TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik
Institut für Theoretische Informatik
Lehrstuhl für Automatentheorie

Bachelorarbeit

Hybrides Membrane-Quantum Computing

eingereicht von: Timo Schick
<timo.schick@mailbox.tu-dresden.de>

Bearbeitungszeitraum: 01. Mai – 27. Juli 2015

1. Gutachter: Prof. Dr.-Ing. Franz Baader

2. Gutachter / Betreuer: Dr.-Ing. Monika Sturm

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbstständig, ohne fremde Hilfe und ohne Benutzung anderer als der von mir angegebenen Quellen angefertigt zu haben. Alle aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche gekennzeichnet. Die Arbeit wurde noch keiner Prüfungsbehörde in gleicher oder ähnlicher Form vorgelegt.

Dresden, den 07.07.2015

_____

Timo Schick

Inhaltsverzeichnis

1 Einführung

> Hybrid subjects are often astonishingly fertile, whereas if a scientific discipline remains too pure it usually wilts.
>
> Francis Crick

Die vorliegende Arbeit hat zum Ziel, von einem theoretischen Standpunkt aus aufzuzeigen, wie zwei unkonventionelle Berechnungsmodelle sinnvoll miteinander kombiniert werden können: zum einen ein *membranbasiertes Rechensystem*, für das biologische Membranen und Moleküle als Berechnungsgrundlage dienen, zum anderen ein *Quantenrechner*, dessen Funktionsweise auf den Prinzipien und Gesetzen der Quantenphysik beruht.

Diese zwei Berechnungsmodelle haben auf den ersten Blick kaum etwas gemeinsam – abgesehen vielleicht von der Tatsache, dass sich beide zum aktuellen Zeitpunkt nicht in praxistauglichem Umfang realisieren lassen. Es ist daher naheliegend zu fragen, welchen Sinn die Beschäftigung mit einer Kombination aus zwei so grundverschiedenen – und obendrein bislang überwiegend theoretischen – Ansätzen überhaupt ergibt. Zumal bekannte Quantenalgorithmen zum Lösen praxisrelevanter Probleme zwar eine deutliche Beschleunigung gegenüber jeglichen Algorithmen auf klassischen Rechnern erreichen, aber immer noch weitaus weniger zeiteffizient als entsprechende molekulare Verfahren sind.

Warum also lohnt die theoretische Konstruktion eines hybriden Berechnungsmodells aus Membran- und Quantenrechner?

Wir halten es mit Francis Crick und antworten auf diese Frage, dass möglicherweise gerade die Kombination zweier bislang nicht realisierbarer Systeme ungeahnte Möglichkeiten bietet, nicht zuletzt hinsichtlich deren Realisierung. So scheitert etwa die Umsetzung vieler zeiteffizienter Verfahren im Bereich des molekularen Rechnens momentan an deren immensem Speicherbedarf; die meisten relevanten und bekannten Quantenalgorithmen sind hingegen sehr platzeffizient, benötigen aber deutlich mehr Rechenzeit als vergleichbare membranbasierte Ansätze. Lässt sich eine zu lösende Aufgabe in geeigneter Weise in zwei Teilaufgaben zerlegen, so kann ein hybrides System diese inhärenten Schwachstellen *beider* Berechnungsmodelle und die damit verbundenen Probleme bei deren Realisierung bis zu einem gewissen Grad kompensieren.

Wir verdeutlichen diesen Gedanken beispielhaft anhand des praxisrelevanten Problems *Circuit Minimization*, das fragt, ob zu einem gegebenen Schaltkreis ein äquivalenter Schaltkreis mit einer vorgegebenen Anzahl an Gattern existiert. Die Lösung dieses Problems lässt sich leicht in zwei Aufgaben teilen: erstens die Erzeugung sämtlicher als Lösung denkbarer Schaltkreise, zweitens die anschließende Überprüfung jedes dieser Lösungskandidaten. Unter exponentiellem Platzaufwand kann ein membranbasiertes Rechensystem die erste Teilaufgabe übernehmen und alle Lösungskandidaten in linearer Zeit erzeugen. Um

den Verbrauch an Speicherplatz zu begrenzen, kann jeder dieser Kandidaten anschließend von einem Quantenrechner in vertretbarer Rechenzeit und mit nur polynomiellem Platzbedarf überprüft werden. Den genauen Ablauf einer solchen hybriden Berechnung werden wir im letzten Abschnitt dieser Arbeit anhand eines mit Circuit Minimization vergleichbaren Problems aufzeigen.

Wie in [SU02] nachzulesen, existieren zahlreiche Probleme, die eine ähnliche Struktur wie Circuit Minimization aufweisen. Dies zeigt die Relevanz eines Berechnungsmodells, mit dem solche Probleme effektiv gelöst werden können.

Es gibt also durchaus Anlass, über die Konstruktion eines hybriden Systems nachzudenken. Aber wie ist eine Kombination zweier so verschiedener Modelle sinnvoll realisierbar?

In dieser Arbeit werden wir einen erstmals in [RAB$^+$14a] vorgestellten Ansatz verfolgen. Das Grundgerüst des hybriden Systems bildet hier ein Membranrechner, in den Quantensysteme „eingebettet“ werden können. Die Vorstellung, eine solche Funktionalität in ein membranbasiertes Rechensystem zu integrieren, ist zwar ungewöhnlich, aber keineswegs undenkbar; allerdings soll die praktische Umsetzung eines solchen hybriden Systems in dieser Arbeit nicht weiter thematisiert werden. Das mit unserem Ansatz erhaltene Modell kann auf zwei verschiedenen Ebenen Berechnungen durchführen: einer *Membran-Ebene* und einer *Quanten-Ebene*. Durch diese klare Trennung bleiben im hybriden System wesentliche Eigenschaften beider Berechnungsmodelle erhalten. Es muss allerdings ein Formalismus bereitgestellt werden, mittels dessen beide Ebenen auf geeignete Weise miteinander kommunizieren können.

An dieser Stelle sei erwähnt, dass es auch andere Ansätze zur Kombination von Membran- und Quantenrechnern gibt. Ein solcher Ansatz findet sich etwa in [Lep07]; dort werden sogenannte QUREM (Quantum Unit Rules and Energy assigned to Membranes) P-Systeme eingeführt. Bei diesen Systemen handelt es sich im Wesentlichen um Membranrechner, deren Berechnungsschritte Quantentransformationen entsprechen.

Wir werfen abschließend einen kurzen Blick auf die Zielsetzungen und Schwerpunkte der einzelnen Kapitel dieser Arbeit.

Schlüssel zum Verständnis der Arbeitsweise eines hybriden Systems, wie wir es hier vorstellen, ist die Kenntnis der beiden Teilsysteme, aus denen es aufgebaut ist. Daher werden wir, nachdem in Kapitel 2 einige mathematische Grundlagen geklärt und Festlegungen zur Notation getroffen wurden, in den Kapiteln 3 und 4 einen Überblick über die Berechnungsmodelle des *Membrane Computing* und des *Quantum Computing* geben – allerdings stets im Hinblick auf das eigentliche Ziel, die Kombination beider Modelle. In Kapitel 5 fassen wir die Ergebnisse der vorhergehenden Abschnitte zusammen und konstruieren das hybride System aus Membran- und Quantenrechner. Berechnungen, die dieses System durchführt, bezeichnen wir mit dem für diese Arbeit titelgebenden Begriff als *(hybrides) Membrane-Quantum Computing*.

Schließlich widmen wir uns in Kapitel 6 der Anwendung hybrider Systeme. Wie in [RAB$^+$14b] angeregt, zeigen wir dort, wie mittels Membrane-Quantum

Computing zwei der schwersten Probleme aus den Komplexitätsklassen Σ_2^{P} und Π_2^{P} sinnvoll gelöst werden können.

Während die Definitionen der in den ersten Kapiteln aufgezeigten Berechnungsmodelle überwiegend aus der jeweils angegebenen Literatur entnommen sind, stellen die in Kapitel 5 vorgenommenen Modifikationen und Ergänzungen in der Definition hybrider Systeme sowie insbesondere das Aufzeigen einer möglichen Anwendung in Kapitel 6 die Ergebnisse eigener Forschung im Rahmen dieser Bachelorarbeit dar.

2 Mathematische Grundlagen und Notationen

Mengen

Wir verwenden in dieser Arbeit die üblichen Definitionen und Notationen der Mengenlehre. Dass ein Objekt a Element einer Menge A ist, notieren wir mit $a \in A$, ist A Teilmenge von B, so schreiben wir $A \subseteq B$ und $A \subset B$, falls diese Teilmengenbeziehung echt ist. Die Potenzmenge einer Menge A notieren wir mit $\mathcal{P}(A)$. Mit $A \times B$ bezeichnen wir das kartesische Produkt zweier Mengen A und B. Die leere Menge schreiben wir als $\emptyset$, die Menge $\{0, 1, 2, \ldots\}$ der natürlichen Zahlen als $\mathbb{N}$. Mit $\mathbb{R}$ bezeichnen wir die Menge der reellen Zahlen, die Menge der komplexen Zahlen mit $\mathbb{C}$.

Formale Sprachen

Ein *Alphabet* Σ ist eine endliche Menge voneinander unterscheidbarer Zeichen. Ein *Wort* über Σ ist eine endliche Konkatenation von Zeichen aus Σ. Die n-fache Konkatenation desselben Zeichens $a \in \Sigma$ schreiben wir als a^n. Die Länge eines Wortes w notieren wir als $|w|$, das leere Wort mit $|w| = 0$ als ε. Die Anzahl der Vorkommen eines Zeichens $a \in \Sigma$ in einem Wort w schreiben wir als $|w|_a$. Mit Σ^* bezeichnen wir die Menge aller Wörter über Σ. Jede Teilmenge von Σ^* nennen wir *(formale) Sprache* über Σ. Das *Komplement* einer formalen Sprache $L \subseteq \Sigma^*$ ist die Menge $\overline{L} := \Sigma^* \setminus L$.

Multimengen

Eine *Multimenge* M über einer Menge A ist eine Abbildung $M : A \to \mathbb{N}$. Dabei beschreibt die Zahl $M(x)$ für ein $x \in A$ die Anzahl der Vorkommen von x in M. Für zwei Multimengen M_1, M_2 sagen wir, M_1 ist Teilmenge von M_2 und wir schreiben $M_1 \subseteq M_2$, wenn für alle $a \in A$ gilt: $M_1(a) \leq M_2(a)$. Die Differenz zweier Multimengen M_1 und M_2, für die $M_2 \subseteq M_1$ gilt, ist die Abbildung $M_1 - M_2 : A \to \mathbb{N}$ mit $(M_1 - M_2)(a) = M_1(a) - M_2(a)$.

Jede Multimenge lässt sich, sofern die Menge A endlich und von der Form $A = \{a_1, ..., a_n\}$, $n \in \mathbb{N}$ ist, in der Form $\{(a_1, M(a_1)), \ldots, (a_n, M(a_n))\}$ veranschaulichen. Noch kompakter ist die Darstellung einer solchen Multimenge als Wort $w = a_1^{M(a_1)} a_2^{M(a_2)} \ldots a_n^{M(a_n)}$. Im Folgenden werden wir Multimengen über endlichen Grundmengen A ausschließlich in letzterer Form angeben. Wenn wir für ein solches Wort w über A von der „Multimenge w“ sprechen, meinen wir damit stets „die Multimenge, die durch w beschrieben wird“.

Relationen

Eine *binäre Relation* R auf einer Menge A ist eine Teilmenge von $A \times A$. Wir nennen eine solche Relation

(1) *reflexiv*, wenn $(a, a) \in R$ für alle $a \in A$,

(2) *symmetrisch*, wenn $(a, b) \in R \Rightarrow (b, a) \in R$ für alle $a, b \in A$,

(3) *transitiv*, wenn $(a,b) \in R \wedge (b,c) \in R \Rightarrow (a,c) \in R$ für alle $a,b,c \in A$.

Eine reflexive, symmetrische und transitive Relation bezeichnen wir als *Äquivalenzrelation*. Mit $[a]_R$ bezeichnen wir die *Äquivalenzklasse* von $a \in A$ bezüglich einer Äquivalenzrelation $R \subseteq A \times A$, und es gilt: $[a]_R = \{x \in A \mid (x,a) \in R\}$. Die *reflexiv-transitive Hülle* R^* einer binären Relation R ist die kleinste reflexive und transitive Relation R^* auf A mit $R \subseteq R^*$.

Binärzahlen

Eine Binärzahl z lässt sich als ein Wort über $\{0,1\}$ in der Form $z = z_{n-1} \dots z_0$ mit $n \in \mathbb{N}$, $z_i \in \{0,1\}$, $0 \leq i \leq n-1$ darstellen. Wir werden ein solches Wort der Länge n häufig als ein n-Tupel über $\{0,1\}$ auffassen, die Darstellungen als $z_{n-1} \dots z_0$ und in der Form $(z_{n-1}, \dots, z_0) \in \{0,1\}^n$ betrachten wir als äquivalent. Das n-fache kartesische Produkt $\{0,1\}^n$ ist für uns dementsprechend gleichbedeutend mit der Menge aller Wörter über $\{0,1\}$ mit Länge n.

Das *exklusive Oder* zweier Zahlen $x, y \in \{0,1\}$ ist die Operation $\oplus$, die definiert ist als

$$x \oplus y := \begin{cases} 1 & \text{wenn } x \neq y, \\ 0 & \text{andernfalls.} \end{cases}$$

Diese Definition lässt sich auf zwei Binärzahlen $x = x_n \dots x_0$ und $y = y_n \dots y_0$ beliebiger, aber gleicher Länge erweitern als $x \oplus y = z_n \dots z_0$ mit $z_i = x_i \oplus y_i$, $0 \leq i \leq n$. Für solche x und y bezeichnen wir $x \circ y := \bigoplus_{i=0}^{n} x_i y_i$ als deren *Skalarprodukt*.

Komplexe Zahlen

Für eine komplexe Zahl $z \in \mathbb{C}$ der Form $z = a + bi$ mit der imaginären Einheit i und $a, b \in \mathbb{R}$ nennen wir $|z| := \sqrt{a^2 + b^2}$ den *Betrag* von z. Die zu z *komplex konjugierte* Zahl ist $\bar{z} := a - bi$.

Tensorprodukt

Das *Tensorprodukt* zweier Vektorräume U und V über demselben Körper K ist der Vektorraum $U \otimes V$ über K. Ist $E = \{e_0, \dots, e_{n-1}\}$ eine Basis von U und $F = \{f_0, \dots, f_{m-1}\}$ eine Basis von V mit $n, m \in \mathbb{N}$, dann ist $E \times F$ eine Basis des Vektorraums $U \otimes V$, der folglich die Dimension $m \cdot n$ hat. Die Elemente dieser Basis $(e_i, f_i) \in E \times F$ schreiben wir als $e_i \otimes f_i$.

Für zwei beliebige Vektoren u und v der Form

$$u = \sum_{i=0}^{n-1} \alpha_i e_i \in U \qquad\qquad v = \sum_{i=0}^{m-1} \beta_i f_i \in V$$

mit $\alpha_i \in K$, $0 \leq i \leq n-1$ und $\beta_i \in K$, $0 \leq i \leq m-1$ ist deren Tensorprodukt definiert als

$$u \otimes v := \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} \alpha_i \beta_j (e_i \otimes f_j) \in U \otimes V.$$

Es gilt $\alpha(u \otimes v) = \alpha u \otimes v = u \otimes \alpha v$ für alle $u \in U$, $v \in V$ und $\alpha \in K$.

Unitäre Matrizen

Zur Matrix $A = (a_{ij}) \in \mathbb{C}^{n \times n}$ bezeichnen wir $A^T = (a_{ji})$ als ihre *transponierte*, $A^\dagger = (\overline{a_{ij}})^T = (\overline{a_{ji}})$ als ihre *adjungierte* Matrix. Wir nennen eine solche Matrix A *unitär*, falls $A^\dagger$ zu A invers ist, also falls gilt: $A^{-1} = A^\dagger$. Eine für unsere Zwecke wesentliche Eigenschaft unitärer Matrizen ist ihre Invertierbarkeit.

Komplexitätstheorie

Die Komplexitätsklasse P ist definiert als die Menge aller formalen Sprachen, die von einer polynomiell zeitbeschränkten, *deterministischen* Turingmaschine entschieden werden können. Analog bezeichnen wir mit NP die Klasse der Sprachen, die eine polynomiell zeitbeschränkte, *nichtdeterministische* Turingmaschine entscheiden kann. Zu jeder Komplexitätsklasse C definieren wir die *komplementäre* Klasse $coC := \{L \mid \overline{L} \in C\}$. Ist die formale Sprache $L \subseteq \Sigma^*$ polynomialzeitreduzierbar auf $L' \subseteq \Sigma^*$, so schreiben wir $L \leq_p L'$. Liegt in einer Komplexitätsklasse C eine Sprache L_0, sodass $L \leq_p L_0$ für alle $L \in C$ gilt, dann nennen wir L_0 *C-vollständig*.

Zu einer Turingmaschine M und einer formalen Sprache $A \subseteq \Sigma^*$ erhalten wir die *Orakel-Turingmaschine* M^A, indem wir M um ein zusätzliches Eingabeband ergänzen, mittels welchem M für ein Wort $w \in \Sigma^*$ innerhalb eines Schrittes entscheiden kann, ob $w \in A$ gilt. Mit NP^C bezeichnen wir für eine Komplexitätsklasse C die Menge aller Sprachen L, für die eine Sprache $A \in C$ und eine polynomiell zeitbeschränkte, nichtdeterministische Orakel-Turingmaschine M^A existiert, sodass $L(M^A) = L$ gilt. Die Komplexitätsklassen Σ_i^P und Π_i^P sind induktiv definiert als

$$\begin{aligned}
\Sigma_0^\mathrm{P} &:= \Pi_0^\mathrm{P} := \mathrm{P} \\
\Sigma_i^\mathrm{P} &:= \mathrm{NP}^{\Sigma_{i-1}^\mathrm{P}},\ i \geq 1 \\
\Pi_i^\mathrm{P} &:= co\Sigma_i^\mathrm{P},\ i \geq 1.
\end{aligned}$$

Aussagenlogik

Für aussagenlogische Formeln verwenden wir in dieser Arbeit die Buchstaben ψ, ϕ und π. Die Wahrheitswerte *wahr* und *falsch* repräsentieren wir durch die Konstanten 1 und 0. Mit $Var(\psi)$ bezeichnen wir die Menge aller Variablen, die in einer Formel ψ vorkommen.

Als *Literal* bezeichnen wir eine Variable x oder ihre Negation $\neg x$, eine endliche Disjunktion $L_1 \vee L_2 \vee \ldots \vee L_n$ von Literalen $L_1, \ldots, L_n$ nennen wir *Klausel*.

Eine Formel ψ ist in *konjunktiver Normalform* (KNF), wenn sie von der Form $C_1 \wedge C_2 \wedge \ldots \wedge C_m$ mit Klauseln $C_1, \ldots, C_m$ ist, und in 3-KNF, wenn jede dieser Klauseln aus genau 3 Literalen besteht.

Wir nennen eine Formel, die mindestens einen der Quantoren $\exists$ und $\forall$ enthält, *quantifiziert*. Ist eine quantifizierte Formel ψ von der Form

$$\psi = Q_1 x_1 \ldots . Q_n x_n . \phi,\ Q_i \in \{\exists, \forall\},\ 1 \leq i \leq n$$

und ist ϕ eine quantorenfreie Formel mit $Var(\phi) \subseteq \{x_1, \ldots, x_n\}$, so bezeichnen wir ψ als *geschlossen*.

Für $\alpha \in \{0, 1\}$ bezeichnen wir mit $\psi_{x=\alpha}$ die Formel, die aus ψ entsteht, indem jedes Vorkommen von x durch α ersetzt wird. Gibt es für eine quantorenfreie Formel ψ mit $Var(\psi) = \{x_1, \ldots, x_n\}$ mindestens eine Variablenbelegung $x_1 = \alpha_1,\ \ldots,\ x_n = \alpha_n$, $\alpha_i \in \{0, 1\}$, $1 \leq i \leq n$, sodass die Formel $\psi_{x_1=\alpha_1, \ldots, x_n=\alpha_n}$ zu 1 evaluiert wird, so sagen wir, diese Variablenbelegung *erfüllt* ψ und wir nennen ψ *erfüllbar*. Wird eine Formel durch jede Variablenbelegung erfüllt, nennen wir sie *allgemeingültig*. Alle quantorenfreien, erfüllbaren Formeln fassen wir zu der Menge SAT zusammen. Sind diese Formeln zusätzlich in 3-KNF, so liegen sie außerdem in der Menge 3-SAT. Ob eine Formel in SAT bzw. 3-SAT liegt, ist jeweils ein NP-vollständiges Entscheidungsproblem.

