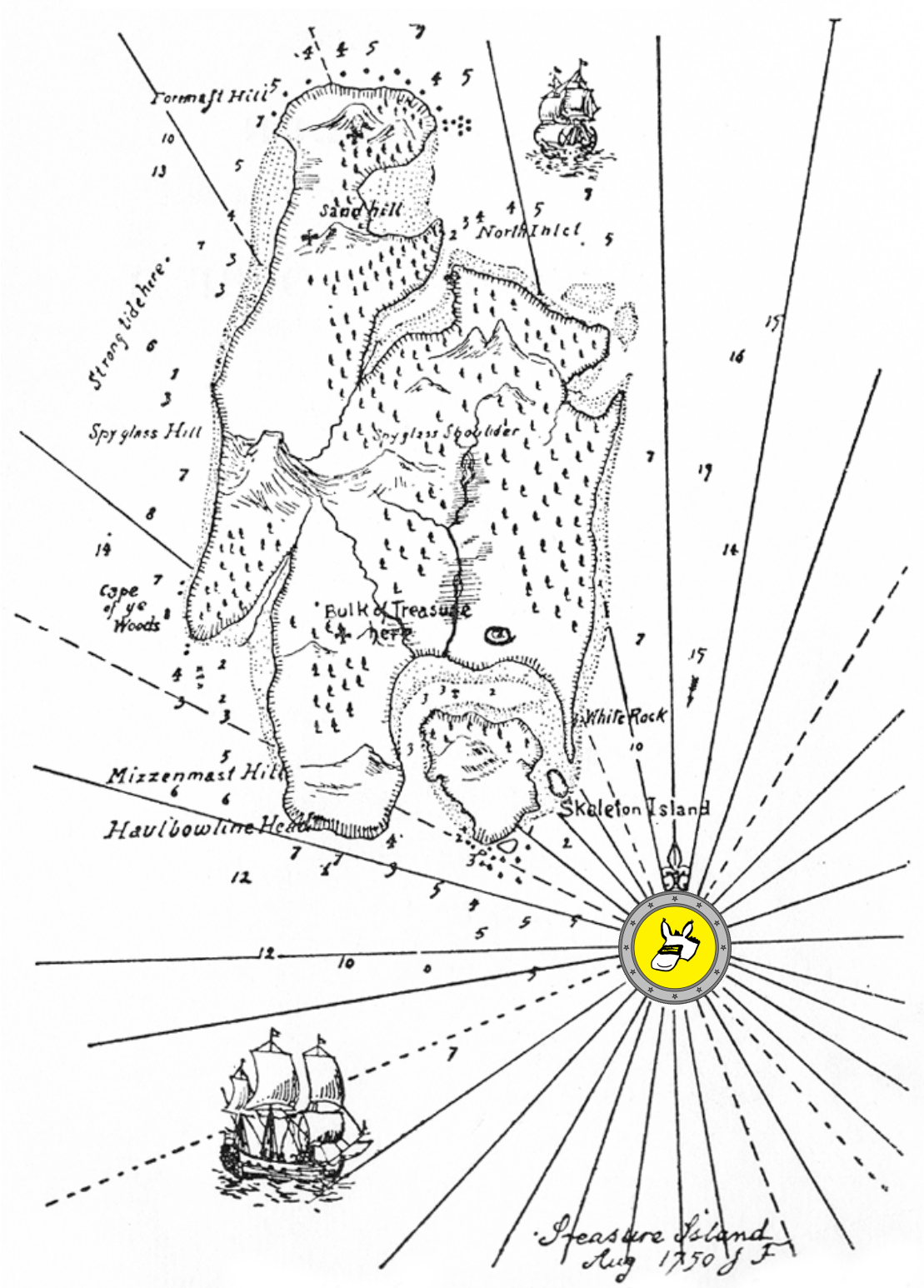
The Quantum Quest



Welkom bij de *Quantum Quest*!

In de komende weken gaan we op avontuur om de beginstelen van quantum computing te leren. Aan het einde zal je begrijpen wat quantumbits zijn, en waarom deze nuttig zijn. Onderweg zal je goede vrienden worden met Alice en Bob, die in het jaar 2058 leven en (net als jij) na schooltijd aan een quantumcomputer sleutelen. Terwijl wetenschappers nu nog quantumcomputers bouwen in hun laboratoria, zijn quantumcomputers in 2058 overal, zelfs in je broekzak! Maar zoals met alle technologie het geval is, gebruikt niet iedereen het voor de goede zaak, dus je zult je twee vrienden - Alice en Bob - moeten helpen met verschillende trucjes om zich te beschermen tegen de kwaadaardige hacker Eve. Veel succes op je avontuur!

Met hartelijke quantum-groet,
Maris Ozols & Michael Walter

Leesgids

Dit boek is onderdeel van de webklas Quantum Quest. Je vindt alle informatie op: `www.quantum-quest.nl`. Heb je feedback, of zie je fouten in deze tekst? Stuur ons dan graag een email (adressen zijn te vinden op de website).

In deze tekst staan veel oefenopgaven (in groene kaders) en huiswerkopdrachten (in rode kaders). De oefenopgaven zijn bedoeld om je begrip te testen terwijl je de tekst leest, en we geven de uitwerkingen aan het einde van elke Quest. De huiswerkopdrachten moeten aan het einde van elke week worden ingeleverd. Sommige problemen zijn gemarkeerd als 'optioneel' – we denken dat deze niet strikt noodzakelijk zijn voor het volgen van de cursus, maar ze bieden goede extra oefening. Sommige zijn ook gemarkeerd als 'uitdagend' – deze zijn een beetje moeilijker dan de rest (en daarom ook niet verplicht).

Dankwoord

We bedanken graag de volgende mensen die deze webclass mogelijk maken: Doutzen Abma, Sebastian Bach, Valerie Bettaque, Amalia Böttger, Milo Camardese, Arjan Cornelissen, Bas Dirkse, Oliver Dorogi, Jari Egbers, Yassine Ferjani, Marten Folkertsma, Koen Groenland, Galina Pass, Philip Verduyn Lunel, Anurudh Peduri, Simon Schmidt, Quinten Tupker, Mees de Vries, Jordi Weggemans, Peter Ypma. Hartelijk dank aan Milo Camardese voor de Nederlandse vertaling van deze tekst. We zijn ook Craig Gidney erg dankbaar: zijn quantumsimulator QUIRK vormt de basis van onze eigen QUIRKY. Tenslotte bedanken we alle enthousiaste studenten die deelnemen aan deze webklas!

De Quantum Quest

Maris Ozols en Michael Walter

November 2023

Inhoudsopgave

Quest 1: Componeren met kansen

Welkom bij de eerste week van *The Quantum Quest* – je staat aan het begin van een spannend avontuur!

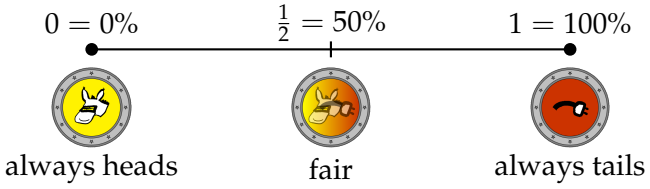
Quantum computing is een fascinerend onderwerp dat snel tot je verbeelding kan spreken. Je belandt in een vreemde wereld vol met nieuwe verschijnselen en interessante paradoxen. Quantummechanica kan echter ook erg verwarrend zijn! Je kan gemakkelijk verstrengeld raken in deze contra-intuïtieve wereld, of verdwalen in de exponentieel grote ruimte van quantumtoestanden. Om zulke problemen te voorkomen, moet je je voorbereiden door eerst vertrouwd te raken met de wereld van waarschijnlijkheden. Zodra je een meester bent geworden in kansrekening, kan je ook de deur naar de quantumwereld openen!

Het doel van de eerste quest is om meer te leren over waarschijnlijkheden en probabilistische bits: wat zijn de toestanden van een probabilistische bit, welke bewerkingen kunnen we erop loslaten, en hoe kunnen we door een meting informatie uit een probabilistische bit halen? In de tweede quest staan quantumbits centraal, die erg lijken op probabilistische bits.

1.1 Probabilistische bits

We gebruiken waarschijnlijkheden of kansen wanneer een proces meerdere mogelijke uitkomsten heeft, en we nog niet weten welke uitkomst zich daadwerkelijk voordoet. Een uitkomst die helemaal zeker is, heeft een waarschijnlijkheid van 1 (ofwel 100%), terwijl een uitkomst die zich nooit voordoet een waarschijnlijkheid van 0 heeft.

Als voorbeeld kunnen we denken aan het opgooien van een muntje. Als je een muntje opgooit en bedekt met je handen zonder ernaar te kijken, zijn er twee mogelijke uitkomsten – het muntje laat 'kop' of 'munt' zien. Bij een *zuiver* muntje hebben beide uitkomsten dezelfde waarschijnlijkheid, dus geven we aan beide een kans van $\frac{1}{2} = 0.50$ of 50% (dit wordt aangegeven met in Fig. 1.1). Maar het muntje kan een afwijking hebben, waardoor de kans groter is dat het op de ene kant valt dan op de andere. Afhankelijk van hoe partijdig het muntje is, kunnen we een heel spectrum van mogelijkheden voorstellen: een extreem partijdig muntje zou altijd op 'kop' kunnen vallen, terwijl een ander altijd op 'munt' zou kunnen vallen (zie en aan de linker- en rechterkant van Fig. 1.1). Het eerste muntje laat 'munt' zien met waarschijnlijkheid 0 (omdat het altijd op 'kop' valt), terwijl het tweede muntje 'munt' laat zien met een waarschijnlijkheid van 1.



Figuur 1.1: Een probabilistische bit die de toestand van een willekeurig ezelsmuntje beschrijft.

Omdat we ons geen zorgen willen maken over de vorm en het formaat van de muntjes, is het handig om de informatie van het opgooien van een muntje wat abstracter te maken. We kunnen dit doen door de uitkomsten 'kop' en 'munt' te associëren met de bitwaarden 0 en 1. Het opgooien van het muntje kan dan worden beschreven door een probabilistische bit die gelijk is aan 0 met een bepaalde waarschijnlijkheid p_0 en gelijk aan 1 met een bepaalde waarschijnlijkheid p_1, waarbij $p_0, p_1 \geq 0$ zijn en $p_0 + p_1 = 1$. Als het muntje bijvoorbeeld zuiver is, is $p_0 = p_1 = \frac{1}{2}$, zoals hierboven is uitgelegd.

Is het je opgevallen dat het genoeg is om maar één van de waarschijnlijkheden p_0 of p_1 te geven, omdat je de ene uit de andere kunt halen? Voor de duidelijkheid geven we toch altijd beide aan en schrijven we ze op als een vector:

$$p = \begin{pmatrix} p_0 \\ p_1 \end{pmatrix}. \tag{1.1}$$

We noemen deze vector de kansverdeling of de toestand van de probabilistische bit. Deze vectornotatie is niet alleen een handige manier om alle waarschijnlijkheden te verzamelen in een mooie tabel, maar geeft ook een mooie geometrische manier om een probabilistische bit voor te stellen. Bovendien zal het ons helpen de gelijkenis te zien tussen probabilistische en quantumbits.

We noemen de toestanden die horen bij 0 of 'kop' en bij 1 of 'munt' deterministisch, omdat ze de toestand van het muntje volledig beschrijven (er is geen onduidelijkheid over welke kant boven ligt). In Vgl. (1.1) komen deze overeen met de kansverdelingen met respectievelijk $p_0 = 1$, $p_1 = 0$ en $p_0 = 0$, $p_1 = 1$. Aangezien we ze vaak zullen gebruiken, is het handig er een verkorte notatie voor in te voeren:

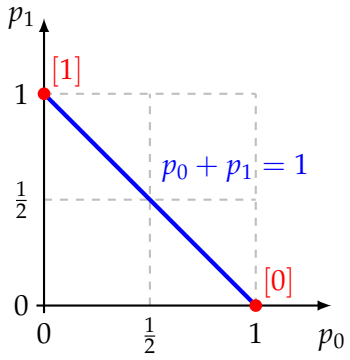
$$[0] = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad [1] = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{1.2}$$

Deze notatie kan in eerste instantie verwarrend lijken, maar je kunt $[0]$ en $[1]$ ook zien als en  of als [kop] en [munt].

Deze twee toestanden vormen een basis van alle toestanden, wat betekent dat we elke andere mogelijke toestand van een probabilistische bit kunnen uitdrukken als een lineaire combinatie van deze twee toestanden:

$$\begin{pmatrix} p_0 \\ p_1 \end{pmatrix} = p_0 \begin{pmatrix} 1 \\ 0 \end{pmatrix} + p_1 \begin{pmatrix} 0 \\ 1 \end{pmatrix} = p_0[0] + p_1[1]. \tag{1.3}$$

Je kunt nu alle mogelijke toestanden van een probabilistische bit voorstellen als een lijnstuk dat de punten $[0]$ en $[1]$ verbindt die overeenkomen met de twee deterministische toestanden (zie Fig. 1.2).



Figuur 1.2: De mogelijke toestanden van een probabilistische bit vormen het blauwe lijnstuk.

Oefenopgave 1.1: Het blauwe lijnstuk beter begrijpen

Volgens Fig. 1.2 vormen de mogelijke toestanden van een probabilistische bit een lijnstuk. Neem de tijd om dit te bestuderen. Kijk dan of je de volgende vragen kunt beantwoorden:

1. Waarom liggen de mogelijke toestanden van een probabilistische bit op een lijn?

2. Waarom stopt deze lijn op de assen en gaat de lijn niet verder?

3. Welk punt op het lijnstuk komt overeen met een zuivere munt?

1.1.1 Vermenigvuldiging van waarschijnlijkheden

Als je twee muntjes opgooit, wat is dan de waarschijnlijkheid dat beide muntjes op 'kop' vallen? Neem aan dat de twee muntjes beschreven worden door probabilistische bits

$$a = \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} \qquad \text{en} \quad b = \begin{pmatrix} b_0 \\ b_1 \end{pmatrix}, \tag{1.4}$$

waarbij de uitkomst 0 overeenkomt met 'kop' en de uitkomst 1 met 'munt'. Dan is voor muntje a de waarschijnlijkheid van 'kop' a_0 en voor muntje b is het b_0. (We nemen niet aan dat de muntjes zuiver zijn, dus deze waarschijnlijkheden zijn niet per se 50%.) De waarschijnlijkheid dat beide muntjes 'kop' tonen wordt gegeven door de waarschijnlijkheden van de twee afzonderlijke uitkomsten te *vermenigvuldigen*:

$$p_{00} = a_0 b_0. \tag{1.5}$$

Merk op dat $p_{00} \leq a_0$ en $p_{00} \leq b_0$, aangezien $a_0 \leq 1$ en $b_0 \leq 1$. Dit is logisch, want het is niet waarschijnlijker (en meestal minder waarschijnlijk) om bij beide muntjes kop te krijgen dan het is bij elk muntje afzonderlijk. Je kan op dezelfde manier de waarschijnlijkheden berekenen van alle andere combinaties van kop en munt. We kunnen alle vier mogelijkheden hiervan samenvatten in de volgende tabel:

$$\begin{aligned} p_{00} &= a_0 b_0, & p_{01} &= a_0 b_1, \\ p_{10} &= a_1 b_0, & p_{11} &= a_1 b_1. \end{aligned} \tag{1.6}$$

We noemen twee uitkomsten onafhankelijk als ze van twee verschillende bronnen komen en het voordoen van de ene uitkomst niets zegt over het voordoen van de andere. Meestal wordt zo'n situatie beschreven met het woord 'en'. Bijvoorbeeld: "Het eerste muntje is 'kop' *en* het tweede muntje is 'munt'". We vermenigvuldigen waarschijnlijkheden als we willen weten of twee onafhankelijke uitkomsten tegelijk hebben plaatsgevonden.

Oefenopgave 1.2: Waarschijnlijkheden vermenigvuldigen

Alice verveelt zich tijdens haar wiskundeles en begint op haar digitale horloge te kijken. De seconde-teller op haar horloge kan waarden weergeven van 00 tot 59. Stel dat Alice op een willekeurig moment in de komende minuut naar de seconde-teller op haar horloge kijkt.

1. Wat is de waarschijnlijkheid dat ze 00 ziet?

2. Wat is de waarschijnlijkheid dat het laatste getal 0 is?

3. Wat is de waarschijnlijkheid dat het eerste getal 0 is?

4. Leg uit dat de waarden van beide getallen onafhankelijk van elkaar zijn. Controleer je antwoord op vraag 1 door de waarschijnlijkheden uit de vragen 2 en 3 te vermenigvuldigen.

1.1.2 Waarschijnlijkheden optellen

We bekijken nu een ingewikkelder probleem, waarbij je de muntjes a en b opgooit. Wat is de waarschijnlijkheid dat beide muntjes dezelfde uitkomst hebben? Dit kan op twee manieren

gebeuren: óf beide muntjes zijn 'kop' óf beide muntjes zijn 'munt'. We weten al van Vgl. (1.6) dat de kansen op deze twee afzonderlijke uitkomsten

$$p_{00} = a_0 b_0, \qquad p_{11} = a_1 b_1.$$

zijn. De waarschijnlijkheid dat één van deze twee uitkomsten zich voordoet, krijgen we dan door deze waarschijnlijkheden *op te tellen*:

$$p_{00} + p_{11} = a_0 b_0 + a_1 b_1. \tag{1.7}$$

Je tel waarschijnlijkheden op wanneer je meerdere uitkomsten van hetzelfde experiment wilt combineren. Zulke gecombineerde uitkomsten kunnen meestal beschreven worden met het woord 'of'. Bijvoorbeeld: "Beide muntjes zijn 'kop' *of* beide muntjes zijn 'munt'". Let op: dat mag alleen als de muntjes elkaar niet beïnvloeden!

Oefenopgave 1.3: Waarschijnlijkheden optellen

Bob verveelt zich ook tijdens de wiskundeles. Hij ziet dat Alice naar haar horloge zit te staren, dus hij kijkt ook op zijn eigen horloge. Verrassend genoeg geeft de seconde-teller 44 aan, wat Bob erg onwaarschijnlijk lijkt. Wat is de waarschijnlijkheid dat beide getallen van de seconde-teller hetzelfde zijn als Bob op een willekeurig moment binnen een minuut op zijn horloge kijkt?

Nu dat we hebben geleerd wanneer je waarschijnlijkheden moet optellen en vermenigvuldigen, kan je proberen je eerste huiswerkopdracht op te lossen!

Huiswerkopdracht 1.1: Tegenovergestelde muntjes

Alice gooit twee muntjes op, die we a en b noemen, met kansverdelingen

$$a = \begin{pmatrix} 2/3 \\ 1/3 \end{pmatrix}, \qquad b = \begin{pmatrix} 3/4 \\ 1/4 \end{pmatrix}.$$

Wat is de waarschijnlijkheid dat de twee muntjes *tegenovergestelde* uitkomsten krijgen?

1.1.3 Waarschijnlijkheid en berekeningen

Zijn probabilistische bits eigenlijk wel nuttig voor berekeningen? In eerste instantie lijkt het misschien dat ze niet bijzonder zinvol zijn, omdat de waarden 0 en 1 van een gewone bit staan voor definitieve informatie, terwijl een probabilistische bit staat voor *beperkte* informatie (of het *gebrek* aan informatie). Waarom zou ik opslagruimte op mijn computer verspillen aan probabilistische bits die mijn gebrek aan informatie aangeven, als ik in plaats daarvan ook de werkelijke informatie die ik heb kan opslaan, ook al is die incompleet? Het voordeel van probabilistische bits is dat ze gedeeltelijke kennis nauwkeuriger beschrijven – als je iets niet weet, is het beter om dat te erkennen en een gok te wagen dan het is om te doen alsof je het juiste antwoord met zekerheid weet. Dit wordt toegelicht in de volgende opdracht, waarin Alice's robot-ezel een beslissing moet maken zonder alle informatie te hebben.

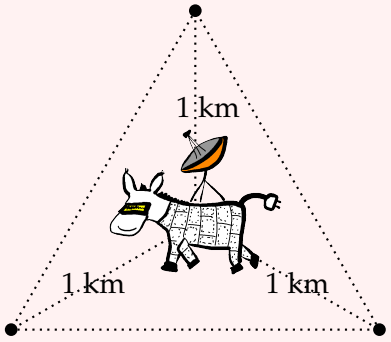
Huiswerkopdracht 1.2: Alice's ezel

Het probleem: Alice wil haar robot-ezel zo programmeren dat hij uit zichzelf naar een laadstation kan lopen en zichzelf kan opladen. Er zijn drie stations in de buurt, elk op een afstand van 1 km van de ezel, die een gelijkzijdige driehoek vormen met de ezel in het midden. Alice's ezel heeft nog maar genoeg batterij over om 2,8 km te lopen.

Alice is van plan om een programma te uploaden naar de robot-ezel die hem vertelt waar

hij heen moet lopen, maar ze weet dat haar kwaadaardige klasgenoot Eve probeert haar te saboteren.

Omdat Eve alles kan lezen wat over de WiFi wordt uitgezonden, kan Eve ook zien welk programma Alice aan het uploaden is. Daarom gaat Alice, zodra het programma is geüpload, haar ezel loskoppelen van de WiFi, zodat Eve zijn bewegingen niet kan volgen. Terwijl de ezel loopt, is de enige manier voor Eve om de ezel te saboteren, het hacken en uitschakelen van de laadstations waar hij naar toe is geprogrammeerd. Maar Eve kan slechts twee stations uitschakelen voordat haar inbraak wordt ontdekt. Omdat Eve de bewegingen van de ezel niet kan volgen, moet ze alleen op basis van Alice' programma beslissen welke twee laadstations ze wil uitschakelen.

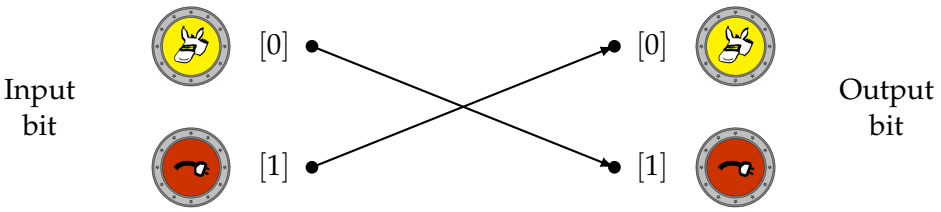


Vragen:

1. Hoeveel laadstations kan de ezel bezoeken voordat haar batterij leeg is?
2. Stel dat Alice haar ezel zo programmeert dat hij de stations in een bepaalde volgorde bezoekt. Kan Eve voorkomen dat de ezel een werkend laadstation bereikt? Hou in gedachten dat Eve volledige toegang heeft tot Alice's programma en dus weet in welke volgorde de ezel geprogrammeerd is om de stations te bezoeken.
3. Stel dat Alice de ezel programmeert om voor zichzelf willekeurige keuzes te maken over waar hij heen gaat. (Eve kan zien dat Alice dit heeft geprogrammeerd, maar ze kan niet voorspellen welke keuzes de ezel zal maken als hij eenmaal begint te lopen). Welke strategie moet Alice uploaden naar de ezel, en welke hackstrategie moet Eve gebruiken om deze strategie tegen te gaan? Wat is de kans dat Alice's ezel met succes een werkend laadstation bereikt als zowel Alice als Eve optimale strategieën gebruiken?

1.2 Bewerkingen op een probabilistische bit

Nu dat we informatiebits hebben beschreven als vectoren, kunnen we bewerkingen op deze bits beschrijven door lineaire transformaties, waarbij we hulpmiddelen uit de lineaire algebra kunnen gebruiken. Denk bijvoorbeeld aan de bewerking waarbij 'kop' en 'munt' van een ezelsmuntje worden verwisseld:



We noemen deze bewerking NOT en schrijven het wiskundig zo op:

(1.8)

Als we de notatie van Vgl. (1.2) gebruiken, kunnen we dit ook schrijven als

$$\text{NOT}\,[0] = [1], \qquad \text{NOT}\,[1] = [0]. \tag{1.9}$$

Merk op dat we NOT p schrijven als afkorting van NOT(p) – beide betekenen gewoon dat de operatie NOT wordt toegepast op een vector p.[1] We zullen bewerkingen meestal met hoofdletters schrijven om ze te kunnen onderscheiden van getallen en vectoren.

Kijk nu even terug naar Vgl. (1.2), waarin staat dat de vectoren $[0]$ en $[1]$ staan voor de deterministische toestanden 0 en 1 van een probabilistische bit. Omdat de NOT-bewerking deze twee vectoren verwisselt, wordt de waarde van de bit omgewisseld. Dit is precies waarom we het 'NOT' hebben genoemd – het staat voor de logische ontkenning! Een eenvoudige toepassing van NOT is het invoeren van gegevens in je computer. Als alle bits van je computer in het begin op 0 staan, kun je er een paar veranderen in 1 om gegevens in te voeren – dit is vaak de eerste stap van een berekening.

Hoe kunnen we nu de NOT-bewerking op een probabilistische bit $p = \binom{p_0}{p_1}$ definiëren? Met een waarschijnlijkheid van p_0 is de bit nul en wordt hij dus omgezet in een één. Met een waarschijnlijkheid van p_1 is de bit een één en wordt deze dus omgezet in een nul. Het effect van de NOT-bewerking op een probabilistische bit is dus simpelweg

$$\text{NOT}\begin{pmatrix} p_0 \\ p_1 \end{pmatrix} = \begin{pmatrix} p_1 \\ p_0 \end{pmatrix}. \tag{1.10}$$

Dit geeft precies het resultaat van Vgl. (1.9) als $p_0 = 1$ (en $p_1 = 0$) of $p_0 = 0$ (en $p_1 = 1$). De NOT-bewerking en Vgl. (1.10) kun je intuïtief als volgt zien: als je je een probabilistische bit voorstelt als een muntje dat je hebt opgegooid maar nog niet naar hebt gekeken, dan komt de NOT-bewerking overeen met het omdraaien van het muntje (wederom zonder ernaar te kijken).

Oefenopgave 1.4: De NOT-bewerking visualiseren

Zoals we in Fig. 1.2 hebben gezien, komen alle mogelijke toestanden van een probabilistische bit overeen met een lijnstuk. Laten we ons nu proberen voor te stellen hoe de NOT-bewerking dit lijnstuk omvormt.

1. Bekijk een arbitrair[a] punt met coördinaten (p_0, p_1) op dit lijnstuk. Waar wordt dit punt door de NOT-bewerking op afgebeeld?
2. Waar worden de twee eindpunten van het lijnstuk op afgebeeld?
3. Is er een punt op het lijnstuk dat op zichzelf wordt afgebeeld?

[a] Met 'arbitrair' bedoelen we dat je berekening moet werken voor elke waarde van p_0 and p_1! Het is verstandig om verder te rekenen met deze symbolen zonder ze in te vullen: behandel ze als een variabele, of een nog onbekend getal.

1.2.1 Uitbreiden door lineariteit

Hoe moeten we $M\binom{p_0}{p_1}$ definiëren als M een willekeurige bewerking van een bit is? Zoals voorheen nemen we aan dat we het effect van M op de twee mogelijke waarden van de bit al kennen, en schrijven we $M[0]$ op voor het resultaat van de bewerking als de bit nul is, en $M[1]$ voor het resultaat van de bewerking als de bit één is. (Voor de NOT operatie was dit precies wat we deden in Vgl. (1.9)). Laten we nu proberen dezelfde redenering als hierboven toe te passen. Met een waarschijnlijkheid van p_0 is de bit nul en krijgen we dus $M[0]$ door de bewerking M toe te passen. Met waarschijnlijkheid p_1 is de bit één en krijgen we in plaats daarvan $M[1]$. Bij

[1] We kunnen ook NOT $\cdot\, p$ gebruiken, omdat deze bewerking overeenkomt met een matrix-vector-vermenigvuldiging.

elkaar zien we dat we $M\left(\begin{smallmatrix}p_0\\p_1\end{smallmatrix}\right)$ zouden moeten definiëren als

$$M\begin{pmatrix}p_0\\p_1\end{pmatrix} = p_0\,M\,[0] + p_1\,M\,[1], \tag{1.11}$$

waarbij $p_0\,M\,[0]$ aangeeft dat we de vector $M\,[0]$ vermenigvuldigen met de waarschijnlijkheid p_0.

Oefenopgave 1.5: NOT op probabilistische bits

Laat zien dat als M de NOT-bewerking is, Vgl. (1.11) precies op Vgl. (1.10) uitkomt.

Door gebruik te maken van Vgl. (1.3) kunnen we Vgl. (1.11) ook als volgt schrijven:

$$M\Big(p_0\,[0] + p_1\,[1]\Big) = p_0\,M\,[0] + p_1\,M\,[1]. \tag{1.12}$$

Merk op dat het onderscheid tussen de twee kanten zit in de volgorde van de bewerkingen: aan de linkerkant nemen we eerst een lineaire combinatie en dan passen we M toe, terwijl we aan de rechterkant eerst M toepassen en dan pas de lineaire combinatie nemen. Deze vergelijking ziet er erg uit als de bekende regel $a(b+c) = ab + ac$ voor getallen (de 'distributieve wet').

Als M een bewerking van probabilistische bits is die voldoet aan Vgl. (1.12), dan zeggen we dat M lineair is. Onze regel uit Vgl. (1.11) en (1.12) voor het uitbreiden van M van bits naar probabilistische bits noemen we uitbreiden door lineariteit. Hetzelfde concept zullen we vaak ook gebruiken voor quantumbits.

1.2.2 Willekeurige bewerkingen

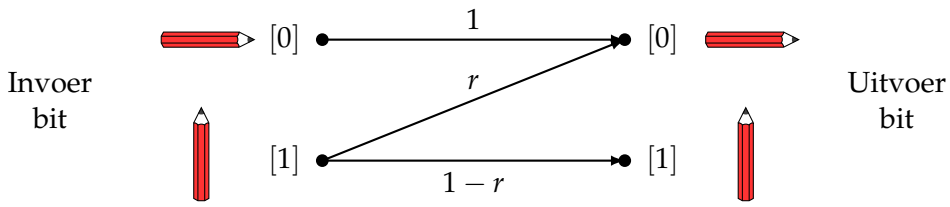
Merk op dat we in onze afleiding van Vgl. (1.11) niet hebben aangenomen dat $M\,[0]$ en $M\,[1]$ weer één van de twee deterministische toestanden $[0]$ of $[1]$ van een bit waren (al was dit toevallig wel het geval voor de NOT-bewerking). Dit betekent dat Vgl. (1.11) net zo goed werkt als $M\,[0]$ of $M\,[1]$ probabilistische bits zijn! In dat geval zeggen we dat M een willekeurige bewerking is.

Een van de simpelste voorbeelden van een willekeurige bewerking is als volgt. Stel je voor dat je $[0]$ beschrijft door een potlood horizontaal op een tafel te leggen en $[1]$ door het verticaal te balanceren. Als je voorzichtig met je hand op de tafel slaat, kan het potlood omvallen, waardoor de toestand verandert van $[1]$ in $[0]$. Maar als het al op de tafel lag, verandert de toestand $[0]$ niet. Dus door de tafel te slaan wordt de toestand van het potlood *met een zekere waarschijnlijkheid gereset* naar $[0]$; Hoe harder je slaat, hoe groter de waarschijnlijkheid dat je het potlood reset.

Wiskundig gezien wordt deze probabilistische reset beschreven door een bewerking $R(r)$ die gedefinieerd is als

$$R(r)\,[0] = [0] = \begin{pmatrix}1\\0\end{pmatrix}, \qquad R(r)\,[1] = r\,[0] + (1-r)\,[1] = \begin{pmatrix}r\\1-r\end{pmatrix}, \tag{1.13}$$

waarbij $r \in [0,1]$ de *reset*-waarschijnlijkheid is. Je kunt het effect van deze bewerking als volgt voor je zien:



Door lineariteit kunnen we deze bewerking uitbreiden naar alle toestanden:

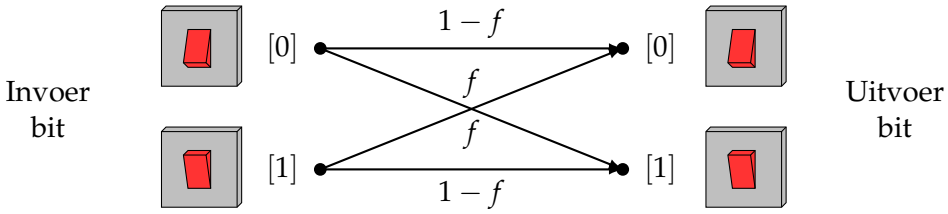
$$R(r)\begin{pmatrix}p_0\\p_1\end{pmatrix} = p_0\,R(r)\,[0] + p_1\,R(r)\,[1]$$
$$= p_0\begin{pmatrix}1\\0\end{pmatrix} + p_1\begin{pmatrix}r\\1-r\end{pmatrix} = \begin{pmatrix}p_0 + p_1 r\\ p_1(1-r)\end{pmatrix}.$$

De speciale gevallen van deze vergelijking zijn dat $R(0)$ de toestand helemaal niet verandert, terwijl $R(1)$ elke toestand reset naar $[0]$.

Een ander interessant voorbeeld van een willekeurige bewerking is de probabilistische flip bewerking $F(f)$ die de invoerbit flipt met kans f en met rust laat met kans $1-f$:

$$F(f)\,[0] = (1-f)\,[0] + f\,[1], \qquad F(f)\,[1] = f\,[0] + (1-f)\,[1], \tag{1.14}$$

waarbij $f \in [0,1]$ de *flip*-waarschijnlijkheid is. Om het wat intuïtiever te maken, stel je voor dat $[0]$ en $[1]$ twee toestanden van een lichtschakelaar aan de muur voorstellen. Als je een kussen naar de schakelaar gooit, zul je hem met waarschijnlijkheid f met succes raken en flippen, en met waarschijnlijkheid $1-f$ onveranderd laten. Je kunt het effect van $F(f)$ dus als volgt voorstellen:



De volgende oefenopgave zal helpen om de probabilistische flip-bewerking beter te leren kennen.

Oefenopgave 1.6: Probabilistische flip

$F(f)$ staat hier voor de probabilistische flip bewerking van Vgl. (1.14).

1. Schrijf $F(f)\,[0]$ en $F(f)\,[1]$ op als vectoren.
2. Voor welke waarde van f gedraagt $F(f)$ zich als NOT? Hoe kunnen we met behulp van F een probabilistische bit in een willekeurige toestand $\left(\begin{smallmatrix}p\\1-p\end{smallmatrix}\right)$ voorbereiden vanuit $[0]$?
3. Breid $F(f)$ door lineariteit uit naar probabilistische bits door $F(f)\left(\begin{smallmatrix}p_0\\p_1\end{smallmatrix}\right)$ te berekenen.
4. Laat $\left(\begin{smallmatrix}p_0\\p_1\end{smallmatrix}\right)$ een arbitraire kansverdeling zijn. Laat zien dat $F(1/2)\left(\begin{smallmatrix}p_0\\p_1\end{smallmatrix}\right) = \left(\begin{smallmatrix}1/2\\1/2\end{smallmatrix}\right)$.

Uit het laatste deel van deze opgave blijkt dat $F(1/2)$ altijd de uniforme verdeling oplevert, ongeacht de verdeling waarmee je begint. Dit kan gebruikt worden om het opgooien van een zuiver muntje te simuleren. In de volgende opgave zul je zien dat je door het zorgvuldig aanpassen van de flip-waarschijnlijkheid f , je ook $F(f)$ kunt gebruiken om de partijdigheid van een muntje te veranderen.

Huiswerkopdracht 1.3: Chocolade muntje

Bob is vandaag jarig! Omdat hij dol is op chocolade besluit Alice een chocolade muntje voor hem te maken. Om het cadeau extra speciaal te maken, wil ze het muntje zo maken dat het, als je het op tafel laat tollen, zal landen op 5 met een waarschijnlijkheid $q = 5/15$,

wat staat voor de verjaardag van Bob, 15 mei. Na wat experimenteren met verschillende vormen slaagt Alice erin een chocolade muntje met de juiste waarschijnlijkheid te produceren. Enthousiast laat ze het op tafel liggen en rent ze naar de winkel om een mooie verjaardagskaart te kopen.

Wanneer ze terugkeert, komt Alice er plotseling achter dat het muntje in de zon is blijven liggen en dat de rand ervan is gesmolten. Na wat uitproberen stelt Alice vast dat de nieuwe waarschijnlijkheid om te krijgen, $p = 4/15$ is. Er is nog te weinig tijd om het probleem op te lossen, dus Alice schrijft op de verjaardagskaart dat zodra het muntje is gevallen, Bob hem moet omdraaien met waarschijnlijkheid f, en alleen dan zal hij met de juiste waarschijnlijkheid q waarnemen. Help Alice de juiste waarde van f te bepalen.

Hint: De waardes p, q en f moeten voldoen aan $F(f)\binom{p}{1-p} = \binom{q}{1-q}$.

Oefenopgave 1.7: Flip vanuit de reset en NOT (optioneel en uitdagend)

Hoe kan je $F(f)$ maken vanuit de bewerkingen $R(r)$ en NOT?

1.3 Een probabilistische bit meten

Als je een zuiver muntje opgooit en onmiddellijk afdekt, kan je niet weten aan welke kant het muntje terechtkwam. In dit geval wordt je kennis over de toestand van het muntje beschreven door de uniforme verdeling.

$$= \begin{pmatrix} 1/2 \\ 1/2 \end{pmatrix} = \frac{1}{2}[0] + \frac{1}{2}[1]. \tag{1.15}$$

Maar als je je hand weghaalt en 'kop' ziet, wordt je kennis geupdate naar

$$= [0] \tag{1.16}$$

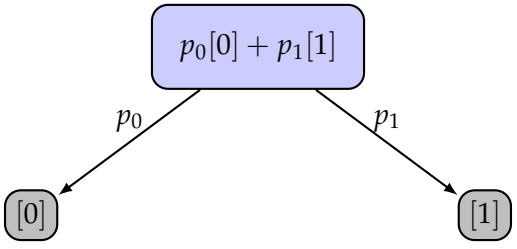
omdat je nu met zekerheid weet dat kop boven ligt. Het proces van het openleggen van een probabilistisch muntje om te bepalen welke kant boven ligt, noemen we een meting. [2]

Merk op uit Vgl. (1.15) en (1.16) dat de toestand van het muntje voor en na de meting anders is. Na de meting ben je namelijk niet meer onwetend over welke kant boven ligt. Stel nu dat je het muntje weer bedekt nadat je het net gemeten hebt. Wat is de toestand nu? Uiteraard nog steeds

$$= [0] \tag{1.17}$$

omdat je al weet dat 'kop' omhoog ligt. Sterker nog, als je het muntje opnieuw meet (bekijkt), zal je nog steeds 'kop' zien. Op dezelfde manier, als je de eerste keer dat je een willekeurig muntje opmeet 'munt' kreeg, zal je 'munt' blijven krijgen, hoe vaak je het ook opnieuw opmeet.

In het algemeen, als je een probabilistische bit hebt die beschreven wordt door de verdeling $\binom{p_0}{p_1}$, dan levert het meten ervan de uitkomst 0 (of 'kop') op met waarschijnlijkheid p_0 en 1 (of 'munt') met waarschijnlijkheid p_1:



(1.18)

[2] We hebben deze term overgenomen van quantumcomputing, waar we zullen zien dat er een vergelijkbare procedure bestaat.

De toestand van de probabilistische bit *na* de meting is niet meer $\binom{p_0}{p_1}$, maar een van de basistoestanden $[0] = \binom{1}{0}$ of $[1] = \binom{0}{1}$, afhankelijk van het meetresultaat. Als je bijvoorbeeld de uitkomst 1 (of 'munt') krijgt, is de nieuwe toestand $[1]$. In het algemeen *verandert* een meting dus de toestand!

Een belangrijk kenmerk van metingen is dat je er *niet* de waarschijnlijkheden p_0 en p_1 uit kunt halen – het enige wat je als meetresultaat krijgt is een enkele bit 0 of 1. Bovendien is je oorspronkelijke probabilistische bit $\binom{p_0}{p_1}$ verloren na de meting, dus je kunt het niet opnieuw meten. Dit is eigenlijk vrij intuïtief. Als we een muntje slechts één keer opgooien, krijgen we maar één willekeurig resultaat – maar uit dit ene resultaat alleen kunnen we niet bepalen of het muntje zuiver of partijdig was!

Maar stel dat we hetzelfde muntje nu een groot aantal keren opgooien. In dat geval zouden we verwachten dat de fractie van de keren dat we de uitkomst 1 krijgen ongeveer p_1 is. Met andere woorden,

$$\frac{N_1}{N} \approx p_1, \tag{1.19}$$

waarbij N het totale aantal metingen is en N_1 het aantal keren dat we de uitkomst 1 hebben gekregen. Hoe meer uitkomsten we verzamelen, hoe beter de benadering wordt.[3]

Dit geeft ons een procedure voor het schatten van p_1. Omdat $p_0 + p_1 = 1$ geeft dit uiteraard ook een schatting van p_0. Alice kan deze procedure bijvoorbeeld gebruiken om de waarschijnlijkheid te schatten dat haar partijdige muntje in Huiswerkopdracht 1.3 op  en  terechtkomt. Dit kan je nu zelf uitproberen.

Huiswerkopdracht 1.4: Muntjes opgooien

1. Zoek een muntje en teken met een marker een 0 en een 1 op de twee kanten. Gooi het muntje 30 keer op en schrijf de uitkomsten op in een tabel zoals hieronder:

Aantal keer opgegooid N	1	2	3	4	5	6	7	8	9	…	30
De N^e uitkomst	1	0	1	0	0	0	1	1	1	…	1

   (De grijze uitkomsten zijn alleen als voorbeeld bedoeld. Vervang ze door je eigen uitkomsten.)

2. Schat met Vgl. (1.19) de waarschijnlijkheid dat je muntje de uitkomst 1 krijgt.

3. Het is interessant om te zien hoe de schatting verandert als je het aantal keer dat je het muntje opgooit N verhoogt. Breid de tabel uit deel 1 hiervoor uit met drie rijen, zodat het er zo uitziet:

Aantal keer opgegooid N	1	2	3	4	5	6	7	8	…	30
De N^e uitkomst	1	0	1	0	0	0	1	1	…	1
Cumulatieve som N_1	1	1	2	2	2	2	3	4	…	16
De verhouding N_1/N	1	1/2	2/3	2/4	2/5	2/6	3/7	4/8	…	16/30
De numerieke waarde	1.00	0.50	0.67	0.50	0.40	0.33	0.43	0.50	…	0.53

   Deze rijen hebben de volgende betekenis: (1) het aantal keer N dat er tot nu toe is opgegooid, (2) de uitkomst van de N^e keer opgooien, (3) de som van de eerste N uitkomsten, (4) de schatting van de waarschijnlijkheid van het krijgen van de uitkomst 1 op basis van de eerste N keer opgooien, (5) de numerieke waarde van deze schatting. Je kan gerust *Excel* of een vergelijkbaar programma gebruiken om deze tabel te maken.

[3] Maar hoe goed is deze schatting? Er kan worden aangetoond dat de fout gemiddeld van de orde van $1/\sqrt{N}$ is, en dus snel naar nul gaat als we het experiment vele malen herhalen.

4. Maak een grafiek van de laatste rij van je tabel als functie van het aantal keer dat er opgegooid wordt N.

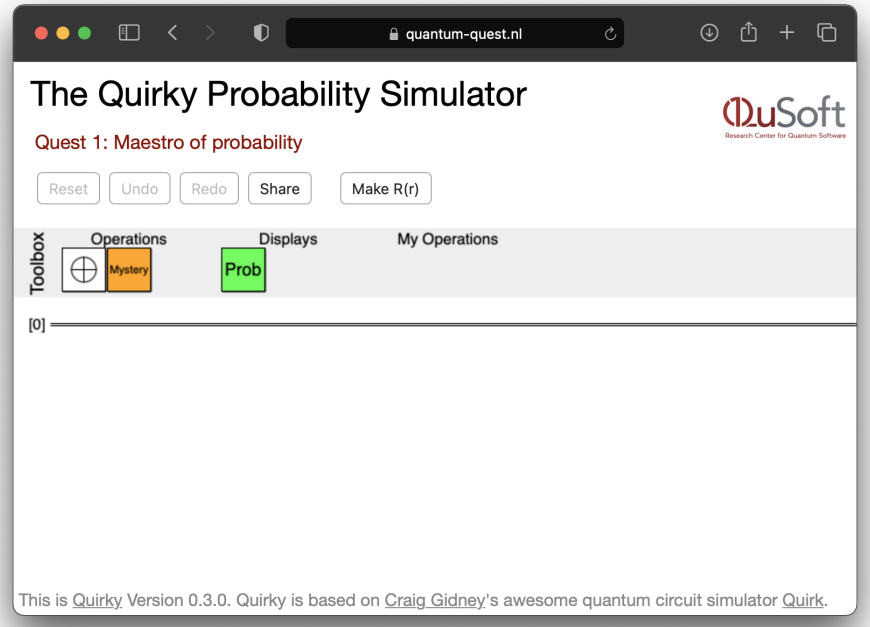
1.4 De QUIRKY simulator

De wetten van de kansberekening kunnen soms contra-intuïtief zijn. Gelukkig kan je altijd proberen het proces te simuleren met de computer die je thuis hebt (of die in je zak). [4]

In deze cursus gebruiken we een simulator genaamd QUIRKY. QUIRKY is het kleine broertje van Quirk, een simulator met veel meer mogelijkheden, ontwikkeld door Craig Gidney bij Google. Omdat Craig zijn code heeft vrijgegeven onder een 'open source' licentie, hebben we zijn simulator kunnen aanpassen voor ons gebruik in deze cursus. Een van de beste eigenschappen van QUIRKY is dat het direct in je webbrowser draait - er is geen installatie nodig! Je gaat gewoon naar:

https://www.quantum-quest.org/quirky

en klikt op "Quest 1". Probeer het maar eens – je kunt QUIRKY zelfs op je mobiele telefoon openen! Als je QUIRKY voor het eerst opent, zou het eruit moeten zien als in Fig. 1.3.



Figuur 1.3: De eerste keer dat QUIRKY geopend wordt.

1.4.1 Aan de slag gaan

Laten we de interface van QUIRKY stap voor stap doorlopen. Bovenaan zie je een *menubalk* met wat handige commando's:

[4] Tot op zekere hoogte geldt dit zelfs voor quantumcomputers – maar nu lopen we op de zaken vooruit...

Met de 'Reset'-knop kan je QUIRKY resetten en opnieuw beginnen. De 'Undo'-knop maakt je laatste verandering ongedaan en met de 'Redo'-knop breng je dit weer terug. Het is handig om te weten dat je zelfs het resetten van de simulator ongedaan kan maken. Druk op 'Share' om een paar opties te krijgen om je programma te kunnen delen met je vrienden. Later zullen we het hebben over de 'Make $R(r)$'-knop.

Onder het menu zit de *toolbox* met daarin de basisbewerkingen die we tot nu toe geleerd hebben:

Als voorbeeld, het eerste vakje, $\boxed{\oplus}$, is de NOT-bewerking uit §1.2, die de $[0]$ toestand omzet in $[1]$ en vice versa. De andere twee bewerkingen zullen we zometeen bespreken. Gelukkig hoef je dit niet allemaal te onthouden – je hoeft alleen maar je muis over elk vakje te bewegen om de beschrijving ervan te zien.

Onder de toolbox zit het hart van QUIRKY, de *probabilistische bit*:

De dubbele lijn of 'draad' komt overeen met een bit, die in de toestand $[0]$ is ingesteld. Je kan bewerkingen eenvoudig toevoegen door ze vanuit de toolbox op de draad te slepen. Probeer nu eens de volgende eenvoudige berekening te maken in QUIRKY:[5]

Hoe kunnen we het resultaat van zo'n berekening weergeven? Omdat we in het algemeen met waarschijnlijkheden werken, willen we een manier om *waarschijnlijkheden weer te geven*. Dit wordt gedaan door het groene vakje met de tekst Prob in de 'Display' sectie van de toolbox. Laten we dit toevoegen aan onze berekening en kijken wat er gebeurt:

Het lijkt erop dat het meetresultaat 100% van de gevallen 'één' zal zijn (beweeg de muis over het vakje om ons vermoeden te bevestigen). Dit is natuurlijk precies wat we zouden verwachten. De begintoestand $[0]$ wordt door de NOT-bewerking omgezet in een $[1]$ toestand, zodat de uitkomst altijd 'één' zal zijn volgens de meetregels van Vgl. (1.18).

Oefenopgave 1.8: Een bewerking weghalen

In QUIRKY kan je ook operaties weghalen door ze gewoon weg te slepen van de draad en terug te zetten in de toolbox. Haal de NOT-bewerking uit de berekening, en controleer of het meten van de bit nu met zekerheid 'nul' geeft.

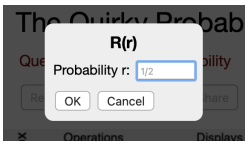
1.4.2 Je eigen bewerkingen maken

Tot nu toe hebben we alleen de $[0]$ en $[1]$ toestanden kunnen maken met QUIRKY. Om een interessantere kansverdeling te maken kunnen we de resetbewerking $R(r)$ van §1.2.2 gebruiken.

[5] Als je een digitale versie van deze tekst leest, kan je op een van de afbeeldingen klikken om QUIRKY in je browser te openen. De operaties in de afbeelding staan er dan automatisch al in! Als dit niet werkt, ga dan zelf naar https://www.quantum-quest.org/quirky.

Aangezien er eindeloos veel van deze bewerkingen zijn (één voor elke gekozen r), kunnen we ze niet allemaal aan de toolbox toevoegen. In plaats daarvan kan je je eigen resetbewerkingen aan de toolbox toevoegen!

Laten we oefenen door een operatie toe te voegen die reset met waarschijnlijkheid $r = \frac{1}{2} = 50\%$. Klik om te beginnen op 'Make $R(r)$' in de menubalk. Er verschijnt een nieuw venster waarin je een waarde voor r kunt invoeren:



Voer `1/2` in, en bevestig door op de knop te drukken. Gefeliciteerd! Je bent erin geslaagd de bewerking $R(1/2)$ toe te voegen aan de toolbox, die er nu zo uitziet:

Om onze nieuwe bewerking te testen, maken we de volgende berekening in QUIRKY:

Laten we even kijken of deze uitkomst logisch is. We begonnen met de toestand $[0] = \binom{1}{0}$. De NOT-bewerking zet de bit om in de toestand $[1] = \binom{0}{1}$. Volgens Vgl. (1.13) reset de operatie $R(1/2)$ een bit in toestand $[1]$ met waarschijnlijkheid $\frac{1}{2}$, dus verandert de toestand naar

$$R(1/2)\,[1] = \frac{1}{2}\,[0] + \frac{1}{2}\,[1] = \binom{1/2}{1/2} = \binom{50\%}{50\%}.$$

Dit is precies waar QUIRKY op uitkwam.

In de volgende opdracht ga je QUIRKY gebruiken om een ingewikkelder experiment uit te voeren.

Huiswerkopdracht 1.5: Twee keer resetten

1. Maak de volgende reeks operaties met QUIRKY: Stel eerst de toestand $[1]$ op, reset deze dan met waarschijnlijkheid $\frac{1}{4}$ en reset vervolgens weer met waarschijnlijkheid $\frac{2}{3}$. Gebruik de 'Probability Display' in QUIRKY om de waarschijnlijkheid van de meetuitkomsten te bepalen.

2. Laat zien dat het antwoord van QUIRKY klopt.

1.4.3 Een mysterieuze bewerking

We hebben de mysterieuze oranje doos nog niet bekeken. Laten we deze bewerking M noemen. Hoe kunnen we erachter komen wat er in die doos gebeurt? Laten we als eerste stap bekijken hoe we $M\,[0]$ kunnen bepalen, oftewel het resultaat van het toepassen van de mysterieuze bewerking M op een bit in toestand $[0]$. In QUIRKY komt dit overeen met de volgende opstelling:

Hoe kunnen we nu $M\,[0]$ aflezen? Het is nu een goed idee om ons eraan te herinneren dat wanneer willekeurige bits in de natuur voorkomen, we ze *niet* zomaar kunnen bekijken en hun waarschijnlijkheid aflezen. In plaats daarvan moeten we, zoals uitgelegd in §1.3, veel metingen doen (bijv. vele keren een muntje opgooien) en uit de uitkomsten de waarschijnlijkheid schatten. Het voordeel van een simulator als QUIRKY is dat we niet aan deze regels vastzitten – we kunnen de 'Probability Display' gebruiken om de toestand te bepalen:

Zo kunnen we zien dat

$$M\,[0] = 0.2\,[0] + 0.8\,[1].$$

Nu is het jouw beurt!

Huiswerkopdracht 1.6: Tijd voor een mysterie

1. Bepaal de toestand $M\,[1]$.
2. Zijn $M\,[0]$ en $M\,[1]$ genoeg om de willekeurige bewerking M volledig te bepalen? Zo ja, schrijf een formule op voor $M\left(\begin{smallmatrix}1/2\\1/2\end{smallmatrix}\right)$ en controleer die in QUIRKY. Zo nee, leg uit waarom niet.

In de komende weken zullen we de sprong van gewone bits naar quantumbits maken, en leren hoe we hiermee op steeds slimmere manieren kunnen rekenen. QUIRKY zal ons trouwe gereedschap zijn, en zal nieuwe functies krijgen naarmate we verder komen. Je wordt van harte aangemoedigd om QUIRKY te gebruiken om de stof die je leert te onderzoeken, en om je te helpen bij het oplossen van je huiswerkopdrachten.

