ oberhalb und $\mathbb{N}$ unterhalb der Konversions-Schicht. Dementsprechend existieren in der Systematik von Hope diese zwei Typen für Tabellen.

Objekten, nämlich zum einen die numerischen Objekte bzw. *Numerik-Objekte* und zum anderen die binären Objekte bzw. die sogenannten *Binär-Objekte*. Dieses Typ-Merkmal ist in der `flags`-Komponente eines `Objects` kodiert. In Abbildung 7.1 (S. 139) sind die Numerik-Objekte dünn und die Binär-Objekte dick umrandet. Beim Objekt `6` handelt es sich aufgrund einer speziellen Erweiterung der Binärkodierung für Intervall-Einträge in der Systematik von Hope dennoch um ein Binär-Objekt, bei dem zum Typ ein entsprechendes Attribut hinzugefügt wird.

Die Verkettung von Objekten endet in der Regel bei einem Binär-Objekt, d. h. die resultierenden Daten stellen eine *binäre* Tabelle dar, die eine PLA-Tabelle repräsentieren kann. Innerhalb der Verkettung sind die aus Kapitel 5 bekannten Methoden sowie in ähnlicher Weise bezeichnete Operationen angedeutet. So deuten Scl_Mul(8) und Scl_Add(10) Schritte einer linearen Skalierung aller Tabellenwerte an, wobei zunächst mit dem Faktor 8 multipliziert und später mit dem Summanden 10 addiert wird; näheres dazu später im Abschnitt 7.2.3 (S. 143f).

Dadurch, daß für jedes Binär-Objekt nur *genau ein* Vorgängerobjekt existiert, ist die Entstehung jeder PLA-Tabelle, und damit die Spezifizierung, durch Zurückverfolgen der Objekt-Verkettung eindeutig bestimmbar. Die in Abbildung 7.1 (S. 139) dargestellte Baumstruktur veranschaulicht fünf solcher zurückverfolgbaren Spezifizierungen, die im folgenden zusammengestellt werden:

1. Objektkette `1-2-5`: File(*file*) → Scl_Mul(2^7) → Rnd_Round.
2. Objektkette `1-2-6-9`: File(*file*) → Scl_Mul(2^7) → Int_UpDn → Sel_Min1.
3. Objektkette `1-2-6-10`: File(*file*) → Scl_Mul(2^7) → Int_UpDn → Sel_Mean.
4. Objektkette `1-3-7`: File(*file*) → Scl_Mul(8) → Scl_Add(10).
5. Objektkette `1-4-8-11`: File(*file*) → Scl_Mul(1E3) → Rnd_Floor → Sim_Num(5).

Offenbar wurde der mit der 4. Objektkette eingeschlagene Weg nicht vollständig bis zu einem Binär-Objekt durchlaufen, sondern schon vorher beendet, da möglicherweise die beabsichtigte Kombination der verketteten Operationen verworfen wurde.

7.2.1.3 Zusammenfassung von HOPE-Objekten

Die Gesamtheit aller Objekte in Hope wird als Menge in einem sogenannten `Set` zusammengefaßt, d. h. für jede neue Aufgabenstellung einer zu bearbeitenden PLA-Spezifikation existiert ein eigenes `Set`. Die darin enthaltenen Binär-Objekte haben gemeinsame Formatparameter, d. h. n und m sind für alle darin repräsentierten PLA-Tabellen identisch.

Innerhalb eines `Set` können bestimmte Binär-Objekte ausgezeichnet werden. Auf diese Weise können Objekte im `Set` als globale Vergleichspartner deklariert werden. Beispielsweise kann ein Binär-Objekt mit dem Begriff `Reference` belegt werden; die durch seine Komponente `terms` angegebenen PLA-Kosten werden zum Vergleich mit den Kosten anderer Spezifizierungen als c_0 herangezogen.

7.2.2 Systemkomponenten in HOPE

Die in Hope implementierte Funktionalität gliedert sich in mehrere Systemkomponenten, die in Abbildung 7.2 (S. 141) dargestellt sind. Das `Set`, also die Gesamtheit aller in Hope aktuell existierenden Objekte, liegt im Zentrum der Grafik. Es wird umlagert von verschiedenen Komponenten, die unterschiedliche Dienste anbieten.

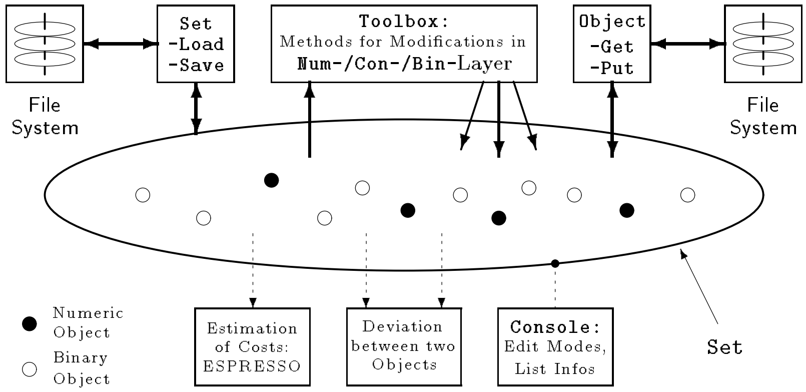

Abbildung 7.2: Systemkomponenten in Hope

- `Set-Load` bzw. `Set-Save`:
  Diese Komponente bewirkt das Laden bzw. Speichern des `Set`s auf dem Massenspeicher. Dabei werden zusätzlich zu sämtlichen Objekten auch noch einige Systemeinstellungen sowie globale Deklarationen geladen bzw. gespeichert, so daß eine Fortführung von Optimierungssitzungen nach einer Unterbrechung später möglich ist. Die Dateien zur Aufnahme eines `Set`s tragen die Extension `.set` . Das zugehörige Datei-Format ist in Anhang C auf Seite 164 beschrieben.

- `Toolbox`: Methods for Modifications
  Dieses ist die zentrale Komponente in Hope; sie dient zur Erzeugung neuer Objekte, wobei sie hauptsächlich mit den in Kapitel 5 vorgestellten Methoden aus existierenden Objekten neue kreieren kann. Wie in Kapitel 5 können die einsetzbaren Methoden den drei Schichten aus Abschnitt 4.1.4 (S. 60f), nämlich Numerik-, Konversions- und Binär-Schicht, zugeordnet werden. Sie greifen in der Regel auf existierende Objekte zurück und erzeugen häufig sogar mehrere neue Objekte. Im folgenden Abschnitt 7.2.3 (S. 143) wird näher auf die implementierten Methoden eingegangen.

- `Object-Get` bzw. `Object-Put`:
  Für den Import bzw. Export von Objekten stellt diese Komponente eine Schnittstelle zum Massenspeicher her. Dabei können sowohl *numerische* als auch *binäre* Objekte importiert bzw. exportiert werden. Hierfür sind einfache Datei-Formate implementiert; im Falle *binärer* Objekte werden die folgenden drei Formate von Hope unterstützt:

  `.esp` Espresso-Format für PLAs [Esp-88].

  `.pla` PLA-Format der ES2-Entwurfsumgebung Solo-2030 [ESS-92].

`.tex` LaTeX-Format einer PLA-Tabelle, das nur zur Exportierung von Binär-Objekten dient und lediglich für die Dokumentation von Beispielen unterstützt wird; zu LaTeX siehe [Kop-95].

Beispiele zu den ersten beiden Formaten sind in Anhang C auf Seite 163 abgebildet. Eine von Hope erstellte Beispiel-Datei im LaTeX-Format ist in Anhang C auf Seite 162 gelistet und demonstriert.

- `Estimation of Costs`:
  Durch diese Komponente wird *kein* Objekt kreiert, sondern lediglich auf ein Objekt im `Set` zugegriffen; daher der aus dem `Set` herausführende gestrichelte Pfeil in Abbildung 7.2 (S. 141). Für jeweils ein Binär-Objekt werden die PLA-Kosten ermittelt, d. h. es wird die Termanzahl der PLA-Tabelle bestimmt. Dazu wird der in Hope integrierte Logik-Minimierer Espresso [Bra*82b, Bra*84] verwendet, wobei die Minimierung in einem von fünf zur Verfügung stehenden Qualitätsmodi stattfinden kann. Näheres dazu ist in Anhang D (vgl. Abschnitt D.3.1, S. 175) zu finden.

- `Deviation between two Objects`:
  Zum Zweck des Vergleichs wird auf zwei Binär-Objekte zugegriffen. Dabei wird eines der Objekte als *korrekt* und das andere als *abweichend* interpretiert. Durch diese Sichtweise ergibt sich der Abweichungsvektor d für das *abweichende* Objekt (vgl. Abschnitt 6.2.5.3, S. 113). Nicht selten wird global für alle *binären* Objekte im `Set` jenes Binär-Objekt, das die PLA-Tabelle p_0 der Referenz repräsentiert, als das *korrekte* Vergleichsobjekt deklariert (vgl. Abschnitt 7.2.1.3, S. 140).

- `Console`: Edit Modes, List Infos
  Die Bedienung von Hope erfolgt mit Hilfe einer Kommandozeilen-orientierten Schnittstelle, d. h. Hope wird in der Regel als Applikation einer sogenannten Shell gestartet und gesteuert. Zur Überwachung der Optimierungsarbeiten an einem `Set` sind durch diese Komponente Betriebszustände beeinflußbar (Edit Modes) sowie Informationen über den Zustand und Umfang des `Set`s abrufbar (List Infos). So können beispielsweise Objekt-Listen nach verschiedenen Kriterien sortiert werden. Ferner können Objekte, die sich als unbrauchbar erweisen, verborgen werden. Obwohl sie gelöscht werden können, empfiehlt es sich, sie verdeckt beizubehalten, da sie die Unbrauchbarkeit des eingeschlagenen Weges dokumentieren. Auf diese Weise kann verhindert werden, daß die gleichen Operationen, die zu Erzeugung der betreffenden Objekte geführt haben, möglicherweise unbeabsichtigt wiederholt werden.

7.2.3 Implementierte Methoden in HOPE

Als zentrale Systemkomponente ging im vorhergehenden Abschnitt 7.2.2 die `Toolbox` mit den Methoden zur Modifikation hervor. Es handelt sich dabei um Methoden, die in Kapitel 5 vorgestellt wurden und die sich den drei Schichten aus Abschnitt 4.1.4 (S. 60f) zuordnen lassen.

- **Numerik-Schicht:** Methoden der Numerik-Schicht erzeugen *numerische* Objekte, d. h. solche Objekte repräsentieren numerische Tabellen mit reellen Einträgen. Teilweise greifen die Methoden dabei auf vorhandene Numerik-Objekte zurück.

  - **Funktions-Methoden:** Fct_Sine, Fct_Root, Fct_Expo, Fct_Loga oder Fct_Hbol.
    Durch diese Systemkomponenten kann ein neues Numerik-Objekt kreiert werden. Alternativ stehen die *Sinus-*, die *Quadratwurzel-*, die *Exponential-*, die *Logarithmus-* oder die *Hyperbel-Funktion* zur Auswahl. Die implementierten Funktionen sind entsprechend der Zusammenstellung der Beispiele in Abschnitt 6.3.2 (S. 127f) ausgewählt und dienten zur Generierung der Testdaten.
  - **Skalierungs-Methoden:** Scl_Add, Scl_Sub, Scl_Mul oder Scl_Div.
    Es wird mit den vier Grundrechenarten linear skaliert, d. h. alternativ kommen entweder *Addition*, *Subtraktion*, *Multiplikation* oder *Division* zum Einsatz. Diese Operationen greifen jeweils auf ein Numerik-Objekt zu und erzeugen ein weiteres, bei dem alle Einträge entsprechend der gewählten Grundrechenart skaliert sind.

- **Konversions-Schicht:** Die Methoden dieser Schicht führen prinzipiell die Konvertierung von Numerik-Objekten zu Binär-Objekten durch, d. h. sie greifen auf ein *numerisches* Objekt zurück und erzeugen daraus ein binäres Objekt; in der Regel resultieren gleich mehrere Binär-Objekte daraus.

  - **Rundungs-Methoden:** Rnd_Round, Rnd_Floor und Rnd_Ceil.
    Diese drei Operationen werden an einem *numerischen* Objekt durchgeführt und erzeugen je ein neues Binär-Objekt, dessen Einträge jeweils durch den entsprechenden Rundungsmodus von den reellen Einträgen des *numerischen* Objektes zu Ganzzahlen konvertiert sind.
  - **2er-Potenz-Methoden:** P2_Low, P2_High und P2_Near.
    Ausgehend von einem Numerik-Objekt, werden hierbei durch drei Operationen, nämlich zu einem reellen Wert jeweils den unterhalb-, oberhalb- bzw. nächstliegende 2er-Potenzwert zuzuordnen, neue Binär-Objekte erzeugt.
  - **Intervall-Methoden:** Int_UpDn, Int_Abs oder Int_Rel.
    Bezogen auf ein *numerisches* Objekt, können in drei Varianten zu jedem reellen Wert ganzzahlige Intervalle generiert werden. Zur Auswahl stehen Intervalle aus Ab- und Aufrundwert oder durch Angabe eines absoluten bzw. relativen Toleranzbereiches. Da es in Hope für diese Art von nicht komplett konvertierten Objekten keinen eigenen Objekt-Typ gibt, werden hierbei durch eine spezielle Form in der Binärkodierung Binär-Objekte erzeugt. Sie weisen im Unterschied zu anderen Binär-Objekten allerdings ein hinzugefügtes Intervall-Attribut in der `flags`-Komponente auf.
  - **Selektions-Methoden:** Sel_Low, Sel_High, Sel_Mean, Sel_Min1 und Sel_NP2.
    Es handelt sich hierbei um fünf Operationen, die auf Binär-Objekte mit Intervall-Attribut zugreifen. Solche Objekte werden ausschließlich durch die drei Intervall-Varianten generiert. Bei der Selektion werden anhand der Intervalle einzelne Werte

für jeden Eintrag ausgewählt. Auf diese Weise werden fünf neue Binär-Objekte (ohne Intervall-Attribut) erzeugt.

- **Binär-Schicht:** Methoden der Binär-Schicht arbeiten nur mit *binären* Objekten, d. h. sie greifen nur auf solche zurück und erzeugen auch nur Objekte mit diesem Typ.

