Univerzita Komenského v Bratislave
Fakulta Matematiky, Fyziky a Informatiky
Katedra Informatiky

Konečné Kvantové Výpočtové Modely

(diplomová práca)

Peter Košinár

Vedúci: doc. RNDr. Daniel Olejár, PhD.

Bratislava, 2005

Čestne vyhlasujem, že som túto diplomovú prácu vypracoval samostatne s použitím citovaných zdrojov.

....

Ďakujem svojmu vedúcemu diplomovej práce doc. RNDr. Danielovi Olejárovi, PhD. za zadanie zaujímavej témy a za cenné námety a rady pri jej spracovávaní.

Obsah

Kapitola 1

Úvod

V súčasnosti sa často skloňuje prídavné meno "kvantový" v spojeniach ako "kvantový počítač" či "kvantová kryptografia" a operuje sa pritom formulkami ako "nerozbiteľné šifrovanie" alebo "žiadna klasická šifra nie je bezpečná". Aj keď niektoré z týchto spojení sú len mediálne dobre znejúcim preháňaním, väčšina je založená na známych výsledkoch – existuje napríklad polynomiálny algoritmus pre problémy faktorizácie a diskrétneho logaritmu [Sho94], ktoré sa považujú za "ťažké" pre klasický počítač, či protokoly pre výmenu kľúča [BB84], ktorých bezpečnosť je za istých podmienok garantovaná fyzikou. Oba tieto algoritmy majú spoločného menovateľa – na ich realizáciu je potrebné využiť fenomény popisované kvantovou fyzikou. Zatiaľčo v prípade druhého z nich boli zaznamenané veľké experimentálne úspechy, praktické využitie prvého vyžaduje zdroje, ktoré zatiaľ nie sú k dispozícii. Niektorí odborníci dokonca zastávajú názor, že jeho praktická realizácia nebude možná nikdy.

Na druhej strane, rovnako, ako je zmysluplné študovať napríklad Turingove stroje s nerekurzívnymi orákulami, či rôzne modely hypervýpočtov, má zmysel študovať aj modely motivované fyzikou, no nemajúce fyzikálnu realizáciu. Práve "fyzikálne" znejúce slovko *kvantový* však často vytvára dojem, že štúdium tejto oblasti vyžaduje hlboké fyzikálne základy a tým odradí mnohých potenciálnych záujemcov.

Hlavným cieľom tejto práce je preto vysvetlenie základných pojmov potrebných pre pochopenie mechanizmu kvantových výpočtov, oprostené od potreby akýchkoľvek netriviálnych znalostí z oblasti fyziky. Na druhej strane, od čitateľa sa vyžadujú matematické znalosti (práca s maticami, predstava o vektorových priestoroch, ...) a základné znalosti z oblasti klasickej teórie výpočtov. Požadované znalosti neprekračujú rozsah prednášok zo základných vysokoškolských kurzov matematiky a informatiky.

V kapitole 2 sú definované a vysvetlené základné pojmy z klasickej teórie výpočtov a zložitosti, na ktoré sa budeme neskôr odvolávať. Kvôli zjednoteniu klasických a kvantových výpočtov bol zvolený prístup využívajúci abstraktný jednotný model, vďaka ktorému je možné uskutočniť prechod od klasických modelov k modelom kvantovým jednoduchšie. Daňou za toto zjednodušenie je vyšší stupeň abstrakcie a s tým spojená náročnosť na pochopenie. Preto je vhodné, aby si túto kapitolu preštudoval aj čitateľ oboznámený s klasickými modelmi výpočtov.

Kapitola 3 je jednou z dvoch hlavných kapitol tejto práce. Sú v nej definície základných pojmov, ktoré sa využívajú v teórii kvantových výpočtov, pričom sa na ne snažíme pozerať čisto matematickou optikou.

V kapitole 4 sa zaoberáme niektorými druhmi kvantových výpočtových modelov a špeciálne modelmi konečnými. Práve konečnosť, v podobe, v akej ju zavedieme, je vlastnosť, ktorá by mohla pomôcť pretvoriť čisto matematický model do fyzickej podoby. Nekladieme si za cieľ spraviť prehľad kvantových výpočtových modelov; zamerali sme sa len na modely, ktoré sú dostatočne jednoduché, no napriek tomu je možné demonštrovať na nich jednotlivé odlišnosti medzi klasickými a kvantovými výpočtami. Podrobnejší prehľad mnohých druhov modelov možno nájsť napríklad v [Pet98].

Napokon v poslednej kapitole sú zosumarizované známe výsledky a niektoré z otvorených problémov, ktoré súvisia s kvantovými výpočtovými modelmi.

Kapitola 2

Klasické modely výpočtov

V tejto kapitole zavedieme základné pojmy z oblasti teórie výpočtov a zložitosti, ktoré budeme neskôr využívať pri prechode ku kvantovým zariadeniam. Abstraktný výpočtový model, ktorý tu definujeme, sa napriek svojej názornosti bežne nevyužíva.

2.1 Výpočtové zariadenie

Najprv si ujasníme, čo si budeme predstavovať pod výpočtovým zariadením. Aby bola naša definícia čo najvšeobecnejšia, snažili sme sa vytvoriť čo najabstraktnejší model, ktorý by bol na jednej strane dostatočne všeobecný, aby bol aplikovateľný aj na ne-klasické (v tomto prípade kvantové) modely, no pritom aby bol princíp jeho operovania dostatočne nenáročný na predstavivosť. Intuitívne, ak hovoríme, že sme čosi *vy*počítali, zvyčajne tým myslíme proces podobný nasledujúcej schéme:

1. Dostaneme akési "zadanie" problému; v prípade nášho abstraktného zariadenia ho budeme nazývať *vstup*. Týmto môže byť napríklad rovnica, obrázok či slovné zadanie. V zadaní sa vyskytuje tiež otázka, od ktorej budeme požadovať, aby množina odpovedí na ňu bola konečná (zväčša dokonca bude iba dvojprvková – "áno"/"nie"). Ako taký, vstup je vo väčšine prípadov nevhodný na priamu manipuláciu – preto ho najprv treba preložiť do podoby vhodnej na ďalšie spracovanie.

2. To aj spravíme – vstup si "preložíme" do podoby, s ktorou manipulovať vieme. Napríklad tak môžeme z obrázku vybrať podstatné čiastkové informácie, slovnú úlohu si môžeme previesť do podoby rovníc, ... V prípade abstraktného zariadenia budeme takto pretvorený vstup nazývať *konfigurácia*. Konfigurácia reprezentuje kompletnú informáciu, ktorá v danom momente o úlohe existuje – špeciálne, pokiaľ sa pri riešení dvoch úloh dostaneme do rovnakej konfigurácie, ďalší postup sa v oboch úlohách nemôže líšiť. V našom modeli pozostáva konfigurácia z troch častí – jedna je časť reprezentujúca samotné zariadenie (jeho *(vnútorný) stav*), druhá sa vzťahuje ku vstupu (*vonkajší stav* – vyjadruje momentálny vzťah zariadenia a zadaného vstupu) a vý-

znam tretej, ktorá sa vzťahuje k výsledkom činnosti zariadenia, ozrejmíme zanedlho. Vnútorný a vonkajší stav budeme spoločne nazývať iba *kompletný stav* zariadenia.

3. S "prerobeným" zadaním potom vykonávame určité operácie (napríklad na rovnicu aplikujeme ekvivalentné úpravy), dokým nedospejeme do stavu, ktorý považujeme za finálny (napríklad, keď sa nám podarí vyjadriť hodnotu neznámej pomocou zadaných údajov). Vo formalizme abstraktných zariadení to môžeme sformulovať tak, že realizujeme určité *kroky výpočtu.*

   Počas robenia týchto krokov sa občas môže stať, že prídeme do stavu, keď budeme vedieť povedať niečo v zmysle: "Ak je čosi zelené, tak je odpoveď áno, ak je to modré, je odpoveď nie, inak si zatiaľ nie sme istí odpoveďou." Inak povedané, občas sa stane, že výpočet za určitých podmienok budeme vedieť ukončiť v určitej sade prípadov, no v inej sade ešte budeme potrebovať pokračovať ďalej. Takáto situácia bude nastávať hlavne za podmienky, že pri výpočte sa na istých miestach budeme rozhodovať podľa náhody – napríklad podľa výsledku "hodenia kockou".

   Už spomínanou treťou časťou konfigurácie je teda aj k-tica (kde k je počet možných odpovedí, ktoré môžeme od zariadenia dostať; najčastejšie $k = 2$) reálnych čísel, ktoré reprezentujú pravdepodobnosti toho, že od zariadenia dostaneme v danom momente príslušnú odpoveď. Každý jednotlivý krok výpočtu môže (ale nemusí) túto k-ticu upraviť. Takáto úprava musí byť monotónna – ak sa výpočet ukončí s odpoveďou "áno" s pravdepodobnosťou 30%, je nemysliteľné, aby po vykonaní akejkoľvek ďalšej operácie táto pravdepodobnosť klesla.