1.5 Oplossingen van de oefenopgaven

Oplossing van Oefenopgave 1.1

1. Omdat één van de twee uitkomsten moet plaatsvinden, moeten de twee waarschijnlijkheden bij elkaar opgeteld 1 zijn. Dit betekent dus dat $p_0 + p_1 = 1$. Als je dit schrijft als $p_1 = 1 - p_0$ krijg je de vergelijking van een lijn met helling min één.
2. Als de lijn verder zou gaan, zou één van de waarschijnlijkheden negatief worden. Omdat waarschijnlijkheden niet negatief kunnen zijn, geldt dat $p_0 \geq 0$ en $p_1 \geq 0$, dus het lijnstuk moet op de assen eindigen.
3. Dat is het middelpunt, dus waar $p_0 = p_1 = \frac{1}{2}$.

Oplossing van Oefenopgave 1.2

1. De teller kan op 60 verschillende getallen staan. De waarschijnlijkheid van een van die waardes is $\frac{1}{60}$.
2. Het laatste getal kan 10 verschillende waardes hebben. De waarschijnlijkheid van het zien van een van die waardes is $\frac{1}{10}$.
3. Het eerste getal kan maar 6 verschillende waardes hebben. De waarschijnlijkheid om daar één van te zien is dus $\frac{1}{6}$.
4. Als je alleen het eerste getal ziet, kan het laatste getal met een gelijke waarschijnlijkheid elk van de 10 mogelijke waardes hebben. En als je alleen het laatste getal ziet, kan het eerste getal met gelijke waarschijnlijkheid elk van de 6 mogelijke waardes hebben. De waardes van de twee getallen zijn dus onafhankelijk. Je kan controleren dat de waarschijnlijkheid om 00 te krijgen inderdaad $\frac{1}{60}$ is door de waarschijnlijkheid dat elk getal nul is te vermenigvuldigen:

$$\frac{1}{6} \cdot \frac{1}{10} = \frac{1}{60}.$$

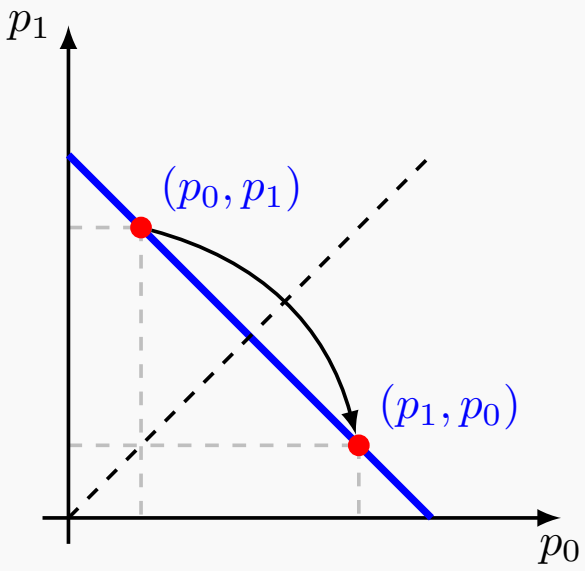
Oplossing van Oefenopgave 1.3

Er zijn zes mogelijke combinaties waarbij beide getallen hetzelfde zijn (van 00 tot 55). Elk van deze gevallen komt voor met een waarschijnlijkheid van $\frac{1}{60}$. We kunnen deze uitkomsten combineren tot één uitkomst waarvan de waarschijnlijkheid gelijk is aan de som van de waarschijnlijkheden van de zes aparte uitkomsten:

$$\underbrace{\frac{1}{60} + \frac{1}{60} + \frac{1}{60} + \frac{1}{60} + \frac{1}{60} + \frac{1}{60}}_{\text{6 termen}} = \frac{6}{60} = \frac{1}{10}.$$

Oplossing van Oefenopgave 1.4

1. Merk op dat volgens Vgl. (1.10) het punt (p_0, p_1) wordt afgebeeld op (p_1, p_0). Met andere woorden, de twee coördinaten van het punt worden omgewisseld. Hier is een voorbeeld van hoe dat eruit ziet:



   Je kan de *NOT*-bewerking dus voor je zien als een spiegeling om de stippellijn die precies tussen het midden van de assen loopt.

2. Volgens Vgl. (1.2) komen de twee eindpunten $(1, 0)$ en $(0, 1)$ van het lijnstuk overeen met de twee deterministische toestanden $[0]$ en $[1]$. In Vgl. (1.9) hebben we gezien dat de NOT-bewerking deze omwisselt.

3. Een punt met coördinaten (p_0, p_1) blijft na de NOT-bewerking hetzelfde als $(p_0, p_1) = (p_1, p_0)$, wat betekent dat $p_0 = p_1$. Omdat $p_0 + p_1 = 1$, betekent dit dat $p_0 = p_1 = 1/2$ wat overeenkomt met het punt $(1/2, 1/2)$. Dit is het enige punt dat op dezelfde plek blijft liggen.

Oplossing van Oefenopgave 1.5

We gebruiken Vgl. (1.9) en (1.11),

$$\text{NOT} \begin{pmatrix} p_0 \\ p_1 \end{pmatrix} = p_0 \, \text{NOT} \begin{pmatrix} 1 \\ 0 \end{pmatrix} + p_1 \, \text{NOT} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = p_0 \begin{pmatrix} 0 \\ 1 \end{pmatrix} + p_1 \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} p_1 \\ p_0 \end{pmatrix},$$

wat hetzelfde is als Vgl. (1.10).

Oplossing van Oefenopgave 1.6

1. We gebruiken de definitie van $F(f)$ uit Vgl. (1.14),

$$F(f)\,[0] = (1-f)\,\left(\begin{smallmatrix}1\\0\end{smallmatrix}\right) + f\,\left(\begin{smallmatrix}0\\1\end{smallmatrix}\right) = \left(\begin{smallmatrix}1-f\\f\end{smallmatrix}\right),$$
$$F(f)\,[1] = f\,\left(\begin{smallmatrix}1\\0\end{smallmatrix}\right) + (1-f)\,\left(\begin{smallmatrix}0\\1\end{smallmatrix}\right) = \left(\begin{smallmatrix}f\\1-f\end{smallmatrix}\right).$$

2. $F(f)$ flipt de bit met zekerheid als $f = 1$, dus NOT $= F(1)$. Om een willekeurige toestand $\left(\begin{smallmatrix}p\\1-p\end{smallmatrix}\right)$ te maken vanuit $[0]$ moeten we $f = 1-p$ kiezen. Uit de eerste vergelijking hierboven volgt dan dat

$$F(1-p)\,[0] = \begin{pmatrix}1-(1-p)\\1-p\end{pmatrix} = \begin{pmatrix}p\\1-p\end{pmatrix}.$$

3. Gebruik Vgl. (1.11),

$$F(f)\,\begin{pmatrix}p_0\\p_1\end{pmatrix} = p_0\begin{pmatrix}1-f\\f\end{pmatrix} + p_1\begin{pmatrix}f\\1-f\end{pmatrix} = \begin{pmatrix}p_0(1-f)+p_1 f\\p_0 f + p_1(1-f)\end{pmatrix}.$$

4. Als je $f = 1/2$ in de vorige vergelijking invult krijg je

$$F(1/2)\,\begin{pmatrix}p_0\\p_1\end{pmatrix} = \begin{pmatrix}p_0/2+p_1/2\\p_0/2+p_1/2\end{pmatrix} = \frac{1}{2}\begin{pmatrix}p_0+p_1\\p_0+p_1\end{pmatrix} = \frac{1}{2}\begin{pmatrix}1\\1\end{pmatrix}.$$

Oplossing van Oefenopgave 1.7

We bekijken twee gevallen:

- $\frac{1}{2} \le f \le 1$: In dit geval geldt $0 \le \frac{1-f}{f} \le 1$. We stellen dat de flipbewerking $F(f)$ kan worden opgebouwd door eerst $R(\frac{1-f}{f})$ toe te passen, dan NOT, en tenslotte $R(1-f)$. Als je dit uitwerkt zie je inderdaad:

$$R(1-f)\,\text{NOT}\,R(\tfrac{1-f}{f})\,[0] = R(1-f)\,\text{NOT}\,[0] = R(1-f)\,[1] = (1-f)\,[0] + f\,[1]$$

en

$$\begin{aligned} R(1-f)\,\text{NOT}\,R(\tfrac{1-f}{f})\,[1] &= R(1-f)\,\text{NOT}\left(\tfrac{1-f}{f}\,[0] + (1-\tfrac{1-f}{f})\,[1]\right)\\ &= R(1-f)\left((1-\tfrac{1-f}{f})\,[0] + \tfrac{1-f}{f}\,[1]\right)\\ &= (1-\tfrac{1-f}{f})\,[0] + \tfrac{1-f}{f}\,((1-f)\,[0] + f\,[1])\\ &= f\,[0] + (1-f)\,[1]. \end{aligned}$$

- $0 \le f \le \frac{1}{2}$: Dit geval kunnen we omschrijven naar het eerste geval, aangezien $F(f)$ hetzelfde is als eerst $F(1-f)$ toepassen en daarna NOT.

Quest 2: Overwin de qubit

Nu dat je waarschijnlijkheden en probabilistische bits onder de knie hebt, ben je klaar om te leren over quantumbits. Quantumbits lijken heel erg op probabilistische bits – je hoeft alleen maar waarschijnlijkheden te vervangen door quantumamplitudes. Deze week ga je leren over de toestanden van een quantumbit, welke bewerkingen zijn toegestaan, en hoe je hier informatie uit kunt halen. Ook krijg je de kans om een nieuwe *quantum* versie van QUIRKY uit te proberen.

2.1 Quantumbits

Bits zijn de fundamentele eenheid van informatie in onze huidige computers. Om een bit te maken, heb je een fysiek object nodig dat zich in één van twee duidelijk te onderscheiden toestanden kan bevinden, zoals een muntje met twee zijden of een condensator die elektrische lading kan opslaan op twee verschillende spanningsniveaus. [6] Het gedrag van zulke objecten (en dus de bits die ze coderen) kan worden beschreven door natuurkundige theorieën zoals mechanica (voor muntjes) of elektromagnetisme (voor condensatoren).

Maar voor heel kleine[7] objecten gelden deze theorieën niet meer en moet je een meer fundamentele theorie gebruiken, die quantummechanica heet. Een elektron heeft bijvoorbeeld een bepaalde eigenschap, spin, die (net als een muntje) één van twee waarden kan aannemen - omhoog of omlaag - en dus gebruikt kan worden om een bit op te slaan. Maar in tegenstelling tot een muntje kan de spin van een elektron zich niet in maar één van deze twee toestanden bevinden, maar ook in een "superpositie" van beide! Intuïtief gezien kan je dit vergelijken met een probabilistische bit, die zich ook in een tussentoestand tussen 0 en 1 kan bevinden.

Er is wel een subtiel verschil tussen waarschijnlijkheden en 'superposities' (zie §2.6.1 over interferentie). Zoals we zullen zien, leiden de wetten van de quantummechanica tot een veel fundamenteler begrip van informatie dan een bit - een quantumbit of qubit. Om de gewone bits te onderscheiden van hun exotischere quantumvrienden, zullen we de gewone bits klassiek noemen.

We zullen quantumbits beschrijven aan de hand van een eenvoudig wiskundig model en ons verder geen zorgen maken over hoe we hun vreemde gedrag moeten interpreteren, in plaats daarvan stellen we de vraag: "Waarvoor kunnen ze worden gebruikt?". Ook zullen we ons geen zorgen maken over hoe ze in de praktijk geïmplementeerd kunnen worden of wat voor soort fysieke objecten gebruikt kunnen worden om ze op te slaan. Als je hier toch nieuwsgierig naar bent, bespreken we in §2.6.2 kort hoe de polarisatie van licht gebruikt kan worden om een qubit te representeren.

2.1.1 Waarschijnlijkheden versus amplitudes

Quantumbits lijken erg op probabilistische bits. Er zijn maar twee belangrijke verschillen:

1. waarschijnlijkheden zijn vervangen door amplitudes (die ook negatief kunnen zijn),
2. amplitudes worden gekwadrateerd bij een meting (waarschijnlijkheden worden dit niet).

We zullen deze verschillen later in meer detail uitleggen, maar laten we eerst de mogelijke toestanden van een qubit beschrijven. Denk terug aan hoe we de twee zijden van een muntje hebben gebruikt om de twee mogelijke deterministische toestanden van een probabilistisch bit aan te duiden (zie Fig. 1.1). Bij quantum computing worden deze twee toestanden aangeduid

[6] Dit is in feite de manier waarop bits worden weergegeven in je computer, mobiele telefoon, enz.

[7] Met 'heel klein' bedoelen we *echt heel klein*! Als je elektronen naast elkaar in een rij zou zetten, is het aantal elektronen dat je nodig hebt om een lengte van 1 cm te bereiken vergelijkbaar met het aantal pagina's dat je op elkaar moet leggen om de maan te bereiken.

met $|0\rangle$ en $|1\rangle$ om ze te onderscheiden van de klassieke bits $[0]$ en $[1]$. Net als bij probabilistische bits, is een algemene qubit-toestand $|\psi\rangle$ dan een lineaire combinatie oftewel superpositie van deze twee deterministische toestanden:

$$|\psi\rangle = \psi_0 |0\rangle + \psi_1 |1\rangle . \tag{2.1}$$

Hier is de Griekse letter ψ (spreek uit: "psi") de naam van de toestand (net zoals we meestal p gebruiken voor een probabilistische bit). De haakjes $|\cdot\rangle$ vormen een zogenaamde "ket" om een quantumtoestand aan te duiden. Ter vergelijking: in Vgl. (1.3) is gegeven dat een algemene probabilistische bit p geschreven kan worden als

$$p = p_0[0] + p_1[1]. \tag{2.2}$$

Merk op dat Vgl. (2.1) er precies hetzelfde uitziet, behalve dat de waarschijnlijkheden p_0 en p_1 zijn vervangen door de amplitudes ψ_0 en ψ_1, en de klassieke notatie $[0]$ en $[1]$ is vervangen door de quantumnotatie $|0\rangle$ en $|1\rangle$! Maar er is wel één heel belangrijk verschil: terwijl de kansen in Vgl. (2.2) moeten voldoen aan

$$p_0, p_1 \geq 0 \qquad \text{en} \qquad p_0 + p_1 = 1, \tag{2.3}$$

moeten de amplitudes voldoen aan

$$\psi_0^2 + \psi_1^2 = 1. \tag{2.4}$$

Dit betekent dat $\psi_0^2 \leq 1$ en $\psi_1^2 \leq 1$, en dus dat $\psi_0, \psi_1 \in [-1, 1]$. De beperkingen op de waarschijnlijkheden van 2.3 leiden er daarintegen tot dat $p_0, p_1 \in [0, 1]$. Het cruciale verschil is dat amplitudes ook negatief mogen zijn, terwijl waarschijnlijkheden dat niet mogen![8]

Net als bij probabilistische bits is het handig om qubit-toestanden met vectoren weer te geven. Op precies dezelfde manier als Vgl. (1.2) geven we de deterministische qubit-toestanden $|0\rangle$ en $|1\rangle$ weer met de twee basisvectoren:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

Een algemene quantumtoestand als in Vgl. (2.1) kan je dus opschrijven als

$$|\psi\rangle = \psi_0 \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \psi_1 \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \psi_0 \\ \psi_1 \end{pmatrix}.$$

2.1.2 Een qubit als een cirkel

Merk op dat de vergelijking Vgl. (2.4) voor qubit-amplitudes lijkt op de vergelijking $x^2 + y^2 = 1$ van een cirkel. Laten we deze overeenkomst verder uitwerken, omdat dat ons zal helpen quantumbits te visualiseren en op een meer intuïtief niveau te begrijpen.

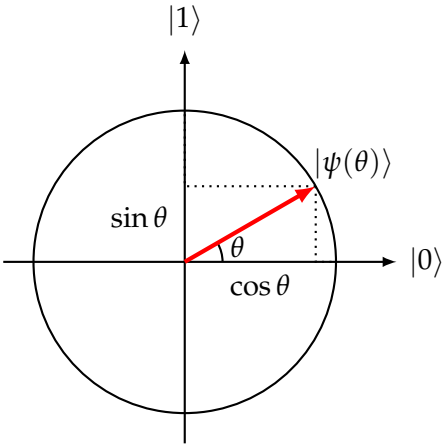
Een handige manier om de qubit-amplitudes te parametriseren is als

$$\psi_0 = \cos\theta, \qquad \psi_1 = \sin\theta$$

met een hoek $\theta \in [0, 2\pi)$. In de praktijk zal het vaak handig zijn als we toestaan dat de hoek θ een arbitrair reëel getal is (wat prima is zolang we in gedachten houden dat twee hoeken die 2π verschillen dezelfde amplitudes geven). Aangezien $\cos^2\theta + \sin^2\theta = 1$, voldoen we automatisch aan Vgl. (2.4). Met deze keuze ziet een algemene qubit-toestand er als volgt uit:

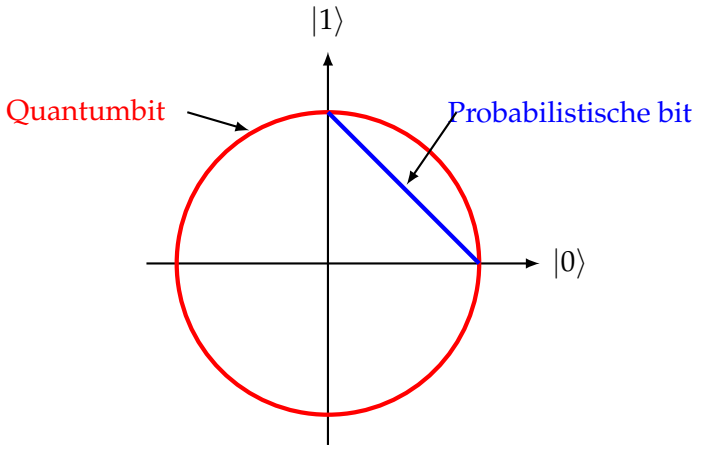
$$|\psi(\theta)\rangle = \cos\theta |0\rangle + \sin\theta |1\rangle = \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix}. \tag{2.5}$$

[8] In feite mogen amplitudes zelfs zogenaamde *complexe getallen* zijn. We hebben ze niet nodig in deze cursus, maar als je hier meer over wilt leren kan je gerust op het internet rondzoeken.



Figuur 2.1: De qubit-toestand $|\psi(\theta)\rangle$ als een punt op de eenheidscirkel.

Je kan dit voorstellen als een eenheidsvector in twee dimensies die begint bij de oorsprong en een hoek θ maakt met de horizontale $|0\rangle$-as (zie Fig. 2.1). Als voorbeeld geldt dat $|0\rangle = |\psi(0)\rangle$ en $|1\rangle = |\psi(\frac{\pi}{2})\rangle$. De verzameling van alle qubit-toestanden komt dan overeen met een eenheidscirkel met het middelpunt in de oorsprong. Bij probabilistische bits liggen de mogelijke toestanden daarentegen op een lijnstuk dat de punten $\binom{1}{0}$ en $\binom{0}{1}$ op de assen verbindt, zoals we in Fig. 1.2 hebben gezien. In Fig. 2.2 worden de twee verzamelingen vergeleken.



Figuur 2.2: De toestandruimtes van een probabilistische bit (blauw) en een quantumbit (rood).

Oefenopgave 2.1: Toestanden op de cirkel

Bekijk de volgende twee toestanden van een qubit:

$$|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}, \qquad |-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}.$$

Waar liggen deze twee toestanden op de cirkel? Welke hoeken θ horen daarbij?

2.2 Het meten van een quantumbit

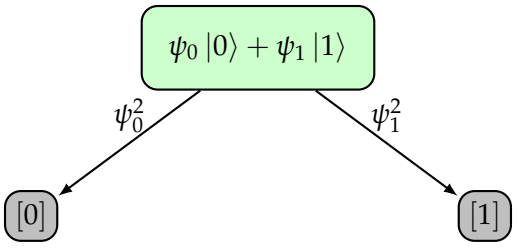
We weten nu dat elke qubit-toestand de vorm $|\psi(\theta)\rangle$ heeft. Stel nu dat je een toestand van $|\psi(\theta)\rangle$ hebt en je wilt weten wat de waarde van θ is. Helaas kan je dat niet te weten komen volgens de quantummechanica! Dit lijkt een groot probleem – waar is een quantumcomputer goed voor

als je niet eens achter het antwoord kan komen? Nou, niet zo snel! Volgens Vgl. (1.18) geldt hetzelfde voor probabilistische bits: als je een probabilistische bit met de verdeling $\binom{p_0}{p_1}$ meet, kan je ook niet p_0 of p_1 vaststellen. Je krijgt alleen een enkele bit van informatie: 0 met kans p_0 en 1 met kans p_1.

De quantum-meting lijkt hier veel op en wordt beschreven door de zogenaamde Born-regel. Als je een qubit in de toestand $\binom{\psi_0}{\psi_1} = \psi_0 |0\rangle + \psi_1 |1\rangle$ hebt en deze meet, krijg je ook maar één bit: je krijgt 0 of 1 met waarschijnlijkheid

$$p_0 = \psi_0^2, \qquad p_1 = \psi_1^2. \tag{2.6}$$

Het kwadraat is misschien verrassend, maar merk op dat $p_0 + p_1 = \psi_0^2 + \psi_1^2 = 1$. Het kwadraat is precies wat er voor zorgt dat $\binom{p_0}{p_1}$ een geldige kansverdeling is, dus de bovenstaande regel is best logisch! Na de meting is de qubit verdwenen en blijft er maar één bit over die de uitkomst van de meting bevat. Met andere woorden, het meetproces zet een qubit om in een gewone bit waarvan de waarde probabilistisch wordt bepaald door Vgl. (2.6):

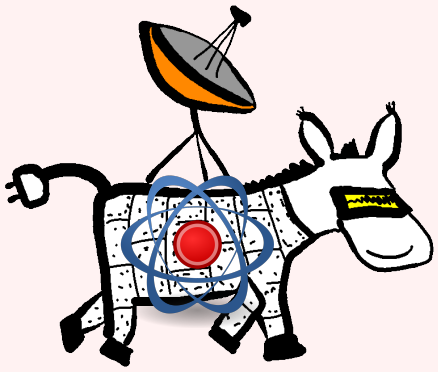


(2.7)

Zoals je kunt zien in Vgl. (1.18) en (2.7), lijken de meetregels voor probabilistische bits en qubits sterk op elkaar. In beide gevallen is de oorspronkelijke toestand verdwenen en blijft alleen een enkele bit over, waarvan de waarde probabilistisch afhangt van de oorspronkelijke toestand die je gemeten hebt. (Als je de toestand meer dan één keer meet, krijg je altijd dezelfde uitkomst als de eerste keer – herhaalde metingen geven dus geen extra informatie over wat de oorspronkelijke toestand was). Het enige verschil is dat je voor qubits de amplitudes moet kwadrateren om de waarschijnlijkheden te krijgen, zoals in Vgl. (2.6), terwijl je voor probabilistische bits ze direct krijgt en dus niets hoeft te kwadrateren. Dit lijkt misschien een klein verschil, maar het heeft grote gevolgen voor de toegestane toestanden, omdat de amplitudes van een qubit negatief mogen zijn, terwijl de waarschijnlijkheden van een probabilistische bit altijd positief zijn (zie Fig. 2.2).

Er is eigenlijk een ander, wat subtieler verschil. Namelijk dat niemand de uitkomst van een quantum-meting van tevoren kan voorspellen. Dit is subtiel omdat je zou denken dat hetzelfde ook moet gelden voor probabilistische bits. Waarin zit het verschil? In het kort is het antwoord dat probabilistische bits willekeurig lijken door ons gebrek aan kennis over hun toestand, terwijl quantumbits willekeurig zijn zelfs als we alles weten over hun toestand. Stel bijvoorbeeld dat je vriend een zuiver muntje opgooit en het muntje meteen bedekt zodra het landt. Normaal gesproken zou je de toestand van zo'n muntje beschrijven als uniform willekeurig, zie Vgl. (1.15). Maar als je de munt filmt met een hogesnelheidscamera, kan je op basis van je beelden misschien nauwkeurig voorspellen op welke kant hij terechtkomt. In dit opzicht heeft de willekeur van probabilistische bits te maken met onze onwetendheid. Voor quantumbits ontstaat de willekeur daarentegen op een fundamenteler niveau. Ongeacht onze voorkennis is het in het algemeen *onmogelijk* de uitkomst van een quantum-meting perfect te voorspellen. Aan de andere kant betekent dit dat de uitkomsten van quantum-metingen gebruikt kunnen worden als een goede bron van willekeur!

Huiswerkopdracht 2.1: Een willekeurige bit op een quantum manier maken



Het probleem: Alice's robot-ezel heeft weer te weinig stroom en moet zijn weg vinden naar een oplaadstation. Helaas zijn Eve's hackingvaardigheden deze keer verbeterd – ze heeft uitgevonden hoe ze de random-number-generator van de ezel kan hacken en herprogrammeren zodat die elke willekeurige getallenreeks genereert die zij wil! Gelukkig is Alice hiervan op de hoogte omdat Eve er onlangs over heeft lopen opscheppen op een hackerforum. Om Eve's kwaadaardige plan tegen te gaan, heeft Alice besloten een miniatuur quantumcomputer met een enkele qubit in haar robot-ezel te installeren. Door gebruik te maken van de intrinsieke onvoorspelbaarheid van quantummeetresultaten, wil Alice uniform willekeurige bits genereren die Eve niet kan raden.

Vragen: Alice kan elke qubit-toestand $|\psi(\theta)\rangle$ produceren en wil een uniform willekeurige bit genereren door dit te meten.

1. Wat is de kans op het meetresultaat 0 als je een meting van de toestand $|\psi(\theta)\rangle$ uitvoert? Wat is de kans op het meetresultaat 1?

2. Alice wil een hoek θ vinden zodat beide kansen gelijk zijn aan $1/2$. Welke θ moet ze kiezen? (Er kunnen meerdere mogelijkheden zijn!)

2.3 Quantumbits simuleren met QUIRKY

De wetten van quantumcomputing zijn nogal vreemd en de meesten van ons hebben geen quantumcomputer om mee te experimenteren. Gelukkig heeft QUIRKY sinds vorige week nieuwe krachten gekregen en kunnen we nu een *quantum*bit simuleren![9] Ga om te beginnen naar:

https://www.quantum-quest.org/quirky

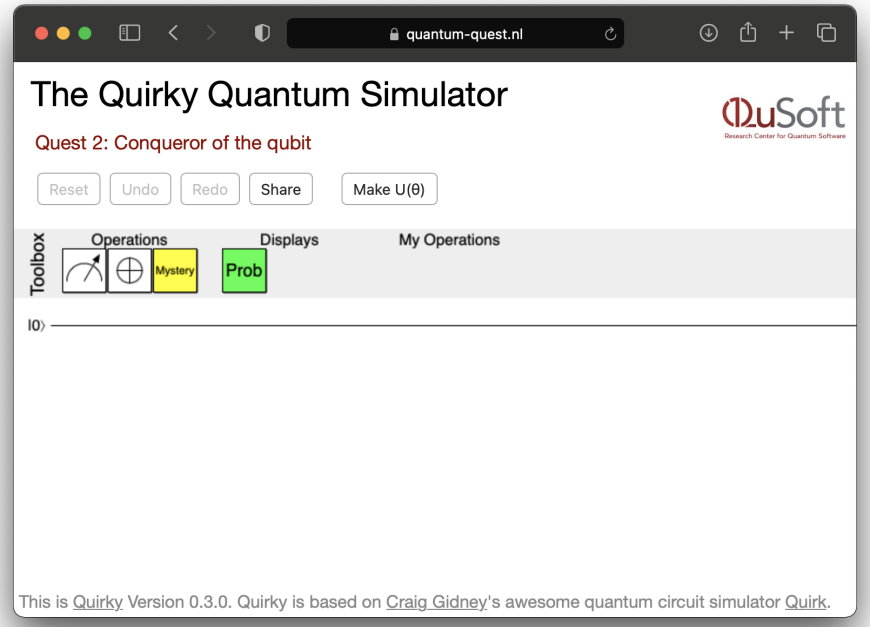
en klik op "Quest 2". Je webbrowser zal er ongeveer hetzelfde uitzien als Fig. 2.3.

Het belangrijkste verschil met vorige week is dat de 'draad' nu overeenkomt met een quantumbit, die in de toestand $|0\rangle$ is ingesteld.

Net als vorige week bevat de *toolbox* bewerkingen die we kunnen toepassen door ze vanuit de toolbox naar de draad te slepen:

Het eerste vakje,  , laat ons een quantumbit meten. Laten we de volgende eenvoudige quantumberekening maken in QUIRKY:

[9] Waarom willen we überhaupt quantumcomputers bouwen als we ze zo mooi kunnen simuleren op al bestaande computers? De reden is dat simulators zoals QUIRKY goed werken als je maar een paar quantumbits hebt, maar snel stuk gaan als het aantal quantumbits toeneemt. Je zal hier veel meer over zien als je aan Quest 4 en 5 werkt.



Figuur 2.3: QUIRKY voor Quest 2

Je zult zien dat de enkele lijn een dubbele lijn is geworden. In QUIRKY verwijzen enkele lijnen naar quantumbits, en dubbele lijnen naar gewone of 'klassieke' bits. We weten immers uit §2.2 dat als we een quantumbit meten we een uitkomst krijgen die ofwel nul ofwel één is met bepaalde waarschijnlijkheden, oftewel een probabilistische bit.

Om de waarschijnlijkheid van de uitkomsten te bekijken, kunnen we de 'Probability Display' gebruiken die we al kennen van vorige week. Laten we die toevoegen aan onze berekening en kijken wat er gebeurt:

Het lijkt erop dat het meetresultaat 100% van de tijd 'nul' zal zijn (beweeg met de muis over het vakje om ons vermoeden te bevestigen). Dit is natuurlijk precies wat we verwachten. Als we $|0\rangle$ meten, zal de uitkomst altijd 'nul' zijn volgens de meetregels in Vgl. (2.7).

In het vervolg van dit hoofdstuk zullen we de andere vakjes in de toolbox bespreken.

2.4 Bewerkingen op een quantumbit

Voordat we een toestand meten, willen we er misschien een bewerking op uitvoeren. Maar welke soort bewerkingen kunnen we op een qubit uitvoeren? Bijvoorbeeld, wanneer we onze quantumcomputer opstarten, zal de quantumbit altijd in toestand $|0\rangle$ zitten, dus moeten we een bewerking uitvoeren om een interessante toestand $|\psi(\theta)\rangle$ te creëren. Wat de bewerking ook is, het moet een andere qubit-toestand als uitvoer opleveren. Met andere woorden, het moet de qubit-toestandsruimte op zichzelf afbeelden. Dit betekent dat de bewerking elke willekeurige toestand, wat je kan zien als een punt in de toestandsruimte, afbeeldt op een punt dat ook in de toestandsruimte ligt. Denk terug aan Fig. 2.1, waarin is vastgesteld dat deze toestandsruimte overeenkomt met een cirkel, dus we zoeken naar manieren om de cirkel op zichzelf af te beelden.

Laten we eerst de NOT-bewerking bekijken, die we op precies dezelfde manier kunnen definiëren als in Vgl. (1.9) voor probabilistische bits:

$$\text{NOT}\,|0\rangle = |1\rangle\,, \qquad \text{NOT}\,|1\rangle = |0\rangle\,.$$

Hoe kunnen we NOT uitbreiden naar willekeurige qubit-toestanden? Net zoals we deden voor probabilistische bits in §1.2.1, zullen we het concept van lineariteit gebruiken. Als een operatie M gedefinieerd is op $|0\rangle$ en $|1\rangle$ dan kunnen we deze op een willekeurige qubit-toestand definiëren door

$$M\Big(\psi_0\,|0\rangle + \psi_1\,|1\rangle\Big) = \psi_0\,M\,|0\rangle + \psi_1\,M\,|1\rangle\,. \tag{2.8}$$

We kunnen Vgl. (2.8) ook uitschrijven door de vectornotatie te gebruiken:

$$M\begin{pmatrix}\psi_0\\ \psi_1\end{pmatrix} = M\left(\psi_0\begin{pmatrix}1\\0\end{pmatrix} + \psi_1\begin{pmatrix}0\\1\end{pmatrix}\right) = \psi_0\,M\begin{pmatrix}1\\0\end{pmatrix} + \psi_1\,M\begin{pmatrix}0\\1\end{pmatrix}. \tag{2.9}$$

Zoals al eerder is opgemerkt, wordt in de wiskunde een bewerking M die aan deze voorwaarde voldoet lineair genoemd, en het uitbreiden van een bewerking op deze manier heet uitbreiden "door lineariteit". Het belangrijkste punt is dat als M lineair is en we weten hoe het werkt op $|0\rangle$ en op $|1\rangle$, we kunnen afleiden hoe het werkt op willekeurige qubit-toestanden!

In Vgl. (2.8) hebben we alleen de vectoren $|0\rangle$ en $|1\rangle$ beschouwd. In het algemeen geldt ook dat

$$M\Big(a\,|\psi\rangle + b\,|\phi\rangle\Big) = a\,M\,|\psi\rangle + b\,M\,|\phi\rangle \tag{2.10}$$

voor arbitraire vectoren $|\psi\rangle$, $|\phi\rangle$ en getallen a, b. Zie je hoe (2.10) volgt uit (2.8)?

De wetten van de quantummechanica garanderen dat elke lineaire bewerking M een mogelijke qubit-bewerking is – zolang het de hele qubit-toestandsruimte op zichzelf afbeeldt! Hiermee bedoelen we dat elke qubit-toestand (een punt op de cirkel) wordt afgebeeld op een qubit-toestand (een punt op de cirkel).

In het geval van de NOT-bewerking is het resultaat van de uitbreiding door lineariteit

$$\text{NOT}\Big(\psi_0\,|0\rangle + \psi_1\,|1\rangle\Big) = \psi_0\,|1\rangle + \psi_1\,|0\rangle\,, \quad \text{of} \quad \text{NOT}\begin{pmatrix}\psi_0\\ \psi_1\end{pmatrix} = \begin{pmatrix}\psi_1\\ \psi_0\end{pmatrix}. \tag{2.11}$$

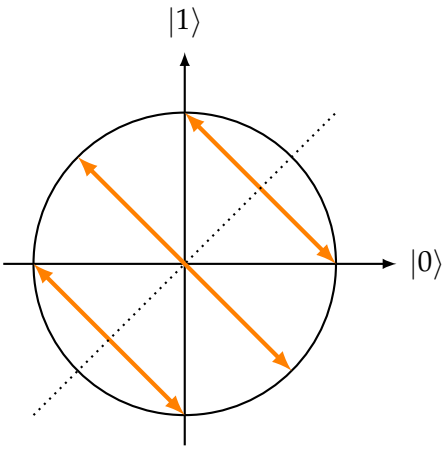
Merk op dat Vgl. (2.8), (2.9) en (2.11) er precies zo uitzien als Vgl. (1.10) en (1.12) – behalve dat nu ψ_0 en ψ_1 ook negatief kunnen zijn. Vertaald naar Fig. 2.2 komt de NOT-bewerking neer op een *spiegeling* rond de 45-graden-as (dit geldt ook voor probabilistische bits). Dit wordt weergegeven in Fig. 2.4. Het is hieruit duidelijk dat *NOT* de qubit-toestandsruimte (cirkel) op zichzelf afbeeldt. De NOT-bewerking is dus een geldige bewerking op een quantumbit.

In QUIRKY ziet de *NOT*-bewerking op qubits er net zo uit als de *NOT*-bewerking op bits, namelijk $\boxed{\oplus}$. Probeer nu de volgende quantumberekening te maken:

Het lijkt erop dat het meetresultaat 100% van de tijd *'één'* zal zijn. De oorspronkelijke $|0\rangle$ wordt immers door de NOT-bewerking omgezet in een $|1\rangle$ toestand, zodat de uitkomst altijd 'één' zal zijn volgens de meetregels in Vgl. (2.7).

We kunnen op dezelfde manier qubitbewerkingen bepalen door spiegelingen door andere assen te bekijken. Bijvoorbeeld, de Z-bewerking, gedefinieerd door

$$Z\,|0\rangle = |0\rangle\,, \qquad Z\,|1\rangle = -\,|1\rangle\,, \tag{2.12}$$



Figuur 2.4: De NOT-bewerking op een qubit, zoals gedefinieerd in Vgl. (2.11), komt overeen met een spiegeling om de as van 45 graden (of $\pi/4$) (gestippeld).

komt overeen met een spiegeling om de horizontale $|0\rangle$-as. Als we Z uitbreiden door lineariteit werkt het namelijk op een willekeurige qubit-toestand als

$$Z\begin{pmatrix}\psi_0\\ \psi_1\end{pmatrix} = \begin{pmatrix}\psi_0\\ -\psi_1\end{pmatrix},$$

wat zeker qubit-toestanden op qubit-toestanden afbeeldt.

Huiswerkopdracht 2.2: Z-bewerking

We beschouwen de volgende twee toestanden van een qubit:

$$|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}, \qquad |-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}.$$

1. Bereken $Z|+\rangle$ en $Z|-\rangle$.
2. Geef de Z-bewerking grafisch weer op de cirkel, zoals in Fig. 2.4.

Oefenopgave 2.2: Lineariteit is niet genoeg (optioneel)

Beschouw de bewerking MAD, die je krijgt door $MAD|0\rangle = |0\rangle$ en $MAD|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ lineair uit te breiden. Zoek een toestand $|\psi\rangle$ zo dat MAD $|\psi\rangle$ geen geldige qubit-toestand is. MAD is dus *geen* geldige bewerking op qubits!

2.4.1 Rotaties

Tot nu toe hebben we alleen geleerd hoe we de $|0\rangle$ en $|1\rangle$ toestanden kunnen maken met QUIRKY. Quantumcomputing zou niet heel leuk zijn als dat onze enige opties waren! Om meer interessante toestanden te kunnen maken, moeten we andere quantumbewerkingen bedenken.

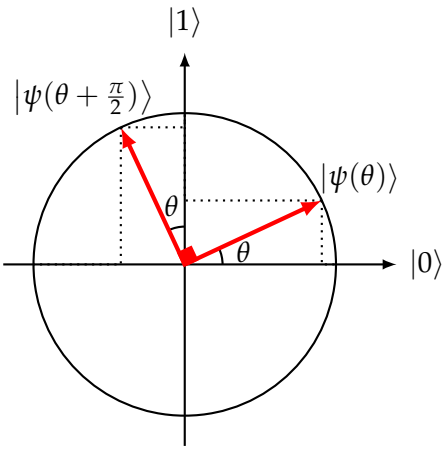
Een logische bewerking is om de cirkel met een bepaalde hoek te *roteren*. Laten we de rotatie met een hoek θ aanduiden met $U(\theta)$. Je kan altijd aannemen dat de hoek in $[0, 2\pi)$ ligt. Aangezien $|0\rangle = |\psi(0)\rangle$ en $|1\rangle = |\psi(\frac{\pi}{2})\rangle$, werkt deze bewerking als volgt op de basisvectoren (zie Fig. 2.5):

$$U(\theta)|0\rangle = |\psi(\theta)\rangle, \qquad U(\theta)|1\rangle = |\psi(\theta + \tfrac{\pi}{2})\rangle. \tag{2.13}$$

We kunnen dit ook expliciet uitschrijven in vectornotatie:

$$U(\theta)\begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}\cos\theta\\ \sin\theta\end{pmatrix}, \qquad U(\theta)\begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}-\sin\theta\\ \cos\theta\end{pmatrix}, \tag{2.14}$$

waarbij we hebben gebruikt dat $\cos(\theta + \frac{\pi}{2}) = -\sin\theta$ and $\sin(\theta + \frac{\pi}{2}) = \cos\theta$.



Figuur 2.5: De toestanden $|0\rangle$ en $|1\rangle$ geroteerd met een hoek θ, zie Vgl. (2.13).

Net als hiervoor zullen we lineariteit gebruiken om $U(\theta)$ uit te breiden van de basisvectoren naar arbitraire qubit-toestanden. In de volgende opgave zul je laten zien dat de bewerking $U(\theta)$ daadwerkelijk werkt als een rotatie op qubit-toestanden. Dit betekent concreet dat het qubit-toestanden afbeeldt op qubit-toestanden, dus $U(\theta)$ is een geldige bewerking op qubits!

Oefenopgave 2.3: Qubit rotatie

1. Bereken $U(\alpha)\begin{pmatrix}\psi_0\\ \psi_1\end{pmatrix}$ met behulp van Vgl. (2.8) en (2.13).

2. Ga aan de hand van de definitie van $|\psi(\theta)\rangle$ uit Vgl. (2.5) na dat voor alle hoeken α en β,
$$U(\alpha)\,|\psi(\beta)\rangle = |\psi(\alpha+\beta)\rangle\,. \tag{2.15}$$
Dit betekent dat $U(\theta)$ werkt als een rotatie op een arbitraire qubit-toestand $|\psi(\beta)\rangle$.

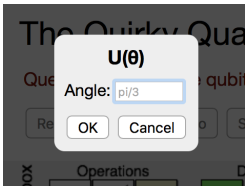
Hint: De goniometrische hoeksom- en hoekverschil-identiteiten kunnen van pas komen:

$$\sin(\alpha \pm \beta) = \sin\alpha\cos\beta \pm \cos\alpha\sin\beta, \qquad \cos(\alpha \pm \beta) = \cos\alpha\cos\beta \mp \sin\alpha\sin\beta. \tag{2.16}$$

Merk op dat een rotatie van 90 graden (d.w.z. $\pi/2$) niet hetzelfde is als een spiegeling. Zowel de *NOT*-bewerking als de rotatie $U(\pi/2)$ beelden $|0\rangle$ af op $|1\rangle$, maar ze hebben een andere werking op $|1\rangle$:

$$\text{NOT}\,|1\rangle = |0\rangle\,, \qquad U(\pi/2)\,|1\rangle = -\,|0\rangle\,.$$

Hoe kunnen we een quantumbit roteren in QUIRKY? Omdat er oneindig veel rotatiebewerkingen $U(\theta)$ zijn, kunnen we ze niet allemaal aan de toolbox toevoegen. In plaats daarvan kan je je eigen rotaties aan de toolbox toevoegen! Laten we om te oefenen een rotatie van 30° toevoegen. Klik om te beginnen op 'Make $U(\theta)$' in de menubalk. Er verschijnt een nieuw venster waarin je een hoek kan invoeren:



Voer `pi/6` in, wat overeenkomt met 30°, en bevestig met de knop. Gefeliciteerd! Je hebt zojuist de rotatie $U(\pi/6)$ toegevoegd aan de toolbox, die er nu als volgt uitziet:

Laten we, om onze nieuwe rotatie te testen, de volgende berekening maken in QUIRKY:

Laten we snel kijken of deze uitkomst logisch is. We begonnen met de toestand $|0\rangle = \binom{1}{0}$. Volgens Vgl. (2.14) stuurt elke rotatie $U(\theta)\,|0\rangle$ naar $|\psi(\theta)\rangle = \binom{cos(\theta)}{sin(\theta)}$. In ons geval is $\theta = \pi/6$, en

$$|\psi(\pi/6)\rangle = \begin{pmatrix}\cos(\pi/6)\\ \sin(\pi/6)\end{pmatrix} = \begin{pmatrix}\sqrt{3}/2\\ 1/2\end{pmatrix}.$$

Met behulp van de quantummeetregel in Vgl. (2.7) komen we tot de conclusie dat de kans om uitkomst 1 te krijgen is

$$p_1 = \left(\frac{1}{2}\right)^2 = \frac{1}{4} = 25\%,$$

wat precies is wat Quirky ons vertelde. In de volgende opdracht gebruik je QUIRKY om op dezelfde manier het effect van de rotatie $U(\theta)$ op de andere basisvector, $|1\rangle$, te testen.

Huiswerkopdracht 2.3: De 30°-rotatie testen

1. Maak de volgende reeks bewerkingen met QUIRKY: Bereid eerst de qubit-toestand $|1\rangle$ voor, roteer dan met dezelfde hoek $\pi/6$ en voer tenslotte een meting van de qubit uit.
2. Gebruik de 'Probability Display' van QUIRKY om de waarschijnlijkheid van de meetresultaten te bepalen. Laat zien dat het antwoord van QUIRKY juist is.
3. Pas je oplossing voor de eerste vraag zo aan dat de kans op het meetresultaat nul 42 procent is.

2.4.2 Quantumbewerkingen samenstellen

We kunnen altijd twee gegeven qubitbewerkingen M en N combineren tot een nieuwe qubitbewerking. Want als $|\psi\rangle$ de begintoestand is en we passen eerst M toe, dan krijgen we $M(|\psi\rangle) = M\,|\psi\rangle$. Als we dan N toepassen is de resulterende toestand $N(M\,|\psi\rangle)$. We duiden deze samengestelde bewerking aan met NM, zodat

$$NM\,|\psi\rangle = N(M\,|\psi\rangle).$$

Pas op dat je de volgorde van de twee bewerkingen niet verwart. Als de samengestelde operatie NM is, betekent dit dat M als eerste wordt toegepast en N als tweede! Dit komt omdat M naast $|\psi\rangle$ staat en dus als eerste op de toestand moet worden toegepast.

Oefenopgave 2.4: Lineariteit van een samengestelde bewerking (optioneel)

Ga na dat NM ook lineair is.

Hint: Gebruik Vgl. (2.10).

Zo kunnen we ook drie of meer qubitbewerkingen samenstellen. Dit schrijven we op als ONM enzovoort. We kunnen met name nieuwe qubitbewerkingen krijgen door rotaties en spiegelingen samen te stellen. Dit zullen we hieronder bespreken.

Het is interessant om op te merken dat alle qubitbewerkingen die we tot nu toe hebben besproken inverteerbaar zijn. Dit betekent dat er voor elke bewerking M een andere bewerking bestaat, die we schrijven als M^{-1}, zo dat wanneer we eerst M toepassen en dan M^{-1} (of andersom) de toestand van de qubit niet verandert. [10] In formules kunnen we dit schrijven als

$$M^{-1}M = MM^{-1} = I, \tag{2.17}$$

waarbij I de identiteit-bewerking is, die de "triviale" eigenschap

$$I\,|0\rangle = |0\rangle\,, \qquad I\,|1\rangle = |1\rangle \tag{2.18}$$

heeft. (We hadden I ook kunnen definiëren als $U(0)$, de rotatie met een hoek van nul.) Door uitbreiding door lineariteit geldt dus voor elke toestand $|\psi\rangle$ dat $I\,|\psi\rangle = |\psi\rangle$.

Neem bijvoorbeeld de bewerking U, die we geometrisch zien als een rotatie over de cirkel. Dan geldt dat roteren met een hoek β ongedaan wordt gemaakt door een rotatie met een hoek $-\beta$. Om dit wat formeler te zien, hoeven we alleen maar twee keer Vgl. (2.15) te gebruiken:

$$U(-\beta)U(\beta)\,|\psi(\alpha)\rangle = U(-\beta)\,|\psi(\alpha+\beta)\rangle = |\psi(\alpha+\beta-\beta)\rangle = |\psi(\alpha)\rangle$$

en hetzelfde geldt als we eerst roteren met $-\beta$ en dan met β. Dit betekent dat de inverse bewerking $U(\beta)^{-1}$ gewoon $U(-\beta)$ is:

$$U(\beta)^{-1} = U(-\beta).$$

Op dezelfde manier, aangezien de NOT-bewerking neerkomt op een spiegeling, is het duidelijk dat het tweemaal toepassen ervan de qubit-toestand onveranderd laat. Zo blijkt uit Vgl. (1.9) dat

$$\text{NOT}\,\text{NOT}\,|0\rangle = \text{NOT}\,|1\rangle = |0\rangle \quad \text{en} \quad \text{NOT}\,\text{NOT}\,|1\rangle = \text{NOT}\,|0\rangle = |1\rangle\,.$$

Door lineariteit betekent dit dat $\text{NOT}\,\text{NOT}\,|\psi\rangle = |\psi\rangle$ voor elke toestand $|\psi\rangle$, dus NOT is niet alleen inverteerbaar maar ook zijn eigen inverse, d.w.z. $\text{NOT}^{-1} = \text{NOT}$.

Oefenopgave 2.5: Inverse van een samengestelde bewerking

Laat zien dat als M en N inverteerbaar zijn, NM dat ook is. Druk de inverse $(NM)^{-1}$ van de samengestelde bewerking uit in termen van de afzonderlijke inverses N^{-1} en M^{-1}.

Het blijkt dat er kan worden aangetoond dat elke lineaire bewerking die de qubit-toestandsruimte op zichzelf afbeeldt, inverteerbaar moet zijn. Dit is inderdaad het geval voor rotaties $U(\theta)$ en zal ook het geval zijn voor spiegelingen $V(\theta)$ die we hierna zullen bespreken. Dit is in tegenstelling tot bewerkingen op probabilistische bits, waarbij bijvoorbeeld de probabilistische flip-bewerking $F(1/2)$ elke toestand omzet in de uniforme verdeling $\binom{1/2}{1/2}$ (zie Oefenopgave 1.6) en dus niet inverteerbaar is.

2.4.3 Spiegelingen

Elke qubitbewerking is of een rotatie of een spiegeling. We zijn al bekend met de meest algemene rotatie, $U(\theta)$, gedefinieerd in Vgl. (2.13). Wat betreft spiegelingen zijn we er tot nu toe maar twee tegengekomen: Z en NOT, zie Vgl. (2.11) en (2.12). Maar hoe ziet de meest algemene spiegeling eruit?

Eén manier om een arbitraire spiegeling te krijgen is door een vaste spiegeling te nemen (bijvoorbeeld de NOT-spiegeling) en deze samen te stellen met geschikte rotaties zodat de as van de spiegeling met de juiste hoeveelheid wordt aangepast. In de volgende opdracht ga je laten zien hoe je op twee verschillende manieren de spiegeling Z kan krijgen uit de NOT-spiegeling.

Huiswerkopdracht 2.4: Z vanuit NOT

Laat Z, NOT en $U(\theta)$ de qubitbewerkingen zijn zoals gedefinieerd in Vgl. (2.11) tot (2.13).

1. Vind een hoek θ, zodat $Z = U(\theta)\,\mathrm{NOT}\,U(-\theta)$.
2. Vind een hoek θ, zodat $Z = \mathrm{NOT}\,U(\theta)$.

Kan je deze twee reeksen transformaties op de cirkel visualiseren?

Hint: Kijk terug naar Fig. 2.4 en de figuur die je voor Huiswerkopdracht 2.2 hebt getekend.

Het blijkt dat je elke spiegeling kunt verkrijgen met een vergelijkbare truc. De meest algemene spiegeling heeft de vorm

$$V(\theta) = \mathrm{NOT}\,U(\theta) = U(-\theta)\,\mathrm{NOT}. \tag{2.19}$$

Een heel handige bewerking is bijvoorbeeld de Hadamard transformatie die als volgt werkt op de basistoestanden (zie Fig. 2.6):

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = |+\rangle\,, \qquad H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = |-\rangle\,. \tag{2.20}$$

Dit wordt verkregen als het volgende speciale geval van de algemene spiegeling:

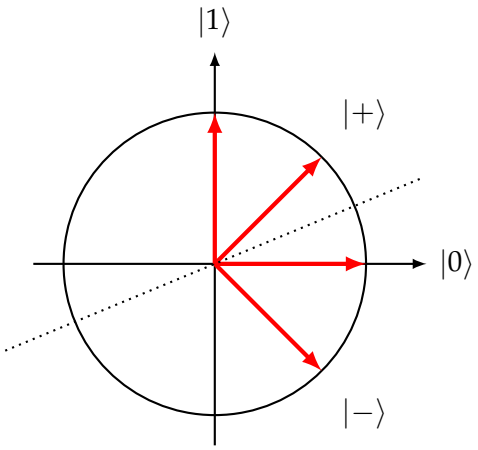
$$H = V(\pi/4). \tag{2.21}$$

Kortom, elke qubitbewerking is óf een rotatie $U(\theta)$ of een spiegeling $V(\theta)$, voor een bepaalde hoek θ.

2.5 Quantumtoestanden onderscheiden

Alice kijkt naar een hardloopwedstrijd voor robot-ezels, en noteert of haar favoriete ezel wint: een 1 als dit wel het geval is, en een 0 als dit niet het geval is. Ze zou de uitslag ook in een qubit kunnen coderen: in het meest algemene geval maakt ze een toestand $|\psi(\theta_0)\rangle$ in de situatie 0 (geen winst), en een toestand $|\psi(\theta_1)\rangle$ in de situatie 1 (wel winst). Hierbij zijn θ_0 en θ_1 arbitraire hoeken (waarmee we dus elke mogelijke combinatie van toestanden bekijken). Alice kan deze toestanden maken door simpelweg $U(\theta_0)$ of $U(\theta_1)$ toe te passen op $|0\rangle$, zie Vgl. (2.13). Stel nu dat Alice het resulterende qubit aan Bob geeft. Kan Bob dan raden welke bitwaarde (1 of 0) Alice heeft gecodeerd, met alleen de informatie die in de qubit zit? Heeft het wellicht zin om eerst een rotatie of spiegeling te doen, voordat Bob de qubit meet? We kunnen dit oefenen in de volgende opgave.

[10] Deze notatie en Vgl. (2.17) zullen je misschien aan het volgende doen denken: Als x een getal is dat niet nul is, dan is $x^{-1} = \frac{1}{x}$ zijn inverse, wat betekent dat $xx^{-1} = x^{-1}x = 1$.



Figuur 2.6: De Hadamard-bewerking H op een qubit komt overeen met een spiegeling om de as van $45/2$ graden (of $\pi/8$) (gestippeld). Verder zijn de toestanden $|0\rangle$, $|1\rangle$, $|+\rangle$, en $|-\rangle$ van Vgl. (2.20) afgebeeld.