  - **Pseudo-Übertragsbildung:** Psd_Inc und Psd_Dec.
    Diese Operationen nehmen Modifikationen an einem Binär-Objekt vor und erzeugen für jedes abweichende Resultat ein neues *binäres* Objekt. Zunächst werden allerdings die Wertebereiche der für beide Methoden notwendigen Parameter ermittelt. Anschließend werden in einer Vielfalt, die der Anzahl der Parameter entspricht, neue Binär-Objekte erzeugt.
  - **Ähnlichkeit von Einträgen:** Sim_Bin und Sim_Num.
    Auch hier wird ein Binär-Objekt modifiziert und für jeweils neue Resultate werden *binäre* Objekte kreiert. Analog zu den Psd-Methoden werden erst die Wertebereiche der Parameter ermittelt und anschließend alle Methoden für die Parameter evaluiert. Insbesondere beim Sim_Num wird Redundanz dadurch vermieden, daß ein neues Binär-Objekt erst dann erzeugt wird, wenn die betreffende PLA-Tabelle noch nicht bereits generiert ist.
  - **Hyper-Term Überdeckung:** Hyp_Trm.
    Nachdem für ein Binär-Objekt die Anzahl der Hyper-Terme ermittelt ist, wird die Methode für alle Hyper-Terme evaluiert und auf diese Weise jeweils ein neues Binär-Objekt erzeugt.

7.2.4 C-Quelltext-Module in HOPE

Das Werkzeug Hope wurde in der Programmiersprache *C* [Ker*88, Bar-89] implementiert. Die Quelltext-Dateien sind auf der ersten der beigefügten Disketten (beide im IBM-DD-Format, 720 KBytes) abgespeichert; die zweite Diskette enthält die Quelltext-Dateien einer angepassten Espresso-Version (vgl. Abschnitt C.4, S. 165).

Der gesamte Quelltext für Hope ist in zwölf *C*-Module unterteilt, die jeweils thematisch verwandte Funktionen bzw. Routinen zusammenfassen. In Anhang C wird in Abschnitt C.1 (S. 159) auf die einzelnen Module näher eingegangen.

Kapitel 8

Zusammenfassung

Die vorliegende Arbeit zeigt, daß bei der Realisierung von Logikfunktionen, wie sie in Kapitel 2 (S. 13ff) erläutert worden ist, der Art und Weise der Spezifikation (vgl. Kapitel 3, S. 35ff) eine entscheidende Bedeutung zukommt. Die Untersuchungen zu verdeckten Optionen und speziellen Aspekten zur inhärenten Unschärfe (vgl. Kapitel 4, S. 57ff) geben Aufschluß darüber, daß eine Spezifikation im allgemeinen nicht exakt sein kann, sondern daß sie in der dort diskutieren Weise zwangsläufig Ungenauigkeiten mit sich führt.

Diese Erkenntnis dient als Grundlage für die in Kapitel 5 (S. 75ff) vorgestellten Heuristiken, die als Methoden zur Modifikation der Spezifizierung einer PLA-Tabelle zum Einsatz kommen. Dabei treten die erzielbare Reduzierung bei den PLA-Kosten in Wechselwirkung mit den gleichzeitig verursachten Abweichungen zwischen der *ad hoc* spezifizierten und der modifiziert-spezifizierten PLA-Tabelle. Für eine nutzbringende Optimierung ist daher ein strategisches Vorgehen notwendig, wie es in Kapitel 6 (S. 103ff) erläutert wird; ferner sind darin die vorgestellten Methoden anhand von Beispielen evaluiert.

8.1 Bewertung der Ergebnisse

Die in der Entwurfspraxis gegenwärtig hemmenden Probleme erscheinen mit den Ergebnissen dieser Arbeit lösbar, in extremen Fällen zumindest entschärfbar. Darunter fallen insbesondere solche Situationen, in denen die Kosten bei der Implementierung von Logikfunktionen mit PLA-Strukturen entweder kritische Vorgabewerte oder spezifizierte Grenzwerte übersteigen; dieses trifft vor allem auf die statische Stromaufnahme des PLA-Blockes zu, die nicht selten Größenordnungen annimmt, welche im Zuge der allgemeinen Bemühungen zur Energieeinsparung beim Chip-Entwurf nicht akzeptabel sind.

Optimierungsergebnisse

Die durch das Schichtenmodell abstrahierte Gliederung der Spezifizierung läßt in vielfältiger Weise Modifikationen mit Hilfe von heuristisch arbeitenden Methoden zu (vgl. Kapitel 5, S. 75). Die Evaluation der vorgestellten Methoden zeigt bei den PLA-Kosten Möglichkeiten zur Reduzierung in Größenordnungen von über 60%. Einschränkend ist hierbei allerdings, daß hohe Werte bei der Reduzierung im allgemeinen nur durch solche Modifikationen an der Spezifizierung erzielt werden können, die gleichzeitig zu einem hohen Maß an Abweichungen, gemessen an den ursprünglich spezifizierten Daten, führen. Insofern geht das Erstellen von *PLA-gerechten* Spezifikationen (vgl. Abschnitt 5.1, S. 75) nicht ohne eine strategische Beurteilung der spezifizierten PLA-Tabelle, wie sie in Kapitel 6 (S. 103ff) vorgestellt worden ist. Eine optimale Lösung bzw. die Nähe zum Optimum kann aufgrund der Beliebigkeit der Abweichungen nicht angegeben werden.

Entwurfswerkzeug HOPE

Die erzielbaren Werte bei der Reduzierung der PLA-Kosten und die gegenwärtige Bedarfssituation in der Entwurfspraxis motivierten zur Implementierung des in Kapitel 7 (S. 135ff) vorgestellten Werkzeugs Hope, das sich in die Palette der CAD-Werkzeuge für den VLSI-Entwurf nutzbringend einreihen läßt. Die in Kapitel 6 (vgl. Abschnitt 6.3, S. 118) dargestellten Resultate bei der Evaluation der Methoden wurden mit Hilfe von Hope ermittelt.

Die Implementierung erfolgte in der Programmiersprache *C* und unabhängig von einem bestimmten Betriebssystem, so daß eine hohe Portabilität erreicht werden konnte. Ferner ist in Hope der weithin akzeptierte Logik-Minimierer Espresso integriert, so daß in Hope die Logik-Minimierung in einer allgemein anerkannten „Qualität“ durchgeführt werden kann.

8.2 Ausblick

Obwohl sich die Motivation zu dieser Arbeit aus den sich zunehmend verschärfenden Entwurfsproblemen bei der Implementierung von Logikfunktion mit Hilfe von PLA-Strukturen ergab, kann die vorgestellte Optimierung auch von der PLA-Implementierung losgelöst durchgeführt werden. Es sollte sich lediglich um Logikfunktionen bzw. um eine Wahrheitstabelle handeln, deren Daten einen numerischen Ursprung haben. Die Optimierung kann insbesondere auch für alternative Realisierungsstrukturen aus Standardzellen-Bibliotheken erfolgen.

Erweiterungen in der Methodik

Da die Kombination von *Intervall-* und *Selektions*-Methoden gemäß der Ergebnisse sehr hoffnungsvoll ist, erscheint beispielsweise eine weitere Untersuchung der folgenden Vorgehensweise interessant: Die bislang vorgestellte Kombination aus Intervall- und Selektions-Methoden bestimmt für jeden Eintrag der Tabelle *einzeln* einen Wert. Vor der Selektion könnte nun durch eine übergeordnete Betrachtung der Intervalle von *vergleichbaren* Einträgen, also in der Tabelle nebeneinander liegenden Einträgen (vgl. Abschnitt 5.2.2.2, S. 94f), nach Möglichkeiten zur Selektion von Werten gesucht werden, die innerhalb von Schnittmengen oder Häufungsbereichen zweier Intervalle liegen.

Ansatzweise kann dies bereits mit den bisherigen Mitteln dadurch vorab getestet werden, indem auf PLA-Tabellen, die aus einer Kombination von Intervall- und Selektions-Methoden hervorgegangen sind, anschließend die binäre Sim-Methode, also die *Ähnlichkeit von Einträgen*, angewendet wird. Die neue Vorgehensweise würde nun die Reihenfolge von *Selektion* und *Angleichung* umkehren, was einer noch PLA-gerechteren Selektion zugute käme.

Bezüglich der in dieser Arbeit insgesamt vorgestellten Methoden ist anzumerken, daß sie sich vornehmlich auf eine Steigerung der Effektivität bei der Logik-Minimierung konzentrieren, die in der Regel anschließend an die Spezifikation der PLA-Tabelle zum Einsatz kommt. Weitere Möglichkeiten für die Ausrichtung heuristischer Methoden bestehen in den speziellen Belangen von optimierten PLA-Strukturen, wie z. B. die gefalteten oder dekodierten PLAs (vgl. Kapitel 2, S. 13ff).

Erweiterungen in HOPE

Das im Rahmen dieser Arbeit implementierte Entwurfswerkzeug Hope bietet noch umfangreiche Gelegenheiten zur Erweiterung. Insbesondere ist hier eine grafische Benutzeroberfläche zu nennen, deren Implementierung sich allerdings an gängige Standards bzw. Normen orientieren sollte, damit die bisher hohe Portabilität von Hope weitestgehend erhalten bleibt.

Daneben könnte die in Hope implementierte Funktionalität mit Hilfe einer Steuerdatei automatisiert werden; dies könnte in Gestalt eines *Batch-Interpreters* realisiert werden, für den eine Kommandosprache entwickelt werden kann, die sich an die im Anhang D erwähnten Sequenzen aus sogenannten *Kennbuchstaben* anlehnt (vgl. Abschnitt D.2.3, S. 168).

Ferner könnte eine grafische Wiedergabe von Tabellendaten bei der Optimierung sehr nützlich sein, da auf diese Weise Abweichungen wesentlich leichter zu beurteilen wären; die Graphen in den Abbildungen 6.3, 6.4 und 6.5 (S. 125f) demonstrieren dies eindrucksvoll.

Weiter könnte eine leicht zu interpretierende grafische Darstellung von Abweichungen zwischen zwei Tabellen wie folgt erstellt werden: In einem zwei-dimensionalen Koordinatensystem werden für jeden Index in x-Richtung der Funktionswert der einen Tabelle und in y-Richtung der Funktionswert der anderen abgetragen. Der sich daraus ergebende Schnittpunkt wird grafisch markiert. Die Erscheinung aller Schnittpunkte bildet bei geringfügig voneinander abweichenden Tabellen die Gestalt einer 45°-Linie. Je nach Tendenz bzw. Zunahme der Abweichungen neigt sich die Linie bzw. wird die Punktemenge diffuser, was einer leichteren Beurteilung der Abweichungen zuträglich wäre.

Literatur

[Aho*74] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, 1974.

[Bac-84] J. Bach. *Output Partitioning of Programmable Logic Arrays.* PhD thesis, Department of Computer Science, University of Illionis at Urbana-Champaign, September 1984.

[Bai*93] U. Baitinger, F. Dresig, P. Lanches, and O. Rettig. *Simulation and Reduction of CMOS Power Dissipation at Logic Level.* In *Proceedings of EUROASIC 93*, pages 341–346, Paris, 1993.

[Bar-89] K. A. Barclay. *C: Problem Solving and Programming.* Number 1 in Computer systems. Programming languages: C language. Prentice Hall International (UK) Ltd., Hemel Hempstead, Hertfordshire, HP2 4RG, 1989.

[Bra*81] R. K. Brayton, G. D. Hachtel, and A. L. Sangiovanni-Vincentelli. *A Survey of Optimization Techniques for Integrated Circuit Design.* Research Report RC 8882, IBM, June 1981.

[Bra*82a] R. K. Brayton, J. Cohen, G. D. Hachtel, B. Trager, and D. Yun. *Fast Recursive Boolean Function Manipulation.* In *Proceedings of International Symposium on Circuits and Systems*, pages 58–62, May 1982.

[Bra*82b] R. K. Brayton, G. D. Hachtel, L. Hemachandra, A. R. Newton, and A. L. Sangiovanni-Vincentelli. *A Comparision of Logic Minimization Strategies Using ESPRESSO: An APL Program Package for Partitioned Logic Minimization.* In *Proceedings of International Symposium on Circuits and Systems*, pages 42–48, Rome, May 1982.

[Bra*84] R. K. Brayton, G. D. Hachtel, C. T. McMullen, and A. L. Sangiovanni-Vincentelli. *Logic Minimization Algorithms for VLSI Synthesis.* Kluwer Academic Publishers, Boston, MA, 1984.

[Bra-87a] R. K. Brayton. *Algorithms for Multilevel Logic Synthesis and Optimization.* In *Proceedings of NATO ASI on Logic Synthesis and Silicon Compilation for VLSI*, The Netherlands, 1987.

[Bra*87b] R. K. Brayton, R. L. Rudell, A. L. Sangiovanni-Vincentelli, and A. R. Wang. *MIS: Multiple-Level Logic Optimization System. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, CAD-6:1062–1081, November 1987.

[Bra*88] K. A. Bartlett, R. K. Brayton, G. D. Hachtel, R. M. Jacobi, C. R. Morrison, R. L. Rudell, A. L. Sangiovanni-Vincentelli, and A. R. Wang. *Multi-Level Logic Minimization Using Implicit Don't Cares. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, CAD-7:726–740, June 1988.

[Bra*89a] R. K. Brayton and F. Somenzi. *Boolean Relations and the Incomplete Specification of Logic Networks.* In G. Musgrave and U. Lauther, editors, *International Conference on Very Large Scale Integration and Computers*, pages 231–240. Elsevier Science Publishers B. V. (North-Holland), Munich, August 1989.

[Bra*89b] R. K. Brayton and F. Somenzi. *An Exact Minimizer for Boolean Relations.* In *Proceedings of IEEE International Conference on Computer-Aided Design*, pages 316–319, November 1989.

[Bra*89c] R. K. Brayton and F. Somenzi. *Minimization of Boolean Relations.* In *Proceedings of ISCAS*, Portland, OR, May 1989.

[Bra-90a] R. K. Brayton. *New Directions in Logic Synthesis.* In *Proceedings of Synthesis and Simulation Meeting and International Interchange*, Kyoto, Japan, 1990.

[Bra*90b] R. K. Brayton, T. Kam, S. Malik, and A. Srinivasan. *Algorithms for Discrete Function Manipulation.* In *Proceedings of IEEE International Conference on Computer-Aided Design*, pages 92–95, November 1990.

[Bra*93] R. K. Brayton and Y. Watanabe. *Heuristic Minimization of Multiple-Valued Relations. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, CAD-12(10):1458–1472, October 1993. first presented at the *International Workshop on Logic Synthesis,* May 1991.

[Bre-75] M. A. Breuer. *Digital Systems Design Automation: Languages, Simulation and Data Base.* Computer Science Press, 1975.