   Ďalej, aby sa nám nestalo, že by sme sa pri výpočte "zasekli" – t.j. nevedeli, aký krok spraviť ďalej, budeme tiež požadovať, aby sme v každej konfigurácii vedeli jednoznačne povedať, čo máme robiť ďalej. Špeciálnu výnimku tvoria konfigurácie, ktoré označíme ako *ukončujúce.* Niekedy budeme napríklad požadovať, aby výpočet skončil, až keď už bude úplne isté, že dostaneme niektorú z odpovedí (zjavne, akonáhle takýto moment nastane, niet dôvodu ďalej pokračovať vo výpočte). Na druhej strane, nie vždy nás budú zaujímať skutočné pravdepodobnosti tej či onej odpovede – občas nám môže stačiť, ak pravdepodobnosť niektorej z odpovedí prekročí istý prah.

Formálne je tento postup zachytený v definíciach 1, 2, 3 a 4.

Ešte pred ich uvedením je vhodné uvedomiť si isté detaily. Čo napríklad robiť v prípade, že výpočet bude bežať "donekonečna" – že nikdy nenarazíme na ukončujúcu konfiguráciu? V takomto prípade urobíme jeden nevýpočtový krok – vezmeme limity pravdepodobností prislúchajúcich jednotlivým odpovediam. Keďže všetky pravdepodobnosti sú zhora ohraničené a monotónne rastú, tieto limity budú existovať.

Teraz už môžeme uviesť formálne definície jednotlivých pojmov:

Definícia 1 *Usporiadanú šesticu $M = (I, i, S, \vdash, T, O)$ budeme nazývať (abstraktné)* ***výpočtové zariadenie*** *M, ak platia nasledujúce podmienky. Označme množinu (neúplných) pravdepodobnostných rozložení $P = \{(p_o) \in \langle 0, 1\rangle^{|O|} \| \sum_{o \in O} p_o \leq 1\}$ a $C = S \times P$ množinu*

konfigurácií *zariadenia. Pre danú konfiguráciu bude $p_o(C)$ označovať príslušnú zložku rozloženia pravdepodobnosti. Podobne, $s(C)$ bude označovať príslušný kompletný stav zariadenia v danej konfigurácii. Vysvetlenie použitej symboliky:*

- *I je akákoľvek množina* ***vstupov****.*
- *S je akákoľvek množina* ***kompletných stavov*** *zariadenia.*
- *O je akákoľvek množina* ***výstupov****; budeme na nej predpokladať nejaké intuitívne usporiadanie – napríklad $\{a,b\}$ bude formálne iná ako množina $\{b,a\}$.*
- *$T \subseteq C$ je množina* ***ukončujúcich konfigurácií***
- *$i : I \to S$ je zobrazenie prekladajúce vstupy na stavy, ktoré triviálne rozšírime na zobrazenie $\hat{i} : I \to C$ pomocou predpisu $\hat{i}(x) = (i(x), 0, \ldots, 0)$. Pre zjednodušenie budeme písať $i(x)$ namiesto $\hat{i}(x)$; význam bude zrejmý z kontextu.*
- *$\vdash$: $(S - s(T)) \to S \times P$ je zobrazenie, ktoré danému kompletnému stavu priradí nasledujúci kompletný stav a určí pravdepodobnosti, že sa výpočet ukončí s jednotlivými odpoveďami.*

Pokiaľ to nebude zrejmé z kontextu, jednotlivé zložky usporiadanej šestice patriacej k zariadeniu M budeme označovať dolným indexom M (napríklad I_M, $\vdash_M$, …).

Definícia 2 *Nech M je abstraktné výpočtové zariadenie. Potom postupnosť konfigurácií $\{C_k \in C_M\}_{k=0}^n$ budeme nazývať (konečný)* ***výpočet*** *dĺžky n, ak je splnené:*

1. *$\vdash (s(C_k)) = (s(C_{k+1})), (d_o)_{o\in O})$. Inak povedané, konfigurácie musia na seba pomocou zobrazenia $\vdash$ nadväzovať.*
2. *$p_o(C_{k+1}) = p_o(C_k) + d_o\left(1 - \sum_{o\in O} p_o(C_k)\right)$ pre každé $o \in O$. Pravdepodobnostné rozloženie odpovedí v nasledujúcej konfigurácii teda dostaneme tak, ako nám hovorí zobrazenie $\vdash$, akurát ho ešte naškálujeme na interval $\langle 0, p\rangle$, kde p označuje pravdepodobnosť, že výpočet ešte neskončil.*

Keďže druhá podmienka jednoznačne rozširuje zobrazenie $\vdash$ na zobrazenie $\hat{\vdash} : C - T \to C$, budeme ho naďalej používať aj v takejto forme. Pre zjednodušenie zápisu budeme písať $x \vdash y$ namiesto $\vdash (x) = y$ a pokiaľ nás pravdepodobnosti nebudú explicitne zaujímať, budeme ich vynechávať.

Výpočet budeme nazývať ***ukončený****, ak konfigurácia C_n je ukončujúca (t.j. $C_n \in T_M$).*

Definícia 3 *Nech M je abstraktné výpočtové zariadenie. Potom výpočet $\{C_k\}_{k=0}^n$ sa nazýva* ***výpočtom*** *(dĺžky n)* ***na vstupe*** *w, ak platí $C_0 = i_M(w)$.*

Keďže relácia $\vdash$ je zobrazením (podľa definície 1), každý výpočet abstraktného zariadenia na vstupe $w \in I$ je buď ukončený, alebo je jednoznačne predĺžiteľný o ďalší krok. Má preto zmysel zaviesť nasledujúcu definíciu:

Definícia 4 *Nech M je abstraktné výpočtové zariadenie. Potom pravdepodobnostné rozloženie počítané zariadením M je zobrazenie: $f : I_M \to \langle 0,1\rangle$, $f : w \mapsto (f_o(w))_{o\in O}$, kde $f_o(w) = p_o(C_n)$, keď $\{C_k\}_{k=0}^n$ je ukončený výpočet na vstupe w.*

Podobne, môžeme povedať, že postupnosť $\{C_k \in C_M\}_{k=0}^\infty$ budeme nazývať ***nekonečný výpočet****, ak každý jeho počiatočný úsek (t.j. každá podpostupnosť tvaru $\{C_k \in C_M\}_{k=0}^n$) je výpočtom. Vtedy kladieme $f_o(w) = \lim_{n\to\infty} p_o(C_n)$.*

Čitateľ oboznámený s pojmami z teórie výpočtov si akiste povšimol, že naše abstraktné zariadenie sa javí byť deterministickým. Je to skutočne tak; v každom okamihu je jasné, ako výpočet bude pokračovať ďalej. Avšak vhodnou voľbou zobrazenia "krok výpočtu" možno realizovať aj mnohé nedeterministické modely, ako ukážeme v časti 2.4.

Tradične sa výpočet zariadenia, ktoré nie je stopercentne deterministické (t.j. má viac ako jeden možný výpočet) znázorňuje ako strom, ktorého uzlami sú konfigurácie a listami ukončujúce konfigurácie a jednotlivé hrany či uzly v tomto strome sú občas ohodnotené. Naše zariadenie takýto strom vo svojej podstate skryte obsahuje – jednotlivé jeho konfigurácie zodpovedajú nie jednotlivým uzlom či listom stromu, ale celým poschodiam. Význam tohto prístupu sa ukáže v ďalšom texte.

2.2 Výpočtový model

Abstraktné zariadenie ako také je príliš "silné". Ak si vezmeme ľubovoľnú funkciu $f : A \to B$ (kde B je konečná množina), stačí vziať zariadenie $M = (I, i, S, \vdash, T, O)$ definované nasledovne:

- $I = A$, $O = B$
- $S = A \times \cup\{\Box\}$ (kde $\Box \notin A$)
- $T = (\Box, p_1, \ldots, p_{|B|})$
- $i(x) = x$
- $x \vdash (\Box, \overbrace{0, \ldots, 0}^{f(x)-1}, 1, \overbrace{0, \ldots, 0}^{|B|-f(x)})$ (inak povedané, pravdepodobnosť odpovede $f(x)$ sa nastaví na 1, ostatné budú nulové).

Toto zariadenie má tú vlastnosť, že na každom vstupe je jeho výpočet jednokrokový a na vstupe x je výstupom pravdepodobnostné rozloženie, v ktorom má hodnota $f(x)$ priradenú pravdepodobnosť 1. Možno teda povedať, že zariadenie M "dokonale počíta" funkciu f. Zjavne, zaoberať sa čímsi natoľko triviálnym by nemalo zmysel.