Oefenopgave 2.6: Plus en min

Stel je voor dat je een qubit krijgt die in één van de volgende twee toestanden zit:

$$|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}, \qquad |-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}.$$

Je wilt raden welke toestand je gekregen hebt. Je kunt een rotatie toepassen en dan meten. Welke rotatie moet je hier toepassen en met welke waarschijnlijkheid kan je dan de juiste toestand raden?

Als je verschillende (bit)waarden wilt weergeven door verschillende quantumtoestanden, moet je oppassen dat je niet $|\psi(\theta)\rangle$ en $|\psi(\theta+\pi)\rangle$ gebruikt, want deze toestanden zijn niet van elkaar te onderscheiden.

Oefenopgave 2.7: Niet-onderscheidbare toestanden

Laat zien dat de twee toestanden $|\psi(\theta)\rangle$ en $|\psi(\theta+\pi)\rangle = -|\psi(\theta)\rangle$ op geen enkele manier te onderscheiden zijn. Met andere woorden, het maakt niet uit welke qubitbewerking je uitvoert, als je vervolgens de verkregen toestand meet, zullen de meetresultaten in beide gevallen altijd dezelfde waarschijnlijkheid hebben.

Het is best interessant om Oefenopgaven 2.6 en 2.7 met elkaar te vergelijken. Als twee toestanden van elkaar verschillen door een algemeen minteken, zijn ze niet te onderscheiden, zoals in Oefenopgave 2.7. Praktisch gezien beschrijven de twee vectoren $\pm|\psi(\theta)\rangle$ dezelfde toestand. Daarentegen zijn 'relatieve' mintekens zoals in Oefenopgave 2.6 belangrijk en kunnen ze zelfs leiden tot toestanden die perfect te onderscheiden zijn!

In de volgende huiswerkopdracht ga je na wat de optimale manier is om twee *arbitraire* quantumtoestanden te onderscheiden.

Huiswerkopdracht 2.5: Twee toestanden onderscheiden.

Laat θ, θ' twee hoeken zijn. Neem voor het gemak aan dat $-\frac{\pi}{2} \leq \theta \leq \theta' \leq \frac{\pi}{2}$. Stel dat Eve je een enkele qubit geeft, die of in de toestand $|\psi(\theta)\rangle$ of in de toestand $|\psi(\theta')\rangle$ zit, met voor beide een 50% kans. (Ze kan bijvoorbeeld een zuiver muntje opgooien om te beslissen welke van de twee toestanden ze aan je geeft). Jouw taak is om te bepalen welke van de twee toestanden je hebt gekregen. Met een aantal stappen ga je een optimale procedure

vinden:

1. Pas eerst een rotatie $U(\phi)$ toe met een bepaalde hoek ϕ. Welke twee mogelijke toestanden krijg je dan?

2. Meet vervolgens de qubit en interpreteer het resultaat als volgt: Als de uitkomst 0 is, dan raad je dat je de toestand $|\psi(\theta)\rangle$ hebt gekregen, anders raad je $|\psi(\theta')\rangle$. Wat is de waarschijnlijkheid dat je de toestand die je hebt gekregen juist hebt geïdentificeerd? Stel dit op in een formule in termen van θ, θ' en ϕ.

   Hint: Bereken eerst de kans van slagen ervan uitgaande dat je de eerste toestand hebt gekregen, dan de kans van slagen ervan uitgaande dat je de tweede toestand hebt gekregen, en bedenk dat je in werkelijkheid één van de twee toestanden krijgt met voor elk eem 50% kans.

3. Je hebt nog steeds de mogelijkheid om de rotatiehoek ϕ op een slimme manier te kiezen. Wat is de kans van slagen als functie van θ en θ' als je ϕ optimaal kiest?

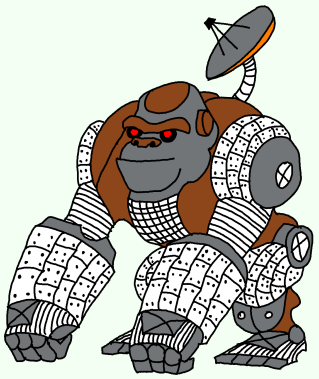
   Hint: Je kan de goniometrische identiteiten van Vgl. (2.16) gebruiken. Hiermee kun je laten zien dat

$$\sin^2\alpha = \frac{1}{2}\big(1-\cos(2\alpha)\big), \qquad \cos^2\alpha = \frac{1}{2}\big(1+\cos(2\alpha)\big). \tag{2.22}$$

   Als je vastloopt, kan je Wolfram Alpha gebruiken.

Oefenopgave 2.8: Gebroken been en arm (uitdagend)

Het probleem Alice en Bob verkennen graag de wildernis rond hun dorp. Hiervoor hebben ze twee grote gorillarobots gebouwd die door ruw terrein kunnen navigeren terwijl ze Alice en Bob comfortabel op hun rug dragen. Maar dit is geen goede dag voor Bob, zijn robot valt per ongeluk van een klif! Gelukkig overleeft Bob de val met alleen wat blauwe plekken, maar zijn robot raakt behoorlijk beschadigd: een arm, een been, en zijn communicatie-apparaat zijn allemaal gebroken. Bob heeft geen reserveonderdelen voor de benen en armen, maar het lukt hem tenminste om zijn communicatie-apparaat voor even te repareren. Helaas kan het maar één bit of één qubit versturen voor het weer stopt met werken. Bob wil Alice laten weten welk been (links of rechts) en welke arm (ook links of rechts) van zijn robot gebroken is, zodat zij het bijbehorende ledemaat van haar robot kan halen en naar hem toe kan sturen. Alice kan hem maar één ledemaat sturen (been of arm) omdat beide robots terug naar huis moeten kunnen lopen (wat ze nog steeds kunnen met drie ledematen). De situatie wordt nog ingewikkelder omdat Alice niet al het benodigde gereedschap heeft om elk willekeurig ledemaat van haar robot af te halen. Bob herinnert zich dat Alice óf het gereedschap voor benen óf dat voor armen (en niet beide) heeft meegenomen, maar hij kan zich niet herinneren welk gereedschap.



Er zijn vier mogelijke combinaties van welk been en welke arm van Bob's robot zijn gebroken – je kunt aannemen dat elk daarvan met kans 1/4 is gebeurd. Ook zijn er twee soorten ledematen die Alice van haar robot kan verwijderen (ze heeft gereedschap voor het verwijderen van benen of armen) en je kunt aannemen dat ze voor elk het juiste gereedschap heeft genomen met kans 1/2.

Vragen:

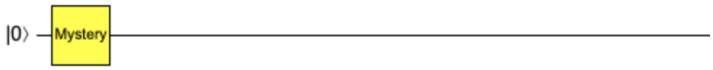
1. Als Bob maar één bit naar Alice kan sturen, hoe moet hij dan beslissen welke waarde die heeft, afhankelijk van op welke van de vier manieren zijn robot is gebroken? Hoe moet Alice zijn boodschap interpreteren en beslissen of ze het linker- of rechterbeen stuurt? (Bedenk dat Alice alleen benen of alleen armen kan sturen, en Bob niet weet of het benen of armen zijn). Als ze beiden een optimale strategie gebruiken, met welke waarschijnlijkheid zal Alice dan Bob's boodschap juist interpreteren en het juiste ledemaat voor zijn robot sturen?

2. Wat als Bob in plaats hiervan een quantumbit kan sturen? Afhankelijk van zijn situatie kan hij één van vier toestanden kiezen en Alice kan, afhankelijk van haar situatie, één van twee rotaties toepassen voordat zij het meet. Wat is hun optimale gezamenlijke strategie en wat is de kans van slagen?

Je kan ervan uitgaan dat Alice en Bob weten hoe ze elkaars berichten moeten interpreteren, aangezien ze van tevoren hebben besproken wat ze moeten doen als deze specifieke noodsituatie zich ooit voordoet.

2.5.1 Nog een mysterieuze bewerking

We hebben het nog niet gehad over het gele vakje in QUIRKY. In tegenstelling tot de mysterieuze bewerking van vorige week, die op bits werkte, werkt het mysterieuze vakje van deze week op quantumbits. Laten we deze mysterieuze quantumbewerking M noemen. Hoe komen we erachter wat er in het vakje gebeurt? Laten we eerst bepalen wat $M|0\rangle$ is. In QUIRKY kunnen we deze toestand creëren met de volgende opstelling:



Het bepalen van een onbekende toestand wordt quantumtoestand-tomografie genoemd, omdat we een onbekende quantumtoestand van buitenaf willen reconstrueren door verschillende metingen uit te voeren. Dit is een fundamentele taak waarmee experimentalisten elke dag te maken krijgen: is de toestand die in het laboratorium wordt gemaakt, inderdaad de toestand die zij wilden maken?

We kunnen al een hoop informatie krijgen door een meting uit te voeren op de onbekende toestand. Om dit te kunnen zien, stellen we

$$M|0\rangle = \begin{pmatrix} \psi_0 \\ \psi_1 \end{pmatrix}.$$

Als we een meting uitvoeren, krijgen we volgens Vgl. (2.6) de uitkomst 1 met een waarschijnlijkheid van ψ_1^2. Dit betekent dat als we het bovenstaande experiment vele malen herhalen, we verwachten dat de fractie van de keren dat we uitkomst 1 krijgen ongeveer ψ_1^2 is. Dit is geheel analoog aan hoe je de zuiverheid van een muntje kan schatten door het vele malen op te gooien en het aantal koppen en munten te tellen, zoals we vorige week hebben besproken in §1.3. Dit geeft ons een procedure voor het schatten van ψ_1^2. In QUIRKY kunnen we simpelweg de 'Probability Display' na de meting gebruiken om de kans op uitkomst 1 te bepalen:

Dus concluderen we dat $\psi_1^2 \approx 11.7\%$. Omdat $M|0\rangle$ een eenheidsvector is, kunnen we ook afleiden dat $\psi_0^2 = 1 - \psi_1^2 \approx 88.3\%$. Maar de amplitudes kunnen ook negatief zijn, dus dit

bepaalt alleen ψ_0 en ψ_1 tot aan de tekens! Denk nu terug aan Oefenopgave 2.7, waar staat dat $|\psi\rangle$ en $-|\psi\rangle$ niet te onderscheiden zijn, dus we kunnen $|\psi\rangle = M|0\rangle$ alleen bepalen tot een totaal $\pm$-teken. Er blijven dus twee mogelijkheden over:

$$\pm\begin{pmatrix}\sqrt{88.3\%}\\ \sqrt{11.7\%}\end{pmatrix}, \quad \pm\begin{pmatrix}\sqrt{88.3\%}\\ -\sqrt{11.7\%}\end{pmatrix}$$

Merk op dat deze situatie erg lijkt op Oefenopgave 2.6, waar we moesten kiezen tussen $|+\rangle$ en $|-\rangle$. In de laatste huiswerkopdracht ga je de situatie ophelderen en de innerlijke werking van de mysteriebox onthullen.

Huiswerkopdracht 2.6: Tijd voor nog een mysterie

1. Hoe bepaal je welke van de twee opties het geval is? Gebruik QUIRKY om de quantumtoestand $M|0\rangle$ tot aan het teken te bepalen.
2. Bepaal ook de quantumtoestand $M|1\rangle$ tot aan het teken.
3. *Bonus vraag:* Bepalen de stappen 1 en 2 de quantumbewerking M volledig? Zo ja, schrijf een formule op voor M. Zo nee, hoe kun je dan M achterhalen?

2.6 Natuurkundig intermezzo (optioneel)

In *The Quantum Quest* ligt de nadruk vooral op de *wiskunde* van quantum computing. Maar omdat er intussen al kleine quantumcomputers zijn gebouwd in laboratoria over de hele wereld, is het toch ook nuttig om iets te weten over de *natuurkunde* van quantumcomputers. Welke fysische effecten zorgen ervoor dat quantumcomputers werken?

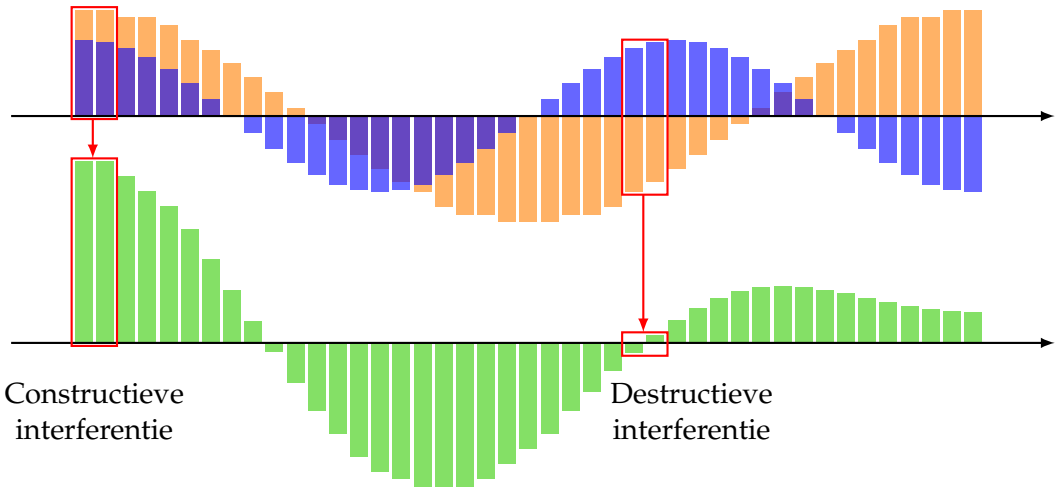
2.6.1 Interferentie

Een van de belangrijkste natuurkundige effecten die bij quantum computing wordt gebruikt is interferentie: de interactie tussen overlappende golven of trillingen. Eén manier om interferentie te kunnen zien is door te kijken naar watergolven die ontstaan door twee boten die elkaar passeren, of door tegelijkertijd twee stenen te gooien in een rustig meer. Als de golven elkaar versterken, noemen we de interferentie *constructief*, en als ze elkaar opheffen noemen we het *destructief* (zie Fig. 2.7).

Interferentie komt ook voor in andere contexten, bijvoorbeeld bij geluidsgolven. Een bekend voorbeeld is de destructieve interferentie in koptelefoons met ruisonderdrukking. Die werken door het achtergrondgeluid op te vangen en het naar je terug te spelen, maar met een tegengestelde trillingsrichting. Als dit opgenomen geluid het oorspronkelijke geluid overlapt, heffen ze elkaar op: $1-1=0$. Als de koptelefoon de trillingsrichting niet zou omkeren, maar het geluid zou afspelen zoals het is opgevangen, zou je een veel harder geluid horen: $1+1=2$. Dit zou van je hoofdtelefoon een gehoorapparaat maken!

Een belangrijk verschil tussen quantumberekeningen en probabilistische berekeningen is dat quantumberekeningen gebruik kunnen maken van beide soorten interferentie - constructieve en destructieve - terwijl probabilistische berekeningen alleen constructieve interferentie kunnen gebruiken. Om dit wiskundig te laten zien, denken we terug aan de probabilistische flipbewerking $F(1/2)$ en de Hadamard-berwerking H uit Vgl. (1.14) en (2.20):

$$\begin{aligned} F(1/2)\,[0] &= \frac{1}{2}\,[0] + \frac{1}{2}\,[1], \qquad & H|0\rangle &= \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\,, \\ F(1/2)\,[1] &= \frac{1}{2}\,[0] + \frac{1}{2}\,[1], \qquad & H|1\rangle &= \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\,. \end{aligned} \tag{2.23}$$



Figuur 2.7: Interferentie van twee golven: op elke plaats tellen de amplitudes van de blauwe en oranje golven op om de groene golf te vormen. Als beide amplitudes hetzelfde teken hebben, is de interferentie *constructief* en krijgen we een nog grotere amplitude. Als de interfererende amplitudes tegenovergestelde tekens hebben, is de interferentie *destructief* en krijgen we een veel kleinere amplitude.

Naast de wortels zijn de twee bewerkingen vrijwel identiek. Merk alleen op dat $F(1/2)\,[1]$ een plusteken heeft, terwijl $H\,|1\rangle$ een minteken heeft. Dit lijkt een klein verschil, maar het kan grote gevolgen hebben.

Laten we het effect van deze twee bewerkingen op de uniforme verdeling $\frac{1}{2}\,[0] + \frac{1}{2}\,[1]$ en het quantumanaloog daarvan, de plustoestand $|+\rangle = \frac{1}{\sqrt{2}}\,|0\rangle + \frac{1}{\sqrt{2}}\,|1\rangle$ bekijken. De probabilistische flip-bewerking $F(1/2)$ werkt als volgt op de uniforme verdeling:

$$\begin{aligned} F(1/2)\left(\frac{1}{2}\,[0] + \frac{1}{2}\,[1]\right) &= \frac{1}{2}\,F(1/2)\,[0] + \frac{1}{2}\,F(1/2)\,[1] \\ &= \frac{1}{2}\left(\frac{1}{2}\,[0] + \frac{1}{2}\,[1]\right) + \frac{1}{2}\left(\frac{1}{2}\,[0] + \frac{1}{2}\,[1]\right) \\ &= \left(\frac{1}{2}\cdot\frac{1}{2} + \frac{1}{2}\cdot\frac{1}{2}\right)[0] + \left(\frac{1}{2}\cdot\frac{1}{2} + \frac{1}{2}\cdot\frac{1}{2}\right)[1] \\ &= \frac{1}{2}\,[0] + \frac{1}{2}\,[1], \end{aligned}$$

waarbij we lineariteit hebben gebruikt, 2.23, en de kansen van $[0]$ en $[1]$ bij elkaar hebben geschreven. Merk op hoe de kansen op $[1]$ van beide termen elkaar versterken, wat de uiteindelijke kans van $1/2$ oplevert. Dit is vrij logisch: als je een uniform willekeurige bit flipt, blijft het uniform willekeurig.

Maar laten we nu de werking van de Hadamard-bewerking H op de plustoestand $|+\rangle$ bekijken:

$$\begin{aligned} H\left(\frac{1}{\sqrt{2}}\,|0\rangle + \frac{1}{\sqrt{2}}\,|1\rangle\right) &= \frac{1}{\sqrt{2}}\,H\,|0\rangle + \frac{1}{\sqrt{2}}\,H\,|1\rangle \\ &= \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}\,|0\rangle + \frac{1}{\sqrt{2}}\,|1\rangle\right) + \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}\,|0\rangle - \frac{1}{\sqrt{2}}\,|1\rangle\right) \\ &= \left(\frac{1}{\sqrt{2}}\cdot\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}\cdot\frac{1}{\sqrt{2}}\right)|0\rangle + \left(\frac{1}{\sqrt{2}}\cdot\frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}}\cdot\frac{1}{\sqrt{2}}\right)|1\rangle \\ &= |0\rangle\,. \end{aligned}$$

De berekening is bijna identiek, maar het resultaat is totaal anders: de amplitudes bij $|1\rangle$ heffen elkaar volledig op en we houden alleen $|0\rangle$ over. Zo'n destructieve interferentie is onmogelijk met probabilistische bits omdat waarschijnlijkheden altijd positief zijn – ze kunnen elkaar alleen versterken maar nooit opheffen.

Hoewel probabilistische en quantumbits nogal op elkaar lijken, laat dit voorbeeld zien hoe ze zich toch totaal anders kunnen gedragen dankzij destructieve interferentie. Veel van de quantumverrassingen die je in de komende weken zult tegenkomen, zijn op de een of andere manier een gevolg van dit verschijnsel. De mogelijkheid van destructieve interferentie is precies wat een quantumcomputer een voordeel geeft ten opzichte van klassieke computers – het stelt de quantumcomputer in staat om alleen het juiste antwoord te geven, terwijl de foute antwoorden worden opgeheven door de destructieve interferentie. Als je kiest om met Quest 4 en 5 door te gaan, dan zal je zien hoe dit een belangrijke rol speelt in quantumalgoritmes!

2.6.2 Polarisatie

Nu dat we de wiskunde van qubit-toestanden en -bewerkingen kennen, zou het fijn zijn om ze met iets fysisch te kunnen verbinden.

Een van de simpelste manieren om een qubit fysisch voor te stellen is door de polarisatie van licht. Licht is een elektromagnetische golf die in een rechte lijn door de ruimte beweegt. Deze golf trilt in een richting loodrecht op de richting waarin het beweegt. Merk op dat er verschillende van zulke richtingen mogelijk zijn – een golf die voorwaarts beweegt kan van links naar rechts of van boven naar beneden trillen. Deze horizontale en verticale trillingsrichtingen kunnen worden gebruikt om de twee basistoestanden van een qubit weer te geven:

$$|\leftrightarrow\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad |\updownarrow\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

In het algemeen kunnen we een golf die trilt onder een hoek θ met de horizontale as gebruiken om de toestand

$$|\psi(\theta)\rangle = \cos\theta\,|\leftrightarrow\rangle + \sin\theta\,|\updownarrow\rangle = \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix}$$

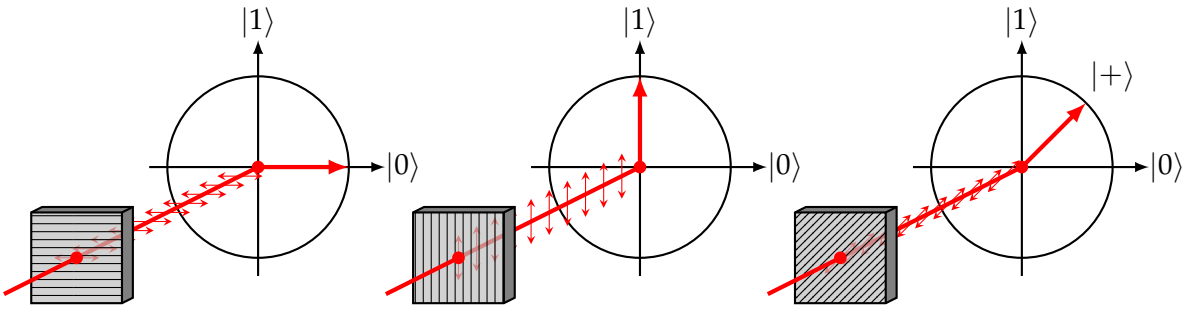
voor te stellen.

Diagonaal gepolariseerd licht dat trilt onder een hoek van 45° tussen de verticale en horizontale richtingen vertegenwoordigt bijvoorbeeld de toestand $|\psi(\pi/4)\rangle = |+\rangle$. Merk op dat in deze weergave de trillingsrichting van de elektromagnetische golf overeenkomt met de richting van de vector die we gebruikten in Fig. 2.1 van §2.1.2 om een qubit-toestand op een cirkel voor te stellen.

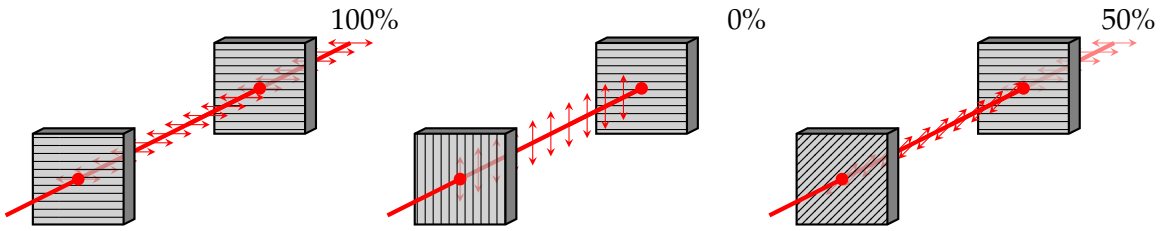
Om deze toestanden voor te bereiden, kunnen we gewoon een lichtstraal door een polarisator sturen, zoals die in je zonnebril of in een 3D-bril van de bioscoop. Een polarisator laat alleen een deel van de golf door – dat waarvan de trillingsrichting overeenkomt met de richting van de polarisator. Om de toestand $|\psi(\theta)\rangle$ te maken, kunnen we gewoon de polarisator schuin houden onder een hoek θ ten opzichte van de horizontale as. In het voorbeeld van de polarisatoren is te zien hoe de toestanden $|0\rangle$, $|1\rangle$ en $|+\rangle$ kunnen worden bereid.

Een interessante eigenschap van het weergeven van qubit-toestanden door gepolariseerd licht is dat de toestanden $|\psi(\theta)\rangle$ en $|\psi(\theta+\pi)\rangle$ op dezelfde manier worden bereid – het schuin houden van de polarisator onder een hoek θ. Dit betekent dat deze twee toestanden identiek moeten zijn! Polarisatie geeft dus een intuïtieve verklaring voor waarom de toestanden $|\psi\rangle$ en $-|\psi\rangle$ niet onderscheidbaar zijn (zie Oefenopgave 2.7).

Een andere handige eigenschap van qubits als gepolariseerd licht zien is dat we een meting makkelijk kunnen voorstellen. Stel dat we de toestand $|\psi(\theta)\rangle$ willen meten om de waarschijnlijkheid van de uitkomst 0 te bepalen. Als de toestand ons wordt gegeven als een lichtstraal, gepolariseerd onder een hoek θ, kunnen we deze simpelweg door een horizontale polarisator



Figuur 2.8: Horizontaal, verticaal en diagonaal gepolariseerd licht kan worden gebruikt om de qubit-toestanden $|0\rangle$, $|1\rangle$ en $|+\rangle$ voor te stellen.



Figuur 2.9: De mate van horizontale polarisatie in licht kan worden bepaald door het door een horizontale polarisator te laten gaan en dan de helderheid te meten. Voor horizontaal, verticaal en diagonaal gepolariseerd licht geeft dit een helderheid van 100%, 0% en 50%, die overeenkomen met de waarschijnlijkheid van het waarnemen van uitkomst 0 bij het meten van de toestanden $|0\rangle$, $|1\rangle$, en $|+\rangle$.

laten gaan en zien hoeveel licht er doorkomt – als de helderheid is afgenomen tot 70%, is de kans op uitkomst 0 70%. In het bijzonder, als de inkomende bundel horizontaal gepolariseerd was, gaat alles erdoorheen, en als de bundel verticaal gepolariseerd was, gaat er niets doorheen. Als een diagonaal gepolariseerde lichtstraal door een horizontale polarisator wordt gestuurd, neemt de helderheid met 50% af (zie Fig. 2.9).

Oefenopgave 2.9: Polarisatie experiment

Als je thuis toevallig een gepolariseerde zonnebril hebt liggen, kan je die opzetten en naar het scherm van je telefoon of computer kijken. Meestal zenden schermen gepolariseerd licht uit (de polarisatierichting zal afhangen van het apparaat). Als je je hoofd schuin houdt, zou je moeten zien dat het scherm lichter of donkerder wordt. Kan je uitleggen waarom dit het geval is?

Polarisatie van licht is slechts één voorbeeld van hoe een qubit in een laboratorium zou kunnen worden gerealiseerd. Een ander voorbeeld is de *locatie* van het lichtdragende deeltje dat een foton wordt genoemd – aangezien een foton zich gedraagt volgens de wetten van de quantummechanica, kan het zich tegelijkertijd op twee plaatsen in een superpositie bevinden. Als we deze plaatsen 0 en 1 noemen, komt de toestand van het foton overeen met een qubit. Er zijn vele andere mogelijkheden: de stroom in een supergeleidend circuit kan tegelijkertijd in beide richtingen stromen, een elektron kan tegelijkertijd in twee banen rond een atoom zitten, enzovoort. Kortom, elk quantummechanisch systeem dat zich in twee verschillende toestanden kan bevinden, kan zich ook in de superpositie daarvan bevinden, en kan dus potentieel worden gebruikt als een fysieke weergave van een qubit.

2.7 Oplossingen van de oefenopgaven

Oplossing van Oefenopgave 2.1

Merk op dat

$$|+\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix} = |\psi(\pi/4)\rangle\,, \qquad |-\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix} = |\psi(-\pi/4)\rangle\,.$$

De hoeken zijn dus $\theta = \pm\pi/4$ en de twee toestanden liggen respectievelijk 45 graden boven en onder $|0\rangle$.

Oplossing van Oefenopgave 2.2

Neem $|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, wat een geldige quantumtoestand is. Omdat MAD is verkregen door lineair uit te breiden, volgt uit Vgl. (2.8) dat

$$\begin{aligned}\mathrm{MAD}\,|\psi\rangle &= \mathrm{MAD}\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) = \frac{1}{\sqrt{2}}\mathrm{MAD}\,|0\rangle + \frac{1}{\sqrt{2}}\mathrm{MAD}\,|1\rangle \\ &= \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{2}(|0\rangle + |1\rangle) = \left(\frac{1}{\sqrt{2}} + \frac{1}{2}\right)|0\rangle + \frac{1}{2}|1\rangle\,.\end{aligned}$$

Maar

$$\left(\frac{1}{\sqrt{2}} + \frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^2 = 1 + \frac{1}{\sqrt{2}} \neq 1,$$

dus $\mathrm{MAD}\,|\psi\rangle$ is niet een geldige qubit-toestand

Oplossing van Oefenopgave 2.3

1. $U(\alpha)$ werkt als volgt op een arbitraire toestand:

$$\begin{aligned}U(\alpha)\begin{pmatrix}\psi_0\\\psi_1\end{pmatrix} &= U(\alpha)\left(\psi_0\begin{pmatrix}1\\0\end{pmatrix} + \psi_1\begin{pmatrix}0\\1\end{pmatrix}\right) = \psi_0\begin{pmatrix}\cos\alpha\\\sin\alpha\end{pmatrix} + \psi_1\begin{pmatrix}-\sin\alpha\\\cos\alpha\end{pmatrix} \\ &= \begin{pmatrix}\psi_0\cos\alpha - \psi_1\sin\alpha\\\psi_0\sin\alpha + \psi_1\cos\alpha\end{pmatrix}.\end{aligned}$$

2. Omdat $|\psi(\beta)\rangle = \begin{pmatrix}\cos\beta\\\sin\beta\end{pmatrix}$,

$$\begin{aligned}U(\alpha)\,|\psi(\beta)\rangle &= U(\alpha)\begin{pmatrix}\cos\beta\\\sin\beta\end{pmatrix} = \begin{pmatrix}\cos\beta\cos\alpha - \sin\beta\sin\alpha\\\cos\beta\sin\alpha + \sin\beta\cos\alpha\end{pmatrix} = \begin{pmatrix}\cos(\alpha+\beta)\\\sin(\alpha+\beta)\end{pmatrix} \\ &= |\psi(\alpha+\beta)\rangle\,.\end{aligned}$$

Oplossing van Oefenopgave 2.4

Neem een arbitraire toestand $\psi_0 |0\rangle + \psi_1 |1\rangle$, gebruik eerst de lineariteit van M en dan de lineariteit van N:

$$\begin{aligned} NM(\psi_0 |0\rangle + \psi_1 |1\rangle) &= N\big(M(\psi_0 |0\rangle + \psi_1 |1\rangle)\big) \\ &= N\big(\psi_0 M |0\rangle + \psi_1 M |1\rangle\big) = \psi_0 NM |0\rangle + \psi_1 NM |1\rangle . \end{aligned}$$

In de laatste stap hebben we gebruik gemaakt van 2.10.

Oplossing van Oefenopgave 2.5

Er geldt $(NM)^{-1} = M^{-1}N^{-1}$, want voor elke $|\psi\rangle$ geldt dat

$$M^{-1}N^{-1}NM |\psi\rangle = M^{-1}(N^{-1}N(M |\psi\rangle)) = M^{-1}(M |\psi\rangle) = M^{-1}M |\psi\rangle = |\psi\rangle$$

en

$$NMM^{-1}N^{-1} |\psi\rangle = N(MM^{-1}(N^{-1} |\psi\rangle)) = N(N^{-1} |\psi\rangle) = NN^{-1} |\psi\rangle = |\psi\rangle .$$

Oplossing van Oefenopgave 2.6

Pas $U(-\pi/4)$ toe en voer een meting uit. Je kunt nu de toestand met zekerheid raden!

Oplossing van Oefenopgave 2.7

We zagen hierboven dat elke combinatie M van rotaties en spiegelingen lineair is. Dus als $M |\psi(\theta)\rangle = |\psi(\theta')\rangle = \binom{\cos\theta'}{\sin\theta'}$, dan is $M\big(-|\psi(\theta)\rangle\big) = -|\psi(\theta')\rangle = \binom{-\cos\theta'}{-\sin\theta'}$. Volgens Vgl. (2.6) zijn de kansen p_0 en p_1 op meetresultaten voor beide toestanden gelijk.

Oplossing van Oefenopgave 2.9

Door je hoofd schuin te houden verander je de hoek tussen de polarisator in je zonnebril en de richting waarin de elektromagnetische lichtgolven van je scherm trillen. Omdat de hoeveelheid licht die door een polarisator heen kan gaan afhangt van deze hoek, zal het scherm helderder of donkerder lijken. Op dezelfde manier zal het veranderen van de hoek θ de waarschijnlijkheid veranderen dat een meting van de toestand $|\psi(\theta)\rangle$ de uitkomst 0 oplevert.

Oplossing van Oefenopgave 2.8

In een ideale situatie zou Bob 2 bits willen sturen die aangeven welk been en welke arm gebroken is. Voor Alice is maar één van deze bits van belang, omdat zij maar één soort gereedschap bij zich heeft.

1. We duiden de twee mogelijke waarden van elke bit aan met L (links) en R (rechts). Een strategie die Bob kan gebruiken is om de "meerderheid"van zijn twee bits te verzenden. Hij kan dan namelijk de volgende codering gebruiken: $LL \mapsto L$, $RR \mapsto R$. De overige twee gevallen kan hij willekeurig coderen, bijvoorbeeld $LR \mapsto L$, $RL \mapsto R$. De strategie van Alice is simpelweg het versturen van de ledemaat die overeenkomt met de boodschap van Bob (de linker-ledemaat als zij L ontvangt en de rechter-ledemaat als zij R ontvangt). Dit werkt met waarschijnlijkheid

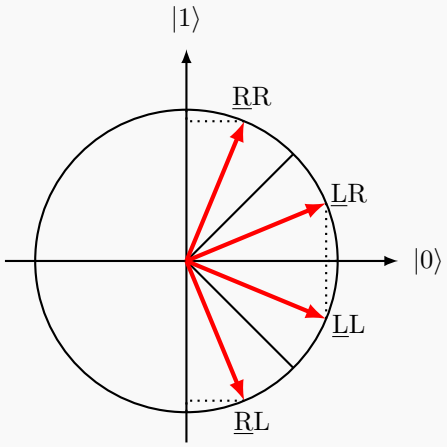
$$\frac{1}{4}\left(1+1+\frac{1}{2}+\frac{1}{2}\right) = \frac{3}{4} = 0.75, \tag{2.24}$$

   waarbij de vier termen tussen haakjes de kansen zijn dat Alice de juiste beslissing neemt voor elk van de vier situaties van Bob.

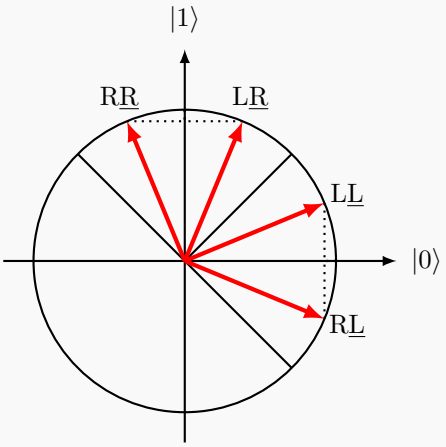
2. Bob kan de volgende qubit-toestand verzenden, afhankelijk van welke ledematen gebroken zijn (d.w.z. LL betekent linkerbeen en linkerarm, LR betekent linkerbeen en rechterarm, enz.)

$$\begin{aligned}
|LL\rangle &= \cos(\pi/8)\,|0\rangle - \sin(\pi/8)\,|1\rangle \\
|LR\rangle &= \cos(\pi/8)\,|0\rangle + \sin(\pi/8)\,|1\rangle \\
|RR\rangle &= \cos(3\pi/8)\,|0\rangle + \sin(3\pi/8)\,|1\rangle \\
|RL\rangle &= \cos(3\pi/8)\,|0\rangle - \sin(3\pi/8)\,|1\rangle
\end{aligned}$$

   Om de bit van het been te vinden, doet Alice gewoon een meting. Om de bit van de arm te vinden, past Alice $U(\pi/4)$ toe en meet dan. (Merk op dat ze niet beide bits kan krijgen omdat de oorspronkelijke toestand niet meer bestaat na de meting). Ze zal dan slagen met waarschijnlijkheid $\cos^2(\pi/8) = \frac{1}{2} + \frac{1}{2\sqrt{2}} \approx 0.85$. Dit is beter dan in het eerste scenario!



Het been-bit meten



Het arm-bit meten

Quest 3: Magie van verstrengeling

In de vorige Quests zagen we hoe één probabilistische bit en hoe één quantumbit zich gedragen. Deze week ga je leren wat er gebeurt als je twee (quantum)bits hebt. We bespreken eerst de mogelijke toestanden van twee probabilistische bits, en hoe ze *gecorreleerd* kunnen zijn. Daarna bekijken we twee quantumbits, en hoe deze een unieke quantum-eigenschap kunnen hebben: *verstrengeling*.

3.1 Twee probabilistische bits

Als je twee muntjes op een tafel hebt liggen, dan zijn er vier mogelijke configuraties:

Deze komen overeen met de vier mogelijke bitstrings[11]: 00, 01, 10, 11.

Als je twee probabilistische bits hebt, wordt hun toestand beschreven door een kansverdeling over deze vier mogelijke deterministische toestanden. Met andere woorden, hun toestand wordt beschreven door vier getallen $p_{00}, p_{01}, p_{10}, p_{11} \geq 0$, waarbij $p_{00} + p_{01} + p_{10} + p_{11} = 1$. Net als in het geval van een enkele bit kunnen we dit opschrijven als een vector

$$\begin{pmatrix} p_{00} \\ p_{01} \\ p_{10} \\ p_{11} \end{pmatrix}. \tag{3.1}$$

Hoewel deze weergave volledig correct is, is het nogal onhandig omdat het moeilijk kan zijn om bij te houden welke waarschijnlijkheid hoort bij welke bitconfiguratie (is het eerst p_{10} of p_{01}?!), en het wordt alleen maar ingewikkelder als we meer dan twee bits hebben. Het is veel handiger om een notatie te gebruiken waarmee we direct kunnen bijhouden welke waarschijnlijkheden zijn toegewezen aan bepaalde bitstrings. We breiden daarom de notatie uit die we gebruikten in Vgl. (1.3) voor een enkele probabilistische bit en schrijven de bovenstaande twee-bits-toestand als volgt:

$$p_{00}[00] + p_{01}[01] + p_{10}[10] + p_{11}[11]. \tag{3.2}$$

Als je wilt, kun je deze notatie altijd omzetten naar de 4-vector notatie in Vgl. (3.1) door gebruik te maken van de volgende uitbreiding van Vgl. (1.2):

$$[00] = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \qquad [01] = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \qquad [10] = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \qquad [11] = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}. \tag{3.3}$$

Een voordeel van deze notatie in het geval van meerdere bits is dat we elementen die nul zijn gewoon weg kunnen laten. We kunnen bijvoorbeeld gewoon schrijven

$$\frac{1}{2}[00] + \frac{1}{2}[11]$$

in plaats van het veel onhandigere

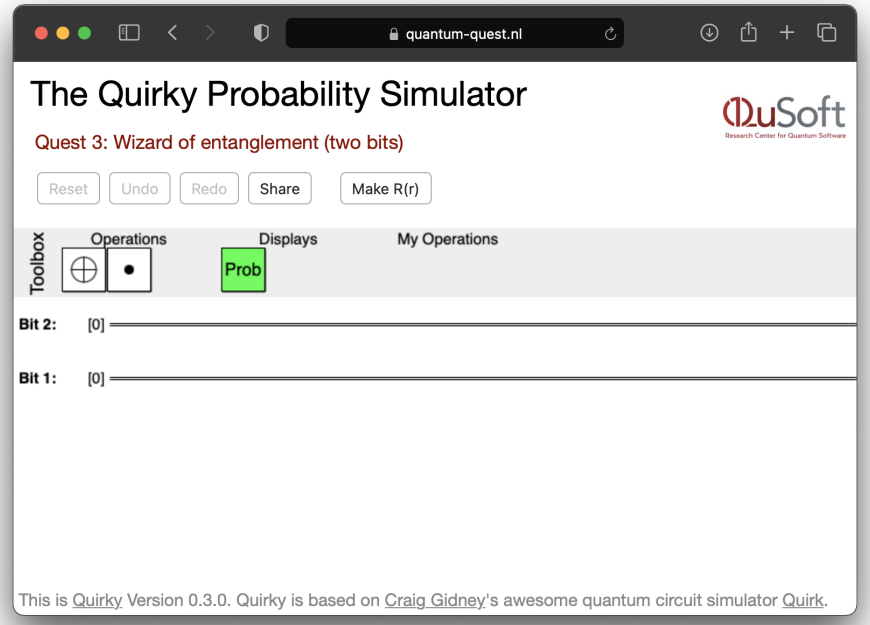
$$\frac{1}{2}[00] + 0\,[01] + 0\,[10] + \frac{1}{2}[11].$$

[11] Met een 'string' bedoelen we een 'reeks van tekens', of in dit geval een reeks van bit-waarden. Dit is een handige notatie om aan te geven hoe je muntjes liggen, vooral als we naar meerdere muntjes kijken.

Met deze notatie is het ook veel eenvoudiger om metingen en bewerkingen op meerdere probabilistische bits te beschrijven. We kunnen het gedrag van twee probabilistische bits onderzoeken met QUIRKY, dat sinds vorige week weer nieuwe krachten heeft gekregen. Ga om te beginnen naar:

https://www.quantum-quest.org/quirky

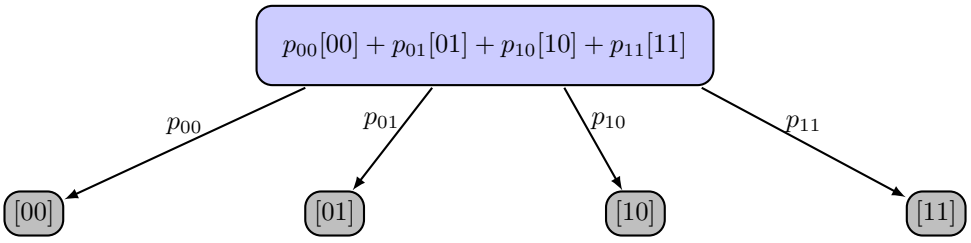
en klik op 'Quest 3' en selecteer 'Two Bits'. Je webbrowser zal eruit zien als Fig. 3.1. Merk op dat we nu twee draden hebben, die overeenkomen met twee bits, die zijn ingesteld in de toestand $[00]$. Het is misschien verrassend dat het eerste bit overeenkomt met de *onderste* draad en het tweede bit met de *bovenste* draad. Daarnaast is er een nieuwe vakje: $\boxed{\bullet}$ (maar het mysterievakje is verdwenen). De betekenis van dit vakje bespreken we later in dit hoofdstuk.



Figuur 3.1: Quirky voor Quest 3.

3.1.1 Beide bits meten

Het meten (of 'bekijken') van twee probabilistische bits werkt op dezelfde manier als Vgl. (1.18) voor een enkel bit. Je krijgt een van de vier mogelijke uitkomsten (00, 01, 10, of 11) met de bijbehorende waarschijnlijkheid:



(3.4)

De toestand van beide bits na de meting is niet meer een verdeling over vier mogelijkheden: als je de meetuikomst weet, dan heb je een deterministische toestand. Om dit verschil te benadrukken, gebruiken we lichtblauw voor probabilistische bits en grijs voor deterministische bits na de meting. Hoe kunnen we beide bits meten in QUIRKY? We gebruiken hiervoor gewoon weer de 'Probability Display':

Merk op dat de 'Probability Display' standaard verbonden is met *beide* draden, dus het toont de waarschijnlijkheden voor *beide* bits. De volgorde van de waarschijnlijkheden is hetzelfde als in de 4-vector notatie van Vgl. (3.1). Maar gelukkig hoef je de volgorde niet te onthouden. Je kunt gewoon met je muis over de tabel gaan om jezelf eraan te herinneren (bedankt, Craig!).

Als de twee probabilistische bits bijvoorbeeld in de toestand

$$\frac{1}{2}[00] + \frac{1}{2}[11], \tag{3.5}$$

zitten, krijgen we als meetuitkomst of 00 of 11, elk met een waarschijnlijkheid van 50%. Merk op dat deze toestand speciaal is – als we zien dat de meetuitkomst van het eerste bit 0 is, weten we meteen dat de uitkomst van het tweede bit ook 0 moet zijn, en net zo voor wanneer één van de twee uitkomsten 1 is. Omdat de meetuitkomsten van beide bits altijd gelijk zijn, noemen we de twee bits in toestand (3.5) perfect gecorreleerd. We zullen hieronder zien hoe zulke toestanden gemaakt kunnen worden.

3.1.2 Lokale bewerkingen

Als je twee of meer probabilistische bits hebt, kun je daar op veel verschillende manieren een bewerking op uitvoeren. In het bijzonder kun je een bewerking op alle bits tegelijk uitvoeren met een globale bewerking of slechts op één of een paar tegelijk met een lokale bewerking. Laten we eerst de lokale bewerkingen bekijken.

Denk aan de NOT-bewerking uit §1.2 die een bit flipt. Wat gebeurt er als we twee bits hebben en we NOT alleen op de eerste toepassen? In dat geval moet het eerste bit geflipt worden terwijl het tweede bit hetzelfde moet blijven. Dit betekent dat de *lokale* NOT-bewerking op het eerste bit, die we zullen aanduiden met NOT_1, als volgt werkt:

$$\mathrm{NOT}_1\,[00] = [10], \quad \mathrm{NOT}_1\,[01] = [11], \quad \mathrm{NOT}_1\,[10] = [00], \quad \mathrm{NOT}_1\,[11] = [01]. \tag{3.6}$$

Op dezelfde manier, als we alleen NOT toepassen op het tweede bit, werkt de resulterende bewerking NOT_2 als volgt:

$$\mathrm{NOT}_2\,[00] = [01], \quad \mathrm{NOT}_2\,[01] = [00], \quad \mathrm{NOT}_2\,[10] = [11], \quad \mathrm{NOT}_2\,[11] = [10]. \tag{3.7}$$

Wat we zojuist beschreven hebben zijn lokale NOT-bewerkingen op deterministische bits. Hoe moeten we ze uitbreiden naar probabilistische bits? Denk terug aan §1.2.1, waar staat dat elke bewerking die volledig gedefinieerd is op deterministische bits kan worden uitgebreid *door lineariteit* naar probabilistische bits. Als voorbeeld: NOT_2 werkt als volgt op twee probabilistische bits:

$$\begin{aligned}
&\mathrm{NOT}_2\,(p_{00}[00] + p_{01}[01] + p_{10}[10] + p_{11}[11]) \\
&\quad = p_{00}[01] + p_{01}[00] + p_{10}[11] + p_{11}[10] \\
&\quad = p_{01}[00] + p_{00}[01] + p_{11}[10] + p_{10}[11],
\end{aligned}$$

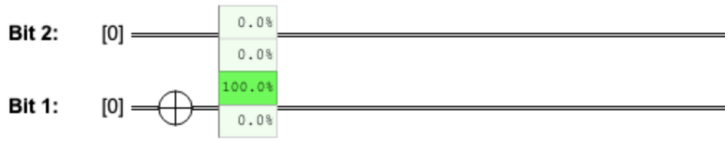
waarbij we in de eerste stap Vgl. (3.7) gebruikten en in de tweede stap alleen de termen in de gebruikelijke volgorde hebben gezet. Je kunt dit ook schrijven in de 4-vector notatie, maar dat is iets minder intuïtief:

$$\mathrm{NOT}_2 \begin{pmatrix} p_{00} \\ p_{01} \\ p_{10} \\ p_{11} \end{pmatrix} = \begin{pmatrix} p_{01} \\ p_{00} \\ p_{11} \\ p_{10} \end{pmatrix}. \tag{3.8}$$

Oefenopgave 3.1: NOT_1 in de 4-vector notatie (optioneel)

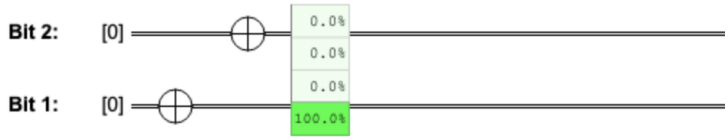
Noteer, net als bij Vgl. (3.8), de werking van NOT_1 op twee probabilistische bits in de 4-vector notatie.

Om een enkele-bitbewerking in QUIRKY toe te passen, zetten we het bijbehorende vakje op ofwel de eerste of de tweede draad. De volgende reeks bewerkingen maakt bijvoorbeeld de toestand [10] en toont de uitkomstwaarschijnlijkheden bij het meten van beide bits:

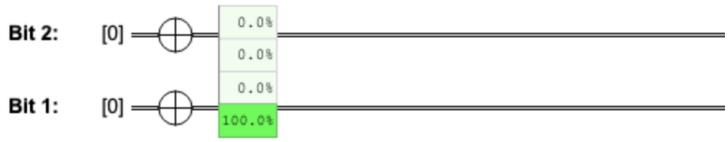


Dit is logisch omdat de onderste draad in QUIRKY overeenkomt met het eerste bit.

Op dezelfde manier, als we eerst één bit omdraaien en dan de andere, is het resultaat de toestand [11]:



Het is duidelijk dat de volgorde waarin we de twee NOT-bewerkingen toepassen er niet toe doet. Dit betekent dat we ze ook *parallel* kunnen toepassen:



We kunnen op dezelfde manier random-bewerkingen toepassen op een van de bits. Stel bijvoorbeeld dat we op het eerste bit de bewerking $R(r)$ toepassen die een bit reset met kans r (Vgl. (1.13)). Aangezien $R(r)[0] = [0]$, geldt dat

$$R(r)_1[00] = [00], \qquad R(r)_1[01] = [01]. \tag{3.9}$$

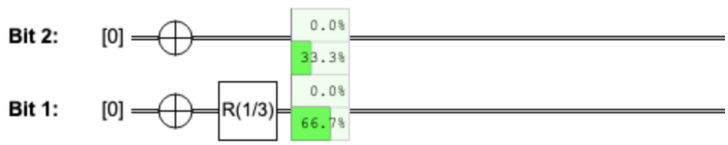
En omdat $R(r)[1] = r[0] + (1-r)[1]$, volgt dat

$$R(r)_1[10] = r[00] + (1-r)[10], \qquad R(r)_1[11] = r[01] + (1-r)[11]. \tag{3.10}$$

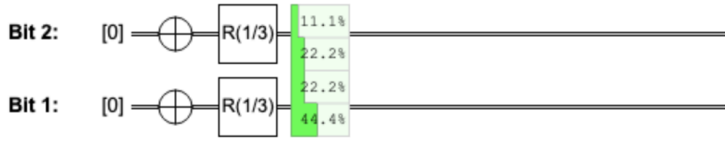
Als we bijvoorbeeld de toestand [11] klaarmaken en $R(1/3)$ toepassen op het eerste bit, krijgen we

$$R(1/3)_1[11] = \frac{1}{3}[01] + \frac{2}{3}[11], \tag{3.11}$$

zoals QUIRKY ook bevestigt:



Hier is een nog interessanter voorbeeld, dat je in de onderstaande opdracht kunt bekijken:



(3.12)

Huiswerkopdracht 3.1: $R(r)$ op het tweede bit

1. Schrijf formules op voor $R(r)_2$, analoog aan Vgl. (3.9) en (3.10).
2. Leg uit waarom QUIRKY het juiste antwoord geeft in (3.12).

3.1.3 Het meten van één bit

Als je twee probabilistische bits hebt en je meet er maar één, wat zijn dan de kansen dat je elk van de twee mogelijke uitkomsten krijgt? Onze notatie is bijzonder geschikt om dit uit te zoeken. We nemen weer een algemene probabilistische twee-bits toestand

$$p_{00}[00] + p_{01}[01] + p_{10}[10] + p_{11}[11].$$

Als je de kans op uitkomst 0 wilt vinden bij het meten van een bit, tel je gewoon de kansen op van alle termen waarbij het bit dat je meet in de gewenste toestand 0 zit. Hetzelfde geldt voor de uitkomst 1.

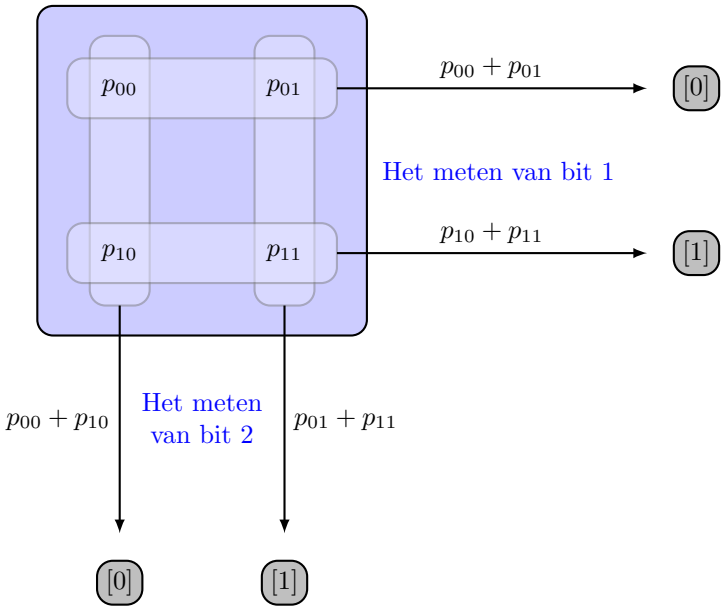
De kans om uitkomst 1 te krijgen als je *eerste* bit meet, is bijvoorbeeld

$$p_{10} + p_{11}, \tag{3.13}$$

wat overeenkomt met de waarschijnlijkheden van Vgl. (3.4) die leiden tot $[10]$ en $[11]$, de twee bitstrings die beginnen met 1. Op dezelfde manier is de kans op het waarnemen van uitkomst 0 bij het meten van de *tweede* bit

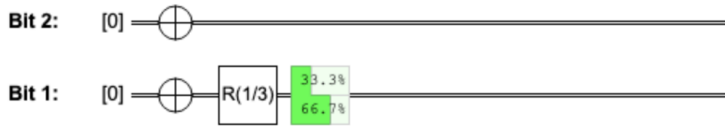
$$p_{00} + p_{10},$$

wat overeenkomt met de kansen die leiden tot de twee bitstrings die eindigen op nul, $[00]$ en $[10]$. Een makkelijke manier om dit te berekenen is door de vier kansen in een 2×2 vierkant te zetten, zoals in Fig. 3.2.