[Bro*81] I. Bronstein and K. Semendjajew. *Taschenbuch der Mathematik.* Herausgegeben von G. Grosche und V. Ziegler, Lizenzausgabe für den Verlag Harri Deutsch, Thun. B. G. Teubner Verlagsgesellschaft, Leipzig, 20. Auflage, 1981.

[Bry-86] R. E. Bryant. *Graph Based Algorithms for Boolean Function Manipulation. IEEE Transactions on Computers*, C-35:677–691, August 1986.

[Cha*92] S. Chakroun, M. Dana, E. Melcher, and W. Röthig. *Power Consumption Estimation Using Statistical Signal Properties.* In *Proceedings of EUROMICRO 92*, pages 691–696, Paris, 1992.

[Cha*95] A. P. Chandrakasan, M. Potkonjak, R. Mehra, J. Rabaey, and R. W. Brodersen. *Optimizing Power Using Transformations. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, CAD-14(1):12–31, January 1995.

[Cir-87] M. Cirit. *Estimating dynamic power consumption of CMOS circuits.* In *Proceedings of IEEE International Conference on Computer-Aided Design*, pages 534–537, 1987.

[Dag*85] V. K. Agarwal, M. R. Dagenais, and N. C. Rumin. *The McBoole Logic Minimizer.* In *Proceedings of 22nd Design Automation Conference*, page 667, Las Vegas, NV, June 1985.

[Dag*-86] V. K. Agarwal, M. R. Dagenais, and N. C. Rumin. *McBoole: A New Procedure for Exact Logic Minimization. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, CAD-5:229–238, January 1986.

[deM*-87] P. Antognetti, G. de Micheli, and A. L. Sangiovanni-Vincentelli. *Design Systems for VLSI Circuits: Logic Synthesis and Silicon Compilation*. Martinus Nijhoff Publishers, 1987.

[Dev*-86] S. Devadas and A. R. Newton. *GENIE: A Generalized Array Optimizer for VLSI Synthesis*. In *Proceedings of 23rd Design Automation Conference*, pages 631–637, July 1986.

[Dev*-90] S. Devadas and A. R. Newton. *Exact algorithms for output encoding, state assignment and four-level boolean minimization*. In *Proceedings of Hawaii International Conference on System Sciences*, pages 387–396, 1990.

[Dud*-88] H. Engesser *et al.*, editors. *Duden Informatik*. Der Duden: Sachlexikon für Studium und Praxis. Bibliographisches Institut & F. A. Brockhaus AG, Mannheim Wien Zürich, 1988. Bearb. von V. Claus und A. Schwill.

[Dud*-90] G. Drosdowski *et al.*, editors. *Duden Fremdwörterbuch*, volume 5 of *Der Duden: in 10 Bänden; das Standardwerk zur deutschen Sprache, hrsg. vom Wissenschaftlichen Rat der Dudenredaktion*. Bibliographisches Institut & F. A. Brockhaus AG, Mannheim Wien Zürich, fünfte Auflage, 1990.

[Ega*-82] J. Egan and C. L. Liu. *Optimal Bipartite Folding of PLA*. In *Proceedings of 19th Design Automation Conference*, pages 141–146, 1982.

[Esp-88] R. Rudell. *Manual Files* `espresso.1` *and* `espresso.5` *refering to Version 2.3 of* ESPRESSO. Department of Electrical Engineering and Computer Science, University of California at Berkeley, CA 94720, January 1988. ESPRESSO is part of the Berkeley OCTTOOLS and was available with the SIS-1.1 package, released 4-May-1992 by Ellen M. Sentovich.

[ESS-92] European Silicon Structures, Rousset, France. *ES2 ECPD15 & ECPD12 Library Databook*, June 1992. Version No. E03A04, Part No. 002-0879, Associated Software: SOLO 2030 V4.5.

[Fle*-75] H. Fleisher and L. Maissel. *An Introduction to Array Logic. IBM Journal of Research and Development*, 19:98–109, March 1975.

[Gho*-90] A. Ghosh, S. Devadas, and A. R. Newton. *Heuristic Minimization of Boolean Relations Using Testing Techniques*. In *Proceedings of IEEE International Conference on Computer and Design*, pages 277–281, Cambridge, September 1990. Also published with associated Editor R. K. Brayton in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, CAD-11(9):1166–1172, September, 1992.

[Gil*-80] W. Giloi and H. Liebig. *Logischer Entwurf digitaler Systeme*. Springer, Berlin Heidelberg New York, zweite Auflage, 1980.

[Giv-70] D. Givone. *Introduction to Switching Circuit Theory*. McGraw-Hill, New York, 1970.

[Gra-82] W. Grass. *A Depth-First Branch and Bound Algorithm for Optimal PLA Folding.* In *Proceedings of 19th Design Automation Conference*, pages 133–140, 1982.

[Hac*80] G. D. Hachtel and A. Newton. *Some Results in Optimal PLA Folding.* In *Proceedings of International Conference on Circuits and Computers*, pages 1023–1026, Port Chest, NY, October 1980.

[Hac*82] G. D. Hachtel, A. R. Newton, and A. L. Sangiovanni-Vincentelli. *An Algorithm for Optimal PLA Folding. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, CAD-1(2):63–77, April 1982.

[Hac*87] G. D. Hachtel and M. R. Lightner. *Don't Care Conditions in Top Down Synthesis.* In *Proceedings of IEEE International Conference on Computer-Aided Design*, pages 316–319, Santa Clara, November 1987.

[Hon*74] S. J. Hong, R. G. Cain, and D. L. Ostapko. *MINI: A Heuristic Approach for Logic Minimization. IBM Journal of Research and Development*, 18:443–458, September 1974.

[Hon*84] S. J. Hong and S. Muroga. *Binary Logic Theory for Design of Minimal Decoded-PLA's.* Technical Report UIUCDCS-R-84-1192 / UILU-ENG-84-1726, Illinois University Urbana, Urbana, Illinois, December 1984.

[Kan-81] S. Kang. *Synthesis and Optimization of Programmable Logic Arrays.* Technical Report 216, Department of Electrical Engineering and Computer Science, Stanford University, Stanford, CA, July 1981.

[Ker*88] B. W. Kernighan and D. M. Ritchie. *The C Programming Language.* Prentice Hall, Englewood Cliffs, New Jersey, 2nd edition, 1988.

[Kop-95] H. Kopka. *Einführung.* LaTeX, Band 1. Addison-Wesley, Bonn Paris Reading, 1994/95.

[Kov*79] T. Kovylars and A. Al-Najjar. *An Examination of the Cost Function for Programmable Logic Arrays. IEEE Transactions on Computers*, C-28(8):586–590, August 1979.

[Kra-94] A. Krause. *Konzeption und Entwicklung eines heuristischen Verfahrens zur PLA-Minimierung auf Basis von Unschärfen.* Studienarbeit, Fachbereich Informatik, Universität Hamburg, Hamburg, January 1994.

[Lan*93] P. E. Landman and J. M. Rabaey. *Power estimation for high level synthesis.* In *Proceedings of European Design Automation Conference*, pages 361–366, Paris, France, February 1993.

[Lar-75] J. A. Larkin. *The Application of Iterated Consensus to Multiple-Valued Logic Design.* Technical Report GE/EE/75-6, Air Force Institute of Technique, Wright-Patterson AFB, Ohio, June 1975.

[Lew*84] J. L. Lewandowski and C. L. Liu. *A Branch and Bound Algorithm for Optimal PLA Folding.* In *Proceedings of 21st Design Automation Conference*, pages 426–431, 1984.

[Lin*90a] B. Lin and F. Somenzi. *Minimization of Symbolic Relations.* In *Proceedings of IEEE International Conference on Computer-Aided Design*, pages 88–91, November 1990.

[Lin*90b] B. Lin, G. S. Whitcomb, and A. R. Newton. *Symbolic don't cares and equivalence in high-level synthesis.* In *Proceedings of IFIP International Working Conference on Logic and Architecture Synthesis*, May 1990.

[McC-56] E. J. McCluskey, Jr. *Minimization of Boolean Functions. Bell System Tech. Journal*, 35:1417–1444, November 1956.

[McC-65] E. J. McCluskey, Jr. *Introduction to the Theory of Switching Circuits.* McGraw-Hill, 1965.

[Mea*80] C. Mead and L. Conway. *Introduction to VLSI Systems.* Addison-Wesley, Reading, 1980.

[Mur*74] S. Muroga and H. C. Lai. *Minimization of Logic Networks Under a Generalized Cost Function.* Technical Report UIUCDCS-R-74-649(?), Department of Computer Science, University of Illinois at Champaign-Urbana, Urbana, Illinois, April 1974.

[Mur-79] S. Muroga. *Logic Design and Switching Theory.* John Wiley & Sons, New York, 1979.

[Mur-82] S. Muroga. *VLSI Systems Design.* John Wiley & Sons, New York, 1982.

[Nak*71a] T. Nakagawa and H. C. Lai. *A Branch-And-Bound Algorithm for Optimal NOR Networks (The Algorithm Description).* Report 438, Department of Computer Science, University of Illinois at Champaign-Urbana, Urbana, Illinois, April 1971.

[Nak*71b] T. Nakagawa, H. C. Lai, and S. Muroga. *Pruning and Branching Methods for Designing Optimal Networks by the Branch-And-Bound Methods.* Report 471, Department of Computer Science, University of Illinois at Champaign-Urbana, Urbana, Illinois, August 1971.

[Pai-81] J. F. Paillotin. *Optimization of the PLA Area.* In *Proceedings of 18th Design Automation Conference*, pages 406–410, 1981.

[Pos-89] H.-U. Post. *Entwurf und Technologie hochintegrierter Schaltungen.* Leitfäden und Monographien der Informatik. Teubner, Stuttgart, 1989.

[Pur-93] Pure Software GmbH und Borland International, Heidelberg. *Handbuch zum C-Entwicklungssystem für die Atari-ST-, -TT- und -Falcon-Familie*, zweite Auflage, 1993.

[Rau-92] R. Rauscher. *Entwicklung und Implementation eines ICs zur krümmungsadaptiven Deltamodulation.* Mitteilung FBI-HH-M-210/92, Fachbereich Informatik, Universität Hamburg, Hamburg, August 1992.

[Rau-95] R. Rauscher. *A System for PLA Minimization based on Slight Behaviour Modifications.* In *Proceedings of EUROMICRO 95*, pages 8–15, Como, September 1995.

[Reu-72] B. Reusch. *On Boolean Function Minimization.* Seminarbericht 51, GMD-Institut für Theorie der Automaten und Schaltnetzwerke, St. Augustin, Bonn, August 1972.

[Rot-78] J. P. Roth. *Programmable Logic Array Optimization*. *IEEE Transactions on Computers*, C-27:174–176, February 1978.

[Rot-80] J. P. Roth. *Computer Logic, Testing and Verification*. Computer Science Press, 1980.

[Rud*86] R. L. Rudell and A. L. Sangiovanni-Vincentelli. *Exact Minimization of Multiple-Valued Functions for PLA Optimization*. In *Proceedings of IEEE International Conference on Computer-Aided Design*, pages 352–355, 1986.

[Rud*87] R. L. Rudell and A. L. Sangiovanni-Vincentelli. *Multiple-Valued Minimization for PLA Optimization*. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, CAD-6:727–750, September 1987.

[Rud*90] K. L. Brace, R. L. Rudell, and R. E. Bryant. *Efficient implementation of a BDD package*. In *Proceedings of Design Automation Conference*, pages 40–45, June 1990.

[Sas*79] T. Sasao and H. Terada. *Multiple-valued Logic and the Design of Programmable Logic Arrays with Decoders*. In *Proceedings of 9th International Symposium on Multi-Valued Logic*, pages 27–37, 1979.

[Sas-81] T. Sasao. *Muliple-Valued Decomposition of Generalized Boolean Functions and the Complexity of Programmable Logic Arrays*. *IEEE Transactions on Computers*, C-30(9):635–643, September 1981.

[Sas-83] T. Sasao. *Input Variable Assignment and Output Phase Optimization of PLA's*. Research Report Computer Science / Mathematics RC 10003 (44411), Mathematical Sciences Department, IBM Thomas J. Watson Research Center, Yorktown Heights, New York 10598, June 1983. Also published in *IEEE Transactions on Computers*, October 1984.

[Sas-88] T. Sasao. *Multiple-valued Logic and Optimization of Programmable Logic Arrays*. *IEEE Transactions on Computers*, 21(4):71–80, 1988.

[Sas-93] T. Sasao. *Logic Synthesis and Optimization*. Kluwer Academic Publishers, 1993.

[Shi*72] S. Shiva and H. Nagle. *A Fast Algorithm for Complete Minimization of Boolean Functions*. Technical Report AU-T-24, Vice President for Research, Auburn University, Auburn, Alabama, June 1972.

[Suw-82] I. Suwa. *Topological Design of Segmented-Folded PLAs*. Thesis, Department of Computer Science, University of Illionis at Urbana-Champaign, Urbana-Champaign, October 1982.

[Tim-82] M. Timm. *Grundlagen von Anforderungs- und Entwurfspezifikationen im Prozeß der Software-Entwicklung*. GMD-Studie 66, Gesellschaft für Mathematik und Datenverarbeitung, St. Augustin bei Bonn, 1982.

[Ull-84] J. D. Ullman. *Computational Aspects of VLSI*. Computer Science Press, Rockville, 1984.

[Wes*85] K. Eshraghian and N. H. E. Weste. *Principles of CMOS VLSI Design (A Systems Perspective)*. VLSI Systems Series. Addison-Wesley, Reading, 1985.

Anhang A

ES2 Entwurfsumgebung Solo-2030

Zur Veranschaulichung der diskutierten Entwurfsprobleme bei PLAs werden hier beispielhaft konkrete Angaben bezüglich der Wahl einer Entwurfsumgebung gemacht. Am Fachbereich für Informatik der Universität Hamburg kommt in der Entwurfspraxis u. a. das SOLO-2030-System zum Einsatz [ESS-92]. Es handelt sich dabei um eine Entwurfsumgebung der Firma *European Silicon Structures (ES2)*.

A.1 Grenzwerte beim PLA-Entwurf

Unabhängig von der Wahl des Fertigungsprozesses sind bei der Entwurfsumgebung SOLO-2030 für die Anzahl der Ein- und Ausgänge sowie für die Anzahl der Produktterme folgende Bereiche zulässig:

Grenzwerte für PLA-Makrozellen	
Eingänge n	$1 \div 64$
Ausgänge m	$1 \div 32$
Produktterme ℓ	$1 \div 128$

Tabelle A.1: Grenzwerte für PLAs bei SOLO-2030

A.2 Dokumentation zum PLA-Entwurf

Die aktuellen Entwurfsbibliotheken des SOLO-2030 unterstützen[1] den Entwurf von PLA-Strukturen in mehreren Herstellungstechnologien. Nachfolgend werden Angaben zu den Fertigungsprozessen ECPD15, ECPD12 und ECPD10 gemacht. Dabei stehen die letzten beiden Ziffern für die minimale Strukturfeinheit: Beispielsweise handelt es sich beim ECPD10 um einen 1μm-Prozeß.