Namiesto toho si budeme všímať, čo všetko dokáže zariadenie počítať, ak budeme uplatňovať nejaké obmedzenia na množiny stavov S, ukončujúcich konfigurácií T a zobrazenie "krok výpočtu" $\vdash$. Tieto obmedzenia môžu byť mnohých druhov:

- Množina S môže mať predpísaný tvar – napríklad musí byť tvaru $K \times Q \times \Sigma^*$, kde K a Q sú dve množiny a Σ je nejaká abeceda. Obyčajne množiny K, Q a Σ nebudeme bližšie špecifikovať (t.j. ich voľbou budú parametrizované jednotlivé abstraktné zariadenia, ktoré týmto obmedzeniam vyhovujú), iba určíme nejaké veľmi všeobecné podmienky typu "K je konečná neprázdna množina" a podobne.

- Rovnako môžeme predpísať nejakú vlastnosť kroku výpočtu – zväčša zobrazenie kroku výpočtu dostaneme ako predĺženie nejakej jednoduchšej funkcie na celú množinu kompletných stavov. Výpočet sa tým stane istým spôsobom "lokálne definovaný" – predlžovaná funkcia obyčajne nebude mať k dispozícii kompletný stav, ale len jeho časť; nebude sa teda dívať na konfiguráciu globálne, ale len lokálne.

- Ďalšia možná zmena spočíva vo vhodnej voľbe množiny T. Jej vhodným rozšírením/zúžením môžeme niekedy významne regulovať silu zariadenia.

Každá sada obmedzení určuje istý **výpočtový model**. Akonáhle určíme model, má zmysel začať sa zaoberať jeho vlastnosťami – t.j. aké funkcie dokáže vypočítať či ako rýchlo to dokáže (v zmysle časti 2.6).

Teraz sa už môžeme začať venovať jednotlivým klasickým výpočtovým modelom.

2.3 Konečné automaty

Jedným z najjednoduchších výpočtových modelov je (jednosmerný) konečný automat (FSA alebo len FA[1]). Intuitívna predstava je veľmi jednoduchá – konečný automat je zariadenie s "veľmi malou" *pamäťou*, ktoré rozdeľuje *slová* zo vstupnej *abecedy* do dvoch skupín – slová patriace do nejakej množiny (táto množina sa nazýva *jazyk* rozpoznávaný daným automatom) a slová do nej nepatriace. Vykonáva to tak, že vstupné slovo číta písmenko po písmenku a v každom kroku môže na základe prečítaného písmenka zmeniť informáciu uloženú vo svojej pamäti. Keď príde na koniec vstupného slova, na základe obsahu svojej pamäte rozhodne, či slovo do jazyka patrí, alebo nie.

Formálnu definíciu uvedieme vo formalizme abstraktných výpočtových zariadení zavedených v časti 2.1. Pre úplnosť však najprv uvedieme niekoľko užitočných definícií z oblasti formálnych jazykov:

Definícia 5 *Množina Σ sa nazýva **abeceda**, ak je konečná a neprázdna. Prvky abecedy sa nazývajú* písmená *alebo* symboly*. Konečná postupnosť $w = a_1a_2a_3 \dots a_n$ písmen abecedy Σ sa nazýva **slovo** v abecede Σ. Dĺžka tejto postupnosti je **dĺžka slova** a označuje sa $|w|$. Množina všetkých slov v abecede Σ sa označuje Σ^* (označenie pochádza zo všeobecnejšej operácie, ktorou sa tu však nebudeme zaoberať).*

Príkladmi abecied sú množiny $\Sigma_1 = \{a, b, c\}$, $\Sigma_2 = \{a\}$, $\Sigma_3 = \{$a,r,s,š,t,ť,v$\}$. Slovami sú napríklad *bacbba* (v abecede Σ_1), či *štvrťvrstva* v abecede Σ_3. Špeciálne je slovom aj prázdna

[1]z angl. Finite (State) Automaton

postupnosť písmen – takzvané prázdne slovo, ktoré sa označuje ε (používa sa napríklad, ak treba explicitne zdôrazniť, že niekde sa nachádza "nič"; na rozdiel od prípadu, keď sa niekde nenachádza ani "nič").

Na slovách sa zavádza operácia *zreťazenia* (značíme $\cdot$), definovaná ako nadviazanie dvoch postupností písmeniek. Napríklad tak *štvrť*$\cdot$*vrstva* = *štvrťvrstva*. Špeciálne, pre každé slovo w platí $\varepsilon \cdot w = w \cdot \varepsilon = w$. Namiesto označenia $u \cdot v$ budeme občas písať iba uv. Na zjednodušenie zápisu sa používa označenie a^n ako skratka pre $\underbrace{aaa\ldots aa}_{n}$.

Definícia 6 *Ak Σ je abeceda, tak ľubovoľná podmnožina $L \subseteq \Sigma^*$ sa nazýva* ***jazyk*** *nad abecedou Σ.*

Príkladom jazyka (nad abecedou Σ_1) môže byť $L_1 = \{ab, bab, bba\}$ (konečný jazyk), či $L_2 = \{a^{2^n} \mid n \geq 0\}$. Špeciálne, nad každou abecedou existujú jazyky $\emptyset$ a Σ^*.

Teraz sme už pripravení na definíciu konečného automatu. Jeho pamäť bude reprezentovaná (neprázdnou) konečnou množinou K, ktorá sa tradične nazýva množina stavov (vnútorných stavov nášho abstraktného zariadenia). Každý jej prvok zodpovedá jednému možnému obsahu pamäte. Počiatočný obsah pamäte budeme označovať q_0 (kde $q_0 \in K$) a tie obsahy pamäte, ktoré spôsobia kladnú odpoveď na otázku, či slovo patrilo do jazyka – tzv. *akceptačné* stavy – budú označené F. Všetky zvyšné stavy sú z definície *zamietacie*. Zostáva nám ešte určiť, ako automat funguje – t.j. ako si upravuje obsah pamäte na základe prečítaného písmenka. Na tento účel sa zavádza prechodová δ-funkcia, ktorá nám na túto otázku odpovedá z nasledujúcom zmysle: $\delta : K \times \Sigma \to K$ je zobrazenie, ktoré dvojici (aktuálny obsah pamäte, prečítané písmenko) priradí nový obsah pamäte. Tradične sa pod konečným automatom chápe práve usporiadaná pätica $(K, \Sigma, \delta, q_0, F)$; v našom prípade však bude výhodné, ak tieto pojmy prevedieme do podoby abstraktných zariadení.

Definícia 7 *Nech Σ je abeceda, K je konečná neprázdna množina stavov, $F \subseteq K$ je množina akceptačných stavov, $q_0 \in K$ je počiatočný obsah pamäte a $\delta : K \times \Sigma \to K$ je prechodová funkcia. Položme*

1. $I := \Sigma^*$ *(vstupom konečného automatu je akékoľvek slovo)*
2. $S := K \times \Sigma^* \cup \{\Box\}$ *– prvá zložka reprezentuje obsah pamäte, druhá doposiaľ neprečítaný úsek vstupného slova; špeciálny stav $\Box$ je určený na ukončenie výpočtu; predpokladá sa, že $\Box \notin K \times \Sigma^*$*
3. $T := \{\Box\}$
4. $O := \{\top, \bot\}$ *($\top$ reprezentuje odpoveď kladnú, $\bot$ odpoveď zápornú)*
5. $i(w) := (q_0, w)$
6. $(p, a \cdot w) \vdash (q, w, 0, 0)$ *ak $a \in \Sigma$, $w \in \Sigma^*$, $p, q \in K$ a $\delta(p, a) = q$*
7. $(p, \varepsilon) \vdash (\Box, 1, 0)$ *ak $p \in F$*

8. $(p, \varepsilon) \vdash (\Box, 0, 1)$ *ak* $p \in K - F$.

Potom abstraktné zariadenie $M = (I, i, S, \vdash, T, O)$ *nazveme* ***jednosmerný konečný automat*** *(pracujúci nad abecedou* Σ*), skrátene píšeme 1FA.*

Je zrejmé, že takto definovaný model nemôže na nejakom vstupe bežať donekonečna. Navyše, zakaždým je výsledkom jeho práce rozloženie pravdepodobnosti, ktoré je úplné – súčet pravdepodobností je rovný jednotke. Môžeme preto definovať jazyk, ktorý takýto automat rozpoznáva:

Definícia 8 *Ak* M *je 1FA nad abecedou* Σ*, potom jazyk ním rozpoznávaný (akceptovaný) je množina*

$$L(M) = \{w \in \Sigma^* \mid p_{\top}(w) = 1\}$$

.

