Grundlagen des Quantencomputing

Wolfgang Mauerer

Inhaltsverzeichnis

Küchenkolloquium 2003

1 Quantenmechanik „in a nutshell“

Die Quantenmechanik wurde Anfang des 19. Jahrhunderts als Antwort auf die drängenden Probleme entwickelt, die sich bei der Untersuchung mikroskopischer Objekte – historisch vor allem Atomen und Molekülen – ergaben und die durch klassiche Theorien nicht erklärt werden konnten. Sie ist mit ihren zahlreichen Verfeinerungen und Spezialisierungen – Quantenfeldtheorie, Quantenlelektrodynamik, Quantenchromodynamik (Dynamik der Quarks und Gluonen) – neben der Relativitätstheorie die dominierende Theorie der modernen Physik.

Viele Vorhersagen, die mit Hilfe der Quantenmechanik gemacht werden können, widersprechen der menschlichen Intuition auf heftigste; dennoch ist es mittlerweile möglich, die allermeisten Aussagen experimentell zu bestätigen. Auch die praktische Anwendbarkeit der Quantenmechanik ist mittlerweile in beinahe allen Bereiche fortschrittlicher technischer Produktion – allen voran die Halbleiterindustrie – zu finden.

Die Funktionsweise moderner Chips kann zwar nurmehr mit Hilfe der Quantenmechanik verstanden werden; dennoch wird die Quantentheorie nur verwendet, um klassische Ergebnisse auf sehr kleinem Maßstab reproduzieren zu können: Prinzpiell könnte jeder heute verwendete Prozessor durch einen mechanischen Nachbau ersetzt werden – natürlich abzüglich der praktischen Probleme, die dabei entstehen würden.

Erst in den letzten 15 Jahren wurden Überlegungen angestellt, wie Quanteneigenschaften zur Durchführung von Berechungen *an sich* – und nicht nur zur Simulation eines klassischen Ergebnisses – verwendet werden können. Die dazu notwendige Theorie war beinahe seit vielen Jahrzehnten vorhanden und mußt nur unter einem anderen Blickwinkel betrachtet werden.

Wir wollen in diesem Abschnitt auf einige Grundlagen der Theorie eingehen, wobei die Darstellung notwendigerweise oberflächlich und ungenau ist. Eine tiefergreifende Einführung findet sich beispielsweise in [11] oder einer Vielzahl anderer, mehr oder weniger guter Lehrbücher.

1.1 Materiewellen

Grundlegendes Postulat der Quantenmechanik ist, daß ein Teilchen nicht wie in der klassischen Vorstellung durch seinen Ort und Impuls beschrieben wird, sondern mit Hilfe einer *Wahrscheinlichkeitsdichte* ϕ repräsentiert werden muß, die man auch als *Wellenfunktion* bezeichnet. Sämtliche Größen der klassischen Physik können durch geeignete Messungen aus der Wellenfunktion bestimmt werden, *verändern* diese aber dabei! Mit anderen Worten: Ein quantenmechanisches System kann durch Messungen nicht vollständig charakterisiert werden, da man nach der Messung unter Umständen ein anderes System vorliegen hat als vorher.

Ein quantenmechanisches Teilchen wird durch seine Wellenfunktion ϕ *vollständig* beschrieben. Es gibt keine weiteren Parameter, die zu seiner Beschreibung notwendig sind.

Dies führt zu einer Vielzahl von Konsequenzen, auf die hier nicht weiter eingegangen werden soll. Die bekannteste davon ist die Heisenbergsche Unschärferelation, die besagt, daß das Produkt der Unsicherheiten bei der Bestimmung von Ort und Impuls eines Teilchens einer unteren Schranke obliegt:[1]

$$\Delta x \Delta p \geq \frac{\hbar}{2}$$

Die Struktur einer Materiewelle ändert sich mit dem Potential, in dem sie sich befindet. Unter einem *Potential* versteht man, wie die Umgebung eines Teilchens auf das Teilchen selbst einwirkt. Dies ist auch aus der klassischen Physik bekannt; beispielsweise „spürt“ ein Elektron ein Potential, wenn es sich in der Nähe eines Atomkerns befindet, von dem es angezogen wird (was sich letztendlich in einer Kraft zwischen beiden manifestiert). Elektronen untereinander erzeugen bekanntlich ein abstoßendes Potential zwischen sich selbst.

Potentiale sind nicht universell, sondern wirken nur selektiv auf Teilchen mit bestimmten Eigenschaften: Ein Photon, das keine Ladung besitzt, wird von einem elektromagnetischen Potential nicht beeinflußt.

[1] Der tatsächliche Zahlenwert ändert sich mit dem verwendeten Maßsystem.

Wellenfunktionen können berechnet werden, indem die vom Potential vorgegebenen Randbedingungen beachtet werden; außerdem postuliert man, daß die Wellenfunktion eines freien Teilchens, das von keinem Potential beeinflußt wird, durch eine ebene Welle gegeben ist:

$$\psi_{\mathrm{f}}(x,t) = Ae^{i(kx-\omega t)}$$

Die gezeigte Wellenfunktion beschreibt ein Teilchen, das sich mit fortlaufender Zeit in Richtung der positiven x-Achse bewegt. k ist ein Parameter, der die Wellenlänge regelt, während über *omega* die Frequenz angepasst werden kann. Beide Größen ermöglichen eine Anpassung der Materiewelle an die physikalischen Eigenschaften des betrachteten Objekts.

1.2 Darstellung von Zuständen und Superpositionsprinzip

Ein weiteres fundamentales Merkmal der Quantenmechanik ist die Fähigkeit, Wellenfunktionen zu überlagern: Wenn zwei Wellenfunktionen miteinander kombiniert werden, ergibt sich daraus wieder eine physikalische erlaubte Wellenfunktion, die den gleichen Anforderungen wie jede andere Wellenfunktion genügt: $\psi = \psi_1 + \psi_2$.

Allerdings ist nicht die Synthese, sondern die Analyse einer Wellenfunktion besonders interessant: Durch eine Fourier-Transformation, die aus der Signaltheorie wohlbekannt ist, kann eine Wellenfunktion beispielsweise so umgeschrieben werden, daß k anstelle von x der freie Parameter ist (wir beschränken uns der Einfachheit halber auf den Zeitpunkt $t = 0$:

$$\begin{aligned}\phi(k) =& \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \psi(x) e^{-ikx} dx \\ \psi(x) =& \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \phi(k) e^{ikx} dk\end{aligned}$$

Die Formeln können so interpretiert werden, daß die Wellenfunktion $\phi(x)$ eine Superposition unendlich vieler Wellenfunktionen $\psi(k$ ist.

Verwendet man den bekannten physikalischen Zusammenhang zwischen Impuls p und Wellenvektor k, nämlich

$$p = \hbar k$$

kann man die Wellenfunktion auch als Funktion des Impulses anstelle des Orts ausdrücken. Beide Wellenfunktionen werden zwar unterschiedlich notiert, drücken aber den selben physikalischen Sachverhalt aus! Allgemein bedeutet dies, daß eine Wellenfunktion in einer beliebigen Basis dargestellt werden kann, ohne eine Änderung in ihrer physikalischen Aussage zu erfahren.

Strukturell ist dies dadurch bedingt, daß eine Fouriertransformation – die wir zur Umschreibung zwischen Orts- und Impulsdarstellung verwendet haben – nichts anderes als eine Basistransformation in einem Hilbertraum ist.

1.3 Die Schrödinger-Gleichung

Die Zeitpropagation einer Wellenfunktion wird durch die Schrödinger-Gleichung geregelt, die nicht abgeleitet oder bewiesen werden kann, sondern als weiteres Postulat der Quantenmechanik gesehen werden muß (natürlich decken sich die von ihr vorhergesagten Dinge mit allen (nichtrelativistischen) experimentellen Ergebnissen, ansonsten wäre sie eine schlechte Wahl):

$$i\hbar \frac{\partial}{\partial t} \psi(\vec{x},t) = -\frac{\hbar^2}{2m} \Delta^2 \psi(\vec{x},t) + V(\vec{x},t)\psi(\vec{x},t)$$

Durch Lösung dieser partiellen Differentialgleichung (was in den meisten Fällen leider nur numerisch möglich ist) kann der zukünftige Verhalten einer Materiewelle vollständig vorhergesagt werden.[2]

Eine etwas eingeschränktere Variante der Schrödinger-Gleichung ensteht durch Separation der Zeitabhängigkeit der Wellenfunktion. Man bezeichnet sie als *zeitunabhängige* Schrödinger-Gleichung:

$$\hat{H}\psi(x) = E\psi(x)$$

Dabei ist $\hat{H}$ der *Hamilton*-Operator des Systems, auf dessen Herleitung wir hier nicht weiter eingehen wollen.[3]

1.4 Wahrscheinlichkeitsinterpretation

Die (im allgemeinen komplexe) Wellenfunktion ϕ kann nicht direkt gemessen werden, weshalb sie *keinerlei* physikalische Aussagekraft besitzt. Erst ihr Betragsquadrat besitzt die physikalische Aussage einer Wahrscheinlichkeit:

$$p(x) = |\psi(x)|^2$$

wobei für $\psi \in \mathbb{C}$ gilt: $|\psi| = \sqrt{\psi^*\psi}$ (Achtung: p steht hier für eine Wahrscheinlichkeit und nicht für einen Impuls).

Da die Wahrscheinlichkeit, das Teilchen *irgendwo* anzutreffen, gleich 1 sein muß, muß folgende Normierungsbedingung erfüllt sein:

$$1 = \int_{-\infty}^{\infty} p(x)dx = \int_{infty}^{\infty} \psi^*(x)\psi(x)dx \tag{1}$$

Interessanterweise ist es prinzipiell unmöglich, die oben gezeigte Wellenfunktion für ein freies Teilchen so zu normieren, daß Forderung 1 erfüllt ist, da ein Integral über eine komplexe Exponentialfunktion auf

[2] Da die Quantenmechanik eine reversible Theorie ist, ist die umgekehrte Richtung – Rekonstruktion der Vergangenheit des Teilchens – ebenso möglich.

[3] Alle Bewegungen der klassischen Mechanik können über die sogenannte *Laplace*-Funktion beschrieben werden, die sich aus $L = T + U$ zusammensetzt. T ist dabei die kinetische Energie des Teilchens, während U das Potential wiedergibt, in dem es sich befindet. Die Funktion wird als $L = L(q_i, \dot{q}_i, t)$ mit $i \in \mathbb{N}$ parametrisiert, wobei q_i generalisierte Koordinaten sind (beispielweise Position, Winkel oder ähnliches). Als *Wirkungsfunktional* eines Systems bezeichnet man das Integral

$$S(\vec{q}) = \int_{t_1}^{t_2} dtL(\vec{q}, \dot{\vec{q}}, t)$$

S hängt dabei von der Bewegung des Teilchens ab, die durch $\vec{q}$ und $\dot{\vec{q}}$ festgelegt ist. Um aus allen möglichen Bewegungen die physikalisch richtige auszuwählen, muß nach der Lagrangeschen Mechanik – die vollständig kompatibel zur Newtonschen Theorie ist – gelten, daß S stationär ist. Die Berechnung der Dynamik wird dadurch zu einem Variationsproblem, bei dem die Euler-Lagrange-Gleichung

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\vec{q}}} = \frac{\partial L}{\partial \vec{q}}$$

gelöst werden muß, da die Bedingung Sextremal dies impliziert.