[1]Offenbar ist dies gegenwärtig keine Selbstverständlichkeit mehr: Am Fachbereich für Informatik der Universität Hamburg ist eine weitere Entwurfsumgebung in Gebrauch, die den Entwurf von PLAs gar nicht (mehr) unterstützt.

PLA-Fläche

Für die Abschätzung des Flächenbedarfes der PLA-Makrozelle werden in den drei Prozessen für die Breite W_{PLA} und die Höhe H_{PLA} folgende Beziehungen angegeben:

$$\begin{array}{lcll} W_{\mathrm{PLA}} & := & W_1 \cdot n + W_2 \cdot m + W_3 \cdot \mathrm{rnd}(W_4 \cdot m) + W_5 & [\mu\mathrm{m}] \\ H_{\mathrm{PLA}} & := & H_1 \cdot \ell + H_2 \cdot \mathrm{rnd}(H_3 \cdot \ell) + H_4 \cdot \mathrm{rnd}(H_5 \cdot \ell) + H_6 & [\mu\mathrm{m}] \end{array} \tag{A.1}$$

Während *rnd()* eine nicht näher spezifizierte Rundung bezeichnet, sind die Konstanten je nach Prozeß wie folgt gegeben:

- Prozeß ECPD12:

$$\begin{array}{llllll} W_1 := 10.8 & W_2 := 5.4 & W_3 := 3.0 & W_4 := \frac{1}{6} & W_5 := 55.8 & \\ H_1 := 5.7 & H_2 := 12 & H_3 := \frac{1}{12} & H_4 := 5.9 & H_5 := \frac{1}{4} & H_6 := 141.0 \end{array}$$

- Prozeß ECPD15:

$$\begin{array}{llllll} W_1 := 14.4 & W_2 := 7.2 & W_3 := 4.0 & W_4 := \frac{1}{6} & W_5 := 74.4 & \\ H_1 := 7.6 & H_2 := 16 & H_3 := \frac{1}{12} & H_4 := 6.8 & H_5 := \frac{1}{4} & H_6 := 188.0 \end{array}$$

- Prozeß ECPD10:

$$\begin{array}{llllll} W_1 := 9.0 & W_2 := 4.5 & W_3 := 2.5 & W_4 := \frac{1}{6} & W_5 := 46.5 & \\ H_1 := 4.8 & H_2 := 10.0 & H_3 := \frac{1}{12} & H_4 := 4.3 & H_5 := \frac{1}{4} & H_6 := 117.5 \end{array}$$

PLA-Stromaufnahme

Bei der Abschätzung der Stromaufnahme müssen neben den technologischen Aspekten die beabsichtigten Einsatzbedingungen des Chips berücksichtigt werden. Im Falle der Makrozellenbibliothek des SOLO-2030 ergibt sich nach der Dokumentation in [ESS-92] im Detail folgende Beziehung:

$$I_{\mathrm{PLA}} := I_1 \cdot (\ell - 1 + m) + I_2 \cdot (I_3 \cdot \ell \cdot n + C_{\text{OP-Load}} \cdot m) \cdot V_{\mathrm{DD}} \cdot f_{\mathrm{clk}} \qquad [\mathrm{mA}] \tag{A.2}$$

Während I_1, I_2 und I_3 reelle Konstanten sind, ist mit $C_{\text{OP-Load}}$ eine Kapazität [pF] gemeint. Der Wert liegt maximal bei 2 pF und beträgt laut Dokumentation typischerweise 0.95 pF. Ferner beschreibt V_{DD} [V] die Betriebsspannung und f_{clk} [MHz] die Frequenz des Systemtaktes. Die Konstanten sind je nach Prozeß wie folgt gegeben:

- Prozeß ECPD12 und ECPD15:

$$I_1 := 0.254 \qquad I_2 := 0.001 \qquad I_3 := 0.01$$

- Prozeß ECPD10:

$$I_1 := 0.325 \qquad I_2 := 0.001 \qquad I_3 := 0.0131$$

Anhang B

Beispiel einer PLA-Tabelle

Als Beispiel für eine komplette PLA-Tabellen ist die im Rahmen von Kapitel 3 erarbeitete Spezifikation zum *Sinus*-Beispiel in Tabelle B.1 (S. 158) dargestellt; die Erzeugung wird in Abschnitt 3.3.5.3 (S. 54) diskutiert.

Wahrheits- bzw. PLA-Tabelle			
i	$f_{\mathrm{int}}(i)$	$\mathrm{BIN}_7\,(i)$ $\xrightarrow{f_{\mathrm{tab}}}$	$\mathrm{BIN}_{10}\,(f_{\mathrm{int}}\,(i))$
X	Y	$x_6 x_5 x_4 x_3 x_2 x_1 x_0$	$y_9 y_8 y_7 y_6 y_5 y_4 y_3 y_2 y_1 y_0$
0	0	0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0
1	16	0 0 0 0 0 0 1	0 0 0 0 0 1 0 0 0 0
2	32	0 0 0 0 0 1 0	0 0 0 0 1 0 0 0 0 0
3	48	0 0 0 0 0 1 1	0 0 0 0 1 1 0 0 0 0
4	63	0 0 0 0 1 0 0	0 0 0 0 1 1 1 1 1 1
5	79	0 0 0 0 1 0 1	0 0 0 1 0 0 1 1 1 1
6	95	0 0 0 0 1 1 0	0 0 0 1 0 1 1 1 1 1
7	111	0 0 0 0 1 1 1	0 0 0 1 1 0 1 1 1 1
8	127	0 0 0 1 0 0 0	0 0 0 1 1 1 1 1 1 1
9	142	0 0 0 1 0 0 1	0 0 1 0 0 0 1 1 1 0
10	158	0 0 0 1 0 1 0	0 0 1 0 0 1 1 1 1 0
11	174	0 0 0 1 0 1 1	0 0 1 0 1 0 1 1 1 0
12	189	0 0 0 1 1 0 0	0 0 1 0 1 1 1 1 0 1
13	205	0 0 0 1 1 0 1	0 0 1 1 0 0 1 1 0 1
14	220	0 0 0 1 1 1 0	0 0 1 1 0 1 1 1 0 0
15	236	0 0 0 1 1 1 1	0 0 1 1 1 0 1 1 0 0
16	251	0 0 1 0 0 0 0	0 0 1 1 1 1 1 0 1 1
17	266	0 0 1 0 0 0 1	0 1 0 0 0 0 1 0 1 0
18	282	0 0 1 0 0 1 0	0 1 0 0 0 1 1 0 1 0
19	297	0 0 1 0 0 1 1	0 1 0 0 1 0 1 0 0 1
20	312	0 0 1 0 1 0 0	0 1 0 0 1 1 1 0 0 0
21	327	0 0 1 0 1 0 1	0 1 0 1 0 0 0 1 1 1
22	342	0 0 1 0 1 1 0	0 1 0 1 0 1 0 1 1 0
23	357	0 0 1 0 1 1 1	0 1 0 1 1 0 0 1 0 1
24	372	0 0 1 1 0 0 0	0 1 0 1 1 1 0 1 0 0
25	386	0 0 1 1 0 0 1	0 1 1 0 0 0 0 0 1 0
26	401	0 0 1 1 0 1 0	0 1 1 0 0 1 0 0 0 1
27	415	0 0 1 1 0 1 1	0 1 1 0 0 1 1 1 1 1
28	430	0 0 1 1 1 0 0	0 1 1 0 1 0 1 1 1 0
29	444	0 0 1 1 1 0 1	0 1 1 0 1 1 1 1 0 0
30	458	0 0 1 1 1 1 0	0 1 1 1 0 0 1 0 1 0
31	472	0 0 1 1 1 1 1	0 1 1 1 0 1 1 0 0 0
32	486	0 1 0 0 0 0 0	0 1 1 1 1 0 0 1 1 0
33	500	0 1 0 0 0 0 1	0 1 1 1 1 1 0 1 0 0
34	514	0 1 0 0 0 1 0	1 0 0 0 0 0 0 0 1 0
35	527	0 1 0 0 0 1 1	1 0 0 0 0 0 1 1 1 1
36	541	0 1 0 0 1 0 0	1 0 0 0 0 1 1 1 0 1
37	554	0 1 0 0 1 0 1	1 0 0 0 1 0 1 0 1 0
38	567	0 1 0 0 1 1 0	1 0 0 0 1 1 0 1 1 1
39	580	0 1 0 0 1 1 1	1 0 0 1 0 0 0 1 0 0
40	593	0 1 0 1 0 0 0	1 0 0 1 0 1 0 0 0 1
41	606	0 1 0 1 0 0 1	1 0 0 1 0 1 1 1 1 0
42	618	0 1 0 1 0 1 0	1 0 0 1 1 0 1 0 1 0
43	631	0 1 0 1 0 1 1	1 0 0 1 1 1 0 1 1 1
44	643	0 1 0 1 1 0 0	1 0 1 0 0 0 0 0 1 1
45	655	0 1 0 1 1 0 1	1 0 1 0 0 0 1 1 1 1
46	667	0 1 0 1 1 1 0	1 0 1 0 0 1 1 0 1 1
47	679	0 1 0 1 1 1 1	1 0 1 0 1 0 0 1 1 1
48	690	0 1 1 0 0 0 0	1 0 1 0 1 1 0 0 1 0
49	701	0 1 1 0 0 0 1	1 0 1 0 1 1 1 1 0 1
50	713	0 1 1 0 0 1 0	1 0 1 1 0 0 1 0 0 1
51	724	0 1 1 0 0 1 1	1 0 1 1 0 1 0 1 0 0
52	735	0 1 1 0 1 0 0	1 0 1 1 0 1 1 1 1 1
53	745	0 1 1 0 1 0 1	1 0 1 1 1 0 1 0 0 1
54	756	0 1 1 0 1 1 0	1 0 1 1 1 1 0 1 0 0
55	766	0 1 1 0 1 1 1	1 0 1 1 1 1 1 1 1 0
56	776	0 1 1 1 0 0 0	1 1 0 0 0 0 1 0 0 0
57	786	0 1 1 1 0 0 1	1 1 0 0 0 1 0 0 1 0
58	796	0 1 1 1 0 1 0	1 1 0 0 0 1 1 1 0 0
59	805	0 1 1 1 0 1 1	1 1 0 0 1 0 0 1 0 1
60	815	0 1 1 1 1 0 0	1 1 0 0 1 0 1 1 1 1
61	824	0 1 1 1 1 0 1	1 1 0 0 1 1 1 0 0 0
62	833	0 1 1 1 1 1 0	1 1 0 1 0 0 0 0 0 1
63	841	0 1 1 1 1 1 1	1 1 0 1 0 0 1 0 0 1

Wahrheits- bzw. PLA-Tabelle			
i	$f_{\mathrm{int}}(i)$	$\mathrm{BIN}_7\,(i)$ $\xrightarrow{f_{\mathrm{tab}}}$	$\mathrm{BIN}_{10}\,(f_{\mathrm{int}}\,(i))$
X	Y	$x_6 x_5 x_4 x_3 x_2 x_1 x_0$	$y_9 y_8 y_7 y_6 y_5 y_4 y_3 y_2 y_1 y_0$
64	850	1 0 0 0 0 0 0	1 1 0 1 0 1 0 0 1 0
65	858	1 0 0 0 0 0 1	1 1 0 1 0 1 1 0 1 0
66	866	1 0 0 0 0 1 0	1 1 0 1 1 0 0 0 1 0
67	874	1 0 0 0 0 1 1	1 1 0 1 1 0 1 0 1 0
68	881	1 0 0 0 1 0 0	1 1 0 1 1 1 0 0 0 1
69	889	1 0 0 0 1 0 1	1 1 0 1 1 1 1 0 0 1
70	896	1 0 0 0 1 1 0	1 1 1 0 0 0 0 0 0 0
71	903	1 0 0 0 1 1 1	1 1 1 0 0 0 0 1 1 1
72	910	1 0 0 1 0 0 0	1 1 1 0 0 0 1 1 1 0
73	916	1 0 0 1 0 0 1	1 1 1 0 0 1 0 1 0 0
74	922	1 0 0 1 0 1 0	1 1 1 0 0 1 1 0 1 0
75	928	1 0 0 1 0 1 1	1 1 1 0 1 0 0 0 0 0
76	934	1 0 0 1 1 0 0	1 1 1 0 1 0 0 1 1 0
77	940	1 0 0 1 1 0 1	1 1 1 0 1 0 1 1 0 0
78	945	1 0 0 1 1 1 0	1 1 1 0 1 1 0 0 0 1
79	950	1 0 0 1 1 1 1	1 1 1 0 1 1 0 1 1 0
80	955	1 0 1 0 0 0 0	1 1 1 0 1 1 1 0 1 1
81	959	1 0 1 0 0 0 1	1 1 1 0 1 1 1 1 1 1
82	964	1 0 1 0 0 1 0	1 1 1 1 0 0 0 1 0 0
83	968	1 0 1 0 0 1 1	1 1 1 1 0 0 1 0 0 0
84	972	1 0 1 0 1 0 0	1 1 1 1 0 0 1 1 0 0
85	975	1 0 1 0 1 0 1	1 1 1 1 0 0 1 1 1 1
86	979	1 0 1 0 1 1 0	1 1 1 1 0 1 0 0 1 1
87	982	1 0 1 0 1 1 1	1 1 1 1 0 1 0 1 1 0
88	985	1 0 1 1 0 0 0	1 1 1 1 0 1 1 0 0 1
89	987	1 0 1 1 0 0 1	1 1 1 1 0 1 1 0 1 1
90	990	1 0 1 1 0 1 0	1 1 1 1 0 1 1 1 1 0
91	992	1 0 1 1 0 1 1	1 1 1 1 1 0 0 0 0 0
92	994	1 0 1 1 1 0 0	1 1 1 1 1 0 0 0 1 0
93	995	1 0 1 1 1 0 1	1 1 1 1 1 0 0 0 1 1
94	997	1 0 1 1 1 1 0	1 1 1 1 1 0 0 1 0 1
95	998	1 0 1 1 1 1 1	1 1 1 1 1 0 0 1 1 0
96	999	1 1 0 0 0 0 0	1 1 1 1 1 0 0 1 1 1
97	999	1 1 0 0 0 0 1	1 1 1 1 1 0 0 1 1 1
98	1000	1 1 0 0 0 1 0	1 1 1 1 1 0 1 0 0 0
99	1000	1 1 0 0 0 1 1	1 1 1 1 1 0 1 0 0 0
100	-	1 1 0 0 1 0 0	* * * * * * * * * *
101	-	1 1 0 0 1 0 1	* * * * * * * * * *
102	-	1 1 0 0 1 1 0	* * * * * * * * * *
103	-	1 1 0 0 1 1 1	* * * * * * * * * *
104	-	1 1 0 1 0 0 0	* * * * * * * * * *
105	-	1 1 0 1 0 0 1	* * * * * * * * * *
106	-	1 1 0 1 0 1 0	* * * * * * * * * *
107	-	1 1 0 1 0 1 1	* * * * * * * * * *
108	-	1 1 0 1 1 0 0	* * * * * * * * * *
109	-	1 1 0 1 1 0 1	* * * * * * * * * *
110	-	1 1 0 1 1 1 0	* * * * * * * * * *
111	-	1 1 0 1 1 1 1	* * * * * * * * * *
112	-	1 1 1 0 0 0 0	* * * * * * * * * *
113	-	1 1 1 0 0 0 1	* * * * * * * * * *
114	-	1 1 1 0 0 1 0	* * * * * * * * * *
115	-	1 1 1 0 0 1 1	* * * * * * * * * *
116	-	1 1 1 0 1 0 0	* * * * * * * * * *
117	-	1 1 1 0 1 0 1	* * * * * * * * * *
118	-	1 1 1 0 1 1 0	* * * * * * * * * *
119	-	1 1 1 0 1 1 1	* * * * * * * * * *
120	-	1 1 1 1 0 0 0	* * * * * * * * * *
121	-	1 1 1 1 0 0 1	* * * * * * * * * *
122	-	1 1 1 1 0 1 0	* * * * * * * * * *
123	-	1 1 1 1 0 1 1	* * * * * * * * * *
124	-	1 1 1 1 1 0 0	* * * * * * * * * *
125	-	1 1 1 1 1 0 1	* * * * * * * * * *
126	-	1 1 1 1 1 1 0	* * * * * * * * * *
127	-	1 1 1 1 1 1 1	* * * * * * * * * *