Môžeme si tiež všimnúť triedu všetkých jazykov, ktoré sú akceptované nejakým 1FA:

Definícia 9 *Trieda jazykov akceptovaných 1FA sa nazýva* ***trieda regulárnych jazykov*** *a označuje sa* $\mathcal{R}$.

Je mnoho známych charakterizácii triedy $\mathcal{R}$, viac o nich sa možno dozvedieť v ktorejkoľvek publikácii venovanej formálnym jazykom. Nás budú zaujímať hlavne uzáverové vlastnosti tejto triedy vzhľadom na isté operácie.

Najprv všeobecná definícia:

Definícia 10 *Nech* A *je množina,* $g : A^n \to A$ *akákoľvek* n*-árna operácia na množine* A *a* $X \subseteq A$*. Potom množina* X ***je uzavretá na operáciu*** g *ak platí* $g(x_1, \ldots, x_n) \in X$ *pre všetky* $(x_1, \ldots, x_n) \in X^n$.

Keďže prvkami triedy jazykov sú jazyky a jazyky sú množiny, základnými operáciami, ktoré nás budú zaujímať, sú operácie množinové – prienik, zjednotenie, prípadne komplement vzhľadom na určitú špecifikovanú množinu (napríklad Σ^*). Ďalšia zaujímavá operácia je už nemnožinového charakteru – je to takzvané zreťazenie dvoch jazykov:

Definícia 11 *Nech* $L_1 \subseteq \Sigma^*$*,* $L_2 \subseteq \Sigma^*$ *sú dva jazyky nad abecedou* Σ*. Potom jazyk*

$$L_1 \cdot L_2 := \{u \cdot v \mid u \in L_1, v \in L_2\}$$

sa nazýva ***zreťazenie jazykov*** L_1 ***a*** L_2.

Napokon, ďalšie dve zaujímavé a bežne študované operácie sú operácie homomorfizmu a inverzného homomorfizmu. Pod homomorfizmom rozumieme presne to isté, čo sa pod ním chápe v algebre, keďže množina Σ^* s operáciou zreťazenia tvorí pologrupu.

Definícia 12 *Nech $L \subseteq \Sigma_1^*$, $L' \subseteq \Sigma_2^*$ sú jazyky a $h : \Sigma_1^* \to \Sigma_2^*$ je homomorfizmus (t.j. zobrazenie spĺňajúce $h(u \cdot v) = h(u) \cdot h(v)$). Potom* ***homomorfný obraz jazyka*** *L je jazyk*

$$h(L) := \{h(w) \mid w \in L\}$$

a ***inverzným homomorfným obrazom jazyka*** *L je jazyk*

$$h^{-1}(L') := \{w \in \Sigma_1^* \mid h(w) \in L'\}$$

.

Trieda regulárnych jazykov $\mathcal{R}$ je jednou z mála "bežných" tried, ktorá je uzavretá na všetky vymenované uzáverové operácie.

2.4 Nedeterminizmus a iné rozšírenia

Ako už bolo povedané, konečný automat je jedným z najjednoduchších výpočtových modelov. Daňou za túto jeho jednoduchosť je jeho malá výpočtová sila – trieda regulárnych jazykov neobsahuje množstvo jednoducho-vyzerajúcich jazykov; napríklad jazyk $\{a^n b^n \mid n \geq 0\}$ už nie je regulárny.

Bolo preto navrhnutých mnoho úprav, ktoré by konečnému automatu pridali výpočtovú silu, no súčasne nezvýšili príliš jeho zložitosť.

- Najbežnejšie rozšírenie, používané aj u iných modelov, je pridanie vlastnosti **nedeterminizmu**. Intuitívna predstava za ním je veľmi jednoduchá; zmení sa iba prechodová δ-funkcia automatu. Namiesto toho, aby si zmenil obsah pamäte na jednu konkrétnu hodnotu podľa prečítaného písmenka zo vstupu, automat dostane možnosť *voľby* – bude mať na výber viacero nových obsahov pamäte. Automat sa "bude snažiť" vybrať si taký, ktorý spôsobí, že skončí s výstupom $\top$. V prípade, že je to nemožné, vyberie si ľubovoľný z nich.

  Takýto pohľad však nesúhlasí s našou predstavou o počítaní prezentovanou v časti 2.1 – v zmysle, v akom chápeme výpočet, ďalší krok výpočtu nemôžeme uhádnuť, ale musíme vedieť sformulovať pravidlo, podľa ktorého ho jednoznačne určíme.

  Preto sa pokúsime pretvoriť priamočiaru "hádaciu" predstavu do podoby "nehádacej". Namiesto toho, že automat v danom mieste uhádne správnu odpoveď, bude ďalej skúmať súčasne všetky možné ďalšie pokračovania. Pokiaľ príde na koniec a zistí, že sa mu podarilo v aspoň jednom prípade sa dostať do stavu, v ktorom dáva odpoveď $\top$, odpovie $\top$; inak dá odpoveď $\perp$. Inak povedané, keby sme si výpočet predstavovali ako strom (kde uzly by boli konfigurácie), automat by vlastne robil jeho prehľadávanie do šírky.

  Obyčajne sa táto konštrukcia používa práve na dôkaz ekvivalentnej výpočtovej sily deterministických a nedeterministických konečných automatov. V našom prípade je užitočnejšie vnímať nedeterminizmus práve z pohľadu prehľadávania všetkých možných pokračovaní vo výpočte, ako z pohľadu akéhosi "uhádnutia" správnej cesty – podobne sa totiž budú správať niektoré kvantové modely.

Formálna definícia nedeterministického konečného automatu je prakticky totožná s definíciou konečného automatu (označenie 2^X používame pre potenčnú množinu tvorenú všetkými podmnožinami množiny X):

Definícia 13 *Nech Σ je abeceda, K je konečná neprázdna množina stavov, $F \subseteq K$ je množina akceptačných stavov, $q_0 \in K$ je počiatočný obsah pamäte a $\delta : K \times \Sigma \to K$ je prechodová funkcia. Položme*

1. $I := \Sigma^*$
2. $S := 2^K \times \Sigma^* \cup \{\Box\}$ *– prvá zložka reprezentuje možné obsahy pamäte, druhá doposiaľ neprečítaný úsek vstupného slova; špeciálny stav $\Box$ je určený na ukončenie výpočtu; predpokladá sa, že $\Box \notin 2^K \times \Sigma^*$*
3. $T := \{\Box\}$
4. $O := \{\top, \bot\}$
5. $i(w) := (\{q_0\}, w)$
6. $(A, a \cdot w) \vdash (B, w, 0, 0)$ *ak* $a \in \Sigma$, $w \in \Sigma^*$, $A, B \subseteq K$ *a* $B = \{q \in K \mid p \in A \wedge \delta(p, a) = q\}$
7. $(A, \varepsilon) \vdash (\Box, 1, 0)$ *ak* $A \cap F \neq \emptyset$
8. $(A, \varepsilon) \vdash (\Box, 0, 1)$ *ak* $A \cap F = \emptyset$.

Uvažujme teraz nasledovný príklad (nedeterministického) automatu:

1. $K = \{q_0, q_a, q_{ab}, q_{abb}, q_{abba}\}$
2. $\Sigma = \{a, b\}$
3. $F = \{q_{abba}\}$
4. $\delta(q_0, a) = \{q_0, q_a\}$
5. $\delta(q_0, b) = \{q_0\}$
6. $\delta(q_a, a) = \emptyset$
7. $\delta(q_a, b) = \{q_{ab}\}$
8. $\delta(q_{ab}, a) = \emptyset$
9. $\delta(q_{ab}, b) = \{q_{abb}\}$
10. $\delta(q_{abb}, a) = \{q_{abba}\}$
11. $\delta(q_{abb}, b) = \emptyset$

12. $\delta(q_{abba}, a) = \{q_{abba}\}$
13. $\delta(q_{abba}, b) = \{q_{abba}\}$

Tento automat akceptuje jazyk, ktorý pozostáva práve zo slov obsahujúcich (pod)slovo *abba*. Formálnejšie, $L(M) = \{w \in \Sigma^* \mid \exists u, v \in \Sigma^* : w = u \cdot abba \cdot v\}$. Vďaka nedeterminizmu automat "vie uhádnuť", kedy sa má začať pokúšať o spracovanie podpostupnosti *abba*.

Napríklad, na slove *aabbaa* by výpočet automatu vyzeral nasledovne:

$(\{q_0\}, aabbaa, 0, 0) \vdash (\{q_0, q_a\}, abbaa, 0, 0) \vdash (\{q_0, q_a\}, bbaa, 0, 0) \vdash (\{q_0, q_{ab}\}, baa, 0, 0) \vdash (\{q_0, q_{abb}\}, aa, 0, 0) \vdash (\{q_0, q_a, q_{abba}\}, a, 0, 0) \vdash (\{q_0, q_a, q_{abba}\}, \varepsilon, 0, 0) \vdash (\Box, 1, 0)$.