Eine alternative Beschreibung eines mechanischen Systems ist durch die *Hamilton*-Funktion gegeben, die der Parametrisierung $H = H(q, p, t)$ genügen muß, wobei q wiederum generalisierte Koordinaten und p generalisierte Impulse (gegeben durch $p_j = \frac{\partial L}{\partial \dot{q}_j}$) darstellen. Allgemein kann die Hamilton-Funktion durch eine *Legendre*-Transformation aus der Laplace-Funktion hergeleitet werden; für eine sehr große Klasse physikalischer Systeme gilt aber einfach

$$H = T + U$$

wobei natürlich auf die korrekte (und nicht mit der Lagrange-Funktion identische) Parametrisierung zu achten ist.

Ein quantenmechanisches System wird beschrieben, indem der Hamilton-Operator für das korrespondierende klassische System aufgestellt wird und anschließend alle auftretenden physikalischen Observablen (d.h. Dinge, die gemessen werden können, beispielsweise Ort oder Impuls) durch die assoziierten quantenmechanischen Operatoren ersetzt werden. Man bezeichnet diese Vorgehensweise als *Korrespondenzprinzip*.

Abbildung 1: Wellenfunktionen im unendlich hohen Potentialtopf in einer Dimension

einem nicht-kompakten Träger zwangsläufig divergiert. Physikalisch betrachtet ist dies aber sinnvoll, da ein vollständig freies Teilchen zwar oftmals angenehm zur Durchführung von Rechenoperationen ist, aber in der Realität nicht existiert.

Ein realistischeres Beispiel ist ein Teilchen, das in einem unendlich hohen Kastenpotential eingesperrt ist, was beispielsweise bei einem Elektron in einer passend gewählten Halbleiterstruktur sehr gut erfüllt ist. Das eindimensionale Potential (mit Breite α) ist definiert durch

$$U(x) = \begin{cases} +\infty & \text{für } x > 0 \text{ und } x < L \\ 0 & \text{sonst} \end{cases}$$

wobei m die Masse des Teilchen ist.

Als Bedindung für die Wellenfunktion an den Rändern eines unendlich hohen Potential gilt $\psi_{\mathrm{R}(x)} = 0$, da das Teilchen nicht in die Wand eindringen kann und die Aufenthaltswahrscheinlichkeit außerhalb des Kastens daher 0 ist[4]

Löst man die zeitunabhängige Schrödingergleichung unter Beachtung der Randbedingungen mit dem Ansatz $\psi(x) = \sin\left(\frac{l\pi}{L}x\right)$ mit $l \in \mathbb{N}$, erhält man als Resultat für die E:

$$E = \frac{\hbar^2 l^2 \pi^2}{2mL^2}$$

Die möglichen Energiewerte sind also nicht – wie in klassischen Systemen – kopntinierlich verteilt, sondern dürfen nur diskrete Werte annehmen. Man bezeichnet solche Systeme als *quantisiert*, was den Namen „Quantenmechanik“ verständlicher macht. Nicht nur die Energie eines Systems, sondern auch viele andere Größen – allen voran der Drehimpuls – können in der Quantenmechanik nicht beliebige Werte annehmen, sondern unterliegen einer Quantisierungsvorschrift.

Abbildung 1 zeigt, wie die ersten Zustände der verschiedenen Wellenfunktionen aussehen, die im unendlich hohen 1D-Potentialtopf existieren können.

1.5 Messungen und Operatoren

Wir haben bereits bemerkt, daß Messprozesse in der Quantenmechanik einen wesentlich höheren Stellenwert als in klasssichen Theorien einnehmen, da sie nicht nur eine Eigenschaft des Systems „abfragen“, sondern dieses unausweichlich modifizieren. Strukturell gesehen entspricht ein Messprozess der Anwendung eines *Operators* auf eine Wellenfunktion.

Definition 1 (Linearer Operator) *Unter einem Operator F versteht man eine Vorschrift, die eine Funktion ψ in eine andere Funktion $F\psi$ abbildet. Linearität des Operators ist gegeben, wenn für beliebige $\mu, \lambda \in \mathbb{C}$ gilt:*

$$F(\lambda\psi_1 + \mu\psi_2) = \lambda F\psi_1 + \mu F\psi_2$$

Aus der nicht zeitabhängigen Schrödinger-Gleichung ($H\psi = E\psi$) ist ersichtlich, daß der Hamilton-Operator mit dem Energieoperator identifiziert werden kann.

Der Erwartungswert eines Operators F kann für eine gegebene Wellenfunktion ψ über folgende Integration berechnet werden:

[4] Bei nicht unendlich hohen Wänden fällt die Wahrscheinlichkeitsamplitude innerhalb des Potentials exponentiell ab, weshalb ein Teilchen die Barriere mit einer gewissen Wahrscheinlichkeit passieren kann. Dies ist als *Tunneleffekt* bekannt.

$$\langle F\rangle = \int \psi^* F\psi d^3r$$

Die Darstellung eines Operators ist von der Basis abhängig, in der die Wellenfunktion entwickelt wird. Tabelle 1 zeigt, wie einige bekannte physikalische Größen in Operatorform dargestellt werden können. Achtung: Obwohl die Operatoren in unterschiedlichen Basen unterschiedlich definiert sind, verwendet man immer das gleiche Symbol, was eine hohe intrinsische (potenzielle?[5]) Kraft zur Verwirrung$^{\mathrm{TM}}$ besitzt.[6]

Physikalische Größe		Ortsdarstellung	Impulsdarstellung
Position	$\vec{x}$	$\vec{x}$	$i\hbar\nabla_p$
Impuls	$\vec{p}$	$\frac{\hbar}{i}\nabla$	$\vec{p}$
Drehmoment	$\vec{L}$	$\vec{r}\times\frac{\hbar}{i}\nabla$	$i\hbar\nabla_{\vec{p}}\times\vec{p}$
Kinetische Energie	T	$-\frac{\hbar^2}{2m}\nabla^2$	$\frac{\vec{p}^2}{2m}$
Potentielle Energie	V	$V(\vec{r})$	$V(i\hbar\nabla_{\vec{p}})$

Tabelle 1: Einige Operatoren in Koordinaten- und Impulsdarstellung

Eine wichtige Beziehung zwischen zwei Operatoren ist durch den ihren Kommutator gegeben, der wie folgt definiert wird:

Definition 2 (Kommutator) *Als Kommutator zwischen zwei (linearen) Operatoren F und G bezeichnet man folgende Berechnungsvorschrift:*

$$[F,G] = FH - GF$$

Wenn der Kommutator zweier Operatoren verschwindet ($[F,G] = 0$), sagt man, daß die Operatoren miteinander kommutieren.[7]

Wenn zwei Operatoren miteinander kommutieren, können sie *gleichzeitig* ohne Unschärfe gemessen werden. Wir wollen dies hier aber nicht genauer begründen, da dies eine durchaus umfangreiche Aufgabe ist.[8]

[5] Uh, grausam...

[6] Achtung: Die Tabelle soll nicht den Eindruck vermitteln, daß nur klassisch erfassbare Größen in die Quantenmechanik übertragen werden können. Es gibt durchaus auch Observable, die klassisch nicht vorhanden sind (und aufgrund der Struktur der Theorie es auch nicht sein können), quantenmechanisch aber sehr wohl. Dazu zählt beispielsweise der Eigendrehimpuls (Spin) eines Teilchens.

[7] Anmerkung für Liebhaber mathematischer Strukturen: Die Kommutatorrelation entspricht der Verknüpfung, die zu Bildung von Lie-Algebren benötigt wird. Die Heisenbergsche Unschärferelation, die bereits angesprochen wurde, kann für zwei hermitesche Operatoren verallgemeinert werden:

$$\Delta A\cdot\Delta B \geq \frac{1}{2}\left|\langle[A,B]\rangle\right|$$

Es ist daher möglich, die quantenmechanische Unschärfe als Folge der Eigenschaften von Lie-Algebren zu interpretieren.

[8] Der tiefere Grund liegt in folgendem

Satz 1 *Zwei Observable $\hat{A}$ und $\hat{B}$sind genau dann miteinander vertauschbar, wenn sie ein gemeinsames Orthonormalsystem besitzen.*

den wir allerdings nicht beweisen wollen (entsprechende Beweise finden sich in den gängigen Lehrbüchern der Quantenmechanik). Observable werden mit physikalischen Meßgrößen identifiziert. Nach der Messung einer Observable kollabiert der gemessene Zustand, da nur mehr der Eigenraum des Operators zur Darstellung des Zustands zur Verfügung steht. Wenn ein anderer Operator den selben Eigenraum besitzt, sind alle relevanten Informationen aber noch im Zustand enthalten, die für die Messung benötigt werden, weshalb beide Observablen unabhängig voneinander bestimmt werden können.

1.6 Hilberträume in der Quantenmechanik

Wie wir in Abscnitt 1.2 festgestellt haben, kann eine Wellenfunktion in verschiedenen Basen dargestellt werden, ohne daß ihre physikalischen Eigenschaften dadurch modifiziert werden. Diese Feststellung gab für Dirac den Anlass, eine spezielle Notation für quantenmechanische Zustände zu entwickeln, die eine wesentlich elegantere Formulierung der Theorie erlaubt. Basis dieser Notation ist ein Ket-Zustand, der eine Wellenfunktion unabhängig von ihrer Basis repräsentiert und der symbolisch durch $|x\rangle$ (gesprochen: „Ket x") dargestellt wird. Da praktisch jede quantenmechanische Publikation (und dazu zählen natürlich auch die Arbeiten über Quantencomputing) diese Schreibweise verwendet, sollte man zumindest einigermaßen damit vertraut sein – auch wenn der durch sie erzielte Gewinn auf den ersten Blick etwas gering zu sein scheint.

Jeder quantenmechanische Zustand kann als Element eines Hilbert-Raums $\mathcal{H}$ dargestellt werden.[9]

Definition 3 (Hilbert-Raum) *Der Hilbert-Raum $\mathcal{H}$ ist ein linearer Vektorraum über dem Körper der komplexen Zahlen mit einem hermiteschen Skalarprodukt. Auf dem Hilbert-Raum sind zwei Verknüpfungen definiert:*

- *Mit $|\psi\rangle \in \mathcal{H}$ und $\lambda \in \mathbb{C}$ ist auch $\lambda|\psi\rangle$ ein Element (Zustandsvektor) des Hilbert-Raums.*
- *Mit $|u\rangle, |v\rangle \in \mathcal{H}$ ist auch $|u\rangle + |v\rangle \in \mathcal{H}$.*

Die Definition ist nichts anderes als das bereits bekannte Superpositionsprinzip für quantenmechanische Zustände.