3 Membrane Computing

In diesem Kapitel werden wir uns mit einem Berechnungsmodell befassen, das von der Struktur und Funktionsweise lebender Zellen inspiriert ist: mit sogenannten P-Systemen. Strukturgebend für lebende Zellen sind biologische Membranen, die sie als Trennschichten in mehrere Regionen unterteilen. In jeder dieser Regionen können sich verschiedene Stoffe befinden, bei denen es sich um einfache Ionen oder komplexere Moleküle wie lange DNA-Stränge handeln kann. Neben ihrer Funktion als Trennschicht sind Membranen auch dafür zuständig, zu steuern, wie diese Stoffe zwischen verschiedenen Regionen transportiert werden können. Entsprechend spielen Membranen in einem Berechnungsmodell, das auf der Funktionsweise lebender Zellen aufbaut, eine zentrale Rolle – man spricht daher auch vom *membranbasierten Rechnen* (Membrane Computing).

Im ersten Abschnitt dieses Kapitels werden wir das unserem Berechnungsmodell zugrunde liegende Gerüst, die Membranstruktur, formal definieren, bevor wir anschließend das vollständige Berechnungsmodell betrachten. Dabei gibt es eine große Zahl verschiedener Ausprägungen von P-Systemen, die nur schwer auf einen gemeinsamen Nenner zu bringen sind. Wir werden daher zunächst ein allgemeines P-System betrachten und uns anschließend auf einen Spezialfall, sogenannte P-Systeme mit aktiven Membranen, beschränken, die wir für die Konstruktion unseres hybriden Systems benötigen. Einen umfassenden Überblick über weitere P-Systeme bieten [Pău02] und [CP01], denen auch die Definitionen der hier vorgestellten Varianten und der Membranstruktur entnommen sind.

3.1 Die Membranstruktur

Eine *Membranstruktur* bildet eine hierarchische Anordnung von Membranen, die alle von einer einzigen, äußeren Membran umgeben sind – der *Haut*. Den Bereich außerhalb dieser Haut bezeichnen wir als *Umgebung* der Membranstruktur. Eine solche Anordnung kann beispielsweise als Venn-Diagramm veranschaulicht werden, wobei jede Membran durch eine Menge dargestellt wird. Dabei gibt es keine Überschneidungen und alle Mengen sind in einer Obermenge, der Haut, enthalten (siehe Abbildung 1).

Um die von den Membranen begrenzten Regionen eindeutig referenzieren zu können, beschriften wir sie mit sogenannten *Labels* $l \in \mathbb{N}$, wobei mit einem solchen Label im Folgenden sowohl die Membran selbst, als auch ihre *innere Region* bezeichnet wird. Innere Region meint dabei den gesamten Bereich, der in der Darstellung als Venn-Diagramm unmittelbar von dieser Membran eingeschlossen wird – also ohne, dass sich eine weitere Membran zwischen der betrachteten Membran und dem zugehörigen Bereich befindet. Analog bezeichnen wir für $i, j \in \mathbb{N}$ die Region i, die eine Membran j unmittelbar umgibt, als deren *äußere Region* und sagen, dass Region i *adjazent* zu Membran j ist. Membranen, die selbst keine weiteren Membranen enthalten, bezeichnen wir als *elementar*.

Wir formalisieren diese Darstellung durch Betrachtung der Sprache LMS, der Labelled Membrane Structures, über dem Alphabet $\Sigma = \{[_i,]_i \mid i \in \mathbb{N}\}$, die wir wie folgt rekursiv definieren:

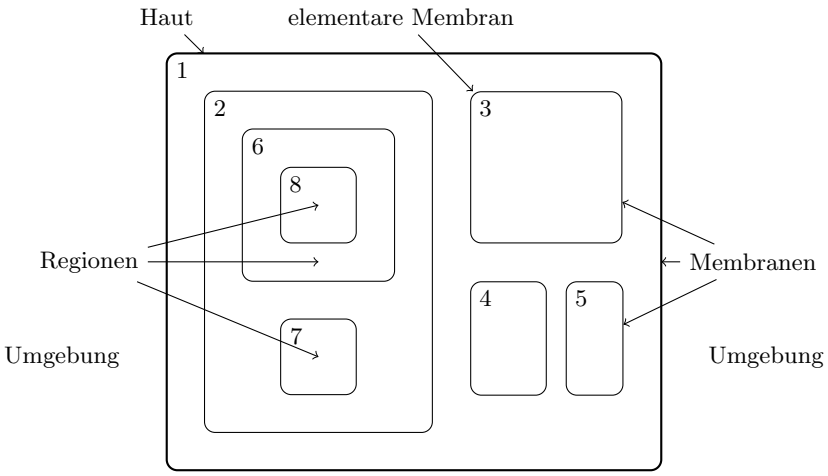

Abbildung 1: Membranstruktur als Venn-Diagramm

(1) $[_i]_i \in \text{LMS}$ für $i \in \mathbb{N}$

(2) $\mu_1, \ldots, \mu_n \in \text{LMS}, n \geq 1 \Rightarrow [_i \mu_1 \ldots \mu_n]_i \in \text{LMS}, i \in \mathbb{N}$

Die Wörter $\mu \in \text{LMS}$ bezeichnen wir als Membranstruktur[1]. Dabei stellt jedes zusammengehörige Paar von Klammern $[_i,]_i$ eine Membran dar. In dieser Darstellung sind in einer Membranstruktur μ genau die Membranen elementar, die in der Form $[_i]_i$ vorkommen. Die in Abbildung 1 gezeigte Membranstruktur entspricht beispielsweise dem Wort $\mu = [_1[_2[_6[_8]_8]_6[_7]_7]_2[_3]_3[_4]_4[_5]_5]_1$.

3.2 P-Systeme

Um ein vollständiges Berechnungsmodell zu erhalten, erweitern wir unsere Membranstruktur um Multimengen von *Objekten*, die sich in den einzelnen Regionen befinden, und um *Entfaltungsregeln*, nach denen sich diese Objekte entwickeln und zwischen den Regionen transportiert werden können. Das so erhaltene P-System werden wir häufig als Venn-Diagramm wie in Abbildung 1 darstellen, wobei wir zu jeder Region die Multimenge der darin befindlichen Objekte und die mit ihr assoziierten Entfaltungsregeln hinzufügen.

Definition 3.1 Ein allgemeines *P-System* vom Grad m, $m \geq 1$, ist ein Tupel $\Pi = (O, \mu, w_1, \ldots, w_m, R_1, \ldots, R_m, i_o)$, wobei gilt:

(1) O ist ein Alphabet, dessen Elemente wir als *Objekte* bezeichnen,

[1] Weil die Reihenfolge der Membranen in derselben Hierarchiestufe irrelevant ist, werden gelegentlich anstelle der Wörter über LMS deren Äquivalenzklassen bezüglich der reflexiv-transitiven Hülle der Relation $\sim$ mit $x \sim y \Leftrightarrow x = \mu_1\mu_2\mu_3\mu_4 \wedge y = \mu_1\mu_3\mu_2\mu_4$ für $\mu_1\mu_4 \in \text{LMS}$, $\mu_2, \mu_3 \in \text{LMS}$ als Membranstruktur bezeichnet.

(2) μ ist eine Membranstruktur, die aus genau m Membranen mit den Labels $1, \ldots, m$ besteht,

(3) w_i, $1 \leq i \leq m$ sind Wörter über O, die die Multimengen an Objekten in den Regionen $1, \ldots, m$ repräsentieren,

(4) R_i, $1 \leq i \leq m$ sind endliche Mengen von Entfaltungsregeln über O, die jeweils in der Region i angewandt werden können und von der Form $u \to v$ sind, wobei u ein Wort über O ist und v ein Wort über $O_{tar} = O \times \text{TAR}$ (dabei bezeichnet $\text{TAR} := \{here, out\} \cup \{in_j \mid 1 \leq j \leq m\}$ die Menge der *Zielinformationen*),

(5) $i_o \in 1, \ldots, m$ ist das Label einer elementaren Membran, der Ergebnismembran.

Jedes Tupel der Form $(w'_1, \ldots, w'_m)$, $w'_i \in O^*$, $1 \leq i \leq m$ nennen wir eine *Konfiguration* von Π. Das Tupel $(w_1, \ldots, w_m)$ bezeichnen wir als seine initiale Konfiguration.

Für zwei Konfigurationen C_1 und C_2 schreiben wir $C_1 \Rightarrow C_2$ und wir sagen, es gibt eine *Transition* von C_1 zu C_2, falls es durch *synchrone*, *nichtdeterministische* und *maximal parallele* Anwendung der Entfaltungsregeln $R_1, \ldots, R_m$ in den jeweils zugehörigen Regionen $1, \ldots, m$ möglich ist, von C_1 zu C_2 zu gelangen. Wie genau diese Anwendung der Entfaltungsregeln abläuft, soll im Folgenden beschrieben werden.

Dazu betrachten wir zunächst die in einer gegebenen Konfiguration zu Membran i zugehörige Multimenge w'_i sowie eine einzelne Entfaltungsregel der Form $u \to v$ aus der Menge R_i. Die Anwendung einer solchen Regel bedeutet, die durch u bestimmte Multimenge von der Multimenge w'_i abzuziehen, und dafür die durch v spezifizierten Objekte in den Regionen hinzuzufügen, die durch die Zielinformationen der Regel bestimmt werden. Für jedes Zeichen $v_k \in O \times \text{TAR}$, $1 \leq k \leq n$ aus $v = v_1 \ldots v_n$ verfahren wir wie folgt:

(1) Wenn $v_k = (o, here)$, $o \in O$, so wird das Objekt o der Multimenge w'_i hinzugefügt, also genau der Region, in der die Regel angewandt wird. Zur vereinfachten Darstellung werden wir auf die Zielinformation *here* zukünftig verzichten, also für $(o, here)$ kurz o schreiben.

(2) Wenn $v_k = (o, out)$, $o \in O$, so wird das Objekt o der zur Membran i adjazenten Region hinzugefügt, also zu der Region, in der sich Membran i unmittelbar befindet. Insbesondere bedeutet dies, dass Objekte auch in die Umgebung abgegeben werden können, falls Membran i die Haut ist.

(3) Wenn $v_k = (o, in_j)$, $o \in O$, $1 \leq j \leq m$, wird das Objekt o der Region mit Label j hinzugefügt, falls Membran j unmittelbar innerhalb der Region i liegt, also falls sie adjazent zur Region i ist. Gibt es keine Membran j, die diese Bedingung erfüllt, so kann die entsprechende Regel nicht angewandt werden.

Eine Regel $u \to v$ aus R_i kann nur dann angewandt werden, wenn die Multimenge u eine Teilmenge der Multimenge w_i' ist, da die Anwendung der Regel die Objekte aus u „verbraucht“.

Wir erweitern unsere Betrachtung nun von einer einzelnen Entfaltungsregel auf eine einzelne Membran i und die zugehörige Menge an Entfaltungsregeln R_i. Die Regeln aus R_i, nach welchen sich die Objekte aus w_i' in einem Zeitschritt entwickeln, werden nichtdeterministisch ausgewählt. Außerdem laufen alle Regeln, die in einem Schritt angewandt werden, synchron ab, sie benötigen also alle dieselbe Zeitspanne und beginnen zum exakt selben Zeitpunkt. Die Anwendung dieser Regeln findet maximal parallel statt – das heißt, es werden in jedem Schritt so lange Regeln auf die noch nicht verbrauchten Objekte aus w_i' angewandt, bis auf die verbleibenden Objekte keine Regel mehr angewandt werden kann. Diese übrigen Objekte verbleiben unverändert in der Membran i, während alle anderen Objekte sich gemäß der auf sie angewandten Regeln entwickeln. Maximale Parallelität bedeutet also nicht, dass die Anzahl zugleich angewandter Regeln oder die Anzahl beteiligter Objekte zwangsläufig maximal sein muss.

Sei beispielsweise $w_i' = aaaabbb$, $R_i = \{r_1, r_2\}$ und $r_1 = ab \to c$, $r_2 = aaaa \to aa$. Es ist im folgenden Schritt sowohl die dreifache Anwendung von r_1, als auch eine einfache Anwendung von r_2 zulässig. Im ersteren Fall bliebe nur ein a unverändert, im letzteren bbb. Nicht zulässig ist hingegen die nur ein- oder zweifache Anwendung von r_1, weil dann die Eigenschaft der maximalen Parallelität nicht erfüllt wäre: auf die unverbrauchten Objekte könnten noch weitere Regeln angewandt werden.

Nochmals erweitern wir unsere Betrachtung und analysieren nun nicht mehr eine einzelne Membran des Systems, sondern das komplette System. Indem wir für jede Membran gleichzeitig, wie eben beschrieben, verfahren, also gleichzeitig in allen Membranen i von Π die Regeln aus R_i synchron, nichtdeterministisch und maximal parallel anwenden, erhalten wir eine Transition von einer Konfiguration C_1 zu einer neuen Konfiguration C_2. Die Parallelität und Synchronität findet also auf gleich zwei Ebenen statt: zum einen werden in jeder Membran parallel und synchron Regeln angewandt, zum anderen geschieht dies parallel und synchron in allen Membranen des Systems.

Eine Folge $C_1 \Rightarrow \ldots \Rightarrow C_n$, $n \in \mathbb{N}$ von mehreren solcher Transitionen, beginnend mit der initialen Konfiguration, bezeichnen wir als *Berechnung*. Eine Berechnung ist *erfolgreich*, falls in der letzten Konfiguration C_n keine weiteren Regeln mehr angewandt werden können. Eine solche Konfiguration bezeichnen wir als *Haltekonfiguration*. Das Ergebnis einer erfolgreichen Berechnung ist die Anzahl an Objekten, die sich in der Haltekonfiguration in der Ergebnismembran befinden.

Beispiel 3.2 Das P-System $\Pi_1 = (\{a\}, [_1[_2]_2]_1, a, \varepsilon, \{r_1, r_2\}, \emptyset, 2)$ mit

$r_1 = a \to a(a, in_2)(a, in_2)$

$r_2 = a \to (a, out)$

berechnet nichtdeterministisch eine beliebige gerade Zahl. In jedem Schritt kann in Membran 1 genau eine der Regeln r_1, r_2 genau einmal angewandt werden. Sobald Regel r_2 angewandt wird, erreicht das System eine Haltekonfiguration und die Berechnung endet. Wird zuvor n mal r_1 angewandt, dann ist $w = a^{2n}$ die Multimenge in Region 2, also ist das Ergebnis der Berechnung die Zahl $2n$. Abbildung 2 zeigt das P-System in seiner initialen Konfiguration.

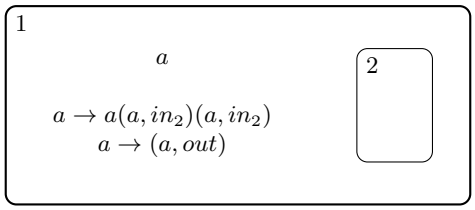

Abbildung 2: Π_1 in der initialen Konfiguration

Es gibt eine Vielzahl von Möglichkeiten, das vorgestellte, allgemeine P-System zu erweitern. Häufige Modifikationen sind beispielsweise der Verzicht auf eine Ausgabemembran, die Hinzunahme von Prioritäten unter den Entfaltungsregeln, die Beschränkung und Modifikation ebendieser Regeln oder die Einführung von Mechanismen, die das Auflösen von Membranen ermöglichen (siehe [Pău02]).

3.3 P-Systeme mit aktiven Membranen

Während in unserem bisherigen P-System die Entfaltungsregeln nur die Objekte, nicht aber die Membranen selbst beeinflusst haben, betrachten wir nun ein Modell, in dem auch die Membranstruktur während der Berechnung verändert werden kann. Wir erweitern unsere Regeln dazu um solche, die das Auflösen von Membranen und das Entstehen neuer Membranen durch Teilung ermöglichen. Allerdings beschränken wir uns in dem hier betrachteten System auf die Teilung elementarer Membranen.

Neben dieser Erweiterung nehmen wir noch zwei weitere Modifikationen vor. Zum einen teilen wir allen Membranen eine *Polarisation* zu, die entweder positiv (+), negativ (−) oder neutral (0) sein kann, zum anderen verzichten wir auf eine Ausgabemembran i_o und betrachten stattdessen die Objekte, die in die Umgebung abgegeben werden, als Ausgabe.

Definition 3.3 Ein *P-System mit aktiven Membranen (ohne nicht-elementare Division)* ist ein Tupel $\Pi = (O, H, \mu, w_1, \ldots, w_m, R)$. Dabei gilt:

(1) $m \geq 1$ ist die Anzahl an Membranen in der initialen Konfiguration,

(2) O ist wie im allgemeinen P-System das Alphabet der Objekte,

(3) $H \subset \mathbb{N}$ ist eine endliche Menge von Labels für die Membranen,

(4) μ ist eine Membranstruktur aus genau m Membranen, wobei das Label jeder dieser Membranen Element von H ist[2] und alle Membranen zu Beginn eine neutrale Polarisation aufweisen,

(5) $w_1, \ldots, w_m$ sind die Multimengen von Objekten, die sich anfangs in den m Regionen aus μ befinden,

(6) R ist eine endliche Menge aus *Entwicklungsregeln*, wobei jede diese Regeln eine der folgenden fünf Formen (a) bis (e) annehmen kann. Wir ergänzen in Klammern eine Erklärung der jeweiligen Regel:

   (a) $[_h a \to v]_h^\alpha$ für $h \in H$, $\alpha \in \{+, -, 0\}$, $a \in O$, $v \in O^*$
   (Diese Regeln können angewandt werden auf ein Objekt a, das sich in der Region einer Membran mit Label h und Polarisation α befindet. Durch Anwendung der Regel wird dieses Vorkommen von a durch v ersetzt.)

   (b) $a[_h]_h^\alpha \to [_h b]_h^{\alpha'}$ für $h \in H$, $\alpha, \alpha' \in \{+, -, 0\}$, $a, b \in O$
   (Diese Regeln können auf Membranen mit Label h und Polarisation α angewandt werden, wenn die adjazente Region das Objekt a enthält. Dieses Objekt a wird durch Anwendung der Regel in die zur Membran h gehörende Region geschickt und dabei zu b verändert. Gleichzeitig ändert sich dadurch die Polarisation der Membran von α zu α'.)

   (c) $[_h a]_h^\alpha \to [_h]_h^{\alpha'} b$ für $h \in H$, $\alpha, \alpha' \in \{+, -, 0\}$, $a, b \in O$
   (Wenn eine Membran mit Label h und Polarisation α das Objekt a enthält, kann a durch eine solche Regel in die diese Membran umgebende Region geschickt werden, wobei sich a zu b und die Polarisation der Membran zu α' ändert.)

   (d) $[_h a]_h^\alpha \to b$ für $h \in H$, $\alpha \in \{+, -, 0\}$, $a, b \in O$
   (Eine Membran mit Label h und Polarisation α, die das Objekt a enthält, wird durch Anwendung dieser Regel aufgelöst, wobei a durch b ersetzt und der komplette Inhalt der Membran in die sie umgebende Region geschickt wird.)

   (e) $[_h a]_h^\alpha \to [_h b]_h^{\alpha'} [_h c]_h^{\alpha''}$ für $h \in H$, $\alpha, \alpha', \alpha'' \in \{+, -, 0\}$, $a, b, c \in O$
   (Eine solche Regel kann angewandt werden auf eine elementare Membran mit Label h und Polarisation α, wodurch diese in zwei Membranen aufgespalten wird, die beide dasselbe Label wie die ursprüngliche Membran, aber gegebenenfalls davon abweichende, unterschiedliche Polarisationen α' und α'' erhalten. Das Objekt a wird in den neuen

[2] Es ist in P-Systemen mit aktiven Membranen durchaus möglich, mehreren Membranen dasselbe Label zuzuteilen, insbesondere muss also nicht $|H| = m$ gelten. Eine solche Zuteilung erschwert allerdings die eindeutige Referenzierung von Regionen, sodass wir uns im Rahmen dieser Arbeit auf solche Systeme beschränken, die in der initialen Konfiguration jeder Region injektiv ein Label aus H zuteilen.