Tabelle B.1: Beispiel zur Spezifikation einer PLA-Tabelle für die Sinus-Viertelwelle

Anhang C

HOPE — Dokumentation

C.1 C-Quellcode: Die Module

Zur Erleichterung des Überblickes über die Dateien des *C*-Quellcodes, wird im folgenden für jedes der zwölf Module eine grobe Charakterisierung gegeben. Dabei ist dies anhand einer Analyse der Header-Dateien `*.h` geschehen, wobei zu bemerken ist, daß die Namensgebung der Funktionen überwiegend einem Prinzip folgt: Der komplette Name setzt sich aus eine Bezeichnung für die individuelle Funktion und die des Moduls nach dem Schema *function_module* zusammen; dieses dient zur leichten Auffindbarkeit der Funktionen.

`bin.c` Operationen auf binären Tabellen bzw. Objekten.

`cli.c` Benutzerschnittstelle *Command Line Interface*: Dialog mit dem Benutzer.

`esp.c` Treiber für ESPRESSO.

`hop.c` Hauptmodul von HOPE: Initialisierung des Systems und Start der Applikation.

`int.c` Operationen für Intervalle.

`mix.c` Gemixte Routinen für verschiedene Zwecke.

`mth.c` Treiber für Methoden.

`num.c` Operationen bzw. Methoden auf numerischen Tabellen und Objekten.

`obj.c` Objekt-Handling.

`set.c` Set-Handling.

`trm.c` Operationen auf Term-Mengen.

`val.c` Binäre Methoden, nur für Werte (engl. *Values*), nicht für Intervalle.

C.2 Erstellen der ausführbaren Programm-Datei

Bei der Erzeugung einer ausführbaren Programmcode-Datei, deren Quelltexte aus mehreren Modulen besteht, ist der Einsatz von Projektverwaltungsmechanismen zu empfehlen. Diese berücksichtigen automatisch die zwischen den Dateien herrschenden Beziehungen und steuern effektiv entsprechend die Übersetzungs- und Bindevorgänge. Bekanntester Vertreter derartiger Systeme ist das Projektverwaltungssystem `Make`.

C.2.1 MAKE-kompatible Projektverwaltung

Neben den im vorhergehenden Abschnitt C.1 (S. 159) aufgeführten *C*-Modulen, die jeweils aus einer Header-Datei `.h` und einer Quelltext-Datei `.c` bestehen, sind zur Steuerung des Übersetzungsvorgangs zwei `Make`-kompatible Dateien erstellt worden, die sich in dem zugrunde liegenden Betriebssystem und den verwendeten Werkzeugen unterscheiden; die Code-Generierung geschieht durch die folgenden Aufrufe:

`make -f unix_gnu.hop` Code-Generierung mit GNU-*C*-Compiler/-Linker unter UNIX.

`make -f tos_pure.hop` Code-Generierung mit PURE-*C*-Compiler/-Linker unter ATARI-TOS.

Für andere Betriebssyteme können entsprechende `Make`-kompatible Dateien anhand der vorhandenen erstellt werden.

C.2.2 Abhängigkeiten der Modul-Dateien

Bei nicht `Make`-kompatiblen Projektverwaltungsmechanismen sind die in Tabelle C.1 (S. 160) aufgeführten Abhängigkeiten der einzelnen Module zu berücksichtigen. Links von der vertikalen Linie stehen für jedes Modul die Namen der Objekt- sowie der zugehörigen Quelltext-Datei; rechts sind die jeweils von den Modulen benötigten Header-Dateien aufgeführt.

Abhängigkeiten der Objekte von Quellen und Header													
Objekt	Quelle	Benötigte Header-Dateien											
bin.o	bin.c	bin.h	cli.h	esp.h	hop.h		mix.h		num.h	obj.h		trm.h	
cli.o	cli.c		cli.h		hop.h			mth.h		obj.h	set.h		
esp.o	esp.c	bin.h	cli.h	esp.h	hop.h		mix.h					trm.h	
hop.o	hop.c	bin.h	cli.h	esp.h	hop.h	int.h	mix.h	mth.h	num.h	obj.h	set.h	trm.h	val.h
int.o	int.c	bin.h	cli.h		hop.h	int.h	mix.h		num.h	obj.h			
mix.o	mix.c	bin.h			hop.h		mix.h			obj.h			
mth.o	mth.c	bin.h	cli.h	esp.h	hop.h	int.h	mix.h	mth.h	num.h	obj.h	set.h	trm.h	val.h
num.o	num.c	bin.h	cli.h		hop.h		mix.h		num.h	obj.h			
obj.o	obj.c	bin.h	cli.h	esp.h	hop.h	int.h	mix.h		num.h	obj.h	set.h	trm.h	val.h
set.o	set.c	bin.h	cli.h		hop.h	int.h	mix.h		num.h	obj.h	set.h	trm.h	val.h
trm.o	trm.c	bin.h			hop.h		mix.h					trm.h	
val.o	val.c	bin.h		esp.h	hop.h		mix.h			obj.h		trm.h	val.h

Tabelle C.1: Abhängigkeiten der einzelnen Module von `.h`-Dateien

C.3 Datei-Formate

In Hope werden folgende Datei-Formate bzw. -Extensionen unterstützt:

`.esp` Binär-Objekt: Espresso-Format für PLA-Tabellen; bei Espresso nicht bindend.

`.pla` Binär-Objekt: ES2-Format für PLA-Tabellen; bei Solo-2030 auch erforderlich.

`.num` Numerik-Objekt: Numerische Tabellen mit reellen Werten.

`.set` Hope-Set: Enthält von Hope erstelltes `Set`.

`.tex` Binär-Objekt: LaTeX-Format zur Dokumentation von PLA-Tabellen.

C.3.1 Format für Numerik-Objekte

Für Numerik-Objekte existiert in Hope ein einfaches Datei-Format mit nur zwei Direktiven.

Datei-Format für numerische Objekte in HOPE

```
; HOPE-Set: 'hope'
; ObjectId: 001
;
; ObjectId   Specification-Method
; -------------------------------------
;      001: Math Fct_Expo(16)=255
;
.ENTRIES      16
.MAXIMUM      255
;
    0 1
    1 1.446892
    2 2.093495
    3 3.029061
    4 4.382723
    5 6.341326
    6 9.175211
    7 13.275537
    8 19.208263
    9 27.792276
   10 40.212412
   11 58.183003
   12 84.184501
   13 121.805853
   14 176.239872
   15 255
```

Beschreibung der Direktiven:

`;`	Kommentarsymbol
`.ENTRIES`	Anzahl der numerischen Einträge
`.MAXIMUM`	maximaler Wert in den Einträgen

Anschließend sind zeilenweise *Index*, *Leerzeichen* und *Eintrag* gelistet. Näheres in den Quelltext-Dateien `num.c` bzw. `num.h` .

C.3.2 Formate für Binär-Objekte

Für Binär-Objekte existieren in Hope zunächst zwei Datei-Formate, zum einen das Espresso- und zum anderen das ES2-Format für PLA-Tabellen. Die Datei `tab.esp` (S. 163, oben) ist an ein PLA-Beispiel aus der Dokumentation zum Espresso-Paket angelehnt; lediglich die unwesentliche `.p`-Direktive und die `.e`-Endemarke sowie Kommentarsymbole wurden zur Vervollständigung ergänzt. Zum Vergleich ist dasselbe PLA im ES2-Format wiedergegeben (S. 163, unten).

Zusätzlich für die Dokumentation von PLA-Tabellen bietet Hope ein zum Textsatzsystem LaTeX kompatibles Ausgabeformat an. Auf Seite 162 ist links die LaTeX-Datei und rechts die entsprechend gesetzte Tabelle abgebildet.

LaTeX-Format für binäre HOPE-Objekte

Von Hope erstellte LaTeX-Datei eines PLAs (links) und die daraus gesetzte Tabelle (rechts).

```
% TeX-File: tab.tex generated by HOPE as LaTeX table, just \input{tab}
% HOPE-Set: hopetest  In/Out-Format:  4x3-Bits
% ObjectId: 001       Terms: 11       Espresso-Quality: Strong
%
% ObjectId  Identity (Tool applied)
%------------------------------------------------------------
%       001: External File(tab.esp)
%
\begin{scriptsize}
\def\0{\mbox{\tt0}}\def\1{\mbox{\tt1}}\def\2{\mbox{\tt*}}
\setlength{\doublerulesep}{.4mm}
\renewcommand{\arraystretch}{.6}
\begin{tabular}{|*{2}{@{\hspace{1ex}}r}@{\hspace{2ex}}%
|@{\hspace{2ex}}c*{3}{@{\/}c}|@{\hspace{2ex}}c*{2}{@{\/}c}|}
\hline
\multicolumn{9}{|c|}{%
 \small\sf File(tab.esp)%
 \rule[-1ex]{0mm}{3.5ex}} \\
\hline\hline
\rule[-1ex]{0em}{3.5ex}%
$X$& $Y$&
 $x_{3}$& $x_{2}$& $x_{1}$& $x_{0}$& %
 $y_{2}$& $y_{1}$& $y_0$\\
\hline
\rule{0em}{2ex}%
    0&       0&  \0&\0&\0&\0& \0&\0&\0\\
    1&       1&  \0&\0&\0&\1& \0&\0&\1\\
    2&       2&  \0&\0&\1&\0& \0&\1&\0\\
    3&       3&  \0&\0&\1&\1& \0&\1&\1\\
    4&       1&  \0&\1&\0&\0& \0&\0&\1\\
    5&       2&  \0&\1&\0&\1& \0&\1&\0\\
    6&       3&  \0&\1&\1&\0& \0&\1&\1\\
    7&       4&  \0&\1&\1&\1& \1&\0&\0\\
    8&       2&  \1&\0&\0&\0& \0&\1&\0\\
    9&       3&  \1&\0&\0&\1& \0&\1&\1\\
   10&       4&  \1&\0&\1&\0& \1&\0&\0\\
   11&       5&  \1&\0&\1&\1& \1&\0&\1\\
   12&       3&  \1&\1&\0&\0& \0&\1&\1\\
   13&       4&  \1&\1&\0&\1& \1&\0&\0\\
   14&       5&  \1&\1&\1&\0& \1&\0&\1\\
   15&       6&  \1&\1&\1&\1& \1&\1&\0\\
\hline
\end{tabular}\end{scriptsize}%
%
\endinput
% -eof-
```

LaTeX erzeugt aus dem Text der nebenstehenden Datei die folgende PLA-Tabelle:

File(tab.esp)								
X	Y	x_3	x_2	x_1	x_0	y_2	y_1	y_0
0	0	0	0	0	0	0	0	0
1	1	0	0	0	1	0	0	1
2	2	0	0	1	0	0	1	0
3	3	0	0	1	1	0	1	1
4	1	0	1	0	0	0	0	1
5	2	0	1	0	1	0	1	0
6	3	0	1	1	0	0	1	1
7	4	0	1	1	1	1	0	0
8	2	1	0	0	0	0	1	0
9	3	1	0	0	1	0	1	1
10	4	1	0	1	0	1	0	0
11	5	1	0	1	1	1	0	1
12	3	1	1	0	0	0	1	1
13	4	1	1	0	1	1	0	0
14	5	1	1	1	0	1	0	1
15	6	1	1	1	1	1	1	0

Näheres zu LaTeX findet sich u. a. in [Kop-95]. Die PLA-Tabelle des *Sinus*-Beispiels (vgl. Abbildung B.1, S. 158) wurde ausgehend von einer durch Hope erzeugten Datei erstellt; dabei wurde lediglich die Aufspaltung in zwei Hälften ergänzt sowie geringfügige Änderungen in der Beschriftung eingearbeitet.

ESPRESSO-Format für binäre HOPE-Objekte

```
# file: tab.esp
#
# 2-bit by 2-bit binary adder (with no carry input)
#
.i 4
.o 3
.p 11
#
0000  000
0001  001
0010  010
0011  011
0100  001
0101  010
0110  011
0111  100
1000  010
1001  011
1010  100
1011  101
1100  011
1101  100
1110  101
1111  110
.e
```

Beschreibung der Direktiven:

`#` Kommentarsymbol
`.i` Anzahl der PLA-Eingangsleitungen (Inputs)
`.o` Anzahl der PLA-Ausgangsleitungen (Outputs)
`.p` Optionale Anzahl der Produktterme (Productterms)
`.e` Optionale Endemarke

Das EXPRESSO-Format ist kompatibel mit dem *Berkeley Standard Format for the Physical Description of a PLA*. Weitere Dokumentation dazu ist in [Bra*82b] und im Softwarepaket zu ESPRESSO, Version #2.3 vom 31.01.1988 zu finden; dort insbesondere in den Manual-Dateien `espresso.1` und `espresso.5` [Esp-88].