Neben den Ket-Elementen gibt es auch Bra-Elemente[10], die wir $\langle x|$ dargestellt werden. Es handelt sich dabei um die Elemente des dualen Vektorraums zum Hilbert-Raum $\mathcal{H}$. Für unsere Zwecke dient allerdings auch die Sichtweise, daß Bra- und Ketvektor miteinander kombiniert das Skalarprodukt zweier Zustaände ergeben und daher als Abbildung $\mathcal{H} \times \mathcal{H} \rightarrow \mathbb{C}$ interpretiert werden können. Durch diese Defintion kann man sich viel Schreibarbeit sparen, wie man leicht erkennt, wenn man die Defintion des Skalarprodukts in Braket- und Ortraumnotation betrachtet:

$$\langle x|\, y\rangle \hat{=} \int d^3x \psi^*(x)\psi(x)$$

Auch die Notation von Erwartungswerten wird durch diese Schreibweise erleichtert:

$$\langle\psi|\, \hat{A}\, |\psi\rangle \hat{=} \int d^3x \psi^*(x) A(x) \psi(x)$$

Auch die Darstellung von Basen oder die Repräsentation allgemeiner Basistransformationen wird durch diese Notation wesentlich erleichtert, worauf wir hier aber nicht näher eingehen werden.

Operatoren können auch zusammen mit der Bra-Ket-Notation verwendet werden, wobei die weiter oben aufgeführten Eigenschaften weiterhin ihre Gültigkeit behalten.

Analog zur linearen Algebra läßt sich folgendes Eigenwertproblem formulieren:

Definition 4 *Eigenvektoren und Eigenwerte Eine komplexe Zahl α wird als Eigenwert des linearen Operators $\hat{A}$ bezeichnet, wenn bei Anwendung des Operators auf den zugehörigen Eigenzustand gilt:*

$$\hat{A}\,|u\rangle = \alpha\,|u\rangle$$

[9] Etwas hochgestochener kann man sagen, daß jedes Ket-Symbol Mitglied eines abstrakten Hilbertraums in basisfreier Darstellung ist.

[10] Dieser Begriff löst bei englischsprachigen Hörern, die die Pubertätsgrenze nur knapp überwunden haben, üblicherweise Heiterkeitsschübe aus.

Eine sehr wichtige Klasse von Operatoren ist durch alle *hermiteschen* Operatoren gegeben, die quantenmechanischen Observablen entsprechen:[11]

Definition 5 (Hermitesche Operatoren) *Ein Operator $\hat{A}$ ist* hermitesch*, wenn folgende Eigenschaften gelten:*

- *Alle Eigenwerte von $\hat{A}$ sind reell.*
- *Eigenvektoren zu verschiedenen Eigenwerten sind orthogonal.*

Ebenfalls wichtig sind unitäre Operatoren, die folgendermaßen definert sind:

Definition 6 (Unitäre Operatoren) *Ein Operator $\hat{U}$ ist* unitär*, wenn gilt:*

$$\hat{U}\hat{U}^\dagger = \hat{U}^\dagger\hat{U} = 1_{\mathcal{H}}$$

Der adjugierte Operator ist also identisch zum inversen Element des Operators.

Unitäre Operatoren lassen das Skalarprodukt invariant, wenn sie auf zwei Zustände $|psi\rangle$ und $|\phi\rangle$ angewandt werden ($|\psi'\rangle = \hat{U}\,|psi\rangle\,, |\phi'\rangle = \hat{U}\,|\phi\rangle$):

$$\langle\,\psi'|\,\phi'\rangle = \langle\psi|\,\hat{U}^\dagger\hat{U}\,|\psi\rangle\, = \langle\psi|\,1\,|\phi\rangle\, = \langle\,\psi|\,\phi\rangle$$

Operatoren, die zur Implementierung von Quantenalgorithmen benötigt werden, sind im allgemeinen unitär, wie wir weiter unten anhand einiger Beispiele sehen werden.

2 Berechenbarkeits- und Komplexitätstheorie

Ziel der Berechenbarkeitstheorie ist es, den intuitiven Berechenbarkeitsbegriff – d.h. die menschliche Anschauung darüber, was berechnet werden kann oder nicht – in mathematisch präziser Weise zu formalisieren, um Aussagen darüber treffen zu können, ob Funktionen überhaupt mit Hilfe von Computern berechenbar sind oder nicht. Dazu wurden verschiedene Modelle eingeführt, die bezogen auf ihre Berechnungsfähigkeiten zueinander äquivalent sind. Mit ihrer Hilfe kann man die Grenzen der Berechenbarkeit ausloten.

Aufbauend auf diesen formalen Modellen versucht die Komplexitätstheorie festzustellen, *wie schnell* bzw. mit *welchen Ressourcen* ein gegebener Algorithmus arbeitet, um ein gewünschtes Ergebnis zu berechnen. Ein weiteres Ziel ist die Suche nach der Frage, unter welchen Umständen ein Algorithmus terminiert oder nicht.

2.1 Intuitiver Berechenbarkeitsberiff und Churchsche These

Eine Funktion wird als berechenbar angesehen, wenn ein Verfahren angegeben werden kann, das für jede Folge möglicher Eingabewerte einen definierten Satz von Rechenoperationen anwendet, um einen oder mehrere Resultatwerte zurückzuliefern. Wenn die Funktion an manchen Stellen nicht definiert ist (man spricht in diesem Fall von *partiellen Funktionen*), soll der Algorithmus in eine Endlosschleife eintreten. Es ist uninteressant, welches konkrete Gerät verwendet wird – ein simpler Taschenrechner, ein Grossrechner oder eine Horde dressierter Ameisen sind dafür genau gleich gut geeignet, solange der Algorithmus deterministisch wiederholt werden kann.

Der hier vorgestellte Berechenbarkeitsbegriff scheint stark subjektiv zu sein, da er letzendlich auf der (subjektiven) Einschätzung eines Menschen beruht, der entscheiden muß, ob er ein Problem für berechenbar

[11] Für diese Tatsache gibt es keine einleuchtende theoretische Begründung; allerdings beschreibt die These den Ausgang aller bisher bekannter Experimente perfekt, weshalb es keinen physikalischen Grund gibt, von ihr abzuweichen. Sollte jemand eine einleuchtende Begründung für diese Tatsache zusammen mit einer zufriedenstellenden Erklärung des quantenmechanischen Messprozesses finden, ist die Hoffnung auf einen Anruf aus Stockholm nicht unberechtigt.

hält oder nicht. In der Tat ist es in vielen Fällen nicht sicher möglich, zu entscheiden, ob eine Funktion berechnenbar ist oder nicht, da das bisherige Wissen der Menschheit nicht ausreichend ist. Betrachten wir dazu ein Beispiel aus [14]:

$$g(n) = \begin{cases} 1 & \text{falls } n \text{ irgendwo in der Dezimalentwicklung von } \pi \text{ vorkommt.} \\ 0 & \text{sonst} \end{cases}$$

Da das momentane mathematische Wissen über π nicht ausreicht, um diese Frage zu beantworten, ist unklar, ob die Funktion berechenbar ist oder nicht. Fest steht nur, daß sie gemäß ihrer Definition zwischen den beiden Werten 0 und 1 hin- und herspringen wird.

Eine andere Funktion, die auf den ersten Blick ebenfalls nicht berechenbar scheint, ist wie folgt definert:

$$g(n) = \begin{cases} 1 & \text{George W. Bush bombardiert im Jahr 2003 ein Land, dessen Namen er nicht aussprechen kann.} \\ 0 & \text{sonst} \end{cases}$$

Obwohl momentan niemand voraussagen kann, wie die Pläne des genannten Präsidenten sind, ist die Funktion dennoch berechenbar! Warum? Es gibt nur zwei verschiedene Möglichkeiten, die eintreffen können:

- G. W. B. entscheidet sich zur Bombardierung, in diesem Fall liefert die Funktion eine 1 für alle Eingabewerte n zurück.
- Die Vernunft siegt, weshalb die Funktion für alle Eingabewerte n eine 0 zurückliefert.

Es gibt also nur zwei Möglichkeiten, die beide auf eine konstante Funktion hinauslaufen – $f(n) = 0$ oder $f(n) = 1$ –, und beide sind offenbar leicht zu berechnen.

Beide Beispiele verdeutlichen ein weiteres Prinzip, das bei der Definition der formalen Berechenbarkeit zum Einsatz gelangt: Die Beschränkung auf sogenannte *Entscheidungsprobleme*. Wie in den Standardwerken der Literatur (beispielsweise [14], [2] oder [9]) gezeigt wird, stellt dies keine Einschränkung des Kerns des Problems dar, da alle Probleme auf Entscheidungsprobleme zurückgeführt oder durch gleich schwierige Entscheidungsprobleme ersetzt werden können. [12].

Offensichtlich kann eine gewisse Unsicherheit bei der Frage nach der Berechenbarkeit einer Funktion nicht vermieden werden. Dennoch ist es möglich, diese Unsicherheit so weit wie möglich zu reduzieren, indem eine formale Maschine definiert wird, der man die Fähigkeit zur Berechung aller intuitiv berechenbaren Probleme zuspricht. Natürlich ist es prinzipiell unmöglich, die Korrektheit dieser Forderung zu beweisen, was im mathematischen Sinn sehr unbefriedigend ist. Dennoch hat die Erfahrung der letzten Jahrzehnte gezeigt, daß kein Gegenbeispiel gefunden werden konnte, daß die gleich näher formalisierten Forderungen widerlegen würden, was dem genannten „Glaubensartikel" zumindest aus pragmatischer Sicht zu einer starken Rechtfertigung verhilft. Eine ähnliche Situation findet sich auch an vielen Stellen der Physik, man denke beispielsweise nur an die Hauptsätze der Thermodynamik oder das Prinzip der minimalen Wirkung, die prinzipiell nicht bewiesen werden können, durch die Erfahrung der letzten Jahrhunderte aber sehr stark plausibel gemacht werden.

Die Hypothese der formalen Berechenbarkeit ist in der Literatur unter den Namen *Church'sche These* oder *Church-Turing-These* bekannt und existiert in einigen verschiedenen Varianten, die aber allesamt äquivalent zueinander sind:

Definition 7 (Church-Turing) *Alle Funktionen, die im intuitiven Sinne durch ein realistisches physikalisches Gerät berechenbar sind, können durch eine Turing-Maschine berechnet werden.*

[12] Die Einschränkung auf Entscheidungsprobleme besitzt einen weiteren, tieferen Grund: Alle Berechnungsprobleme können auf das Akzeptieren oder Verwerfen von Strings zurückgeführt werden, die Mitglieder (oder auch nicht-Mitglieder) verschiedener Klassen formaler Sprachen sind. Dadurch können Resultate aus dem Bereich der formalen Sprachen in die Theorie der allgemeinen Berechenbarkeit übernommen werden.

Es zeigt sich, daß neben einer Turing-Maschine noch viele andere formale Automaten denkbar sind, die beweisbar die gleiche Mächtigkeit wie eine Turing-Maschine besitzen; einge davon werden im nächsten Abschnitt genauer beschrieben.