Membranen durch b beziehungsweise c ersetzt, alle anderen Objekte aus der Membran werden in beide neu entstandenen Membranen kopiert.)

Offensichtlich kann in einem solchen System eine Konfiguration nicht wie im allgemeinen Fall angegeben werden; der Zustand des Systems wird nicht mehr nur durch die Multimengen in den Regionen, sondern auch durch die Membranstruktur an sich bestimmt. Außerdem verändert sich während einer Berechnung die Anzahl an Regionen und damit auch an Multimengen, die zu berücksichtigen sind, und nicht zuletzt erschwert die Möglichkeit, mehrere Membranen mit demselben Label zu erzeugen, die eindeutige Referenzierung von Regionen.

Abhilfe schafft die Darstellung einer Konfiguration als Membranstruktur, wobei unmittelbar nach einer öffnenden Klammer $[_i$ alle Objekte angegeben werden, die sich in der zur Membran gehörenden Region befinden. Ist die Polarisation einer Membran $\alpha \neq 0$, so schreiben wir die zugehörige schließende Klammer in der Form $]_i^\alpha$. Eine Konfiguration, in der $\mu = [_1[_3]_3]_1$, $w_1 = ab$, $w_3 = aaa$ gilt und $\alpha_1 = +$ die Polarisation von Membran 1, $\alpha_3 = 0$ die Polarisation von Membran 3 ist, wird also beispielsweise als $[_1 ab[_3 aaa]_3]_1^+$ dargestellt. Die Verwendung von Venn-Diagramm ist ebenfalls zur Darstellung der Konfigurationen eines P-Systems mit aktiven Membranen geeignet – hier ergänzen wir Polarisationen $\neq 0$ gegebenenfalls in der oberen rechten Ecke der zugehörigen Membran.

Wieder bezeichnen wir Übergänge zwischen Konfigurationen als Transitionen. Diese erhalten wir durch Anwendung der Regeln (a) bis (e) in der beschriebenen Weise, wobei diese Anwendung wieder nichtdeterministisch, synchron und maximal parallel erfolgt. Es sind einige Festlegungen zu treffen, um trotz Auflösung und Teilung von Membranen bei gleichzeitiger maximaler Parallelität Widersprüche zu vermeiden:

(1) In einem Schritt kann auf jede Membran h höchstens eine Regel vom Typ (b) - (e) angewandt werden. Diese Einschränkung ist notwendig, um etwa die gleichzeitige Teilung und Auflösung derselben Membran oder die gleichzeitige Überschreibung ihrer Polarisation durch mehrere Regeln zu vermeiden.

(2) Wenn eine Membran gleichzeitig durch eine Regel der Form (e) geteilt und ihre Objekte durch Regeln der Form (a) verändert werden, gehen wir davon aus, dass innerhalb des einen Zeitschrittes, den die Anwendung der Regeln benötigt, zuerst die Regeln vom Typ (a) durchgeführt werden und anschließend die Teilung.

(3) Auf die Haut dürfen keine Regeln vom Typ (b), (d) oder (e) angewandt werden. Es können also keine Objekte aus der Umgebung in die Haut geschickt werden, außerdem darf sie weder geteilt, noch aufgelöst werden.

(Erfolgreiche) Berechnungen und Haltekonfigurationen lassen sich für P-Systeme mit aktiven Membranen in gleicher Weise wie für allgemeine P-Systeme

definieren. Als Ergebnis einer erfolgreichen Berechnung betrachten wir allerdings nicht mehr die Anzahl an Objekten, die sich in einer bestimmten Membran befinden, sondern alle Objekte, die in die Umgebung abgegeben wurden. Wir fügen alle diese Objekte in der Reihenfolge, in der sie in die Umgebung abgegeben wurden, zu einem Wort zusammen; verlassen mehrere Objekte zum selben Zeitpunkt das System, so wählen wir für diese eine beliebige Reihenfolge. Auf diese Weise erzeugt das System eine Menge von Wörtern über O, die wir zur Sprache $L(\Pi)$ zusammenfassen und wir sagen, $L(\Pi)$ wird von Π *generiert*.

Um P-Systeme nicht nur zur Generierung von Sprachen, sondern auch als Entscheidungsverfahren nutzen zu können, führen wir nun den Begriff der *Eingabe* ein.

Definition 3.4 Ein *P-System mit aktiven Membranen und Eingabe* ist ein Tupel $\Pi_{in} = (O, \Sigma, H, \mu, w_1, \ldots, w_m, R, i_{in})$, wobei gilt:

(1) Die Komponenten O, H, μ, w_1, $\ldots$, w_m und R sind festgelegt wie in Definition 3.3,

(2) $\Sigma \subseteq O$ ist ein Alphabet, das wir als *Eingabealphabet* bezeichnen,

(3) $i_{in} \in H$ ist das Label der *Eingabemembran*, von der wir fordern, dass sie in der initialen Membranstruktur μ enthalten ist.

Die Berechnung eines solchen P-Systems wird erst durch Eingabe einer Multimenge $w \in \Sigma^*$ initiiert. Dabei wird w noch vor der ersten Regelanwendung der Multimenge $w_{i_{in}}$ hinzugefügt.

Mit einer geeigneten Codierung ist es möglich, P-Systeme mit aktiven Membranen und Eingabe zu konstruieren, die NP-vollständige Probleme wie 3-SAT in linearer Zeit entscheiden. Intuitiv liegt das daran, dass die Teilung von Membranen es erlaubt, in n Zeitschritten 2^n Membranen zu erzeugen – durch geschickte Auswahl der Entwicklungsregeln ist es so möglich, für eine Formel ψ mit n Variablen in n Zeitschritten alle 2^n möglichen Variablenbelegungen zu erzeugen und für all diese Variablenbelegungen anschließend parallel zu testen, ob sie ψ erfüllen. Um aus dieser Berechnung eine Ausgabe zu erhalten, führen wir ein neues Objekt `Ja` ein, das als einziges Objekt das System verlassen kann und genau dann in die Umgebung abgegeben wird, wenn die Formel erfüllbar ist. Somit gilt:

$$L(\Pi) = \begin{cases} \{\texttt{Ja}\} & \text{wenn } \psi \text{ erfüllbar ist,} \\ \emptyset & \text{andernfalls.} \end{cases}$$

Um zu entscheiden, ob eine Formel erfüllbar ist, genügt es also, zu untersuchen, ob die Sprache $L(\Pi)$ bei Eingabe ψ leer ist, also ob während der Berechnung ein Objekt das System verlässt. Diese Vorgehensweise ermöglicht es, das P-System als Entscheidungsverfahren einzusetzen. Ist die Anzahl an benötigten Rechenschritten nicht bekannt, kann analog zu `Ja` die zusätzliche Einführung eines Objekts `Nein` sinnvoll sein.

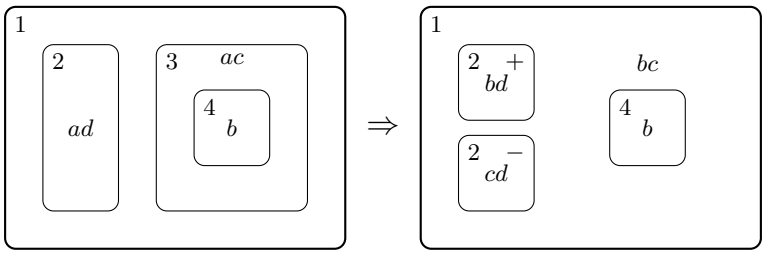

Abbildung 3: Transition von C_1 zu C_2 im System Π_2

Aufgrund des großen Umfangs an verschiedenen Regeln und Objekten, die für die Realisierung eines P-Systems zur Entscheidung von 3-SAT notwendig sind, verzichten wir an dieser Stelle auf eine explizite Angabe. Eine mögliche Realisierung samt Erklärung der gewählten Regeln findet sich in [Pău02]. Wir beschränken uns hier auf die Untersuchung eines kleineren Beispiels, um uns mit den Berechnungen eines P-Systems mit aktiven Membranen vertraut zu machen.

Beispiel 3.5 Wir betrachten das P-System mit aktiven Membranen und Eingabe $\Pi_2 = (O, \Sigma, H, \mu, \varepsilon, ad, \varepsilon, b, \{r_1, r_2, r_3\}, 3)$, wobei gilt:

$$O = \Sigma = \{a, b, c, d\} \qquad H = \{1, 2, 3, 4\} \qquad \mu = [_1[_2]_2[_3[_4]_4]_3]_1$$
$$r_1 = [_2a]_2^0 \to [_2b]_2^+[_2c]_2^- \qquad r_2 = [_3a]_3^0 \to b \qquad r_3 = [_1c]_1^0 \to [_1]_1^0a.$$

Eingabe sei das Wort $ac \in \Sigma^*$. Eine gültige Transition von der mit dieser Eingabe entstandenen, initialen Konfiguration C_1 erhalten wir durch Anwendung der Regeln r_1 und r_2, also die Teilung von Membran 2 und die gleichzeitige Auflösung von Membran 3. Das Resultat dieser Regelanwendung ist die neue Konfiguration C_2, in der nur Regel r_3 angewandt werden kann und somit a in die Umgebung abgegeben wird. Anschließend erreicht das System eine Haltekonfiguration, in der keine der Regeln r_1, r_2 und r_3 mehr angewandt werden kann: die Berechnung ist abgeschlossen. Abbildung 3 veranschaulicht die erste Transition und die beiden Konfigurationen C_1 und C_2, wobei zu jeder Membran ihre Polarisation angegeben ist, sofern diese $\neq 0$ ist.

Wie leicht nachzuprüfen ist, gibt es in diesem System keine anderen möglichen Berechnungen als die gezeigte, somit ist für Eingabe ac die von Π_2 generierte Sprache $L(\Pi_2) = \{a\}$.

4 Quantum Computing

Das Berechnungsmodell, dem wir uns in diesem Kapitel widmen, basiert auf den Prinzipien der Quantenphysik – wir bezeichnen es entsprechend als *Quantenrechnen* (Quantum Computing). Im Rahmen dieser Arbeit werden wir physikalische Grundlagen nur sehr oberflächlich und vereinfachend betrachten, gelegentlich auch als gegeben hinnehmen, und uns ausschließlich mit den Möglichkeiten beschäftigen, die diese Prinzipien für unser Berechnungsmodell bieten. Einen tieferen Einblick in die Problemstellungen und Möglichkeiten des Quantum Computing bietet beispielsweise [Hom13], dem einige Ausführungen dieses Kapitels entnommen sind.

Wir beginnen zunächst mit einer Einführung in die Begriffe Qubit und Quantenregister. In Abschnitt 4.2 werden wir uns anschließend mit den Möglichkeiten beschäftigen, auf diesen Quantenregistern zu rechnen, ehe wir in Abschnitt 4.3 das vollständige Berechnungsmodell, sogenannte Quantenschaltkreise, betrachten. Im letzten Abschnitt dieses Kapitels untersuchen wir schließlich den Grover-Algorithmus, der die effiziente Suche in einer unsortierten Datenbank ermöglicht und Grundlage einiger Quantenalgorithmen ist.

4.1 Qubits und Quantenregister

Qubits

Als *Qubit* bezeichnen wir, analog zu den Bits eines klassischen Computers, die kleinstmögliche Berechnungseinheit eines Quantencomputers. Ein klassisches Bit befindet sich stets entweder im Zustand 0 oder 1. Auch für ein Qubit gibt es nur diese beiden sogenannten *Basiszustände*, allerdings kann es sich in einer beliebigen Überlagerung beider Zustände befinden – man spricht von einer *Superposition*. Mögliche Zustände notieren wir zukünftig in der *Dirac-Notation* als $|\phi\rangle$, die beiden Zustände eines klassischen Bits schreiben wir also als $|0\rangle$ und $|1\rangle$. Damit lässt sich jeder Zustand $|\phi\rangle$ eines Qubits schreiben als $|\phi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$ mit $\alpha, \beta \in \mathbb{C}$, wobei wir fordern, dass stets $|\alpha|^2 + |\beta|^2 = 1$ gilt. Die beiden Zahlen α und β bezeichnen wir als *Amplituden*, sie drücken jeweils den Anteil der beiden Zustände $|0\rangle$ und $|1\rangle$ am Zustand des Qubits aus.

Um den Wert eines Qubits zu erfahren, müssen wir es *messen*. Durch eine solche Messung wird die Superposition zerstört, wodurch das Qubit in einen *nicht-überlagerten* Zustand übergeht, nämlich genau einen der Basiszustände $|0\rangle$ und $|1\rangle$.[3] Die Wahrscheinlichkeit für eine Messung von $|0\rangle$ ist dabei genau $|\alpha|^2$, für eine Messung von $|1\rangle$ entsprechend $|\beta|^2$.

Der Zustand eines Qubits lässt sich auch als Vektor eines zweidimensionalen Vektorraums über den komplexen Zahlen auffassen. Wir werden diese Darstellung zukünftig gelegentlich verwenden und den Zustand $\alpha\,|0\rangle + \beta\,|1\rangle$ eines Qubits

[3] Tatsächlich ist es auch möglich, bezüglich anderer Basen als $|0\rangle$ und $|1\rangle$ zu messen und entsprechend einen anderen nicht-überlagerten Zustand zu erhalten, wir beschränken uns hier aber auf die Messung bezüglich dieser Standardbasis.

als *Zustandsvektor* in der Form $(\alpha, \beta)^T$ schreiben. Die Menge

$$\mathcal{B} = \{|0\rangle := \begin{pmatrix}1\\0\end{pmatrix}, |1\rangle := \begin{pmatrix}0\\1\end{pmatrix}\}$$

bildet die *Standardbasis* dieses Vektorraums. Jeder Zustandsvektor kann als Linearkombination der Vektoren dieser Standardbasis geschrieben werden:

$$\begin{pmatrix}\alpha\\\beta\end{pmatrix} = \alpha \begin{pmatrix}1\\0\end{pmatrix} + \beta \begin{pmatrix}0\\1\end{pmatrix} = \alpha |0\rangle + \beta |1\rangle .$$

Quantenregister

Analog zum klassischen Register erhalten wir ein *Quantenregister*, indem wir mehrere Qubits miteinander verknüpfen. Diese Verknüpfung realisiert das Tensorprodukt $\otimes$. Durch $(n-1)$-fache Anwendung des Tensorprodukts erhalten wir aus n Qubits $|x_{n-1}\rangle, \ldots, |x_1\rangle, |x_0\rangle$ das Register $R = |x_{n-1}\rangle \otimes \ldots \otimes |x_1\rangle \otimes |x_0\rangle$. Eine solche Verknüpfung schreiben wir kompakter als $R = |x_{n-1}\rangle \ldots |x_1\rangle |x_0\rangle$, gelegentlich auch als $R = |x_{n-1}, \ldots x_1, x_0\rangle$. Gilt $x_i \in \{0, 1\}$ für alle i, so hat sich auch die Notation des Registers in der Form $R = |x_{n-1} \ldots x_1 x_0\rangle$ etabliert.

Beispiel 4.1 Verknüpfen wir zwei Qubits, so erhalten wir einen Zustandsvektor über einem vierdimensionalen Vektorraum, für den sich aus der Standardbasis $\mathcal{B} = \{|0\rangle, |1\rangle\}$ für einzelne Qubits die neue Basis

$$\begin{aligned}\mathcal{B}' &= \{|0\rangle \otimes |0\rangle, |0\rangle \otimes |1\rangle, |1\rangle \otimes |0\rangle, |1\rangle \otimes |1\rangle\}\\ &= \{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}\end{aligned}$$

ergibt. Für zwei Qubits $|x_1\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle$ und $|x_0\rangle = \beta_0 |0\rangle + \beta_1 |1\rangle$ in beliebigen Zuständen erhalten wir als Zustand des Registers $R = |x_1, x_0\rangle$ eine Superposition aus den vier Basiszuständen dieses Vektorraumes:

$$\begin{aligned}R &= (\alpha_0 |0\rangle + \alpha_1 |1\rangle) \otimes (\beta_0 |0\rangle + \beta_1 |1\rangle)\\ &= \alpha_0\beta_0(|0\rangle \otimes |0\rangle) + \alpha_0\beta_1(|0\rangle \otimes |1\rangle) + \alpha_1\beta_0(|1\rangle \otimes |0\rangle) + \alpha_1\beta_1(|1\rangle \otimes |1\rangle)\\ &= \alpha_0\beta_0 |00\rangle + \alpha_0\beta_1 |01\rangle + \alpha_1\beta_0 |10\rangle + \alpha_1\beta_1 |11\rangle .\end{aligned}$$

Allgemein sind die Zustände, in denen sich ein Quantenregister aus n Bit befinden kann, alle von der Form

$$R = \sum_{i=0}^{2^n-1} \alpha_i |b_i\rangle , \text{ wobei gilt: } \sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1.$$

Dabei ist $\alpha_i \in \mathbb{C}$ und b_i das n-stellige Wort über $\{0, 1\}$, welches der Binärdarstellung des zugehörigen Index i (gegebenenfalls mit führenden Nullen) entspricht. Auch der Zustand eines solchen Registers lässt sich als Zustandsvektor in der Form $(\alpha_0 \ldots \alpha_{2^n-1})^T$ angeben. Beim Messen des Registers erhalten wir jeden der Zustände b_i mit der Wahrscheinlichkeit $|\alpha_i|^2$.

In den folgenden Abschnitten werden wir häufig kurz von Bits und Registern reden, meinen damit aber stets Qubits und Quantenregister, wenn nicht ausdrücklich von *klassischen* Bits und Registern gesprochen wird.

4.2 Unitäre Transformationen

Wir betrachten nun Möglichkeiten, mit Qubits und Quantenregistern zu rechnen. Diese Berechnungen unterliegen einigen der Quantenphysik inhärenten Einschränkungen. Die Rechenschritte auf einem Register, die diesen Einschränkungen genügen, sind genau die, die sich als Multiplikation einer unitären Matrix mit dem Zustandsvektor des Registers darstellen lassen. Einen solchen Rechenschritt bezeichnen wir als *unitäre Transformation*. Aus dieser Tatsache geht als eine der erwähnten Beschränkungen sofort hervor, dass alle Berechnungen, die auf Quantenregistern ausgeführt werden, umkehrbar sein müssen, denn zu jeder unitären Matrix A existiert eine (leicht zu bestimmende) inverse Matrix A^{-1}.

Wir betrachten im Folgenden beispielhaft zwei unitäre Transformationen, die für viele Quantenalgorithmen von Bedeutung sind.

Definition 4.2 Die unitäre Transformation CNOT ist als die zur Matrix

$$C = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

zugehörige Abbildung definiert. Auf ein Quantenregister mit 2 Qubits angewandt, bewirkt CNOT : $|x, y\rangle \mapsto |x, x \oplus y\rangle$ die „kontrollierte Negation“ des zweiten Bits; letzteres wird genau dann negiert, wenn das erste Bit gleich 1 ist, andernfalls bleibt es unverändert.

Diese Funktion lässt sich erweitern auf Register $|x\rangle = |x_n \dots x_0\rangle$, $|y\rangle = |y_n \dots y_0\rangle$ beliebiger, aber gleicher Größe, indem die CNOT-Operation auf jedes Paar von Qubits $|x_i\rangle$, $|y_i\rangle$, $0 \leq i \leq n$ einzeln angewandt wird. Die so erhaltene Transformation bezeichnen wir als C_n.

Definition 4.3 Die 2×2-Matrix

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

heißt *Hadamard-Matrix*. Die zugehörige unitäre Transformation, die auf ein einzelnes Qubit angewandt werden kann, nennen wir *Hadamard-Transformation*.

Auch diese Transformation auf einzelne Qubits lässt sich erweitern zu einer Transformation H_n auf ein Quantenregister mit n Qubits. Die Matrizen H_n können rekursiv wie folgt bestimmt werden:

(1) $H_1 = H$

(2) $H_n = \frac{1}{\sqrt{2}} \begin{pmatrix} H_{n-1} & H_{n-1} \\ H_{n-1} & -H_{n-1} \end{pmatrix}$, $n \geq 2$

Die Anwendung von H_n auf das gesamte Register entspricht der Anwendung von H auf jedes einzelne Qubit des Registers.