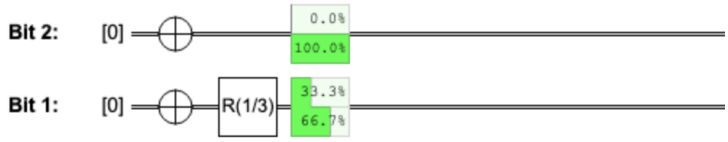


Figuur 3.2: Waarschijnlijkheden van meetresultaten wanneer je maar één van de twee probabilistische bits meet.

We kunnen ook QUIRKY gebruiken om de waarschijnlijkheden te laten zien als je een enkele bit meet. Je hoeft alleen maar het formaat van de Probability Display aan te passen zodat het alleen op een enkele draad staat, zoals in het volgende voorbeeld:



Je kan zelfs tegelijkertijd meten wat de waarschijnlijkheden van uitkomsten zijn als je beide bits los van elkaar meet.



Merk op dat dit resultaat erg logisch is. Omdat de twee bits nooit "gecorreleerd" zijn, dus ze hebben niet met elkaar kunnen communiceren, is het duidelijk dat het eerste bit in toestand $\frac{1}{3}[0] + \frac{2}{3}[1]$ moet zitten en het tweede bit in toestand $[1]$.

3.1.4 De toestand van het andere bit

Na de meting zit het bit dat gemeten is in een deterministische toestand die hoort bij de uitkomst van de meting (net als bij het meten van een enkele bit – zie Vgl. (1.18)). Maar hoe zit het met het andere bit dat niet gemeten is? De toestand na de meting zal in het algemeen niet deterministisch zijn. Bijvoorbeeld, als de begintoestand van de twee probabilistische bits

$$\frac{1}{2}[10] + \frac{1}{2}[11] \tag{3.14}$$

is en het eerste bit wordt gemeten, is de kans op het waarnemen van uitkomst 1 gelijk aan $1/2 + 1/2 = 1$. Met andere woorden, het eerste bit in Vgl. (3.14) is deterministisch en de twee waarschijnlijkheden beschrijven eigenlijk alleen het tweede bit. Je kunt deze toestand dus beschouwen als het resultaat van het samenvoegen van twee afzonderlijke probabilistische bits: $[1]$ en $\frac{1}{2}[0] + \frac{1}{2}[1]$ (we zullen meer vertellen over het combineren van twee probabilistische bits in §3.1.7). Het is dus logisch dat de toestand van het tweede bit na de meting uniform random is, namelijk

$$\frac{1}{2}[0] + \frac{1}{2}[1].$$

In het algemeen kunnen we beginnen met twee probabilistische bits in een arbitraire toestand

$$p_{00}[00] + p_{01}[01] + p_{10}[10] + p_{11}[11] \tag{3.15}$$

en we meten het eerste bit. De toestand van het overgebleven bit hangt in het algemeen af van de uitkomst van de meting. Als de uitkomst bijvoorbeeld 1 was, verzamelen we om de toestand van het tweede bit te vinden eerst alle termen van Vgl. (3.15) waarbij het eerste bit de waarde 1 heeft:

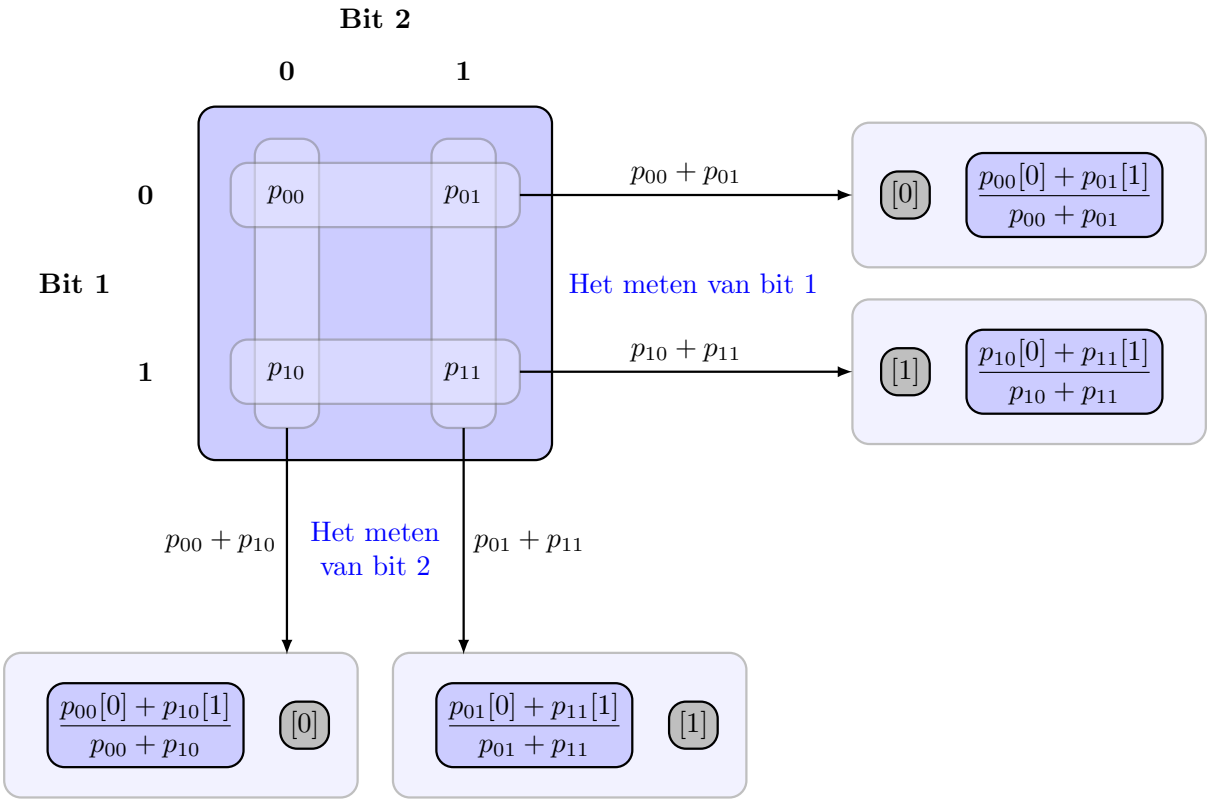
$$p_{10}[10] + p_{11}[11].$$

Dan negeren we het eerste bit omdat we al weten dat die waarde 1 is:

$$p_{10}[0] + p_{11}[1].$$

Als laatste, omdat deze twee kansen waarschijnlijk niet samen één zijn, delen we ze door hun som $p_{10} + p_{11}$:

$$\frac{p_{10}}{p_{10} + p_{11}}[0] + \frac{p_{11}}{p_{10} + p_{11}}[1]. \tag{3.16}$$



Figuur 3.3: De mogelijke uitkomsten wanneer we slechts een van de twee probabilistische bits meten. Voor elke mogelijke uitkomst geven we de waarschijnlijkheid (zwarte pijl) en de overgebleven toestand. Na de meting wordt het gemeten bit deterministisch (grijs) terwijl het andere bit probabilistisch blijft (lichtblauw).

Dit is de kansverdeling voor het tweede bit wanneer het eerste bit gemeten wordt en uitkomst 1 oplevert (zie de rechterkant van Fig. 3.3). De overige gevallen worden ook samengevat in Fig. 3.3.

Om te controleren of deze regels logisch zijn, kunnen we nagaan of het meten van het eerste bit en vervolgens het tweede bit tot dezelfde waarschijnlijkheden leidt als het direct meten van beide bits. De kans dat je bijvoorbeeld 1 krijgt van het eerste bit en 0 van het tweede bit zou gewoon p_{10} moeten zijn. Volgens Vgl. (3.13), krijgen we uit het eerste bit de uitkomst 1 met waarschijnlijkheid $p_{10} + p_{11}$ en, gegeven deze uitkomst, stelt Vgl. (3.16) dat het tweede bit 0 geeft met waarschijnlijkheid $p_{10}/(p_{10} + p_{11})$. Dus de totale kans om eerst 1 uit het eerste bit te krijgen en dan 0 uit het tweede bit is

$$(p_{10} + p_{11}) \times \frac{p_{10}}{p_{10} + p_{11}} = p_{10},$$

wat precies is wat we willen volgens Vgl. (3.4). De andere gevallen kunnen op dezelfde manier worden gecontroleerd.

Oefenopgave 3.2: Alice' muntje raden

Het probleem: Alice heeft drie muntjes die we u, q en r noemen, met de volgende kansverdelingen:

$$u = \begin{pmatrix} 1/2 \\ 1/2 \end{pmatrix}, \qquad q = \begin{pmatrix} 3/4 \\ 1/4 \end{pmatrix}, \qquad r = \begin{pmatrix} 1/3 \\ 2/3 \end{pmatrix}.$$

Alice gooit de muntjes op volgens het volgende stappenplan:

1. Ze gooit muntje u op.
2. Afhankelijk van de uitkomst gooit ze een van de andere twee muntjes op:
   (0) als u op 0 uitkwam, gooit ze q op;
   (1) als u op 1 uitkwam, gooit ze r op.
3. Alice vertelt haar vriend Bob wat de uitkomst (0 of 1) van de laaste worp is (maar ze vertelt niet of deze uitkomst van muntje q of r komt).

In deze situatie zijn er *twee* probabilistische bits: Alice' eerste worp en Alice' tweede worp (die precies hetzelfde is als Bob's probabilistische bit).

Vragen:

1. Wat is de kansverdeling van Alice' twee worpen?
2. Wat is de kansverdeling als Bob zijn probabilistische bit meet?
3. Als gegeven is dat Bob zijn bit meet en uitkomst 0 krijgt, is het dan waarschijnlijker dat Alice' eerste munt op 0 of 1 was uitgekomen? Hoe zit het als zijn uitkomst 1 is?

3.1.5 De SWAP-bewerking

Nu we weten hoe we individuele probabilistische bits kunnen manipuleren, willen we ook bewerkingen toepassen op meerdere bits tegelijk. Een van de eenvoudigste bewerkingen is de SWAP-bewerking die twee bits verwisselt:

$$\begin{aligned}\text{SWAP}\,[00] &= [00],\\ \text{SWAP}\,[01] &= [10],\\ \text{SWAP}\,[10] &= [01],\\ \text{SWAP}\,[11] &= [11].\end{aligned}$$

SWAP komt in feite neer op het omwisselen van de strings $[01]$ en $[10]$, terwijl de andere twee strings met rust worden gelaten. De bovenstaande vergelijkingen kunnen als volgt beknopter geschreven worden:

$$\text{SWAP}\,[a,b] = [b,a], \tag{3.17}$$

voor alle $a, b \in \{0,1\}$, waarbij we een komma gebruiken om de twee bits te onderscheiden. Zoals gebruikelijk kunnen we SWAP uitbreiden van deterministische naar probabilistische bits door lineariteit:

$$\begin{aligned}&\text{SWAP}\left(p_{00}[00] + p_{01}[01] + p_{10}[10] + p_{11}[11]\right)\\ &\quad= p_{00}[00] + p_{01}[10] + p_{10}[01] + p_{11}[11]\\ &\quad= p_{00}[00] + p_{10}[01] + p_{01}[10] + p_{11}[11].\end{aligned}$$

Oefenopgave 3.3: SWAP in de 4-vector notatie (optioneel)

Schrijf de werking van SWAP op twee probabilistische bits op in de 4-vector notatie.

3.1.6 De Controlled-NOT-bewerking

De bewerkingen NOT en SWAP kunnen samen gebruikt worden om individuele bits te veranderen en ze in een andere volgorde te zetten. Maar ze zorgen er niet voor dat de bits met

elkaar kunnen communiceren. Wat we willen is een nieuwe, meer geavanceerde bewerking die de waarde van een bit kan veranderen afhankelijk van de waarde van een andere bit. De eenvoudigste en belangrijkste van zulke bewerkingen is CNOT of de controlled-NOT-bewerking. Deze draait het doelbit om als het controlebit op 1 staat.

Als het eerste bit de controle is en het tweede bit het doel, schrijven we deze bewerking als $CNOT_{1\to 2}$. In dit geval geldt dus:

$$\begin{aligned} \mathrm{CNOT}_{1\to 2}\,[00] &= [00], \\ \mathrm{CNOT}_{1\to 2}\,[01] &= [01], \\ \mathrm{CNOT}_{1\to 2}\,[10] &= [11], \\ \mathrm{CNOT}_{1\to 2}\,[11] &= [10]. \end{aligned} \tag{3.18}$$

Dit komt neer op het verwisselen van de strings $[10]$ en $[11]$, terwijl de andere twee strings met rust worden gelaten.

Een andere manier om over $\mathrm{CNOT}_{1\to 2}$ na te denken is als een optelling: het telt de twee bits op (modulo 2) en slaat het resultaat op in het tweede bit. Dit kan als volgt beknopt worden samengevat:

$$\mathrm{CNOT}_{1\to 2}\,[a,b] = [a, a\oplus b], \tag{3.19}$$

voor elke $a, b \in \{0,1\}$, waarbij '$\oplus$' een *optelling modulo* 2 aangeeft en we een komma gebruiken om de waarden van de twee bits te scheiden. De regels voor optelling modulo 2 zijn als volgt:

$$0\oplus 0 = 1\oplus 1 = 0, \qquad\qquad 0\oplus 1 = 1\oplus 0 = 1. \tag{3.20}$$

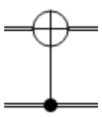
Soms wordt '$\oplus$' ook wel XOR- of de exclusieve OR-bewerking genoemd, omdat het resultaat 1 is als precies een van de twee bits 1 is. Zoals gebruikelijk kunnen we $\mathrm{CNOT}_{1\to 2}$ uitbreiden van deterministische naar probabilistische bits door lineariteit.

Daarnaast zullen we ook gebruik maken van controlled-NOT-bewerkingen waarbij het *tweede* bit de controle is en het *eerste* bit het doel. Op vergelijkbare manier duiden we deze bewerking aan met $CNOT_{2\to 1}$.

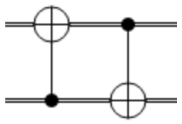
Oefenopgave 3.4: Controle en doel omdraaien

1. Geef op dezelfde manier als in Vgl. (3.19) een formule voor $\mathrm{CNOT}_{2\to 1}$.
2. Hoe kan je $\mathrm{CNOT}_{2\to 1}$ uitvoeren door alleen gebruik te maken van SWAP en $\mathrm{CNOT}_{1\to 2}$?

In QUIRKY kan je een controlled-NOT-bewerking maken door de volgende twee stappen uit te voeren. Sleep eerst het vakje $\boxed{\bullet}$ naar een van de draden - dit markeert de draad als het controle bit. Sleep vervolgens het vakje $\boxed{\oplus}$ naar de overeenkomstige locatie op de andere draad, wat de draad markeert als doelbit. Er ontstaat een verbinding om aan te geven dat je met succes een controlled-NOT-bewerking hebt gemaakt. Als we bijvoorbeeld het onderste bit als controle en het bovenste bit als doel kiezen, krijgen we het volgende resultaat:



Onthoud dat het onderste (!) bit het eerste bit is en het bovenste bit het tweede bit, dan zien we dat dit overeenkomt met de bewerking $\mathrm{CNOT}_{1\to 2}$. Hier volgt een wat ingewikkelder voorbeeld, waarbij we eerst $\mathrm{CNOT}_{1\to 2}$ toepassen en daarna $\mathrm{CNOT}_{2\to 1}$:



Huiswerkopdracht 3.2: SWAP vanuit CNOTs

Het probleem: Het is bijna middernacht, maar Bob is nog steeds aan het sleutelen aan zijn prototype van een probabilistische-bitcomputer die hij de volgende dag op school wil presenteren. Hij is zo druk bezig met het kalibreren van de randomness-generator dat hij vergeten is zijn papegaai Ziggy eten te geven. Om Bob's aandacht te trekken, stoot Ziggy Bob's koffiekopje om en morst de koffie over zijn op maat gemaakte toetsenbord, waarmee hij verschillende bewerkingen kan activeren. Bob is helemaal geschokt omdat de SWAP-knop niet meer werkt! Gelukkig werkt de CNOT-knop nog wel.

De vraag: Hoe kan Bob de SWAP-bewerking uitvoeren met alleen controlled-NOT-bewerkingen? (Als je wilt bewijzen dat twee bewerkingen hetzelfde zijn, hoef je dit alleen aan te tonen voor de basistoestanden vanwege uitbreiding door lineariteit).

Hint: Je hebt drie CNOT-bewerkingen nodig.

3.1.7 Productverdelingen

Laten we nu in wat meer detail kijken naar de toestanden van een twee-bits systeem. Stel bijvoorbeeld dat we twee probabilistische bits hebben, $q = q_0[0] + q_1[1]$ en $r = r_0[0] + r_1[1]$. Hoe kunnen we hieruit een toestand van een twee-bits systeem opbouwen? Ook al hebben we het niet uitdrukkelijk zo gezegd, hebben we deze vraag al besproken in §1.1.1, waar we de waarschijnlijkheid dat twee gebeurtenissen tegelijk plaatsvinden definieerden als het product van de waarschijnlijkheden van de twee afzonderlijke gebeurtenissen. Zo is bijvoorbeeld de waarschijnlijkheid dat de bits q en r in toestand $[00]$ zitten q_0r_0. Op dezelfde manier geldt dat de kans dat ze in de toestand $[01]$ zitten q_0r_1 is. Als we rekening houden met alle vier de mogelijkheden, krijgen we

$$q_0r_0[00] + q_0r_1[01] + q_1r_0[10] + q_1r_1[11]. \tag{3.21}$$

Met andere woorden, de vier kansen p_{ab} van de gecombineerde toestand

$$p_{00}[00] + p_{01}[01] + p_{10}[10] + p_{11}[11]$$

worden gegeven door de formule

$$p_{ab} = q_a r_b. \tag{3.22}$$

Je kan nagaan dat de waarschijnlijkheden bij het meten van het eerste bit gegeven worden door q, en die voor het tweede bit gegeven worden door r (je kan dit doen door gebruik te maken van de regel uit Fig. 3.2 en het feit dat $r_0 + r_1 = 1$ en $q_0 + q_1 = 1$). Maar onze twee-bits toestand heeft nog een speciale eigenschap: de waarden van de twee bits zijn van elkaar onafhankelijk. Dit betekent dat als we een van de twee bits meten, we geen informatie over het andere bit te weten komen. Je kunt dit controleren in de volgende oefening:

Oefenopgave 3.5: Onafhankelijke bits (optioneel)

Stel dat we de eerste van de twee bits van de toestand (3.21) meten en de uitkomst $a \in \{0, 1\}$ noemen. Laat zien dat de toestand van het tweede bit r is, onafhankelijk van de meetuitkomst a van het eerste bit. In andere woorden, het samenvoegen van de twee bits en vervolgens meten van het eerste bit heeft de toestand van het tweede bit helemaal niet beïnvloed (zoals het hoort)!

Laten we wat notatie introduceren om de speciale structuur van deze toestand wat duidelijker te maken. Laten we '$\otimes$' gebruiken om de bewerking aan te geven waarbij twee probabilistische bits samengevoegd worden en beschouwd worden als een enkel systeem bestaande uit twee bits:

$$q \otimes r = \big(q_0[0] + q_1[1]\big) \otimes \big(r_0[0] + r_1[1]\big) \tag{3.23}$$

Het symbool '$\otimes$' wordt het tensorproduct of *Kroneckerproduct* genoemd. Hoe kunnen we deze vreemde uitdrukking omzetten naar een werkelijke kansverdeling van twee bits, zoals in Vgl. (3.21)? Merk ten eerste op dat voor deterministische bits de bewerking '$\otimes$' simpelweg neerkomt op het samenvoegen van strings. Bijvoorbeeld,

$$[0] \otimes [1] = [01]. \tag{3.24}$$

Dit is logisch omdat een bit in toestand $[0]$ en een andere bit in toestand $[1]$ hetzelfde is als twee bits in toestand $[01]$. Om deze regel uit te breiden naar probabilistische bits, vragen we zoals altijd onze goede vriend lineariteit om hulp! Door lineariteit kunnen we de termen van Vgl. (3.23) uitschrijven en dan de samenvoegingsregel (Vgl. (3.24)) toepassen:

$$\begin{aligned}
&\big(q_0[0] + q_1[1]\big) \otimes \big(r_0[0] + r_1[1]\big) \\
&= q_0 r_0 \big([0] \otimes [0]\big) + q_0 r_1 \big([0] \otimes [1]\big) + q_1 r_0 \big([1] \otimes [0]\big) + q_1 r_1 \big([1] \otimes [1]\big) \\
&= q_0 r_0 [00] + q_0 r_1 [01] + q_1 r_0 [10] + q_1 r_1 [11].
\end{aligned}$$

Merk op dat we weer de verdeling van Vgl. (3.21) hebben gekregen. Met andere woorden, we krijgen de volgende identiteit tussen (3.23) en (3.21):

$$\big(q_0[0] + q_1[1]\big) \otimes \big(r_0[0] + r_1[1]\big) = q_0 r_0 [00] + q_0 r_1 [01] + q_1 r_0 [10] + q_1 r_1 [11]. \tag{3.25}$$

Dit betekent dat de manier waarop we het tensorproduct '$\otimes$' hebben gedefinieerd inderdaad consistent is met onze eerdere redenering dat de kansverdeling van een twee-bits systeem wordt verkregen door de kansen van individuele bits te vermenigvuldigen, zie Vgl. (3.22).

Merk op dat Vgl. (3.25) erg lijkt op de distributiewet voor optellen en vermenigvuldigen:

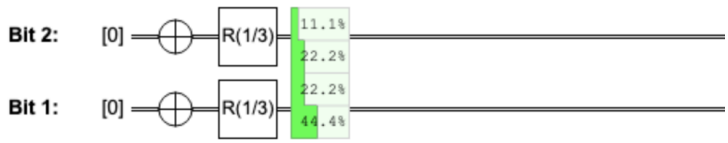
$$(a + b)(c + d) = ac + ad + bc + bd.$$

Het enige verschil is dat we in plaats van getallen vectoren hebben en in plaats van vermenigvuldiging de samenvoegingsregel $[a] \otimes [b] = [a, b]$. Een belangrijk verschil tussen samenvoegen en vermenigvuldigen is dat de volgorde van elementen belangrijk is bij samenvoegen. Over het algemeen is $[a, b] \neq [b, a]$ omdat het samenvoegen van $[a]$ en $[b]$ niet hetzelfde is als het samenvoegen van $[b]$ en $[a]$. Overigens kan je zelf controleren dat we het tensor product als volgt kunnen schrijven in vectornotatie:

$$\begin{pmatrix} q_0 \\ q_1 \end{pmatrix} \otimes \begin{pmatrix} r_0 \\ r_1 \end{pmatrix} = \begin{pmatrix} q_0 \begin{pmatrix} r_0 \\ r_1 \end{pmatrix} \\ q_1 \begin{pmatrix} r_0 \\ r_1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} q_0 r_0 \\ q_0 r_1 \\ q_1 r_0 \\ q_1 r_1 \end{pmatrix},$$

waarbij de tweede uitdrukking een blokvector is waarvan beide elementen vectoren zijn.

Het tensorproduct geeft een snelle manier om te begrijpen wat er gebeurt in (3.12), wat we hier voor het gemak herhalen:



Merk op dat we elk bit onafhankelijk klaarzetten in de toestand $\frac{1}{3}[0] + \frac{2}{3}[1]$. De gezamenlijke toestand van beide bits is dus

$$\left(\frac{1}{3}[0] + \frac{2}{3}[1]\right) \otimes \left(\frac{1}{3}[0] + \frac{2}{3}[1]\right) = \frac{1}{9}[00] + \frac{2}{9}[01] + \frac{2}{9}[10] + \frac{4}{9}[11] = \begin{pmatrix} 1/9 \\ 2/9 \\ 2/9 \\ 4/9 \end{pmatrix} \approx \begin{pmatrix} 11.1\% \\ 22.2\% \\ 22.2\% \\ 44.4\% \end{pmatrix},$$

wat in overeenstemming is met QUIRKY.

Huiswerkopdracht 3.3: Het tensorproduct

Vind twee probabilistische bits q en r waarvoor geldt dat

$$q \otimes r = 0.48[00] + 0.32[01] + 0.12[10] + 0.08[11].$$

Met het tensorproduct kunnen we ook compactere formules schrijven voor lokale operaties. Als M namelijk een bewerking op één bit is, dan is

$$M_1([a] \otimes [b]) = M[a] \otimes [b], \qquad M_2([a] \otimes [b]) = [a] \otimes M[b]. \tag{3.26}$$

Dit komt overeen met de formules uit §3.1.2.

Omdat de hierboven beschreven twee-bit verdelingen verkregen worden door het product te nemen van twee een-bit verdelingen, $p = q \otimes r$, worden ze producttoestanden of productverdelingen genoemd. Zoals je hebt gezien in Oefenopgave 3.5, zijn productverdelingen een ideaal model voor een situatie waarin twee bits onafhankelijk van elkaar ontstaan, zoals bij het gooien van twee munten. Maar is elke verdeling van twee bits een productverdeling? Interessant genoeg is dat niet het geval, zoals we in de volgende stuk zullen zien.

3.1.8 Gecorreleerde verdelingen

Sommige twee-bit verdelingen zijn *niet* een productverdeling – ze kunnen niet geschreven worden in de vorm van Vgl. (3.21) of Vgl. (3.23), welke waarden van q_a en r_b je ook kiest. We zeggen dat een verdeling gecorreleerd is als het geen productverdeling is. Een voorbeeld van een gecorreleerde verdeling is

$$\frac{1}{2}[00] + \frac{1}{2}[11]. \tag{3.27}$$

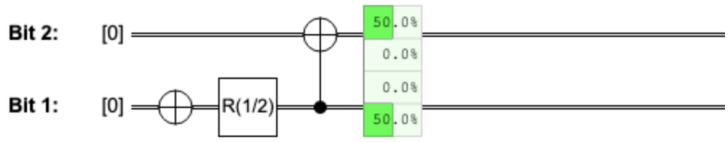
Om te zien dat dit geen productverdeling is, stellen we voor dat we een van de bits meten. De uitkomst a zal volledig willekeurig zijn, d.w.z. of 0 of 1 met elk een waarschijnlijkheid van 50%. Zodra we echter de uitkomst a weten, staat de toestand van het overige bit volledig vast – een meting zou dezelfde uitkomst opleveren met een waarschijnlijkheid van 100%. De toestand van het overige bit is dus $b = a$, wat afhangt van de uitkomst a van de meting van het andere bit. We zagen in Oefenopgave 3.5 dat dit niet het geval kan zijn voor een productverdeling. We hebben dus bewezen dat Vgl. (3.27) een gecorreleerde toestand beschrijft. Sterker nog, de twee bits zijn *perfect* gecorreleerd omdat beide meetuitkomsten volledig willekeurig zijn maar altijd identiek ($a = b$). Door deze eigenschap zeggen we dat Vgl. (3.27) een stel perfect gecorreleerde willekeurige bits beschrijft.

Gecorreleerde verdelingen komen van nature voor door een vorm van interactie. Stel bijvoorbeeld dat je een zuiver muntje opgooit, de uitkomst op een stuk papier schrijft, het papier in een envelop stopt en de envelop aan een vriend geeft. Vanuit het perspectief van je vriend (die de bereidingsprocedure kent, maar niet wat er op het papier in de envelop staat), wordt de toestand van je muntje (bit 1) en van het stuk papier in hun envelop (bit 2) beschreven door

$$\frac{1}{2}\otimes\boxed{\text{kop}}+\frac{1}{2}\otimes\boxed{\text{munt}}$$

wat niets anders is dan een leuke manier om de twee-bits toestand van Vgl. (3.27) op te schrijven.

Hoe kunnen we gecorreleerde toestanden maken in QUIRKY? Alleen lokale bewerkingen zijn niet genoeg, omdat die alleen producttoestanden kunnen maken. We kunnen wel de controlled-NOT-bewerking gebruiken om de twee bits met elkaar te laten communiceren, zoals in de volgende QUIRKY berekening:



Waarom werkt dit? Dat kan je in de volgende opgave uitwerken:

Oefenopgave 3.6: Perfect gecorreleerde willekeurige bits maken

Leg uit waarom de bovenstaande QUIRKY berekening de toestand $\frac{1}{2}[00]+\frac{1}{2}[11]$ maakt.

Om te bepalen of een willekeurige twee-bit verdeling

$$p = p_{00}[00]+p_{01}[01]+p_{10}[10]+p_{11}[11]$$

overeenkomt met een product- of een gecorreleerde toestand, kun je simpelweg de volgende waarde berekenen:

$$\Delta(p) = p_{00}p_{11}-p_{01}p_{10}. \tag{3.28}$$

Als $\Delta(p)=0$ dan is p een productverdeling; anders is het gecorreleerd. Bijvoorbeeld, voor de toestand in Vgl. (3.27) geldt dat

$$\Delta\left(\frac{1}{2}[00]+\frac{1}{2}[11]\right) = \frac{1}{2}\cdot\frac{1}{2}-0\cdot 0 = \frac{1}{4}\neq 0,$$

wat bevestigt dat het inderdaad gecorreleerd is en geen producttoestand. Als je nieuwsgierig bent waarom deze eenvoudige voorwaarde werkt, kan je de uitleg hieronder lezen. Maar omdat het niet essentieel is voor het begrijpen van de rest van de stof, mag je het ook gerust overslaan.

Om te bewijzen dat $\Delta(p)=0$ overeenkomt dat p een producttoestand is, moeten we twee dingen laten zien. Ten eerste moeten we laten zien dat als p een producttoestand is, $\Delta(p)=0$. Als $p=q\otimes r$ dan geldt inderdaad dat

$$\Delta(p) = q_0r_0q_1r_1-q_0r_1q_1r_0 = 0.$$

Hoe zit het met de omgekeerde stelling – kan $\Delta(p)=0$ ook als p geen producttoestand is? Het blijkt dat dit niet mogelijk is! Om dit te bewijzen nemen we aan dat $\Delta(p)=0$ en laten we zien dat $p=q\otimes r$, waarbij q en r probabilistische bits zijn. We kiezen deze twee bits als volgt:

$$\begin{aligned} q &= q_0[0]+q_1[1] = (p_{00}+p_{01})[0]+(p_{10}+p_{11})[1], \\ r &= r_0[0]+r_1[1] = (p_{00}+p_{10})[0]+(p_{01}+p_{11})[1]. \end{aligned} \tag{3.29}$$

Merk op dat q en r gewoon de verdelingen van uitkomsten van Fig. 3.3 zijn die we zouden krijgen als we het eerste of het tweede bit zouden meten. We gaan nu na of deze keuze van q en r inderdaad de toestand p oplevert:

$$\begin{aligned}
q \otimes r &= \big((p_{00}+p_{01})[0] + (p_{10}+p_{11})[1]\big) \otimes \big((p_{00}+p_{10})[0] + (p_{01}+p_{11})[1]\big) \\
&= (p_{00}+p_{01})(p_{00}+p_{10})[00] + \cdots \\
&= (p_{00}p_{00} + p_{00}p_{10} + p_{01}p_{00} + \boldsymbol{p_{01}p_{10}})[00] + \cdots \\
&= (p_{00}p_{00} + p_{00}p_{10} + p_{01}p_{00} + \boldsymbol{p_{00}p_{11}})[00] + \cdots \\
&= p_{00}(p_{00} + p_{10} + p_{01} + p_{11})[00] + \cdots \\
&= p_{00}[00] + \cdots \\
&= p.
\end{aligned}$$

Hier gebruikten we eerst Vgl. (3.25), daarna hebben we het tensorproduct uitgeschreven. Hierna hebben we $\Delta(p) = 0$ gebruikt om $\boldsymbol{p_{01}p_{10}}$ te vervangen door $\boldsymbol{p_{00}p_{11}}$ (zie Vgl. (3.28)) en tenslotte hebben we $p_{00} + p_{01} + p_{10} + p_{11} = 1$ gebruikt, wat geldt omdat p een kansverdeling is. De drie termen die we hebben afgekort met '…' kunnen op dezelfde manier worden berekend. Probeer zelf de details in te vullen en een ervan te controleren.

We hebben eerder gezien in Oefenopgave 3.5 dat p geen producttoestand kan zijn als het meten van het eerste bit de toestand van het tweede bit 'verstoort'. In feite is het omgekeerde ook waar, zoals je kunt laten zien in de volgende huiswerkopdracht.

Huiswerkopdracht 3.4: Onafhankelijkheid betekent product (optioneel)

Stel dat p een willekeurige twee-bit kansverdeling is waarbij de toestand van het tweede bit niet afhangt van de uitkomst van een meting van het eerste bit. Laat zien dat zo'n p een productverdeling is. Je kunt dit in twee stappen doen:

1. De meting op het eerste bit kan zowel uitkomst 0 als 1 opleveren. Gebruik Fig. 3.3 om de overige toestand van het tweede bit in deze twee gevallen te vergelijken en laat zien dat de volgende identiteiten gelden:

$$\frac{p_{00}}{p_{00}+p_{01}} = \frac{p_{10}}{p_{10}+p_{11}}, \qquad \frac{p_{01}}{p_{00}+p_{01}} = \frac{p_{11}}{p_{10}+p_{11}}.$$

2. Gebruik deze vergelijkingen om aan te tonen dat $\Delta(p) = 0$ vanwege Vgl. (3.28).

3.2 Twee quantumbits

De manier waarop we twee quantumbits kunnen beschrijven lijkt heel erg op de manier waarop we twee probabilistische bits beschrijven. Het belangrijkste verschil is dat we met amplitudes werken in plaats van waarschijnlijkheden, wat betekent dat ze negatief kunnen zijn en op een andere manier genormaliseerd worden (zie §2.1.1). Een algemene twee-qubit quantumtoestand ziet er als volgt uit:

$$|\psi\rangle = \psi_{00}\,|00\rangle + \psi_{01}\,|01\rangle + \psi_{10}\,|10\rangle + \psi_{11}\,|11\rangle \tag{3.30}$$

waarbij $\psi_{ij} \in [-1, 1]$ en

$$\psi_{00}^2 + \psi_{01}^2 + \psi_{10}^2 + \psi_{11}^2 = 1.$$

We noteren $|a, b\rangle$ in plaats van $[a, b]$ om duidelijk te maken dat we nu te maken hebben met quantumbits en niet met probabilistische bits. Net zoals we deden in Vgl. (3.3) voor probabilistische

bits, kunnen we de vier basistoestanden $|a,b\rangle$ identificeren met de vier basisvectoren

$$|00\rangle = \begin{pmatrix}1\\0\\0\\0\end{pmatrix}, \qquad |01\rangle = \begin{pmatrix}0\\1\\0\\0\end{pmatrix}, \qquad |10\rangle = \begin{pmatrix}0\\0\\1\\0\end{pmatrix}, \qquad |11\rangle = \begin{pmatrix}0\\0\\0\\1\end{pmatrix}. \tag{3.31}$$

Net als bij Vgl. (3.1), kan de algemene twee-qubit-toestand in Vgl. (3.30) gerepresenteerd worden door een 4-vector

$$\begin{pmatrix}\psi_{00}\\\psi_{01}\\\psi_{10}\\\psi_{11}\end{pmatrix}.$$

Het wordt alleen al snel omslachtig om met zulke vectoren te werken. Ook zijn ze niet meer zo makkelijk te visualiseren, dus we zullen meestal werken met de $|a,b\rangle$ notatie van Vgl. (3.30).

Een ander voordeel van deze notatie is dat het veel eenvoudiger is om quantumsystemen te combineren. Denk terug aan §3.1.7 waarin we de tensorproductbewerking '$\otimes$' gebruikten om twee onafhankelijke probabilistische bits te combineren tot een gezamenlijk twee-bits systeem. Dezelfde bewerking (maar dan met $|a\rangle$ in plaats van $[a]$) werkt ook voor qubits. Net zoals we de basisvectoren van probabilistische bits hebben gecombineerd in Vgl. (3.24), kunnen we dit ook doen voor qubits:

$$|0\rangle \otimes |0\rangle = |00\rangle, \qquad |0\rangle \otimes |1\rangle = |01\rangle, \qquad |1\rangle \otimes |0\rangle = |10\rangle, \qquad |1\rangle \otimes |1\rangle = |11\rangle. \tag{3.32}$$

Merk op dat dit hetzelfde is als gewoon de bitstrings aan elkaar te plakken. Dit geeft je een andere manier om de vier twee-qubit basisvectoren in Vgl. (3.31) te zien.

We kunnen de tensorproductbewerking uitbreiden om twee willekeurige één-qubit toestanden $|\alpha\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$ en $|\beta\rangle = \beta_0|0\rangle + \beta_1|1\rangle$ te combineren op de volgende manier:

$$\begin{aligned}|\alpha\rangle \otimes |\beta\rangle &= (\alpha_0|0\rangle + \alpha_1|1\rangle) \otimes (\beta_0|0\rangle + \beta_1|1\rangle)\\ &= \alpha_0\beta_0|00\rangle + \alpha_0\beta_1|01\rangle + \alpha_1\beta_0|10\rangle + \alpha_1\beta_1|11\rangle.\end{aligned} \tag{3.33}$$

Twee-qubit toestanden van deze vorm worden producttoestanden genoemd. In §3.2.3 bespreken we hoe we deze toestanden kunnen maken met behulp van quantumbewerkingen. Belangrijk is dat, net als voor twee probabilistische bits, niet alle twee-qubit toestanden producttoestanden zijn.

Oefenopgave 3.7: Tensorproduct en producttoestanden

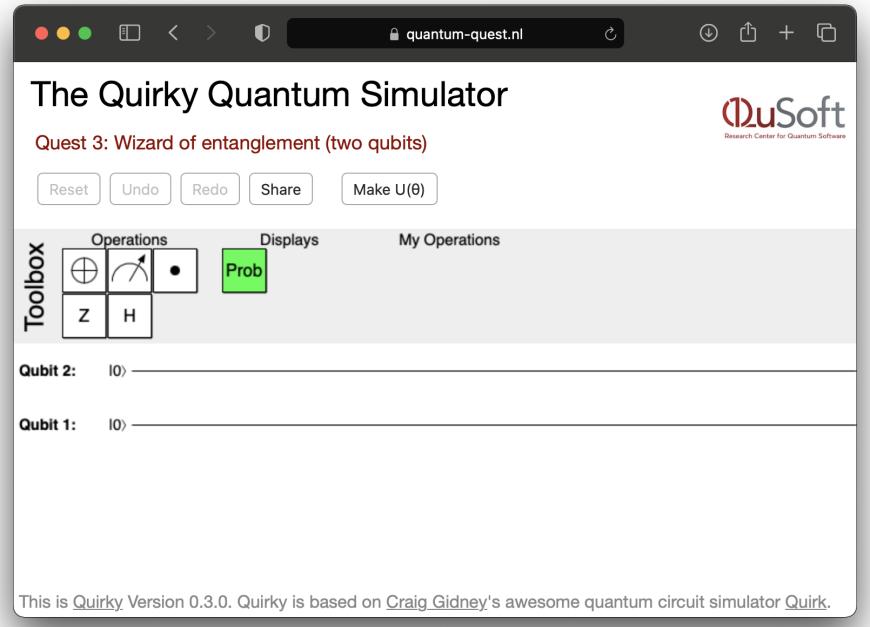
Gebruik de toestanden $|+\rangle$ en $|-\rangle$ uit Oefenopgave 2.1.

1. Schrijf $|+\rangle \otimes |-\rangle$ in dezelfde vorm als Vgl. (3.30).
2. Is de toestand $\frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$ een producttoestand?

Hieronder bespreken we de regels voor het meten en bewerken van twee quantumbits. Ook al zijn deze regels volledig vergelijkbaar met het geval van twee probabilistische bits, zullen we onderweg toch een aantal nieuwe en verrassende verschijnselen tegenkomen. Gelukkig kan QUIRKY ons deze week ook helpen om de wereld van twee quantumbits te verkennen! Ga om te beginnen naar:

https://www.quantum-quest.org/quirky

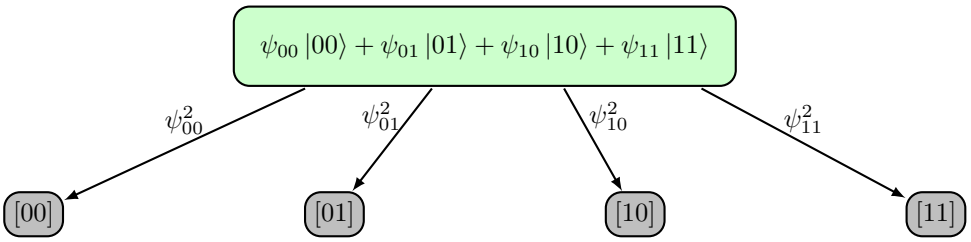
en klik op 'Quest 3' en dan op 'Two Qubits'. Je webbrowser zal er ongeveer hetzelfde uitzien als Fig. 3.4. Vergeleken met de QUIRKY van vorige week, hebben we nu twee *quantumbits*, die zijn ingesteld in de toestand $|00\rangle$. Daarnaast zijn er drie nieuwe vakjes: $\boxed{Z}$, $\boxed{H}$, en $\boxed{\bullet}$ (en het mysterievakje is weer verdwenen). Deze zullen we in de rest van dit hoofdstuk bespreken.



Figuur 3.4: QUIRKY voor Quest 3.

3.2.1 Het meten van twee qubits

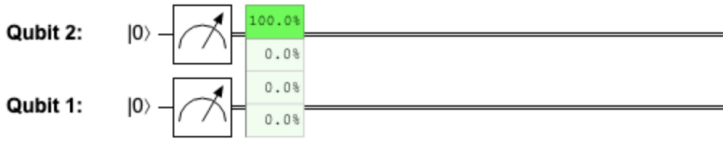
Een meting voor twee-qubit toestanden wordt op een vergelijkbare manier gedefinieerd als Vgl. (2.7) voor één-qubit toestanden, met als verschil dat je als uitkomst twee bits krijgt met de volgende waarschijnlijkheden:



(3.34)

Hier gebruiken we lichtgroen voor quantumbits en grijs voor de twee deterministische bits die je krijgt uit de meting.

Hoe kunnen we beide qubits meten in QUIRKY? We voegen gewoon twee metingen toe, een voor elke qubit, en gebruiken de Probability Display net als voorheen:



3.2.2 Lokale bewerkingen

Als je twee qubits hebt, werkt een *lokale bewerking* maar op één van de qubits. Elke één-qubit bewerking kan worden gebruikt als een lokale bewerking die werkt op één van de twee qubits, net zoals het geval was voor probabilistische bits in §3.1.2. Als je bijvoorbeeld terugdenkt aan de één-bit NOT-bewerking van Vgl. (1.9) en hoe we die hebben omgezet in lokale NOT-bewerkingen in Vgl. (3.6) en (3.7), dan kunnen we precies hetzelfde doen voor de één-*qubit*

NOT. De verkregen lokale quantum NOT-bewerkingen lijken hier erg op:

$$\begin{aligned}
&\mathrm{NOT}_1\,|00\rangle = |10\rangle\,, \quad \mathrm{NOT}_1\,|01\rangle = |11\rangle\,, \quad \mathrm{NOT}_1\,|10\rangle = |00\rangle\,, \quad \mathrm{NOT}_1\,|11\rangle = |01\rangle\,,\\
&\mathrm{NOT}_2\,|00\rangle = |01\rangle\,, \quad \mathrm{NOT}_2\,|01\rangle = |00\rangle\,, \quad \mathrm{NOT}_2\,|10\rangle = |11\rangle\,, \quad \mathrm{NOT}_2\,|11\rangle = |10\rangle\,.
\end{aligned}$$

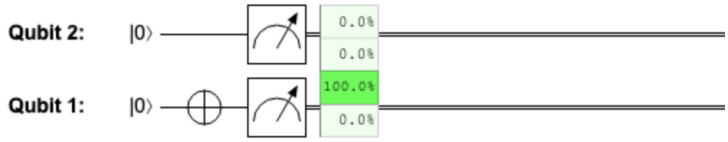
Het enige verschil is dat we de bitnotatie $[ab]$ hebben vervangen door de qubitnotatie $|ab\rangle$.

Laten we dit eens toepassen door alle vier mogelijke twee-qubit basistoestanden op te bouwen uit $|00\rangle$. Dit kan altijd gedaan worden door een reeks NOT-bewerkingen:

$$|00\rangle = |00\rangle\,, \quad |01\rangle = \mathrm{NOT}_2\,|00\rangle\,, \quad |10\rangle = \mathrm{NOT}_1\,|00\rangle\,, \quad |11\rangle = \mathrm{NOT}_2\,\mathrm{NOT}_1\,|00\rangle\,.$$

Merk op dat we in het laatste geval de twee NOT-bewerkingen in de omgekeerde volgorde hadden kunnen uitvoeren, omdat beide gevallen neerkomen op het omdraaien van beide bits. Met andere woorden, $\mathrm{NOT}_2\mathrm{NOT}_1 = \mathrm{NOT}_1\mathrm{NOT}_2$.

Om een lokale bewerking in QUIRKY toe te passen, plaats je het corresponderende vakje op de eerste of de tweede draad. De volgende procedure maakt bijvoorbeeld de toestand $|10\rangle$ en toont de waarschijnlijkheden bij het meten van beide qubits:



Dit is logisch omdat de onderste draad in QUIRKY overeenkomt met het eerste qubit.

Dit is het makkelijkst te formuleren met behulp van de tensorproductnotatie uit Vgl. (3.33) en lineariteit. Als U een arbitraire één-qubit bewerking is , definiëren we U_1 als de twee-qubit bewerking die werkt op elke basisvector $|a,b\rangle = |a\rangle \otimes |b\rangle$, waarbij $a, b \in \{0,1\}$, als volgt:

$$U_1\,|a,b\rangle = U|a\rangle \otimes |b\rangle\,. \tag{3.35}$$

Voor de duidelijkheid, de rechterkant betekent $(U|a\rangle) \otimes |b\rangle$, oftewel het tensorproduct van de toestand $U\,|a\rangle$ en de toestand $|b\rangle$. Dit is intuïtief, want het betekent gewoon dat we U toepassen op het eerste quantumbit en het tweede quantumbit met rust laten.

Om U_1 toe te passen op een arbitraire twee-qubit toestand breiden we het uit door lineariteit. Net als in de eerste week betekent dit dat we eerst $|\psi\rangle$ uitschrijven in de vorm van Vgl. (3.30) en dan de bewerking toepassen op elke basisvector. Dat wil zeggen,

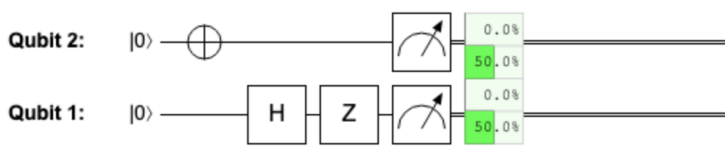
$$U_1\,|\psi\rangle = \psi_{00}\,U_1\,|00\rangle + \psi_{01}\,U_1\,|01\rangle + \psi_{10}\,U_1\,|10\rangle + \psi_{11}\,U_1\,|11\rangle$$

en nu kunnen we Vgl. (3.35) gebruiken voor elk van de vier termen. Op dezelfde manier definiëren we U_2 door

$$U_2\,|a,b\rangle = |a\rangle \otimes U\,|b\rangle \tag{3.36}$$

en breiden dit uit door lineariteit. Dit is vergelijkbaar met de formules in Vgl. (3.26) voor gewone bits.

Naast de NOT-bewerking zijn er twee belangrijke quantumoperaties die we voortdurend zullen gebruiken: de Z-bewerking uit Vgl. (2.12) en de Hadamard-bewerking uit Vgl. (2.20). Omdat deze zo belangrijk zijn, hebben we ze hun eigen vakjes gegeven in QUIRKY, namelijk $\boxed{Z}$ en $\boxed{H}$. De volgende serie bewerkingen past bijvoorbeeld een NOT toe op het tweede (!) qubit, dan een Hadamard op het eerste qubit en tenslotte een Z, weer op het eerste qubit:



(3.37)

Wat is de toestand die we op deze manier krijgen? De wiskundige uitdrukking voor deze toestand is

$$Z_1 H_1 \text{NOT}_2 \left|00\right\rangle . \tag{3.38}$$

Merk op dat, in tegenstelling tot de grafische voorstelling van (3.37), de invoertoestand $\left|00\right\rangle$ in deze uitdrukking aan de rechterkant staat. Hierdoor lijkt de volgorde van bewerkingen omgedraaid. Maar zowel (3.37) als Vgl. (3.38) beschrijven hetzelfde proces – de eerste bewerking die wordt toegepast op $\left|00\right\rangle$ is NOT_2, dan H_1 en tenslotte Z_1. Het enige verschil tussen (3.37) en Vgl. (3.38) is de manier waarop de richting van de tijd wordt weergegeven: het gaat van links naar rechts in (3.37) en van rechts naar links in Vgl. (3.38). Helaas zijn deze twee tegenstrijdige conventies standaard in quantum computing, we kunnen er niets aan doen. Je moet dus voorzichtig zijn met het vertalen van QUIRKY-plaatjes naar vergelijkingen en vice versa!

Oefenopgave 3.8: Heeft QUIRKY gelijk?

Bepaal de twee-qubit toestand van Vgl. (3.38) (dus vlak voor de meting in (3.37)). Bereken de kans van de meetresultaten en vergelijk je resultaat met QUIRKY.

Vorige week, in §2.4.3, hebben we besproken dat elke bewerking op een enkele qubit óf een rotatie $U(\theta)$ óf een spiegeling $V(\theta) = \text{NOT}\, U(\theta)$ is. Omdat we arbitraire rotaties kunnen maken in QUIRKY (zie §2.4.1 in de aantekeningen van vorige week), kunnen we dus arbitraire lokale operaties toepassen op elk van de qubits van QUIRKY.

De regels van Vgl. (3.35) en (3.36) werken niet alleen voor basisvectoren, maar in feite voor alle producttoestanden. Dat wil zeggen, als $\left|\alpha\right\rangle$ en $\left|\beta\right\rangle$ arbitraire één-qubit toestanden zijn , dan geldt

$$U_1 \left(\left|\alpha\right\rangle \otimes \left|\beta\right\rangle\right) = U \left|\alpha\right\rangle \otimes \left|\beta\right\rangle , \tag{3.39}$$
$$U_2 \left(\left|\alpha\right\rangle \otimes \left|\beta\right\rangle\right) = \left|\alpha\right\rangle \otimes U \left|\beta\right\rangle . \tag{3.40}$$

Oefenopgave 3.9: Lokale bewerkingen op producttoestanden (optioneel)

Kan je laten zien dat Vgl. (3.39) of (3.40) klopt?

3.2.3 Parallelle bewerkingen

Als je een bewerking toepast op het eerste qubit en een andere op het tweede qubit, dan maakt de volgorde van deze twee bewerkingen niet uit. Dat wil zeggen, als U en V willekeurige één-qubit bewerkingen zijn , dan geldt dat

$$U_1 V_2 = V_2 U_1 . \tag{3.41}$$

Dit is al een vrij intuitief resultaat, maar we kunnen het ook wiskundig laten zien met Vgl. (3.39) en (3.40). Voor elke basistoestand $\left|a, b\right\rangle$, waarbij $a, b \in \{0, 1\}$, geldt

$$U_1 V_2 \left|a, b\right\rangle = U_1 \left(\left|a\right\rangle \otimes V \left|b\right\rangle\right) = U \left|a\right\rangle \otimes V \left|b\right\rangle = V_2 \left(U \left|a\right\rangle \otimes \left|b\right\rangle\right) = V_2 U_1 \left|a, b\right\rangle ,$$

dus Vgl. (3.41) volgt door lineariteit.

Aangezien de twee bewerkingen op verschillende qubits werken, kunnen we ze zelfs parallel uitvoeren! Dit wijst op de introductie van een nieuwe notatie voor de bewerking in Vgl. (3.41):

$$U \otimes V .$$

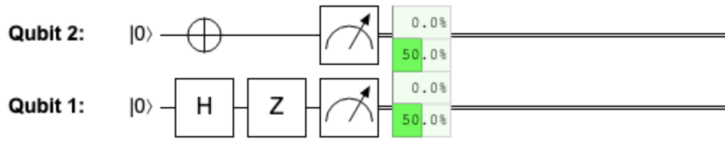
We maken opnieuw gebruik van het tensorproductsymbool dat we eerst introduceerden om twee onafhankelijke één-qubitstoestanden te combineren tot een twee-qubitstoestand. Dezelfde

interpretatie geldt ook voor quantumbewerkingen: $U \otimes V$ staat voor de gecombineerde bewerking die bestaat uit het toepassen van U en V op twee verschillende delen van een groter systeem. Deze notatie is vooral handig omdat het mooi aansluit bij het originele tensorproduct voor toestanden:

$$(U \otimes V)(|\alpha\rangle \otimes |\beta\rangle) = U|\alpha\rangle \otimes V|\beta\rangle\,. \tag{3.42}$$

Deze vergelijking zegt dat als je twee onafhankelijke toestanden hebt en je past een bewerking toe die op elk van de twee toestanden onafhankelijk werkt, dat je dan uiteindelijk elk van de twee bewerkingen op de bijbehorende toestand toepast. We noemen $U \otimes V$ het tensorproduct van twee quantumbewerkingen of een parallelle bewerking.