Computer (und auch ihre Programmiersprachen) sind übrigens allesamt Turing-vollständig, d.h. besitzen die nötige Leistungsfähigkeit, um eine Turing-Maschine simulieren und damit jede berechenbare Funktion berechnen zu können, die denkbar ist – wenn man einmal davon absieht, daß reale Computer nur endliche Speicherkapazitäten besitzen, was in den formalen Modellen nicht berücksichtigt wird. Jede (vernünftige) Programmiersprache kann verwendet werden, um eine Turing-Maschine zu simulieren. Das nächste Kapitel wird zeigen, daß dies sogar mit sehr reduzierten Sprachen möglich ist, die nur aus einigen elementaren Anweisungen bestehen. Beispielsweise kann eine Turing-Maschine auch in TeX oder PostScript simuliert werden, oder anders ausgedruckt: Laserdrucker sind in der Lage, alle Funktionen zu berechnen, die überhaupt berechnet werden können – was auf den ersten Blick ein durchaus erstaunliches Resultat ist.

2.2 Turing-Berechenbarkeit

Wie ist eine Turing-Maschine aufgebaut? Man benötigt folgende Komponenten:

- Ein (unendlich langes) Band, das in voneinander anbgegrenzte Zellen unterteilt ist.
- Ein Schreib-Lesekopf, der sich über genau einer Bandzelle befindet und das darin enthaltene Symbol lesen oder ein neues Symbol hineinschreiben kann. Desweiteren besitzt der Kopf die Fähigkeit, sich eine Zelle nach links oder rechts zur angrenzenden Bandzelle zu bewegen.
- Eine Kontrolleinheit, in der der aktuelle Maschinenzustand festgehalten wird. Hiermit verknüpft ist eine Sammlung von Übergangsfunktionen, die anhand des aktuellen Maschinenzustands die nächste auszuführende Aktion vorgibt.

Die Maschine gleich einem menschlichen Rechner, dem kariertes Papier und Bleistift/Radiergummi zur Verfügung stehen, um eine Rechnung nach einem vorher festgelegten Plan durchzuführen. Die einzelnen Schritte des Plans dürfen sich dabei nur auf den Inhalt beziehen, der sich momentan auf dem Papier befindet; desweiteren hat der Rechner die Möglichkeit, mit dem Bleistift entweder neue Zeichen auf das Papier zu bringen oder bestehende Zeichen mit Hilfe des Radiergummis zu entfernen. Dennoch stellen diese simplen Mittel alle Möglichkeiten zur Berechnung beliebiger entscheidbarer Funktionen bereit!

Bevor wir die Funktionsweise einer Turing-Maschine anhand einer einfachen Rechnung klarmachen, soll sie noch mathematisch exakt definiert werden:

Definition 8 (Deterministische Turing-Maschine DTM) *Eine Turing-Maschine TM ist ein 6-Tupel $T = (Q, \Sigma, \delta, q_o, \Box, E)$ mit*

- *einer endlichen Menge Q von Zuständen.*
- *einem Alphabet Σ*[13]
- *einem Blanksymbol $\Box \notin \Sigma$, das die Funktion eines Leerzeichens besitzt.*
- *einem Startzustand q_0.*
- *einer endlichen Menge $E \subseteq Q$ von Endzuständen.*
- *einer Transitionsfunktion $\delta : Q \times \Sigma \to Q \times \Sigma \times \{-1, 0, 1\}$*

[13] Einige Definitionen verwenden unterschiedliche Alphabete für Eingabe- und Bandalphabet, d.h. den Alphabeten, die zur Spezifikation der Eingangsdaten und während der Berechnung verwendet werden.

Die Transitionsfunktion führt die Turingmaschine von einem 2-Tupel, das aus aktuellem Zustand und Symbol auf dem Band besteht, in ein neues 2-Tupel über. Das dritte Element der Überführungsfunktion $\{-1, 0, 1\}$ gibt an, ob sich der Kopf dabei nach links (-1), recht (1) oder gar nicht (0) bewegt.

Standardbeispiel zur Funktionsweise einer Turingmaschine ist die Addition von 1 zu einer binär kodierten Zahl: Der Turing-Maschine wird eine Zahl auf dem Band als Eingabewert übergeben, und nach der Abarbeitung der vorher festgelegten Schritte muß sich die um 1 erhöhte Zahl auf dem Band befinden. In diesem Fall genügt es, als Alphabet die Menge $\{0, 1\}$ zu verwenden; desweiteren benötigt man die Zustände z_0, z_1, z_2 und z_e, wobei z_0 der Start- und z_e der Endzustand ist. Die Transitionsfunktion ist folgendermaßen definiert:

$$\begin{aligned}\delta(z_0, 0) &\to (z_0, 0, 1)\\ \delta(z_0, 1) &\to (z_0, 0, 1)\\ \delta(z_0, \square) &\to (z_1, \square, -1)\end{aligned}$$

$$\begin{aligned}\delta(z_1, 0) &\to (z_2, 0, -1)\\ \delta(z_1, 1) &\to (z_1, 0, -1)\\ \delta(z_1, \square) &\to (z_e, \square, 0)\end{aligned}$$

$$\begin{aligned}\delta(z_2, 0) &\to (z_2, 0, -1)\\ \delta(z_2, 1) &\to (z_2, 0, -1)\\ \delta(z_2, \square) &\to (z_e, \square, 1)\end{aligned}$$

Am Anfang (d.h. im Startzustand z_e) findet sich nur die Eingabenummer auf dem Band; die restlichen Bandpositionen (links und rechts von dieser Zahl) sind mit Blank-Symbolen gefüllt. Der erste Schritt der Maschine besteht zunächst darin, sich so lange von links nach rechts zu bewegen, bis das Ende der Zahl erreicht ist, in dem sich das niedrigwertigste Bit befindet: Im Zustand z_0 bewegt sich die Maschine immer nach rechts, unabhängig davon, ob sich eine 0 oder eine 1 auf dem Band befindet. Der Zustand bleibt in jedem Schritt auf z_0.

Erst wenn das Blank-Symbol erreicht ist, bewegt sich der Kopf wieder nach links, außerdem wechselt die Maschine in den Zustand z_1. Der Plan ist nun wie folgt: Solange sich Einsen unter dem Kopf befinden, werden diese durch Nullen ersetzt; außerdem erfolgt eine Bewegung nach links – hin zu den höherwertigen Zahlen –, die den Zustand aber unverändert in z_1 läßt.

Normalerweise besitzt die Zahl mindestens an einer Stelle eine Null, die sich an irgendeiner Position befindet. Trifft der Kopf nach einigen Schritten auf eine Null, wird diese durch eine Eins ersetzt. Anschließend wechselt die Maschine in den Zustand z_2, dessen einzige Aufgabe es ist, den Schreib-Lesekopf wieder an die ursprüngliche Position links von der Zahl zurückzufahren.

Sollte die Zahl nur aus Einsen bestehen, trifft der Kopf (im Zustand z_1) irgendwann wieder auf das erste Blank-Symbol. Die entsprechende Regel für diesen Fall lautet, daß das Blank-Symbol durch eine Eins ersetzt wird und die Maschine in den Endzustand übergeht.

Wenn sich der Kopf im Endzustand befindet, ist daher in jedem Fall sichergestellt, daß sich eine gegenüber der Ausgangszahl um Eins erhöhte Zahl auf dem Band befindet.

2.3 Alternative Modelle

Obwohl das Inkrementieren einer Zahl um Eins im allgemeinen keine sonderlich schwierige Aufgabe ist (sofern die Zahlen noch durch einen Taschenrechner dargestellt werden können...), benötigt die angegebene Turing-Maschine immerhin 9 Transitionsfunktionen. Bei schwierigeren Problemen wird die Zahl der benötigten

Transitionen sehr schnell völlig unübersichtlich, weshalb sich Turing-Maschinen zwar sehr gut für abstrakte berechnungstheoretische Beweise eignen, in der Praxis aber nicht zum Lösen von Problemen eingesetzt werden können.

Es gibt daher einige andere formale Berechnungsmodelle, die in der Praxis wesentlich leichter handhabbar sind, um Algorithmen zu spezifizieren. Es genügt dabei, einmal die Äquivalenz der alternativen Modelle mit einer Turing-Maschine zu zeigen (d.h. Modell B kann eine Turing-Maschine A effizient simulieren und umgekehrt ist B durch eine Turing-Maschine effizient simulierbar), um in Zukunft Algorithmen in dem einfacheren Modell angeben zu können. Da die Modelle äquivalent sind, kann man sich dennoch auf alle mit Hilfe von Turingmaschinen erzielten Resultate beziehen, ohne ein explizites Turing-Maschinen-Programm für das Problem angeben zu müssen.

Beweise für die Äquivalenz zwischen den nachfolgend vorgestellten Modellen und einer Turing-Maschine finden sich an vielen Stellen in der Literatur und werden hier nicht wiedergegeben.

2.3.1 Registermaschinen

Registermaschinen orientieren sich sehr stark an Konzepten, die in realen Computern verwendet werden. In der Literatur existieren unterschiedliche (gleichwertige) Definitionen, wobei das Konzept einer Random-Access-Machine (RAM) sehr verbreitet ist. Eine RAM besteht aus:

- Einen Speicher mit unendlich vielen Speicherzellen, die fortlaufend durchnummeriert werden können. Jede Zelle kann eine ganze Zahl beliebiger Länge aufnehmen.
- Einem Akkumulator, in dem analog zu einer Speicherzelle ein Wert festgehalten werden kann.
- Einem Satz von Befehlen, die auf den Akkumulator wirken: `LOAD` und `STORE` zum Austausch von Werten zwischen Speicher und Accu, `ADD`, `SUB` als arithmetische Operationen. Alle bisher genannten Befehle können sowohl mit direkter und indirekter Adressierung (wie Stackvariablen und Pointer in höheren Sprachen) wie auch mit Konstanten als Operatoren verwendet werden. Außerdem existieren die Anweisungen `GOTO`, `JUMP ZERO` und `END` zur Steuerung des Programmflußes.
- Ein Programm, das aus einer Sammlung von Befehlen besteht.

Eine RAM ist damit vom Aufbau her gesehen beinahe mit einem C64 identisch[14].

Wie folgendes Programm zeigt, ist die Inkrementierung einer gegebenen Zahl (die sich in Register 1 befindet) nun wesentlich einfacher und übersichtlicher durchzuführen:

```
LOAD 1
ADD #1
STORE 1
END
```

Die Zahl wird dazu zuerst aus Register 1 geladen, anschließend um 1 erhöht (die Schreibweise `ADD #1` bedeutet, daß die numerische Konstante 1 zum Wert im Akkumulator hinzugefügt wird) und danach wieder zurückgeschrieben.

2.3.2 WHILE- und GOTO-Berechenbarkeit

Die Programmierung von Algorithmen mit Hilfe von Registermaschinen ist immer noch sehr aufwendig. Einfacher geht es mit zwei Berechnungsformalismen, die sich zwar nur bedingt zur Durchführung von Beweisen eignen, aber den verwendeten Algorithmus sehr klar wiedergeben: WHILE- und GOTO-Berechenbarkeit.