Im Hinblick auf den am Ende dieses Kapitels über Quantum Computing vorgestellten Grover-Algorithmus lohnt es sich, schon jetzt einen genaueren Blick auf die Hadamard-Transformation und ihre Wirkung auf ein Register R zu werfen. Wir betrachten dazu zunächst ihre Wirkung auf ein einzelnes Bit.

Ein Qubit im Zustand $|0\rangle$ wird durch die Hadamard-Transformation H in eine Superposition versetzt, in der das Messen beider Basiszustände gleich wahrscheinlich ist. Es gilt:

$$H|0\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle).$$

Analog verhält es sich offensichtlich mit der Anwendung von H_n, $n \geq 2$ auf das Register $R = |0 \ldots 0\rangle$ aus n Qubits, denn sie entspricht der Anwendung von H auf jedes einzelne Qubit:

$$H_n|0 \ldots 0\rangle = \frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} |y\rangle .$$

Den so erhaltenen Zustand bezeichnen wir als *gleichgewichtete Superposition*.

Wir betrachten abschließend das Ergebnis der Anwendung auf ein Quantenregister der Größe n in beliebigem Basiszustand $|x\rangle$, $x \in \{0,1\}^n$:

$$H_n|x\rangle = \frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} (-1)^{x \circ y} |y\rangle .$$

Dabei bezeichne $x \circ y$ das Skalarprodukt der beiden Dualzahlen x und y. Auch hier erhalten wir eine gleichgewichtete Superposition, denn bis auf das Vorzeichen sind die Amplituden aller Basiszustände identisch.

Verschränkung

Ein quantenphysikalisches Phänomen, das für uns nur eine untergeordnete Rolle spielt, aber dennoch einer kurzen Betrachtung lohnt, ist die *Verschränkung* zweier oder mehrerer Qubits. In einem verschränkten Zustand beeinflusst die Manipulation eines Bits das Ergebnis der Messung eines anderen. Eine solche Verschränkung kann beispielsweise erreicht werden, indem auf das erste Bit eines Zwei-Bit-Registers R im Zustand $|00\rangle$ die Hadamard-Transformation angewandt wird, und anschließend die Operation CNOT auf beide Bits. Wie in [Hom13] gezeigt, ist der anschließende Zustand des Registers $R = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. Während vor der Messung beide Bits mit gleicher Wahrscheinlichkeit als 0 oder 1 gemessen werden können, steht nach Messung eines der beiden Bits sofort das Ergebnis bei Messung des anderen Bits fest: Wird $x \in \{0,1\}$ gemessen, so muss die Messung des anderen Bits auch x ergeben.

Wir werden diese Verschränkungen im Weiteren nicht ausnutzen (wie es etwa die Quantenteleportation tut, siehe [Hom13]), müssen sie allerdings bei der Konstruktion des Quantenteils unseres hybriden Systems berücksichtigen.

4.3 Quantenschaltkreise

Wir betrachten nun auf Basis der vorangegangenen Überlegungen ein vollständiges Berechnungsmodell für einen Quantenrechner: die *Quantenschaltkreise*. Analog zu klassischen Schaltkreisen sind solche Quantenschaltkreise aus mehreren Gattern aufgebaut. Jedes dieser Quantengatter realisiert eine unitäre Transformation T. Soll eine solche Transformation T auf ein Register $|x\rangle$ der Größe N angewandt werden, stellen wir dies wie in Abbildung 4 als Quantengatter dar, wobei $|y\rangle_N = T\,|x\rangle_N$ gilt. Durch den Index N geben wir für $N \neq 1$ die Größe des jeweiligen Registers an. Da Quantentransformationen stets umkehrbar sind, müssen $|x\rangle$ und $|y\rangle$ aus derselben Anzahl an Qubits bestehen.

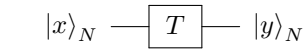

Abbildung 4: Gatter zur unitären Transformation T

Sind an einer Transformation n Register $|x_1\rangle, \ldots |x_n\rangle$ der jeweiligen Größe N_i, $1 \leq i \leq n$ beteiligt, und besteht die Ausgabe aus m Registern $|y_1\rangle, \ldots |y_m\rangle$ der Größe M_i, $1 \leq i \leq m$, so stellen wir das zugehörige Gatter wie in Abbildung 5 gezeigt dar.

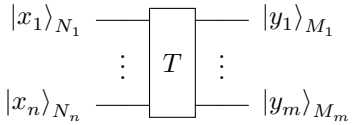

Abbildung 5: Gatter mit mehreren Ein- und Ausgängen

Dabei muss es nicht zwangsläufig gleich viele Eingänge wie Ausgänge geben, allerdings muss die aufsummierte Anzahl an Qubits in den Registern $|x_1\rangle, \ldots |x_n\rangle$ stets gleich der Anzahl in den Registern $|y_1\rangle, \ldots |y_m\rangle$ sein, es muss also die Gleichung $\sum_{i=1}^{n} N_i = \sum_{j=1}^{m} M_j$ erfüllt sein, weil die Anzahl an Qubits durch unitäre Transformationen nicht beeinflusst werden kann.

Die Transformation SWAP : $|x\rangle\,|y\rangle \mapsto |y\rangle\,|x\rangle$, welche die Position zweier Qubits $|x\rangle$ und $|y\rangle$ vertauscht, und die bereits vorgestellte Transformation CNOT erhalten jeweils eine von obigem Schema abweichende Darstellung, die in Abbildung 6 gezeigt wird.

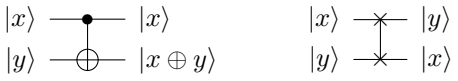

Abbildung 6: Gatter für CNOT (links) und SWAP (rechts)

Ein weiteres bedeutendes Gatter ist das *Fredkin-Gatter* FG, das auf drei Qubits angewandt die unitäre Transformation

$$\begin{aligned} \text{FG} : |0\rangle\,|y\rangle\,|z\rangle &\mapsto |0\rangle\,|y\rangle\,|z\rangle \\ |1\rangle\,|y\rangle\,|z\rangle &\mapsto |1\rangle\,|z\rangle\,|y\rangle \end{aligned}$$

realisiert. Bei FG handelt es sich um ein *universelles* Gatter: jede logische und mathematische Operation kann unter ausschließlicher Verwendung von Fredkin-Gattern – durch Zuhilfenahme einiger auf $|0\rangle$ oder $|1\rangle$ initialisierter Hilfsbits – realisiert werden. Beispielsweise lässt sich mit FG und einem zusätzlichen Bit im Zustand $|1\rangle$ das logische Oder zweier boolescher Variabler x und y berechnen: $\mathrm{FG}\,|x\rangle\,|y\rangle\,|1\rangle = |x\rangle\,|x \vee y\rangle\,|\neg x \vee y\rangle$.

Die Existenz universeller Quantengatter hat zur Konsequenz, dass es zu jeder berechenbaren Funktion $f : \{0,1\}^n \to \{0,1\}^m$, $n, m \in \mathbb{N}$ einen Quantenschaltkreis V_f gibt, der unter Verwendung von $k \in \mathbb{N}$ Hilfsbits genau diese Funktion berechnet.

Wie schon erwähnt, spielt das Messen in der Quantenphysik eine bedeutende Rolle. Obwohl eine Messung wegen ihrer Unumkehrbarkeit keine unitäre Transformation ist, stellen wir sie aufgrund ihrer großen Bedeutung in Form eines eigenen Gatters dar (siehe Abbildung 7). Ein solches Gatter bedeutet, dass der aktuelle Zustand von Register $|x\rangle$ gemessen wird. Dadurch wird die Superposition zerstört, im Register befindet sich anschließend das Ergebnis der Messung, der Basiszustand $|y\rangle$ mit $y \in \{0,1\}^N$.

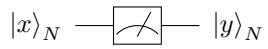

Abbildung 7: Messung als Quantengatter

Durch Zusammenschaltung mehrerer Quantengatter in derselben Weise, wie sie bei einem gewöhnlichen Schaltkreis erfolgt, erhalten wir einen Quantenschaltkreis. Zu beachten ist, dass der Schaltkreis keine Verzweigungen aufweisen darf. Ein wichtiges Resultat der Quantenphysik, das *No-Cloning-Theorem*, besagt, dass es nicht möglich ist, ein Qubit in beliebigem Zustand zu kopieren. Daher stellen wir grundsätzlich jedes Qubit durch eine eigene, von links nach rechts durchgehend gezeichnete Linie dar; allerdings werden wir, wie in den obigen Gattern, häufig mehrere Qubits zu einem Register zusammenfassen. Um trotz des No-Cloning-Theorems beliebige klassische Schaltkreise simulieren zu können, kann das Fredkin-Gatter eingesetzt werden, welches das Kopieren von Bits in den Basiszuständen $|0\rangle$ und $|1\rangle$ ermöglicht.

Wir verzichten an dieser Stelle auf die formale Beschreibung des Ablaufs einer Berechnung auf einem Quantenschaltkreis und verweisen stattdessen auf die Analogie zu einem klassischen Schaltkreis: von links nach rechts werden schrittweise alle durch die Gatter repräsentierten Transformationen ausgeführt. Ein ausführliches Beispiel für einen Quantenalgorithmus findet sich im folgenden Abschnitt.

4.4 Grover-Algorithmus

Viele Quantenalgorithmen nutzen den 1996 in [Gro96] veröffentlichten *Grover-Algorithmus* – auch wir werden in der Anwendung unseres hybriden Systems davon Gebrauch machen. Der Algorithmus ermöglicht es einem Quantencomputer, eine unsortierte Datenbank der Größe N in $\mathcal{O}(\sqrt{N})$ zu durchsuchen, während

jeder Algorithmus auf einem klassischen Computer dafür im ungünstigsten Fall $\mathcal{O}(N)$ Schritte benötigt.

Aufgabenstellung

Gegeben sei eine unsortierte Datenbank mit $N = 2^n$ Einträgen für ein $n \in \mathbb{N}$ und ein Suchkriterium C, dessen charakteristische Funktion f_C wir als

$$f_C(x) = \begin{cases} 1 & \text{wenn } x \text{ das Suchkriterium } C \text{ erfüllt,} \\ 0 & \text{andernfalls} \end{cases}$$

gegeben haben. Die N Einträge der Datenbank repräsentieren wir durch Wörter der Länge n über $\{0,1\}$. Wir gehen im Folgenden davon aus, dass die gegebene Datenbank genau *einen* Eintrag $\hat{x} \in \{0,1\}^n$ enthält, der unserem Suchkriterium C entspricht, für den also $f_C(\hat{x}) = 1$ gilt. Alle anderen Einträge entsprechen diesem Suchkriterium nicht, es gilt also $f_C(x) = 0$ für alle $x \in \{0,1\}^n \setminus \{\hat{x}\}$. Aufgabenstellung des Algorithmus ist es, eben diesen Eintrag $\hat{x}$ unter den gegebenen N Einträgen zu finden.

Weil die Funktion f_C im Allgemeinen nicht der Anforderung einer unitären Transformation genügt, invertierbar zu sein, gehen wir davon aus, dass sie uns als *Quantenorakel* – also als unitäre Transformation mit vernachlässigbarer Berechnungszeit – in der Form $U_{f_C} : |x, y\rangle \mapsto |x, f_C(x) \oplus y\rangle$ für $x \in \{0,1\}^n$, $y \in \{0,1\}$ gegeben ist.

Grundlegender Ablauf

Das dem Algorithmus zugrundeliegende Prinzip ist wesentlich für viele Quantenalgorithmen, der genaue Ablauf lässt sich in drei Schritte untergliedern:

(1) Das Register R im Zustand $|0\rangle_n$ wird durch Anwendung der Hadamard-Transformation in eine gleichgewichtete Superposition aller möglichen Zustände – und damit aller möglichen Datenbankeinträge – gebracht. Das zusätzliche Bit $|y\rangle$ wird in den Zustand $|1\rangle$ versetzt, anschließend wird darauf ebenfalls die Hadamard-Transformation angewandt. Es befindet sich danach im Zustand $H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$.

(2) Um die Wahrscheinlichkeit einer Messung des gesuchten Zustandes $\hat{x}$ zu vergrößern, wird in dieser Superposition der Betrag der Amplitude von $\hat{x}$ schrittweise vergrößert, bis er sein Maximum erreicht hat. Dieser als *Amplitudenverstärkung* (Amplitude Amplification) bezeichnete Schritt wird durch mehrfache Anwendung der sogenannten *Grover-Iteration* erreicht. Dem Ablauf und der Wirkungsweise dieser Iteration widmen wir uns gleich.

(3) Register R wird gemessen. Der gemessene Zustand ist für große N mit Wahrscheinlichkeit annähernd 1 gleich dem gesuchten Eintrag $\hat{x}$. Die Fehlerwahrscheinlichkeit kann durch mehrmalige Anwendung des Algorithmus beliebig reduziert werden.

Grover-Iteration

Der erste Schritt der Grover-Iteration ist die Anwendung des Quantenorakels U_{f_C} auf das Register R und das zusätzliche Qubit $|y\rangle$. Wir betrachten die Wirkung von U_{f_C} auf einen der Basiszustände $|x\rangle$ der gleichverteilten Superposition

$$|s\rangle = \frac{1}{\sqrt{N}} \sum_{x \in \{0,1\}^n} |x\rangle$$

und auf das Bit $|y\rangle$. Es gilt:

$$\begin{aligned}
U_{f_C}(|x\rangle \otimes |y\rangle) &= U_{f_C}(|x\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)) \\
&= |x\rangle \otimes \frac{1}{\sqrt{2}}(|f_C(x)\rangle - |1 \oplus f_C(x)\rangle) \\
&= |x\rangle \otimes (-1)^{f_C(x)} \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \\
&= (-1)^{f_C(x)}(|x\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)) \\
&= \begin{cases} -|x\rangle \otimes |y\rangle & \text{wenn } x = \hat{x}, \\ |x\rangle \otimes |y\rangle & \text{andernfalls.} \end{cases}
\end{aligned}$$

Die Auswirkung der Transformation U_{f_C} angewandt auf $|s\rangle \otimes |y\rangle$ ist also, dass das Vorzeichen der Amplitude des gesuchten Zustandes $|\hat{x}\rangle$ negiert wird, während die Vorzeichen aller anderen Amplituden unverändert bleiben. Register R befindet sich damit im Zustand

$$|s\rangle = \frac{1}{\sqrt{N}}(-|\hat{x}\rangle + \sum_{\substack{x \in \{0,1\}^n \\ x \neq \hat{x}}} |x\rangle).$$

Der zweite Schritt der Grover-Iteration ist die *Spiegelung* jeder Amplitude am Mittelwert aller Amplituden. Wir bezeichnen mit a_x, $x \in \{0,1\}^n$ im Folgenden die Amplitude des Zustands x im Register R. Der Mittelwert A dieser Amplituden nach Anwendung von U_{f_C} auf die gleichverteilte Superposition ist

$$A = \frac{1}{N}(\frac{N-1}{\sqrt{N}} - \frac{1}{\sqrt{N}}) = \frac{N-2}{N\sqrt{N}}$$

und die Spiegelung einer Amplitude entspricht der Abbildung $s : a_x \mapsto 2 \cdot A - a_x$. Vor Ausführung der Spiegelung gilt $a_{\hat{x}} = -\frac{1}{\sqrt{N}}$ und $a_x = \frac{1}{\sqrt{N}}$ für $x \neq \hat{x}$, alle Amplituden haben also noch denselben Betrag. Wir betrachten die Auswirkung der Spiegelung auf die Amplituden:

$$\begin{aligned}
s(a_{\hat{x}}) &= 2 \cdot \frac{N-2}{N\sqrt{N}} - (-\frac{1}{\sqrt{N}}) = \frac{3N-4}{N\sqrt{N}}, \\
s(a_x) &= 2 \cdot \frac{N-2}{N\sqrt{N}} - \frac{1}{\sqrt{N}} = \frac{N-4}{N\sqrt{N}} \quad \text{für } x \neq \hat{x}.
\end{aligned}$$

Durch die Spiegelung vergrößert sich also der Betrag der Amplitude von $\hat{x}$, während die Beträge aller anderen Amplituden kleiner werden[4] – die Wahrscheinlichkeit, bei einer Messung des Registers R den gesuchten Wert $\hat{x}$ zu erhalten, hat sich erhöht. Damit ist das Ziel der Grover-Iteration erreicht.

Offensichtlich muss die beschriebene Spiegelung eine unitäre Transformation sein, damit die Grover-Iteration mit einem Quantenschaltkreis realisierbar ist. Wir verzichten an dieser Stelle auf eine ausführliche Herleitung und beschränken uns darauf zu erwähnen, dass die Spiegelung der Hintereinanderausführung der unitären Transformationen H_n, $-R_N$ und H_n mit

$$R_N = \begin{pmatrix} -1 & 0 & \dots & 0 \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 1 \end{pmatrix} \in \mathbb{R}^{N \times N}$$

entspricht. Eine ausführliche Herleitung und die konkrete Realisierung von R_N finden sich beispielsweise in [Hom13].

Grover-Algorithmus

Durch mehrmalige Ausführung der Grover-Iteration kann die Amplitude von $\hat{x}$ schrittweise maximiert werden. Wird die Iteration allerdings zu oft angewandt, so kann sich $a_{\hat{x}}$ wieder verringern – die genaue Zahl $r(N)$ an Iterationen, die in Abhängigkeit der Größe N der Datenbank durchgeführt werden, ist für den Algorithmus daher von entscheidender Bedeutung. Es gilt $r(N) \approx \frac{\pi}{4}\sqrt{N} \in \mathcal{O}(\sqrt{N})$, an dieser Stelle verzichten wir auf eine Herleitung.

Wir ergänzen die Grover-Iterationen um die Vorbereitung der Register und die abschließende Messung. Der so erhaltene Grover-Algorithmus lässt sich durch den in Abbildung 8 gezeigten Quantenschaltkreis darstellen; dabei wird die umrahmte Grover-Iteration genau $r(N)$-mal hintereinander ausgeführt. Mit einer Wahrscheinlichkeit $p > \frac{1}{2}$ gilt für $N \geq 4$, dass der gemessene Zustand $x' \in \{0,1\}^n$ der gesuchte Eintrag ist, also dass $x' = \hat{x}$, für $N \gg 1$ erreicht p annähernd den Wert 1.

Durch wiederholte Anwendung des Grover-Algorithmus kann die Wahrscheinlichkeit, das korrekte Ergebnis zu erhalten, beliebig erhöht werden.

Varianten

Der Grover-Algorithmus kann leicht modifiziert werden, um für die Suche in einer unsortierten Datenbank, in der entweder genau ein Element, oder kein Element dem Suchkriterium entspricht, geeignet zu sein. Dazu wird nach der

[4] Im Fall einer Datenbank mit nur $N = 2$ Einträgen bewirkt die Spiegelung keine Amplitudenverstärkung des gesuchten Zustandes, für eine solche Datenbank ist die Suche nach $\hat{x}$ mit gegebenem U_{f_C} aber trivial. Daher gehen wir bei unseren Betrachtungen stets von $N \geq 4$ Einträgen aus.

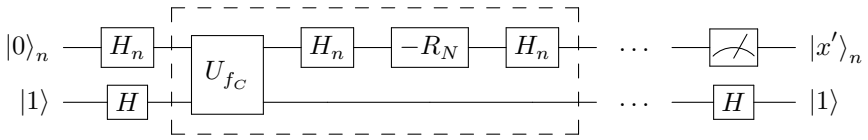

Abbildung 8: Grover-Algorithmus als Quantenschaltkreis

Messung auf den gemessenen Zustand x' nochmals die Transformation U_{f_C} angewandt. Gilt $f_C(x') = 1$, so ist x' offensichtlich der gesuchte Eintrag, andernfalls gibt es mit Wahrscheinlichkeit p wie oben keinen solchen Eintrag.