ES2-Format für binäre HOPE-Objekte

```
; file: tab.pla
;
; 2-bit by 2-bit binary adder (with no carry input)
;
.INPUTS 4
.OUTPUTS 3
.MINTERMS 11
.PROG
;
0000  000
0001  001
0010  010
0011  011
0100  001
0101  010
0110  011
0111  100
1000  010
1001  011
1010  100
1011  101
1100  011
1101  100
1110  101
1111  110
```

Beschreibung der Direktiven:

`;`	Kommentarsymbol
`.INPUTS`	Anzahl der PLA-Eingangsleitungen
`.OUTPUTS`	Anzahl der PLA-Ausgangsleitungen
`.MINTERMS`	Optionale Anzahl der Minterme
`.PROG`	Start der Termzeilen

Weitere Dokumentation zum ES2-Format für PLA-Dateien findet sich in [ESS-92, Abschnitt 4.2.4, Seite 4–10].

C.3.3 Format für Set-Dateien

Zur Aufnahme kompletter `Sets`, die mit Hope erarbeitet worden sind, dient das folgende Datei-Format mit der Extension `.set` .

`Set`-Datei zur Aufnahme von HOPE-Objekten

```
; Set       hope
.OBJECTS    10
.INPUTS     4
.OUTPUTS    8
.REFERENCE  ---
.STATUS     1c1R2 ; Order|Cost|Deviation|Relate|Quality
.COMMENT
;
;id fl rel org tool trms litls one dcs idev longdevi par1 par2 par3 long-par name...
001 50 999 999 0252 0000 00000 -01 -01 0000 00000000 0010 0000 0000 000000ff tab
002 98 999 001 08d1 0000 00000 044 000 0000 00000000 0010 0000 0000 00000000
003 98 999 001 08d2 0000 00000 045 000 0000 00000000 0010 0000 0000 00000000
004 98 999 001 08d3 0000 00000 049 000 0000 00000000 0010 0000 0000 00000000
005 af 999 001 10e3 0000 00000 035 007 0000 00000000 0000 0000 0000 00000000
006 8a 999 005 20b1 0014 00066 045 000 0000 0064c000 000e 0000 0000 00000000
007 8a 999 005 20b2 0014 00070 049 000 0000 0064c000 000e 0000 0000 00000000
008 8a 999 005 20b3 0014 00069 042 000 0000 0064c000 000e 0000 0000 00000000
009 8a 999 005 20b4 0014 00067 042 007 0000 00634000 0007 0000 0000 00000000
010 8a 999 005 20b5 0016 00080 023 000 0006 00e98210 000e 0000 0000 00000000
```

Beschreibung der Direktiven:

`;`	Kommentarsymbol
`.OBJECTS`	Anzahl der Objekte
`.INPUTS`	Anzahl der PLA-Eingangsleitungen
`.OUTPUTS`	Anzahl der PLA-Ausgangsleitungen
`.REFERENCE`	max. zwei `Ids`, sonst `---` statt der ersten `Id`
`.STATUS`	String (*siehe* Quelltextdatei `set.c`)
`.COMMENT`	optionale Kommentarzeile (max. 60 Zeichen)

Nachfolgend wird pro Zeile ein `Object` gelistet; teilweise werden die Komponenten in hexadezimaler Darstellung aufgeführt. Nähere Angaben können den Quelltext-Dateien `set.c` bzw. `set.h` entnommen werden.

C.4 Inhalt der beiden Disketten

Der *C*-Quelltext für Hope und die angepasste Espresso-Version ist auf den beiden beigefügten Disketten abgespeichert, die beide im IBM-DD-Format (720KBytes) formatiert sind. Ferner befinden sich auf den Disketten die evaluierten Beispiele sowie eine unter Atari-tos ausführbare Programm-Datei von Hope. Jede Diskette enthält eine Datei, die den Inhalt der jeweiligen Diskette angibt; beide sind nachfolgend wiedergegeben.

Datei: disk-a.dir

```
disk-a.dir

a:bin:
hope.ttp*

a:hop:
bin.c        hop.c        mix.h        obj.c        trm.c
bin.h        hop.h        mth.c        obj.h        trm.h
cli.c        int.c        mth.h        set.c        unix_gnu.hop
cli.h        int.h        num.c        set.h        val.c
esp.c        mix.c        num.h        tos_pure.hop val.h
esp.h

634880  bytes in        28 file(-s)
```

Datei: disk-b.dir

```
disk-b.dir

a:esp:
ansi.h       dominate.c   hack.c       pair.c       sminterf.c
changes.txt  equiv.c      indep.c      part.c       solution.c
cofactor.c   espresso.c   irred.c      port.h       sparse.c
cols.c       espresso.h   main.c       portutil.c   sparse.h
compl.c      essen.c      main.h       portutil.h   sparse_i.h
contain.c    exact.c      map.c        primes.c     tos_pure.esp
cubestr.c    expand.c     matrix.c     reduce.c     unate.c
cvrin.c      gasp.c       mincov.c     rows.c       unix_gnu.esp
cvrm.c       getopt.c     mincov.h     set.c        utility.h
cvrmisc.c    gimpel.c     mincov_i.h   setc.c       verify.c
cvrout.c     globals.c    opo.c        sharp.c

a:exa:
expo.set     hbolhyp.set  logaupdn.set sin4bin.set  sineabs2.set
expoabs2.set hbolrel5.set root.set     sin4can.set  sinehyp.set
expohyp.set  hbolupdn.set rootabs2.set sin4con.set  sinerel5.set
exporel5.set loga.set     roothyp.set  sin4fav.set  sineupdn.set
expoupdn.set logaabs2.set rootrel5.set sin4hit.set  test.set
hbol.set     logahyp.set  rootupdn.set sin4num.set  toplist.txt
hbolabs2.set logarel5.set sin4.set     sine.set

626688  bytes in        89 file(-s)
```

Anhang D

HOPE — Benutzeranleitung

Im folgenden wird die Bedienung des Werkzeugs HOPE vorgestellt. Dabei wird die Kenntnis der in Abschnitt 7.2.2 (S. 141f) vorgestellten Systemkomponenten vorausgesetzt. Dennoch sei hier mit Abbildung D.1 (S. 166) nochmals die Grafik von Abbildung 7.2 (S. 141) wiederholt dargestellt.

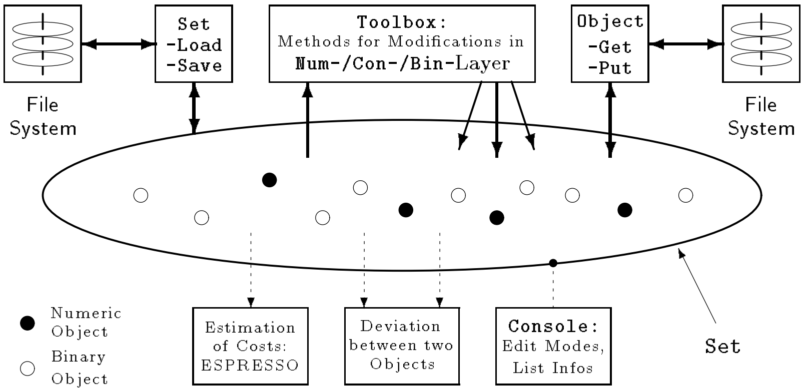

Abbildung D.1: Systemkomponenten in HOPE

D.1 Starten von HOPE

Bei HOPE handelt es sich um eine Kommandozeilen-orientierte Benutzerschnittstelle; das bedeutet, daß HOPE in der Regel als Applikation einer sogenannten `Shell` gestartet wird. Derartige Start-Programme bilden meistens eine Umgebung für Applikationen, das sogenannte `Environment`. Dort kann mit Hilfe von extra definierten Variablen die Arbeitsweise der Applikationen beeinflußt werden. Für HOPE sind folgende `Environment`-Variablen relevant:

`HOPEHOME` Wenn diese Variable nicht definiert sein sollte, wird HOPE noch nach der folgenden suchen.

`hopehome` Diese beiden optionalen Variablen definieren den Default-Pfad für die `.set`-Dateien von HOPE, die die sogenannten `Set`s aufnehmen.

`DEBUG` Diese ebenfalls optionale Variable wird zum setzen des Debug-Modus' von HOPE verwendet. Dafür ist Voraussetzung, daß bei der Übersetzung der *C*-Module das `DEBUG`-Makro definiert war (vgl. `Make`-Dateien in Abschnitt C.2.1, S. 160).

Die in Abschnitt C.2.1 (S. 160) definierten `Make`-kompatiblen Projektverwaltungsmechanismen erzeugen abschließend eine ausführbare Programm-Datei mit dem Namen `hope`, so daß sich das Werkzeug mit dem folgenden Kommando aufrufen läßt:

`hope` [*filename*`(.set)`]

Die optionale Angabe der Datei *filename* ohne Extension ist für das sofortige Laden eines HOPE-`Sets`; dabei wird der ggf. vorangestellte Pfad für alle weiteren Datei-Zugriffe der begonnenen HOPE-Sitzung verwendet. Die Pfad-Einstellung kann selbstverständlich explizit verändert werden. Es bietet sich generell an, mit der `Shell` das Verzeichnis der `Set`-Dateien als aktuelles Verzeichnis zu setzen.

D.2 Funktionalität und Bedienung

Obwohl HOPE nicht mit Hilfe einer grafischen Benutzeroberfläche bzw. einer GUI (engl. *Grafical User Interface*) gesteuert wird, sondern stattdessen über die Kommandozeile eines CLI (engl. *Command Line Interface*) bedient wird, erfolgt die Bedienung in Anlehnung an GUI-ähnliche Menü-Strukturen. Für eine übersichtliche Darstellung der Menüs ist es erforderlich, daß die HOPE umgebende `Shell` mindestens 80 Zeichen pro Zeile darstellt.

D.2.1 Prinzip der Menü-Einträge

HOPE präsentiert die Menüs in einzeiliger Form; die Darstellung der HOPE umgebenden `Shell` sollte daher mindestens 80 Zeichen pro Zeile ermöglichen. Ein Menü bietet innerhalb einer Zeile seine Einträge an, die zur Separierung durch ein oder mehrere ␣ (*Blanks*) voneinander getrennt sind.

Prinzip eines HOPE-Menüs
`-Begriff:  Eintrag-1  eIntrag-2  eiNtrag-3  einTrag-4  eintRag-5  eintrAg-6`

In der Regel weist der erste Eintrag auf den *Begriff* hin, zu dem mit den anschließenden Einträge eine Auswahl besteht. Er dient nur zur Orientierung und kann *nicht* selektiert werden; dies wird durch einen vorangestellten Bindestrich `-` angedeutet.

D.2.2 Selektion von Menü-Einträgen durch Kennbuchstaben

Die einzelnen Menü-Einträge sind durch Eingabe des jeweiligen Kennbuchstabens zu selektieren, der jedem Menü-Eintrag eindeutig zugeordnet ist. Ein Kennbuchstabe eines Menü-Eintrags ist der *erste Großbuchstabe* innerhalb des Menü-Eintrags; die Eingabe des Kennbuchstabens kann allerdings auch als Kleinbuchstabe erfolgen, so daß die Betätigung der $\boxed{\text{Shift}}$-Taste nicht erforderlich ist.

Die Auswahl wird durch Betätigen der Return-Taste abgeschlossen. Soll allerdings bei einem präsentierten Menü *keine* Auswahl von den vorhandenen Menü-Einträgen erfolgen, so kann ohne Eingabe eines Zeichens durch Betätigen der Return-Taste zum nachfolgend erläuterten Hauptmenü zurückgekehrt werden; in seltenen Fällen ist vorher die Space-Taste zu betätigen.

D.2.3 Kurzform der Menüauswahl durch Kennbuchstaben-Sequenz

Für den versierten Benutzer von HOPE, dem insbesondere die Struktur der Menüs vertraut ist, existiert bei der Selektion von Aktionen, die vom Hauptmenü aus nur durch mehrere Menüebenen hindurch erreicht werden können, die Möglichkeit, die jeweilige Aktion durch Eingabe der kompletten Sequenz der erforderlichen Kennbuchstaben direkt auszulösen.

Vom Hauptmenü ausgehend kann beispielsweise durch Selektion der Menü-Einträge `Set`, `Edit` und `Set-Comment` der in der `.set`-Datei abgespeicherte Kommentar-String abgerufen bzw. verändert werden. Die Kurzform lautet für das direkte Erreichen dieses Dialoges `s e s` und muß nur noch durch Return abgeschlossen werden.

Diese Art der Kurzform ist nur innerhalb der Menü-Struktur implementiert, d. h. sie gilt nicht für Dialoge. Sie erfordert eine gute Kenntnis der nachfolgend erläuterten Menü-Struktur.

D.2.4 Hauptmenü

Das Hauptmenü von HOPE erscheint nach dem Start und nach der Beendigung von selektierten Aktionen; es stellt damit eine Art Grundstellung des Systems dar.

Startmeldung und HOPE-Menü

```
HOPE - _Heuristic _Optimization of _PLA-Specifications to Minimize _Expense.
Version 1.0 - (c) 1995 by Andreas H. Krause.

Choose Menu-Item with 1st Upper-Case-Letter or quit Menu with RETURN or BLANK.

-HOPE: Info  Set  Object  Toolbox  Configure  Quit
-------------------------------------------------------------------------------
Your Choice? ->
```

Durch Selektion der Menü-Einträge `Set`, `Object`, `Toolbox` oder `Configure` mit Hilfe der `s`-, `o`-, `t`- bzw. `c`-Taste können weitere Sub-Menüs aufgerufen werden, die im einzelnen in den folgenden Abschnitten erläutert werden.

Die Menü-Einträge `Info` und `Quit` führen bereits abschließend zu Aktionen, d. h. es folgen darauf keine weiteren Sub-Menüs. Während beim Betätigen der `i`-Taste allgemeine Informationen zu HOPE und dem Systemstatus ausgegeben werden, kann mit Hilfe der `q`-Taste die HOPE-Sitzung beendet werden; falls die Daten des aktuellen `Sets` noch nicht gesichert sind, folgt vorher eine Sicherheitsabfrage.