[14] Really. Das Ding hat auch nur einen Akkumulator anstelle eines Registersatzes und nur unbedeutend mehr Maschinenbefehle.

Beide Konstrukte finden sich mehr oder weniger direkt in vielen heute üblichen Sprachen, sofern sie auf der imperativen Linie stammen.

Beide Sprachen gehen von einem gemeinsamen Satz syntaktischer Komponenten aus, hängen ansonsten aber nicht voneinander ab und sind separat gleichmächtig zu einer Turing-Maschine:

- Variablen $x_0, x_1, \ldots$ werden zum Speichern von Integer-Werten beliebiger Länge benutzt, wobei `:=` zur Wertzuweisung benutzt werden kann.
- Konstanten aus den Ziffern $0, 1, \ldots 9$ repräsentieren ganze Zahlen.
- Das Trennsymbol `;` separiert zwei Anweisungen voneinander.
- Addition und Subtraktion wird mit `+` und `-` durchgeführt.

Jede Wertzuweisung $x_i := x_j \pm c$ ist ein syntaktisch korrektes Programm P; ebenso die Konkatenation zweier Programme $P_1; P_2$.

Die Additionsfunktion ist nun mehr oder weniger trivial:

```
x1 := x1 + 1
```

Um auch kompliziertere Dinge (wie beispielsweise die Simulation einer Turing-Maschine erledigen zu können), muß die definierte Sprache erweitert werden. Bei WHILE-Programmen gibt es zusätzlich die syntaktische Regel `WHILE` $x_i \neq 0$ `DO` P `END`. Diese kann beispielsweise verwendet werden, um die Multiplikation $x_0 := x_1 \cdot x_2$ auf der Addition aufbauend zu implementieren:

```
x0 := 0;
WHILE x2 ≠ 0 DO
  x0 := x0 + x1;
  x2 := x2 - 1;
END
```

Es läßt sich zeigen, daß Programme mit diesen Möglichkeiten äquivalent zu Turing-Maschinen sind.

Alternativ kann man die Basisregeln auch um zwei Sprunganweisungen erweitern:

- `GOTO` M_i springt die Programmzeile M_i an.
- `IF` $x_i = c$ `THEN GOTO` M_j springt zu Zeile M_j, wenn die Bedingung $x_i = c$ erfüllt ist, und setzt anderenfalls die Ausführung in der nächsten Zeile fort.

Offensichtlich lassen sich WHILE-Schleifen durch GOTO-Befehle ersetzen:

```
WHILE x_0 ≠ 0 DO P END
...
```

läßt sich ersetzen durch

```
1: IF x_0 = 0 THEN GOTO 4;
2: P;
3: GOTO 1;
4: ...
```

Umgekehrt lassen sich GOTO-Programme durch WHILE-Programme ersetzen, wobei es interessanterweise möglich ist, mit nur einer einizgen WHILE-Schleife auszukommen, siehe beispielsweise [9].

2.3.3 Alternative Turing-Maschinen

Neben der oben eingeführten Turing-Maschine mit einem einzigen Band gibt es noch Modelle von Turing-Maschinen, die mit mehreren Bändern (und damit auch mehreren Schreib-Leseköpfen) arbeiten. Diese Varianten sind allerdings nicht leistungsfähiger als eine Ein-Band-DTM (da sie von einer Ein-Band-DTM simuliert werden können), weshalb sie hier nicht von Bedeutung sind.

Bei der Definition der Turing-Maschine auf Seite 10 wurde der Zusatz *deterministisch* verwendet, da alle Operationen von vorneherein durch die Transitionsfunktionen festgelegt werden und daher deterministisch sind. Eine Erweiterung dieses Konzepts ist die nichtdeterministische Turing-Maschine NTM, die folgendermaßen definiert ist:

Definition 9 (Nichtdeterministische Turing-Maschine NTM) *Eine nichtdeterministische Turing-Maschine ist ein 6-Tupel $T = (Q, \Sigma, \delta, q_o, \Box, E$, wobei $Q, \Sigma, q_o, \Box$ und $E)$ wie bei einer deterministischen Turing-Maschine definiert sind. Die Transitionsfunktion δ ist nun eine Abbildung des Typs*

$$\delta : Q \times \Sigma \to \mathcal{POT}(Q \times \Sigma \times \{-1, 0, 1\})$$

wobei $\mathcal{POT}$ die Potenzmenge darstellt.

Die Übergangsfunktion hat nun für einen Zustand eventuell mehrere verschiedene Möglichkeiten, aus denen sie eine auswählen muß. Da die Auswahl nicht deterministisch erfolgt, kann auch die Turing-Maschine nicht mehr deterministisch sein.

Wozu ist eine derartig aufgebohrte Version gut? Zunächst muß bemerkt werden, daß die Einführung des Indeterminismus verblüffenderweise *nicht* mit sich bringt, daß schwierigere Probleme als mit einer normalen DTM gelöst werden können[15].

Verbindet man die NTM mit einem zusätzlichen Orakel, das an einer Gabelung der Transitionsfunktion intuitiv den richtigen Schritt auswählt, kann man die für eine Berechung benötigte Zeit (und damit ihre Komplexität) drastisch reduzieren. Da sich Orakel seit den alten Griechen allerdings etwas rar gemacht haben und sich heutzutage nur mehr in Börsen, bei Wahlforschern und ähnlichen Institutionen finden (dort allerdings auch nicht allzugut funktionieren), läßt sich dieses Konzept leider nicht praktisch nutzen. Allerdings ist es dennoch möglich, durch einfaches Raten (anstelle eines präzisen Orakels) Algorithmen zu beschleunigen; dabei muß als Nachteil in Kauf genommen werden, daß das richtige Ergebnis nicht mehr garantiert werden kann, sondern nur noch mit einer bestimmten Wahrscheinlichkeit geliefert wird.

2.4 Berechnungskomplexität

Neben der Frage, welche Funktionen überhaupt berechenbar sind, gibt es einen weiteren wichtigen Komplex, der vor allem für die Praxix der Berechnung von Algorithmen große Bedeutung besitzt: Wie lange braucht ein Programm bei einer bestimmten Eingabe, um zu einem Ergebnis zu kommen, und wie verhält es sich, wenn immer längere Eingaben verwendet werden? Allgemein versucht die Komplexitätstheorie, algorithmische Probleme anhand ihres Bedarfs an Berechnungsressourcen zu charakterisieren, im Fall von computerlösbaren Problemen also über ihren Zeit- und Speicherplatzbedarf[16]. Offensichtlich ist es sehr wichtig, die effizientesten Algorithmen zur Lösung von Problemem zu verwenden, um die vorhandenen Möglichkeiten optimal zu nutzen.

Interessant bei der Betrachtung von Komplexitätsproblemen sind untere Schranken, die angeben, wie schnell ein Algoritmus laufen kann. In vielen Fällen exisitieren prinzipielle Grenzen, die ein Problem nicht

[15] Für Liebhaber dreibuchstabiger Abkürzungen: Die Einführung von Indeterminismus in Modelle, die einer Turingmaschine in ihrer Berechnungskraft unterlegen sind, läßt die Menge der mit ihnen lösbaren Probleme nicht immer unverändert: Während indeterministische und deterministische endliche Automaten NEA und DEA zueinander äquivalent ist, kann ein nichtdeterministischer Kellerautomat NKA eine größere Klasse von Funktionen berechnen als sein deterministisches Gegenstück DKA.

[16] Die Speicherplatzkomplexität wird in Anlehung an das Turing-Maschinen-Modell auch häufig als Bandkomplexität bezeichnet.

schneller lösbar machen, als es durch diese Grenze vorgegeben wird. Leider kann aus der Existenz einer unteren Schranke aber noch nicht auf einen Algorithmus geschlossen werden, der die Berechnung auch tatsächlich in dieser Zeit (oder diesem Speicherbedarf) durchführt; reale Methoden arbeiten in vielen Fällen über der unteren Grenze, da ein besseres Verfahren noch nicht gefunden wurde (und vielleicht auch nicht gefunden werden kann, da es nicht existiert).

Die Komplexität eines Algorithmus wird immer an seinem Verhalten im *worst case* gemessen, wenn wirklich – frei nach Murphy – sämtliche Anfangsparameter so gewählt werden, daß die schlechtmöglichste Leistung erreicht wird. Aus diesem Grund sind viele Aussagen der Komplexitätstheorie mit Vorsicht zu geniessen: Ein Algorithmus mit schlechter Worst-Case-Komplexität kann in vielen Fällen besser als ein Algorithmus arbeiten, der eine im schlechtesten Fall niedrigere Komplexitätsstufe besitzt, dafür im durchschnittlichen Fall aber schlechter ist.

2.5 Komplexitätsklassen

Algorithmen, die vergleichbare Komplexität aufweisen, werden in *Komplexitätsklassen* zusammengefasst. Grob gesprochen können Probleme in zwei Kategorien unterteilt werden: Solche, die auf Computern effizient gelöst werden können, und solche, für die dies nicht möglich ist.

Die präzise Definition einer Komplexitätsklasse im Modell der Turing-Maschinen lautet (zitiert nach [14]) folgendermaßen:

Definition 10 (Komplexitätsklasse) *Sei $f : \mathbb{N} \to \mathbb{N}$ eine Funktion. Die Klasse TIME($f(n)$) besteht aus allen Sprachen A, für die es eine deterministische Mehrband-Turingmaschine M gibt mit $A = T(M)$ und $time_M(x) \leq f(|x|)$.*

$\mathrm{time}_M : \Sigma^* \to \mathbb{N}$ bedeutet hierbei die Anzahl der Rechenschritte bei der Eingabe von M, d.h. die Anzahl der Transitionen, die gebraucht werden.

Mehrband-Turingmaschinen werden zur Definition der Komplexität verwendet, da sie äquivalent zu Kostenbetrachtungen bei WHILE-Programmen sind und daher mehr der Sichtweise entsprechen, die man bei der Analyse von Algorithmen in Hochsprachen verwendet (allerdings müssen unterschiedliche Kostenmaße beachtet werden, auf die wir gleich eingehen werden).

Wie kann man die Komplexität eines Algorithmus angeben? Betrachten wir die Multiplikation zweier $N \times N$-Matrizen A und B, die leicht mit einem Programm der folgenden Art erledigt werden kann[17]:

```
temp = 0;
for n = 1 to N do
  for m = 1 to N do
    for p = 0 to N do
      temp = temp + A[n,p]*B[p,m];
    od;
    C[n,m] = temp;
    temp = 0;
  od;
od;
```

Die innerste Schleife (über den Laufindex p) erfordert N Multiplikationen und Additionen. Da die Schleife innerhalb zweier äußeren Schleifen ausgeführt wird, ist die Gesamtzahl der benötigten Operationen insgesamt proportional zu N^3. Um die errechneten Ergebnisse in die Zielmatrix C zu schreiben, sind insgesamt N^2 Operationen erforderlich, weshalb die Komplexität insgesamt proportional zu $N^3 + N^2$ ist.