Wie in [Hom13] beschrieben, lässt sich der Grover-Algorithmus durch geeignete Wahl von $r(N)$ auch auf Datenbanken anwenden, in denen mehrere Elemente dem Suchkriterium entsprechen – selbst dann, wenn die Anzahl dieser Elemente nicht bekannt ist – und findet auch hier in $\mathcal{O}(\sqrt{N})$ eine Lösung. Wir werden auf diese Variationen nicht weiter eingehen, merken jedoch an, dass der Grover-Algorithmus so auch zum Lösen NP-vollständiger Probleme geeignet ist. Um beispielsweise SAT zu lösen, betrachten wir für eine aussagenlogische Formel ϕ, in der o.B.d.A. genau die Variablen $x_1, \ldots, x_n$, $n \in \mathbb{N}$ vorkommen, die unsortierte Datenbank mit $N = 2^n$ Einträgen, in der jeder Eintrag $(\alpha_n, \ldots, \alpha_1) \in \{0,1\}^n$ genau der Variablenbelegung $x_1 = \alpha_1, \ldots, x_n = \alpha_n$ entspricht. Als Suchkriterium verwenden wir die Funktion f_C, die eine gegebene Variablenbelegung genau dann auf 1 abbildet, wenn sie ϕ erfüllt, andernfalls auf 0. Zu dieser Funktion lässt sich leicht ein Quantenschaltkreis U_{f_C} konstruieren. Durch Anwendung des Grover-Algorithmus erhalten wir eine Variablenbelegung, für die wir testen, ob sie ϕ tatsächlich erfüllt. Ist dies der Fall, so gilt offensichtlich $\phi \in \text{SAT}$, andernfalls gilt mit Wahrscheinlichkeit p wie oben $\phi \notin \text{SAT}$.

Es bleibt zu erwähnen, dass der Zeitaufwand des beschriebenen Algorithmus trotz deutlicher Beschleunigung gegenüber dem naiven Testen aller 2^n Lösungen auf einem klassischen Computer immer noch von der Größenordnung $\sqrt{N} = 2^{n/2}$ ist. Durch weitere Modifikationen, die in dieser Arbeit nicht weiter vertieft werden sollen, lassen sich Quantenalgorithmen konstruieren, die SAT weitaus effizienter als der oben beschriebene Ansatz lösen, siehe etwa [Amb04].

5 Membrane-Quantum Computing

Wir kombinieren nun die beiden in den vorangegangenen Kapiteln vorgestellten Berechnungsmodelle zu einem hybriden System, dessen Berechnungen wir entsprechend als *Membrane-Quantum Computing* bezeichnen. Dabei verfolgen wir den in [RAB+14a] und [RAB+14b] beschriebenen Ansatz, in dem das Grundgerüst des hybriden Systems ein P-System bildet, wobei manche Membranen – sogenannte *Quantenmembranen* – zusätzlich mit einem Quantenrechner ausgestattet werden. In diesen Membranen kann mittels bestimmter Auslöser zwischen zwei Ebenen umgeschaltet werden: der *Membran-Ebene* und der *Quanten-Ebene*. In erstgenannter Ebene funktioniert die Quantenmembran wie eine gewöhnliche Membran des P-Systems. Findet ein Wechsel in die Quanten-Ebene statt, so beginnt der in der Membran enthaltene Quantenrechner seine Berechnung. Während dieser Berechnung kann die Membran nicht durch Regeln des zugehörigen P-Systems beeinflusst werden.

Im Folgenden werden wir die beiden Ebenen, auf denen die Berechnung abläuft, zunächst kurz einzeln betrachten, um auf Besonderheiten und notwendige Modifikationen einzugehen, die sich durch die Kombination der Systeme ergeben. Anschließend werden wir die Erkenntnisse aus diesen Betrachtungen zusammenfassen und daraus das hybride System konstruieren.

5.1 Membran-Ebene

Den Rahmen eines hybriden Systems bildet ein P-System Π, wobei wir an dieser Stelle keine Einschränkung bezüglich der genauen Art des P-Systems machen. Je nach Anwendung sind also sowohl klassische P-Systeme denkbar, als auch solche mit aktiven Membranen. Auch treffen wir keine allgemeine Festlegung bezüglich der Ein- und Ausgabe dieses P-Systems.

Wir erweitern das bisherige Modell eines solchen Systems und führen, zusätzlich zu den klassischen Membranen, die eingangs erwähnten Quantenmembranen ein, in denen sich ein Quantenschaltkreis befindet. In der Darstellung als Venn-Diagramm ergänzen wir, wie in Abbildung 9 gezeigt, das Label dieser Membranen um den Index q und kennzeichnen so ihre spezielle Funktion. Wir legen fest, dass nur elementare Membranen um diese Quantenfunktion erweitert werden können. Wird eine Quantenmembran geteilt, so erhalten die daraus entstandenen Membranen beide den in der ursprünglichen Membran enthaltenen Quantenschaltkreis; bei der Auflösung einer Quantenmembran verschwindet auch der zugehörige Schaltkreis. Mit $H_Q = \{h_1, \ldots, h_n\}$ bezeichnen wir im Folgenden die Menge der Labels aller Quantenmembranen. Für jedes Label h_i, $1 \leq i \leq n$ fordern wir, dass in der initialen Membranstruktur μ von Π mindestens eine Membran mit ebendiesem Label h_i enthalten ist.

Wir unterscheiden zwischen zwei verschiedenen Zuständen, die eine Quantenmembran h_i annehmen kann. Zu Beginn einer Berechnung des hybriden Systems ist jede Quantenmembran *aktiv*: die Regeln aus der Regelmenge R des P-Systems wirken in gewohnter Weise – also nichtdeterministisch, synchron und maximal parallel – auf h_i ein. Durch einen bestimmten Mechanismus, den wir

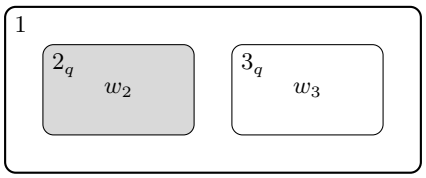

Abbildung 9: P-System mit aktiver und inaktiver Quantenmembran

später genauer beschreiben, ist es möglich, die Berechnung des zu h_i zugehörigen Quantenschaltkreises in Gang zu setzen. Dadurch wechselt die Membran für eine durch den Quantenschaltkreis festgelegte Anzahl an Zeitschritten in den Zustand *inaktiv*. In diesem Zustand können auf die Membran und die darin enthaltenen Objekte keine Regeln aus R angewandt werden. Eine solche Beschränkung ist notwendig, um beispielsweise das Teilen einer inaktiven Membran und der darin enthaltenen Qubits, das dem No-Cloning-Theorem widersprechen würde, zu verhindern. Erst nach Abschluss der Quantenberechnung wird die Membran wieder aktiv. In der grafischen Repräsentation eines P-Systems stellen wir inaktive Membranen zukünftig grau hinterlegt dar (siehe Abbildung 9).

5.2 Quanten-Ebene

In jeder Quantenmembran mit Label $h \in H_Q$ befindet sich ein zugehöriger Quantenschaltkreis ${q_h}^5$, dessen Aufgabe die Berechnung einer Funktion $f : \mathbb{N} \to \mathbb{N}$ ist. Jede solche Berechnung lässt sich grob in drei Phasen untergliedern: die *Initialisierung* der Register, die eigentliche *Berechnung* und die anschließende *Rückgabe* des gemessenen Ergebnisses an die Membran-Ebene. Sowohl die Initialisierung, als auch die Rückgabe sind wesentliche Bestandteile der Kommunikation zwischen den beiden Ebenen des Systems und sollen in Abschnitt 5.3 gesondert betrachtet werden.

In diesem Abschnitt beschäftigen wir uns daher nur mit der eigentlichen Berechnung des Quantenschaltkreises. Den genauen Schaltkreis werden wir dabei je nach zu lösendem Problem bestimmen, wir beschreiben hier daher nur die allgemeine Struktur, die all diesen Quantenschaltkreisen zugrunde liegen wird. Ein- und Ausgabe der Funktion f werden wir stets als Binärzahlen darstellen, die Eingabe zerlegen wir in zwei Teilwörter der Längen K und N über $\{0,1\}$, $K,\ N \in \mathbb{N}$. Wir betrachten also die Funktion $f : \{0,1\}^{K+N} \to \{0,1\}^M$ mit $M \in \mathbb{N}$ und die zugehörige unitäre Transformation V_f, die der Schaltkreis aus Abbildung 10 realisiert.

Die Ergebnisse aus Abschnitt 4.3 stellen sicher, dass sich zu jeder berechenbaren Funktion f, gegebenenfalls durch Ergänzung eines Registers $|w\rangle$ aus $R \in \mathbb{N}$ *Hilfsbits*, ein solcher Schaltkreis konstruieren lässt. Die Eingabe ist dabei auf die

[5] Besitzen mehrere Membranen dasselbe Label h, so bezeichnet q_h keinen konkreten Quantenschaltkreis, sondern den Typ der Quantenschaltkreise, die in all diesen Membranen enthalten sind.

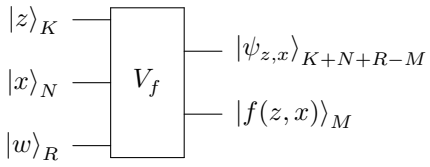

Abbildung 10: Quantenschaltkreis zur Berechnung von f

Register $|z\rangle$ und $|x\rangle$ aufgeteilt; nach Anwendung von V_f enthalten die unteren M Qubits das Ergebnis $|f(z,x)\rangle_M$. Um die Invertierbarkeit der Transformation zu gewährleisten, ist im Allgemeinen ein zusätzliches Ausgaberegister der Größe $K+N+R-M$ notwendig, dessen Zustand wir mit $|\psi_{z,x}\rangle$ kennzeichnen.

Dieser erste Schaltkreis zur Realisierung von f hat den zumeist unerwünschten Nebeneffekt, dass die Qubits aus den beiden Registern $|\psi_{z,x}\rangle$ und $|f(z,x)\rangle$ nach abgeschlossener Berechnung gegebenenfalls miteinander verschränkt sind. Wir modifizieren ihn daher unter Anwendung der zu V_f inversen Transformation $V_f^\dagger$ und der kontrollierten Negation C_M für M Qubits, ergänzen ein Register $|y\rangle_M$ und konstruieren einen neuen Schaltkreis, der die Verschränkung der Qubits nach abgeschlossener Berechnung ausschließt. Die Darstellung dieses Schaltkreises in Abbildung 11 ist aus [RAB+14b] übernommen.

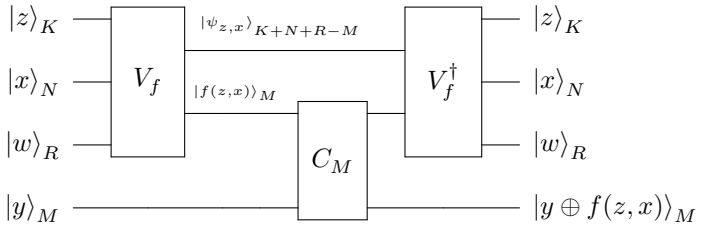

Abbildung 11: Modifizierter Quantenschaltkreis zur Berechnung von f

Die Bedeutung der einzelnen Register dieses modifizierten Schaltkreises ergibt sich wie folgt:

- Die Qubits aus Register $|z\rangle$ stellen den Teil der Eingabe dar, der durch das P-System Π initialisiert wird. Sie bieten der Quantenmembran in der Membran-Ebene damit die Möglichkeit, Einfluss auf den Quantenschaltkreis zu nehmen und so die Berechnung der Quanten-Ebene zu steuern.

- Das Register $|x\rangle$ bildet den zweiten Teil der Eingabe. Es enthält die Bits, auf welche die von $|z\rangle$ „gesteuerte“ Transformation V_f angewandt wird. Zu Beginn der Quantenrechnung befindet sich $|x\rangle$ im Zustand $|0\rangle_N$. Häufig wird allerdings der erste Schritt von V_f sein, $|x\rangle$ durch Anwendung der Hadamard-Transformation in eine gleichgewichtete Superposition aller Basiszustände zu versetzen, um – wie etwa im Fall des Grover-Algorithmus – für alle möglichen Werte von $|x\rangle$ zeitgleich Berechnungen durchzuführen.

- Das mit $|0\rangle_M$ initialisierte Register $|y\rangle$ enthält nach abgeschlossener Quantenrechnung das Ergebnis $f(z, x)$. Dieses Register wird abschließend gemessen, das Ergebnis der Messung wird – in geeigneter Weise codiert – an die Membran-Ebene zurückgegeben.

- Das Register $|w\rangle$ enthält Hilfsbits, die sich zu Beginn der Quantenrechnung alle im Zustand $|0\rangle$ befinden. Sie stellen sicher, dass die Funktion f als unitäre Transformation V_f realisiert werden kann. Diese Hilfsbits werden wir in unseren weiteren Betrachtungen vernachlässigen.

Wir gehen im Folgenden davon aus, dass jeder Quantenschaltkreis des hybriden Systems genau die in Abbildung 11 gezeigte Struktur aufweist. Bei der Spezifizierung eines hybriden Systems ist es daher ausreichend, anstelle der gesamten Quantenschaltkreise q_h nur den jeweiligen Quantenschaltkreis zur Realisierung von V_f sowie die Größen der zugehörigen Register anzugeben.

5.3 Hybrides System

In einem System, in dem zwei verschiedene Berechnungsmodelle kombiniert werden, spielt die Synchronisation zwischen diesen beiden Modellen eine entscheidende Rolle – die Dauer einer Quantenberechnung muss aus Sicht des P-Systems beschrieben werden können. Daher führen wir den Begriff des *Zeitschritts* eines hybriden Systems ein und wir legen fest, dass eine Transition des P-Systems genau einen solchen Zeitschritt benötigt. Die Dauer einer Quantenberechnung werden wir für jeden Quantenschaltkreis einzeln festlegen.

Wir fassen nach dieser Vorüberlegung die Erkenntnisse aus den beiden vorhergehenden Abschnitten zusammen und konstruieren damit das hybride System. Schwerpunkt dieses Abschnitts soll neben einer formalen Definition des Systems die Beschreibung des Mechanismus zum Wechsel zwischen Membran- und Quanten-Ebene sein.

Definition 5.1 Ein *hybrides System* ist ein Tupel

$$\beta = (\Pi, T, T', H_Q, Q_K, Q_M, Inp, Outp, t, q_{h_1}, \ldots, q_{h_n}),$$

wobei gilt:

(1) Π ist ein P-System mit Membranstruktur μ, Objektalphabet O und Regelmenge R, das, wie in Abschnitt 5.1 beschrieben, über Quantenmembranen verfügt,

(2) $T,\ T' \in O$ sind zwei spezielle Symbole aus dem Objektalphabet des P-Systems, wobei wir im Folgenden T als *Trigger* und T' als *Signal* bezeichnen,

(3) $H_Q = \{h_1, \ldots, h_n\}$ ist die Menge der Labels aller Quantenmembranen und damit Teilmenge der Labels aller Membranen aus μ,

(4) $Q_K : H_Q \to \mathbb{N}$ und $Q_M : H_Q \to \mathbb{N}$ sind Funktionen, die zu jeder Quantenmembran $h \in H_Q$ die Größen K und M der jeweils zugehörigen Register $|z\rangle_K$ und $|y\rangle_M$ angeben,

(5) $Inp : H_Q \to \mathcal{P}(O)$ und $Outp : H_Q \to \mathcal{P}(O)$ sind Funktionen für die Kommunikation zwischen den Ebenen des Systems, wobei wir Inp zukünftig *Eingabefunktion* und $Outp$ entsprechend *Ausgabefunktion* nennen,

(6) $t : H_Q \to \mathbb{N}$ ist die *Zeitfunktion*, welche für jede Quantenmembran die Dauer einer Quantenberechnung in Zeitschritten des hybriden Systems angibt,

(7) q_{h_i}, $1 \leq i \leq n$ ist ein mit Membran h_i assoziierter Quantenschaltkreis, welcher von der im Abschnitt 5.2 beschriebenen Struktur ist.

Wir betrachten zunächst die Bedeutung der beiden Symbole T und T'. Aufgabe des Triggers T ist es, die Quantenberechnung einer Quantenmembran zu initiieren, während das Signal T' der Membran mitteilt, dass eine solche Berechnung abgeschlossen wurde. Damit sind diese beiden Symbole für den Wechsel zwischen Membran- und Quanten-Ebene von entscheidender Bedeutung. Abbildung 12 veranschaulicht diesen Zusammenhang.

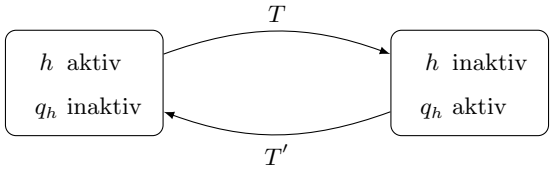

Abbildung 12: Wechsel zwischen Membran- und Quanten-Ebene

Den durch T ausgelösten Wechsel von der Membran- in die Quanten-Ebene bezeichnen wir im Folgenden als *Initialisierung*, weil zentraler Bestandteil dieses Schritts die Initialisierung der Quantenregister ist. Analog nennen wir den Wechsel von der Quanten-Ebene zurück in die Membran-Ebene *Rückgabe*. Den Ablauf dieser beiden Übergänge betrachten wir jeweils getrennt für eine einzelne Quantenmembran h.

Initialisierung

Wir definieren die Eingabefunktion $Inp : H_Q \to \mathcal{P}(O)$ für alle Anwendungen, die in dieser Arbeit gezeigt werden, in der gleichen Weise:

$$Inp(h) := \{I_{k,b} \mid 0 \leq k \leq Q_K(h) - 1,\ b \in \{0,1\}\} \cup \{T\} \text{ für alle } h \in H_Q.$$

Insbesondere fordern wir also, dass alle Symbole der Form $I_{k,b}$ im Objektalphabet O des P-Systems enthalten sind. Die Bedeutung eines Vorkommens von $I_{k,b}$ in Membran h zum Zeitpunkt der Initialisierung ist, dass im Quantenschaltkreis q_h das k-te Bit des Steuerregisters $|z\rangle$ auf den Wert b gesetzt werden soll.

Wenn durch eine der Regeln aus R in einem Zeitschritt $c \in \mathbb{N}$ der Trigger T zu Membran h hinzugefügt wird, beginnt die Initialisierung; nach Abschluss der zugehörigen Transition von Π, also nach Anwendung sämtlicher Regeln in Zeitschritt c, aber noch vor Beginn der folgenden Transition, also des $(c+1)$-ten Zeitschritts, werden folgende Modifikationen vorgenommen, wobei w'_h die in der betrachteten Konfiguration zur Membran zugehörige Multimenge sei:

(1) Der Zustand der Membran h wird auf inaktiv gesetzt. Damit können in den Zeitschritten $c+1, \ldots, c+t(h)$ keine Regeln aus R auf diese Membran und die darin enthaltenen Objekte angewandt werden.

(2) Aus $w'_h = v_1 \ldots v_k$ wird jedes Objekt $v_i, 1 \leq i \leq k$ entfernt, wenn $v_i \in Inp(h)$ gilt, alle anderen Objekte bleiben unberührt.

(3) Der Quantenschaltkreis q_h wird in den initialen Zustand $|z\rangle |x\rangle |w\rangle |y\rangle = |z_{K-1} \ldots z_0\rangle_K |0\rangle_N |0\rangle_R |0\rangle_M$ mit

$$z_i = \begin{cases} 1 & \text{wenn } |w'_h|_{I_{i,1}} \geq 1, \\ 0 & \text{andernfalls} \end{cases}$$

für $0 \leq i \leq K-1$ versetzt. Durch diese Definition sind auch die Randfälle abgedeckt, in denen keine oder mehrere Instanzen von $I_{k,0}$ und $I_{k,1}$ in w_h vorkommen. Wie in [RAB+14a] gezeigt, lässt sich auch ein System konstruieren, das in solchen Fällen das Qubit $|z_i\rangle$ in eine Superposition beider Basiszustände versetzt, die sich mit $s = |w'_h|_{I_{i,0}}$ und $t = |w'_h|_{I_{i,1}}$ für $s + t > 0$ zu $|z_i\rangle = \frac{1}{\sqrt{s^2+t^2}}(s|0\rangle + t|1\rangle)$ ergibt. Wir werden diesen Ansatz hier allerdings nicht weiter verfolgen.

Im Zeitschritt $c+1$ beginnt der Quantenrechner seine Berechnung, die genau $t(h)$ Schritte andauert und somit nach Zeitschritt $c + t(h)$ abgeschlossen ist.

Rückgabe

Besteht das Ausgaberegister $|y\rangle$ aus mehreren Qubits, so definieren wir die Ausgabefunktion $Outp : H_Q \to \mathcal{P}(O)$ analog zur Eingabefunktion Inp als

$$Outp(h) := \{O_{k,b} \mid 0 \leq k \leq Q_M(h) - 1,\ b \in \{0,1\}\} \cup \{T'\} \text{ für alle } h \in H_Q,$$

wobei wir wieder voraussetzen, dass die Objekte $O_{k,b}$ in O enthalten sind.