D.2.5 Menü: Set

```
Menü: Set
-Set:  New  Load  Save  Info  View  Edit  Costs  Deviation
------------------------------------------------------------------------------
Your Choice? ->
```

Das `Set`-Menü faßt alle auf das gesamte `Set` bezogenen Aktionen zusammen. Im einzelnen ergeben sich folgende Möglichkeiten:

`New` — *Dialog* zum Anlegen eines neuen `Sets`. Anzugeben sind ein optionaler *Name* sowie die Anzahl der PLA-Ein- und -Ausgänge.

`Load` — *Dialog* zum Laden eines von HOPE bereits erstellten `Sets`. Anzugeben ist der *Name* der `.set`-Datei ohne Pfad und Extension. Zur Einstellung eines neuen Zugriffs-Pfades in HOPE kann vom Hauptmenü ausgehend, durch die Eingabesequenz `c d`, ein entsprechender Dialog aufgerufen werden.

`Save` — *Dialog* zum Abspeichern des aktuellen `Sets`. Es wird der aktuelle *Name* als Vorgabe präsentiert; er kann durch Return beibehalten oder durch Neueingabe eines anderen überschrieben werden. Die `.set`-Datei wird unter dem aktuellen Pfad abgespeichert; HOPE gibt den benutzten Datei-Namen inklusive Pfad protokollierend aus.

`Info` — *Ausgabe* von Informationen über ein `Set`.

`View` — *Menü* für die weitere Selektion von Aktionen; siehe Abschnitt D.2.9 (S. 173).

`Edit` — *Menü* für die weitere Selektion von Aktionen; siehe Abschnitt D.2.10 (S. 173).

`Costs` — *Bestätigung* für das Starten des Logik-Minimierers ESPRESSO für alle nicht verborgenen Objekte. Tatsächlich wird dies nur für Binär-Objekte ohne Intervall-Attribut durchgeführt; dieser Vorgang kann viel Laufzeit beanspruchen.

`Deviation` — *Bestätigung* für das Starten der Abweichungsbestimmung für alle nicht verborgenen Objekte. Tatsächlich wird dies nur für Binär-Objekte ohne Intervall-Attribut durchgeführt. Jedes Objekt trägt dafür mit der Komponente `relid` (vgl. Abbildung 7.1, S. 138) einen Verweis auf das zugeordnete Vergleichsobjekt in sich. Näheres dazu in Abschnitt D.3.3 (S. 177).

D.2.6 Menü: Object

Menü: Object

`-Object: Select Get Put Info List Terms Hide Kill Relate Cost Deviation`
`---------------------------------------------------------------------------`
`Your Choice? ->`

Das `Object`-Menü faßt alle auf ein einzelnes `Object` bezogenen Aktionen zusammen. Im einzelnen ergeben sich folgende Möglichkeiten:

`Select` *Menü* zum Selektieren eines beliebigen Objektes; siehe Abschnitt D.2.11 (S. 174).

`Get` *Menü* zum Importieren eines Objektes; siehe Abschnitt D.2.12 (S. 175).

`Put` *Menü* zum Exportieren des selektierten Objektes; siehe Abschnitt D.2.13 (S. 175).

`Info` *Ausgabe* von Informationen über das selektierte `Object`.

`List` *Ausgabe* der Tabelle des selektierten Objektes.

`Terms` *Ausgabe* der Termmenge des selektierten Binär-Objektes; ggf. ist zunächst die Erzeugung der Termmenge durch Starten der Logik-Minimierung erforderlich.

`Hide` *Bestätigung* zum Wechseln des `Hide`-Merkmals beim selektierten Objekt. Jedes Objekt hat dafür ein Merkmal innerhalb der Komponente `flags` (vgl. Abbildung 7.1, S. 138), das zum Verbergen bei künftigen *Ausgaben* dient.
Näheres dazu in Abschnitt D.3.2 (S. 175).

`Kill` *Bestätigung* zum Löschen des selektierten Objektes; ggf. wird davor gewarnt, ein Objekt zu löschen, das in der Verkettung als Vorgänger-Objekt für andere Objekte relevant ist; diese Aktion sollte nur mit äußerster Vorsicht exekutiert werden.

`Relate` *Menü* zum Festlegen des Vergleichsobjektes (`Relate-Object`) beim selektierten Objekt; siehe Abschnitt D.2.14 (S. 175).

`Cost` *Bestätigung* für das Starten des Logik-Minimierers ESPRESSO für das selektierte Binär-Objekt (ohne Intervall-Attribut); dieser Vorgang kann viel Laufzeit beanspruchen.

`Deviation` *Bestätigung* für das Starten der Abweichungsbestimmung für das selektierte Binär-Objekt (ohne Intervall-Attribut). Jedes Objekt trägt dafür mit der Komponente `relid` (vgl. Abbildung 7.1, S. 138) einen Verweis auf das zugeordnete Vergleichsobjekt in sich.
Näheres dazu in Abschnitt D.3.3 (S. 177).

D.2.7 Menü: Toolbox

Menü: Toolbox

`-Tools/Methods:  Numeric-Layer  Conversion-Layer  Binary-Layer`
`------------------------------------------------------------------------------`
`Your Choice? ->`

Das `Toolbox`-Menü verzweigt in die drei Schichten, nämlich die Numerik-, die Konversions- und die Binär-Schicht, die alle Aktionen zusammenfassen, die mit den implementierten Methoden bewirkt werden können. Dementsprechend folgen nun drei Sub-Menüs:

Menü: Numeric-Layer

`-Numeric-Layer:  Function  Scale`
`------------------------------------------------------------------------------`
`Your Choice? ->`

Menü: Conversion-Layer

`-Conversion:  Rounding  Power-of-2  Intervals  Selection`
`------------------------------------------------------------------------------`
`Your Choice? ->`

Menü: Binary-Layer

`-Binary-Layer:  Pseudo-Inc/Dec  Similar-Entries  Hyper-Terms  Bin2Num-Utility`
`------------------------------------------------------------------------------`
`Your Choice? ->`

Erläuterungen zu den selektierbaren Methoden sind in Abschnitt 7.2.3 (S. 143f) sowie in den vorhergehenden Kapiteln 5 (S. 75ff) und 6 (S. 103ff) zu finden.

D.2.8 Menü: Configure

```
Menü: Configure
-Configure:  Directory  Object-Amount  Behaviour
-----------------------------------------------------------------------------
Your Choice? ->
```

`Directory` *Dialog* zur Eingabe eines Datei-Pfads; dies kann schrittweise durch sequentielle Eingabe der Unterverzeichnisse (ohne Schrägstrich) erfolgen oder auch komplett durch Eingabe der Beginnzeichen für Pfade im jeweiligen Betriebsystem. Für Atari-Tos und MSdos wäre letztere Möglichkeit z. B. mit der Eingabe von `a:\exa` und bei Unix mit `/users/tech9/tech_3/1krause/da/exa` gegeben.

`Object-Amount` *Ausgabe* der Maximal-Anzahl für Objekte im `Set`. Aus Gründen der Speicher-Allokation wird beim Laden eines `Sets` zur Anzahl der darin bereits vorhandenen Objekte Speicherplatz für eine bestimmte Anzahl von weiteren Objekten alloziert. Die Gesamtzahl kann durch Abspeichern des `Sets` und erneutes Laden wieder erhöht werden.

Ein *Dialog* zur Eingabe der zu allozierenden Anzahl von Objekten ist nur unmittelbar nach der Neueröffnung eines `Sets` mit `Set New` zu erhalten.

`Behaviour` *Menü* für die Einstellung von Betriebsmodi:

- `Configure Behaviour Verbosity`
  Menü für die Ausführlichkeit bei den *Ausgabe*.
- `Configure Behaviour Bell`
  Wechseln der Einstellung für die Terminalglocke zwischen `Bell=ON` und `Bell=OFF`; alle in Hope ablaufenden Aktivitäten, die mehr als fünf Sekunden Laufzeit benötigen, lösen signalgebend bei `Bell=ON` die Terminalglocke aus, welches bei `Bell=OFF` unterbleibt.
- `Configure Behaviour LOG-File`
  Wechseln der Einstellung für ein Datei-Protokoll der Hope-Stitzung zwischen `LOG-File=ON` und `LOG-File=OFF`; das Protokoll wird eine Datei geschrieben, die den Namen des `Sets` trägt und die Extension `.log` bekommt. Existiert sie bereits, so wird daran angehängt geschrieben.
- `Configure Behaviour Prompt`
  Wechseln der Einstellung für das `Set`-Listing, welches durch `Set View List` ausgegeben wird; bei `Prompt=ON` wird die *Ausgabe* nach jeweils 15 ausgegebenen Objekten angehalten, welches bei `Prompt=OFF` unterbleibt.
- `Configure Behaviour Hiding`
  Wechseln der Einstellung für das Verbergen bzw. `Hiding`.
  Näheres in Abschnitt D.3.2 (S. 175).

D.2.9 Menü: Set-View

```
Menü: View
-View: List  Tree  Hiding
-----------------------------------------------------------------------------
Your Choice? ->
```

`List` *Ausgabe* aller nicht verborgenen Objekte im `Set` als Listing.

`Tree` *Ausgabe* aller nicht verborgenen Objekte im `Set` als Baumstruktur.

`Hiding` *Menü* zur Auswahl der `Object`-Komponente, die für das Verbergen bzw. `Hiding` relevant ist.
Näheres in Abschnitt D.3.2 (S. 175).

D.2.10 Menü: Set-Edit

```
Menü: Edit
-Edit:  Cost  Deviation  Order  Quality  Set-Comment
-----------------------------------------------------------------------------
Your Choice? ->
```

`Cost` *Menü* für folgende Aktivitäten:

- Deklarierung der Referenz bzw. des `Reference-Objects`, also jenem Objekt, das für alle Binär-Objekte im `Set` die Bezugsgröße c_0 (Kosten der Referenz) zur Bestimmung der Reduzierung $r_0(c)$ (vgl. Gleichung 6.13, S. 120) angibt.
- Wechseln der Einstellung des `Cost-Estimation`-Modus. Er bestimmt, ob für Binär-Objekte unmittelbar nach ihrer Entstehung die Logik-Minimierung mit ESPRESSO erfolgen soll (`Cost-Estimation=AUTO`)[1] oder ob dies nur durch den Benutzer veranlaßt wird (`Cost-Estimation=USER`); vorgegeben ist bei jedem Systemstart von HOPE stets `Cost-Estimation=USER`.

`Deviation` *Menü* zur

- Deklarierung des Vergleichsobjektes bzw. `Relate-Object`, also jenem Objekt, das für den Modus `GLOBAL-Relating` bei der Bestimmung der Abweichungen eines Binär-Objektes als *korrekt* angesehen wird.
  Näheres in Abschnitt D.3.3 (S. 177).
- Wechseln der Einstellung des `Relating`-Modus.
  Näheres in Abschnitt D.3.3 (S. 177).
- *Menü* zur Einstellung des `Deviation-Criterions`.
  Näheres in Abschnitt D.3.4 (S. 178).

[1] Wegen des hohen Laufzeitbedarfs ist dies nur in Ausnahmefällen zu empfehlen. Im Falle der Hyp_Trm-Methode gelingt in manchen Fällen die Reproduktion der Termmengen durch ESPRESSO nicht identisch, so daß eine spätere Logik-Minimierung fehlschlägt. Es empfiehlt sich dann, die Logik-Minimierung durch `Cost-Estimation=AUTO` sofort bei der Enstehung der Hyp_Trm-Objekte zu veranlsssen. Der hohe Laufzeitbedarf dürfte aber angesichts der meistens wenig günstigen Ergebnisse der Hyp_Trm-Methode kaum zu rechtfertigen sein.

`Order` *Menü* zur Auswahl des Sortierkriteriums für die `Set`-Liste, die mit Hilfe von `Set View List` ausgegeben werden kann.

`Quality` *Menü* zur Auswahl des Modus der Minimierungsqualität von Espresso; Grundeinstellung ist der Modus `Normal`.

`Set-Comment` *Dialog* zum Einsehen bzw. Überschreiben des `Set`-Kommentar-String; empfohlen ist eine Maximal-Länge von 60 Zeichen.

D.2.11 Menü: Object-Select

Menü: `Select`

```
-Select-Obj:  Default=003  Enter-Id  Next  Previous  Origin  Top-  Bottom-List
------------------------------------------------------------------------------
Your Choice? ->
```

`Default=003` Selektiert das vorgegbene Objekt, in diesem konkreten Fall das Objekt mit der `Id 003`; im allgemeinen unterstützt Hope durch sachdienliche Vorgaben im veränderlichen `Id`-Feld den beabsichtigten Arbeitsablauf. Um falsche Selektionen zu vermeiden, sollte jedoch immer auf den eingeblendeten `Id`-Eintrag geachtet werden.

`Enter-Id` *Dialog* zur Eingabe einer `Id`-Nummer; Hope präsentiert den zulässigen Bereich.

`Next` Selektiert jenes Objekt, dessen `Id` sich aus der Inkrementierung der bei `Default` vorgebenen `Id` ergibt.

`Previous` Selektiert jenes Objekt, dessen `Id` sich aus der Dekrementierung der bei `Default` vorgebenen `Id` ergibt.

`Origin` Selektiert das Vorgänger-Objekt zu dem bei `Default` vorgebenen Objekt.

`Top-List` Selektiert jenes Objekt, das an erster Stelle in der durch `Set View List` abrufbaren Objekt-Liste steht.

`Bottom-` Selektiert jenes Objekt, das an letzter Stelle in der durch `Set View List` abrufbaren Objekt-Liste steht.

D.2.12 Menü: Object-Get

Die hier verfügbare Funktionalität dient der Importierung von *numerischen* oder *binären* Objekten (ohne Intervall-Attribut):

- *Menü* für die Festlegung des Objekt-Typs.
- *Dialog* zur Eingabe des Datei-Namens. Sofern aktuell ein Objekt selektiert ist, erscheint zuvor eine *Bestätigung*, ob das selektierte Objekt überladen werden soll oder nicht.
- *Menü* für Binär-Objekte zur Auswahl eines der drei Datei-Formate (vgl. Abschnitt C.3, S. 161).

D.2.13 Menü: Object-Put

Die hier verfügbare Funktionalität dient der Exportierung des selektierten Objektes. Es erscheint ein *Dialog* zur Eingabe des Datei-Namens. Für ein *binäres* Objekt wird zuvor allerdings ein *Menü* zur Auswahl eines der drei Datei-Formate (vgl. Abschnitt C.3, S. 161) präsentiert.

D.2.14 Menü: Object-Relate

Die hier verfügbare Funktionalität dient der Festlegung des Vergleichsobjektes beim selektierten Binär-Objekt (ohne Intervall-Attribut).
Näheres in Abschnitt D.3.3 (S. 177).