[17] Die `for`-Schleifen können leicht durch WHILE-Schleifen ersetzt werden, ohne daß sich die Komplexität des Algorithmus ändert. Das Problem ist daher WHILE- und damit auch Turing-berechenbar.

In der Komplexitätstheorie interessiert man sich vor allen für das asymptotische Verhalten von Funktionen für große n, weshalb die $\mathcal{O}$-Notation verwendet wird, die man als Landau- bzw. Bachmann-Symbol bezeichnet [7]:

Definition 11 (Landau-Symbol) *$f(x) = \mathcal{O}(g(x))$ für $x \to \infty$, wenn Kostanten $K \in \mathbb{R}_+$ und $R \in \mathbb{R}$ existieren, so daß $|f(x)| \leq K \cdot |g(x)|$ für alle $x \geq R$.*

Dies bedeutet, daß die Funktion f für ein asymptotisch gegen unendlich laufendes x höchstens von gleicher Größenordnung wie g ist. Das Gleichheitszeichen hat in diesem Fall natürlich nicht die übliche Bedeutung der Mathematik, sondern wird „schlampig" verwendet, z.B. $\mathcal{O}(7 \cdot x^2 + x) = \mathcal{O}(3 \cdot x^2) = \mathcal{O}(n^2)$.

Es zählt daher nur die führende Ordnung der Komplexität; Konstanten und Faktoren werden ebenfalls vernachlässigt. Die Komplexität der Matrixmultiplikation ist daher $\mathcal{O}(n^3)$.

Der gezeigte Algorithmus ist nicht die sinnvollste Art, um zwei Matrizen miteinander zu multiplizieren. Wie [10] zeigt, läßt sich ein Algorithmus angeben, der die Operation in $\mathcal{O}(n^{2.376})$ durchführt. Die Komplexität dritter Potenz ist daher eine *obere Grenze*. Die untere Grenze der Matrixmultiplikation ist für den allgemeinen Fall leicht einsehbar: Da zumindest alle Elemente der Zielmatrix ausgefüllt werden müssen, was n^2 Operationen erfordert, ist es nicht möglich, einen Algorithmus zu finden, dessen Zeitkomplexität geringer als $\mathcal{O}(n^2)$ ist.

Die Multiplikation der skalaren Elemente bei der Matrixmultiplikation wurde in der obigen Betrachtung mit 1 angesetzt und damit als genau so schwierig wie eine Addition bewertet. Dies ist in der Realität natürlich falsch: Da Multiplikationen wesentlich schwieriger als Additionen durchgeführt werden können, muß dies bei langen Zahlen berücksichtigt werden. Man unterscheidet aus diesem Grund zwischen zwei verschiedenen *Kostenmaßen*:

- Beim *uniformen Kostenmaß* wird die Komplexität einer Multiplikation mit 1 angesetzt. Dies ist realistisch, wenn kleine Zahlen verwendet werde, die relativ schnell hardwaremäßig multipliziert werden können.

- Das *logarithmische Kostenmaß* besagt, daß die Kosten einer Multiplikation ungefähr mit $\log n$ bewertet werden. Diese Größe entspricht in etwa der Anzahl an Bits, die benötigt wird, um eine dezimale Zahl in Binärnotation darzustellen. Bei Rechungen mit langen Zahlen muß dieses Kostenmaß verwendet werden, um ein besseres Bild der Wirklichkeit zu erhalten. Die Komplexität der Matrixmultiplikation ist in diesem Fall $\mathcal{O}(N^3 \log q)$.

2.6 P und NP

Eine Aufgabe, die mit klassischen Mitteln nicht in polynomialer Zeit gelöst werden kann, ist die Unterscheidung zwischen konstanten und balanzierten Funktionen:

Definition 12 (Konstante und balanzierte Funktionen) *Eine Funktion $f : \{0,1\}^n \to \{0,1\}$ ist*

- konstant*, wenn $f(x) = c$ für alle x (mit $c \in \{0,1\}$)*
- balanziert*, wenn der Resultatwert* 0 *genauso häufig auftritt wie der Resultatwert* 1.

In der balanzierten Variante können die Resultate auch oszillieren, eine Funktion darf beispielsweisweise auf $0, 1, 0, 1$ zurückgeben. Die Funktion steht dabei als Orakel zur Verfügung, daß einige bestimmte garantierte Eigenschaften hat (in diesem Fall: die Lösung ist entweder konstant oder balanziert) und mit bestimmten Eingabewerden gefüttert werden kann, um eine Antwort zu liefern, ansonsten aber nicht weiter spezifiziert ist.

Wie oft muß das Orakel befragt werden, um sicher entscheiden zu können, ob es sich bei der „Black Box" um eine konstante oder balanzierte Funktion handelt? Aus einem Eingabebitstring der Länge n können im binären Fall 2^n Zahlen konstruiert werden. Betrachten wir eine Funktion mit $n = 3$, die folgende Zahlen zurückgibt:

$$0, 0, 0, 0, 1, 1, 1, 1$$

Wenn nur die Hälfte aller Zahlen abgefragt wird, kann ein Algorithmus in schlechtesten Fall noch nicht erkennen, um welche Funktion es sich handelt: Im Beispiel wäre er auf vier Nullen gestossen, was den Schluss nahelegt, daß es sich um eine konstante Funktion handelt; Abfragen der nächsten Zahl liefert aber eine Eins und damit den Beweis, daß die Funktion doch balanziert ist. Im allgemeinen benötigt man daher $2^{n-1} + 1$ Abfragen, um sicher zwischen den Varianten zu unterscheiden. [18]

Die Komplexität des Problems ist $\mathcal{O}(2^n)$, man kann es also nicht mehr in polynomialer Zeit lösen. Anders gesagt: Es handelt sich um ein hartes Problem, das mit Hilfe eines Computers nicht mehr effizient gelöst werden kann, da mit steigender Eingabelänge exponentiell viel Rechenzeit verbraucht wird.

Es gibt allerdings eine Möglichkeit, das Problem dennoch in polynomialer Zeit – nämlich $\mathcal{O}(n)$ zu lösen, wenn man sich mit einem Ergebnis zufrieden gibt, daß mit einem gewissen Fehler behaftet ist. Dies wird durch die Verwendung eines nichtdeterministischen Algorithmus ermöglicht.

Zuerst müssen unabhängig voneinander k Paare (x, y) gewählt werden. Wenn $f(x) = f(y)$ für alle gewählten Paare gilt, entscheidet sich der Algorithmus dafür, daß f konstant ist, anderenfalls ist das Resultat balanziert.

Wie groß ist der Fehler, wenn f balanziert ist? Die Wahrscheinlichkeit, daß $f(x) = f(y)$ für ein Paar (x, y) gilt, ist $1/2$. Entsprechend ist die Wahrscheinlichkeit, daß die Beziehung für k Paare gilt, gleich $(1/2)^k$. Wählt man $k = n$, sinkt die Wahrscheinlichkeit für einen Fehler auf 2^{-n}, ist damit exponentiell klein. Die Komplexität des Algorithmus ist jedoch auf $\mathcal{O}(n)$ gesunken, es handelt sich also um eine polynomiale Lösung!

Generell laßen sich viele wichtige Problem, die auf Computer berechnet werden können, in zwei Kassen einteilen: P und NP. Sie sind wie folgt definiert [14]:

- $P = \{A|$ es gibt eine Turingmaschine M und ein Polynom p mit $T(M) = A$ und $\text{time}_M \leq p(|x|) = \bigcup_{p \text{ ist Polynom}} TIME(p(n))\}$
- $NP = \bigcup_{p \text{ ist Polynom}} NTIME(p(n))$

Hierbei ist die Definition von $NTIME$ äquivalent zur Definition von $TIME$ aus Seite 15 mti dem Unterschied, daß eine nichtdeterministische Turing-Maschine anstelle einer DTM verwendet wird.

Es ist offensichtlich, daß $P \subseteq NP$ gilt, da alle deterministisch in polynomialer Zeit lösbaren Probleme leichter als die nichtdeterministisch zu lösenden Probleme sind. Eine der zentralen Fragen der theoretischen Informatik des letzten Jahrzehnte ist jedoch:

$$P \stackrel{?}{=} NP$$

Auch wenn es bisher keine effizienten (d.h. polynomialen) Algorithmen für NP-Probleme gibt, kann niemand ausschließen, daß diese existieren. Die überwiegende Lehrmeinung ist momentan allerdings, daß $P \neq NP$ gilt, d.h. daß Probleme existieren, die mit klassischen Computern prinzipiell nicht in polynomialer Zeit gelöst werden können.

2.7 Polynomiale Reduzierbarkeit

In den 70er Jahren des letzten Jahrhundert wurde eine Strukturtheorie für das $NP - P$-Problem aufgebaut, mit dessen Hilfe bereits einige tiefgreifende und überraschende Aussagen über dessen Eigenschaften möglich sind, selbst wenn nicht bekannt ist, ob die Klassen echt ineinander enthalten sind oder nicht. Das wichtigste Resultat ist das der polynomialen Reduzierbarkeit: Es ist möglich, bestimmte Probleme ineinander

[18] Dies ist die *worst case*-Komplexität. Es gibt natürlich Fälle, bei denen die Entscheidung schneller möglich ist, beispielsweise, wenn die Funktion zuerst eine Null und dann eine Eins zurückliefert. Die *durchschnittliche* Komplexität ist also geringer.

überzuführen, wobei nur ein polynomialer Aufwand zur Transition verwendet werden muß. Wenn man ein Problem A effizient lösen kann und es eine polynomiale Reduktion auf ein anderes Problem B gibt, existiert auch automatisch eine effiziente Lösung für diese Problem!

Dies ist insbesondere im Zusammenhang mit der Tatsache von Bedeutung, daß es Probleme in NP gibt, die besonders hart zu lösen sind. Präziser formuliert bedeutet das: Jedes Problem aus NP (und damit natürlich auch aus P) kann polynomial darauf reduziert werden:

Definition 13 (NP-vollständig) *Eine Problem $A \in NP$ heißt NP-*vollständig*, wenn für alle anderen Probleme $L \in NP$ gilt: $L \leq_p A$.*

„$\leq_p$" bezeichnet hierbei die polynomiale Reduzierbarkeit.

Aus der Tatsache, daß es besonders schwierige Probleme in NP gibt, resultiert folgender Satz für ein NP-vollständiges A::

Satz 2 $A \in P \Leftrightarrow P = NP$

Um die Äquivalenz zwischen P und NP nachzuweisen, genügt es also, einen polynomialen Algorithmus für *ein einziges* Problem aus NPC – der Menge aller NP-vollständigen Probleme – zu finden!

Abbildung 2 zeigt drei verschiedene Übersichts„karten", in denen mögliche Strukturen des $NP - P$-Problems wiedergegeben werden. Die linke Variante gilt, wenn $P \neq NP$ zutrifft, während die rechte eine große vereinigte Komplexitätsklasse aus P und NP ist, die entsteht, wenn die Gleichheit von P und NP gezeigt werden kann – die GUT der Informatik.