In vielen Fällen werden wir das Quantensystem q_h als Entscheidungssystem nutzen und nur ein einziges Ausgabebit benötigen. In diesen Fällen definieren wir $Outp$ als

$$Outp(h) := \{\texttt{Ja}, \texttt{Nein}, T'\} \text{ für alle } h \in H_Q.$$

Wurde nach dem c-ten Zeitschritt eine Quantenberechnung initialisiert, ist diese nach Zeitschritt $c + t(h)$ abgeschlossen. Noch vor dem $(c + t(h) + 1)$-ten Zeitschritt werden folgende Modifikationen vorgenommen:

(1) Das Register $|y\rangle$ des Quantenschaltkreises q_h wird gemessen. Anschließend befindet das Register sich in einem Basiszustand der Form $|y_{M-1}, \ldots, y_0\rangle$, $y_k \in \{0, 1\}$, $0 \leq k \leq M-1$.

(2) Der Quantenmembran werden, falls $M > 1$ gilt, sämtliche Objekte aus der Menge $\{O_{k,y_k} \mid 0 \leq k \leq M-1\} \cup \{T'\}$ hinzugefügt. Besteht Register $|y\rangle$ hingegen aus nur einem einzelnen Qubit, so wird zur Multimenge der Membran das Signal T' hinzugenommen und im Fall $|y\rangle = |1\rangle$ das Symbol `Ja`, andernfalls das Symbol `Nein`.

(3) Membran h wird aktiv gesetzt und kann im Zeitschritt $c + t(h) + 1$ wieder von Regeln aus R beeinflusst werden.

Bestimmung der Zeitfunktion

In den Definitionen von Initialisierung und Rückgabe spielt die Zeitfunktion t eine entscheidende Rolle – um die Synchronität der Berechnung des hybriden Systems zu gewährleisten, muss mit Hilfe von t die Dauer einer Quantenrechnung in Zeitschritten von β ausgedrückt werden. Es muss also bekannt sein, wie viele Transitionen Π während der Quantenrechnung durchführen kann.

Auf die grundsätzliche Frage, wie dieser Zeitparameter bestimmt werden kann, geht auch [RAB+14b] ein; dort wird als Abschätzung für jede Quantenrechnung der Wert 3 angegeben – damit werden der Initialisierung und Rückgabe, die in der obigen Beschreibung aus Sicht des hybriden Systems *zwischen* zwei Zeitschritten stattfinden, und der eigentlichen Rechnung jeweils ein Zeitschritt zugeteilt.

Wir werden diese Abschätzung in der Konstruktion unseres hybriden Systems in Abschnitt 6.1 übernehmen. Um die Funktion des Systems unabhängig von der Wahl der Zeitfunktion zu garantieren, werden wir dort zusätzlich dafür sorgen, dass sich das P-System während der Quantenberechnung in einer Haltekonfiguration befinden. Erst die Objekte der Rückgabe stoßen die Berechnung des P-Systems wieder an. So hängt die Korrektheit des Ergebnisses nicht von einer exakten Synchronisierung zwischen den beiden Ebenen ab.

Berechnungen

Wir ergänzen die Ausführungen aus [RAB+14a] und [RAB+14b], denen die vorhergehende Beschreibung des hybriden Systems entnommen ist, um Betrachtungen der Begriffe *Konfiguration*, *Transition* und *Berechnung* eines hybriden Systems, die wir für P-Systeme bereits definiert haben.

Eine Berechnung des hybriden Systems β setzt sich offensichtlich aus der Berechnung des zugehörigen P-Systems Π und denen der darin enthaltenen Quantenschaltkreise q_h, $h \in H_Q$ zusammen. Die Begriffe *Konfiguration* und *Transition* können wir mit geringfügigen Modifikationen wie im Fall des entsprechenden P-Systems behandeln. Jede Konfiguration muss allerdings um einige Komponenten erweitert werden: um die seit Beginn der Berechnung vergangenen Zeitschritte, den Zustand jeder Quantenmembran $h \in H_Q$, den aktuellen

Zustand $|z\rangle\,|x\rangle\,|w\rangle\,|y\rangle$ des zugehörigen Quantenschaltkreises q_h sowie die bisherige Dauer jeder aktiven Quantenberechnung in Zeitschritten.

Wir verzichten an dieser Stelle auf die formale Definition und Darstellung einer so erhaltenen Konfiguration. Eine Transition von einer Konfiguration C_1 des hybriden Systems zu einer neuen Konfiguration C_2 erhalten wir durch Inkrementierung aller Zeitparameter, Aktualisierung des Zustandes aller Quantenmembranen und – sofern sich Π in keiner Haltekonfiguration befindet – Anwendung einer gültigen Transition auf die aktuelle Konfiguration von Π.

Zu beachten ist, dass der Begriff der *Haltekonfiguration* für hybride Systeme neu definiert werden muss, denn es ist durchaus möglich, dass sich Π in einer Haltekonfiguration befindet, während β noch aktiv ist: Ein hybrides System befindet sich genau dann in einer Haltekonfiguration, wenn sich das zugehörige P-System Π in einer Haltekonfiguration befindet *und* keiner der Quantenschaltkreise $q_{h_1}, \ldots, q_{h_n}$ eine Berechnung durchführt, also keine Quantenmembran inaktiv ist.

Die *Eingabe* und das *Ergebnis* der Berechnung eines hybriden Systems sind für uns gleichbedeutend mit Eingabe und Ergebnis des zugehörigen P-Systems. Erreicht das hybride System β eine Haltekonfiguration, so entspricht daher das Ergebnis der Berechnung von β dem Ergebnis von Π – also je nach Definition entweder den Objekten in einer ausgewählten Ergebnismembran oder den Objekten, die an die Umgebung abgegeben wurden. Betrachten wir hybride Systeme, deren P-System Π über eine Eingabe verfügt, so bezeichnen wir diese auch als Eingabe von β.

Ein ausführliches Beispiel, das den Ablauf der Berechnung eines hybriden Systems detailliert aufzeigt, findet sich am Ende von Abschnitt 6.1.

6 Anwendung hybrider Systeme

Wir zeigen nun, wie zwei ausgewählte Probleme aus den Komplexitätsklassen Σ_2^{P} und Π_2^{P} mittels hybrider Systeme gelöst werden können. Bei diesen für ihre jeweilige Komplexitätsklasse vollständigen Problemen handelt es sich um abgeschwächte Versionen des Problems QBFSAT, das nach der Erfüllbarkeit quantifizierter aussagenlogischer Formeln fragt. Für beide Probleme findet sich in der Literatur keine einheitliche Bezeichnung, wir nennen sie daher der zugehörigen Komplexitätsklasse entsprechend Π_2^{P}-QBFSAT und Σ_2^{P}-QBFSAT. Erstgenanntes Problem findet sich gelegentlich auch unter dem Namen $\forall\exists$3-SAT, letztgenanntes unter den Namen QBF_2 und QSAT_2.

6.1 Π_2^{P}-QBFSAT

Problembeschreibung

Gegeben seien zwei Zahlen $m, l \in \mathbb{N}$ und die quantifizierte aussagenlogische Formel

$$\psi = \forall x_1. \cdots \forall x_n. \exists x_{n+1}. \cdots \exists x_m. \phi,\ 0 \leq n \leq m,$$

wobei ψ geschlossen und ϕ eine quantorenfreie Formel in 3-KNF[6] mit genau l Klauseln, also von der Form $\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_l$ für $C_i = (L_{i,1} \vee L_{i,2} \vee L_{i,3})$, $L_{i,j} \in \{x_{i_j}, \neg x_{i_j}\}$, $1 \leq i \leq l$, $1 \leq i_j \leq m$, $1 \leq j \leq 3$ ist.

Die Menge aller Formeln dieser Art bezeichnen wir als Π_2^{P}-QBF. Eine solche Formel liegt genau dann in Π_2^{P}-QBFSAT, wenn sie erfüllbar ist.

Hybrides System

Das hybride System β, das Π_2^{P}-QBFSAT für eine Formel ψ der obigen Form mit festem l und m, aber beliebigem $n \leq m$ entscheidet, ist das Tupel

$$\beta = (\Pi, T, T', \{2\}, Q_K, Q_M, Inp, Outp, t, q_2),$$

wobei gilt:

$$\begin{aligned} Q_K(2) &= 3 * l * (1 + \lceil \log_2(m+1) \rceil) \\ Q_M(2) &= 1 \\ Inp(2) &= T \cup \{I_{k,b} \mid 0 \leq k \leq Q_K(2) - 1, b \in \{0,1\}\} \\ Outp(2) &= \{T', \texttt{Ja}, \texttt{Nein}\} \\ t(2) &= 3 \end{aligned}$$

Das P-System Π und das Quantensystem q_2 werden in den folgenden Abschnitten genauer beschrieben. Ist ψ erfüllbar, so gibt das hybride System `Ja` aus, andernfalls `Nein`.

[6] Mittels *Tseitin-Transformation* kann jede Formel $\forall x_1. \cdots \forall x_n. \exists x_{n+1}. \cdots \exists x_m. \phi$, $0 \leq n \leq m$ in linearer Zeit in eine erfüllbarkeitsäquivalente Formel $\forall x_1. \cdots \forall x_n. \exists x_{n+1}. \cdots \exists x_l. \phi'$, $m \leq l$ überführt werden, wobei ϕ' in 3-KNF vorliegt.

Der prinzipielle Ablauf der Berechnung von β lässt sich in drei Schritte gliedern:

(1) Die Membran-Ebene erzeugt in linearer Zeit alle 2^n möglichen Belegungen für die allquantifizierten Variablen $x_1, \ldots, x_n$.

(2) Für jede dieser möglichen Belegungen $(\alpha_n, \ldots, \alpha_1) \in \{0,1\}^n$ betrachten wir die Formel $\psi' = \exists x_{n+1}. \cdots \exists x_m. \phi_{x_1=\alpha_1,\ldots,x_n=\alpha_n}$. Diese Formel wird an die Quanten-Ebene weitergegeben, die sie auf Erfüllbarkeit testet. Ist die Formel erfüllbar, wird `Ja` an die Membran-Ebene zurückgegeben, andernfalls `Nein`.

(3) Erreicht mindestens eine Instanz von `Nein` die Membran-Ebene, dann gibt es mindestens eine Belegung für $x_1, \ldots, x_n$, mit der ψ' unerfüllbar ist. Damit ist auch ψ unerfüllbar, die Membran-Ebene gibt `Nein` aus. Andernfalls gibt es für jede Belegung der allquantifizierten Variablen eine erfüllende Belegung der existenziell quantifizierten Variablen, die Ausgabe ist `Ja`.

Membran-Ebene

Wir verwenden für die Membran-Ebene des hybriden Systems ein P-System mit aktiven Membranen und erweitern es, wie in Definition 3.4 eingeführt, um ein Eingabealphabet Σ und eine Eingabemembran i_{in}. Aufgabe dieses P-Systems ist es, alle möglichen Variablenbelegungen für $x_1, \ldots, x_n$ zu erzeugen, anschließend die Eingabe für die Quanten-Ebene vorzubereiten, nach abgeschlossener Quantenrechnung das Ergebnis auszuwerten und schließlich `Ja` oder `Nein` in die Umgebung abzugeben.

Wir betrachten zunächst eine geeignete Codierung von ψ für das P-System. Dazu nutzen wir das Eingabealphabet

$$\Sigma = \{\forall_i \mid 0 \leq i \leq m\} \cup \{x_{i,j,k,0}, \bar{x}_{i,j,k,0} \mid 1 \leq i \leq m,\ 1 \leq j \leq l, 1 \leq k \leq 3\}.$$

Dabei hat ein Vorkommen von $x_{i,j,k,0}$ in der Eingabe die Bedeutung, dass x_i in Klausel C_j an k-ter Stelle vorkommt. Entsprechend meint $\bar{x}_{i,j,k,0}$, dass C_j an k-ter Stelle das Literal $\neg x_i$ enthält. Das Zeichen $\forall_i$ bedeutet, dass alle Variablen $x_1, \ldots x_i$ allquantifiziert sind. Aufgrund der Geschlossenheit von ϕ ist es nicht nötig zu codieren, welche Variablen existenziell quantifiziert sind. Beispielsweise codieren wir die Formel $\psi = \forall x_1. \forall x_2. \exists x_3. \exists x_4. (\neg x_1 \vee x_3 \vee x_4) \wedge (x_1 \vee \neg x_2 \vee \neg x_3)$ als die Multimenge $\forall_2 \bar{x}_{1,1,1,0} x_{3,1,2,0} x_{4,1,3,0} x_{1,2,1,0} \bar{x}_{2,2,2,0} \bar{x}_{3,2,3,0} \in \Sigma^*$.

Mit dieser Vorüberlegung können wir nun das vollständige P-System angeben als

$$\Pi = (O, \Sigma, \{1,2\}, [_1[_2]_2]_1, \varepsilon, \varepsilon, R, 2)$$

mit dem Objektalphabet

$$\begin{aligned} O = \Sigma &\cup Inp(2) \cup Outp(2) \cup \{o, \texttt{Ja}'\} \\ &\cup \{0_{j,k}, 1_{j,k} \mid 1 \leq i \leq l,\ 1 \leq k \leq 3\} \\ &\cup \{x_{i,j,k,t}, \bar{x}_{i,j,k,t} \mid 1 \leq i \leq m,\ 1 \leq j \leq l,\ 1 \leq k \leq 3,\ 1 \leq t \leq m\}. \end{aligned}$$

Die Zeichen $0_{j,k}$ und $1_{j,k}$ haben dabei jeweils die Bedeutung, dass in Klausel j an k-ter Stelle die entsprechende Konstante 0 oder 1 steht.

Die Menge der Entwicklungsregeln R lässt sich in die vier verschiedenen Aufgabenbereiche *Teilung*, *Variablenersetzung*, *Binärcodierung* und *Auswertung* unterteilen. Wir betrachten im Folgenden einige Regelmengen und einzelne Regeln gruppiert nach ihrer Funktion, wobei wir zu jedem Aufgabenbereich und den jeweils zugehörigen Regeln eine Erklärung geben:

- **Teilung**: Membran 2 wird n-mal geteilt, wobei nach jeder Teilung eine der allquantifizierten Variablen auf 0 oder 1 gesetzt werden soll.
  - $T_1 = \{[\forall_i]_2^e \to [\forall_{i-1}]_2^+ [\forall_{i-1}]_2^- \mid 1 \leq i \leq m,\, e \in \{0, +, -\}\}$
    Bei jeder Teilung wird die Variable $\forall_i$ dekrementiert. Damit entspricht der Index i der Anzahl allquantifizierter Variablen, denen noch kein Wert zugewiesen wurde. Sobald $i = 0$ gilt, wird nicht mehr weiter geteilt. Positive Polarisation $(+)$ entspricht der Festlegung, dass die nächste allquantifizierte Variable mit 1 belegt wird, negative Polarisation $(-)$ bedeutet entsprechend eine Belegung mit 0.

- **Variablenersetzung**: Der letzte Index jedes Objekts $x_{i,j,k,t}$ und $\bar{x}_{i,j,k,t}$ zählt die Anzahl vollzogener Teilungen, wir bezeichnen diesen Index daher als *Teilungsindex*. Nach der i-ten Teilung, also wenn $i = t$ gilt, muss jedes Vorkommen von x_i durch 0 oder 1 ersetzt werden.
  - $V_1 = \{[L_{i,j,k,0} \to L_{i,j,k,1}]_2^0 \mid L_{i,j,k,c} \in \{x_{i,j,k,c}, \bar{x}_{i,j,k,c}\},\, 0 \leq c \leq 1,\, 1 \leq i \leq m,\, 1 \leq j \leq l,\, 1 \leq k \leq 3$
    Der Teilungsindex aller Literale wird im ersten Schritt auf 1 gesetzt.
  - $V_2 = \{[L_{i,j,k,t} \to L_{i,j,k,t+1}]_2^e \mid L_{i,j,k,c} \in \{x_{i,j,k,c}, \bar{x}_{i,j,k,c}\},\, t \leq c \leq t{+}1,\, t < i \leq m,\, 1 \leq j \leq l,\, 1 \leq k \leq 3,\, 0 \leq t \leq m-1,\, e \in \{+, -\}\}$
    Ist der Index i des Literals L_i größer als Teilungsindex t, bleibt das Literal unverändert und der Teilungsindex wird weitergezählt.
  - $V_3 = \{[x_{i,j,k,i} \to 1_{j,k}]_2^+ \mid 1 \leq i \leq m,\, 1 \leq j \leq l,\, 1 \leq k \leq 3\}$
    Nach der i-ten Teilung wird in der Membran mit Polarisation + jedes Vorkommen von x_i auf 1 gesetzt. Dabei muss die Information, in welcher Klausel und an welcher Stelle x_i vorkam, erhalten bleiben.
  - $V_4 = \{[\bar{x}_{i,j,k,i} \to 0_{j,k}]_2^+ \mid 1 \leq i \leq m,\, 1 \leq j \leq l,\, 1 \leq k \leq 3\}$
    Analog zu V_3 wird durch diese Regel $\neg x_i$ nach der i-ten Teilung auf 0 gesetzt.
  - $V_5 = \{[x_{i,j,k,i} \to 0_{j,k}]_2^- \mid 1 \leq i \leq m \mid 1 \leq j \leq l,\, 1 \leq k \leq 3\}$
    Negative Polarisation $(-)$ nach der i-ten Teilung entspricht der Festlegung $x_i = 0$.
  - $V_6 = \{[\bar{x}_{i,j,k,i} \to 1_{j,k}]_2^- \mid 1 \leq i \leq m,\, 1 \leq j \leq l,\, 1 \leq k \leq 3\}$
    Analog zu V_5 wird $\neg x_i$ durch 1 ersetzt.

- **Binärcodierung**: Im Anschluss an die n Teilungen müssen die Multimengen binär codiert werden, um an die Quanten-Ebene weitergegeben werden zu können. Für die Codierung einer einzelnen Variable x_i, $1 \leq i \leq m$ oder der Konstanten 0 benötigen wir genau $s_{var} = \lceil \log_2(m+1) \rceil$ Qubits. Ein zusätzliches Qubit ist notwendig, um anzugeben, ob die Variable oder Konstante negiert ist, womit sich für die Codierung eines Literals oder der Konstanten 0 bzw. 1 genau $s_{lit} = s_{var} + 1$ Qubits ergeben. Für die Codierung einer Klausel benötigen wir folglich $s_{kla} = 3 * s_{lit}$ Qubits. Damit ergibt sich folgende Codierungsfunktion, wobei $b_i \in \{0,1\}^{s_{var}}$ die Binärdarstellung der Zahl i (gegebenenfalls mit führenden Nullen) und $\cdot$ die Konkatenation zweier Wörter über $\{0,1\}$ sei:

$$\begin{aligned} 0 &\mapsto 0 \cdot 0^{s_{var}} \\ 1 &\mapsto 1 \cdot 0^{s_{var}} \\ x_i &\mapsto 0 \cdot b_i \\ \bar{x}_i &\mapsto 1 \cdot b_i \end{aligned}$$

  Über den Index j und k der Zeichen $x_{i,j,k,t}$, $\bar{x}_{i,j,k,t}$, $0_{j,k}$ und $1_{j,k}$ lässt sich außerdem die genaue Position der Codierung jedes Literals und jeder Konstanten bestimmen. Wir verwenden dazu eine Funktion $\mathtt{p} : \mathbb{N}^2 \to \mathbb{N}$, die wir definieren als

$$\mathtt{p}(j,k) = s_{kla} * (j-1) + s_{lit} * (k-1),\ j,k \in \mathbb{N}.$$

  Diese Funktion liefert zur Position eines Literals oder einer Konstanten innerhalb der Formel ϕ die zugehörige Startposition innerhalb des binären Wortes, das der Quanten-Ebene übergeben wird.

  Die Positionsbestimmung und Codierung geschieht durch die Regelmengen $B_1, \ldots, B_4$. Dabei benötigen alle diese Regeln die Polarisation 0. Diese wird erst durch eine der Regeln aus B_5 erzeugt, wenn die Teilung und Variablenersetzung abgeschlossen ist und die Binärcodierung beginnen kann. Damit nicht bereits vor der ersten Teilung codiert wird, obwohl auch dann die Polarisation 0 ist, fordern wir für den Teilungsindex $t \geq 1$. Im Folgenden bezeichne $b_i(j) = \lfloor i \div 2^j \rfloor \mod 2$ für $i,j \in \mathbb{N}$ das j-te Zeichen der Binärdarstellung von i.