D.3 Implementierte Steuerungsmechanismen

D.3.1 Qualitätsmodi bei der Logik-Minimierung mit ESPRESSO

In HOPE ist der Minimierer ESPRESSO [Bra*82b, Bra*84] integriert. Er bietet durch unterschiedliche Zusammenstellung der internen Algorithmen fünf verschiedene Modi an, die sich zum einen in der Qualität der Minimierungsergebnisse, zum anderen aber auch in der dafür aufzuwendenden Laufzeit unterscheiden. In Tabelle D.1 (S. 176) sind die fünf Modi in abgestufter Minimierungsqualität aufgeführt, wobei nicht garantiert werden kann, ob nicht auch gelegentlich ein niedrig eingestufter Modus ein minimaleres Ergebnis erzeugt als ein höher eingestufter. Links in der Tabelle stehen zunächst die Stufe und die HOPE-interne Bezeichnung. Anschließend ist zum Vergleich die entsprechende Aufruf-Option, falls ESPRESSO als eigenständiges Werkzeug von der Kommmandozeile gestartet werden würde. Schließlich ist aus der Dokumentation zu ESPRESSO eine Erläuterung zitiert.

D.3.2 Verbergen von Objekten

Über das *Menü* `Object Select` läßt sich jedes `Object` im `Set` selektieren. Mit den durch das `Object` *Menü* präsentierten Funktionalität lassen sich Aktivitäten an einem selektierten `Object` durchführen. Für manche Aktivitäten, die vom *Menü* `Set` ausgehen ist es nicht zweckmäßig, dabei immer *alle* Objekte im `Set` mit einzubeziehen, wie z. B. bei der Bestimmung der Kosten durch `Set Costs`. Daher ist in HOPE ein Mechanismus zum Verbergen von Objekten implementiert. Zum Verbergen eines Objektes gibt es zwei Möglichkeiten:

Qualität Stufe	in HOPE	Option im Aufruf espresso [*options*] [*file*]	Erläuterung nach ESPRESSO Dokumentation [Esp-88]
1	`Fast`	`-efast`	Stop after the first EXPAND and IRREDUNDANT operations (i.e., do not iterate over the solution).
2	`Normal`	*no option*	Standard operation of ESPRESSO algorithms.
3	`Exact`	`-Dexact`	Exact minimization algorithm (guarantees a minimum number of product terms, and heuristically minimizes number of literals). Potentially expensive.
4	`Strong`	`-estrong`	Uses the alternate strategy SUPER_GASP (as a replacement for LAST_GASP) which is more expensive, but occasionally provides better results.
5	`QMC`	`-Dqm`	Exact minimization as of Quine-McCluskey.

Tabelle D.1: Rang der Qualitätsstufen des ESPRESSO-Minimierers in HOPE

- `Object Hide`:
  Gezielt kann hierdurch für ein selektiertes Objekt das Verbergen zwischen `Hidden` und `Unhidden` gewechselt werden.

- `Set View Hide`:
  Für mehrere Objekte kann durch dieses *Menü* das Verbergen zwischen dem Objekt-Merkmal `Hidden` und `Unhidden` gewechselt werden; jedes `Object` hat dieses Merkmal innerhalb der `flags`-Komponente.

Menü: `Set View Hide`

```
-Hide: Id  Costs  Deviation  Reduction  Tool  Origin  All-Hide  Un-Hide-All
----------------------------------------------------------------------------
Your Choice? ->
```

- `Id`
  Dialog zum Verbergen anhand einer `Id`-Schranke.
- `Costs`
  Dialog zum Verbergen anhand einer Kosten-Schranke.
- `Deviation`
  Dialog zum Verbergen anhand einer Abweichungs-Schranke; zu beachten ist das vorgebene Kostenkriterium, wie z. B. `MaxAbs` in `Enter Deviation to Hide Objects with more 'MaxAbs'` ... Das aktuelle Abweichungs-Kriterium kann durch `Set Edit Deviation Criterion` eingestellt werden; Vorgabe ist `MaxAbs`.
  Näheres in Abschnitt D.3.4 (S. 178).
- `Reduction`
  Dialog zum Verbergen anhand einer Reduzierungs-Schranke.
- `Tool`
  Dialog zum Verbergen anhand des angewendeten `Tools`.
- `Origin`
  Dialog zum Verbergen des Vorgänger-Objektes des anzugebenden Objektes.
- `All-Hide`
  Bestätigung zum Verbergen *aller* Objekte; macht nur Sinn, wenn anschließend Objekte gezielt wieder sichtbar gemacht werden.

- `Un-Hide-All`
  Bestätigung zum Sichtbar-Machen *aller* Objekte. Sollen die Einstellungen bezüglich des Verbergens bei den Objekten vorübergehend deaktiviert werden, so bietet sich die folgende Aktivität an.

- `Configure Behaviour Hiding`
  Wechseln der Wirksamkeit des `Hide`-Merkmals bei allen Objekten im `Set`; das Merkmal selbst bleibt erhalten.

Hinweis: Zum Sichtbar-Machen einer Teilmenge von Objekten anhand ihrer `Id`s im Bereich von $a \leq \texttt{Id} < b$ empfiehlt sich folgende Vorgehensweise:

1. `Set View Hide All-Hide`
   Verbergen *aller* Objekte.
2. `Set View Hide Id`
   Dialog zur Eingabe von b und Auslösen von `Un-Hide(?_Objects)`, um all jene Objekte wieder sichtbar zu machen, deren `Id` echt kleiner ist als b.
3. `Set View Hide Id`
   Dialog zur Eingabe von a und Auslösen von `Hide(?_Objects)`, um all jene Objekte wieder zu verbergen, deren `Id` größer oder gleich a ist.

D.3.3 Zuordnung des Relate-Object

Obwohl in HOPE mehere Möglichkeiten für die Bestimmung des Vergleichsobjektes bzw. des sognannten `Relate-Object` implementiert sind, existiert für jedes Binär-Objekt immer eine *eindeutige* Zuordnung. Zunächst gibt es zwei Kategorien für den sogenannten `Relating`-Modus, nämlich auf das gesamte `Set` oder nur ein einzelnes `Object` bezogen.

- `GLOBAL-Relating`: Allen Binär-Objekten im `Set` wird ein bestimmtes `Relate-Object` zugeordnet; dieser Modus ist immer dann sinnvoll, wenn die PLA-Tabellen von durch Methoden erzeugten Objekten mit der *ad hoc* spezifizierten PLA-Tabelle verglichen werden sollen. Dieser Modus kann durch `Set Edit Deviation LOCAL` umgeschaltet werden.

- `LOCAL-Relating`: Für alle Binär-Objekte geschieht die Zuordnung des `Relate-Object`s individuell, und zwar existieren hier wiederum zwei Alternativen, die mit dem Menü nach `Object Relate` manipuliert werden können.

  - `Static-Relate-Object`: Es kann für das selektierte Binär-Objekt (ohne Intervall-Attribut) ein beliebiges Binär-Objekt (mit oder ohne Intervall-Attribut) als Vergleichsobjekt bzw. als`Relate-Object` definiert werden; eignet sich für einzelne Untersuchungen bei besonderem Interesse.
  - `Dynamic-Relate-Object`: Dem selektierten Binär-Objekt wird das am weitesten zurückliegende Vorgängerobjekt zugeordnet, welches ebenfalls ein Binär-Objekt ist; dieser Zuordnungsmodus ist für jedes erzeugte Objekt die Grundeinstellung und stellt gewissermaßen die *natürlichste* Zuordnung dar. Bei Objekten, die aus einer `Sel`-Methode hervorgingen, führt dies zum Vergleich mit dem unmittelbaren Vorgängerobjekt, nämlich dem durch die `Int`-Methode erzeugten Binär-Objekt.

Dieser Modus kann durch `Set Edit Deviation GLOBAL` umgeschaltet werden.

Mit `Set Info` kann der für das gesamte `Set` geltende `Relating`-Modus, also `GLOBAL` oder `LOCAL`, abgerufen werden. Ferner ist dort auch das für `GLOBAL-Relating` erforderliche `Relate-Object` ersichtlich; es kann durch `Set Edit Deviation Relate-Object` deklariert werden.

D.3.4 Auswahl des Abweichungs-Kriteriums

Beim Verbergen von Objekten anhand einer Abweichungs-Schranke mit `Set View Hide Deviation` ist das aktuell eingestellte Abweichungs-Kriterium relevant; es wird dort im folgenden *Dialog* aktuell eingeblendet.

Das aktuelle Abweichungs-Kriterium kann durch `Set Edit Deviation Criterion` eingestellt werden; Vorgabe ist `MaxAbs`. Die präsentierten Abweichungs-Kriterien werden in Kapitel 6 (vgl. Abschnitt 6.2.5.3, S. 113) erläutert.

Anhang E

HOPE — Demonstration

Im folgenden wird ein Protokoll der HOPE-Sitzung zur Bearbeitung des `sine`-Beispiels (vgl. Abschnitt 6.3.2, S. 127ff) wiedergegeben.

1. `hope`
   Aufruf zum Starten von HOPE.

2. `Set New`
   Dialog zur Eröffnung eines neues `Sets`:

   - Eingabe des Namens: `sine`
   - Eingabe der Anzahl der PLA-Eingänge: `7`
   - Eingabe der Anzahl der PLA-Ausgänge: `10`

3. `Toolbox Numeric-Layer Function Sine`
   Dialog zur Eingabe der Anzahl der Einträge: Return für `128`

4. `Toolbox Numeric-Layer Scale Default=001 Multiply`
   Dialog zur Eingabe des Skalierungsfaktors: Return für `1023`

5. `Toolbox Conversion-Layer Rounding Default=002`
   Generierung von drei Binär-Objekten mit den drei Rnd-Modi.

6. `Toolbox Conversion-Layer Power-of-2 Default=002`
   Generierung von drei Binär-Objekten mit den drei P2-Modi.

7. `Toolbox Binary-Layer Pseudo-Inc/Dec Enter-Id`
   Dialog zur Eingabe der `Id` des Binär-Objektes mit der Rnd_Round-Methode: `3`
   Es folgt die Generierung der Binär-Objekte mit den Psd-Modi.

8. `Toolbox Binary-Layer Similar-Entries Enter-Id`
   Dialog zur Eingabe der `Id` des Binär-Objektes mit der Rnd_Round-Methode: `3`
   Es folgt die Generierung der Binär-Objekte mit den Sim-Modi.

9. `Set Save`
   Dialog zum Abspeichern des erstellten `Sets`: Bestätigung des vorgebenen Namens mit Return

10. `Toolbox Conversion-Layer Intervals Default=002 Up-Down`
    Generierung eines Binär-Objektes mit Intervall-Attribut mit der Int_UpDn-Methode.

11. `Set Edit Deviation Relate-Object Default=045`
    Bestätigung der Deklaration des Vergleich-Objektes bzw. `Relate-Object` im gleichzeitig einzustellenden Modus `GLOBAL-Relating` mit Eingabe von `y` $\boxed{\text{Return}}$

12. `Toolbox Conversion-Layer Selection Default=045`
    Generierung von fünf Binär-Objekten mit den fünf Sel-Modi.

13. `Set Deviation`
    Bestätigung zum Starten der Abweichungsbestimmung für alle nicht verborgenen Objekte; die Abweichungen werden immer vollständig für alle fünf Abweichungs-Kriterien ermittelt.

14. `Set Edit Deviation Criterion MaxAbs`
    Einstellen des Abweichungs-Kriteriums entsprechend dem Intervall-Modus, d. h. für eine Absolut-Intervall, wie Int_Abs und Int_UpDn, wird `MaxAbs` (M_{abs}) und für ein Relativ-Intervall wird `maxRel` (M_{rel}) eingestellt.

15. `Set View Hide Deviation`
    Dialog zum Verbergen anhand einer `MaxAbs`-Abweichungs-Schranke. Eingabe von `0`, um all jene Objekte zu verbergen, die Einträge außerhalb der Intervalle aufweisen, d. h. es ist bezüglich der Intervalle der Int-Methode nur eine Abweichung von $M_{abs} := 0$ zulässig.

16. `Set Costs`
    Bestätigung zum Starten des Logik-Minimierers ESPRESSO für alle nicht verborgenen Objekte. Die Logik-Minimierung findet in der eingestellten Qualitätsstufe bzw. `Quality` statt, die durch `Set Edit Quality` eingestellt werden kann; vorgegeben ist `Quality=Normal`.

17. `Set View Hide Reduction`
    Dialog zum Verbergen anhand einer Reduzierungs-Schranke. Eingabe von `1`, um all jene Objekte zu verbergen, die keine Kosten-Reduzierung erzielen.

18. `Set Edit Deviation Relate-Object Enter-Id`
    Dialog zur Eingabe der `Id` der Referenz bzw. des `Reference-Object`s mit der Rnd_Round-Methode: `3`. Damit können die anhand der Intervalle ausgelesenen Objekte (die aktuell nicht verborgenen Objekte) in ihren Abweichungen auf dasjenige Objekt bezogen werden, das in diesem Kontext die *ad hoc* spezifizierte PLA-Tabelle repräsentiert.

19. `Set Deviation`
    Bestätigung zum Starten der Abweichungsbestimmung für alle nicht verborgenen Objekte, bei denen durch den vorhergehenden Schritt die bisher ermittelten Abweichung ungültig geworden waren.

20. `Set View List`
    Ausgabe aller nicht verborgenen Objekte im `Set` als Listing.

21. `Set Save`
    Dialog zum Abspeichern des erstellten `Sets`: Ersetzen des vorgebenen Namens durch beispielsweise `sineupdn`.

Mit dieser Bediensequenz sind die *Lösungskandidaten* anhand von Intervallen des Int_UpDn-Modus' ermittelt worden. Für die alternativen Ermittlungen anhand der Int_Abs- und Int_Rel-Modi kann zunächst die Situation vor Schritt 10 durch das Laden der Datei `sine.set` wiederhergestellt werden. Anschließend sind folgende Änderungen notwendig:

- Int_Abs: In Schritt 10 wird statt `Up-Down` nunmehr `Absolut` ausgewählt, wobei noch die Eingaben der beiden Intervall-Parameter mit jeweils `2` hinzugefügt werden muß.
- Int_Rel: In Schritt 10 wird statt `Up-Down` nunmehr `Relativ` ausgewählt, wobei noch die Eingaben der beiden Intervall-Parameter mit jeweils `5` (%) hinzugefügt werden muß. Zusätzlich muß in Schritt 14 statt `MaxAbs` für die Relativ-Intervalle nunmehr `maxRel` eingestellt werden.

Die `Set`s sollten unter den Datei-Namen `sineabs2` bzw. `sinerel5` abgespeichert werden.

Stichwortregister