Teraz si všimneme jednu zaujímavú reprezentáciu (nedeterministických) konečných automatov. Predstavme si, že množina K pozostáva zo stĺpcových vektorov nad množinou $\{0, 1\}$ (to môžeme bez ujmy na všeobecnosti predpokladať; konkrétny tvar prvkov množiny K nie je nikde vo výpočte využívaný):

$$\begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix} \cdots \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix}$$

.

Podmnožinám množiny K potom priradíme vektory, ktoré sú súčtom vektorov prislúchajúcich jednotlivým prvkom a vyberme si fixné písmenko $a \in \Sigma$. δ-funkciu (alebo presnejšie zobrazenie $\vdash$) pre toto jedno písmeno potom môžeme reprezentovať ako vhodnú operáciu na takýchto vektoroch – konkrétne ako násobenie booleovských matíc (resp. matice a vektoru). Násobenie booleovských matíc je definované rovnako, ako štandardné násobenie. Táto operácia je definovaná rovnako, ako obyčajné násobenie matíc, akurát namiesto operácie násobenia dvoch prvkov matice je tu booleovská operácia $\wedge$, sčítaniu zodpovedá operácia $\vee$.

Celý výpočet takéhoto automatu si možno teda predstaviť nasledovne:

1. Automat si v pamäti uloží vektor zodpovedajúci stavu q_0.
2. V každom kroku prečíta písmenko zo vstupu a podľa neho vynásobí aktuálny obsah pamäte maticou prislúchajúcou danému písmenku.
3. Na konci výpočtu prečíta obsah pamäte a podľa neho zistí, či má dať odpoveď $\top$ alebo $\bot$.

Efektívne teda automat nepotrebuje obsah pamäte čítať počas výpočtu; stačí mu schopnosť realizovať príslušnú operáciu násobenia matíc. Až na konci výpočtu je potrebné obsah pamäte skutočne prečítať. Aj keď klasicky je tento rozdiel zanedbateľný, v kvantovom prípade to tak nebude.

Automatu z predchádzajúceho príkladu zodpovedajú matice:

$$M_a = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix} M_b = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

- Ďalším možným rozšírením je pridanie možnosti náhodného výberu nasledujúceho stavu, pričom budeme predpokladať, že z každého stavu automat niekam prejsť musí (t.j. suma pravdepodobností prechodov z daného stavu je rovná jednotke pre každé fixné písmenko a). Na rozdiel od priamočiareho nedeterminizmu, ktorý sme popísali, v tomto prípade nejde o "uhádnutie" správneho pokračovania; automat si nemôže vyberať, len hádže kockou a podľa výsledku sa presunie. Samozrejme, týmto znáhodnením sme opäť v situácii, ktorú si môžeme predstaviť tak, že automat môže byť vo viacerých stavoch. Na rozdiel od predchádzajúceho prípadu, tu majú jednotlivé možné stavy (t.j. obsahy pamäte) pridelené pravdepodobnosti z intervalu $\langle 0, 1 \rangle$, kdežto v prípade čistého nedeterminizmu mali iba dve možnosti – 0 alebo 1.

  Kroku výpočtu v tomto prípade tiež zodpovedá násobenie maticou; v tomto prípade ide o obyčajné násobenie matíc a matica obsahuje práve pravdepodobnosti jednotlivých predchodov medzi stavmi. Táto matica obsahuje iba nezáporné reálne čísla a má navyše tú vlastnosť, že zobrazuje vektor, ktorého súčet zložiek je rovný 1 opäť na vektor s touto vlastnosťou (t.j. v každom kroku automat v nejakom stave byť musí). Opäť, pri takomto výpočte nie je potrebné poznať skutočné hodnoty pravdepodobností, stačí ich vedieť upraviť vhodnou operáciou (v tomto prípade je to násobenie maticou).

  Otázkou je, ako v takomto prípade zaviesť odpoveď automatu – na konci je jeho pamäť reprezentovaná nejakou pravdepodobnostnou distribúciou, takže na ňu nemožno uplatniť klasický prístup "ak je v akceptačnom stave, tak slovo patrí do jazyka, inak nie". Možno však zaviesť pravdepodobnosť akceptovania a neakceptovania – položme $p_{acc} := \sum_{q \in F} p_q$, kde p_q označuje pravdepodobnosť prislúchajúcu stavu q. Podobne, $p_{rej} := \sum_{q \notin F} p_q$ je pravdepodobnosť zamietavej odpovede.

  Teraz máme mnoho možností, ako preložiť tieto pravdepodobnosti na odpovede automatu. Mohli by sme napríklad povedať, že slovo patrí do jazyka, pokiaľ dostaneme na konci výpočtu $p_{acc} = 1$. Rovnako dobre by sme mohli povedať, že slovo je akceptované (t.j. patrí do jazyka), ak $p_{acc} > 0$ (čo je ekvivalentné s požadovaním podmienky $p_{rej} = 1$ pre neakceptované slová). Často sa ešte používa akceptovanie s jednostrannou chybou – požaduje sa, aby pre slová patriace do jazyka bolo $p_{acc} = 1$ a pre ostatné slová aby bolo $p_{rej} > 1 - \epsilon$, kde ϵ je kladná reálna konštanta a aby na každom slove platilo $p_{acc} + p_{rej} = 1$ (v tomto prípade je však táto podmienka zbytočná, keďže všetky výpočty jednosmerného automatu sú konečné).

Je na tvorcovi automatu, aby zabezpečil, že pravdepodobnosti skutočne tieto podmienky spĺňať budú.

- Môžeme tiež automatu dovoliť, aby občas mohol vykonať nejaký krok výpočtu bez toho, aby písmenko zo vstupu spracoval. Keďže ε označuje prázdne slovo, takejto činnosti sa hovorí "prechod na ε". Ako sa však ukazuje, takéto rozšírenie opäť schopnosti automatu nezdokonalí. Dokonca môžeme konečnému automatu povoliť čítať vstup viackrát – či už tým, že sa bude po vstupe posúvať cyklicky, alebo tak, že bude môcť ísť aj opačným smerom – t.j. vrátiť sa na predchádzajúce písmenko/á (skryte to vlastne zahŕňa aj prechody na ε). Akonáhle však toto povolíme, musíme upraviť aj definíciu ukončenia výpočtu. Pri jednosmerných automatoch (bez prechodov na ε) totiž bolo toto ukončenie úplne prirodzené – akonáhle dospel automat na koniec vstupu, už sa nemal na základe čoho ďalej rozhodovať, a tak skončil s výpočtom. Pri dvojsmerných automatoch je situácia odlišná; podmienka na ukončenie tu musí byť iná.

- Inou možnosťou je umožnenie čítania niekoľkých (fixného počtu) písmeniek naraz. Ako je však zrejmé z maticovej reprezentácie nedeterministických aj pravdepodobnostných jednosmerných automatov, takáto úprava nemá nádej im pomôcť – viacnásobné násobenie maticami je ekvivalentné jednému vynásobeniu ich súčinom, a teda možno k danému rozšírenému automatu nájsť ekvivalentný[2] obyčajný automat.

Samozrejme, všetky tieto rozšírenia možno navzájom kombinovať, čím dostávame nepreberné množstvo kombinácií. Ukazuje sa však, že pri mnohých takýchto vylepšeniach sa sila automatu nezvýši. Napríklad, dvojsmerný nedeterministický automat, ktorý môže čítať naraz viac písmeniek[3] stále dokáže rozpoznávať len regulárne jazyky.

Na druhej strane, dvojsmerný pravdepodobnostný automat, ktorý pracuje v exponenciálnom čase už môže akceptovať aj neregulárne jazyky. Ako však bolo ukázané v článku [DS90], je to skutočne len za cenu exponenciálneho času. V časti 4.3 bude demonštrované, že z tohto pohľadu sú kvantové automaty skutočne silnejšie ako klasické – nielenže dokážu určitý takýto jazyk akceptovať v polynomiálnom čase, ale existuje dokonca aj jazyk, ktorý dokáže akceptovať istý kvantový konečný model aj napriek tomu, že to žiadny pravdepodobnostný dvojsmerný konečný automat nedokáže.

2.5 Zásobníkové a počítadlové automaty

Ďalším možným rozšírením konečných automatov, tentokrát už skutočne silnejším, je pridanie neobmedzenej zapisovateľnej pamäte. Najjednoduchším typom takejto pamäte je zásobník – t.j. dátová štruktúra podporujúca dve operácie – pridanie objektu na vrch zásobníka a vybratie objektu, ktorý je momentálne na vrchu (prípadne so signalizáciou, že zásobník je prázdny). Štandardne sa povoľuje, aby na každom mieste v zásobníku mohol

[2]Automaty považujeme za ekvivalentné, pokiaľ akceptujú ten istý jazyk.