Met QUIRKY kunnen we lokale quantumbewerkingen parallel uitvoeren. We hadden bijvoorbeeld de volgorde van bewerkingen in Vgl. (3.37) ook kunnen schrijven als

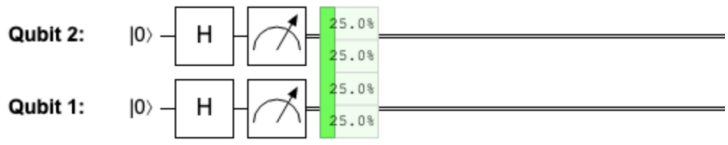


waarbij de NOT-bewerking en de Hadamard-bewerking nu parallel worden toegepast.

Laten we kijken naar een ander voorbeeld. Wat gebeurt er als we H toepassen op beide qubits? Deze bewerking wordt aangeduid met $H \otimes H$ en werkt volgens Vgl. (2.20) en (3.42) als volgt:

$$\begin{aligned}
(H \otimes H)|00\rangle &= (H|0\rangle) \otimes (H|0\rangle) \\
&= \left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle)\right) \otimes \left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle)\right) \\
&= \frac{1}{2}|0\rangle \otimes |0\rangle + \frac{1}{2}|0\rangle \otimes |1\rangle + \frac{1}{2}|1\rangle \otimes |0\rangle + \frac{1}{2}|1\rangle \otimes |1\rangle \\
&= \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle).
\end{aligned}$$

De verkregen toestand wordt de uniforme superpositie over twee qubits genoemd, omdat het bestaat uit elk van de twee-qubit basisvectoren met een gelijke amplitude. Als we een meting uitvoeren op deze toestand kunnen we alle vier de uitkomsten met gelijke waarschijnlijkheid krijgen. We kunnen dit eenvoudig controleren met QUIRKY:



Oefenopgave 3.10: Een producttoestand bouwen

1. Schrijf $\frac{1}{2}(|00\rangle + |01\rangle - |10\rangle - |11\rangle)$ als een tensorproduct van twee één-qubitstoestanden.
2. Hoe kan je deze toestand maken door een aantal lokale bewerkingen op $|00\rangle$ toe te passen?
3. Voer de reeks bewerkingen uit stap 2 uit in QUIRKY.

Vgl. (3.42) laat zien dat als we een parallelle bewerking toepassen op een producttoestand, we een andere producttoestand krijgen. In feite kunnen we, als we starten met $|00\rangle$ op deze manier elke producttoestand krijgen. Dit geeft een eenvoudige interpretatie aan de producttoestanden: dit zijn precies de toestanden die we kunnen krijgen door een arbitraire parallelle bewerking toe te passen op twee qubits die in de toestand $|00\rangle$ zitten.

Huiswerkopdracht 3.5: Producttoestanden uit lokale bewerkingen

Laat zien dat elke producttoestand gemaakt kan worden door een parallele quantumrotatie (dus een bewerking van de vorm $U(\theta) \otimes U(\phi)$) toe te passen op $|00\rangle$. $U(\theta)$ geeft hier een rotatie met hoek θ aan, zoals in Vgl. (2.13).

Hint: In Vgl. (2.5) hebben we gezien dat de meest algemene één-qubit toestand eruit ziet als $|\psi(\theta)\rangle$.

We sluiten onze beschrijving van lokale bewerkingen af met wat algemene opmerkingen. Ten eerste is het niet moeilijk om te controleren dat

$$(U \otimes V)(U' \otimes V') = UU' \otimes VV', \tag{3.43}$$

voor vier willekeurige één-qubit bewerkingen U, U', V, V'. Kun je een plaatje maken om te visualiseren wat hier gebeurt?

We kunnen nu gebruik maken van de identiteitsbewerking I uit Vgl. (2.18), die werkt als $I|\psi\rangle = |\psi\rangle$ (dit kan je uitbreiden naar I_1 en I_2 op twee qubits). Dit is op zichzelf nogal nutteloos, maar kan handig gebruikt worden in de tensorproductnotatie. We kunnen hiermee bijvoorbeeld zeggen dat

$$U_1 = U \otimes I \qquad \text{en} \qquad V_2 = I \otimes V,$$

wat het heel duidelijk maakt dat bijvoorbeeld U_1 de U-bewerking toepast op het eerste qubit en niets op het tweede qubit. De identiteit $U_1V_2 = V_2U_1$ uit Vgl. (3.41) kunnen we nu bijvoorbeeld schrijven als

$$(U \otimes I)(I \otimes V) = U \otimes V = (I \otimes V)(U \otimes I) \tag{3.44}$$

wat vrij intuïtief is.

Je zou je kunnen afvragen of de volgorde van de bewerkingen er eigenlijk wel toe doet, ook al passen we ze toe op *dezelfde* qubit? Als we twee rotaties beschouwen dan volgt uit Vgl. (2.15) dat de volgorde niet belangrijk is omdat

$$U(\theta)U(\theta') = U(\theta + \theta') = U(\theta' + \theta) = U(\theta')U(\theta).$$

Maar als U en V twee arbitraire één-qubit bewerkingen zijn (vooral als één van hen een spiegeling is), dan hangt hun samenstelling in het algemeen af van de volgorde (zie Oefenopgave 3.11 hieronder). Dat wil zeggen,

$$UV \neq VU.$$

Dit probleem blijft natuurlijk ook bestaan als je een andere qubit in de buurt hebt, maar toch met beide bewerkingen op dezelfde qubit werkt. Het geldt dus dat

$$(U \otimes I)(V \otimes I) = UV \otimes I \neq VU \otimes I = (V \otimes I)(U \otimes I). \tag{3.45}$$

We kunnen dit ook schrijven als $U_1V_1 \neq V_1U_1$ (en hetzelfde voor de situatie waarbij we beide bewerkingen toepassen op het tweede qubit).

Om het verschil tussen Vgl. (3.44) en (3.45) wat intuïtiever te maken, kan je je voorstellen dat U staat voor 'een sok aantrekken' en V voor 'een schoen aantrekken'. Het is duidelijk dat wanneer U en V op dezelfde voet worden toegepast, je verschillende resultaten krijgt, afhankelijk van de volgorde! Maar als je U en V op verschillende voeten toepast (je gebruikt bijvoorbeeld $U \otimes I$ en $I \otimes V$), dan krijg je hetzelfde resultaat, ongeacht de volgorde van de twee bewerkingen. Hoe dan ook, als je goed gekleed wilt zijn, moet je eerst $(U \otimes U)$ toepassen en daarna $(V \otimes V)$.

Oefenopgave 3.11: De volgorde is belangrijk

Laat zien dat $HZ \neq ZH$.

3.2.4 'Controlled' bewerkingen

Om meer te kunnen produceren dan producttoestanden, hebben we een bewerking nodig die de wisselwerking tussen twee quantumbits mogelijk maakt. Zoals eerder (zie Vgl. (2.18) voor probabilistische bits), zullen we hiervoor een *controlled-NOT*-bewerking gebruiken, die we op dezelfde manier definiëren als Vgl. (3.18) en (3.19):

$$\begin{aligned} \text{CNOT}_{1\to 2}\ |00\rangle &= |00\rangle\,, \\ \text{CNOT}_{1\to 2}\ |01\rangle &= |01\rangle\,, \\ \text{CNOT}_{1\to 2}\ |10\rangle &= |11\rangle\,, \\ \text{CNOT}_{1\to 2}\ |11\rangle &= |10\rangle\,, \end{aligned} \tag{3.46}$$

of korter gezegd,

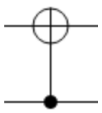
$$\text{CNOT}_{1\to 2}\ |a,b\rangle = |a, a\oplus b\rangle\,. \tag{3.47}$$

De bewerking $\text{CNOT}_{1\to 2}$ wisselt dus bij basistoestanden het tweede qubit om, afhankelijk van de waarde van het eerste qubit. We kunnen ook een bewerking $\text{CNOT}_{2\to 1}$ definiëren die het tweede qubit als controle gebruikt en het eerste als doel, dus,

$$\text{CNOT}_{2\to 1}\ |a,b\rangle = |a\oplus b, b\rangle\,. \tag{3.48}$$

Zoals gebruikelijk breiden we deze formules lineair uit naar arbitraire twee-qubit toestanden.

In QUIRKY kun je een controlled-NOT-bewerking voor quantumbits maken op dezelfde manier als je geleerd hebt voor gewone bits – zie §3.1.6 voor het geval je niet meer weet hoe. De bewerking $\text{CNOT}_{1\to 2}$ voor quantumbits ziet er dus net zo uit als voorheen:



Veel van de dingen die we bewezen hebben voor probabilistische bits zijn nog steeds waar voor quantumbits. Je oplossing van Huiswerkopdracht 3.2 laat je bijvoorbeeld net zo goed twee quantumbits verwisselen! Een ander voorbeeld hiervan is het feit dat twee keer dezelfde controlled-NOT-bewerking uitvoeren neerkomt op helemaal niets doen. Voor $\text{CNOT}_{1\to 2}$ geldt dit bijvoorbeeld omdat

$$\text{CNOT}_{1\to 2}\,\text{CNOT}_{1\to 2}\,|a,b\rangle = \text{CNOT}_{1\to 2}\,|a, a\oplus b\rangle = |a, a\oplus a\oplus b\rangle = |a,b\rangle$$

omdat $a\oplus a = 0$ voor elke $a \in \{0,1\}$. Het gevolg is dat de controlled-NOT-bewerking de inverse is van zichzelf:

$$\text{CNOT}_{1\to 2}^{-1} = \text{CNOT}_{1\to 2} \tag{3.49}$$

waarbij M^{-1} de inverse van de bewerking M is (zie §2.4.2).

Als je een beetje rond hebt gespeeld met QUIRKY, heb je misschien gemerkt dat je $\boxed{\bullet}$ kunt combineren met elke willekeurige één-qubit bewerking, niet alleen met de NOT-bewerking. We kunnen namelijk een controlled-U-bewerking definiëren voor elke één-qubit bewerking U.

Deze worden aangeduid met $CU_{1\to 2}$ en $CU_{2\to 1}$, afhankelijk van welke qubit de controle is en welke het doel. $CU_{1\to 2}$ wordt bijvoorbeeld als volgt gedefinieerd op de vier basistoestanden:

$$\begin{aligned} CU_{1\to 2}\,|00\rangle &= |0\rangle \otimes |0\rangle\,, \\ CU_{1\to 2}\,|01\rangle &= |0\rangle \otimes |1\rangle\,, \\ CU_{1\to 2}\,|10\rangle &= |1\rangle \otimes U\,|0\rangle\,, \\ CU_{1\to 2}\,|11\rangle &= |1\rangle \otimes U\,|1\rangle\,. \end{aligned}$$

Je kunt snel nagaan dat als $U = NOT$ dit neerkomt op onze definitie van $\text{CNOT}_{1\to 2}$.

3.2.5 Verstrengelde toestanden

In Vgl. (3.33) gebruikten we het tensorproduct om een twee-qubitstoestand op te bouwen uit twee één-qubitstoestanden. In §3.2.3 hebben we gezien dat deze producttoestanden precies de toestanden zijn die gemaakt kunnen worden door locale quantumbewerkingen toe te passen op $|00\rangle = |0\rangle \otimes |0\rangle$ (wat zelf een producttoestand is). Er bestaan ook twee-qubitstoestanden die *niet* producttoestanden zijn. We noemen deze toestanden verstrengeld, en zoals we zullen zien zijn ze erg belangrijk voor quantumcomputing.

Hoe kunnen we bepalen of een toestand een producttoestand is of niet? Ook al worden quantumtoestanden bepaald door amplitudes en niet door waarschijnlijkheden, kunnen we alsnog dezelfde methode gebruiken als we in §3.1.8 hebben gebruikt om te bepalen of een kansverdeling gecorreleerd is. Gegeven een twee-qubitstoestand in de vorm (3.30), berekenen we eerst met Vgl. (3.28)

$$\Delta(|\psi\rangle) = \psi_{00}\psi_{11} - \psi_{01}\psi_{10}. \tag{3.50}$$

$|\psi\rangle$ is een producttoestand als en alleen als $\Delta(|\psi\rangle) = 0$. Op deze manier zijn verstrengelde toestanden vergelijkbaar met gecorreleerde kansverdelingen.

Een eenvoudig maar erg belangrijk voorbeeld van een verstrengelde twee-qubitstoestand is

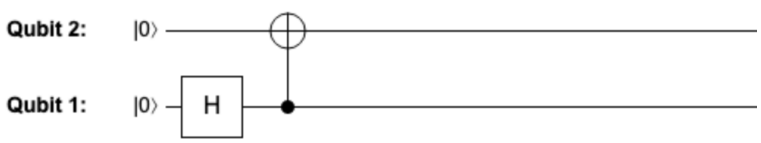
$$\left|\Phi^+\right\rangle = \frac{1}{\sqrt{2}}\,|00\rangle + \frac{1}{\sqrt{2}}\,|11\rangle\,. \tag{3.51}$$

Deze toestand is vergelijkbaar met de perfect gecorreleerde random bits uit Vgl. (3.27). De toestand is verstrengeld omdat

$$\Delta(\left|\Phi^+\right\rangle) = \frac{1}{\sqrt{2}}\cdot\frac{1}{\sqrt{2}} - 0\cdot 0 = \frac{1}{2} \neq 0.$$

We noemen $|\Phi^+\rangle$ de maximaal verstrengelde toestand van twee qubits (al is de reden voor deze naam en de bijzondere notatie op dit moment nog niet erg duidelijk).

Hoe kunnen we verstrengelde toestanden maken? Net als toen we gecorreleerde toestanden uit twee bits wilden maken, kunnen we de controlled-NOT-bewerking gebruiken om twee quantumbits met elkaar te laten wisselwerken. De volgende volgorde van operaties maakt bijvoorbeeld de maximaal verstrengelde toestand $|\Phi^+\rangle$:



Laten we dit snel uitwerken:

$$\text{CNOT}_{1\to 2}\,(H\otimes I)\,|00\rangle = \text{CNOT}_{1\to 2}\left(\frac{1}{\sqrt{2}}\,|00\rangle + \frac{1}{\sqrt{2}}\,|10\rangle\right) = \frac{1}{\sqrt{2}}\,|00\rangle + \frac{1}{\sqrt{2}}\,|11\rangle\,.$$

Merk op dat als je CNOT direct op $|00\rangle$ toepast, of op welke andere basistoestand dan ook, dit *niet* zou hebben gewerkt (zie Vgl. (3.46)).

Huiswerkopdracht 3.6: Een verstrengelde toestand

1. Laat zien dat de toestand $|\psi\rangle = \frac{1}{2}\,|01\rangle + \frac{\sqrt{3}}{2}\,|10\rangle$ verstrengeld is.
   Hint: Bereken Vgl. (3.50).
2. Maak een reeks bewerkingen in QUIRKY die de toestand $|\psi\rangle$ maakt.
   Hint: Je moet misschien een geschikte rotatie maken.
3. Wat zijn de kansen op de meetuitkomsten bij het meten van beide qubits van $|\psi\rangle$? Gebruik QUIRKY om je resultaat te bevestigen.

De maximaal verstrengelde toestand in Vgl. (3.51) is een lid van een familie van vier toestanden, de zogenaamde Bell-toestanden. De Bell-toestanden zijn als volgt gedefinieerd:

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}\,|00\rangle + \frac{1}{\sqrt{2}}\,|11\rangle\,, \tag{3.52}$$

$$|\Phi^-\rangle = \frac{1}{\sqrt{2}}\,|00\rangle - \frac{1}{\sqrt{2}}\,|11\rangle\,, \tag{3.53}$$

$$|\Psi^+\rangle = \frac{1}{\sqrt{2}}\,|01\rangle + \frac{1}{\sqrt{2}}\,|10\rangle\,, \tag{3.54}$$

$$|\Psi^-\rangle = \frac{1}{\sqrt{2}}\,|01\rangle - \frac{1}{\sqrt{2}}\,|10\rangle\,. \tag{3.55}$$

Ze zijn vernoemd naar John Steward Bell, die als een van de eersten de opmerkelijke eigenschappen van quantumverstrengeling herkende. Hoe kunnen we deze vier Bell toestanden maken? We zagen hierboven dat we $|\Phi^+\rangle$ kunnen maken door een Hadamard- en een CNOT-bewerking toe te passen op de basistoestand $|00\rangle$. Je kunt nagaan dat de andere drie Bell-toestanden op dezelfde manier gevormd kunnen worden, dat wil zeggen door dezelfde reeks bewerkingen toe te passen op de andere drie basistoestanden. Met andere woorden, als we

$$U_{\text{Bell}} = \text{CNOT}_{1\to 2}\,(H\otimes I) \tag{3.56}$$

definiëren geldt dat

$$|\Phi^+\rangle = U_{\text{Bell}}\,|00\rangle\,, \qquad |\Phi^-\rangle = U_{\text{Bell}}\,|10\rangle\,,$$
$$|\Psi^+\rangle = U_{\text{Bell}}\,|01\rangle\,, \qquad |\Psi^-\rangle = U_{\text{Bell}}\,|11\rangle\,.$$

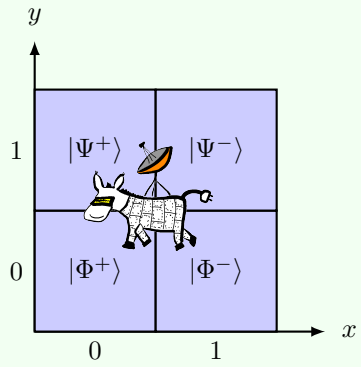
Oefenopgave 3.12: Bell-toestanden maken

Teken hoe je in QUIRKY de andere drie Bell-toestanden kan maken: $|\Phi^-\rangle$, $|\Psi^+\rangle$ en $|\Psi^-\rangle$.

Oefenopgave 3.13: Bell-toestanden onderscheiden

De robot-ezel van Alice is verdwaald geraakt tijdens een verkenningsmissie! De robot wil Alice laten weten waar hij is, zodat ze hem kan redden. De ezel zit in een van de vier wijken rondom de school. Om te laten weten in welke wijk de ezel zit, stuurt hij een twee-qubit

quantumbericht $|x, y\rangle$, waarbij $x \in \{0, 1\}$ het x-coördinaat van zijn locatie aangeeft en $y \in \{0, 1\}$ het y-coördinaat:



Helaas verstoort Alice's gemene klasgenoot Eve het signaal en in plaats van het signaal ontvangt Alice een van de vier Bell-toestanden zoals hierboven afgebeeld. Help Alice om het signaal correct te ontcijferen en haar robot-ezel te vinden! Dat wil zeggen, vind een reeks bewerkingen die elk van de vier Bell-toestanden weer omzet naar de bijbehorende basistoestand $|x, y\rangle$.

3.2.6 Verstrengeling en correlatie

Gezien de overeenkomst tussen verstrengelde toestanden en gecorreleerde kansverdelingen, kun je je afvragen hoe deze twee begrippen verwant zijn. Laten we, om ze te vergelijken, meer in het algemeen de relatie tussen quantumtoestanden en kansverdelingen bespreken.

Stel om te beginnen dat we de één-qubitstoestand $\psi = \psi_0 |0\rangle + \psi_1 |1\rangle$ hebben en we deze meten. Dan weten we uit §2.2 dat we als uitkomst een bit krijgen die nul of één is, met kansen ψ_0^2 en ψ_1^2. We kunnen dit zien als een kansverdeling

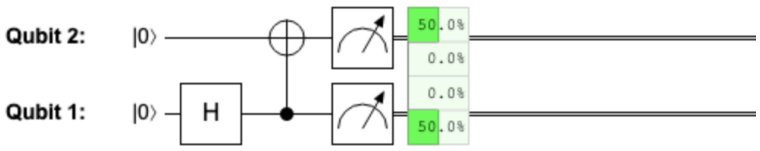
$$\psi_0^2[0] + \psi_1^2[1].$$

Dit modelleert de situatie waarin we de qubit hebben gemeten maar niet echt naar de uitkomst hebben gekeken (als we dat wel zouden doen, zouden we geen probabilistische bit hebben maar een deterministische die ofwel in toestand nul of in toestand één zit).

Dezelfde logica werkt net zo goed voor twee qubits. Als we een twee-qubitstoestand $\psi = \psi_{00} |00\rangle + \psi_{01} |01\rangle + \psi_{10} |10\rangle + \psi_{11} |11\rangle$ meten, kunnen we de meetuitkomsten beschrijven met de kansverdeling

$$p = \psi_{00}^2[00] + \psi_{01}^2[01] + \psi_{10}^2[10] + \psi_{11}^2[11]. \tag{3.57}$$

Als we bijvoorbeeld de maximaal verstrengelde toestand $|\Phi^+\rangle$ maken en vervolgens meten, krijgen we de perfect gecorreleerde probabilistische bits van Vgl. (3.27). We kunnen dit controleren met QUIRKY:



Het is duidelijk dat hetzelfde geldt als we in plaats daarvan de Bell-toestand $|\Phi^-\rangle$ meten. (Hoe zit het met de andere twee Bell-toestanden $|\Psi^+\rangle$ of $|\Psi^-\rangle$? Het meten van een van deze twee toestanden geeft perfect *anti-gecorreleerde* bits, beschreven door de kansverdeling $\frac{1}{2}[01] + \frac{1}{2}[10]$).

Het voorgaande voorbeeld was geen toeval. In feite kan de kansverdeling p van Vgl. (3.57) die verkregen wordt door het meten van een twee-qubit quantumtoestand alleen gecorreleerd zijn als de corresponderende quantumtoestand verstrengeld is. Om dit in te zien, nemen we aan dat $|\psi\rangle$ een producttoestand is, zodat $\Delta(|\psi\rangle) = 0$. Dan is p een productverdeling omdat

$$\begin{aligned}\Delta(p) &= p_{00}p_{11} - p_{01}p_{10} = \psi_{00}^2\psi_{11}^2 - \psi_{01}^2\psi_{10}^2 \\ &= (\psi_{00}\psi_{11} - \psi_{01}\psi_{10})\,(\psi_{00}\psi_{11} + \psi_{01}\psi_{10}) \\ &= \Delta(|\psi\rangle)\,(\psi_{00}\psi_{11} + \psi_{01}\psi_{10}) = 0.\end{aligned} \tag{3.58}$$

Dit bewijst de bewering dat gecorreleerde meetuitkomsten betekenen dat er verstrengeling aanwezig is in de gemeten toestand.

Merk op dat quantumtoestanden over het algemeen minstens zo nuttig zijn als probabilistische bits omdat elke kansverdeling kan worden verkregen door een juist gekozen quantumtoestand te meten. Dat wil zeggen, voor elke willekeurige kansverdeling p kunnen we altijd een quantumtoestand $|\psi\rangle$ vinden waarvan de meetuitkomsten verdeeld zijn volgens p. Om een tweebitsverdeling te beschrijven kunnen we bijvoorbeeld gewoon

$$|\psi\rangle = \sqrt{p_{00}}\,|00\rangle + \sqrt{p_{01}}\,|01\rangle + \sqrt{p_{10}}\,|10\rangle + \sqrt{p_{11}}\,|11\rangle\,.$$

kiezen.

Dit betekent specifiek dat verstrengeling over het algemeen minstens zo nuttig is als probabilistische correlaties, omdat elke gecorreleerde verdeling van twee probabilistische bits kan worden geproduceerd door het meten van een verstrengelde twee-qubitstoestand.

3.2.7 De kracht van verstrengeling

Verstrengelde quantumbits zijn zelfs veel effectiever dan probabilistische bits! In het volgende stuk zullen we hier nog maar een voorproefje van geven. Als je Quest 4 en 5 volgt, dan kom je nog veel meer voorbeelden tegen!

Laten we de kracht van verstrengeling laten zien aan de hand van een verhaaltje dat aansluit op Oefenopgave 3.13. Daar hielp je Alice met het ontcijferen van de locatie van haar robot-ezel, die op een van de vier mogelijke locaties was, aangeduid met twee bits $a, b \in \{0, 1\}$. Alice heeft geen tijd om de ezel op te halen, dus wil ze de locatie doorsturen naar Bob. Helaas is Alice bijna door haar quantum-telefoonabonnement heen, dus ze kan maar één qubit naar Bob sturen. Maar het versturen van een enkele quantumbit is niet genoeg om twee bits correct te verzenden. We weten dit omdat Bob alleen informatie kan krijgen door de qubit te meten – maar uit de meting kan hij maar één bit achterhalen en daarna is de oorspronkelijke toestand van de qubit verdwenen.

Gelukkig hadden Alice en Bob ter voorbereiding de maximaal verstrengelde toestand $|\Phi^+\rangle$ gedeeld. Hiermee bedoelen we dat Alice in het bezit is van het eerste qubit en Bob in het bezit is van het tweede qubit. Kunnen we Alice helpen de locatie van de ezel te verzenden (d.w.z. de twee bits a en b) door een *enkele* qubit te versturen? Dit blijkt inderdaad mogelijk te zijn door gebruik te maken van de volgende techniek:

Oefenopgave 3.14: Bell-toestanden in elkaar omzetten

Laat zien dat Alice, door alleen lokale bewerkingen op haar qubit toe te passen, de maximaal verstrengelde toestand $|\Phi^+\rangle$ kan omzetten in elk van de andere drie Bell-toestanden $|\Phi^-\rangle$, $|\Psi^+\rangle$ en $|\Psi^-\rangle$.

Het is nu duidelijk hoe Alice en Bob het probleem kunnen oplossen. Ga ervan uit dat Alice en Bob van tevoren een afspraak hebben gemaakt over welke van de 4 mogelijke waarden van de twee bits $[ab]$ bij welke van de 4 Bell-toestanden hoort (bijvoorbeeld $[00]$ komt overeen met $|\Phi^+\rangle$, $[01]$ komt overeen met $|\Psi^+\rangle$, enzovoort). Met behulp van Oefenopgave 3.14 past

Alice eerst een bewerking toe op haar deel van de maximaal verstrengelde toestand om deze te veranderen in de Bell-toestand die overeenkomt met de locatie van de ezel. Daarna stuurt ze haar quantumbit naar Bob. Bob heeft nu beide quantumbits in zijn bezit en weet dat ze in een van de vier Bell-toestanden zitten. Hij kan dus gewoon dezelfde bewerkingen uitvoeren als in Oefenopgave 3.13 en de twee qubits meten om de locatie van de ezelrobot te bepalen.

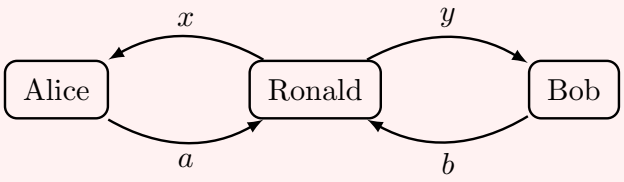
De procedure die we net beschreven hebben staat bekend als het superdense coding protocol, omdat we twee deterministische bits versturen door maar één quantumbit te versturen (ten koste van het gebruik van één maximaal verstrengelde toestand die Alice en Bob met elkaar delen). Hadden Alice en Bob net zo goed probabilistische bits kunnen gebruiken in plaats van een verstrengelde toestand om dit probleem op te lossen (bijvoorbeeld twee perfect gecorreleerde bits)? Merkwaardig genoeg is dit niet het geval. Superdense coding laat dus zien dat quantumverstrengeling een voordeel biedt voor communicatietaken.

In de volgende huiswerkopdracht kom je nog een beroemde situatie tegen waarin quantumverstrengeling een duidelijk voordeel oplevert. De opdracht ziet er misschien wat intimiderend uit – maar dat komt vooral omdat we het niet konden laten om er een klein verhaaltje omheen te vertellen. Zoals altijd, aarzel niet om vragen te stellen op Discord!

Huiswerkopdracht 3.7: Een verstrengeld spel (uitdagend)

Alice en Bob vervelen zich tijdens de les, dus vragen ze hun docent quantummechanica Ronald om hen een uitdagende puzzel te geven. Na een korte pauze legt Ronald hen een interessant spel uit. Het doel van het spel is dat Alice en Bob zo goed mogelijk samenwerken (ze spelen niet tegen elkaar). Maar ze mogen tijdens het spel niet met elkaar communiceren! De spelregels zijn als volgt:

- Om te beginnen gooit Ronald in het geheim twee zuivere muntjes op. Hij vertelt Alice het resultaat van de eerste muntworp (bit x) en Bob het resultaat van de tweede muntworp (bit y). We noemen dit de invoerbits.
- Na het ontvangen van de bits moeten Alice en Bob ieder één bit terugsturen als antwoord (bits a en b).
- Alice en Bob winnen het spel onder de volgende voorwaarde: als $x = y = 1$, dan winnen ze het spel als $a \neq b$; anders winnen ze het spel als $a = b$.



x	y	Voorwaarde om te winnen
0	0	$a = b$
0	1	$a = b$
1	0	$a = b$
1	1	$a \neq b$

Voordat het spel begint, komen Alice en Bob kort bij elkaar om hun strategie te bespreken. Eerst overwegen ze om gewoon twee functies toe te passen $f, g : \{0,1\} \to \{0,1\}$ op hun invoerbits x en y en hun antwoorden als volgt berekenen: $a = f(x)$ en $b = g(y)$.

1. Laat zien dat ze in dit geval een kans van 75% kunnen halen om het spel te winnen, maar niet hoger dan dat.

Vervolgens overwegen ze om voor hun strategie gebruik te maken van gedeelde randomness. Bob stelt voor om ingewikkeldere functies f en g te gebruiken die een extra binair argument hebben en hun antwoorden als volgt te berekenen: $a = f(x, r)$ en $b = g(y, s)$, waarbij r en s twee random bits zijn die gezamenlijk beschreven worden door een twee-bits kansverdeling.

2. Laat zien dat ze nog steeds niet een hogere winkans dan 75% kunnen hebben, ongeacht

wat de functies f en g zijn en wat de gezamenlijke verdeling van de bits r en s is.

Onze spelers beginnen zich te realiseren dat Ronald waarschijnlijk een quantummechanische strategie in gedachten had. Alice heeft een geweldig idee en stelt voor dat ze een maximaal verstrengelde toestand delen voordat het spel begint. Ze stelt voor dat ze na het ontvangen van hun bits haar qubit met een bepaalde hoek θ_x roteert (die afhangt van haar invoerbit x) en deze bit meet om haar antwoord a te krijgen, terwijl Bob in plaats daarvan zijn qubit met een andere hoek ω_y roteert (die afhangt van zijn invoerbit y) en dan meet om zijn antwoord b te krijgen.

3. Geef de toestand nadat Alice en Bob hun rotaties hebben toegepast in de vorm (3.30). Bevestig dat de waarschijnlijkheid om het spel te winnen gelijk is aan

$$\frac{1}{4}\left(\cos^2(\theta_0-\omega_0)+\cos^2(\theta_0-\omega_1)+\cos^2(\theta_1-\omega_0)+\sin^2(\theta_1-\omega_1)\right).$$

   Hint: Gebruik de goniometrische formules van Vgl. (2.16) en (2.22).

Alice en Bob hebben al snel door dat $\theta_0 = 0$, $\theta_1 = \pi/4$ en $\omega_0 = \pi/8$ goede keuzes zijn, maar ze hebben moeite met de laatste hoek en de tijd begint op te raken.

4. Zoek een hoek ω_1 zo dat de kans dat ze het spel winnen groter is dan 75%.

In de eerste twee delen van het huiswerk heb je laten zien dat geen enkele strategie die gebruik maakt van klassieke bits het spel kan winnen met een hogere kans dan 75%. Dit is een voorbeeld van een *ongelijkheid van Bell*. De versie hierboven komt van John Clauser, Michael Horne, Abner Shimony en Richard Holt, dus het spel dat Alice en Bob met Ronald spelen wordt vaak het *CHSH spel* genoemd. De quantummechanische strategie die je in het huiswerk hebt ontdekt, *schendt* de ongelijkheid van Bell. Het is een empirisch feit dat de ongelijkheden van Bell inderdaad geschonden kunnen worden door de quantummechanica. Dit werd voor het eerst aangetoond door Alain Aspect in de jaren 80 en is onlangs bevestigd onder heel strikte voorwaarden in een prachtig experiment door Ronald Hanson en zijn team aan de TU Delft.

Interessant genoeg kunnen Alice en Bob het CHSH-spel niet alleen beter spelen met behulp van een gedeelde maximaal verstrengelde toestand, maar ze kunnen Ronald er in feite van overtuigen dat ze wel quantumtrucs gebruikt moeten hebben om het spel zo goed te spelen. Ze kunnen het spel namelijk niet winnen met een waarschijnlijkheid van meer dan 75% als ze alleen probabilistische strategieën gebruiken. Als ze er op de een of andere manier in slagen om het spel vaker te winnen, dan is de enige mogelijke verklaring dat ze iets krachtigers gebruikt moeten hebben. Met nog wat trucjes kunnen ze Ronald er zelfs van overtuigen dat ze de maximaal verstrengelde toestand gebruikt moeten hebben en rotaties met de gebruikte hoeken moeten hebben toegepast. In feite betekent dit dat ze Ronald op een onweerlegbare manier kunnen laten zien dat ze kleine quantumcomputers hebben die afzonderlijke qubits kunnen manipuleren en verstrengelde toestanden kunnen delen. Dit is een hele belangrijke constatering, want hiermee kun je het CHSH-spel gebruiken om te controleren of iemand echt een quantumcomputer heeft gebouwd! Als twee van je klasgenoten zouden beweren dat ze elk een quantumcomputer in hun garage hebben gebouwd, dan kun je ze gewoon vragen om samen dit spel tegen jou te spelen. Als ze erin slagen te winnen met een waarschijnlijkheid die significant groter is dan 75%, dan weet je dat ze niet tegen je liegen en dat ze echte quantumcomputers hebben. Maar als ze daarentegen niet kunnen winnen met meer dan 75%, dan zou je hun beweringen gemakkelijk kunnen weerleggen. Is dit niet ongelooflijk?

Dit soort controleprocedures is iets waar onderzoekers over de hele wereld momenteel actief aan werken. Dit is een heel belangrijk probleem omdat verschillende grote bedrijven, zoals IBM, Google en Microsoft, proberen een quantumcomputer te bouwen. Als zij beweren dat ze

er een hebben gebouwd, zou je ze waarschijnlijk meer geloven dan je klasgenoten. Toch is het geweldig als je dit ook echt kunt bevestigen!

3.3 Oplossingen van de oefenopgaven

Oplossing van Oefenopgave 3.2

1. We duiden de kansverdelingen van Alice' twee worpen aan met p. We gebruiken Fig. 3.3 om de kansen te berekenen. We weten dat de eerste munt van Alice verdeeld is volgens u, wat betekent dat als we het eerste bit meten, we beide uitkomsten krijgen met kans $1/2$:

$$p_{00} + p_{01} = \frac{1}{2}, \quad p_{10} + p_{11} = \frac{1}{2}.$$

Vervolgens weten we dat als het opgooien van het eerste muntje uitkomst 0 gaf, de toestand van het tweede bit beschreven moet worden door het muntje q. Er moet dus gelden dat

$$\frac{p_{00}[0] + p_{01}[1]}{p_{00} + p_{01}} = \frac{3}{4}[0] + \frac{1}{4}[1],$$

waaruit we afleiden dat $p_{00} = \frac{3}{8}$ en $p_{01} = \frac{1}{8}$. Als de eerste worp uitkomst 1 gaf, dan wordt de toestand van het tweede bit beschreven door munt r, dus

$$\frac{p_{10}[0] + p_{11}[1]}{p_{00} + p_{01}} = \frac{1}{3}[0] + \frac{2}{3}[1],$$

dus $p_{10} = \frac{1}{6}$ en $p_{11} = \frac{2}{6}$. Samen geeft dit

$$p = \frac{3}{8}[00] + \frac{1}{8}[01] + \frac{1}{6}[10] + \frac{2}{6}[11],$$

2. De waarde van Bob's bit is hetzelfde als de uitkomst van Alice' tweede muntworp, dus we volgen gewoon Fig. 3.3 om de waarschijnlijkheid van de uitkomsten te berekenen bij het meten van het tweede bit. De kans dat de uitkomst van Bob 0 is, is dus

$$p_{00} + p_{10} = \frac{3}{8} + \frac{1}{6} = \frac{13}{24}$$

en de kans dat zijn uitkomst 1 is, is $1 - \frac{13}{24} = \frac{11}{24}$. We kunnen dit beschrijven als de kansverdeling

$$\frac{13}{24}[0] + \frac{11}{24}[1].$$

3. Ook hier gebruiken we weer Fig. 3.3. Als het tweede bit gemeten wordt en het resultaat is uitkomst 0 dan wordt de toestand van het eerste bit gegeven door

$$\frac{p_{00}[0] + p_{10}[1]}{p_{00} + p_{10}} = \frac{9}{13}[0] + \frac{4}{13}[1].$$

Het is dus waarschijnlijker dat Alice' eerste muntje op 0 was geland.

Net zo geldt dat als het resultaat van het meten van het tweede bit 1 is, de toestand van het eerste bit wordt beschreven door

$$\frac{p_{01}[0] + p_{11}[1]}{p_{01} + p_{11}} = \frac{3}{11}[0] + \frac{8}{11}[1].$$

In dit geval is het waarschijnlijker dat Alice' eerste munt op 1 was geland.

Oplossing van Oefenopgave 3.1

$$\mathrm{NOT}_1 \begin{pmatrix} p_{00} \\ p_{01} \\ p_{10} \\ p_{11} \end{pmatrix} = \begin{pmatrix} p_{10} \\ p_{11} \\ p_{00} \\ p_{01} \end{pmatrix}.$$

Oplossing van Oefenopgave 3.3

$$\mathrm{SWAP} \begin{pmatrix} p_{00} \\ p_{01} \\ p_{10} \\ p_{11} \end{pmatrix} = \begin{pmatrix} p_{00} \\ p_{10} \\ p_{01} \\ p_{11} \end{pmatrix}.$$

Oplossing van Oefenopgave 3.4

1. $\mathrm{CNOT}_{2\to1}[a,b] = [a \oplus b, b] = [b \oplus a, b]$.
2. Dit kan gedaan worden door eerst een SWAP toe te passen, dan een $\mathrm{CNOT}_{1\to2}$ en tenslotte nog een SWAP. Met behulp van Vgl. (3.17) en (3.19) kunnen we inderdaad zien dat

$$\begin{aligned} &\mathrm{SWAP}(\mathrm{CNOT}_{1\to2}(\mathrm{SWAP}[a,b])) = \mathrm{SWAP}(\mathrm{CNOT}_{1\to2}[b,a]) \\ &= \mathrm{SWAP}[b, b \oplus a] = [b \oplus a, b]. \end{aligned}$$

Oplossing van Oefenopgave 3.5

Met behulp van Fig. 3.3 kan de toestand van het tweede bit na de meting als volgt worden berekend:

$$\frac{p_{a0}[0] + p_{a1}[1]}{p_{a0} + p_{a1}} = \frac{q_a r_0[0] + q_a r_1[1]}{q_a r_0 + q_a r_1} = \frac{r_0[0] + r_1[1]}{r_0 + r_1} = r_0[0] + r_1[1] = r,$$

waarbij we q_a hebben weggestreept en gebruikt hebben dat $r_0 + r_1 = 1$.

Oplossing van Oefenopgave 3.6

De toestand voor de controlled-NOT-bewerking is

$$\left(\frac{1}{2}[0] + \frac{1}{2}[1]\right) \otimes [0] = \frac{1}{2}[00] + \frac{1}{2}[10].$$

Na het toepassen van de controlled-NOT-bewerking, krijgen we

$$\mathrm{CNOT}_{1\to2}\left(\frac{1}{2}[00] + \frac{1}{2}[10]\right) = \frac{1}{2}[00] + \frac{1}{2}[11].$$

Oplossing van Oefenopgave 3.7

1.

$$\begin{aligned}|+\rangle \otimes |-\rangle &= \left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) \otimes \left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right) \\ &= \frac{1}{2}|00\rangle - \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle - \frac{1}{2}|11\rangle .\end{aligned}$$

2.

$$\frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = |+\rangle \otimes |+\rangle .$$

Dit is dus een producttoestand.

Oplossing van Oefenopgave 3.8

We beginnen met de toestand $|00\rangle$. Na de NOT op het tweede qubit krijgen we $|01\rangle$. De Hadamard op het eerste qubit zet dit om in $|+\rangle \otimes |1\rangle = \frac{1}{\sqrt{2}}|01\rangle + \frac{1}{\sqrt{2}}|11\rangle$ en met de laatste Z-bewerking krijgen we vlak voor de meting de volgende twee-qubitstoestand:

$$\frac{1}{\sqrt{2}}|01\rangle - \frac{1}{\sqrt{2}}|11\rangle .$$

Dus we krijgen ofwel 01 of 11 met op een beide een kans van 50%.

Oplossing van Oefenopgave 3.9

We laten alleen zien hoe we Vgl. (3.39) kunnen nagaan (de andere vergelijking kan op precies dezelfde manier worden afgeleid). We stellen hiervoor dat $\alpha = \alpha_0|0\rangle + \alpha_1|1\rangle$ en $\beta = \beta_0|0\rangle + \beta_1|1\rangle$. Dan geldt dat

$$\begin{aligned}U_1(|\alpha\rangle \otimes |\beta\rangle) &= U_1(\alpha_0\beta_0|00\rangle + \alpha_0\beta_1|01\rangle + \alpha_1\beta_0|10\rangle + \alpha_1\beta_1|11\rangle) \\ &= \alpha_0\beta_0 U_1|00\rangle + \alpha_0\beta_1 U_1|01\rangle + \alpha_1\beta_0 U_1|10\rangle + \alpha_1\beta_1 U_1|11\rangle \\ &= \alpha_0\beta_0 U|0\rangle \otimes |0\rangle + \alpha_0\beta_1 U|0\rangle \otimes |1\rangle + \alpha_1\beta_0 U|1\rangle \otimes |0\rangle + \alpha_1\beta_1 U|1\rangle \otimes |1\rangle \\ &= \alpha_0 U|0\rangle \otimes (\beta_0|0\rangle + \beta_1|1\rangle) + \alpha_1 U|1\rangle \otimes (\beta_0|0\rangle + \beta_1|1\rangle) \\ &= \alpha_0 U|0\rangle \otimes |\beta\rangle + \alpha_1 U|1\rangle \otimes |\beta\rangle \\ &= (\alpha_0 U|0\rangle + \alpha_1 U|1\rangle) \otimes |\beta\rangle \\ &= U|\alpha\rangle \otimes |\beta\rangle\end{aligned}$$

volgens Vgl. (3.33), de definitie van U_1 en lineariteit.

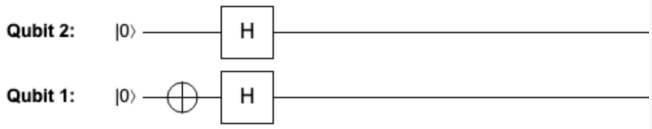
Oplossing van Oefenopgave 3.10

1. De toestand kan beschreven worden als

$$\frac{1}{2}\left(|00\rangle + |01\rangle - |10\rangle - |11\rangle\right) = \frac{1}{\sqrt{2}}\left(|0\rangle - |1\rangle\right) \otimes \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right).$$

2. Dit is gelijk aan

$$H|1\rangle \otimes H|0\rangle = H\,\mathrm{NOT}\,|0\rangle \otimes H|0\rangle = (H\,\mathrm{NOT} \otimes H)\,|00\rangle.$$

3. 

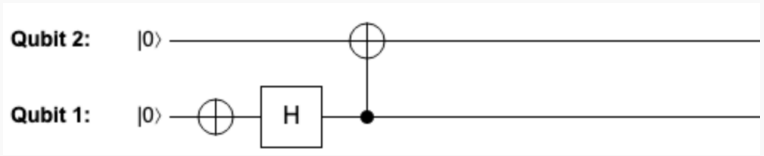
Oplossing van Oefenopgave 3.11

Om te laten zien dat $HZ \neq ZH$ hoeven we alleen na te gaan dat de twee bewerkingen andere resultaten geven als we ze op de $|0\rangle$ toestand uitvoeren. Zo blijkt:

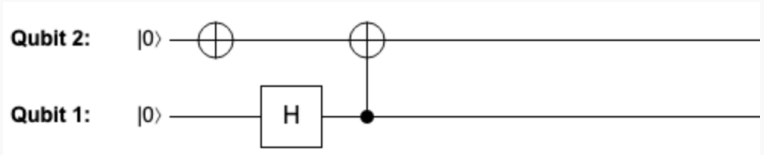
$$HZ|0\rangle = H|0\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle,$$
$$ZH|0\rangle = Z\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle.$$

Oplossing van Oefenopgave 3.12

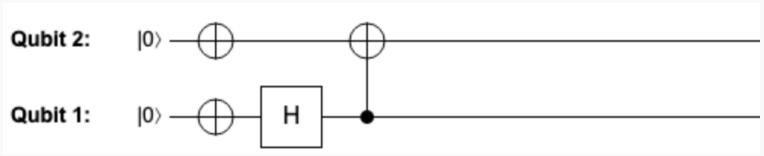
- $|\Phi^-\rangle$:



- $|\Psi^+\rangle$:



- $|\Psi^-\rangle$:



Oplossing van Oefenopgave 3.13

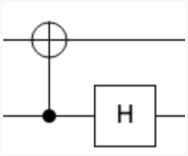
Merk op dat dit hetzelfde is als het inverteren van de bewerking U_{Bell} uit Vgl. (3.56). Bedenk je dat $U_{\text{Bell}} = \text{CNOT}_{1\to 2}\,(H \otimes I)$. Dit is een samengestelde bewerking, dus uit Oefenopgave 2.5 volgt dat

$$U_{\text{Bell}}^{-1} = (H \otimes I)^{-1}\,\text{CNOT}_{1\to 2}^{-1} = (H^{-1} \otimes I)\,\text{CNOT}_{1\to 2}^{-1}.$$

Denk eraan dat volgens Vgl. (3.49) $\text{CNOT}_{1\to 2}$ de inverse is van zichzelf, dus $\text{CNOT}_{1\to 2}^{-1} = \text{CNOT}_{1\to 2}$. Het is ook zo dat $H^{-1} = H$ (dit geldt eigenlijk voor elke spiegeling). Dit betekent dat we U_{Bell} ongedaan kunnen maken door dezelfde twee bewerkingen toe te passen, maar dan in omgekeerde volgorde:

$$U_{\text{Bell}}^{-1} = (H \otimes I)\,\text{CNOT}_{1\to 2}.$$

Dat wil zeggen, we passen eerst $\text{CNOT}_{1\to 2}$ en dan H toe op het eerste qubit, zoals in:



Oplossing van Oefenopgave 3.14

Merk op uit Vgl. (3.52) tot (3.55) dat de Bell-toestanden alleen verschillen door bitflips en mintekens. We weten dat we een bit kunnen wisselen met een lokale *NOT* bewerking en dat we sommige mintekens kunnen toevoegen door lokale Z-bewerkingen toe te passen, waarbij Z gedefinieerd is in Vgl. (2.12). Om $|\Phi^-\rangle$ te maken, kan Alice een Z-bewerking op haar qubit toepassen:

$$(Z \otimes I)\,|\Phi^+\rangle = (Z \otimes I)\left(\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle\right) = \frac{1}{\sqrt{2}}|00\rangle - \frac{1}{\sqrt{2}}|11\rangle = |\Phi^-\rangle\,.$$

Om $|\Psi^+\rangle$ te maken past Alice in plaats daarvan een NOT-bewerking toe op haar qubit:

$$(\text{NOT} \otimes I)\,|\Phi^+\rangle = (\text{NOT} \otimes I)\left(\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle\right) = \frac{1}{\sqrt{2}}|10\rangle + \frac{1}{\sqrt{2}}|01\rangle = |\Psi^+\rangle\,.$$

Om $|\Psi^-\rangle$ te maken past Alice eerst een NOT-bewerking en daarna een Z-bewerking toe:

$$(Z\,\text{NOT} \otimes I)\,|\Phi^+\rangle = (Z \otimes I)\left(\frac{1}{\sqrt{2}}|10\rangle + \frac{1}{\sqrt{2}}|01\rangle\right) = -\frac{1}{\sqrt{2}}|10\rangle + \frac{1}{\sqrt{2}}|01\rangle = |\Psi^-\rangle\,.$$

Quest 4: Een quantum-orkest

Afgelopen week hebben we gekeken naar de toestanden van twee quantumbits en welke bewerkingen je hiermee kan doen. We hebben ook geleerd dat er *vestrengelde* quantumtoestanden bestaan en dat deze gebruikt kunnen worden om efficiënter te communiceren (door bijvoorbeeld twee bits over te brengen terwijl je maar een enkele qubit verstuurt) en om de deterministische en probabilistische strategieën van een bepaald spel te overtreffen. Deze week gaan we gebruik maken van een arbitrair aantal qubits en beginnen met het samenstellen van quantumcircuits, die uit een groot aantal bewerkingen bestaan, om interessante problemen op te lossen.

4.1 Quantumcircuits

Als je veel qubits gebruikt en daar veel bewerkingen op toepast, raak je makkelijk het overzicht kwijt. Net zoals een componist bladmuziek schrijft om elke muzikant van een orkest instructies te geven over welke noot ze op welk moment moeten spelen, zo geven quantumcircuits een mooie beschrijving van welke bewerking op welk moment op welke qubit moet worden toegepast. Elke bewerking kan je zien als een andere noot en elke qubit kan je zien als een andere muzikant in een orkest!

Maar in tegenstelling tot een orkest kunnen sommige quantumbewerkingen op twee (of soms zelfs drie!) qubits tegelijk werken. Dit is alsof je de gitarist vraagt om op de viool te tokkelen terwijl de violist de gitaar strijkt! Zulke interacties tussen muzikanten (of qubits) kunnen heel leuk zijn en een veel voller en complexer geluid maken. Maar in de praktijk kan het lastig zijn om dit uit te voeren (je kunt je voorstellen dat de violist over het podium heen moet rennen om bij de gitarist te komen!). Bovendien kan tijdens deze hectische samenwerking tussen de twee muzikanten het haar van de violist verstrikt raken in de grote oorbellen van de gitarist, waardoor ze de rest van het stuk samen moeten spelen.

Als je meer wisselwerkingen uitvoert op de qubits, dan raken ze vaak meer verstrengeld. Aan het eind van het stuk zijn alle qubits meestal zo met elkaar verstrengeld dat ze een ontzettend ingewikkelde toestand vormen. De enige manier om ze uit elkaar te halen en terug te brengen naar een redelijke toestand is door ze te meten! Dit is precies hoe in het algemeen een quantumberekening werkt. Formeler gezegd bestaat een *quantumcircuit* uit drie ingrediënten:

1. een begintoestand, meestal elke qubit in de toestand $|0\rangle$,

2. een reeks quantumbewerkingen, waarbij elke bewerking op maar een paar qubits tegelijk werkt (meestal één of twee qubits),

3. metingen om de informatie uit te lezen (meestal meten we alle qubits).

We kunnen quantumcircuits grafisch weergeven met hetzelfde soort plaatjes dat we al kennen van QUIRKY. Als we het over quantumcircuits hebben, worden bewerkingen en metingen vaak quantum-gates genoemd (bijvoorbeeld 'Hadamard-gate' in plaats van 'Hadamard-bewerking'). Laten we nu eens wat dieper ingaan op de drie ingrediënten en kijken hoe ze zich in een groot orkest van qubits gedragen.

4.1.1 Veel quantumbits

De regels die we tot nu toe geleerd hebben voor één en twee qubits kunnen op een natuurlijke wijze veralgemeend worden naar quantumsystemen met veel qubits. Een arbitraire toestand

van drie qubits kan bijvoorbeeld als volgt worden geschreven:

$$\begin{aligned} |\psi\rangle = \psi_{000}\,|000\rangle + \psi_{001}\,|001\rangle + \psi_{010}\,|010\rangle + \psi_{011}\,|011\rangle \\ + \psi_{100}\,|100\rangle + \psi_{101}\,|101\rangle + \psi_{110}\,|110\rangle + \psi_{111}\,|111\rangle\,, \end{aligned} \tag{4.1}$$

waar $\psi_{ijk} \in [-1, 1]$ en de kwadraten van deze amplitudes samen weer op één uitkomen, dus,

$$\psi_{000}^2 + \psi_{001}^2 + \psi_{010}^2 + \psi_{011}^2 + \psi_{100}^2 + \psi_{101}^2 + \psi_{110}^2 + \psi_{111}^2 = 1.$$

Merk op dat er in totaal $8 = 2^3$ amplitudes zijn, één voor elke bitstring van drie bits. We kunnen $|\psi\rangle$ ook zien als een vector met acht getallen:

$$|\psi\rangle = \begin{pmatrix} \psi_{000} \\ \psi_{001} \\ \psi_{010} \\ \psi_{011} \\ \psi_{100} \\ \psi_{101} \\ \psi_{110} \\ \psi_{111} \end{pmatrix}.$$

In het algemeen kan een toestand van n qubits beschreven worden met 2^n amplitudes $\psi_{a_1,\ldots,a_n}$, één voor elke bitstring van n bits:

$$|\psi\rangle = \psi_{00\ldots00}\,|00\ldots00\rangle + \psi_{00\ldots01}\,|00\ldots01\rangle + \ldots + \psi_{11\ldots11}\,|11\ldots11\rangle \tag{4.2}$$

Ook hier moet elke amplitude $\psi_{a_1,\ldots,a_n}$ in $[-1, 1]$ liggen en de som van hun kwadraten op één uitkomen:

$$\psi_{00\ldots00}^2 + \psi_{00\ldots01}^2 + \ldots + \psi_{11\ldots11}^2 = 1 \tag{4.3}$$

Als sommige amplitudes $\psi_{a_1,\ldots,a_n}$ nul zijn, kunnen we ze gewoon weglaten. De vijf-qubit-toestand

$$\frac{1}{\sqrt{2}}\left(|00000\rangle + |11111\rangle\right)$$

heeft bijvoorbeeld 32 amplitudes, waarvan er 30 nul zijn.