  - $B_1 = \{[x_{i,j,k,t} \to I_{\mathtt{p}(j,k)+s_{var},0} I_{\mathtt{p}(j,k)+s_{var}-1,b_i(s_{var}-1)} \cdots I_{\mathtt{p}(j,k),b_i(0)}]_2^0 \mid 1 \leq i \leq m,\ 1 \leq j \leq l,\ 1 \leq k \leq 3,\ 1 \leq t \leq m\}$
    Jedes Vorkommen von $x_{i,j,k,t}$ wird als $0 \cdot b_i$ codiert. Diese Codierung erfolgt an den Positionen $\mathtt{p}(j,k)+s_{var}, \ldots, \mathtt{p}(j,k)$ und benötigt genau s_{lit} Qubits.
  - $B_2 = \{[\bar{x}_{i,j,k,t} \to I_{\mathtt{p}(j,k)+s_{var},1} I_{\mathtt{p}(j,k)+s_{var}-1,b_i(s_{var}-1)} \cdots I_{\mathtt{p}(j,k),b_i(0)}]_2^0 \mid 1 \leq i \leq m,\ 1 \leq j \leq l,\ 1 \leq k \leq 3,\ 1 \leq t \leq m\}$
    Jedes Vorkommen von $\bar{x}_{i,j,k,t}$ wird analog zu Regel B_1 als $1 \cdot b_i$ codiert.

- $B_3 = \{[0_{j,k} \to I_{\mathtt{p}(j,k)+s_{var},0} \dots I_{\mathtt{p}(j,k),0}]^0_2 \mid 1 \leq j \leq l,\ 1 \leq k \leq 3\}$
  Jedes Vorkommen von $0_{j,k}$ wird als $0 \cdot 0^{s_{var}} = 0^{s_{lit}}$ codiert und belegt die Qubits $\mathtt{p}(j,k) + s_{var}, \dots, \mathtt{p}(j,k)$.
- $B_4 = \{[1_{j,k} \to I_{\mathtt{p}(j,k)+s_{var},1} I_{\mathtt{p}(j,k)+s_{var}-1,0} \dots I_{\mathtt{p}(j,k),0}]^0_2 \mid 1 \leq j \leq l,\ 1 \leq k \leq 3\}$
  Jedes Vorkommen von $1_{j,k}$ wird analog zu Regel B_3 als $1 \cdot 0^{s_{var}}$ codiert.
- $B_5 = \{[\forall_0]^e_2 \to []^0_2 o \mid e \in \{+,-\}\}$
  Damit die binäre Codierung beginnen kann, wird die Polarisation jeder Membran mit Label 2 auf 0 gesetzt, sobald alle Teilungen abgeschlossen sind.
- $b_6 = []^0_2 o \to [T]^0_2$
  Einen Zeitschritt nach Anwendung von B_5 wurde die Formel durch Anwendung der Regeln aus $B_1, \dots, B_4$ binär codiert, daher kann der Trigger T nun die Quantenberechnung starten.

- **Auswertung**: Die Ergebnisse der Quanten-Ebene müssen verarbeitet werden, um schließlich `Ja` oder `Nein` an die Umgebung abzugeben.

  - $A_1 = \{[E]^0_2 \to []^0_2 \mathrm{E} \mid E \in \{\mathtt{Ja}, \mathtt{Nein}\}\}$
    Die Ergebnisse der abgeschlossenen Quantenberechnungen werden aus Membran 2 in Membran 1 geschickt.
  - $a_2 = [\mathtt{Nein}]^0_1 \to []^-_1 \mathtt{Nein}$
    Wurde durch eine der Regeln aus A_1 mindestens eine Instanz von `Nein` in Membran 1 geschickt, wird diese an die Umgebung abgegeben. Durch Änderung der Polarisation von Membran 1 sind die Regeln a_2, a_3 und a_4 anschließend nicht mehr anwendbar. Das System befindet sich damit in einer Haltekonfiguration, die Berechnung ist beendet.
  - $a_3 = [\mathtt{Ja} \to \mathtt{Ja}']^0_1$
    Zeitgleich zur Regel a_2 wird Regel a_3 angewandt, die `Ja` in $\mathtt{Ja}'$ umwandelt. Diese Regel verzögert die Ausgabe von `Ja` um einen Zeitschritt, sodass es nur dann, wenn keine Instanz von `Nein` erzeugt wurde, tatsächlich ausgegeben wird.
  - $a_4 = [\mathtt{Ja}']^0_1 \to []^+_1 \mathtt{Ja}$
    Wenn nach der Erzeugung von $\mathtt{Ja}'$ die Polarisation immer noch 0 ist, hatte offensichtlich keine der Quantenberechnungen das Ergebnis `Nein`. Daher kann nun `Ja` in die Umgebung abgegeben werden und das System erreicht eine Haltekonfiguration.

Die Regelmenge R des P-Systems ergibt sich damit zu

$$R = T_1 \cup A_1 \cup \{a_2, a_3, a_4, b_6\} \cup \bigcup_{1 \leq i \leq 6} V_i \cup \bigcup_{1 \leq i \leq 5} B_i.$$

Quanten-Ebene

Aus dem Zustand des Registers $|z\rangle = |z_{K-1}, \ldots, z_0\rangle$ nach der Initialisierung lässt sich leicht die zu untersuchende Formel rekonstruieren. Wir erläutern diesen Vorgang am Beispiel der folgenden Konfiguration:

$$l = 2,\ m = 4$$
$$|z\rangle = |000010101011100000110100\rangle$$

Es folgt sofort $s_{var} = \lceil \log_2(m+1) \rceil = \lceil \log_2(5) \rceil = 3$ und $s_{lit} = 4$, $s_{kla} = 3 * s_{lit} = 12$. Daher lässt sich das Binärwort, dem der Zustand von $|z\rangle$ entspricht, in zwei Klauseln unterteilen, die jeweils mit 12 Bit codiert sind:

$$0\underbrace{000}_{0}\ \underbrace{1}_{\neg}\underbrace{010}_{x_2}\ \underbrace{1}_{\neg}\underbrace{011}_{x_3} \qquad \underbrace{1}_{\neg}\underbrace{000}_{0}\ 0\underbrace{011}_{x_3}\ 0\underbrace{100}_{x_4}$$

Damit ergibt sich die gesamte Formel zu

$$\phi' = (0 \vee \neg x_2 \vee \neg x_3) \wedge (1 \vee x_3 \vee x_4).$$

Aufgrund der durch die Regelmengen $B_1, \ldots, B_4$ festgelegten Struktur der gewählten Binärcodierung müssten Reihenfolge der Klauseln und der Literale innerhalb dieser Klauseln noch vertauscht werden, wegen der Kommutativität von $\vee$ und $\wedge$ ist diese Umformung aber nicht notwendig. Aus der Tatsache, dass die ursprüngliche Formel ψ geschlossen ist und alle allquantifizierten Variablen durch die Membran-Ebene bereits mit 0 oder 1 belegt wurden, folgt sofort, dass x_2, x_3 und x_4 in ψ existenziell quantifiziert sind. Weil aber die Formeln ϕ' und $\exists x_2.\exists x_3.\exists x_4.\phi'$ erfüllbarkeitsäquivalent sind, ist es ausreichend, ϕ' auf Erfüllbarkeit zu untersuchen.

Aufgabe der Quanten-Ebene ist es also, die Erfüllbarkeit einer durch ein Binärwort gegebenen Formel in 3-KNF mit genau $m - n$ Variablen zu prüfen.

Diese Aufgabe kann, wie in Abschnitt 4.4 beschrieben, mit dem Grover-Algorithmus gelöst werden, indem geprüft wird, ob es in einer unsortierten Datenbank mindestens einen Eintrag gibt, der ein gegebenes Suchkriterium C erfüllt. Wir betrachten dazu als Einträge dieser unsortierten Datenbank alle möglichen Variablenbelegungen $(\alpha_{m-n-1}, \ldots, \alpha_0) \in \{0,1\}^{m-n}$ für die Variablen $x_{n+1}, \ldots, x_m$, formulieren das Suchkriterium C über die charakteristische Funktion

$$f_C((\alpha_{m-n-1}, \ldots, \alpha_0)) = \begin{cases} 1 & \text{wenn } \phi'_{x_{n+1}=\alpha_0, \ldots, x_m=\alpha_{m-n-1}} \equiv 1, \\ 0 & \text{andernfalls} \end{cases}$$

und nutzen Grovers Algorithmus, um herauszufinden, ob es mindestens einen Eintrag $\hat{\alpha}$ gibt, für den $f_C(\hat{\alpha}) = 1$ gilt. Dazu benötigen wir einen Quantenschaltkreis, der die unitäre Transformation $U_{f_C} : |x, y\rangle = |x, y \oplus f_C(x)\rangle$ realisiert.

Mittels der universellen Gatter, die in Abschnitt 4.2 vorgestellt wurden, lässt sich zunächst ein Quantenschaltkreis konstruieren, der eine gegebene Formel

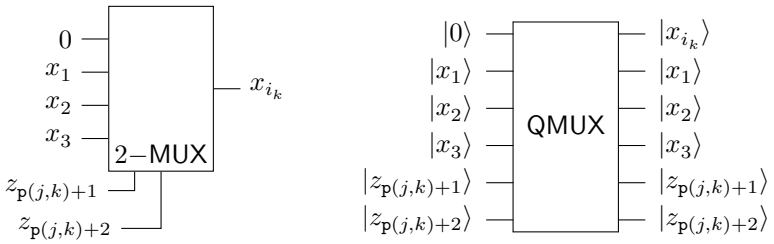

Abbildung 13: Klassischer Multiplexer (links) und Quantenmultiplexer (rechts)

in 3-KNF aus l Klauseln auswertet. Dieser Schaltkreis muss in Abhängigkeit von $|z\rangle$ mit den richtigen Eingangsvariablen und Konstanten beschaltet und in die geforderte Form U_{f_C} gebracht werden. Der resultierende Schaltkreis ist von enormem Umfang, sodass wir auf eine vollständige Angabe verzichten. Dennoch sollen die notwendigen Schritte beispielhaft aufgezeigt werden.

Für die i-te Klausel $C_i = (L_{i,1} \vee L_{i,2} \vee L_{i,3})$, $L_{i,k} \in \{x_{i_k}, \neg x_{i_k}\} \cup \{0, 1\}$ mit $n+1 \leq i_k \leq m$, $1 \leq k \leq 3$, $1 \leq i \leq l$ erhalten wir die Variable an k-ter Stelle x_{i_k} (oder 0, falls an dieser Stelle eine Konstante steht) durch Auswertung von Register $|z\rangle$ mittels einer klassischen Multiplexer-Schaltung, die sich mit den Erkenntnissen aus Abschnitt 4.3 leicht in einen Quantenschaltkreis transformieren lässt. Für $m = 3$ etwa hat der klassische Schaltkreis zur Bestimmung von x_{i_k} die in Abbildung 13 links aufgezeigte Form, während rechts die Struktur eines denkbaren, äquivalenten Quantenschaltkreises dargestellt ist.

Anschließend muss x_{i_k} gegebenenfalls noch negiert werden, um $L_{i,k}$ zu erhalten. Die Information darüber, ob diese Negation durchgeführt werden muss, findet sich im Bit $|z_{\mathtt{p}(j,k)}\rangle$: Im Fall $|z_{\mathtt{p}(j,k)}\rangle = |1\rangle$ muss negiert werden, andernfalls nicht. Um diese kontrollierte Negation durchzuführen, nutzen wir das bereits bekannte CNOT-Gatter (siehe Abbildung 14).

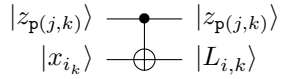

Abbildung 14: CNOT zur Berechnung von $L_{i,k}$

Die Auswertung von C_i lässt sich dann, wie Abbildung 15 zeigt, mittels universeller Fredkin-Gatter unter Zuhilfenahme zweier Hilfsbits leicht realisieren. In gleicher Weise können Fredkin-Gatter eingesetzt werden, um die Konjunktion $C_1 \wedge C_2 \wedge \ldots \wedge C_n \equiv (\ldots (C_1 \wedge C_2) \wedge \ldots \wedge C_n)$ auszuwerten, denn Beschaltung von FG mit $|x\rangle\,|0\rangle\,|y\rangle$ realisiert für zwei boolesche Variable x, y im zweiten Qubit die Verknüpfung $x \wedge y$.

So kann für bekanntes m und l ein Quantenschaltkreis EVAL konstruiert werden, der eine gegebene Formel in 3-KNF evaluiert. Unter Anwendung von CNOT und EVAL† lässt sich daraus – analog zur Konstruktion aus V_f in Abschnitt 5.2 – die unitäre Transformation U_{f_C} realisieren.

Mit diesem Schaltkreis kann nun Grovers Algorithmus in der Variante mit

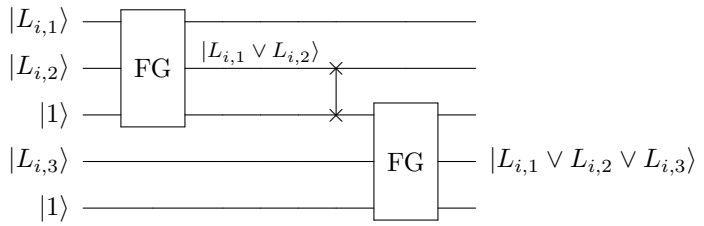

Abbildung 15: Auswertung einer Klausel durch Fredkin-Gatter

unbekannt vielen Lösungen auf das Register $|x\rangle$ angewandt werden. Wir messen anschließend den Zustand der relevanten Bits $|x_{n+1}\rangle, \ldots, |x_m\rangle$ des Registers und erhalten mit hoher Wahrscheinlichkeit p eine erfüllende Variablenbelegung für $x_{n+1}, \ldots, x_m$, wenn ϕ' erfüllbar ist, und eine beliebige Variablenbelegung andernfalls. Das Ergebnis dieser Messung wird anschließend durch erneute Anwendung von U_{f_C} auf $|x, 0\rangle$ überprüft. Gilt $U_{f_C}|x, 0\rangle = |x, 1\rangle$, so ist die gefundene Lösung offensichtlich korrekt, andernfalls gibt es mit Wahrscheinlichkeit p keine Lösung. Der Wert des letzten Bits wird gemessen und in die Membran-Ebene zurückgegeben, wobei 1 als `Ja` codiert wird, 0 entsprechend als `Nein`.

Um die Funktionsweise des Systems β zu verdeutlichen, vollziehen wir nun anhand eines ausführlichen Beispiels mit zahlreichen Illustrationen eine vollständige Berechnung nach.

Beispiel 6.1 Wir nutzen das hybride System β, um die Erfüllbarkeit der Π_2^{P}-QBF-Formel $\psi = \forall x_1.\forall x_2.\exists x_3.\exists x_4.\phi$ mit

$$\phi = (x_1 \vee \neg x_2 \vee x_4) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee \neg x_4) \wedge (\neg x_2 \vee x_3 \vee \neg x_4)$$

zu untersuchen. Zur übersichtlichen Darstellung führen wir eine Funktion `code` ein, die als Parameter eine aussagenlogische Formel $\pi = C_1 \wedge \ldots \wedge C_n$ in 3-KNF und eine natürliche Zahl c übernimmt, und daraus wie folgt eine Multimenge erzeugt:

- Jedes Literal x_i oder $\neg x_i$, das in einer Klausel C_j von π an k-ter Stelle vorkommt, wird auf $x_{i,j,k,c}$ bzw. $\bar{x}_{i,j,k,c}$ abgebildet.
- Jede Konstante 0 oder 1, die in einer Klausel C_j von π an k-ter Stelle vorkommt, wird auf $0_{j,k}$ bzw. $1_{j,k}$ abgebildet.

Damit gilt:

$$\begin{aligned}\texttt{code}(\phi, 0) = {} & x_{1,1,1,0}\bar{x}_{2,1,2,0}x_{4,1,3,0}x_{1,2,1,0}\bar{x}_{2,2,2,0}\bar{x}_{3,2,3,0} \\ & x_{2,3,1,0}x_{3,3,2,0}\bar{x}_{4,3,3,0}\bar{x}_{2,4,1,0}x_{3,4,2,0}\bar{x}_{4,4,3,0}\end{aligned}$$

Die Eingabe, die der Membran-Ebene übergeben wird, entspricht genau der Multimenge $\forall_2\texttt{code}(\phi, 0)$. Daraus ergibt sich die in Abbildung 16 links gezeigte, kompakte Darstellung der initialen Konfiguration C_1.

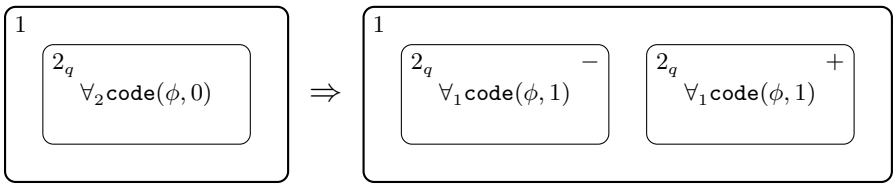

Abbildung 16: Transition von C_1 zu C_2

Im ersten Zeitschritt können nur Regeln aus Regelmenge V_1 sowie eine der Teilungsregeln aus T_1 angewandt werden, wodurch sich anschließend die Konfiguration C_2 in Abbildung 16 ergibt. In dieser Konfiguration können erneut nur Variablenersetzung und Teilung angewandt werden. Durch Anwendung von V_3 und V_5 wird in der Membran mit negativer Polarisation $x_1 = 0$ festgelegt, in der anderen Membran entsprechend $x_1 = 1$. Außerdem wird durch die Regeln aus V_2 der Teilungsindex inkrementiert, bevor erneut durch eine Regel aus T_1 geteilt wird. Es entsteht die in Abbildung 17 gezeigte Konfiguration C_3.

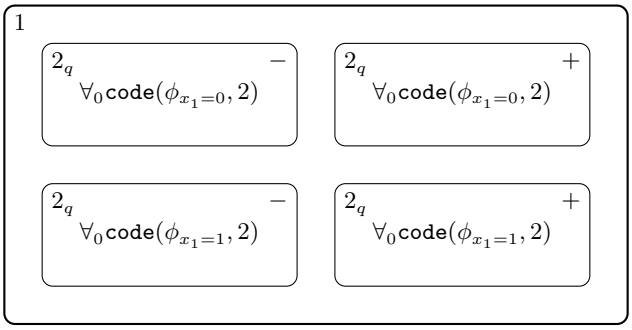

Abbildung 17: Konfiguration C_3

Damit sind die n Teilungen der Membran-Ebene abgeschlossen, die Regeln aus T_1 können nicht mehr angewandt werden. Im folgenden Schritt wird durch $V_3, \ldots, V_6$ in den linken Membranen $x_2 = 0$, in den rechten Membranen $x_2 = 1$ festgelegt. Der Teilungsindex wird nochmals um 1 erhöht, gleichzeitig verlassen alle Objekte $\forall_0$ durch Anwendung von Regeln aus B_5 Membran 2, ändern deren Polarisation zu 0 und gelangen als Objekt o in Membran 1 – es entsteht Konfiguration C_4 (siehe Abbildung 18).