[3]Nie, že má viac hláv!

byť jeden prvok istej pevne zvolenej konečnej množiny – takzvanej zásobníkovej/pracovnej abecedy.

Je zrejmé, že takýto model si už nezaslúži prívlastok "konečný" – tým, že sme mu dali k dispozícii neohraničenú zapisovateľnú pamäť, prestal mať charakter zariadenia, ktoré je nemenné a spracúva akýkoľvek veľký vstup nezávisle od jeho veľkosti. Je ľahké ukázať, že podmienka neohraničenosti je pre tieto automaty kritická – ak by sme povolili akúkoľvek konštantnú maximálnu veľkosť zásobníka, dostali by sme iba pamäť konečnú (a dokonca veľmi obmedzenú – nedá sa do nej pristupovať priamo, iba cez spomínané dve operácie), a tá sa dá simulovať aj v konečnom automate.

Na druhej strane, tento model je zaujímavý z hľadiska rozširovania – na rozdiel od konečných automatov, mnohé rozšírenia tohto modelu sú odlišné z hľadiska výpočtovej sily. Napríklad tak je rozdiel medzi deterministickými a nedeterministickými zásobníkovými automatmi; je tiež ľahko nahliadnuteľné, že dva zásobníky (spolu s ε-prechodmi) sú už ekvivalentné najsilnejšiemu tradičnému výpočtovému modelu – Turingovmu stroju, ...

Je zrejmé, že každý konečný automat si možno predstaviť ako zásobníkový automat, ktorý akurát zásobník nevyužíva. Teda trieda regulárnych jazykov je určite podtriedou triedy jazykov akceptovaných zásobníkovými automatmi. Tieto dve triedy sa však nerovnajú, inklúzia je vlastná, ako ukazuje napríklad jazyk $L = \{a^n b^n \mid n \geq 0\}$[4].

Formálnu definíciu zásobníkového automatu uvádzať nebudeme, presný princíp jeho fungovania nie je z hľadiska tejto práce podstatný.

Zrekapitulujme si teraz uzáverové vlastnosti zásobníkových automatov:

Definícia 14 *Trieda jazykov akceptovaných* ***nedeterministickými*** *zásobníkovými automatmi sa nazýva* ***trieda bezkontextových jazykov*** *a označuje sa $\mathcal{L}_{CF}$ (z anglického "Context-Free").*

Trieda $\mathcal{L}_{CF}$ je uzavretá na operácie zjednotenia, zreťazenia a inverzného homomorfizmu, no nie je uzavretá na operáciu prieniku (a teda ani komplementu). Jej ne-uzavretosť na prienik demonštrujú jazyky $L_1 = \{a^n b^n c^m \mid n, m \geq 0\}$ a $L_2 = \{a^m b^n c^n \mid n, m \geq 0\}$, ktorých prienikom je jazyk $L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 0\}$. Teda platí ostrá inklúzia

$$\mathcal{R} \subset \mathcal{L}_{CF}.$$

Špeciálnym druhom zásobníkových automatov sú takzvané počítadlové automaty. Namiesto zásobníka majú jednu číselnú premennú, ktorej hodnotu môžu v jednom kroku zväčšiť či zmenšiť o jednotku (resp. dozvedieť sa, že premenná je rovná nule). Podobne ako pri zásobníkových automatoch, aj tu je dôležitá podmienka neohraničenosti obsahu premennej; bez nej sme stále na úrovni konečného automatu. Počítadlový automat dokáže rozoznávať napríklad jazyky L_1, či L_2. Obdobne ako pri zásobníkoch, dá sa ukázať, že automat s viacerými počítadlami je už silný ako Turingov stroj.

[4] Dôkaz, že ho nie je možné akceptovať konečným automatom, sa dá spraviť napríklad pomocou tzv. pumpovacej lemy pre regulárne jazyky. Podrobnosti možno nájsť vo väčšine publikácií venovaných formálnym jazykom.

2.6 Časová náročnosť

Zatiaľ vieme porovnávať jednotlivé modely podľa ich výpočtovej sily – čiže podľa toho, *čo* dokážu rozpoznávať. Na druhej strane, nevšímali sme si *ako rýchlo* dokážu ten či onen jazyk rozpoznať. Z hľadiska praktickej využiteľnosti je často práve tento parameter zaujímavý. Aby sme vedeli porovnávať jednotlivé modely aj z takéhoto pohľadu, potrebujeme zaviesť vhodný spôsob merania veličiny "čas". Spravíme tak priamo v abstraktnom modeli:

Definícia 15 *Nech M je abstraktné zariadenie a $w \in I_M$ jeho vstup. Ak existuje ukončený výpočet dĺžky n zariadenia M na slove w, tak hovoríme, že výpočet M na vstupe w **trval** n **krokov** a označujeme to $t_M(w) = n$. Ak výpočet M na w neskončí, kladieme $t_M(n) = \infty$.*

Definícia 16 *Nech M je abstraktné zariadenie a $S \subseteq I_M$ je akákoľvek množina. Potom položíme $t_M(S) = \max_{w \in S} t_M(w)$ a hovoríme, že M **rozpoznáva množinu** S **v čase** $t_M(S)$. Pokiaľ Σ je abeceda a $I_M = \Sigma^*$, hovoríme, že M rozpoznáva jazyk S v čase $t_M(S)$.*

Napríklad, podľa našich definícií výpočet jednosmerného konečného automatu (rovnako ako aj jeho nedeterministickej či pravdepodobnostnej verzie) na slove dĺžky $|w|$ trvá $|w|$ krokov. Teda čas výpočtu je lineárny od dĺžky vstupu. Situácia je výrazne odlišná napríklad pri dvojsmerných nedeterministických konečných automatoch – pri nich veľa závisí od výberu ukončujúcich konfigurácií.

Kapitola 3

Základné kvantové pojmy

V tejto kapitole si ozrejmíme základné pojmy z teórie kvantových výpočtov a všimneme si rozdiely oproti výpočtom klasickým. Niektoré časti výkladu sú motivované knihou [NC00].

3.1 Kvantový stav

Majme akýkoľvek klasický systém obsahujúci informácie (môže to byť bajt v pamäti počítača, stránka v zápisníku, ...) Predpokladajme, že daný objekt môže poňať iba konečne veľa rôznych "obsahov" – napríklad jeden bajt môže nadobudnúť len 256 rozličných hodnôt. Týmto možným hodnotám či obsahom budeme hovoriť **stavy** systému.

Klasické systémy (a ich stavy) majú niekoľko užitočných vlastností, ktoré si budeme všímať aj pri ich kvantových analógiach. Preto si ich najprv zosumarizujme:

1. Všetky stavy klasického systému sú navzájom (dokonale) odlíšiteľné.
2. Je možné získať kompletnú informáciu o akomkoľvek stave – a teda aj vyrobiť jeho presnú kópiu.
3. Ak chceme poznať stav systému pozostávajúceho z dvoch podsystémov, stačí poznať stavy podsystémov.

Ako sa ukáže, ani jedna z týchto vlastností nie je dokonale splnená pri kvantových stavoch. Podrobnejšie si to rozoberieme v ďalšom.

Jeden z postulátov kvantovej mechaniky hovorí, že stavy každého kvantového systému sú reprezentovateľné ako prvky určitého vektorového priestoru (takzvaného unitárneho priestoru) nad poľom komplexných čísel. Odteraz preto nebudeme rozlišovať medzi systémom a jeho reprezentáciou v unitárnom priestore – napríklad budeme hovoriť o systéme A a súčasne o unitárnom priestore A.

Okrem štandardných operácií – sčítania vektorov a násobenia vektoru skalárom – je v tomto priestore definovaná ešte komplexná binárna funkcia nazývaná skalárny súčin[1],

[1]Niekedy sa označuje aj ako *vnútorný súčin*

ktorý spĺňa niekoľko podmienok. Takémuto priestoru sa hovorí **unitárny priestor**; niektorí autori používajú aj názov *Hilbertov priestor*[2]. Presnú definíciu unitárneho priestoru možno nájsť napríklad v [NC00] (sekcia 2.1), či [LG68].