Omdat er 2^n amplitudes zijn, kunnen we $|\psi\rangle$ ook zien als een vector in een 2^n-dimensionale ruimte. Wat wordt meetkundig bedoeld met Vgl. (4.3)? Voor een enkele qubit zagen we in §2.1.2 dat de toestanden overeenkomen met punten op de eenheidscirkel, oftewel twee-dimensionale vectoren met een lengte van één. Volgens de stelling van Pythagoras geldt in elke dimensie dat de som van de kwadraten van alle waarden in een vector gelijk is aan het kwadraat van zijn lengte. Dus, Vgl. (4.3) betekent meetkundig gezien dat $|\psi\rangle$ overeenkomt met een vector met *lengte één* ofwel een *eenheidsvector* in een 2^n-dimensionale ruimte.

Merk op dat het aantal amplitudes heel snel groeit met het aantal qubits. Dit verklaart waarom het al snel onmogelijk wordt om quantumtoestanden direct op te slaan in een klassieke computer. Om bijvoorbeeld een quantumtoestand van $n = 300$ qubits op te slaan, zou je meer amplitudes nodig hebben dan er atomen zijn in het waarneembare universum! Daarom kun je in QUIRKY ook niet meer dan 10 qubits hebben, we willen niet dat je webbrowser zonder geheugen komt te zitten!

Net als in Vgl. (3.33), kunnen we het tensorproduct '$\otimes$' gebruiken om quantumtoestanden van een willekeurig aantal qubits te combineren. Als we twee basistoestanden hebben, definiëren we hun tensorproduct door simpelweg de bitstrings aan elkaar vast te plakken. We veralgemenen het twee-qubit geval van Vgl. (3.32),

$$|a_1, \ldots, a_n\rangle \otimes |b_1, \ldots, b_m\rangle = |a_1, \ldots, a_n, b_1, \ldots, b_m\rangle\,. \tag{4.4}$$

Bijvoorbeeld,

$$|101\rangle \otimes |01\rangle = |10101\rangle .$$

In het algemeen, als $|\alpha\rangle$ een arbitraire quantumtoestand van n qubits is en $|\beta\rangle$ een arbitraire quantumtoestand van m qubits, dan is hun tensorproduct of gecombineerde toestand een toestand van $n+m$ qubits. Om deze toestand te berekenen hoeven we alleen maar 'uit te vermenigvuldigen' met behulp van de distributiviteitswet en vervolgens Vgl. (4.4) toe te passen voor elke term. Als voorbeeld, het tensorproduct van twee maximaal verstrengelde toestanden ziet er als volgt uit:

$$\begin{aligned}
&|\Phi^+\rangle \otimes |\Phi^+\rangle \\
&= \left(\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle\right) \otimes \left(\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle\right) \\
&= \frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}}|00\rangle \otimes |00\rangle + \frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}}|00\rangle \otimes |11\rangle + \frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}}|11\rangle \otimes |00\rangle + \frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}}|11\rangle \otimes |11\rangle \\
&= \frac{1}{2}|0000\rangle + \frac{1}{2}|0011\rangle + \frac{1}{2}|1100\rangle + \frac{1}{2}|1111\rangle .
\end{aligned}$$

Oefenopgave 4.1: Tensorproduct van Bell-toestanden

Bereken het tensorproduct $|\Phi^-\rangle \otimes |\Psi^-\rangle$ van de twee Bell-toestanden uit Vgl. (3.53) en (3.55).

4.1.2 Bewerkingen

Wat zijn de quantumbewerkingen die we kunnen uitvoeren als we meerdere qubits hebben? Om te beginnen kunnen we de bekende bewerkingen op één of twee qubits, zoals besproken in Paragrafen 2 en 3, toepassen op elke gewenste qubit van een veel-qubit-toestand. Dit werkt zoals in §3.2.2.

Als U bijvoorbeeld een één-qubitbewerking is, dus een rotatie of een spiegeling, dan kunnen we een quantumbewerking definiëren, aangeduid met U_1, die U toepast op de eerste qubit van een n-qubit-toestand. Deze bewerking wordt als volgt gedefinieerd op de basistoestanden:

$$U_1 |a_1, \ldots, a_n\rangle = U|a_1\rangle \otimes |a_2, \ldots, a_n\rangle .$$

Merk op dat het tensorproduct de één-qubit-toestand $U|a_1\rangle$ combineert met de $(n-1)$-qubit-basistoestand $|a_2, \ldots, a_n\rangle$ om een toestand van n qubits te vormen, zoals gewenst. Zoals gewoonlijk breiden we U_1 lineair uit tot algemene quantumtoestanden van n qubits. Op dezelfde manier definiëren we de quantumbewerkingen U_2, U_3, etc. die U toepassen op het tweede, derde, etc. qubit.

Oefenopgave 4.2: Een één-qubitbewerking toepassen

Bereken het resultaat van het toepassen van de Hadamard-bewerking op het tweede qubit van de drie-qubit-toestand $|\Phi^+\rangle \otimes |1\rangle$. In andere woorden, bereken $H_2\left(|\Phi^+\rangle \otimes |1\rangle\right)$. Schrijf je resultaat op in de vorm van Vgl. (4.1).

We kunnen op dezelfde manier uitvogelen hoe een twee-qubitbewerking toegepast kan worden op twee geselecteerde qubits van de n qubits. We zullen vooral kijken naar controlled-NOT-bewerkingen: $\text{CNOT}_{k\to l}$ met $k \neq l$ is de bewerking die de l-de qubit (de doelqubit) omdraait afhankelijk van de waarde van de k-de qubit (de controlequbit). Wiskundig gezien ziet de werking op de basistoestanden er als volgt uit:

$$\text{CNOT}_{k\to l} |a_1, \ldots, a_l, \ldots, a_n\rangle = |a_1, \ldots, a_l \oplus a_k, \ldots, a_n\rangle ,$$

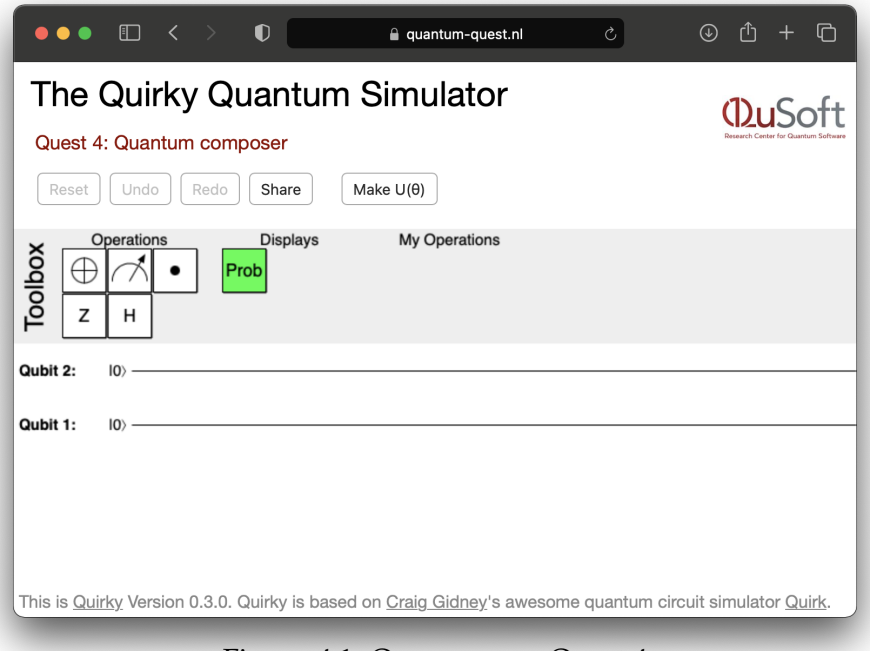
en we breiden dit voorschrift lineair uit naar arbitraire n-qubit-toestanden. De controlled-NOT-bewerking $\text{CNOT}_{1\to3}$ wordt bijvoorbeeld als volgt gedefinieerd voor vier-qubit-basistoestanden:

$$\text{CNOT}_{1\to3}\,|a_1, a_2, a_3, a_4\rangle = |a_1, a_2, a_3 \oplus a_1, a_4\rangle\,.$$

Hoe ziet dit er allemaal uit in QUIRKY? Ga naar

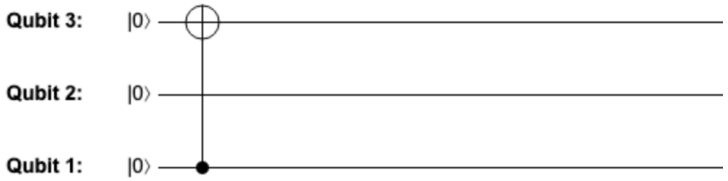
https://www.quantum-quest.org/quirky

en klik op "Quest 4" om het uit te zoeken. Je browser ziet er dan als het goed is hetzelfde uit als Fig. 4.1.



Figuur 4.1: QUIRKY voor Quest 4.

Wacht eens even, het lijkt erop dat QUIRKY er precies hetzelfde uitziet als vorige week!? Maar zodra je een bewerking oppakt uit de toolbox, verschijnt er een nieuwe draad onderaan – waardoor je met een extra qubit kunt werken. (Natuurlijk hebben we het aantal qubits beperkt tot een redelijk aantal, dat je klassieke computer nog comfortabel kan simuleren!) Probeer maar eens een $\text{CNOT}_{1\to3}$-bewerking te maken, zoals in de volgende afbeelding.



Wanneer quantumbewerkingen op verschillende qubits werken, kunnen we ze parallel uitvoeren. Net als in §3.2.3 gebruiken we hiervoor het tensorproduct-symbool. Als U een quantumbewerking op n qubits is en V een quantumbewerking op m qubits, kunnen we een quantumbewerking $U \otimes V$ op $(n+m)$ qubits definiëren die neerkomt op het parallel uitvoeren van beide bewerkingen. Op basistoestanden ziet dit eruit als

$$(U \otimes V)\,|a_1, \ldots, a_n, b_1, \ldots, b_m\rangle = U\,|a_1, \ldots, a_n\rangle \otimes V\,|b_1, \ldots, b_m\rangle\,, \tag{4.5}$$

en we breiden dit uit door lineariteit tot arbitraire toestanden. Uit Vgl. (4.5) volgt dan dat

$$(U \otimes V)(|\alpha\rangle \otimes |\beta\rangle) = U|\alpha\rangle \otimes V|\beta\rangle\,,$$

maar dit geldt alleen als $|\alpha\rangle$ een n-qubit-toestand is en $|\beta\rangle$ een m-qubit-toestand! In de volgende opgave komt dit *niet* zo mooi uit, dus je kunt deze regel *niet* gebruiken!

Oefenopgave 4.3: Niet-uitgelijnde tensorproducten

Beschouw de drie-qubit-toestand $(\text{CNOT}_{2\to1} \otimes I)(|0\rangle \otimes |\Phi^-\rangle)$.

1. Hoe kun je deze toestand maken met QUIRKY?
2. Schrijf de toestand uit in de vorm van Vgl. (4.1).

We kunnen het tensorproduct meerdere keren gebruiken om steeds grotere quantumbewerkingen op te bouwen. Hier zijn drie voorbeelden voor een verschillend aantal qubits:

1. $I \otimes I \otimes U \otimes I$ is dezelfde vier-qubit-bewerking als U_3,
2. $I \otimes \text{CNOT}_{1\to2} \otimes I \otimes I$ is de controlled-NOT-bewerking $\text{CNOT}_{2\to3}$ voor vijf qubits,
3. $Z \otimes I \otimes X$ is de quantumbewerking die Z toepast op de eerste qubit en parallel X op de derde qubit (we kunnen dit ook schrijven als Z_1X_3 of X_3Z_1).

4.1.3 De meest algemene quantumbewerkingen

Wat zijn de meest algemene bewerkingen die we kunnen toepassen op quantumtoestanden van n qubits? Eigenlijk is elke bewerking die de volgende drie eigenschappen heeft:

1. het is lineair,
2. het beeldt quantumtoestanden af op quantumtoestanden.
3. het is inverteerbaar

een geldige quantumbewerking!

Oefenopgave 4.4: Toffoli

Definieer de Toffoli-bewerking op drie qubits door

$$T|a,b,c\rangle = |a,b,c \oplus ab\rangle$$

op de basistoestanden (ab is het product van de twee bits $a, b \in \{0,1\}$, en $\oplus$ is gedefinieerd in Vgl. (3.20)), en breid het lineair uit naar arbitraire drie-qubits-toestanden. Laat zien dat T quantumtoestanden afbeeldt op quantumtoestanden en dat T inverteerbaar is.

Opmerking: T inverteert het derde bit van de basisvector als en alleen als de eerste twee bits beide op één staan – dus het is een soort 'dubbel-controlled-NOT'-bewerking.

Oefenopgave 4.4 laat zien dat de Toffoli-bewerking een geldige quantumbewerking op drie qubits is. Opmerkelijk is dat het eigenlijk mogelijk is om T te schrijven als een reeks van één- en twee-qubitbewerkingen. Dit is zelfs mogelijk voor *elke* quantumbewerking op n qubits – maar dit zullen we niet behandelen in deze cursus omdat je een ervaren quantumcomponist moet zijn om te begrijpen hoe dit kan!

4.1.4 Circuit-regels

Als je te maken hebt met een heel ingewikkeld quantumcircuit, is het handig om wat trucjes te kennen om dingen te vereenvoudigen. Zulke trucs kunnen het niet alleen makkelijker maken om te begrijpen wat het circuit doet, maar maken het circuit ook efficiënter en dus sneller om uit te voeren op een quantumcomputer. Laten we eens kijken naar een paar eenvoudige voorbeelden van zulke trucjes waarbij maar een enkele qubit betrokken is.

In §2.4.3 zagen we dat elke één-qubitbewerking ofwel een rotatie ofwel een spiegeling is. Het is handig om de werking van deze bewerkingen weer te visualiseren door te bedenken dat volgens §2.1.2 qubit-toestanden een cirkel vormen. Je kan meteen vaststellen dat als dezelzelfde reflectie twee keer wordt toegepast op dezelfde qubit, je terugkomt bij de oorspronkelijke toestand. Dit is visueel te zien omdat twee keer spiegelen om dezelfde as alles weer terugbrengt zoals het was (je kunt dit bijvoorbeeld zien in Fig. 2.4 met de NOT-bewerking). Dit is dus ook het geval voor de Hadamard-bewerking H, waarvan we weten dat het een spiegeling is uit Vgl. (2.21). Laten we deze meetkundige intuïtie controleren door een kleine berekening te maken, en dan zullen we gelijk zien dat je met de Hadamard-gate kan omschakelen tussen de NOT en Z gates.

Oefenopgave 4.5: Z en NOT

We weten uit Vgl. (2.20) dat de Hadamard-gate H als volgt werkt:

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

1. Ga na dat je door opnieuw H toe te passen weer op $|0\rangle$ en $|1\rangle$ uitkomt. Dat wil zeggen, ga na dat
$$H|+\rangle = |0\rangle\,, \qquad H|-\rangle = |1\rangle\,.$$
2. Laat zien dat $HZH = \text{NOT}$, waarbij Z gedefinieerd is in Vgl. (2.12).
3. Laat zien dat $H\,\text{NOT}\,H = Z$.

Een andere interessante vraag is wat er gebeurt als je twee arbitraire rotaties of spiegelingen na elkaar toepast. We weten dat het resultaat opnieuw een rotatie of een spiegeling moet zijn. Maar welke is het en onder welke nieuwe hoek is het? Twee opeenvolgende rotaties zijn gewoon hetzelfde als een enkele rotatie over de som van de twee hoeken, dus $U(\varphi_2)U(\varphi_1) = U(\varphi_1 + \varphi_2)$. De volgende opgave geeft je een regel om twee opeenvolgende spiegelingen te vereenvoudigen tot een enkele *rotatie*.

Oefenopgave 4.6: Spiegelingen en rotaties (optioneel)

Laat zien dat het product van twee spiegelingen een rotatie is. Dus laat zien dat

$$V(\theta_2)V(\theta_1) = U(\theta),$$

voor een bepaalde hoek θ. Kan je de hoek θ uitdrukken in θ_1 en θ_2?

Hint: Gebruik Vgl. (2.19) en dat $U(\varphi_2)U(\varphi_1) = U(\varphi_1 + \varphi_2)$.

Je kunt de andere twee gevallen met één rotatie en één spiegeling zelf uitwerken en controleren of ze beide op een spiegeling $V(\theta)$ uitkomen, voor een bepaalde hoek θ.

4.1.5 Alle qubits meten

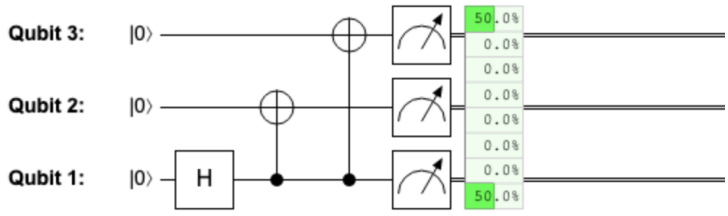
Zodra we klaar zijn met het toepassen van de bewerkingen die onze quantumbits veranderen, willen we wat informatie eruit halen. Net als voorheen is de enige manier om dit te doen door

de qubits te meten.

Wat zijn de regels voor het meten van een quantumtoestand van n qubits? Als we alle n qubits meten, krijgen we als uitkomst n bits, dus een bitstring $a_1 \ldots a_n$. Net als voorheen is de kans op een bepaalde uitkomst $a_1 \ldots a_n$ de gekwadrateerde amplitude:

$$p_{a_1,\ldots,a_n} = \psi^2_{a_1,\ldots,a_n}. \tag{4.6}$$

In QUIRKY kunnen we net als altijd alle qubits meten door voor elke qubit een meetvakje toe te voegen. Om de waarschijnlijkheden van de meetresultaten te bekijken, kunnen we een 'Probability display' toevoegen – maar we moeten er wel voor zorgen dat we het formaat aanpassen zodat dit op alle draden wordt toegepast. Probeer het volgende voorbeeld na te maken:



Deze reeks QUIRKY-bewerkingen maakt de toestand

$$\frac{1}{\sqrt{2}}|000\rangle + \frac{1}{\sqrt{2}}|111\rangle \tag{4.7}$$

en meet alle drie de qubits. Kan je uitwerken hoe dit circuit werkt?

4.1.6 Een paar qubits meten

We kunnen natuurlijk ook alleen maar een deel van de qubits meten. (Dit hebben we vorige week niet eens besproken voor twee qubits, want het hoofdstuk was al erg vol). Stel bijvoorbeeld dat je een drie-qubit-quantumtoestand $|\psi\rangle$ hebt, zoals in Vgl. (4.1), maar in plaats van alle drie de qubits te meten, *meet je alleen de eerste qubit*. Wat is de kans p_a op de uitkomst $a \in \{0,1\}$? Dit is gewoon de som van alle kansen in Vgl. (4.6) die horen bij een string die begint met a:

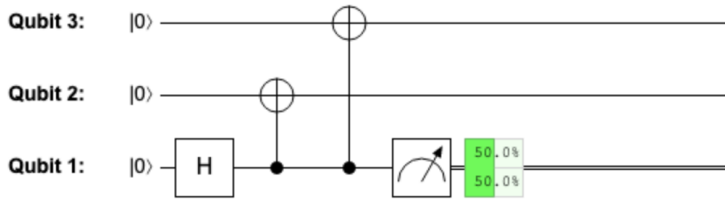
$$p_a = \psi^2_{a00} + \psi^2_{a01} + \psi^2_{a10} + \psi^2_{a11}. \tag{4.8}$$

Als we bijvoorbeeld de eerste qubit van de toestand

$$\frac{1}{\sqrt{8}}|000\rangle + \sqrt{\frac{2}{8}}|010\rangle + \sqrt{\frac{5}{8}}|111\rangle$$

meten, krijgen we de uitkomst 0 met waarschijnlijkheid $1/8 + 2/8 = 3/8$.

We kunnen individuele qubits meten met QUIRKY door maar een enkele meting toe te voegen aan de draad waarin we geïnteresseerd zijn. Om de waarschijnlijkheid van de meetresultaten te bekijken, sleep je een 'Probability display' naar het circuit. Probeer dit nu eens uit. De volgende reeks bewerkingen maakt bijvoorbeeld de toestand uit Vgl. (4.7) en meet alleen de eerste qubit:



(4.9)

Volgens Vgl. (4.8) zouden we inderdaad 0 en 1 met een gelijke waarschijnlijkheid meten.

Wat is de quantumtoestand van de tweede en derde qubit nadat je de eerste qubit hebt gemeten en een uitkomst $a \in \{0,1\}$ hebt gekregen? Volgens dezelfde procedure als voor probabilistische bits in §3.1.3, verzamelen we eerst alle termen van $|\psi\rangle$ waarvan de eerste qubit in de toestand is die overeenkomt met de verkregen uitkomst:

$$\psi_{a00}\,|a00\rangle + \psi_{a01}\,|a01\rangle + \psi_{a10}\,|a10\rangle + \psi_{a11}\,|a11\rangle\,.$$

Vervolgens laten we de eerste qubit in alle vier de termen weg, omdat die al gemeten is:

$$\psi_{a00}\,|00\rangle + \psi_{a01}\,|01\rangle + \psi_{a10}\,|10\rangle + \psi_{a11}\,|11\rangle\,.$$

Tenslotte normaliseren we dit zodat we een geldige twee-qubit-toestand krijgen. Hiervoor willen we een getal c vinden zodat $\frac{\psi_{a00}}{c}\,|00\rangle + \frac{\psi_{a01}}{c}\,|01\rangle + \frac{\psi_{a10}}{c}\,|10\rangle + \frac{\psi_{a11}}{c}\,|11\rangle$ een qubit-toestand is, dus,

$$\left(\frac{\psi_{a00}}{c}\right)^2 + \left(\frac{\psi_{a01}}{c}\right)^2 + \left(\frac{\psi_{a10}}{c}\right)^2 + \left(\frac{\psi_{a11}}{c}\right)^2 = 1.$$

Een algemene plus-/minteken is niet belangrijk, dus we kunnen gewoon

$$c = \sqrt{\psi_{a00}^2 + \psi_{a01}^2 + \psi_{a10}^2 + \psi_{a11}^2},$$

gebruiken, wat de wortel is van de waarschijnlijkheid in Vgl. (4.8).

Kortom, als je de eerste qubit van een drie-qubit-toestand meet zoals in Vgl. (4.1), dan krijg je de uitkomst $a \in \{0,1\}$ met waarschijnlijkheid

$$p_a = \psi_{a00}^2 + \psi_{a01}^2 + \psi_{a10}^2 + \psi_{a11}^2 \tag{4.10}$$

en de resulterende twee-qubit-toestand $|\psi_a\rangle$ van de overgebleven twee qubits is

$$|\psi_a\rangle = \frac{\psi_{a00}\,|00\rangle + \psi_{a01}\,|01\rangle + \psi_{a10}\,|10\rangle + \psi_{a11}\,|11\rangle}{\sqrt{\psi_{a00}^2 + \psi_{a01}^2 + \psi_{a10}^2 + \psi_{a11}^2}}. \tag{4.11}$$

Hoe ziet dit eruit in de situatie van (4.9), waarbij we de toestand $\frac{1}{\sqrt{2}}\,|000\rangle + \frac{1}{\sqrt{2}}\,|111\rangle$ maken en dan de eerste qubit meten? Als de uitkomst van de meting 0 is (wat gebeurt met een waarschijnlijkheid van $1/2$), zitten de andere twee qubits in de toestand

$$\frac{\frac{1}{\sqrt{2}}\,|00\rangle}{\sqrt{\frac{1}{2}}} = |00\rangle\,.$$

Als de uitkomst 1 is, dan zitten de overgebleven qubits in de toestand $|11\rangle$.

Omdat het meten van één van meerdere qubits erg ingewikkeld kan zijn, bespreken we een andere methode om dit te doen. We nemen weer een algemene drie-qubit=toestand $|\psi\rangle$ zoals in Vgl. (4.1) en nemen aan dat we de eerste qubit willen meten. We kunnen de acht termen in de uitdrukking van $|\psi\rangle$ als volgt herschrijven:

$$\begin{aligned} |\psi\rangle = \sqrt{p_0}\,|0\rangle \otimes \frac{\psi_{000}\,|00\rangle + \psi_{001}\,|01\rangle + \psi_{010}\,|10\rangle + \psi_{011}\,|11\rangle}{\sqrt{p_0}} \\ + \sqrt{p_1}\,|1\rangle \otimes \frac{\psi_{100}\,|00\rangle + \psi_{101}\,|01\rangle + \psi_{110}\,|10\rangle + \psi_{111}\,|11\rangle}{\sqrt{p_1}}. \end{aligned} \tag{4.12}$$

Dit kunnen we nu herschrijven als

$$|\psi\rangle = \sqrt{p_0}\,|0\rangle \otimes |\psi_0\rangle + \sqrt{p_1}\,|1\rangle \otimes |\psi_1\rangle\,, \tag{4.13}$$

waarbij de p_a waarschijnlijkheden zijn (namelijk die uit Vgl. (4.10)) en de $|\psi_a\rangle$ quantumtoestanden zijn (de twee-qubit-toestanden uit Vgl. (4.11)).

Als je erin slaagt om je quantumtoestand te schrijven in de vorm van Vgl. (4.13) , kan je gewoon de waarschijnlijkheden van de meetuitkomsten aflezen en ook zien wat de toestand van de overgebleven twee qubits is na de meting. Bijvoorbeeld,

$$\frac{1}{\sqrt{2}}|000\rangle + \frac{1}{\sqrt{2}}|111\rangle = \frac{1}{\sqrt{2}}|0\rangle \otimes |00\rangle + \frac{1}{\sqrt{2}}|1\rangle \otimes |11\rangle\,,$$

Dit bevestigt dat in ons voorbeeld beide uitkomsten gebeuren met een waarschijnlijkheid van $1/2$ en dat de toestand van de overgebleven qubits of $|00\rangle$ of $|11\rangle$ is, afhankelijk van de meetuitkomst. Deze methode om eerst termen te groeperen en ze dan te normaliseren is vrij intuïtief en over het algemeen erg nuttig. Als je deze methode gebruikt, moet je wel oppassen dat je niet vergeet om de toestanden correct te normaliseren! Dat wil zeggen, de constantes $\sqrt{p_a}$ die je voor de twee basistoestanden in Vgl. (4.12) en (4.13) zet, moeten voldoen aan $p_0 + p_1 = 1$ en de toestanden $|\psi_0\rangle$ en $|\psi_1\rangle$ van de overblijvende qubits moeten juist genormaliseerd zijn, zie Vgl. (4.3).

We kunnen precies hetzelfde doen als we meer dan drie qubits hebben, of als we een andere qubit willen meten dan de eerste, of als we meer dan een enkele qubit willen meten! Stel bijvoorbeeld dat we de eerste *twee* qubits van de algemene drie-qubit=toestand $|\psi\rangle$ uit Vgl. (4.1) meten. Dan bestaat het meetresultaat uit twee bits, a en b, die voorkomen met waarschijnlijkheden

$$p_{a,b} = \psi_{ab0}^2 + \psi_{ab1}^2, \tag{4.14}$$

en de overgebleven qubit na de meting zit in de toestand

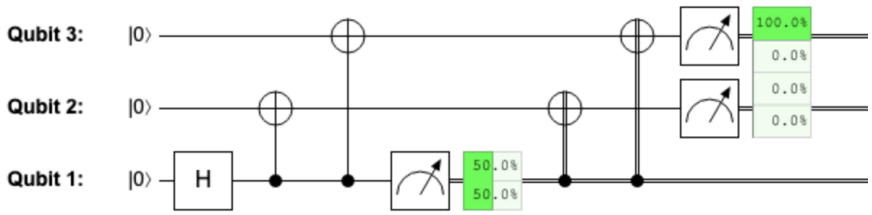
$$|\psi_{a,b}\rangle = \frac{\psi_{ab0}|0\rangle + \psi_{ab1}|1\rangle}{\sqrt{\psi_{ab0}^2 + \psi_{ab1}^2}}. \tag{4.15}$$

Oefenopgave 4.7: Twee van de drie

Wat zijn de kansen op de verschillende mogelijke uitkomsten als je de eerste twee qubits meet van de drie-qubit-toestand uit Vgl. (4.7)? Gebruik QUIRKY om je resultaat te bevestigen.

Als we sommige qubits meten maar andere niet, zullen we vaak de meetresultaten willen gebruiken om te bepalen of een bewerking wel of niet moet worden toegepast op de overgebleven qubits. Stel bijvoorbeeld dat je in de situatie van (4.9) de overgebleven twee qubits wilt resetten naar de toestand $|00\rangle$. Als het meetresultaat nul is, hoeft er niets gedaan te worden. Maar als het meetresultaat één is, dan weten we dat de twee overgebleven qubits in de toestand $|11\rangle$ zitten en willen we ze terugzetten naar de toestand $|00\rangle$. Dit kan gedaan worden door op elk van de qubits een NOT-bewerking toe te passen. Maar hoe weten we of we deze bewerking moeten toepassen of niet, dit hangt af van het eerdere meetresultaat op de eerste qubit. Daarom willen we deze bewerking alleen toepassen als de meetuitkomst 1 is. Met andere woorden, we willen een controlled-NOT-gate toepassen, waarbij de controle nu een klassiek bit is (de uitkomst van de meting) maar het doel nog steeds quantum is.

In QUIRKY kunnen we dit doen op de manier die je zou verwachten, namelijk door een klassiek bit te gebruiken als controle en een quantumbit als doel, zoals in het volgende voorbeeld:



Hier passen we, nadat de eerste qubit gemeten is, nog twee controlled-NOT-bewerkingen toe die gecontroleerd worden door het meetresultaat en dan meten we de resterende twee qubits. De tekening laat zien dat we inderdaad erin geslaagd zijn de twee qubits te resetten, want het meten levert $[00]$ op met een kans van 100%.

Als we dat zouden willen, zouden we deze bewerkingen kunnen beschrijven met behulp van 'hybride' toestanden die bestaan uit één bit en twee qubits, bijvoorbeeld,

$$\mathrm{CNOT}_{1\to 2}[a] \otimes |b, c\rangle = [a] \otimes |a \oplus b, c\rangle \,,$$

maar zo formeel hoeven we niet te zijn.

4.2 Quantum-verrassingen

We gaan nu een aantal interessante verschijnselen bespreken die optreden wanneer we te maken hebben met quantumbits. De volgende secties kunnen vrijwel los van elkaar gelezen worden, dus aarzel niet om te beginnen met het onderdeel dat je het meest interesseert.

4.2.1 Geen klonen

Als we een klassiek bit bekijken, is het eenvoudig om het te *kopiëren* of *klonen* – je hoeft alleen maar naar de bit te kijken en te kopiëren wat je ziet:

$$\begin{aligned} [0] &\mapsto [00], \\ [1] &\mapsto [11]. \end{aligned}$$

Kunnen we quantumbits ook klonen?

Laten we er voor nu even van uitgaan dat dit mogelijk is. Dit zou betekenen dat er een quantumbewerking C bestaat die, gegeven een qubit in de toestand $|\psi\rangle$ en een nieuwe qubit in de toestand $|0\rangle$, als volgt werkt

$$C(|\psi\rangle \otimes |0\rangle) = |\psi\rangle \otimes |\psi\rangle \tag{4.16}$$

om twee kopieën van $|\psi\rangle$ te maken uit een enkele toestand. (Waarom geven we de nieuwe qubit mee? Dit is zodat C evenveel invoer-qubits als uitvoer-qubits heeft).

De kloner werkt bijvoorbeeld als volgt op de basistoestanden:

$$\begin{aligned} C\,|00\rangle &= |00\rangle \,, \\ C\,|10\rangle &= |11\rangle \,. \end{aligned} \tag{4.17}$$

Net als we gemakkelijk een klassiek bit kunnen klonen, is het gemakkelijk om een quantumbewerking te vinden die de basistoestanden kloont. De controlled-NOT-bewerking $\mathrm{CNOT}_{1\to 2}$ van Vgl. (3.46) doet dit bijvoorbeeld.

Maar is er een quantumbewerking die een arbitraire onbekende qubit-toestand kan klonen, dus niet alleen een basistoestand? In de volgende huiswerkopdracht zul je laten zien dat dit *niet* mogelijk is.

Huiswerkopdracht 4.1: Geen klonen

In deze huiswerkopdracht willen we bewijzen dat er geen quantumbewerking C bestaat die voldoet aan Vgl. (4.16). We zullen een techniek gebruiken die we *bewijs door tegenspraak* noemen. Dit betekent dat we laten zien dat als zo'n kloonbewerking C zou bestaan, dit zou leiden tot iets waarvan we weten dat het niet klopt (bijvoorbeeld "$0 = 1$"). Hieruit kunnen we dan concluderen dat zo'n C niet kan bestaan.

We beginnen dus met aan te nemen dat er een quantumbewerking C bestaat die voldoet aan Vgl. (4.16). Je kunt $C(|+\rangle \otimes |0\rangle)$ op twee verschillende manieren berekenen:

1. Gebruik eerst Vgl. (4.16) en schrijf je resultaat in de vorm van Vgl. (3.30).
2. Schrijf eerst $|+\rangle \otimes |0\rangle$ uit in de vorm van Vgl. (3.30), gebruik dan dat C lineair is en pas tenslotte Vgl. (4.16) toe.

Krijg je in beide gevallen hetzelfde antwoord? Zo niet, wat kun je dan concluderen?

Dit beroemde resultaat heet in het Engels de no cloning theorem. Dezelfde conclusie (en waarschijnlijk ook het argument dat je in Huiswerkopdracht 4.1 hebt gegeven) geldt ook voor probabilistische bits! Er is een intuïtieve verklaring waarom we probabilistische en quantuminformatie niet kunnen kopiëren. Als dit mogelijk zou zijn, dan zouden we, gegeven een probabilistische bit in een onbekende toestand p of een qubit in een onbekende toestand $|\psi\rangle$, zoveel kopieën van p en $|\psi\rangle$ kunnen maken als we willen. Met deze kopieën kunnen we ze vervolgens op allerlei manieren meten en de verkregen gegevens gebruiken om de waarschijnlijkheid van p of de amplitudes van $|\psi\rangle$ met willekeurige precisie te schatten (net zoals we in §2.5.1 hebben gedaan om de innerlijke werking van het gele mysterievakje te achterhalen). We zouden dus uit een enkele probabilistische of quantumbit een willekeurige hoeveelheid informatie kunnen halen. Dit zou duidelijk niet mogelijk moeten zijn!

Sterker nog, als dit mogelijk zou zijn, dan zouden we in een heel vreemde wereld leven (veel vreemder dan die beschreven door de quantummechanica)! Stel je bijvoorbeeld een munt voor met een kans op kop van $p = 0,1011010010\ldots$, waarbij de binaire cijfers de hele inhoud van Wikipedia coderen, en ook alle YouTube-video's en alle foto's van katten die je op het internet kan vinden. Als het klonen van probabilistische bits mogelijk zou zijn, zou ik deze munt één keer kunnen opgooien en opschrijven welke uitkomst ik krijg. Dit is een probabilistisch stukje informatie dat gelijk is aan 0 met kans p. Als ik dit probabilistische bit naar jou stuur en je kan het klonen, dan kun je er zoveel kopieën van maken als je wilt en ze dan allemaal meten. Door naar de meetresultaten te kijken en te tellen hoeveel nullen je kreeg, zou je de kans p kunnen schatten. In feite zou je, door voldoende kopieën van het originele bit te maken, deze kans arbitrair goed kunnen schatten! Je zou met name elk binair cijfer van p en dus ook alle informatie die in p gecodeerd is eruit kunnen halen, inclusief het kattenplaatje nummer 65535!

Dit zou duidelijk niet mogelijk moeten zijn, want anders zouden we geen USB-drives, datacentra of dataconnectie voor onze mobiele telefoon nodig hebben – we zouden gewoon al onze informatie in een enkel probabilistisch bit kunnen opslaan en alle informatie kunnen verzenden door dit bit naar iemand anders te sturen! Dit is absoluut te mooi om waar te zijn.

4.2.2 One-time pad

Voordat we teleportatie van quantumtoestanden bespreken, is het handig om eerst een simpelere procedure voor probabilistische bits te begrijpen, die one-time pad wordt genoemd. Met deze procedure kan Alice een bericht versleutelen en naar Bob sturen op een manier dat alleen Bob kan begrijpen wat het bericht is. Dat wil zeggen, in het geval dat iemand anders het bericht onderschept (zoals hun klasgenoot Eve), zou die geen idee hebben wat het echte bericht is. Het feit dat dit überhaupt mogelijk is, is nogal verrassend. Want welk voordeel heeft Bob ten opzichte van Eve zodat hij Alice's bericht wel correct kan ontcijferen terwijl Eve absoluut geen idee heeft wat de inhoud is?

De truc is dat Alice en Bob elkaar eerst moeten ontmoeten in een café. Neem twee muntjes, plak ze aan elkaar met kauwgom, gooi deze 'dubbele munt' op en haal de twee muntjes weer uit elkaar. Alice en Bob nemen elk een van de geworpen muntjes. Nu delen ze een tweetal

willekeurige bits die beschreven worden door de toestand uit (3.5), namelijk

$$r = \frac{1}{2}([00] + [11]).$$

Je kan dit zien als een gezamenlijk geheim! Bovendien weten alleen Alice en Bob wat dit geheim is – om hier achter te komen kunnen ze gewoon naar hun eigen muntjes kijken (en ze dus meten). Ze zullen allebei dezelfde zijde zien, en elk van de zijden komt voor met een waarschijnlijkheid van $1/2$. Dit is een heel goed geheim, want de beste optie voor Eve is om blind te raden!

Laten we nu kijken hoe Alice en Bob hier gebruik van kunnen maken. Stel dat Alice een geheime boodschap $m \in \{0, 1\}$ heeft die ze naar Bob wil sturen. De totale toestand van al hun bits wordt beschreven door de toestand

$$[m] \otimes r = \frac{1}{2}([m00] + [m11]), \tag{4.18}$$

waarbij de eerste twee bits (m en de eerste helft van r) van Alice zijn en de derde bit (de tweede helft van r) van Bob is.

Om haar bericht te versturen, moet Alice naar haar helft van het gedeelde willekeurige bit r kijken en

1. als ze 0 ziet, stuurt ze m naar Bob zonder iets te veranderen,
2. als ze 1 ziet, stuurt ze $\mathrm{NOT}(m)$ naar Bob.

Stel dat Eve deze boodschap onderschept. Wat ziet ze dan? Ongeacht de waarde van m ziet ze 0 met kans $1/2$ en 1 met kans $1/2$. Dit komt omdat Alice m flipt met kans $1/2$, waardoor m effectief gerandomiseerd wordt op zo'n manier dat het voor Eve eruitziet als een uniform willekeurig bit.

Maar hoe zit het met Bob? Lijkt de boodschap van Alice voor hem niet ook uniform willekeurig? Gelukkig heeft Bob de andere helft van het geheime willekeurige bit dat ze gedeeld hebben. Hoewel Alice's bericht oorspronkelijk ook voor hem willekeurig lijkt, kan hij het decoderen door precies dezelfde procedure toe te passen als Alice: kijk naar zijn helft van het gedeelde willekeurige bit en

1. als hij 0 ziet, dan neemt hij Alice's bericht zoals het is,
2. als hij 1 ziet, past hij een NOT-bewerking toe op het bericht van Alice.

In totaal wordt Alice's bericht of verzonden zoals het is of twee keer omgekeerd, wat betekent dat Bob het altijd goed kan begrijpen. Maar vanuit Eve's perspectief is het met waarschijnlijkheid $1/2$ omgekeerd, wat betekent dat zij een uniform willekeurig bit ziet. Dit is dus een volkomen veilige manier voor Alice en Bob om te communiceren!

Laten we eens wat formeler bekijken wat hier gebeurt. Wanneer Alice haar eerste bit omkeert als haar tweede bit gelijk is aan 1, is dit hetzelfde als het toepassen van $\mathrm{CNOT}_{2\to1}$ op haar twee bits. Vervolgens stuurt Alice het eerste bit naar Bob. Wanneer Bob het bericht van Alice ontcijfert, past hij een $\mathrm{CNOT}_{3\to1}$-bewerking toe (wat mogelijk is omdat hij naast het derde bit nu ook het eerste bit heeft). De resulterende toestand is

$$\mathrm{CNOT}_{3\to1}\,\mathrm{CNOT}_{2\to1}([m] \otimes r).$$

We kunnen dit als volgt uitwerken:

$$\mathrm{CNOT}_{3\to1}\,\mathrm{CNOT}_{2\to1}\,\frac{1}{2}([m,0,0] + [m,1,1]) = \mathrm{CNOT}_{3\to1}\,\frac{1}{2}([m,0,0] + [\mathrm{NOT}(m),1,1])$$
$$= \frac{1}{2}([m,0,0] + [\mathrm{NOT}(\mathrm{NOT}(m)),1,1]) = \frac{1}{2}([m,0,0] + [m,1,1]) = [m] \otimes r.$$

Het bericht is dus terug in zijn oorspronkelijke staat, maar nu in het bezit van Bob.

Een interessant aspect van het bovenstaande one-time pad protocol is niet alleen dat Alice een deterministisch bericht $[m]$ naar Bob kan sturen, maar zelfs een probabilistisch bericht. Uit lineariteit volgt dat als Alice's bericht een probabilistisch bit is met distributie p, de begintoestand $p \otimes r$ is en de eindtoestand ook $p \otimes r$, maar dan is p in het bezit van Bob. Vanuit Eve's perspectief is het verzonden bericht nog steeds uniform willekeurig. Het verrassende hieraan is dat Alice door het verzenden van een uniform willekeurig bit erin slaagt om in het geheim een probabilistisch bit te verzenden waarvan ze zelf de verdeling misschien niet eens weet.

Deze procedure lijkt veel op quantumteleportatie, waarbij Alice een qubit-toestand $|\psi\rangle$ naar Bob kan sturen door twee (in plaats van één) uniform willekeurige bits te sturen. Bij teleportatie moeten ze een gedeelde maximaal verstrengelde toestand $|\Phi^+\rangle$ gebruiken in plaats van het gedeelde willekeurige bit r. In beide gevallen wordt de gedeelde toestand gemeten en dus verbruikt tijdens de procedure. In de volgende sectie bespreken we dit verder.

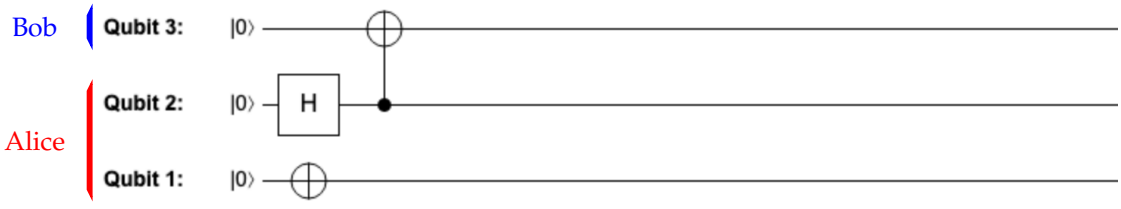
4.2.3 Quantumteleportatie

Hoewel het niet mogelijk is om quantumbits te klonen, kunnen we wel quantumbits van de ene plaats naar de andere verplaatsen. Elke qubit wordt namelijk opgeslagen in een fysiek object of deeltje die de qubit bevat. Als Alice bijvoorbeeld haar qubit opslaat als de polarisatie van een foton, kan ze deze foton gewoon naar Bob sturen. Maar wat nog verrassender is, is dat je een quantumbit van de ene plaats naar de andere kunt verplaatsen door een eindig aantal klassieke bits te sturen (in feite zijn twee bits al voldoende). Dit is zo bijzonder omdat een algemene qubit-toestand $|\psi(\theta)\rangle$ wordt bepaald door een arbitraire hoek θ, zie Vgl. (2.5), die over het algemeen niet kan worden vastgelegd in een eindig aantal bits. Vanwege deze verrassende eigenschap staat deze procedure bekend als teleportatie. We zullen straks zien dat we verstrengeling nodig hebben om dit te laten werken!

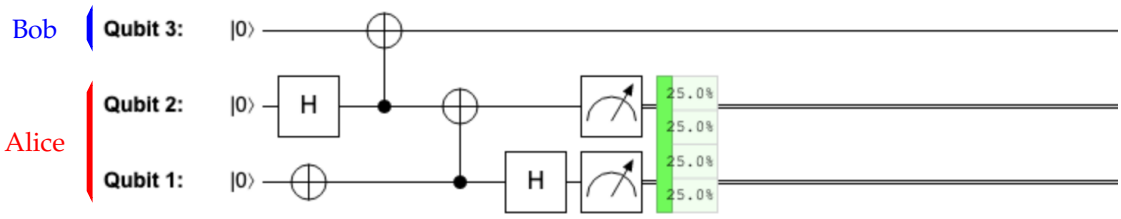
Het uitgangspunt voor teleportatie is het volgende scenario. We stellen ons voor dat Alice twee qubits heeft en Bob één qubit. Alice's eerste qubit is de *boodschap-qubit* die ze naar Bob wil sturen, die bij aanvang in een arbitraire toestand $|\psi\rangle$ zit – die Alice zelf misschien niet eens weet! Haar tweede qubit en Bob's qubit zitten in een maximaal verstrengelde toestand $|\Phi^+\rangle$. De drie qubits zitten dus in de volgende toestand:

$$|\psi\rangle \otimes |\Phi^+\rangle ,$$

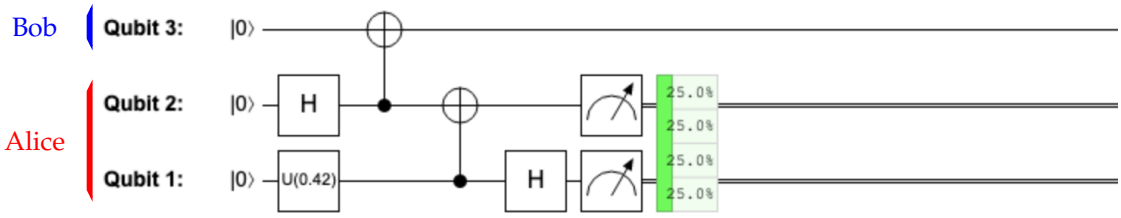
waarbij de eerste twee qubits van Alice zijn en de laatste qubit van Bob (door Vgl. (3.52) weten we dat $|\Phi^+\rangle$ een twee-qubit-toestand is). Het volgende QUIRKY-circuit laat zien hoe dit eruit ziet in het geval dat Alice een qubit wil verzenden in de toestand $|\psi\rangle = |1\rangle = \text{NOT}\,|0\rangle$:



Wat zou de volgende stap kunnen zijn? Uiteindelijk is het de bedoeling dat Bob de qubit van Alice krijgt – aangezien we de qubit niet kunnen klonen, betekent dit dat Alice een actie moet uitvoeren die haar qubit 'vernietigt'. Een goede manier om dit te doen is door een meting uit te voeren. Maar het simpelweg meten van de twee qubits van Alice heeft geen zin, omdat we weten dat we de toestand van $|\psi\rangle$ niet kunnen bepalen uit een enkele meting. Dit betekent dat Alice eerst een quantumbewerking op haar beide qubits moet uitvoeren en ze dan moet meten. Het blijkt dat voor haar de juiste strategie is om dezelfde bewerkingen uit te voeren die je hebt gebruikt om de vier Bell-toestanden uit elkaar te houden in Oefenopgave 3.13:



Merk op dat in dit voorbeeld de vier meetresultaten van Alice elk met 25% kans voorkomen. Hier is een ander voorbeeld, waarin Alice de toestand $|\psi(0.42)\rangle$ probeert te teleporteren:



Het lijkt erop dat, ongeacht de toestand van Alice's boodschap-qubit, de vier waarschijnlijkheden altijd hetzelfde zijn. Dit is een goed teken, want dit betekent dat Alice helemaal geen informatie meer heeft over haar boodschap-qubit! Vergeet niet dat we willen dat haar begintoestand volledig bij Bob terecht komt, wat betekent dat ze geen informatie over deze toestand bij haar mag houden.

In het algemeen ziet de toestand vlak voor de metingen van Alice er als volgt uit:

$$(H \otimes I \otimes I)\,(\mathrm{CNOT}_{1\to 2} \otimes I)\,\left(|\psi\rangle \otimes \left|\Phi^+\right\rangle\right)$$

Merk op dat we hier moeten oppassen, want de laatste twee tensorproducten zijn niet uitgelijnd (zie Oefenopgave 4.3). We doen dus de volgende berekening:

$$\begin{aligned}
&(H \otimes I \otimes I)\,(\mathrm{CNOT}_{1\to 2} \otimes I)\,\left(|\psi\rangle \otimes \left|\Phi^+\right\rangle\right)\\
&= \frac{1}{\sqrt{2}}\,(H \otimes I \otimes I)\,(\mathrm{CNOT}_{1\to 2} \otimes I)\,(\psi_0\,|000\rangle + \psi_0\,|011\rangle + \psi_1\,|100\rangle + \psi_1\,|111\rangle)\\
&= \frac{1}{\sqrt{2}}\,(H \otimes I \otimes I)\,(\psi_0\,|000\rangle + \psi_0\,|011\rangle + \psi_1\,|110\rangle + \psi_1\,|101\rangle)\\
&= \frac{1}{2}\,(\psi_0\,|000\rangle + \psi_0\,|100\rangle + \psi_0\,|011\rangle + \psi_0\,|111\rangle + \psi_1\,|010\rangle - \psi_1\,|110\rangle + \psi_1\,|001\rangle - \psi_1\,|101\rangle)\\
&= |00\rangle \otimes \frac{\psi_0\,|0\rangle + \psi_1\,|1\rangle}{2} + |01\rangle \otimes \frac{\psi_1\,|0\rangle + \psi_0\,|1\rangle}{2}\\
&+ |10\rangle \otimes \frac{\psi_0\,|0\rangle - \psi_1\,|1\rangle}{2} + |11\rangle \otimes \frac{-\psi_1\,|0\rangle + \psi_0\,|1\rangle}{2}.
\end{aligned}$$

Dit is de totale toestand vlak voor de meting van Alice. We kunnen de waarschijnlijkheid van haar meetresultaten berekenen zoals we besproken hebben in Vgl. (4.14). Om de kans $p_{a,b}$ op de uitkomst $[ab]$ te berekenen tellen we gewoon de gekwadrateerde amplitudes van de relevante basistoestanden $|ab0\rangle$ en $|ab1\rangle$ op. In elk van de vier gevallen is een van de amplitudes $\psi_0/2$ en

de andere $\pm\psi_1/2$, dus de som van de kwadraten is altijd hetzelfde:

$$
\begin{aligned}
p_{00} &= \left(\frac{\psi_0}{2}\right)^2 + \left(\frac{\psi_1}{2}\right)^2 = \frac{\psi_0^2+\psi_1^2}{4} = \frac{1}{4}, \\
p_{01} &= \left(\frac{\psi_1}{2}\right)^2 + \left(\frac{\psi_0}{2}\right)^2 = \frac{1}{4}, \\
p_{10} &= \left(\frac{\psi_0}{2}\right)^2 + \left(\frac{-\psi_1}{2}\right)^2 = \frac{1}{4}, \\
p_{11} &= \left(\frac{-\psi_1}{2}\right)^2 + \left(\frac{\psi_0}{2}\right)^2 = \frac{1}{4}.
\end{aligned}
$$

Elke uitkomst komt voor met een 25% kans. Dit bevestigt wat we eerder hebben gezien met QUIRKY.