Aufgrund der Polarisation 0 kann nun in keiner der Membranen mehr eine der Regeln V_i mehr angewandt werden, die Variablenersetzung ist abgeschlossen. Im folgenden Schritt wird durch die Regeln aus $B_1, \ldots, B_4$ die Binärdarstellung der Formeln vorgenommen. Zeitgleich gelangen die Objekte o durch Regel b_6 wieder in die Membranen mit Label 2 und werden dort zum Trigger T, der die Quantenberechnung in Gang setzt. Die Multimenge jeder Membran mit Label 2

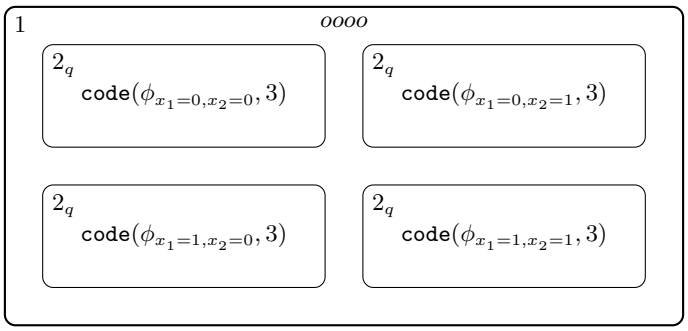

Abbildung 18: Konfiguration C_4

besteht nun aus genau

$$Q_K(2) + 1 = 3 * l * (1 + \lceil \log_2(m+1) \rceil) + 1 = 49$$

Objekten. Wir verzichten aus Gründen der Übersichtlichkeit auf die Angabe all dieser Objekte und betrachten nur beispielhaft die Auswirkung der Anwendung von $B_1, \ldots, B_4$ auf die Objekte der Multimenge $\texttt{code}(\phi_{x_1=0,x_2=1}, 3)$, die zu der neuen Multimenge

$$\begin{aligned}
&\underbrace{I_{47,1}I_{46,1}I_{45,0}I_{44,0}}_{\bar{x}_{4,4,3,3}}\underbrace{I_{43,0}I_{42,0}I_{41,1}I_{40,1}}_{x_{3,4,2,3}}\underbrace{I_{39,0}I_{38,0}I_{37,0}I_{36,0}}_{0_{4,1}} &&\leftarrow \quad (0 \vee x_3 \vee \neg x_4)\\
&\underbrace{I_{35,1}I_{34,1}I_{33,0}I_{32,0}}_{\bar{x}_{4,3,3,3}}\underbrace{I_{31,0}I_{30,0}I_{29,1}I_{28,1}}_{x_{3,3,2,3}}\underbrace{I_{27,1}I_{26,0}I_{25,0}I_{24,0}}_{1_{3,1}} &&\leftarrow \quad (1 \vee x_3 \vee \neg x_4)\\
&\underbrace{I_{23,1}I_{22,0}I_{21,1}I_{20,1}}_{\bar{x}_{3,2,3,3}}\underbrace{I_{19,0}I_{18,0}I_{17,0}I_{16,0}}_{0_{2,2}}\underbrace{I_{15,0}I_{14,0}I_{13,0}I_{12,0}}_{0_{2,1}} &&\leftarrow \quad (0 \vee 0 \vee \neg x_3)\\
&\underbrace{I_{11,0}I_{10,1}I_{9,0}I_{8,0}}_{x_{4,1,3,3}}\underbrace{I_{7,0}I_{6,0}I_{5,0}I_{4,0}}_{0_{1,2}}\underbrace{I_{3,0}I_{2,0}I_{1,0}I_{0,0}}_{0_{1,1}} &&\leftarrow \quad (0 \vee 0 \vee x_4)
\end{aligned}$$

führt (rechts angegeben sind in jeder Zeile die jeweils codierten Klauseln). Jedes Symbol aus $\texttt{code}(\phi_{x_1=0,x_2=1}, 3)$ wird also mit genau $s_{lit} = 4$ Qubits codiert, wobei das in der obigen Darstellung vorderste Qubit angibt, ob eine Negation durchzuführen ist, während die folgenden 3 Qubits den Index der jeweiligen Variablen (oder 000 für eine Konstante) codieren. Die Reihenfolge der so codierten Klauseln und Literale ist dabei, verglichen mit der ursprünglichen Formel, genau umgekehrt. Diese Veränderung ist aber aufgrund der Kommutativität von $\wedge$ und $\vee$ unerheblich.

Im Folgenden bezeichne die Abbildung `bin` die eben beispielhaft aufgezeigte Binärdarstellung einer Formel mittels der Objekte $I_{k,b}$. Damit lässt sich die nach der Codierung entstandene Konfiguration C_5 kompakt darstellen, wie Abbildung 19 zeigt.

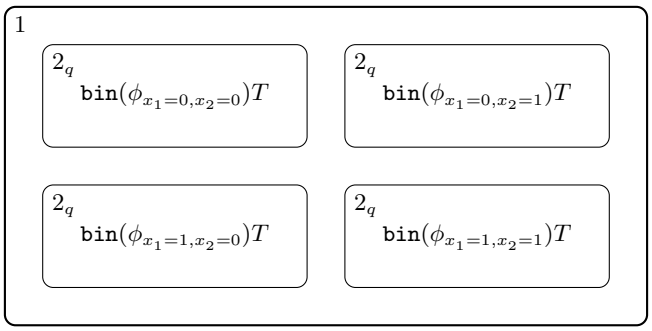

Abbildung 19: Konfiguration C_5

Noch vor Beginn des nächsten Zeitschritts werden die jeweiligen Quantenschaltkreise initialisiert und alle Objekte des Eingabealphabets aus den Membranen entfernt. Außerdem werden alle Quantenmembranen in den Zustand *inaktiv* versetzt, wodurch Π eine (temporäre) Haltekonfiguration C_6 erreicht, die in Abbildung 20 links zu sehen ist.

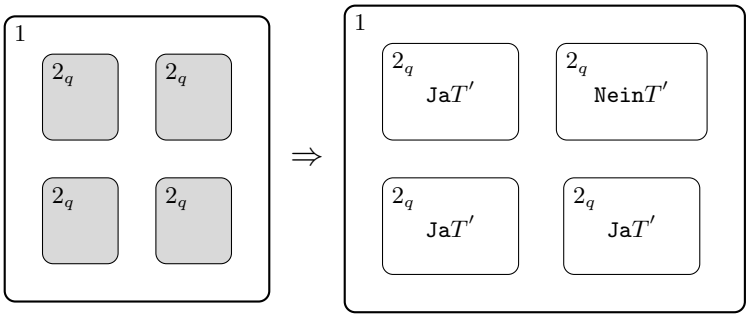

Abbildung 20: Transition von C_6 zu C_7

In den folgenden drei Zeitschritten ruht die Membran-Ebene, während in der Quanten-Ebene die entstandenen Formeln $\phi_0 := \phi_{x_1=0,x_2=0}$, $\phi_1 := \phi_{x_1=1,x_2=0}$, $\phi_2 := \phi_{x_1=0,x_2=1}$ und $\phi_3 := \phi_{x_1=1,x_2=1}$ auf Erfüllbarkeit geprüft werden. Dies geschieht mittels universeller Fredkin-Gatter und unter Anwendung von Grovers Algorithmus. Aufgrund des großen Umfangs des nötigen Schaltkreises verzichten wir an dieser Stelle auf eine explizite Darstellung. Das Ergebnis der Berechnung ist

- für ϕ_0: `Ja`, eine erfüllende Variablenbelegung ist $x_3 = 1,\ x_4 = 0$,
- für ϕ_1: `Ja`, eine erfüllende Variablenbelegung ist $x_3 = 1,\ x_4 = 0$,
- für ϕ_2: `Nein`, es gibt keine erfüllende Variablenbelegung,
- für ϕ_3: `Ja`, eine erfüllende Variablenbelegung ist $x_3 = 1,\ x_4 = 0$.

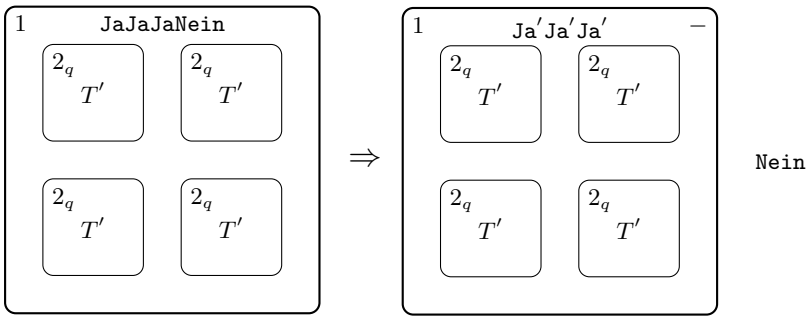

Abbildung 21: Transition von C_8 zur Haltekonfiguration C_9

Daraus ergibt sich für das hybride System nach Abschluss der Quantenrechnung die in Abbildung 20 rechts gezeigte Konfiguration C_7. Unter Anwendung der Regeln aus A_1 erhalten wir von diesem Zustand aus zunächst die in Abbildung 21 links gezeigte Konfiguration, von der aus durch Anwendung von a_2 und a_3 eine Haltekonfiguration (rechts) erreicht wird.

Damit ist die Berechnung abgeschlossen, die Ausgabe des hybriden Systems ist `Nein`.

6.2 $\Sigma_2^{\mathbf{P}}$-QBFSAT

Problembeschreibung

Gegeben seien $m \in \mathbb{N}$ und die quantifizierte aussagenlogische Formel

$$\psi = \exists x_1. \cdots \exists x_n. \forall x_{n+1}. \cdots \forall x_m. \phi,\ 0 \leq n \leq m,$$

wobei ψ geschlossen und ϕ quantorenfrei ist[7]. Die Menge aller Formeln dieser Art bezeichnen wir als Σ_2^{P}-QBF. Eine solche Formel liegt genau dann in Σ_2^{P}-QBFSAT, wenn sie erfüllbar ist.

Hybrides System

Um ψ auf Erfüllbarkeit zu testen, werden wir kein neues hybrides System einführen, sondern das in Abschnitt 6.1 definierte System β wiederverwenden und Ein- sowie Ausgabe geringfügig modifizieren. Dazu formen wir ψ zunächst geeignet um:

$$\begin{aligned}\psi &\equiv \neg(\neg\psi) \equiv \neg(\neg(\exists x_1. \cdots \exists x_n. \forall x_{n+1}. \cdots \forall x_m. \phi) \\ &\equiv \neg(\forall x_1. \cdots \forall x_n. \exists x_{n+1}. \cdots \exists x_m. \neg\phi) =: \neg\psi'\end{aligned}$$

[7] Wir fordern an dieser Stelle nicht, dass ϕ in KNF vorliegt, denn die Umformung von ψ mittels Tseitin-Transformation in eine erfüllbarkeitsäquivalente Formel in KNF ist im Allgemeinen nur mit einem zusätzlichen Quantorenwechsel möglich.

Außerdem nutzen wir folgende Zusammenhänge:

$$\psi \text{ erfüllbar} \overset{(1)}{\Leftrightarrow} \neg\psi' \text{ erfüllbar} \overset{(2)}{\Leftrightarrow} \neg\psi' \text{ allgemeingültig} \overset{(3)}{\Leftrightarrow} \psi' \text{ unerfüllbar}$$

Die Erfüllbarkeitsäquivalenz (1) von ψ und $\neg\psi'$ folgt sofort aus den obigen Umformungen. Weil $\neg\psi'$ eine geschlossene Formel ist, wird sie entweder für jede, oder für keine Variablenbelegung zu 1 evaluiert – Erfüllbarkeit und Allgemeingültigkeit von $\neg\psi'$ sind daher gleichbedeutend (2). Eine Variablenbelegung kann niemals sowohl eine Formel, als auch deren Negation erfüllen; folglich ist die Allgemeingültigkeit von $\neg\psi'$ äquivalent zur Unerfüllbarkeit von ψ' (3).

Die Formel ψ' kann leicht mittels Tseitin-Transformation in KNF gebracht und anschließend von β auf Erfüllbarkeit getestet werden. Ist das Ergebnis der Berechnung `Nein`, so ist ψ' unerfüllbar, woraus mit den obigen Äquivalenzen sofort die Erfüllbarkeit von ψ folgt – die Ausgabe des neuen Systems ist für diesen Fall also `Ja`. Andernfalls ist ψ unerfüllbar, das hybride System gibt `Nein` aus.

7 Schlussbetrachtung

Die abschließende Betrachtung dieser Arbeit soll zweierlei leisten: zum einen soll sie eine Zusammenfassung der Ergebnisse aus den vorhergehenden Kapiteln sein, zum anderen einen Überblick geben, welche offenen Fragen sich daraus ergeben haben und welche Aspekte des Membrane-Quantum Computing weitere Forschung erfordern.

Wir erinnern uns zunächst an das Ziel, das wir in der Einführung dieser Arbeit formuliert hatten: die sinnvolle Konstruktion eines hybriden Systems aus Membran- und Quantenrechner, das inhärente Nachteile beider Berechnungsmodelle kompensiert, ohne sie in ihrer jeweiligen Funktionalität einzuschränken. Um dieses Ziel zu erreichen, haben wir ein Modell eingeführt, das auf zwei verschiedenen Ebenen, einer Membran- und einer Quanten-Ebene, Berechnungen durchführt. Als Grundgerüst unseres Modells haben wir ein P-System gewählt und dieses um spezielle Quantenmembranen ergänzt, innerhalb derer der Wechsel zwischen den genannten Ebenen vollzogen werden kann. Mit diesem Ansatz bleibt einerseits in der Membran-Ebene die komplette Ausdrucksstärke und Zeiteffizienz des P-Systems erhalten – so können etwa auch im hybriden System durch den Mechanismus der Membranteilung innerhalb von n Schritten 2^n Membranen erzeugt werden –, andererseits kann dessen oftmals exponentieller Platzbedarf begrenzt werden durch einen Wechsel in die Quanten-Ebene, in der geeignete Quantenschaltkreise die Berechnung fortführen. Wesentliche Aspekte in der Konstruktion des hybriden Systems waren die Gewährleistung der Synchronität beider Ebenen und die Bereitstellung eines Mechanismus zur Kommunikation. Ersteres haben wir durch Spezifikation einer Zeitfunktion t erreicht, die es ermöglicht, die Dauer einer Quantenberechnung in Zeitschritten des hybriden Systems auszudrücken. Die Kommunikation beider Ebenen miteinander haben wir realisiert, indem wir das Objektalphabet des P-Systems um bestimmte Objekte wie den Trigger T und das Signal T' erweitert und im Quantenschaltkreis ein spezielles Steuerregister $|z\rangle$ bereitgestellt haben.

Unsere theoretischen Überlegungen haben wir in Kapitel 6 schließlich angewandt und ein hybrides System angegeben, das mit Π_2^{P}-QBFSAT ein praxisrelevantes Problem lösen kann. Indem wir eine geeignete Darstellung für die Konfigurationen hybrider Systeme eingeführt haben, konnten wir eine beispielhafte Berechnung schrittweise nachvollziehen und uns davon überzeugen, dass in unserem Ansatz tatsächlich die gewünschten Eigenschaften beider Modelle erhalten bleiben. Das Schema der gezeigten Anwendung – die Erzeugung möglicher Lösungskandidaten in der Membran-Ebene und deren anschließende Validierung in der Quanten-Ebene – kann ohne große Mühe auf weitere Probleme übertragen werden, wie wir an den Beispielen *Circuit Minimization* und Σ_2^{P}-QBFSAT gesehen haben.

Im Rahmen dieser Arbeit haben wir manche Probleme bei der Definition hybrider Systeme nur im Ansatz diskutiert, auch sind wir einige formale Definitionen schuldig geblieben. Dies soll uns Anlass sein, zum Abschluss wie angekündigt auf einige offene Fragen und Problemstellungen im Bereich des hy-

briden Membrane-Quantum Computing einzugehen.

An erster Stelle sei erwähnt, dass wir bei der Einführung hybrider Systeme darauf verzichtet haben, die Begriffe *Konfiguration*, *Transition* und *Berechnung* formal zu definieren; als Begründung hierfür ist anzuführen, dass ihre vollständige Definition von großem Umfang gewesen wäre und kaum zu einem tiefer gehenden Verständnis hybrider Systeme beigetragen hätte. Allerdings haben wir in Kapitel 5 ausführlich dargelegt, welche Besonderheiten sich aus der Kombination von Membran- und Quantenrechnern ergeben und somit die Grundlage für eine formal korrekte Einführung der genannten Begriffe geschaffen.

Auch einer exakten Bestimmung der Zeitfunktion t, die von entscheidender Bedeutung war, um die Synchronität des hybriden Systems zu gewährleisten, haben wir uns in dieser Arbeit nicht gewidmet, und sie stattdessen vereinfachend als konstante Funktion angenommen. Wir haben jedoch eine Möglichkeit aufgezeigt, die Korrektheit der Berechnung eines hybriden Systems durch Herbeiführung temporärer Haltekonfigurationen der Membran-Ebene auch ohne Kenntnis von t sicherzustellen. Um korrekte Ergebnisse bei uneingeschränkt paralleler Berechnung beider Ebenen zu gewährleisten, ist eine exakte Bestimmung der Zeitfunktion allerdings unerlässlich. Auf welche Weise sie sinnvoll ermittelt werden kann, ist eine offene Frage.

Wir haben hybride Systeme als einen Kompromiss zwischen Platz- und Zeiteffizienz kennengelernt. Dabei haben wir uns, wie auch schon [RAB$^+$14a] und [RAB$^+$14b], in der Quanten-Ebene auf Variationen des in Abschnitt 4.4 eingeführten Grover-Algorithmus mit Laufzeit $\mathcal{O}(\sqrt{N})$ beschränkt. Es wäre für zukünftige Untersuchungen interessant, nach sinnvollen Anwendungen unter Verwendung anderer – möglicherweise auch bislang unbekannter – Quantenalgorithmen zu forschen. Wie zeiteffizient solche Algorithmen bestenfalls sein können, ist bisher nicht bekannt. So ist etwa bis heute ungeklärt, ob quantenbasierte Algorithmen existieren, mittels derer NP-vollständige Probleme in polynomieller Zeit gelöst werden können.

Motivation für unsere Beschäftigung mit hybriden Systemen war nicht zuletzt, dass sich daraus neue Perspektiven hinsichtlich der Realisierbarkeit beider Modelle ergeben können. Daher sei abschließend erwähnt, dass die praktische Konstruktion hybrider Systeme eine große Herausforderung darstellt, die hier nicht thematisiert wurde und weiterer Forschungsarbeit bedarf – ebenso wie die getrennte Realisierung von Membran- und Quantenrechnern. Als notwendige Grundlage für eine solche Realisierung letztgenannter Berechnungsmodelle ist deren Theorie bereits in Werken wie [Pău02] bzw. [Wil08] umfassend behandelt worden. Mit der vorliegenden Arbeit ist nun auch für die praktische Umsetzung hybrider Systeme ein theoretisches Fundament gelegt.

8 Literatur

[Amb04] Ambainis, Andris: Quantum Search Algorithms. In: *SIGACT News* 35 (2004), Nr. 2, S. 22–35

[CP01] Calude, Cristian S. ; Păun, Gheorghe: *Computing with Cells and Atoms: An Introduction to Quantum, DNA and Membrane Computing*. Bristol, PA, USA : Taylor & Francis/Hemisphere, 2001

[Gro96] Grover, Lov K.: A Fast Quantum Mechanical Algorithm for Database Search. In: *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*. New York, NY, USA : ACM, 1996, S. 212–219

[Hom13] Homeister, Matthias: *Quantum Computing verstehen: Grundlagen - Anwendungen - Perspektiven*. 3. Aufl. Wiesbaden : Springer, 2013

[Lep07] Leporati, Alberto: Quantum (UREM) P Systems: Background, Definition and Computational Power. In: *Proceedings of the 8th International Workshop on Membrane Computing*. Thessaloniki, Griechenland : Springer, 2007, S. 32–53

[Pău02] Păun, Gheorghe: *Membrane Computing: An Introduction*. Secaucus, NJ, USA : Springer, 2002

[RAB+14a] Rogozhin, Yurii ; Alhazov, Artiom ; Burtseva, Lyudmila ; Cojocaru, Svetlana ; Colesnicov, Alexandru ; Ludmila Malahov : Solving Problems in Various Domains by Hybrid Models of High Performance Computations. In: *The Computer Science Journal of Moldova* 22 (2014), Nr. 1, S. 3–20

[RAB+14b] Rogozhin, Yurii ; Alhazov, Artiom ; Burtseva, Lyudmila ; Cojocaru, Svetlana ; Colesnicov, Alexandru ; Ludmila Malahov : P System Computational Model as Framework for Hybrid (Membrane-Quantum) Computations. In: *Proceedings of the 15th International Conference on Membrane Computing*. Opava, Tschechien : Springer, 2014, S. 373–384

[SU02] Schaefer, Marcus ; Umans, Christopher: Completeness in the Polynomial-Time Hierarchy: A Compendium. In: *SIGACT News* 33 (2002), Nr. 3, S. 32–49

[Wil08] Williams, Colin P.: *Explorations in Quantum Computing*. 2. Aufl. London, UK : Springer, 2008