Kanonickým príkladom komplexného unitárneho priestoru je priestor $\mathbb{C}^n$. Jeho prvky možno reprezentovať ako komplexné n-zložkové stĺpcové vektory. Sčítanie vektorov je priamočiare – sčítavame ich po zložkách, rovnako násobenie vektoru skalárom (t.j. komplexným číslom) je po-zložkové. Akonáhle máme tieto dve operácie, môžeme zaviesť pojem lineárnej kombinácie vektorov:

Definícia 17 *Nech $v_1, \ldots, v_n$ sú vektory, $c_1, \ldots, c_n$ komplexné čísla. Potom vektor $c_1v_1 + \ldots + c_nv_n$ nazývame lineárnou kombináciou vektorov $v_1, \ldots, v_n$ s koeficientami $c_1, \ldots, c_n$.*

Je zrejmé, že niektoré vektory sa dajú vyjadrovať pomocou iných (napríklad, $(1 + i, -2i) = (1 + 3i, 0) - 2(i, i)$). Toto motivuje k definícii pojmu *generujúcej množiny*:

Definícia 18 *Hovoríme, že množina M je **generujúcou množinou** priestoru H, ak každý vektor z H možno napísať ako lineárnu kombináciu vektorov z množiny M. Ak má priestor aspoň jednu konečnú generujúcu množinu, nazýva sa **konečnorozmerný**.*

V ďalšom výklade si budeme všímať iba konečnorozmerné unitárne priestory. Zo všetkých generujúcich množín si budeme všímať iba tie "najmenšie" – také, čo majú čo najmenej prvkov. Je zrejmé, že v takejto množine nemôže byť jeden vektor lineárnou kombináciou ostatných – inak by sme jeho vylúčením zachovali vlastnosť generovania celého priestoru, a to by bolo v spore s "najmenšosťou" množiny. Takéto množiny sa nazývajú **bázy**. Dá sa ukázať, že všetky bázy majú rovnako veľa prvkov, tomuto počtu sa hovorí **dimenzia** priestoru.

Napríklad v priestore $\mathbb{C}^3$ je bázou množina vektorov

$$\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix},$$

keďže každý vektor možno napísať v tvare ich lineárnej kombinácie. Skutočne, každý vektor v takomto $\mathbb{C}^3$ má tvar

$$\begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix},$$

a teda je rovný

$$\alpha_1 \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + \alpha_2 \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + \alpha_3 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.$$

[2]Tento názov sa však v niektorej literatúre používa výhradne na označenie nekonečnorozmerného úplného unitárneho priestoru, preto sa mu budeme snažiť vyhýbať.

Pravdaže, táto báza nie je jediná v tomto priestore. Rovnako dobrá je napríklad báza

$$\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix},$$

Okrem popísaných operácií, spoločných pre všetky vektorové priestory, obsahuje unitárny priestor operáciu skalárneho súčinu.

Poznámka 1 *V knižke [NC00] sa od skalárneho súčinu požaduje pravá lineárnosť (lineárnosť v druhom argumente), v [LG68] je to lineárnosť v prvom argumente. Aj keď je tento rozdiel veľmi malý, implikuje, že si treba pri preberaní výsledkov z iných zdrojov dávať pozor na poradie používané v tom-ktorom zdroji. V ďalšom texte sa budeme držať definície z [NC00].*

Definícia 19 *Nech H je komplexný vektorový priestor. Zobrazenie $< \cdot, \cdot >: H \times H \to \mathbb{C}$ sa nazýva* ***skalárny súčin****, ak spĺňa nasledovné podmienky:*

1. *Lineárnosť v druhom argumente, t.j. rovnosť $\langle u, \sum \alpha_i v_i \rangle = \sum \alpha_i \langle u, v_i \rangle$*
2. *Konjugovaná symetria – $\langle u, v \rangle = \langle v, u \rangle^*$ (kde * označuje číslo komplexne združené)*
3. *Pozitívna definitnosť $\langle v, v \rangle \geq 0$ (t.j. je to nezáporné reálne číslo).*

Príkladom skalárneho súčinu zavedeného v priestore $\mathbb{C}^n$ je

$$\left\langle \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix}, \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} \right\rangle = \sum_{i=1}^{n} u_i^* v_i.$$

Pomocou skalárneho súčinu sa dá zaviesť pojem **dĺžky vektora** u, ktorá sa značí $||u||$ ako $||u|| = \sqrt{\langle u, u \rangle}$ (vďaka tretej vlastnosti skalárneho súčinu táto odmocnina existuje, je reálna a nezáporná). Okrem toho sa zavádza aj pojem *ortogonality* dvoch vektorov – vektory sú **ortogonálne** pokiaľ ich skalárny súčin je nulový.

Z báz sú najzaujímavejšie tie, ktoré spĺňajú podminenku *ortonormálnosti* – báza je ortonormálna, pokiaľ je tvorená ortogonálnymi vektormi, ktorých dĺžky sú rovné jednej. Prvá z horeuvedených báz je ortonormálna, zatiaľčo druhá nie je (nespĺňa ani podmienku normovanosti, ani ortogonality). O spôsobe hľadania takejto bázy a o jej vlastnostiach sa možno viac dozvedieť v [LG68], pod názvom Gram-Schmidtova ortogonalizačný proces. Ak budeme odteraz hovoriť o báze, budeme automaticky predpokladať, že ide o ortonormálnu bázu.

Už spomínaný postulát kvantovej mechaniky hovorí o čosi viac, ako sme zatiaľ povedali. Stavom kvantového systému totiž nemôže byť ľubovoľný vektor unitárneho priestoru, iba vektor normovaný (t.j. jeho veľkosť je rovná jednej). Naopak, každý normovaný vektor môže byť stavom systému.

Špeciálne teda platí, že ak systém môže byť v dvoch stavoch, tak môže byť aj v akejkoľvek ich lineárnej kombinácii s koeficientami α, β, pokiaľ tieto spĺňajú $|\alpha|^2 + |\beta|^2 = 1$. Takému stavu sa hovorí *superpozícia* týchto dvoch stavov.

Je tiež ľahké vidieť, že ak $u_1, \ldots, u_n$ je (ortonormálna!) báza, každý prípustný stav má tvar $\sum_{i=1}^{n} \alpha_i u_i$.

3.2 Kvantový bit, trit, dit

Základná jednotka informácie vo väčšine klasických počítačov je (jeden) *bit*. Ak abstrahujeme od jeho fyzickej realizácie (úroveň napätia v obvode, dierka na diernom štítku, ...), bit je abstraktný "objekt", ktorý môže nadobúdať jednu z dvoch možných hodnôt – 0 alebo 1. Z toho vznikol aj jeho názov – "binary digit".

Jeho kvantovou analógiou je kvantový bit, skrátene **qubit**. Všeobecne, qubit je akýkoľvek kvantový systém s dvoma základnými stavmi. Ak má systém tri základné stavy, nazývame ho *qutrit*; pre viac-stavové systémy sa neujalo špeciálne označenie, spoločne sa označujú *qudit*[3].

3.3 Notácia

Na označovanie stavov kvantových systémov bol zavedený univerzálny a veľmi prehľadný spôsob zápisu – tzv. Diracova bra-ket notácia[4]. V nej sa obyčajný (čistý) stav nazvaný ϕ označuje $|\phi\rangle$ a nazýva sa *ket*-vektor. Je dôležité uvedomiť si, že ϕ je iba *označenie* stavu, rovnako dobre by sme mohli napísať $|\xi\rangle$ či $|1011\rangle$. Špeciálne je dôležité uvedomiť si, že stav $|0\rangle$ *nezodpovedá* nulovému vektoru, 0 je iba jeho názov.

Skalárny súčin vektorov $|\phi\rangle$ a $|\psi\rangle$ sa tu označuje $\langle\phi|\psi\rangle$. Pomocou neho možno ku každému ket-vektoru $|\phi\rangle$ priradiť lineárne zobrazenie – jeho duál, takzvaný bra-vektor – označované $\langle\phi|$ a definované rovnosťou:

$$\langle\phi| \left(|\psi\rangle\right) := \langle\phi|\psi\rangle \,.$$

V algebre sa hovorí, že bra-vektory tvoria priestor duálny k priestoru ket-vektorov. Analogickou konštrukciou možno k danému bra-vektoru priradiť ket-vektor, dokonca to bude ten istý, z ktorého príslušný bra-vektor dostaneme horeuvedenou konštrukciou (jeho *duál*). Zvyčajne budeme zátvorky okolo argumentu bra-operátoru vynechávať, čím sa ukazuje prvá časť pohodlnosti takéhoto zápisu – symbol $\langle\phi|\psi\rangle$ by sme mohli chápať v dvoch rôznych významoch – buď ako skalárny súčin vektorov $|\phi\rangle$ a $|\psi\rangle$ alebo ako aplikáciu operátora $\langle\phi|$ na vektor $|\psi\rangle$. Vďaka horeuvedenej definícii je to jedno.

Ket- a bra- vektory je možné reprezentovať aj vo veľmi elegantnej a veľmi jednoduchej podobe. Ako sme už spomínali, každý konečnorozmerný (n-rozmerný) unitárny priestor je totiž izomorfný s priestorom $\mathbb{C}^n$ (plynie to z existenie ortonormálnej bázy). Bez ujmy na

[3] Toto pomenovanie vzniklo z anglického "quantum digit", podobne, ako vznikol "bit" z "binary digit".