Na de meting van Alice is de enige overgebleven quantumbit de qubit van Bob. Laten we de toestand hiervan aanduiden met $|\psi'_{a,b}\rangle$, omdat deze afhangt van het meetresultaat van Alice. We kunnen deze toestand bepalen met Vgl. (4.15) of, veel eenvoudiger, met de groepering- en normalisatiemethode van Vgl. (4.13). Hoe dan ook, het resultaat is dat Bob's qubit zich in een van de volgende vier toestanden bevindt:

$$
\begin{aligned}
|\psi'_{00}\rangle &= \psi_0\,|0\rangle + \psi_1\,|1\rangle\,, \\
|\psi'_{01}\rangle &= \psi_1\,|0\rangle + \psi_0\,|1\rangle\,, \\
|\psi'_{10}\rangle &= \psi_0\,|0\rangle - \psi_1\,|1\rangle\,, \\
|\psi'_{11}\rangle &= -\psi_1\,|0\rangle + \psi_0\,|1\rangle\,.
\end{aligned}
$$

Als $[ab] = [00]$, valt Bob's toestand precies samen met Alice's oorspronkelijke toestand $|\psi\rangle$, die ze wilde teleporteren:

$$|\psi'_{00}\rangle = |\psi\rangle\,.$$

In de andere drie gevallen is de quantumtoestand van Bob een beetje vervormd. Maar als Alice haar meetresultaten (dus de twee bits a en b) naar Bob stuurt, dan kan hij een geschikte correctiebewerking toepassen om zijn toestand te 'corrigeren':

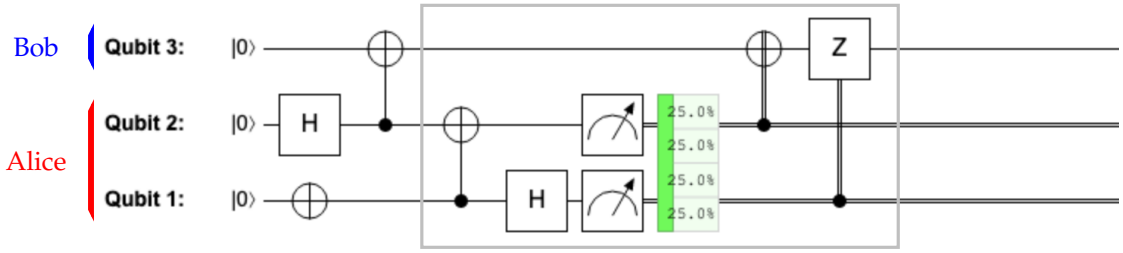
$$
\begin{aligned}
\text{NOT}\,|\psi'_{01}\rangle &= |\psi\rangle\,, \\
Z\,|\psi'_{10}\rangle &= |\psi\rangle\,, \\
Z\,\text{NOT}\,|\psi'_{11}\rangle &= |\psi\rangle\,.
\end{aligned}
$$

Deze vier gevallen kunnen worden samengevat in de volgende simpele procedure voor Bob:

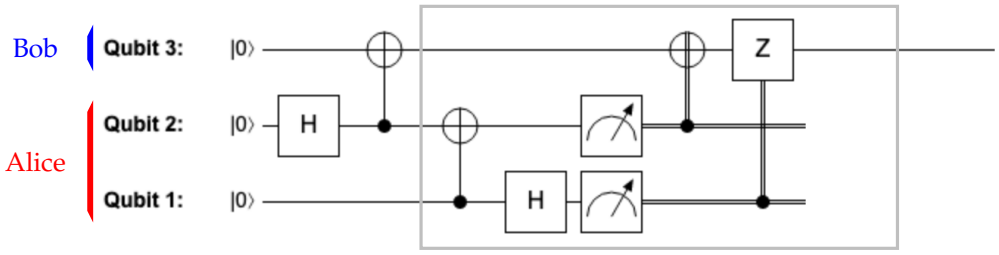
1. kijk naar bit b en als $b = 1$ pas dan NOT toe,
2. kijk naar bit a en als $a = 1$ pas dan Z toe.

We kunnen dit implementeren met een controlled-NOT- en een controlled-Z-bewerking, waarbij de controles klassieke bits zijn. Je kunt de controlled-Z-bewerking op dezelfde manier maken als de controlled-NOT-bewerking – dit is besproken aan het eind van §3.2.4. Jeetje, dit was nogal wat werk!

Hoe ziet dit er allemaal uit in QUIRKY? Het eindresultaat is het volgende quantumcircuit:

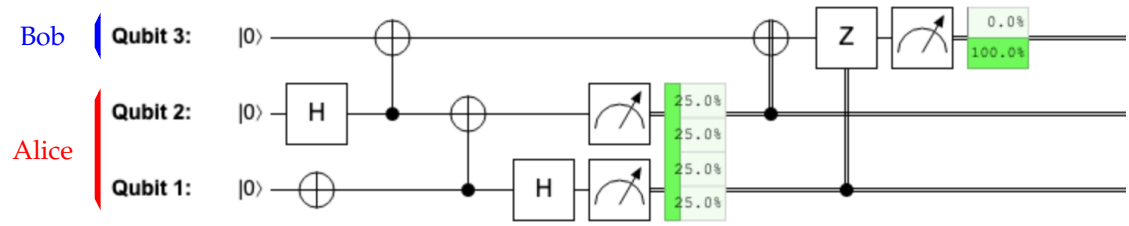


We hebben een grijs blok gemaakt van het relevante deel – het teleportatie-circuit – om het te scheiden van de rest van het circuit dat de invoertoestanden maakt. Hier is een plaatje dat alleen het teleportatie-circuit en het maken van de maximaal verstrengelde toestand laat zien, zonder iets met Alice's eerste qubit te doen:



We hebben ook de draden van de twee klassieke bits van Alice weggeknipt, omdat die niet meer van belang zijn zodra Alice de twee meetresultaten naar Bob heeft gestuurd. Verder hebben we de 'probability display' weggehaald, omdat dit geen echte quantumbewerking is, maar een manier om het circuit in QUIRKYte kunnen bekijken. Er is dus één input-qubit voor de boodschap van Alice, twee qubits voor de maximaal verstrengelde toestand en één output-qubit aan de kant van Bob. Het effect van de teleportatie is dat er een arbitraire toestand doorgegeven wordt van de input-qubit aan Alice's kant naar de output-qubit aan Bob's kant. Het belangrijkste is dat er in het grijze blok alleen klassieke bits van Alice naar Bob worden gestuurd!

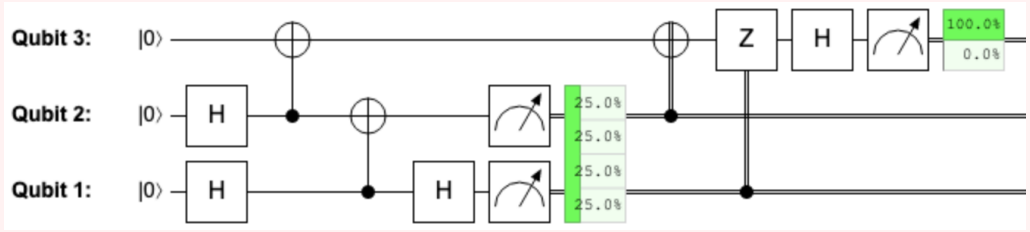
Laten we controleren of we geen fout hebben gemaakt. We verwachten dat de qubit van Bob na de teleportatieprocedure in de toestand $|1\rangle$ terechtkomt, dus we kunnen een simpele meting toevoegen om te testen of dit inderdaad gebeurd is:



We krijgen inderdaad de uitkomst 1 met een 100% kans, wat bevestigt dat we geslaagd zijn in het teleporteren van de $|1\rangle$ toestand! Maar $|1\rangle$ is nou niet bepaald een hele spannende toestand om te teleporteren. Hoe zit het met de toestand $|+\rangle$, die eerder problemen gaf toen we het over klonen hadden? In de volgende huiswerkopdracht ga je het teleportatie-circuit eerst testen voor de $|+\rangle$ toestand en daarna voor arbitraire één-qubit-toestanden.

Huiswerkopdracht 4.2: Teleportatie testen

1. Waarom bevestigt het volgende circuit dat de toestand $|+\rangle$ correct is geteleporteerd?



2. Hoe kun je op een vergelijkbare manier testen of een quantumtoestand $|\psi(\theta)\rangle$ correct geteleporteerd is?

Het teleportatie-circuit is heel opmerkelijk. Het maakt het mogelijk om een quantumbit van Alice naar Bob te sturen door alleen twee klassieke bits te versturen, zolang Alice en Bob een maximaal verstrengelde toestand delen. Maar dit is niet alleen nuttig voor toepassingen (we zullen er hieronder een paar bespreken), maar het geeft ook een interessant perspectief op het verschil tussen klassieke en quantumbits. Aan het eind van vorige week hebben we superdense coding besproken, waarmee we twee bits konden versturen door één quantumbit te versturen, ook weer met behulp van één maximaal verstrengelde toestand tussen Alice en Bob. Hieruit blijkt dat:

Met een onbeperkte hoeveelheid quantumverstrengeling is het versturen van twee bits volledig gelijk aan het versturen van één quantumbit!

Merk op dat we bij zowel teleportatie als superdense coding de maximaal verstrengelde toestand niet kunnen hergebruiken nadat de procedure is voltooid – het is de 'brandstof' die door beide procedures wordt verbruikt.

4.2.4 Een glimp van quantumnetwerken

Door herhaaldelijk teleportatie te gebruiken, kunnen we quantumbits communiceren tussen verafgelegen knooppunten. Stel bijvoorbeeld dat Alice, haar robot-ezel en Bob zich in de volgende situatie bevinden:

$$\text{Alice's robot-ezel} \longleftrightarrow \text{Alice} \longleftrightarrow \text{Bob},$$

waarbij elke '↔-pijl een maximaal verstrengelde toestand aangeeft. Dat wil zeggen, de gezamenlijke toestand van onze drie spelers is de volgende vier-qubit-toestand:

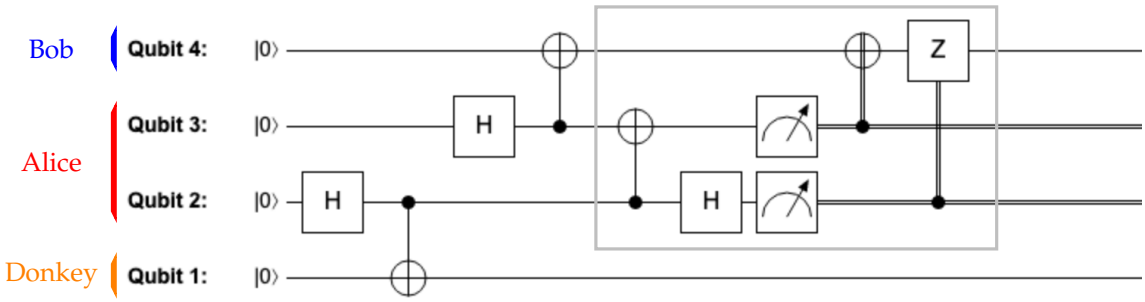
$$\left|\Phi^+\right\rangle \otimes \left|\Phi^+\right\rangle,$$

waarbij de middelste twee qubits van Alice zijn – de eerste is verstrengeld met de robot en de tweede met Bob.

Merk op dat de robot-ezel niet direct verstrengeld is met Bob! Als het toch een quantumboodschap had om naar Bob te sturen, dan zou dit nogsteeds gedaan kunnen worden door de teleportatieprocedure twee keer uit te voeren: Eerst teleporteren we de boodschap van de ezel robot naar Alice (waarbij we de eerste maximaal verstrengelde toestand verbruiken) en vervolgens van Alice naar Bob (waarbij we de overgebleven maximaal verstrengelde toestand verbruiken). Dit is vergelijkbaar met hoe je mobiele telefoon bijvoorbeeld verbinding maakt met een basisstation in de buurt, dat op zijn beurt het signaal 'herhaalt' of 'doorgeeft' aan een andere gsm-mast (enzovoort). Hoewel we quantummechanisch gezien een qubit niet kunnen kopiëren, vanwege het 'no cloning theorem' (c.f. Huiswerkopdracht 4.1), kunnen we de qubit wel over lange afstanden teleporteren!

Verstrengeling is niet alleen nuttig voor teleportatie maar ook voor veel andere toepassingen. Is er ook een manier om teleportatie te gebruiken om verstrengeling te veroorzaken tussen Alice's robot-ezel en Bob (wat ze dan voor andere doeleinden kunnen gebruiken)?

Intuïtief gezien lijkt het erop dat Alice gewoon haar eerste qubit (degene die verstrengeld is met de robot-ezel) naar Bob moet teleporteren. In QUIRKY zou dit er als volgt uitzien:



Hier maken we eerst de twee maximaal verstrengelde toestanden en passen we vervolgens hetzelfde teleportatie-circuit toe als hierboven (grijze blok). We hopen dat dit een maximale verstrengelde toestand oplevert tussen de robot-ezel en Bob. In de volgende huiswerkopdracht kan je laten zien dat dit inderdaad het geval is.

Huiswerkopdracht 4.3: Een vestrengelde qubit teleporteren

Bevestig met QUIRKY dat aan het einde van get circuit de qubit van de ezel en de qubit van Bob in de maximaal verstrengelde toestand $|\Phi^+\rangle$ zitten.

We kunnen op dezelfde manier teleportatie gebruiken om verstrengeling te veroorzaken tussen steeds verder weg gelegen knooppunten. Stel bijvoorbeeld dat we in de volgende situatie zitten:

$$\text{Alice's robot-ezel} \longleftrightarrow \text{Alice} \longleftrightarrow \text{Bob} \longleftrightarrow \text{Bob's robot-eekhoorn.}$$

Als Alice eerst haar eerste qubit naar Bob teleporteert en Bob vervolgens zijn eerste qubit naar zijn robot-eekhoorn teleporteert, geeft dit een maximaal verstrengelde toestand tussen de twee robots. Laten we hopen dat de robots deze verstrengeling alleen voor goedaardige doeleinden gebruiken!

Het tot stand brengen van verstrengeling over lange afstanden zal een belangrijke functie zijn zodra we quantumcomputers met elkaar proberen te verbinden in een klein netwerk of, als we een beetje gewaagd durven dromen, in een toekomstig 'quantum internet'. Een aantal van ons is al druk aan het nadenken over hoe we dit in de praktijk kunnen realiseren en hoe we verstrengeling over lange afstanden het beste kunnen gebruiken voor interessante toepassingen.

4.2.5 De onzekerheidsrelatie

Voor het laatste verschijnsel dat we willen bespreken, hebben we maar één quantumbit nodig. In Vgl. (2.6) hebben we de regels voor het meten van een enkel quantumbit gezien, zoals in de volgende afbeelding:

Gegeven een quantumtoestand $|\psi\rangle = \psi_0 |0\rangle + \psi_1 |1\rangle$, kunnen we twee mogelijke uitkomsten krijgen met waarschijnlijkheden

$$p_0 = \psi_0^2, \qquad p_1 = \psi_1^2. \tag{4.19}$$

Wat zijn de deterministische toestanden waarbij we één van de uitkomsten met zekerheid meten? Dit zijn toestanden waarbij de ene waarschijnlijkheid 100% is en de andere 0%. Met andere woorden, de toestanden waarbij de ene amplitude ± 1 is en de andere nul. Op een eventueel minteken na, waarvan we uit Oefenopgave 2.7 weten dat dit niet relevant is, zijn er maar twee van zulke toestanden:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \tag{4.20}$$

oftewel de basistoestanden. Dit zijn de enige toestanden waarvoor we de meetuitkomst perfect kunnen voorspellen, dus we stellen dat er geen onzekerheid is in de meetuitkomst.

Wat gebeurt er als we eerst een bewerking op de qubit uitvoeren en daarna een meting? Stel bijvoorbeeld dat we eerst een Hadamard-bewerking uitvoeren en dan een meting, zoals in de volgende afbeelding:

Hier zijn de enige toestanden waarvoor we volledig zeker zijn over de meetuitkomst

$$|+\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix}, \qquad |-\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix} \tag{4.21}$$

(op een algemene minteken na). Dit komt omdat de Hadamard-bewerking $|+\rangle, |-\rangle$ weer terugbrengt naar de basistoestanden $|0\rangle, |1\rangle$ (dit heb je laten zien in Oefenopgave 4.5); en deze laatste toestanden zijn precies de toestanden met volledige zekerheid over de uiteindelijke meting, zoals we hierboven hebben besproken. We kunnen dit ook zien door de waarschijnlijkheid van de twee mogelijke meetuitkomsten te berekenen

$$q_0 = \frac{(\psi_0 + \psi_1)^2}{2}, \qquad q_1 = \frac{(\psi_0 - \psi_1)^2}{2}. \tag{4.22}$$

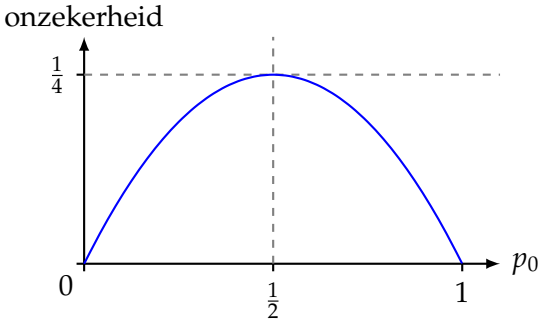
Als $q_1 = 0$ dan is de toestand $|+\rangle$, en als $q_0 = 0$ dan is de toestand $|-\rangle$ (op een algemeen minteken na).

Dit zijn allemaal berekeningen die we al vaker hebben gezien – maar er is een interessante observatie die we nog niet eerder hebben gedaan. Aangezien geen enkele toestand voorkomt in *zowel* Vgl. (4.20) en (4.21) voorkomt, betekent dit dat voor elke toestand minstens één van de twee procedures een onzekerheid in het meetresultaat zal hebben. Dit is niets anders dan de beroemde *onzekerheidsrelatie van Heisenberg*!

Kunnen we deze observatie kwantitatief maken? We hebben eerst een manier nodig om de onzekerheid of 'willekeurigheid' van een kansverdeling $p = \binom{p_0}{p_1}$ te kwantificeren. De volgende functie is hier een goede keuze voor:

$$\text{onzekerheid}(p) = p_0(1 - p_0) = p_0 p_1.$$

Deze functie zet een kansverdeling p om in een getal in het interval $[0, 1/4]$. De uitkomst is precies 0 in het geval dat $p_0 = 0$ of $p_0 = 1$, en is maximaal als beide uitkomsten even waarschijnlijk zijn (d.w.z. als $p_0 = p_1 = 1/2$). Hier is een plot die deze eigenschappen bevestigt:



Stel nu dat we beginnen met een toestand $|\psi\rangle$ en een van de twee hierboven beschreven procedures uitvoeren. Dat wil zeggen, we meten de toestand direct of we passen eerst een Hadamard-bewerking toe en meten dan. De bijbehorende onzekerheden zijn onzekerheid(p) en onzekerheid(q), waarbij de verdelingen $p = \binom{p_0}{p_1}$ en $q = \binom{q_0}{q_1}$ gegeven zijn in Vgl. (4.19) en (4.22) hierboven. Dan geldt:

$$\text{onzekerheid}(p) + \text{onzekerheid}(q) > 0.$$

Deze ongelijkheid betekent inderdaad dat er geen toestand bestaat waarvoor beide onzekerheden tegelijkertijd nul zijn. In de volgende huiswerkopdracht zul je een krachtiger resultaat laten zien:

Huiswerkopdracht 4.4: Afweging van onzekerheid

Laat zien dat voor elke qubit-toestand $|\psi\rangle$:

$$\text{onzekerheid}(p) + \text{onzekerheid}(q) = \frac{1}{4}. \tag{4.23}$$

Vind verder een qubit-toestand $|\psi\rangle$ met $\text{onzekerheid}(p) = \text{onzekerheid}(q)$.

Bonusvraag: Maak deze toestand met behulp van QUIRKY en gebruik QUIRKY om te bevestigen dat $\text{onzekerheid}(p) = \text{onzekerheid}(q)$.

De formule die je zojuist hebt bewezen in Huiswerkopdracht 4.4 is heel opmerkelijk: Het laat zien dat er een simpele afweging is tussen de onzekerheden van de twee procedures. In het bijzonder, als de ene procedure geen onzekerheid heeft dan produceert de andere een uniform willekeurige uitkomst![12]

[12] We kunnen dit ook direct zien. Als we bijvoorbeeld direct $|+\rangle$ meten (een toestand die nul onzekerheid heeft in de tweede procedure) zonder eerst een Hadamard toe te passen, krijgen we de uitkomsten 0 en 1 met gelijke waarschijnlijkheid.

4.3 Oplossingen van de oefenopgaven

Oplossing van Oefenopgave 4.1

$$\begin{aligned}
&\left|\Phi^-\right\rangle \otimes \left|\Psi^-\right\rangle \\
&= \left(\frac{1}{\sqrt{2}}\left|00\right\rangle - \frac{1}{\sqrt{2}}\left|11\right\rangle\right) \otimes \left(\frac{1}{\sqrt{2}}\left|01\right\rangle - \frac{1}{\sqrt{2}}\left|10\right\rangle\right) \\
&= \frac{1}{2}\left|0001\right\rangle - \frac{1}{2}\left|0010\right\rangle - \frac{1}{2}\left|1101\right\rangle + \frac{1}{2}\left|1110\right\rangle .
\end{aligned}$$

Oplossing van Oefenopgave 4.2

Laten we eerst de producttoestand uitschrijven in de vorm van Vgl. (4.2):

$$\left|\Phi^+\right\rangle \otimes \left|1\right\rangle = \frac{1}{\sqrt{2}}\left|001\right\rangle + \frac{1}{\sqrt{2}}\left|111\right\rangle .$$

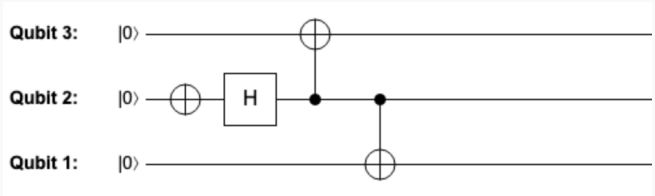
Hieruit volgt:

$$\begin{aligned}
&H_2\left(\left|\Phi^+\right\rangle \otimes \left|1\right\rangle\right) = H_2\left(\frac{1}{\sqrt{2}}\left|001\right\rangle + \frac{1}{\sqrt{2}}\left|111\right\rangle\right) = \frac{1}{\sqrt{2}}H_2\left|001\right\rangle + \frac{1}{\sqrt{2}}H_2\left|111\right\rangle \\
&= \frac{1}{\sqrt{2}}\left|0\right\rangle \otimes H\left|0\right\rangle \otimes \left|1\right\rangle + \frac{1}{\sqrt{2}}\left|1\right\rangle \otimes H\left|1\right\rangle \otimes \left|1\right\rangle \\
&= \frac{1}{\sqrt{2}}\left|0\right\rangle \otimes \left|+\right\rangle \otimes \left|1\right\rangle + \frac{1}{\sqrt{2}}\left|1\right\rangle \otimes \left|-\right\rangle \otimes \left|1\right\rangle \\
&= \frac{1}{\sqrt{2}}\left|0\right\rangle \otimes \left(\frac{1}{\sqrt{2}}\left|0\right\rangle + \frac{1}{\sqrt{2}}\left|1\right\rangle\right) \otimes \left|1\right\rangle \; + \; \frac{1}{\sqrt{2}}\left|1\right\rangle \otimes \left(\frac{1}{\sqrt{2}}\left|0\right\rangle - \frac{1}{\sqrt{2}}\left|1\right\rangle\right) \otimes \left|1\right\rangle \\
&= \frac{1}{2}\left|001\right\rangle + \frac{1}{2}\left|011\right\rangle + \frac{1}{2}\left|101\right\rangle - \frac{1}{2}\left|111\right\rangle ,
\end{aligned}$$

waarbij we Vgl. (2.20) hebben gebruikt om de werking van H op de basistoestanden te vinden.

Oplossing van Oefenopgave 4.3

1. In Oefenopgave 3.12 hebben we al gezien hoe we $\left|\Phi^-\right\rangle$ kunnen maken. De volgende schakeling is dus geschikt:



2. Dit is de uiteindelijke toestand:

$$\begin{aligned}
(\mathrm{CNOT}_{2\to 1} \otimes I)(\left|0\right\rangle \otimes \left|\Phi^-\right\rangle) &= (\mathrm{CNOT}_{2\to 1} \otimes I)\left(\frac{1}{\sqrt{2}}\left|000\right\rangle - \frac{1}{\sqrt{2}}\left|011\right\rangle\right) \\
&= \frac{1}{\sqrt{2}}\left|000\right\rangle - \frac{1}{\sqrt{2}}\left|111\right\rangle .
\end{aligned}$$

Oplossing van Oefenopgave 4.4

We laten

$$\begin{aligned}|\psi\rangle = \psi_{000}\,|000\rangle + \psi_{001}\,|001\rangle + \psi_{010}\,|010\rangle + \psi_{011}\,|011\rangle \\ + \psi_{100}\,|100\rangle + \psi_{101}\,|101\rangle + \psi_{110}\,|110\rangle + \psi_{111}\,|111\rangle\end{aligned}$$

een arbitraire drie-qubit-quantumtoestand zijn. Het resultaat van hierop de Toffoli-bewerking toepassen is

$$\begin{aligned}|\psi'\rangle = T\,|\psi\rangle = \psi_{000}\,|000\rangle + \psi_{001}\,|001\rangle + \psi_{010}\,|010\rangle + \psi_{011}\,|011\rangle \\ + \psi_{100}\,|100\rangle + \psi_{101}\,|101\rangle + \psi_{110}|\mathbf{111}\rangle + \psi_{111}|\mathbf{110}\rangle.\end{aligned}$$

We hebben de twee basistoestanden die zijn veranderd dikgedrukt gemaakt. Het enige dat veranderd is, is dat de amplitudes van $|110\rangle$ en $|111\rangle$ verwisseld zijn. Het is dus duidelijk dat als $\sum_{a,b,c=0}^{1} \psi_{a,b,c}^2 = 1$, dan is ook $\sum_{a,b,c=0}^{1} (\psi'_{a,b,c})^2 = 1$. Dus, T beeldt quantumtoestanden af op quantumtoestanden.

Oplossing van Oefenopgave 4.5

1. Als we H toepassen op $|+\rangle$ en $|-\rangle$ krijgen we

$$\begin{aligned}H\,|+\rangle = H\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = \frac{1}{2}\big((|0\rangle + |1\rangle) + (|0\rangle - |1\rangle)\big) = |0\rangle\,, \\ H\,|-\rangle = H\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = \frac{1}{2}\big((|0\rangle + |1\rangle) - (|0\rangle - |1\rangle)\big) = |1\rangle\,.\end{aligned}$$

2. Om deze identiteit te bewijzen, hoeven we alleen na te gaan dat HZH dezelfde werking op de basistoestanden heeft als NOT (door lineariteit hebben ze dan dezelfde werking op alle qubit-toestanden):

$$\begin{aligned}HZH\,|0\rangle = HZ\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = H\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = H\,|-\rangle = |1\rangle\,, \\ HZH\,|1\rangle = HZ\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = H\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = H\,|+\rangle = |0\rangle\,.\end{aligned}$$

   In beide gevallen inverteert HZH de bit, dus dit is dezelfde bewerking als NOT.

3. In het eerste deel van deze opgave hebben we gezien dat het twee keer toepassen van de Hadamard-gate niks doet: $HH = I$. Dus,

$$Z = (HH)Z(HH) = H(HZH)H = H\,\text{NOT}\,H$$

   waarbij de laatste stap volgt uit deel 2 van deze opgave.

Oplossing van Oefenopgave 4.6

1. Als we gebruik maken van Vgl. (2.15) volgt dat,

$$U(\theta_2)U(\theta_1)\,|\psi(\alpha)\rangle = U(\theta_2)\,|\psi(\alpha+\theta_1)\rangle = |\psi(\alpha+\theta_1+\theta_2)\rangle = U(\theta_1+\theta_2)\,|\psi(\alpha)\rangle\,.$$

   Dit geldt voor elke toestand $|\psi(\alpha)\rangle$, wat betekent dat $U(\theta_2)U(\theta_1) = U(\theta)$, waarbij $\theta = \theta_1 + \theta_2$. Dit is meetkundig gezien ook logisch: als je eerst met een hoek van θ_1 en daarna met een hoek van θ_2 draait, komt dit samen neer op een rotatie met een hoek van $\theta = \theta_1 + \theta_2$.

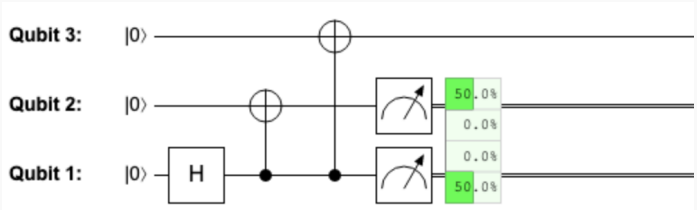
2. Hier is een mooie oplossing. In Vgl. (2.19) zagen we dat een algemene spiegeling op twee manieren kan worden uitgedrukt: $V(\theta) = \mathrm{NOT}\,U(\theta) = U(-\theta)\,\mathrm{NOT}$. Als we voor $V(\theta_1)$ de eerste uitdrukking gebruiken en voor $V(\theta_2)$ de tweede uitdrukking, volgt er dat

$$V(\theta_2)V(\theta_1) = U(-\theta_2)\,\mathrm{NOT}\,\mathrm{NOT}\,U(\theta_1) = U(-\theta_2)U(\theta_1) = U(\theta_1-\theta_2),$$

   waarbij we gebruikt hebben dat het toepassen van twee opeenvolgende NOTs hetzelfde is als niets doen en dat twee opeenvolgende rotaties hetzelfde zijn als een enkele rotatie waarbij de twee hoeken worden opgeteld, zoals hierboven staat.

Oplossing van Oefenopgave 4.7

Volgens Vgl. (4.14), zouden we [00] en [11] moeten krijgen, met voor elk een 50% kans. Dit blijkt inderdaad te kloppen:



Quest 5: Virstuoso van algoritmes

Een van de belangrijkste redenen om quantumcomputers te onderzoeken is omdat quantumcomputers sommige vraagstukken veel sneller kunnen oplossen dan de computers die we nu hebben. Om quantumcomputers te onderscheiden van gewone computers, zullen we de term klassieke computer gebruiken om te verwijzen naar elk rekenapparaat dat op dit moment bestaat. Hieronder vallen je laptop en desktopcomputer, maar ook hele kleine computers zoals in je smartphone of smartwatch en hele grote en krachtige *supercomputers* die misschien wel een hele kamer in beslag nemen. Wat al deze computers onderscheidt van quantumcomputers is niet hoe groot, klein, langzaam of snel ze zijn, maar dat hun interne werking beschreven kan worden door klassieke fysica (elektromagnetisme om precies te zijn). Met andere woorden, hun hardware werkt op een manier die beschreven kan worden door de oude natuurkundige theorieën van voor de quantummechanica. Dit is een beetje te vergelijken met hoe de muziek van Mozart en Bach *klassieke muziek* wordt genoemd. Net als klassieke computers, maakt klassieke muziek ook niet optimaal gebruik van meer geavanceerde muziekinstrumenten, zoals de elektrische gitaar of synthesizers.

Als gevolg van het type hardware van klassieke computers wordt alle informatie die ze opslaan - of het nu een foto, een geluidsbestand, een video of een webpagina is - gepresenteerd door bitstrings, oftewel lange reeksen van nullen en enen. Deze informatie wordt verwerkt door een aantal regels te volgen die beschrijven hoe deze nullen en enen moeten worden aangepast om een bruikbaar antwoord te krijgen. We noemen deze reeks instructies een algoritme. Je kunt een algoritme zien als een recept – het is een reeks instructies die, als je ze zorgvuldig opvolgt, je het gewenste resultaat geeft, zoals een brownie! Een algoritme kan bijvoorbeeld twee binaire strings als invoer nemen en als uitvoer een andere binaire string produceren die bestaat uit de som van de twee oorspronkelijke strings (als je ze interpreteert als getallen). Net als recepten werden algoritmes oorspronkelijk uitgevoerd door mensen. Het woord 'computer' verwees zelfs naar een persoon die berekeningen maakt. Maar tegenwoordig worden algoritmes uitgevoerd op echte computers. Omdat computers over het algemeen niet erg slim zijn, moeten het algoritme op een extreem precieze manier beschreven worden. Deze beschrijving, een programma, is een tastbare uitvoering van het abstracte algoritme op een manier dat de computer het kan begrijpen. Hiervoor gebruiken we een programmeertaal die de computer zelf kan omzetten in basisbewerkingen op de nullen en enen, om deze vervolgens uit te voeren in de hardware.

Als je een algoritme bedenkt, in je computer programmeert en dan uitvoert, wil je het resultaat binnen een redelijke tijdsduur krijgen. De werkelijke tijd die nodig is om je antwoord te krijgen hangt echter af van veel factoren:

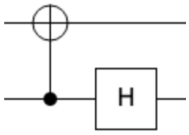
1. hoe snel je computer verschillende basisinstructies kan uitvoeren,
2. of de invoer van je programma wordt gelezen van een harde schijf, een netwerkverbinding of rechtstreeks uit het geheugen,
3. welke programmeertaal je gebruikt om je programma mee te maken (en de versie van de compiler of interpreter die het programma uitvoert),

enzovoort. Dit maakt het erg moeilijk om verschillende algoritmen met elkaar te vergelijken.

Om de vergelijking van verschillende algoritmen makkelijker en eerlijker te maken, kijken computerwetenschappers niet naar hoe lang het duurt om het algoritme uit te voeren op een specifieke computer die op een bepaalde manier is ingesteld. In plaats daarvan tellen ze het aantal elementaire bewerkingen dat het algoritme uitvoert. Op deze manier weten ze zeker dat ze de algoritmes zelf vergelijken en niet de computers waarop de algoritmen worden uitgevoerd (een goed algoritme dat op een hele langzame computer wordt uitgevoerd kan namelijk slechter

lijken dan een slecht algoritme dat op een hele snelle computer wordt uitgevoerd). Om precies te zijn, wat computerwetenschappers willen weten is hoe het aantal bewerkingen groeit met de *grootte* van het probleem dat het algoritme moet oplossen. Hoe groter de hoeveelheid gegevens die je moet verwerken, hoe langer het duurt, ongeacht hoe goed het algoritme is. Wat je dus wilt weten is of je algoritme nog steeds geschikt is als de gegevens extreem groot worden. Het onderdeel van de computerwetenschap dat dit bestudeert, heet computationele complexiteit.

Bij quantumcomputing zijn we geïnteresseerd in het ontwerpen van quantumalgoritmes die problemen oplossen door quantumtoestanden te manipuleren in plaats van bitstrings. De elementaire instructies die we zullen gebruiken zijn gates, zoals de Hadamard-gate, de controlled-NOT-gate of een meting. We zullen onze quantumalgoritmes grafisch beschrijven aan de hand van een quantumcircuit, waarmee je al veel ervaring hebt opgedaan in QUIRKY. Maar we kunnen ook een schriftelijke weergave gebruiken, zoals je zou verwachten van een gewoon computerprogramma. Het linker circuit zou bijvoorbeeld kunnen worden weergegeven door de tekst van het rechter programma:




```
CNOT q1 -> q2;
H q1;
```

waarbij `q1` en `q2` verwijzen naar de twee qubits (denk eraan dat de onderste draad overeenkomt met het *eerste* qubit). Gegeven een rekenprobleem kunnen we dan het aantal elementaire instructies vergelijken die de best bekende quantum- en klassieke algoritmen gebruiken om het op te lossen. Op deze manier kunnen we een duidelijk beeld krijgen van het voordeel van toekomstige quantumcomputers.

5.1 Spreken met orakels

In deze quest zullen we verschillende quantumalgoritmes bekijken en zien hoe ze zich verhouden tot klassieke algoritmes die hetzelfde probleem oplossen. Omdat het over het algemeen erg moeilijk is om te begrijpen hoeveel elementaire bewerkingen nodig zijn om een bepaalde berekening uit te voeren, zullen we in deze quest kijken naar een simpelere maat voor algoritmecomplexiteit (dit is ook wat computerwetenschappers doen als ze hun eigen werk wat simpeler willen maken).

Wanneer een computer een programma uitvoert, moet de computer verschillende soorten bewerkingen uitvoeren. De meest langzame bewerkingen zijn die waarbij toegang tot data nodig is. Denk bijvoorbeeld aan het lezen van gegevens uit het geheugen, de harde schijf of - in het ergste geval - een andere computer die bereikbaar is via het internet. Als de relevante data eenmaal gelezen is, kan de verwerking hiervan relatief snel gaan. Daarom kunnen we een ruw idee krijgen van hoe lang een bepaald algoritme zal duren door alleen de instructies mee te tellen die toegang nodig hebben tot je gegevens.

Je kent dit misschien wel bij het openen van een ingewikkelde webpagina of een groot document. Het kan een tijdje duren voordat deze geladen is, maar als het eenmaal geladen is, werkt het scrollen door het document en het invoegen van een nieuwe regel meestal vrij snel.

Een andere manier om hierover na te denken, die we in deze quest zullen gebruiken, is dat de informatie die je probeert te openen in feite wordt verkregen door een ander algoritme of een subroutine binnen je algoritme. Bovendien is deze subroutine vaak erg traag. Deze subroutine leest de informatie bijvoorbeeld van de harde schijf of haalt het op via het internet. Of misschien is het antwoord niet eens direct beschikbaar en moet het zelf geproduceerd worden door een ingewikkelde berekening uit te voeren. Hoe dan ook, deze subroutine doet er altijd erg lang over om het antwoord te geven, dus je wilt hem zo min mogelijk aanroepen. We noemen zo'n subroutine een orakel, omdat het erg slim is en alle antwoorden weet, maar ook een beetje traag is en dus een tijdje moet nadenken om het antwoord te geven.

Formeler gezegd is een klassiek orakel een functie $f\colon \{0,1\}^n \to \{0,1\}$, waarbij $\{0,1\}^n$ staat voor de verzameling van alle n-bit binaire strings.[13] Je kunt de invoer $x \in \{0,1\}^n$ van deze functie zien als een vraag en de uitvoer, de bit $f(x)$, als het antwoord. Elke keer dat je de functie f uitvoert op een bepaalde invoer, stel je een vraag die hoort bij die specifieke invoer en krijg je een ja/nee antwoord dat hoort bij de functiewaarde.

Je kunt bijvoorbeeld de toegang tot een harde schijf of geheugen modelleren met een orakel. Stel dat je 4 bits aan geheugen hebt, dan kun je deze modelleren als een functie $f : \{0,1\}^2 \to \{0,1\}$ die, gegeven een binair adres, de bijbehorende bit teruggeeft. Als je bijvoorbeeld alle vier de bits wilt weten, moet je f vier keer berekenen om de vier waardes $f(00), f(01), f(10), f(11)$ te krijgen.

Wat belangrijk is om op te merken, is dat in onze situatie de berekeningen die we willen oplossen niet gaan over het vinden van de waarde van f bij een specifieke invoer. Zulke problemen zijn triviaal omdat ze opgelost kunnen worden door het orakel maar één keer te raadplegen, want dit is precies wat het orakel doet – het vertelt je de waarde van de functie op elke gewenste invoer. Het soort problemen waarin we geïnteresseerd zijn, zijn wat subtieler. We willen een bepaalde eigenschap van de functie f bepalen door deze zo weinig mogelijk aan te roepen.

Stel bijvoorbeeld dat we willen weten of $f(x) = 0$ voor alle $x \in \{0,1\}^n$. In dit geval kunnen we het orakel vragen naar willekeurige waarden van x totdat we er een vinden waarbij $f(x) = 1$. We zullen zo andere voorbeelden van zulke problemen zien.

5.1.1 Omkeerbare berekeningen

Voordat we ons laten meeslepen in het vinden van spannende nieuwe quantumalgoritmes, moeten we er eerst zeker van zijn dat we op een quantumcomputer nog steeds alles kunnen berekenen wat we op een gewone computer ook kunnen. Met andere woorden, laten we er eerst voor zorgen dat quantumcomputers eigenlijk niet *minder* krachtig zijn dan gewone (klassieke) computers! Dit is niet helemaal voor de hand liggend, omdat de manier waarop quantumcomputers werken totaal anders is. Zo zijn alle bewerkingen op een quantumcomputer omkeerbaar of inverteerbaar, zoals we al eerder hebben gezien in §2.4.2 en §4.1.3, maar dit is normaal gesproken niet het geval op een gewone computer. Wie heeft er nog nooit per ongeluk een bestand gewist of vergeten de wijzigingen in hun document op te slaan en zo al hun werk verloren? Als alle bewerkingen omkeerbaar zouden zijn, zou je je nooit zorgen hoeven te maken over zulke onbenullige zaken.

Dit roept de volgende vraag op: hoe kunnen we er zeker van zijn dat quantumcomputers alles kunnen berekenen wat gewone computers kunnen berekenen? Een manier om dit aan te pakken is door te laten zien dat omkeerbaarheid in feite geen beperking is voor wat een gewone computer kan berekenen, en dus ook geen beperking voor quantumcomputers. Met andere woorden, we zullen laten zien dat elke berekening omkeerbaar gemaakt kan worden en dus uitgevoerd kan worden op een quantumcomputer.

Om een idee te krijgen van hoe dit gedaan kan worden, nemen we het eenvoudige voorbeeld van het berekenen van de logische AND-functie op twee bits. Als beide bits 1 zijn, geeft AND een waarde van 1, anders een waarde van 0. We kunnen dus de AND-functie weergeven met de volgende waarheidstabel:

x_1	x_2	$\mathrm{AND}(x_1, x_2)$
0	0	0
0	1	0
1	0	0
1	1	1

(5.1)

[13] Om een voorbeeld te geven: $\{0,1\}^2$ is de verzameling van bitstrings van lengte 2, oftewel: $\{00, 01, 10, 11\}$. En de klassieke NOT-operatie werkt als NOT: $\{0,1\}^1 \to \{0,1\}$, omdat het een enkele bit als invoer verwacht, en een enkele bit als uitvoer.

Als we de notatie uit §3.1 gebruiken, kunnen we dit wiskundig opschrijven als de volgende bewerking op twee bits:

$$[x_1, x_2] \mapsto [\text{AND}(x_1, x_2)].$$

Het is duidelijk dat deze bewerking niet omkeerbaar is, omdat ze niet hetzelfde aantal uitvoerbits als invoerbits heeft. Als je alleen weet dat de AND van twee bits 0 is, kun je de precieze toestand van de twee bits niet afleiden – zoals de waarheidstabel laat zien, zijn er drie mogelijke opties.

Hoe kunnen we dit probleem oplossen? Laten we het eerste bit behouden en een tweede uitgangsbit toevoegen waarin we het antwoord opslaan:

$$[x_1, x_2] \mapsto [x_1, \text{AND}(x_1, x_2)].$$

Dit is beter, omdat we nu twee bits op twee bits afbeelden. Maar is het omkeerbaar? Dat wil zeggen, als we de uitvoer krijgen, kunnen we dan altijd de invoer terughalen? Nou, we kunnen zeker het eerste bit van de invoer, x_1, terugrekenen, omdat we deze ook in de uitvoer hebben zitten. Hoe zit het dan met x_2? Als $x_1 = 0$ dan is volgens Vgl. (5.1) de uitvoer $[00]$ – ongeacht de waarde van x_2. Dit betekent dat we ook hier de invoer niet altijd kunnen herleiden en dat deze aanpak dus helaas ook niet werkt.

Dit begint er hopeloos uit te zien. Is het überhaupt mogelijk om de AND-functie op omkeerbare manier uit te voeren? Als je vastloopt, moet je out of the box denken! Wie heeft in dit geval gezegd dat we ons moeten beperken tot twee bits? Als we beide invoerbits behouden en een derde bit introduceren om het antwoord op te slaan, zou dit de bewerking zeker omkeerbaar moeten maken:

$$[x_1, x_2, 0] \mapsto [x_1, x_2, \text{AND}(x_1, x_2)]. \tag{5.2}$$

Nu is er geen probleem om de bewerking te inverteren – gegeven een willekeurige uitvoerbitstring, kunnen we gewoon vergeten wat het laatste bit is en het vervangen door $[0]$ om de invoerbitstring terug te krijgen. Als de uitvoer bijvoorbeeld $[111]$ is, dan moet de invoer $[110]$ zijn geweest.

Zijn we nu dus klaar? Niet zo snel! Merk op dat Vgl. (5.2) alleen de bewerking *gedeeltelijk* bepaalt – als de invoer $[111]$ is, of een andere bitstring die eindigt op 1, dan zegt Vgl. (5.2) niet hoe de bewerking op deze invoer moet werken. Aangezien de vier invoerstrings die eindigen op 0 worden afgebeeld op vier verschillende uitvoerstrings, is het duidelijk dat we Vgl. (5.2) op een arbitraire manier kunnen uitbreiden naar een omkeerbare operatie van drie bits. Maar is er een systematische manier om dit te doen?

Merk hiervoor op dat Vgl. (5.2) het laatste bit omdraait van $[0]$ naar $[1]$ als $\text{AND}(x_1, x_2) = 1$. We kunnen dus ook, als het laatste bit toevallig $[1]$ is, onze bewerking zo definiëren dat het de bit terug flipt naar $[0]$ als $\text{AND}(x_1, x_2) = 1$. Met andere woorden, we kunnen Vgl. (5.2) als volgt uitbreiden naar *alle* mogelijke invoerstrings:

$$[x_1, x_2, y] \mapsto [x_1, x_2, y \oplus \text{AND}(x_1, x_2)], \tag{5.3}$$

voor alle $x_1, x_2, y \in \{0, 1\}$, waarbij '$\oplus$' staat voor optelling modulo 2.

De bewerking in Vgl. (5.3) is nu gedefinieerd op alle mogelijke invoerbits. Maar is het nu eindelijk omkeerbaar? Ja, dat is het! Sterker nog, deze bewerking is zijn eigen inverse! Dat wil zeggen, als we de bewerking twee keer uitvoeren, krijgen we de oorspronkelijke invoer terug:

$$\begin{aligned}[x_1, x_2, y] &\mapsto [x_1, x_2, y \oplus \text{AND}(x_1, x_2)] \\ &\mapsto [x_1, x_2, y \oplus \text{AND}(x_1, x_2) \oplus \text{AND}(x_1, x_2)] = [x_1, x_2, y].\end{aligned}$$

In de derde stap hebben we gebruikt dat $a \oplus a = 0$ voor elke $a \in \{0, 1\}$, zie Vgl. (3.20). We hebben dus een succesvolle manier gevonden om de AND-functie op een omkeerbare manier te berekenen.

5.1.2 Bit-orakels

Dit idee werkt niet alleen voor de logische AND-functie maar in feite voor elke functie

$$f\colon \{0,1\}^n \to \{0,1\}$$

die n bits als invoer neemt en een enkele bit teruggeeft. Voor zo'n functie f kunnen we een omkeerbare bewerking op $(n+1)$ bits definiëren:

$$[x_1,\ldots,x_n,y] \mapsto [x_1,\ldots,x_n,y\oplus f(x_1,\ldots,x_n)] \tag{5.4}$$

voor alle $x_1,\ldots,x_n,y\in\{0,1\}$. Net als Vgl. (5.3) is deze bewerking zijn eigen inverse, dat wil zeggen dat de bewerking twee keer toepassen gelijk is aan niets doen.

Het is geruststellend dat we elke functie $f:\{0,1\}^n\to\{0,1\}$ op een omkeerbare manier kunnen implementeren zoals in Vgl. (5.4). Ten eerste betekent dit dat elke berekening op een klassieke computer op zo'n manier kan worden uitgevoerd dat we de oorspronkelijke invoer terug kunnen krijgen. Ten tweede, als de berekening omkeerbaar is gemaakt, kunnen we deze ook uitvoeren op een quantumcomputer. Dit betekent dat quantumcomputers alles kunnen berekenen wat klassieke computers ook kunnen! Ten derde betekent dit dat we elke functie f kunnen implementeren als een orakel op een quantumcomputer.

Laten we eens kijken hoe dit werkt. De quantumversie van de bewerking in Vgl. (5.4) is als volgt gedefinieerd:

$$U_f\,|x_1,\ldots,x_n,y\rangle = |x_1,\ldots,x_n,y\oplus f(x_1,\ldots,x_n)\rangle \tag{5.5}$$

voor alle $x_1,\ldots,x_n,y\in\{0,1\}$. Vgl. (5.5) definieert hoe U_f werkt op alle basistoestanden van $n+1$ qubits. Zoals gebruikelijk breiden we deze definitie uit naar een arbitraire toestand van $n+1$ qubits door lineariteit. Omdat U_f gewoon de basistoestanden permuteert, kun je nagaan zoals in Oefenopgave 4.4 dat dit een geldige quantumbewerking is.

We noemen de quantumbewerking U_f, zoals gedefinieerd in Vgl. (5.5), het bit-orakel voor f (we zouden de functie in Vgl. (5.4) ook een klassiek bit-orakel kunnen noemen, maar deze naam hebben we niet nodig). De term 'orakel' betekent dat het toepassen van deze bewerking hetzelfde is als het vragen aan een almachtig orakel om ons de waarde van de functie op een gegeven invoer te vertellen. We weten niet hoe het orakel precies geïmplementeerd is of waar het zijn antwoord vandaan haalt, we zullen gewoon tellen hoeveel vragen we het orakel moeten stellen om een bepaalde eigenschap van de functie f te leren. Veel interessante algoritmische problemen kunnen op deze manier gemodelleerd worden, zoals we in het vervolg van deze quest zullen zien.

Het concept van een orakel lijkt een beetje op het spelletje 'raad mijn getal'. Je vriend (het orakel) bedenkt een getal en jij stelt hem vragen van de vorm "is je getal x"? Je vriend beantwoordt elk van deze vragen met "ja" of "nee". Met andere woorden, je vriend verbergt een functie die op alle inputs op "nee" uitkomt, behalve op één (het getal dat hij in gedachten heeft). Met elke vraag die je stelt, krijg je meer informatie over welk getal het zou kunnen zijn. Je kunt je afvragen hoeveel vragen je moet stellen om het getal te bepalen. Bovendien, wat als je je vragen kon stellen aan een quantumorakel zoals U_f in plaats van aan je vriend? Zou je het antwoord kunnen achterhalen met minder vragen? In de quest van deze week zullen we een aantal interessante voorbeelden zien waarbij dit inderdaad het geval is!

Laten we enkele voorbeelden van bit-orakels bespreken. Stel dat f de AND-functie is. Volgens Vgl. (5.1), kunnen we AND interpreteren als vermenigvuldiging modulo 2, aangezien $AND(x_1,x_2)=x_1x_2$. Het bijbehorende bit-orakel

$$U_{\text{AND}}\,|a,b,c\rangle = |a,b,c\oplus ab\rangle$$

is niets anders dan de Toffoli-gate uit Oefenopgave 4.4. Zelfs voor $n = 1$ en de functie $f(x) = x$, die simpelweg zijn invoer teruggeeft, krijgen we een interessant resultaat: voor alle $a, b \in \{0,1\}$ geldt dan

$$U_f \left|a, b\right\rangle = \left|a, b \oplus a\right\rangle .$$

Dit is gewoon de bekende controlled-NOT-gate $CNOT_{1\to 2}$ van Vgl. (3.47) uit §3.2.4! De constructie van het bit-orakel geeft dus verschillende interessante quantumbewerkingen terug die we eerder met de hand gedefinieerd hadden. In de volgende opgave ga je proberen alle andere bit-orakels te implementeren voor functies van één bit.

Oefenopgave 5.1: Bit-orakel voor een één-bitfunctie

Stel dat $f : \{0,1\} \to \{0,1\}$ een functie is met één invoer- en uitvoerbit. Zo'n functie wordt volledig bepaald door de waarden $f(0), f(1) \in \{0,1\}$. Dit zijn twee bits, dus er zijn precies vier van zulke functies. We hebben zojuist besproken hoe je het bit-orakel U_f kunt implementeren voor de functie $f(x) = x$. Kun je het bit-orakel U_f voor de andere drie functies in QUIRKY implementeren?

5.1.3 Fase-orakels

Omdat het bit-orakel U_f een quantumbewerking is, kunnen we het niet alleen toepassen op basistoestanden $\left|x_1, \ldots, x_n, y\right\rangle$ maar ook op algemene quantumtoestanden. Waarom zouden we zoiets willen doen? Wel, als we het orakel alleen 'klassieke' vragen stellen dan is er weinig hoop op het bereiken van een quantumversnelling! Laten we, gegeven deze motivering, eens onderzoeken hoe het bit-orakel U_f voor een arbitraire functie $f : \{0,1\}^n \to \{0,1\}$ zich gedraagt als het laatste register wordt gezet op $\left|-\right\rangle = (\left|0\right\rangle - \left|1\right\rangle)/\sqrt{2}$. We merken eerst het volgende interessante gegeven op:

$$\text{NOT}\left|-\right\rangle = \frac{1}{\sqrt{2}}(\text{NOT}\left|0\right\rangle - \text{NOT}\left|1\right\rangle) = \frac{1}{\sqrt{2}}(\left|1\right\rangle - \left|0\right\rangle) = -\left|-\right\rangle .$$

Dat wil zeggen, als we een qubit in de toestand $\left|-\right\rangle$ inverteren, dan pikken we een minteken op. We kunnen op dezelfde manier berekenen hoe het bit-orakel werkt op een toestand van de vorm $\left|x_1, \ldots, x_n\right\rangle \otimes \left|-\right\rangle$. Door lineariteit volgt uit Vgl. (5.5) dat

$$\begin{aligned}
&U_f\left(\left|x_1, \ldots, x_n\right\rangle \otimes \left|-\right\rangle\right) \\
&= U_f\Big(\frac{1}{\sqrt{2}}\left|x_1, \ldots, x_n, 0\right\rangle - \frac{1}{\sqrt{2}}\left|x_1, \ldots, x_n, 1\right\rangle\Big) \\
&= \frac{1}{\sqrt{2}}\left|x_1, \ldots, x_n, f(x_1, \ldots, x_n)\right\rangle - \frac{1}{\sqrt{2}}\left|x_1, \ldots, x_n, f(x_1, \ldots, x_n) \oplus 1\right\rangle \\
&= \left|x_1, \ldots, x_n\right\rangle \otimes \frac{1}{\sqrt{2}}\left(\left|f(x_1, \ldots, x_n)\right\rangle - \left|f(x_1, \ldots, x_n) \oplus 1\right\rangle\right) \\
&= (-1)^{f(x_1, \ldots, x_n)}\left|x_1, \ldots, x_n\right\rangle \otimes \left|-\right\rangle .
\end{aligned}$$

Met andere woorden, de laatste qubit eindigt weer in de toestand $\left|-\right\rangle$, maar we krijgen een minteken als $f(x_1, \ldots, x_n) = 1$. In feite hebben we de volgende quantumbewerking uitgevoerd op de eerste n qubits:

$$O_f \left|x_1, \ldots, x_n\right\rangle = (-1)^{f(x_1, \ldots, x_n)}\left|x_1, \ldots, x_n\right\rangle , \tag{5.6}$$

voor alle bitstrings $x_1, \ldots, x_n \in \{0,1\}$. We noemen O_f het fase-orakel voor f.