Eine dritte Variante – hier in der Mitte wiedergegeben – kann durch Überlegungen, die hier nicht nachvollzogen werden sollen, ausgeschlossen werden: Es ist unmöglich, daß alle Probleme aus $NP - NPC$ polynomial lösbar sind, während die NP-vollständigen Probleme weiterhin Algorithmen exponentieller Komplexität erfordern.

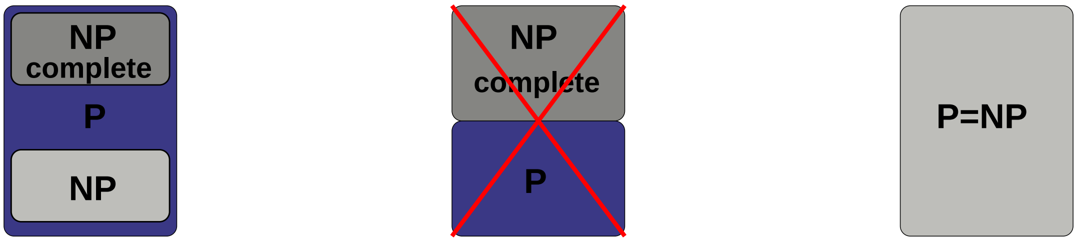

Abbildung 2: Mögliche Varianten von $NP - P$

3 Rechnen in Quantensystemen

Alle bisherigen Überlegungen basieren auf Modellen, die sich an die Gesetze der klassischen Mechanik (bzw. an die der klassischen Informationstheorie) halten. Wie üblich ändert sich die Lage, wenn man das Problem aus den Augen der Quantenmechanik betrachtet: Wie im letzten Jahrzehnt gezeigt werden konnte, können *Quantencomputer* manche Aufgaben wesentlich effizienter lösen als ihre klassischen Analoga. Dennoch ist weiterhin unklar, ob sich diese teilweise drastischen Speedups (beispielsweise bei der effizienten Faktorisierung von Zahlen) auf allgemeine Probleme übertragen lassen.

3.1 Zahlendarstellung

Klassische Computer verwenden Bits, die entweder die Werte 0 oder 1 besitzen können. In einem Quantencomputer tritt an die Stelle des Bits ein *Quantenbit* oder *Qubit*, das als eine Superposition zweier Zustände definert ist:

$$|\Psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$$

Dies sagt nicht anderes aus, als daß ein Quantenbit *gleichzeitig* die Werte 0 und 1 besitzen kann – eine Tatsache, die eigentlich nur sehr schwer zu akzeptieren ist, obwohl es sich um eine simple Folge der Linearität quantenmechanischer Zustände handelt.

Werden mehrere solcher Quantenbits kombiniert, kann man – analog wie bei klassischen Bits – Integer-Zahlen kodieren, beispielsweise nach der Formel

$$\mathrm{val} = \Sigma_{n=0}^{N-1} a_n \cdot 2^n$$

wobei die a_i die einzelnen Bitwerte 1 oder 0 darstellen.

Werden mehrere Quantenbits kombiniert, die sich in einer symmetrischen Überlagerung $1/\sqrt{2}(|0\rangle + |1\rangle)$ befinden, ergibt sich eine gleichzeitige Darstellung mehrerer natürlicher Zahlen (Normierungskonstenten werden der Einfachheit halber nicht wiedergegeben):

$$|\Psi\rangle = (|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle) = |000\rangle + |001\rangle + |010\rangle + |011\rangle + |100\rangle + |101\rangle + |111\rangle$$

Die einzelnen Zustände werden dabei durch ein Tensorprodukt $\otimes$ verknüpft, das aus den Hilberträumen der Einzelzustände (mit Dimension 2) einen kombinierten Gesamthilbertraum der Dimension 2^n erzeugt. Die Tatsache, daß die Dimension des Hilbertraums exponentiell mit der Anzahl der verwendten Qubits ansteigt, ist bereits ein Hinweis darauf, daß sich derartige Quantenzustände besonders gut zur Lösung von Problemen eignen, die exponentielle Komplexität erfordern.

Es ist üblich, anstelle der binären Darstellung dezimale Zahlen in den kets zu verwenden, was die Notation etwas übersichtlicher macht:

$$|\Psi\rangle = |0\rangle + |1\rangle + |2\rangle + |3\rangle + |4\rangle + |5\rangle + |6\rangle + |7\rangle = \Sigma_{n=0}^{7} |n\rangle$$

Ärgerlicherweise kollabiert die Wellenfunktion eines Superpositionszustandes, wenn eine Messung daran vorgenommen wird. Darum können klassische Algorithmen nicht einfach in Quantenvarianten übersetzt werden. Dennoch ist es möglich, *globale* Eigenschaften von Funktionen mit relativ geringer Komplexität zu bestimmen, die klassischen Computern nur durch exponentielle Algorithmen zugänglich sind.

3.2 Quanten-Gatter und Netzwerke

Obwohl Quanten-Turing-Maschinen oder entsprechend angepasste Registerautomaten definiert werden können, ist das Standardberechnungsmodell des Quantencomputings auf *Quantengattern* aufgebaut, die eine übersichtliche Möglichkeit bieten, um Rechenvorgänge in Quantensystemen zu beschreiben (logische Schaltznetzwerke sind ein klassisches Analogon dieser Gatter).

Ein Quantengatter verwendet n Eingangszustände und unterwirft diese einer unitären Transformation – daraus folgt auch, daß mit Quantencomputern nur reversible Vorgänge durchgeführt werden können. Wichtig für die Berechnung der Komplexität eines Algorithmus ist, daß die Operation in einer festen Zeit durchgeführt wird, unabhängig davon, wie die Eingangszustände aussehen.

3.2.1 Das Kronecker-Produkt

Bevor einige wichtige Quantengatter vorgestellt werden, muß noch erläutert werden, wie man im Vektor- und Matrixformalismus der Quantenmechanik von Einteilchenzuständen auf die Beschreibung von Mehrteilchensystemen gelangt.

Die Basiszustände $|0\rangle$ und $|1\rangle$ können als zweidimensionale Vektoren gewählt werden:

$$|0\rangle \equiv \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad |1\rangle \equiv \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Die anderen Zahlen werden durch Tensorproduktbildung aus den beiden gezeigten Basis-Qubits erzeugt. Um eine Tensormultiplikation zwischen zwei Vektoren/Matrizen durchzuführen, wird das Kroneckerprodukt verwendet, das für zwei Vektoren $A = (a_1, a_2, \ldots a_n)^{\mathrm{T}}$, $B = (b_1, b_2, \ldots b_m)^{\mathrm{T}}$ folgendermaßen definiert ist:

$$A \otimes B = \begin{pmatrix} a_1 \cdot B \\ a_2 \cdot B \\ \ldots \\ a_n \cdot B \end{pmatrix}$$

Das Tensorprodukt eines n- mit einem m-dimensionalen Vektor liefert einen Vektor der Dimension $n \cdot m$. Betrachten wir, wie beispielsweise der Vektor $|101\rangle = |1\rangle \otimes |0\rangle \otimes |1\rangle \equiv |5\rangle$ berechnet wird:

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ 1 \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \\ 1 \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

Analog können Matrizen miteinander tensormultiplitziert werden, ebenso eine Matrix mit einem Vektor.

3.2.2 Einige Quantengatter

Als *Hadamard-Gatter* bezeichnet man folgende Matrix:

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

Sie wird durch das in Abbildung 3 gezeigte Diagramm repräsentiert.

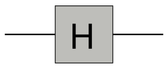

Abbildung 3: Hadamard-Gatter

Das Gatter führt die Transformation

$$|x\rangle = (-1)^x |x\rangle + |1 - x\rangle$$

durch, wobei $x \in \{0, 1\}$.

Standardanwendung dieses Gatters ist die Erzeugung (bzw. Aufhebung) symmetrischer Superpositionen, da gilt:

$\lvert 0\rangle \to \lvert 0\rangle + \lvert 1\rangle$	$\lvert 1\rangle \to \lvert 0\rangle - \lvert 1\rangle$
$\lvert 0\rangle + \lvert 1\rangle \to \lvert 0\rangle$	$\lvert 0\rangle - \lvert 1\rangle \to \lvert 1\rangle$

3.2.3 CNOT-Gatter

Das Controlled-NOT-Gatter führt eine logische NOT-Operation (Invertierung eines Bits) abhängig von einem zweiten Kontrollbit durch. Die Matrixdarstellung lautet

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

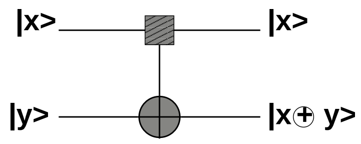

Abbildung 4: Controlled-NOT-Gatter

während die schematische Gatterdarstellung aus Abbildung 4 ersichtlich ist. Das Gatter erzeugt die Transformation

$$|x\rangle\,|y\rangle \longrightarrow |x\rangle\,|x \oplus y\rangle$$

Der Effekt des Gatters kann als Addition $\mod 2$ betrachtet werden:

$$\begin{array}{ccc|ccc} |00\rangle & \to & |00\rangle & |01\rangle & \to & |01\rangle \\ |10\rangle & \to & |11\rangle & |11\rangle & \to & |10\rangle \end{array}$$

3.2.4 Phasenverschiebungs-Gatter

Das Phasenverschiebungsgatter bewirkt eine Phasenverschiebung des Eingangszustands um einen beliebigen Winkel ϕ, sofern es sich dabei um $|1\rangle$ handelt. Der $|0\rangle$-Zustand wird nicht modifiziert:

$$P = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix}$$

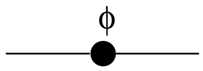

Abbildung 5: Phasenverschiebungsgatter

Abbildung 5 zeigt das zugehörige Gatterdiagramm. Die erzeugte Transformation ist

$$|x\rangle \longrightarrow e^{ix\phi}\,|x\rangle$$

Die Basiszustände werden daher folgendermaßen transformiert:

$$\begin{aligned} |0\rangle &\to |0\rangle \\ |1\rangle &\to e^{i\phi}\,|1\rangle \end{aligned}$$

3.2.5 Quantennetzwerke und vollständiger Gattersatz

Die hier gezeigten Gatter können miteinander kombiniert werden, indem man die Ausgangs„drähte“ von Gatter A mit den Eingangsdrähten von Gatter B verbindet; die Berechnung der entsprechenden Transformationen erfolgt dann sequentiell.

Ein Satz aus Quantengattern ist vollständig, wenn durch Kombination der darin enthaltenen Elemente eine beliebige unitäre Transformation eines Quantenbits erzeugt werden kann:

$$|0\rangle \rightarrow \cos\theta\,|0\rangle + e^{i\phi}\sin\theta\,|1\rangle$$

Es gibt viele verschiedene Möglichkeiten für vollständige Gattersätze, die in der Litaratur gezeigt werden. Für unsere Zwecke genügt es zu wissen, daß Hadamard- und Phasenverschiebungsgatter den gezeigten Zustand erzeugen können und daher einen vollständigen Satz bilden. Abbildung 6 zeigt ein Quantennetzwerk zur Erzeugung des gewünschten Zustandes.

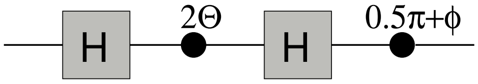

Abbildung 6: Hadamard- und Phasenverschiebungsgatter erzeugen den allgemeinsten Qubit-Zustand

3.2.6 Quantenevaluation von Funktionen

Das vorher gezeigte CNOT-Gatter kann leicht erweitert werden, um allgemeine Funktionen auf Quantenzustände anzuwenden. Da die Transformationsmatrizen unitär sind, müssen die Funktionen reversibel sein. Klassische Funktionen, die diese Bedingung nicht erfüllen, können durch Aufbewahrung eines oder mehrerer Eingangsbits immer in reversible Varianten transformiert werden.

Das Schaltbild einer Funktionsevaluation findet sich in Abbildung 7; die dadurch erzeugte Transformation ist

$$|x\rangle\,|y\rangle \longrightarrow |x\rangle\,U^x\,|y\rangle$$

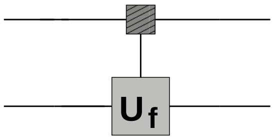

Abbildung 7: Controlled-function Gatter

Aufgrund der Linearität der verwendeten Operatoren wird eine Funktion, die auf einen Superpositionszustand angewandt wird, gleichzeitig für alle repräsentierten Zahlen berechnet:

$$\sum_x |x,0\rangle \longrightarrow \sum_x |x, f(x)\rangle$$

Dieses als *Quantenparallelismus* bekannte Phänomen kann mit klassischen Systemen nicht erzielt werden und ist ein originärer Vorteil der Quantenmechanik.

4 Der Algorithmus von Deutsch-Josza

Wir haben in Abschnitt 12 gezeigt, daß ein klassischer Computer bei der Unterscheidung zwischen konstanten und balanzierten Funktionen Algorithmen mit Komplexität $\mathcal{O}(2^n)$ verwenden muß, wenn das Ergebnis deterministisch berechnet werden soll. Unter der Bezeichnung *Deutsch-Josza*-Algorithmus wurde jedoch einer der ersten Quantenalgorithmen vorgestellt, der diese Berechnung in konstanter Zeit durchführen kann – die Zeit-Komplexität sinkt daher auf $\mathcal{O}(1)$ (es wird sich allerdings zeigen, daß dafür eine Bandkomplexität von $\mathcal{O}(n)$ erforderlich ist).

4.1 Berechnung für $n = 1$

Beschränken wir uns zunächst auf den Fall $n = 1$, um das Prinzip der Berechnung zu demonstrieren. Zur Lösung des Problems wird ein Quantennetzwerk verwendet, das in Abbildung 8 wiedergegeben ist.

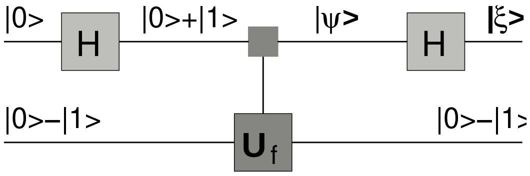

Abbildung 8: Quantennetzwerk, das den Algorithmus von Deutsch für $n = 1$ implementiert

Als Eingangszustand wird $|0\rangle \otimes |1\rangle$ verwendet, wobei das erste Qubit als Rechen- und das zweite als Hilfsbit verwendet wird. Die einzelnen Schritte, die vom gezeigten Quantengatter durchgeführt werden, können formal wie folgt dargestellt werden:

$$
\begin{aligned}
|0\rangle\,|1\rangle \xrightarrow{H_2} & (|0\rangle + |1\rangle)\,(|0\rangle - |1\rangle) \\
= & \,|0\rangle\,(|0\rangle - |1\rangle) + |1\rangle\,(|0\rangle - |1\rangle) \\
\xrightarrow{U_f} & \,|0\rangle\,(|0 \oplus f(0)\rangle - |1 \oplus f(0)\rangle) + \\
& \,|1\rangle\,(|0 \oplus f(1)\rangle - |1 \oplus f(1)\rangle)
\end{aligned}
$$

Dies führt für die einzelnen Möglichkeiten konstanter und balanzierter Funktionen zu folgendem Ergebnis:

f konstant		
$f(0) = 0, f(1) = 0$	$(\|0\rangle + \|1\rangle)(\|0\rangle - \|1\rangle)$	$\xrightarrow{H} +\|0\rangle\,\|1\rangle$
$f(0) = 1, f(1) = 1$	$(\|0\rangle + \|1\rangle)(\|1\rangle - \|0\rangle)$	$\xrightarrow{H} -\|0\rangle\,\|1\rangle$
f balanziert		
$f(0) = 0, f(1) = 1$	$(\|0\rangle - \|1\rangle)(\|0\rangle - \|1\rangle)$	$\xrightarrow{H} +\|1\rangle\,\|1\rangle$
$f(0) = 1, f(1) = 0$	$(\|0\rangle - \|1\rangle)(\|1\rangle - \|0\rangle)$	$\xrightarrow{H} -\|1\rangle\,\|1\rangle$

Das erste Bit findet sich also immer im Zustand $|0\rangle$, wenn es sich um eine konstante Funktion handelt; für balanzierte Funktionen trifft man auf den Zustand $|1\rangle$. Da beide Zustände zueinander orthogonal sind, können sie durch eine Messung eindeutig voneinander unterschieden werden.

Zentraler Punkt bei der gezeigten Rechnung ist, daß die Orakelfunktion f nur *ein einziges mal* aufgerufen werden mußte, um zu einer deterministischen Unterscheidung zwischen beiden Varianten zu gelangen.

4.2 Berechnung für Eingabezustände beliebiger Länge

Der gezeigte Algorithmus kann auf Eingabezustände beliebiger Länge n verändert werden, wozu nur geringfügige Modifikationen des Quantennetzwerks erforderlich sind, wie Abbildung 9 zeigt.

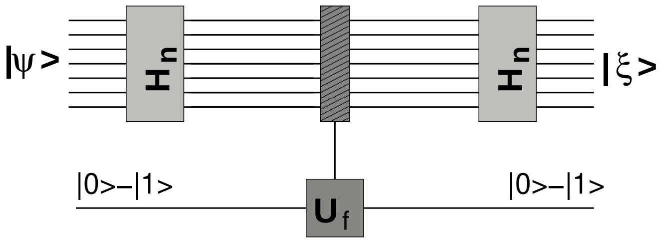

Abbildung 9: Quantennetzwerk für den Deutsch-Josza-Algorithmus mit $n \geq 1$

Die Rechnung verläuft praktisch analog zum Fall $n = 1$:

$$\begin{aligned} |\psi\rangle &= \sum_x |x\rangle \,(|0\rangle - |1\rangle) \\ &\xrightarrow{U_\mathrm{f}} \sum_x (-1)^{f(x)} |x\rangle \,(|0\rangle - |1\rangle) \\ &\xrightarrow{H_n} \sum_{x,y} (-1)^{f(x)+(x\cdot y)} |y\rangle \,(|0\rangle - |1\rangle) \end{aligned}$$

Wie kann hier zwischen den einzelnen Funktionstypen unterschieden werden? Betrachten wir die mittlere Zeile der Rechnung, da es hier am leichtesten ersichtlich ist: Wenn f konstant ist, befindet sich das erste Bit im selben Zustand wie nach der Hadamard-Transformation, d.h. es ist eine Superposition aller verwendeten Eingangszustände, die eventuell mit einem globalen Vorzeichen -1 versehen sind.

Wenn f balanziert ist, wird die Hälfte aller Zustände mit einem positiven und die andere Hälfte mit einem negativen Vorzeichen versehen. Die resultierenden Gesamtzustände für ein konstantes bzw. balanziertes f sind daher beispielsweise (für $n = 2$):

$$\begin{aligned} |\Psi_\mathrm{b}\rangle &= |0\rangle - |1\rangle - |2\rangle + |3\rangle \\ |\Psi_\mathrm{c}\rangle &= |0\rangle + |1\rangle + |2\rangle + |3\rangle \end{aligned}$$

Beide Zustände sind wegen $\langle i|\, j\rangle = \delta_i^j$ orthogonal zueinander, wie folgende Rechnung zeigt:

$$\begin{aligned} \langle \Psi_\mathrm{b}|\, \Psi_\mathrm{c}\rangle &= \langle 0|\, 0\rangle - \langle 1|\, 1\rangle - \langle 2|\, 2\rangle + \langle 3|\, 3\rangle \\ &= 1 - 1 - 1 + 1 = 0 \end{aligned}$$

Durch eine Messung kann also eindeutig zwischen beiden möglichen Resultaten unterschieden werden.

Auch für diesen Algorithmus gilt, daß die Zeitkomplexität $\mathcal{O}(const)$ ist – schließlich ist nur eine einzige Evaluation von f erforderlich, um zum gewünschten Ergebnis zu gelangen. Allerdings ist die „Bandkomplexität“ gleich $\mathcal{O}(n)$, da das Quantennetzwerk für $n \to n+1$ um eine konstante Anzahl weiterer Gatter erweitert werden muß. Dennoch schafft man, die Komplexität des Algorithmus durch Verwendung von Quantenmechanik von exponentieller auf polynomiale Komplexität zu reduzieren, was ein durchaus drastischer Schritt ist.

Literatur

[1] Dorit Aharonov: Quantum computation, arXiv:quant-ph 2001

[2] Alexander Asteroth, Christel Baier: Theoretische Informatik, Pearson, München, 2002

[3] Dirk Bouwmester, Artur Ekert, Anton Zeilinger (eds.): The physics of quantum information, Springer, Heidelberg, 2000

[4] David Deutsch: Quantum theory, the Church-Turing principle and the universal quantum computer, Proceedings of the Royal Society of London A 400, pp. 97 – 117, 1985

[5] David Deutsch, Artur Ekert, Rosella Lupacchini: Machines, logic and quantum physics, arXiv math, 1999

[6] Artur Ekert, Patrick Hayden, Hitoshi Inamori: Basic concepts in quantum computation, arXiv quant-ph, 2001

[7] Otto Forster: Analysis I, Vieweg, Braunschweig, 1999[5]

[8] Gruska, Jozef: Quantum computing, TODO, TODO

[9] J. Hopcroft, J. Ullman: Introduction to automata theory, languages and computation, Addison-Wesley, 1979[3]

[10] Donald E. Knuth: The art of computer programming, Vol. 2: Seminumerical algorithms, Addison-Wesley, 1999[3]

[11] Eugen Merzbacher, Quantum mechanics, Wiley & Sons Inc, New York, 1998[3]

[12] Michael A. Nielsen, Isaac L. Chuang: Quantum computation and quantum information, Cambridge UP, Cambridge 2000

[13] Plenio, Martin B., Quantum mechanics II, Imperial College, London, 2000

[14] Uwe Schöning: Theoretische Informatik - kurzgefasst, Spektrum, Heidelberg, 2001[4]

[15] Vlatko Vedral, Martin B. Plenio: Basics of quantum computation, Progress in quantum electronics (22) pp. 1 – 39, 1998