Opmerkelijk genoeg werkt het fase-orakel O_f voor een functie $f\colon \{0,1\}^n \to \{0,1\}$ op n qubits, omdat het de uitvoer opslaat in de amplitude. Dit is anders dan het bit-orakel U_f, dat de uitvoer in een extra qubit opslaat en daarom op $n+1$ qubits werkt.

Op het eerste gezicht lijkt het fase-orakel niet veel te doen, omdat het alleen een algemeen minteken toevoegt als het werkt op een basistoestand, waarvan we door Oefenopgave 2.7 weten dat het niet kan worden gemeten. Maar als we het toepassen op een superpositie kan het fase-orakel relatieve mintekens veroorzaken, waardoor we een interessant resultaat krijgen. Als we bijvoorbeeld voor $n = 2$ qubits een fase-orakel O_f toepassen op een algemene twee-qubit-toestand

$$|\psi\rangle = \psi_{00}\,|00\rangle + \psi_{01}\,|01\rangle + \psi_{10}\,|10\rangle + \psi_{11}\,|11\rangle$$

dan krijgen we

$$O_f\,|\psi\rangle = (-1)^{f(0,0)}\psi_{00}\,|00\rangle + (-1)^{f(0,1)}\psi_{01}\,|01\rangle + (-1)^{f(1,0)}\psi_{10}\,|10\rangle + (-1)^{f(1,1)}\psi_{11}\,|11\rangle$$

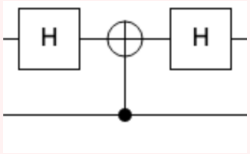
Het blijkt dat het fase-orakel O_f inderdaad handig is en vaak veel gemakkelijker toe te passen is in quantumalgoritmes dan het bit-orakel U_f, dus vanaf nu zullen we het bit-orakel niet meer gebruiken.

Oefenopgave 5.2: Fase-orakel voor een één-bitfunctie

We weten van Oefenopgave 5.1 dat er vier functies $f : \{0,1\} \to \{0,1\}$ met een enkele invoer- en uitvoerbit. Kun je het fase-orakel O_f voor elk van deze functies implementeren in QUIRKY?

Huiswerkopdracht 5.1: Bepaal de functie vanuit het fase-orakel

Beschouw het volgende twee-qubitcircuit (zoals gewoonlijk is de onderste draad de eerste qubit):



Voor welke functie $f\colon \{0,1\}^2 \to \{0,1\}$ is dit het fase-orakel?

Hint: Gebruik dat H NOT $H = Z$, wat volgt uit Oefenopgave 4.5.

Laten we kort samenvatten wat we tot nu toe hebben bereikt: Met behulp van bit-orakels kunnen we een quantumcomputer vragen om een functie $f\colon \{0,1\}^n \to \{0,1\}$ te evalueren op precies dezelfde manier als we dat zouden vragen aan een gewone computer die op een inverteerbare manier werkt (vergelijk Vgl. (5.4) en (5.5)). Dit is belangrijk, want het betekent dat we geen appels met peren vergelijken als we vragen hoeveel keer we het bit orakel U_f moeten vragen om iets te weten te komen over f versus hoeveel keer we f moeten evalueren op een gewone computer om hetzelfde te weten te komen. En omdat we net hebben laten zien dat het fase-orakel O_f altijd geïmplementeerd kan worden met behulp van het bit orakel U_f, maakt het geen verschil als we in plaats daarvan vragen stellen aan het fase-orakel O_f.

5.2 Quantumalgoritmes

In dit hoofdstuk zullen we verschillende quantumalgoritmes zien die een rekenprobleem veel sneller kunnen oplossen dan welk klassiek algoritme dan ook. Zulke versnellingen zijn erg verrassend omdat quantummechanica op het eerste gezicht niets met berekeningen te maken lijkt te hebben. Toch kunnen quantummechanische verschijnselen zoals interferentie zeer indrukwekkende rekenversnellingen mogelijk maken. Zoals al eerder is uitgelegd, werken we in een rekenomgeving waarin we alleen het aantal vragen tellen. Dat wil zeggen, als je de mogelijkheid hebt om een functie f te berekenen op elke invoer, hoe vaak moet je de functie dan uitvoeren om een bepaalde eigenschap van f te bepalen. Met andere woorden, als je toegang hebt tot een orakel dat f kan evalueren, hoe vaak moet je dit orakel dan gebruiken om een bepaalde eigenschap van f te bepalen.

5.2.1 Het algoritme van Deutsch

Het is laat op zondagavond. Alice en Bob zijn net klaar met hun huiswerkopdrachten voor hun vak quantumcomputing en staan op het punt om een 3D-film te gaan kijken. Wanneer ze hun holografische televisie aanzetten, ontdekken ze dat de filmvoorstelling is uitgesteld vanwege onverwachte dramatische nieuwsberichten van het International Transgalactic Station. Er is een verschrikkelijk ongeluk gebeurd en een module met twee bemanningsleden, Hila en Iman, is losgekomen van het moederschip. Het laatste bericht dat we van de module ontvingen was dat Iman gewond is en hevig bloedt – hij heeft dringend bloed nodig. Wat de situatie ingewikkeld maakt, is dat Hila en Iman enigszins in shock zijn en hun eigen bloedgroep vergeten zijn – ze kunnen zich alleen herinneren dat ze allebei bloedgroep A of B hebben. De nieuwspresentator doet een beroep op het publiek om te helpen en een manier te bedenken waarop Hila en Iman kunnen bepalen of ze dezelfde bloedgroep hebben, want als dat het geval is, kan Hila haar bloed aan Iman geven om zijn leven te redden. De medische kit in de module van Hila en Iman bevat namelijk een lymfo-transcoder die een van de twee bloedgroepen kan omzetten in de tegenovergestelde. Dus zelfs als blijkt dat hun bloedgroepen niet bij elkaar passen, kan Hila die van haar met de lymfotranscoder omzetten in de tegenovergestelde bloedgroep.

Bij het horen van dit nieuws laten Alice en Bob hun plan om een film te gaan kijken meteen varen en gaan ze nadenken over wat er gedaan kan worden om Hila en Iman te helpen. Het nieuws gaat verder met wat extra informatie. Gelukkig blijkt dat de module die betrokken was bij het ongeluk een databasechip bevat die de bloedgroep van Hila en Iman opslaat. We kunnen dit modelleren met een functie $f\colon \{0,1\} \to \{0,1\}$, waarbij

$$f(0) = \begin{cases} 0 & \text{als Hila bloedgroep A heeft,} \\ 1 & \text{als Hila bloedgroep B heeft.} \end{cases}$$

en

$$f(1) = \begin{cases} 0 & \text{als Iman bloedgroep A heeft,} \\ 1 & \text{als Iman bloedgroep B heeft.} \end{cases}$$

Wat bepaald moet worden is of $f(0) = f(1)$ of niet!

De oplossing lijkt simpel: Hila en Iman hoeven alleen maar twee keer de database te bevragen om hun bloedgroepen, $f(0)$ en $f(1)$, uit te lezen en dan de twee waarden te vergelijken om te zien of ze hetzelfde zijn. Helaas heeft het ongeluk de besturingslogica van de databasechip gedeeltelijk doorgebrand en de nieuwslezer meldt dat de databasechip na de eerste zoekopdracht waarschijnlijk volledig zal doorbranden.

Onze twee hoofdrolspelers zijn vastgelopen. Het is duidelijk dat elk klassiek algoritme f twee keer moet evalueren om te bepalen of $f(0) = f(1)$. Als je alleen de waarde van $f(0)$ kent, hangt het nog steeds af van de waarde van $f(1)$ of $f(0) = f(1)$ en, tenzij je ook $f(1)$ berekent, zou je niet kunnen zeggen of $f(0) = f(1)$. Op dezelfde manier, als je alleen $f(1)$ kent, kun je het niet vergelijken met $f(0)$ tenzij je ook weet wat de waarde van $f(0)$ is. Welke strategie je ook gebruikt, je moet zowel $f(0)$ als $f(1)$ kennen om te bepalen of $f(0) = f(1)$. Is hier echt geen ontkomen aan?

Na het doorbladeren van wat handleidingen ontdekt Bob dat de databasechip in quantummode kan worden gezet. Wanneer dit is ingeschakeld, evalueert de databasechip de functie niet meer op de klassieke manier, maar maakt in plaats daarvan gebruik van het fase-orakel O_f. Zou dit op de een of andere manier gebruikt kunnen worden om het probleem op te lossen? Alice denkt er even over na en realiseert zich plotseling dat dit precies is waar het algoritme van Deutsch voor bedoeld is! Alice en Bob nemen snel wat berekeningen door om te bevestigen dat het werkt en gaan zitten om een intergalactische e-mail te schrijven met instructies aan Hila en Iman over hoe ze het probleem kunnen oplossen. Hun instructies zijn als volgt:

1. Maak een qubit aan in de toestand $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$.
2. Gebruik de databasechip in quantummode om de bewerking O_f toe te passen.
3. Pas de Hadamard-gate H toe op de uitvoerqubit en meet deze.
4. Als de uitkomst 0 is dan hebben Hila en Iman dezelfde bloedgroep en anders hebben ze verschillende bloedgroepen.

Merk op dat Hila en Iman in deze procedure de databasechip maar één keer hoeven te bevragen om te bepalen of ze dezelfde bloedgroep hebben. Hier is een implementatie van het algoritme in QUIRKY:

De afbeelding laat zien dat de uitkomst 1 is, dus Hila en Iman hebben verschillende bloedgroepen.

Maar waarom werkt het algoritme van Deutsch? Laten we het stap voor stap analyseren. De eerste Hadamard-gate maakt de toestand $|+\rangle = H|0\rangle$. Vervolgens passen we het fase-orakel O_f toe, wat leidt tot de toestand

$$\begin{aligned} O_f|+\rangle &= \frac{1}{\sqrt{2}} O_f|0\rangle + \frac{1}{\sqrt{2}} O_f|1\rangle \\ &= \frac{1}{\sqrt{2}}(-1)^{f(0)}|0\rangle + \frac{1}{\sqrt{2}}(-1)^{f(1)}|1\rangle . \end{aligned}$$

Na het toepassen van de tweede Hadamard krijgen we de volgende toestand:

$$\begin{aligned} HO_f|+\rangle &= \frac{1}{\sqrt{2}}(-1)^{f(0)} H|0\rangle + \frac{1}{\sqrt{2}}(-1)^{f(1)} H|1\rangle \\ &= \frac{1}{\sqrt{2}}(-1)^{f(0)}|+\rangle + \frac{1}{\sqrt{2}}(-1)^{f(1)}|-\rangle \\ &= \frac{1}{2}(-1)^{f(0)}(|0\rangle + |1\rangle) + \frac{1}{2}(-1)^{f(1)}(|0\rangle - |1\rangle) \\ &= \frac{(-1)^{f(0)} + (-1)^{f(1)}}{2}|0\rangle + \frac{(-1)^{f(0)} - (-1)^{f(1)}}{2}|1\rangle . \end{aligned} \tag{5.7}$$

Merk op dat de twee tekens $(-1)^{f(0)}$ en $(-1)^{f(1)}$ worden opgeteld in de eerste amplitude, maar afgetrokken in de tweede amplitude. Afhankelijk van de waardes van $f(0)$ en $f(1)$ zullen we voor elke amplitude of constructieve of destructieve interferentie waarnemen (zie §2.6.1). In feite wordt welke van de twee amplitudes overblijft alleen bepaald door of $f(0)$ en $f(1)$ gelijk zijn of niet:

$$\begin{aligned} f(0) = f(1): &\qquad HO_f|+\rangle = \pm|0\rangle , \\ f(0) \neq f(1): &\qquad HO_f|+\rangle = \pm|1\rangle . \end{aligned} \tag{5.8}$$

Het is een goede oefening om dit expliciet na te gaan:

Oefenopgave 5.3: Het algoritme van Deutsch nagaan

We weten nog van Oefenopgave 5.1 dat er vier functies $f : \{0,1\} \to \{0,1\}$ zijn. Bereken voor elke functie de toestand $HO_f|+\rangle$ met behulp van Vgl. (5.7).

Vgl. (5.8) laat zien dat de eindmeting uitkomst 0 oplevert als en alleen als $f(0) = f(1)$. Het algoritme van Deutsch bepaalt dus correct of $f(0) = f(1)$. Belangrijk is dat het de functie $f : \{0,1\} \to \{0,1\}$ slechts eenmaal uitvoert door het fase-orakel te gebruiken. Daarentegen hebben we hierboven besproken dat elk klassiek algoritme noodzakelijkerwijs beide functiewaarden $f(0)$ en $f(1)$ afzonderlijk moet evalueren.

Een andere manier om het algoritme van Deutsch te interpreteren is dat het de som van de twee bits $f(0)$ en $f(1)$ modulo twee berekent. Dit komt omdat $f(0) \oplus f(1) = 0$ als en slechts als $f(0) = f(1)$. Uit Vgl. (3.20) blijkt namelijk dat optellen modulo twee als volgt werkt:

x_1	x_2	$x_1 \oplus x_2$
0	0	0
0	1	1
1	0	1
1	1	0

(5.9)

Dit is ook de reden waarom de som modulo twee bekend staat als de *XOR* (afkorting voor 'exclusive OR') van de twee bits, omdat het één is als precies één van de twee bits is.

5.2.2 De Hadamard-transformatie en interferentie

Al is het algoritme van Deutsch erg verbazingwekkend, toch levert het maar een kleine verbetering op ten opzichte van het beste klassieke algoritme, namelijk 1 evaluatie van f in plaats van 2. Dit kan alsnog handig zijn als het evalueren van f heel lang duurt, bijvoorbeeld een jaar. Maar als het maar om een milliseconde gaat, dan zouden de meeste mensen het niet erg vinden om twee milliseconden te wachten om het antwoord te krijgen (en er veel minder voor te betalen omdat ze dan geen quantumcomputer hoeven te gebruiken)! Om het nut van quantumcomputers aan te tonen, willen we berekeningen met meer dan een factor 2 versnellen.

We gaan nooit een grotere versnelling bereiken als we alleen kijken naar functies van één bit, omdat die volledig bepaald kunnen worden door twee evaluaties: $f(0)$ en $f(1)$. We zullen daarom wat algemener kijken naar functies $f\colon \{0,1\}^n \to \{0,1\}$ met n invoerbits, omdat er veel meer van zulke functies zijn (er zijn er namelijk 2^{2^n}). Onthoud dat ons doel niet is om een functie te evalueren (dat kunnen we inderdaad doen met een enkele vraag aan het orakel), maar om een interessante eigenschap van de functie te leren die betrekking heeft op de waardes die de functie aanneemt bij verschillende invoerbits. Zijn er eigenschappen van n-bit-functies die we heel efficiënt kunnen bepalen met slimme quantumalgoritmes?

Om het algoritme van Deutsch te generaliseren, moet je bedenken dat het belangrijkste onderdeel was om een Hadamard-gate H te introduceren voor en na het fase-orakel. We kunnen iets soortgelijks doen als we een n-bit-functie hebben. In dit geval is het fase-orakel O_f een quantumbewerking van n qubits, dus we kunnen gewoon Hadamard-gates toepassen op alle qubits voor en na het orakel. De quantumbewerking die Hadamards in parallel toepast op elk van de n qubits wordt de Hadamard-transformatie genoemd. We weten van §3.2.3 dat we deze als volgt kunnen schrijven:

$$H \otimes \cdots \otimes H.$$

Laten we eerst eens kijken wat er gebeurt als we de Hadamard-transformatie toepassen op een basistoestand. Voor $|0\ldots0\rangle$ is het resultaat simpelweg de uniforme superpositie over alle basistoestanden:

$$\begin{aligned}(H \otimes \cdots \otimes H)\,|0\ldots0\rangle &= H\,|0\rangle \otimes \cdots \otimes H\,|0\rangle \\ &= \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) \otimes \cdots \otimes \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) \\ &= \frac{1}{\sqrt{2^n}}\left(|0\ldots00\rangle + |0\ldots01\rangle + \ldots + |1\ldots11\rangle\right).\end{aligned}$$

Deze toestand is een superpositie van 2^n basistoestanden. Er is een compactere manier om dit op te schrijven:

$$(H \otimes \cdots \otimes H)\,|0\ldots0\rangle = \frac{1}{\sqrt{2^n}} \sum_{y_1,\ldots,y_n \in \{0,1\}} |y_1,\ldots,y_n\rangle \tag{5.10}$$

De notatie $\sum_{y_1,\ldots,y_n\in\{0,1\}}$ betekent dat je de som van $|y_1,\ldots,y_n\rangle$ berekent voor alle mogelijke combinaties van de bits $y_1,\ldots,y_n$.

De Hadamard-transformatie topassen op een basistoestand leidt altijd tot een superpositie van alle 2^n basistoestanden. Bovendien zijn alle amplitudes gelijk, op eventuele mintekens na. Het uitzoeken waar die mintekens precies voorkomen is een tikje lastig. Probeer het als opwarmertje eerst voor de gevallen $n = 1$ en $n = 2$.

Oefenopgave 5.4: Twee Hadamards

Uit Vgl. (2.20) volgt dat $H|x_1\rangle = (|0\rangle + (-1)^{x_1}|1\rangle)/\sqrt{2}$, voor elke $x_1 \in \{0,1\}$.

1. Schrijf de toestand $H|x_1\rangle$, voor arbitraire $x_1 \in \{0,1\}$, om in de volgende vorm:

$$H|x_1\rangle = \frac{1}{\sqrt{2}} \sum_{y_1 \in \{0,1\}} (-1)^{\boxed{???}} |y_1\rangle ,$$

   waarbij $\boxed{???}$ een uitdrukking is die bestaat uit $x_1, y_1 \in \{0,1\}$. Bepaal deze uitdrukking.

2. Schrijf de toestand $(H \otimes H)|x_1, x_2\rangle$, voor willekeurige $x_1, x_2 \in \{0,1\}$, in de vorm

$$(H \otimes H)|x_1, x_2\rangle = \frac{1}{2} \sum_{y_1, y_2 \in \{0,1\}} (-1)^{\boxed{???}} |y_1, y_2\rangle ,$$

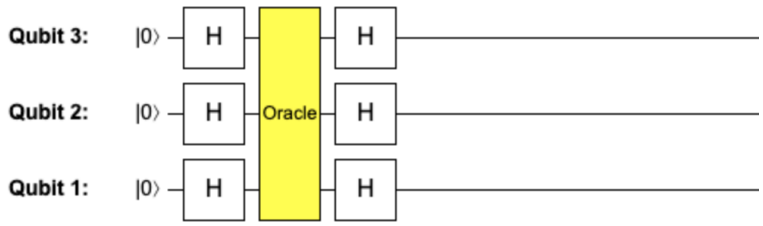
   waarbij $\boxed{???}$ staat voor een uitdrukking van $x_1, x_2, y_1, y_2 \in \{0,1\}$.
   Kan je bepalen wat deze uitdrukking is?

Heb je de oplossing gevonden? Goed zo, dan mag je nu verder lezen!

In het algemeen kan je de volgende formule afleiden die het mintekenpatroon beschrijft van de amplitudes die je krijgt als je de Hadamard-transformatie toepast op een arbitraire n-qubit-basistoestand: Voor elke $x_1, \ldots, x_n \in \{0,1\}$ geldt

$$(H \otimes \cdots \otimes H) \left|x_1, \ldots, x_n\right\rangle = \frac{1}{\sqrt{2^n}} \sum_{y_1,\ldots,y_n \in \{0,1\}} (-1)^{x_1 y_1 + \ldots + x_n y_n} \left|y_1, \ldots, y_n\right\rangle . \tag{5.11}$$

We kunnen nu het algoritme van Deutsch op een eenvoudige manier generaliseren. Beginnend met de n-qubit-basistoestand $|0 \ldots 0\rangle$, passen we eerst de Hadamard-transformatie toe, vervolgens het fase-orakel O_f voor de functie $f\colon \{0,1\}^n \to \{0,1\}$ die we willen onderzoeken, en tenslotte nog een Hadamard-transformatie. Bijvoorbeeld, voor $n = 3$ komt dit overeen met het volgende QUIRKY-circuit:



Voor een algemene n is de wiskundige formule voor de uiteindelijke n-qubit-toestand:

$$(H \otimes \cdots \otimes H) O_f (H \otimes \cdots \otimes H) \left|0 \ldots 0\right\rangle . \tag{5.12}$$

Kunnen we deze toestand wat meer uitwerken? Net als voorheen kunnen we dit stap voor stap berekenen. Eerst passen we de Hadamard-transformatie toe op de nul-basistoestand. Volgens Vgl. (5.10), krijgen we de uniforme superpositie over alle basistoestanden:

$$(H \otimes \cdots \otimes H) \left|0 \ldots 0\right\rangle = \frac{1}{\sqrt{2^n}} \sum_{x_1,\ldots,x_n \in \{0,1\}} \left|x_1, \ldots, x_n\right\rangle .$$

Vervolgens passen we het fase-orakel O_f toe:

$$O_f (H \otimes \cdots \otimes H) \left|0 \ldots 0\right\rangle = \frac{1}{\sqrt{2^n}} \sum_{x_1,\ldots,x_n \in \{0,1\}} (-1)^{f(x_1,\ldots,x_n)} \left|x_1, \ldots, x_n\right\rangle .$$

Wat levert de laatste Hadamard-transformatie op? Door lineariteit kunnen we Hadamard-transformatie toepassen op elke basisvector, zodat we de volgende uitdrukking krijgen voor de toestand in Vgl. (5.12):

$$\begin{aligned}
&(H \otimes \cdots \otimes H) O_f (H \otimes \cdots \otimes H) \left|0 \ldots 0\right\rangle \\
&= \frac{1}{\sqrt{2^n}} \sum_{x_1,\ldots,x_n \in \{0,1\}} (-1)^{f(x_1,\ldots,x_n)} \frac{1}{\sqrt{2^n}} \sum_{y_1,\ldots,y_n \in \{0,1\}} (-1)^{x_1 y_1 + \ldots + x_n y_n} \left|y_1, \ldots, y_n\right\rangle \\
&= \sum_{y_1,\ldots,y_n \in \{0,1\}} \frac{1}{2^n} \left(\sum_{x_1,\ldots,x_n \in \{0,1\}} (-1)^{x_1 y_1 + \ldots + x_n y_n} (-1)^{f(x_1,\ldots,x_n)} \right) \left|y_1, \ldots, y_n\right\rangle ,
\end{aligned} \tag{5.13}$$

waarbij we bij de laatste stap de twee sommen omgewisseld hebben. Voor $n = 1$ is dit precies hetzelfde als Vgl. (5.7). Maar in het algemeen ziet deze uitdrukking er nogal omslachtig en moeilijk te interpreteren uit – het lijkt alsof het circuit een superpositie van basistoestanden berekent, waarbij de amplitude een rare som is van plus- en mintekens!

Laten we proberen wat intuïtie te krijgen door te kijken naar de amplitude van $|0 \ldots 0\rangle$ in Vgl. (5.13). Het kwadraat van dit getal is de kans dat, wanneer we de n qubits meten, alle

uitkomsten nul zijn. Aangezien deze amplitude overeenkomt met $y_1 = \ldots = y_n = 0$, wordt deze simpelweg gegeven door

$$\frac{1}{2^n} \sum_{x_1,\ldots,x_n \in \{0,1\}} (-1)^{f(x_1,\ldots,x_n)}.$$

Wat betekent dit? Stel dat er N_f invoerbitstrings zijn waarvoor f evalueert naar nul en $2^n - N_f$ bitstrings waarvoor f evalueert naar één. Dan geldt het volgende,

$$\frac{1}{2^n} \sum_{x_1,\ldots,x_n \in \{0,1\}} (-1)^{f(x_1,\ldots,x_n)} = \frac{N_f - (2^n - N_f)}{2^n} = \frac{2N_f - 2^n}{2^n}.$$

Er zijn twee interessante uiterste gevallen:[14]

- Als f een *constante functie* is, dan is of $N_f = 0$ (voor de alles-nul-functie) of $N_f = 2^n$ (voor de alles-één-functie). In beide gevallen is de amplitude $\pm 2^n/2^n = \pm 1$. Omdat de kwadraten van alle amplituden moeten optellen tot 1, concluderen we dat alle andere amplituden in Vgl. (5.13) 0 moeten zijn. Met andere woorden, als f constant is, is de toestand in Vgl. (5.13) gewoon $\pm |0 \ldots 0\rangle$. Als we alle n qubits van deze toestand meten dan zullen alle uitkomsten nul zijn.
- Als f een *gebalanceerde functie* is, wat betekent dat er evenveel nullen als enen zijn, dan is $N_f = 2^n/2$ en dus is de amplitude 0. Dit betekent dat als we de toestand in Vgl. (5.13) meten, we nooit kunnen krijgen dat alle uitkomsten gelijk zijn aan nul. Voor een gebalanceerde functie zal dus altijd ten minste één van de meetuitkomsten één zijn.

Merk op hoe deze twee gevallen overeenkomen met heel verschillende interferentiepatronen (zie §2.6.1). In het eerste geval wordt de amplitude van $|0 \ldots 0\rangle$ versterkt tot ± 1 door een zeer gerichte *constructieve* interferentie, terwijl tegelijkertijd alle andere amplitudes verdwijnen door een zeer ver verspreide destructieve interferentie. In het tweede geval ervaart $|0 \ldots 0\rangle$ een *destructieve* interferentie, waardoor op andere plaatsen amplitudes groter dan nul opduiken. De Hadamard-transformatie laat het fundamentele belang van interferentie in quantum computing zien. In de volgende twee secties zullen we gebruik maken van deze interferentie om quantumalgoritmes te ontwerpen voor een groot aantal qubits.

5.2.3 Het Deutsch-Jozsa algoritme

Zijn de bovenstaande observaties ook nog ergens nuttig voor? Jazeker! Stel dat $f\colon \{0,1\}^n \to \{0,1\}$ een onbekende functie is die of constant of gebalanceerd is. Dan is er een eenvoudig quantumalgoritme dat kan bepalen welke van deze twee het geval is, door slechts één enkele evaluatie van f te gebruiken (dus één toepassing van O_f).

Dit algoritme heet het Deutsch-Jozsa algoritme en werkt in vijf eenvoudige stappen:

1. Start met $|0 \ldots 0\rangle$.
2. Pas de Hadamard-transformatie $H \otimes \cdots \otimes H$ toe.
3. Pas het fase-orakel O_f dat hoort bij de functie f toe.
4. Pas weer de Hadamard-transformatie $H \otimes \cdots \otimes H$ toe.
5. Meet alle qubits. Als alle uitkomsten nul zijn, moet de functie f constant zijn. Anders moet de functie gebalanceerd zijn.

[14] Voor $n = 1$ is elke functie of constant of gebalanceerd. Voor $n > 1$ is dit niet waar (bijvoorbeeld, de AND-functie is niet constant of gebalanceerd).

Om te zien hoe dit zich verhoudt tot klassieke algoritmen, moet je bedenken dat elk klassiek algoritme de functie f in het ergste geval minstens $\frac{2^n}{2} + 1$ keer moet evalueren. Stel inderdaad dat we de functie op de helft van de invoerbitstrings toepassen (d.w.z. op $2^n/2$ invoerbits) en we krijgen elke keer hetzelfde antwoord. Dan kunnen we nog steeds niet concluderen dat de functie constant is, omdat het zo zou kunnen zijn dat de functie op de resterende helft van de bitstrings het tegenovergestelde antwoord geeft en dus toch gebalanceerd is. Dit is dus het slechtste geval. In dit scenario hebben we $2^n/2$ vragen verspild en nog steeds niets nuttigs geleerd. Om er zeker van te zijn of f constant of gebalanceerd is, moeten we het evalueren op nog een invoer. In totaal komt dit neer op $\frac{2^n}{2} + 1$ evaluaties van f, vergeleken met 1 evaluatie in het quantumgeval. Merk op dat zelfs voor bescheiden waardes zoals $n = 100$ dit verschil zo dramatisch is dat je niet in staat zou zijn om f zo vaak te evalueren in een redelijke hoeveelheid tijd (tegen die tijd zou de zon zonder brandstof raken en zou je naar een ander zonnestelsel moeten verhuizen).

Samengevat: Als $f\colon \{0,1\}^n \to \{0,1\}$ een functie is die ofwel constant ofwel gebalanceerd is, dan kan het Deutsch-Jozsa algoritme met slechts één evaluatie van f bepalen welke van de twee het geval is. Dit is exponentieel beter dan een klassiek algoritme, dat in het slechtste geval de functie moet evalueren op $\frac{2^n}{2} + 1$ invoerbits.

Huiswerkopdracht 5.2: Deutsch-Jozsa uitvoeren

Het gele vakje Oracle in QUIRKY implementeert het fase-orakel voor een functie die constant of gebalanceerd is. Implementeer het Deutsch-Jozsa algoritme in QUIRKY en gebruik het om te bepalen welke van de twee het geval is.

5.2.4 Het Bernstein-Vazirani algoritme

Hierboven hebben we besproken hoe we de Hadamard-transformatie kunnen gebruiken om een interessant probleem op te lossen. Gegeven een functie $f\colon \{0,1\}^n \to \{0,1\}$ en de belofte dat f constant of gebalanceerd is, was het Deutsch-Jozsa algoritme in staat om te bepalen welke van deze twee mogelijkheden het geval was.

In deze paragraaf bespreken we een ander interessant probleem dat we met een variant van dezelfde procedure kunnen oplossen. Net als voorheen beginnen we met een belofte over de onbekende functie f waarmee we te maken hebben. Deze keer zullen we, in plaats van aan te nemen dat deze constant of gebalanceerd is, aannemen dat deze functie de volgende speciale vorm heeft:

$$f(x_1, \ldots, x_n) = x_1 a_1 \oplus \ldots \oplus x_n a_n, \tag{5.14}$$

waarbij $a_1, \ldots, a_n \in \{0,1\}$ een vaste bitstring is die de functie definieert.

Als $n = 1$, zijn er maar twee van zulke functies:

- $f(x_1) = 0$, in het geval dat $a_1 = 0$, en
- $f(x_1) = x_1$, wat overeenkomt met $a_1 = 1$.

Als $n = 2$ zijn er al vier van zulke functies:

- $f(x_1, x_2) = 0$, als $[a_1, a_2] = [0,0]$,
- $f(x_1, x_2) = x_2$, als $[a_1, a_2] = [0,1]$,
- $f(x_1, x_2) = x_1$, als $[a_1, a_2] = [1,0]$, en
- $f(x_1, x_2) = x_1 \oplus x_2$, als $[a_1, a_2] = [1,1]$.

Merk op dat elk van deze functies de som modulo twee van een deelverzameling van de variabelen x_i berekent. Welke deelverzameling? Als $a_i = 1$, is de bijbehorende variabele x_i onderdeel van de deelverzameling.

In het algemeen zijn er 2^n keuzes van de bitring $a_1, \ldots, a_n$ en dus 2^n functies f met de speciale vorm (5.14). In feite kunnen we Vgl. (5.14) zien als een manier om de bitstring

$$[a_1, \ldots, a_n]$$

te *verbergen* in de functie f. Hoeveel evaluaties van de functie hebben we nodig om deze string te achterhalen? Aangezien

$$\begin{aligned} a_1 &= f(1, 0, \ldots, 0, 0), \\ a_2 &= f(0, 1, \ldots, 0, 0), \\ &\vdots \\ a_n &= f(0, 0, \ldots, 0, 1), \end{aligned}$$

kunnen we concluderen dat n evaluaties van de functie f zeker genoeg zijn. Voor elk klassiek algoritme is dit ook optimaal: elke keer dat je de functie evalueert achterhaal je maar één bit. Aangezien de onbekende bits $a_1, \ldots, a_n$ volledig arbitrair zijn en er n van zijn, moet je de functie minstens n keer evalueren om ze allemaal te achterhalen.

We zullen nu zien dat we dit met een quantumalgoritme veel beter kunnen doen. Laten we om te beginnen bepalen hoe het fase-orakel voor de functie in Vgl. (5.14) werkt op de basistoestanden:

$$\begin{aligned} O_f \left|x_1, \ldots, x_n\right\rangle &= (-1)^{x_1 a_1 \oplus \ldots \oplus x_n a_n} \left|x_1, \ldots, x_n\right\rangle \\ &= (-1)^{x_1 a_1 + \ldots + x_n a_n} \left|x_1, \ldots, x_n\right\rangle . \end{aligned} \tag{5.15}$$

In de tweede stap hebben we gebruikt dat $(-1)^a$ alleen afhangt van a modulo twee (dus of a even of oneven is), dus het maakt niet uit of we optelling modulo 2 ('$\oplus$') of de gewone optelling gebruiken. Hoe kan dit fase-orakel gerealiseerd worden? Voor ons maakt dat niet zoveel uit, omdat ons algoritme het orakel als een black box zal behandelen. Maar omdat het een leuke oefening is, kun je proberen dit uit te zoeken in de volgende opgave.

Oefenopgave 5.5: Het fase-orakel realiseren (optioneel)

In deze opgave ga je het fase-orakel implementeren voor functies van de vorm (5.14).

1. Als $n = 2$ zijn er vier van zulke functies, zoals we hierboven al besproken hebben. Kan je voor elk van deze functies een QUIRKY-circuit vinden dat het fase-orakel implementeert?

2. Leg in woorden of met een plaatje uit hoe je het fase-orakel in het algemene geval kunt implementeren (d.w.z. wanneer $n \geq 1$ en de bits $a_1, \ldots, a_n \in \{0, 1\}$ arbitrair zijn).

We introduceren nu het Bernstein-Vazirani algoritme, dat de verborgen bitstring $[a_1, \ldots, a_n]$ ontrafelt met behulp van een enkele evaluatie van het fase-orakel van f:

1. Start met $\left|0 \ldots 0\right\rangle$.

2. Pas de Hadamard-transformatie $H \otimes \cdots \otimes H$ toe.

3. Pas het fase-orakel O_f dat hoort bij de functie f toe.

4. Pas weer de Hadamard-transformatie $H \otimes \cdots \otimes H$ toe.

5. Meet als laatste alle qubits. De meetuitkomst is exact de bitstring $[a_1, \ldots, a_n]$.

Dit algoritme is identiek aan het Deutsch-Jozsa algoritme van §5.2.3 op de laatste stap na, die in dit geval nog simpeler is.

Huiswerkopdracht 5.3: Bernstein-Vazirani uitvoeren

Het oranje vakje Oracle in QUIRKY implementeert het fase-orakel voor een functie van de vorm (5.14) met $n = 4$. Implementeer het Bernstein-Vazirani algoritme in QUIRKY en gebruik dit om de verborgen bitstring $[a_1, a_2, a_3, a_4]$ te bepalen.

Waarom werkt dit algoritme? Nu is het jouw beurt om de analyse te doen!

Oefenopgave 5.6: Bernstein-Vazirani nagaan

De functie $f(x_1, x_2) = x_2$ komt overeen met de bitstring $[a_1, a_2] = [0, 1]$, zoals we al eerder hebben gezien.
Laat zien dat wanneer je het Bernstein-Vazirani algoritme voor deze functie uitvoert, de meetuitkomst inderdaad altijd $[a_1, a_2] = [0, 1]$ is. Controleer dit niet alleen met QUIRKY, maar schrijf zelf de toestand na elke stap op.

In de volgende huiswerkopdracht kan je het algemene geval analyseren:

Huiswerkopdracht 5.4: Bernstein-Vazirani nagaan (uitdagend)

Laat zien dat als je het Bernstein-Vazirani algoritme uitvoert voor een functie van de vorm (5.14), de meetuitkomst $[a_1, \ldots, a_n]$ is met 100% waarschijnlijkheid.

Hint: Aangezien de eerste vier stappen van het algoritme identiek zijn aan het Deutsch-Jozsa algoritme, wordt de toestand vlak voor de meting gegeven door Vgl. (5.13).

5.3 Op zoek met Grover

Na een veilige terugkeer op aarde zijn Hila en Iman goede vrienden geworden met Alice en Bob. Ze besluiten met z'n vieren nieuwjaarsavond samen door te brengen. Deze dag valt toevallig ook samen met de trekking van de jaarlijkse quantumloterij! Ze weten dat maar één van de vier de hoofdprijs kan winnen, maar wie zal dat zijn? Als we onze vier hoofdpersonen labelen met bitstrings, bijvoorbeeld als

Naam	x_1	x_2
Alice	0	0
Bob	0	1
Hila	1	0
Iman	1	1

dan kunnen we de loterij beschrijven met een functie $f\colon \{0,1\}^2 \to \{0,1\}$ die de waarde 1 geeft voor de bitstring die overeenkomt met de winnaar. Als bijvoorbeeld $f(1,0) = 1$, dan is Hila de winnaar van de loterij van dit jaar.

Hoe kunnen onze vrienden de winnaar bepalen? Met een klassiek algoritme moeten ze de functie tot drie keer evalueren om de winnaar te bepalen. Stel bijvoorbeeld dat ze te weten komen dat $f(0,1) = 0$ en $f(1,0) = 0$ – dan weten ze nog steeds niet of Alice of Iman de winnaar is! Door ouderwetse redenen die al lang vergeten zijn, staat in de regels van de loterij dat de functie maar één keer geëvalueerd mag worden. Maar de loterij vindth het ook prima als we het fase-orakel O_f uitvoeren – het is tenslotte een quantumloterij!

Alice en haar vrienden komen bij elkaar en beginnen na te denken. Na een tijdje wordt Bob ongeduldig en hij stelt voor: *"Laten we gewoon de eerste stappen van Deutsch-Jozsa en Bernstein-Vazirani volgen – dezelfde truc zal vast nog wel een keer werken..."* De anderen weten niet echt een

beter alternatief, dus ze gaan aan de slag en bereiden de volgende toestand voor:

$$(H \otimes H)(|0\rangle \otimes |0\rangle) = |+\rangle \otimes |+\rangle = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle).$$

Vervolgens geven ze de toestand door aan de quantumloterij, die het fase-orakel O_f toepast en de toestand teruggeeft. Laat a en b de twee bits zijn die de winnaar aanduiden, d.w.z. $f(a,b) = 1$ en alle andere functiewaarden zijn nul. De loterij stuurt dan de volgende twee-qubit-toestand terug:

$$\begin{aligned} &\frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle - 2\,|a,b\rangle) \\ &= \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) - |a,b\rangle \\ &= |+\rangle \otimes |+\rangle - |a\rangle \otimes |b\rangle\,, \end{aligned}$$

waarbij de $-2\,|a,b\rangle$ in de eerste regel een van de plustekens vervangt door een minteken. Na het toepassen van een Hadamard-transformatie krijgen ze de volgende toestand:

$$\begin{aligned} &|0\rangle \otimes |0\rangle - H\,|a\rangle \otimes H\,|b\rangle \\ &= |00\rangle - \frac{1}{2}\Big(|0\rangle + (-1)^a\,|1\rangle\Big) \otimes \Big(|0\rangle + (-1)^b\,|1\rangle\Big) \\ &= -\frac{1}{2}\Big(-\,|00\rangle + (-1)^a\,|10\rangle + (-1)^b\,|01\rangle + (-1)^{a+b}\,|11\rangle\Big) \end{aligned}$$

Nu zijn onze vrienden in de war en weten ze niet goed wat ze moeten doen. Hila heeft een idee: *"Ik vind het minteken voor de $|00\rangle$ maar niks. Waarom passen we geen quantumbewerking toe die er als volgt uitziet?"*

$$\begin{aligned} |00\rangle &\mapsto -\,|00\rangle \\ |01\rangle &\mapsto |01\rangle \\ |10\rangle &\mapsto |10\rangle \\ |11\rangle &\mapsto |11\rangle \end{aligned} \tag{5.16}$$

Iman voegt toe: *"Ik denk dat ik weet hoe ik dit kan maken met behulp van een controlled-Z-bewerking en een handvol NOTs..."* In een mum van tijd, komen onze vrienden tot de volgende toestand:

$$-\frac{1}{2}\Big(|00\rangle + (-1)^a\,|10\rangle + (-1)^b\,|01\rangle + (-1)^{a+b}\,|11\rangle\Big)$$

Na een klein poosje realiseert Alice zich: *"Deze twee-qubit-toestand kan worden geschreven als een tensorproduct!"*

$$-\frac{1}{2}\Big(|0\rangle + (-1)^a\,|1\rangle\Big) \otimes \Big(|0\rangle + (-1)^b\,|1\rangle\Big) = -(H \otimes H)\,|a,b\rangle\,.$$

Bob zegt enthousiast: *"Aha! We hoeven alleen nog maar een Hadamard-transformatie toe te passen en beide qubits te meten.* Kan je er ook achter komen wie de winnaar van de quantumloterij van dit jaar wordt?

Huiswerkopdracht 5.5: Quantumloterij

1. Schrijf de kwantumoperatie van Vgl. (5.16) uit met behulp van een controlled-Z-bewerking en een paar NOT-bewerkingen.

   Hint: Houd in gedachten dat de controlled-Z-bewerking $CZ_{1\to 2}$ als volgt werkt op de

basistoestanden $|x, y\rangle$: Als $x = 0$ dan doet het niets. Als $x = 1$ dan werkt het als een Z op de tweede qubit. Je hebt vorige week geleerd hoe je dit kunt maken in QUIRKY.

2. Maak het quantumalgoritme dat Alice, Bob, Hila en Iman hebben bedacht in QUIRKY en bepaal de winnaar van de quantum Loterij van dit jaar.

Het quantumalgoritme dat onze vrienden net ontdekt hebben is een speciaal geval van het algoritme van Grover. Grovers algoritme lost het volgende probleem op: Gegeven een orakel voor een functie $f\colon \{0,1\}^n \to \{0,1\}$, vindt het de $x_1, \ldots, x_n \in \{0,1\}$ waarvoor geldt dat $f(x_1, \ldots, x_n) = 1$. We kunnen dit probleem zien als het vinden van een loterijwinnaar tussen 2^n deelnemers of, wat eenvoudiger gezegd, het vinden van een item dat voldoet aan een bepaalde interessante eigenschap in een ongestructureerde database (waarmee we bedoelen dat de items in de database niet gesorteerd of vergelijkbaar zijn). Met hetzelfde argument dat we eerder gaven, zal elk klassiek algoritme in het slechtste geval naar alle 2^n items moeten kijken voordat het klaar is (gemiddeld zal je nog steeds naar ongeveer de helft van de items moeten kijken). Grovers algoritme daarentegen hoeft het orakel maar een paar keer te gebruiken en dat aantal is evenredig met $\sqrt{2^n}$ - een hoeveelheid die veel langzamener groeit dan de 'klassieke' 2^n. Het algoritme van Grover is een veelzijdig hulpmiddel dat een kwadratische versnelling geeft voor veel rekenproblemen.

5.3.1 Hoekvergroting

Het laatste quantumalgoritme dat we zullen bespreken is een belangrijke subroutine die in veel andere quantumalgoritmes wordt gebruikt (het is bijvoorbeeld de kern van Grovers algoritme voor functies met meer dan $n = 2$ invoerbits).

Het probleem dat deze subroutine oplost lijkt op het eerste gezicht een beetje apart. Het is namelijk een puur quantumprobleem dat niet een heel zinvolle klassieke formulering heeft. Toch komt het op natuurlijke wijze voor in verschillende andere algoritmes, waardoor de subroutine erg nuttig is. Dit laat zien hoe verschillend de ideeën achter quantumalgoritmes zijn en dat er nieuwe manieren van denken nodig zijn voor het uitvinden van nieuwe quantumalgoritmes!

Bovendien is dit een voorbeeld van een probleem waarbij het orakel meer dan één keer geraadpleegd moet worden. Dit is anders dan alle quantumalgoritmes die we eerder in deze quest hebben bekeken, waarbij het orakel maar één keer geraadpleegd hoefde te worden.

Huiswerkopdracht 5.6: Wat is de hoek? (uitdagend)

Alice en Bob zijn overstuur. "Het spijt ons voor de overlast,"vertellen ze. "We hebben deze prachtige reflectie $V(\theta)$ met hoek

$$\theta = +\frac{\pi}{4k} \quad \text{or} \quad \theta = -\frac{\pi}{4k},$$

gemaakt, maar we kunnen ons niet herinneren welke het was – we herinneren ons alleen het gehele getal $k \geq 1$!"[a] Wat hun probleem nog erger maakt is dat de gate zichzelf vernietigt als het meer dan k keer gebruikt wordt.

Kan jij ze helpen? Jouw taak, als je die aanvaardt, is om te bepalen welke van de twee hoeken de juiste is door de gate $V(\theta)$ maximaal k keer te gebruiken.

Hint: Twee opeenvolgende spiegelingen geven een rotatie. Wat krijg je als je NOT en $V(\theta)$ combineert? (Je hebt de algemene formule van Oefenopgave 4.6 niet nodig om deze vraag te beantwoorden).

[a] Als je wilt, kun je deze reflectie zien als een orakel dat één van de twee hoeken op een vreemde manier verbergt.

5.4 Jouw quantumreis

Met deze laatste opdracht sluiten we *The Quantum Quest* af. Wow, er zijn al vijf weken voorbij – deze cursus is een hele reis geweest! We hopen dat je het de afgelopen weken naar je zin hebt gehad en dat je veel interessante wiskunde hebt geleerd.

Als je geen genoeg kunt krijgen van quantum computing, hoef je je geen zorgen te maken. Je bent nu al een ervaren tovenaar op het gebied van quantumbits en goed voorbereid om zelf een meer gevorderd boek te bestuderen. Waarom kijk je niet of de bibliotheek bij jou in de buurt een exemplaar heeft van het boek *'Quantum Computation and Quantum Information'* van Michael Nielsen en Isaac Chuang?

5.5 Oplossingen van de oefenopgaven

Oplossing van Oefenopgave 5.1

De drie andere functies zijn $f = \text{NOT}$ en de twee constante functies $f(0) = f(1) = 0$ en $f(0) = f(1) = 1$.

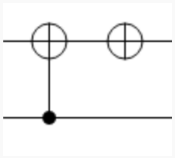
- Voor de NOT-functie geldt:

$$U_{\text{NOT}} |a, b\rangle = |a, b \oplus \text{NOT}(a)\rangle = |a, b \oplus a \oplus 1\rangle = |a, \text{NOT}(b \oplus a)\rangle ,$$

dit kan geschreven worden als de samenstelling van een controlled-NOT-bewerking en een NOT-bewerking op het tweede qubit, dus,

$$U_{\text{NOT}} = (I \otimes \text{NOT})\, \text{CNOT}_{1\to 2}.$$

In QUIRKY ziet dit er als volgt uit:



- Voor de nul-functie $f(0) = f(1) = 0$ geldt:

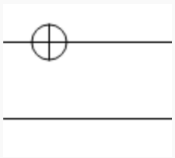
$$U_f |a, b\rangle = |a, b\rangle ,$$

dus we hoeven helemaal niks te doen:

- Voor de één-functie $f(0) = f(1) = 1$ geldt:

$$U_f |a, b\rangle = |a, b \oplus 1\rangle = |a, \text{NOT}(b)\rangle ,$$

dus we hoeven alleen het tweede qubit te inverteren:



Oplossing van Oefenopgave 5.2

Er zijn vier functies: de 'identiteit'-functie $f(x) = x$, de NOT-functie, de nul-functie en de één-functie.

- Voor de identiteit-functie $f(x) = x$, ziet Vgl. (5.6) eruit als

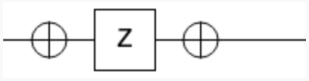
$$O_f |x\rangle = (-1)^x |x\rangle ,$$

dus dit is precies de Z-gate:

- Voor de NOT-functie $f(x) = \text{NOT}(x)$, willen we dat

$$O_f |x\rangle = (-1)^{\text{NOT}(x)} |x\rangle = \text{NOT } Z \text{ NOT } |x\rangle ,$$

wat neerkomt op de volgende reeks bewerkingen:



- Voor de nul-functie met $f(0) = f(1) = 0$ geldt:

$$O_f |x\rangle = |x\rangle ,$$

dus we hoeven helemaal niks te doen:

- Voor de één-functie met $f(0) = f(1) = 1$ geldt:

$$O_f |x\rangle = - |x\rangle ,$$

wat we kunnen bereiken door de eerste twee orakels achter elkaar uit te voeren:

Het eerste orakel geeft inderdaad een minteken als $x = 1$, terwijl het tweede orakel een minteken geeft als $x = 0$, dus in beide gevallen krijgen we een minteken:

$$\begin{aligned} &\text{NOT } Z \text{ NOT } Z |0\rangle = \text{NOT } Z \text{ NOT } |0\rangle = - |0\rangle , \\ &\text{NOT } Z \text{ NOT } Z |1\rangle = \text{NOT } Z \text{ NOT}(- |1\rangle) = -\text{NOT } Z \text{ NOT } |1\rangle = - |1\rangle . \end{aligned}$$

Waar we lineariteit hebben gebruikt om het minteken naar voren te brengen.

Oplossing van Oefenopgave 5.3

We evalueren Vgl. (5.7) voor alle vier de functies.

- Voor $f(x) = x$:

$$HO_f |+\rangle = \frac{1+(-1)}{2} |0\rangle + \frac{1-(-1)}{2} |1\rangle = |1\rangle .$$

- Voor $f(x) = \mathrm{NOT}(x)$:

$$HO_f |+\rangle = \frac{-1+1}{2} |0\rangle + \frac{-1-1}{2} |1\rangle = -|1\rangle .$$

- Voor de nul-functie:

$$HO_f |+\rangle = \frac{1+1}{2} |0\rangle + \frac{1-1}{2} |1\rangle = |0\rangle .$$

- Voor de één-functie:

$$HO_f |+\rangle = \frac{(-1)+(-1)}{2} |0\rangle + \frac{(-1)-(-1)}{2} |1\rangle = -|0\rangle .$$

Oplossing van Oefenopgave 5.4

1. Hier staat $\boxed{???}$ voor $x_1 y_1$.
2. Voor $n = 2$,

$$\begin{aligned}
(H \otimes H) |x_1, x_2\rangle &= (H |x_1\rangle) \otimes (H |x_2\rangle) \\
&= \left(\frac{1}{\sqrt{2}} \sum_{y_1 \in \{0,1\}} (-1)^{x_1 y_1} |y_1\rangle \right) \otimes \left(\frac{1}{\sqrt{2}} \sum_{y_2 \in \{0,1\}} (-1)^{x_2 y_2} |y_2\rangle \right) \\
&= \frac{1}{2} \sum_{y_1, y_2 \in \{0,1\}} (-1)^{x_1 y_1} (-1)^{x_2 y_2} |y_1\rangle \otimes |y_2\rangle \\
&= \frac{1}{2} \sum_{y_1, y_2 \in \{0,1\}} (-1)^{x_1 y_1 + x_2 y_2} |y_1, y_2\rangle ,
\end{aligned}$$

   dus $\boxed{???}$ staat voor $x_1 y_1 + x_2 y_2$.

Oplossing van Oefenopgave 5.5

1. Hier zijn de vier functies en hun fase-orakels:

   - Voor $[a_1, a_2] = [0, 0]$, $O_f |x_1, x_2\rangle = |x_1, x_2\rangle$, dus het fase-orakel doet niks.
   - Voor $[a_1, a_2] = [0, 1]$, $O_f |x_1, x_2\rangle = (-1)^{x_2} |x_1, x_2\rangle$, wat hetzelfde is als $I \otimes Z$.
   - Voor $[a_1, a_2] = [1, 0]$, $O_f |x_1, x_2\rangle = (-1)^{x_1} |x_1, x_2\rangle$, wat hetzelfde is als $Z \otimes I$.
   - Voor $[a_1, a_2] = [1, 1]$, $O_f |x_1, x_2\rangle = (-1)^{x_1+x_2} |x_1, x_2\rangle = (-1)^{x_1}(-1)^{x_2} |x_1, x_2\rangle$, wat hetzelfde is als $Z \otimes Z$.

   Het is dus duidelijk hoe deze vier bewerkingen eruit zien in QUIRKY.

2. Het algemene patroon is vrij duidelijk. Voor een arbitraire functie in de vorm van (5.14) geldt dus:

$$O_f |x_1, \ldots, x_n\rangle = (-1)^{x_1 a_1 + \ldots + x_n a_n} |x_1, \ldots, x_n\rangle = (Z^{a_1} \otimes \cdots \otimes Z^{a_n}) |x_1, \ldots, x_n\rangle ,$$

   waar we stellen dat $Z^1 = Z$ en $Z^0 = I$. In andere woorden, we passen een Z-gate toe op het j-de qubit als en alleen als $a_j = 1$.

Oplossing van Oefenopgave 5.6

In stap 1 beginnen we met:

$$|00\rangle$$

In stap 2 passen we $H \otimes H$ toe en krijgen we:

$$\begin{aligned} &\frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) \\ =&\frac{1}{2} (|00\rangle + |01\rangle + |10\rangle + |11\rangle) \end{aligned}$$

In stap 3 passen we het fase-orakel O_f van $f(x_1, x_2) = x_2$ toe:

$$\frac{1}{2} (|00\rangle - |01\rangle + |10\rangle - |11\rangle)$$

In stap 4 passen we weer $H \otimes H$ toe, met als resultaat:

$$\begin{aligned} &\frac{1}{2}\Big((H \otimes H) |00\rangle - (H \otimes H) |01\rangle + (H \otimes H) |10\rangle - (H \otimes H) |11\rangle\Big) \\ =&\frac{1}{4}\Big((|00\rangle + |01\rangle + |10\rangle + |11\rangle) - (|00\rangle - |01\rangle + |10\rangle - |11\rangle) \\ &+ (|00\rangle + |01\rangle - |10\rangle - |11\rangle) - (|00\rangle - |01\rangle - |10\rangle + |11\rangle)\Big) \\ =&|01\rangle . \end{aligned}$$

In stap 5 meten we beide qubits en krijgen we altijd het meetresultaat $[0, 1]$.