[4] z angl. bra-c-ket; význam bude vysvetlený zanedlho

všeobecnosti preto môžeme (a budeme) predpokladať, že sa všetko odohráva v priestore $\mathbb{C}^n$ (pre vhodné n). Ket-vektory sú v takomto priestore reprezentované stĺpcovými komplexnými vektormi s n zložkami a zodpovedajúci bra-vektor k danému ket-vektoru dostaneme jeho transponovaním a komplexnou konjugáciou – takzvanou *Hermitovskou konjugáciou.*

Namiesto ket- a bra- vektorov teda možno hovoriť o stĺpcových a riadkových komplexných vektoroch toho istého rozmeru. Napríklad, najvšeobecnejší možný qubit (resp. jemu zodpovedajúci ket-vektor) má tvar

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

kde $|\alpha|^2 + |\beta|^2 = 1$. Podobne, najvšeobecnejší možný bra-vektor je

$$\begin{pmatrix} \alpha & \beta \end{pmatrix}.$$

Tieto dva vektory nie sú navzájom duálne; na to, aby boli, museli by byť ešte zložky bra-vektora nahradené komplexne združenými číslami.

Skúsme si teraz predstaviť, že by sme ket-vektor $|\phi\rangle$ a bra-vektor $\langle\psi|$ vynásobili v opačnom poradí, ako sme to robili doteraz – spravíme súčin $|\phi\rangle\langle\psi|$. Na prvý pohľad to nedáva zmysel, no v maticovej reprezentácii je jednoduché vidieť, že výsledkom bude matica – a teda operátor! Vďaka asociativite násobenia matíc je ľahké vidieť, že jeho správanie je možné popísať nasledovne (druhá rovnosť platí, lebo skalárny súčin je iba číslo, možno ho preto písať na obe strany vektora):

$$|\phi\rangle\langle\psi|\,(|x\rangle) = |\phi\rangle\,(\langle\psi|x\rangle) = \langle\psi|x\rangle\,|\phi\rangle\,.$$

Ak tento vzťah použijeme ako definíciu operátora $|\phi\rangle\langle\psi|$, opäť sa raz ukáže pohodlnosť Diracovej notácie – dve pol-zátvorky (bra- a ket-) môžeme spojiť do celej zátvorky (bra(c)ket).

Hermitovskú konjugáciu môžeme rovnako, ako v prípade, keď sme vytvárali bra-vektor z ket-vektoru, použiť aj na ľubovoľný operátor (resp. na jeho maticovú reprezentáciu) M. Dostaneme tým jeho Hermitovský duál, označovaný $M^\dagger$. Hermitovská konjugácia má niekoľko zrejmých vlastností – napríklad $(AB)^\dagger = B^\dagger A^\dagger$. Viac o ich vlastnostiach možno nájsť napríklad v [LG68] alebo [NC00].

3.4 Zložený systém

Ďalší z postulátov kvantovej mechaniky hovorí, že ak máme dva systémy A a B, ktoré majú bázy $\{|\alpha_i\rangle\}_{i\in I}$ resp. $\{|\beta_j\rangle\}_{j\in J}$, tak systém zložený z týchto dvoch podsystémov má množinu bázových stavov $\{|\alpha_i, \beta_j\rangle\}_{i\in I, j\in J}$. Formálne tento fakt zapisujeme tak, že povieme, že priestor zodpovedajúci zloženému systému A a B je **tenzorovým súčinom** priestorov zodpovedajúcich systému A a systému B a značíme ho $A \otimes B$. Vektory $|\alpha_i, \beta_j\rangle$ označujeme aj $|\alpha_i\rangle \otimes |beta_j\rangle$.

Formálne má tenzorový súčin napríklad nasledovné dve vlastnosti:

1. Pre ľubovoľný skalár α platí $\alpha(|x\rangle \otimes |y\rangle) = (\alpha\,|x\rangle) \otimes |y\rangle = |x\rangle \otimes (\alpha\,|y\rangle) =$
2. Lineárnosť v oboch zložkách $|x\rangle \otimes (|y_1\rangle + |y_2\rangle) = (|x\rangle \otimes |y_1\rangle) + (|x\rangle \otimes |y_2\rangle)$ a $(|x_1\rangle + |x_2\rangle) \otimes (|y\rangle) = (|x_1\rangle \otimes |y\rangle) + (|x_2\rangle \otimes |y\rangle)$

Ak máme v systéme A operátor X a v systéme B operátor Y, možno zaviesť aj pojem tenzorového súčinu operátorov – súčin $X \otimes Y$ je operátor, ktorý stavu $|\alpha, \beta\rangle$ priradí stav $X\,|\alpha\rangle\,, Y\,|\beta\rangle$. Špeciálne môžeme takto zaviesť v zloženom priestore $A \otimes B$ skalárny súčin nasledovne:

$$\left\langle \sum_{i,j} c_{i,j}\,|\alpha_i, beta_j\rangle \mid \sum_{i,j} d_{i,j}\,|\alpha_i, beta_j\rangle \right\rangle := \sum c^*_{ij} d_{ij}$$

Všimnime si to teraz celé z pohľadu komplexných vektorov a ich maticovej reprezentácie. Pokiaľ fixujeme bázu v nejakom priestore dimenzie k, každému lineárnemu operátoru zodpovedá komplexná matica (typu $k \times k$) a naopak, každej komplexnej matici zodpovedá operátor.

Predstavme si, že máme priestor A dimenzie m a priestor B dimenzie n. Ako sme už povedali, bázu priestoru A aj B môžeme reprezentovať akostĺpcové vektory. Otázkou je, ako vhodne reprezentovať prvky bázy ich tenzorového súčinu (ktorý by mal mať dimenziu mn)?

Jedna z možných reprezentácii využíva *Kroneckerov súčin* matíc a jeho definícia znie:

Definícia 20 *Nech A je matica $m \times n$, B je matica $p \times q$. Potom* ***Kroneckerov súčin*** *matíc A a B je:*

$$A \otimes B = \overbrace{\begin{pmatrix} A_{11}B & A_{12}B & \dots & A_{1n}B \\ A_{21}B & A_{22}B & \dots & A_{2n}B \\ \vdots & \vdots & \vdots & \vdots \\ A_{m1}B & A_{m2}B & \dots & A_{mn}B \end{pmatrix}}^{nq}$$

Napríklad tak máme

$$\begin{pmatrix} 1 \\ 2 \end{pmatrix} \otimes \begin{pmatrix} 3 \\ 4 \end{pmatrix} = \begin{pmatrix} 1.3 \\ 1.4 \\ 2.3 \\ 2.4 \end{pmatrix} = \begin{pmatrix} 3 \\ 4 \\ 6 \\ 8 \end{pmatrix}$$

3.5 Kvantový register

Tak, ako klasický register v počítači je systém zložený z niekoľkých klasických bitov, rovnako je aj kvantový register systém zložený z niekoľkých qubitov. Tam sa však podobnosť končí…

Tak, ako v klasickom registri hodnoty často zapisujeme v binárnej podobe (napríklad 0010, alebo 1101), v kvantovom registri podobne označujeme bázu – $|0010\rangle$ a $|1101\rangle$[5]. Napríklad, dvojqubitový kvantový register môže byť v stave $\frac{1}{5}(3\,|01\rangle + 4i\,|10\rangle)$ (zlomok $\frac{1}{5}$ je potrebný kvôli normovanosti).

Predstavme si teraz veľmi jednoduchý kvantový register, pozostávajúci len z dvoch qubitov. Z predchádzajúcich sekcií vieme, že všeobecný stav takéhoto registra je vyjadriteľný v tvare:

$\alpha_{00}\,|0,0\rangle + \alpha_{01}\,|0,1\rangle + \alpha_{10}\,|1,0\rangle + \alpha_{11}\,|1,1\rangle$, kde $|\alpha_{00}|^2 + |\alpha_{01}|^2 + |\alpha_{10}|^2 + |\alpha_{11}|^2 = 1$. Príkladom takéhoto stavu môže byť $\frac{1}{2}(|0,0\rangle + |0,1\rangle + |1,0\rangle + |1,1\rangle)$. Tento stav sa dá rovnako dobre napísať v tvare $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. Inak povedané, jeho stav sa dá vyjadriť pomocou stavu jedného qubitu a druhého qubitu. Nič zvláštne, v klasickom svete je to dokonca úplne očakávateľné...

Čo však so stavom $\frac{1}{\sqrt{2}}(|0,0\rangle + |1,1\rangle)$? Ako zachvíľku ukážeme, akokoľvek sa budeme snažiť, tento stav sa nám v podobnej podobe vyjadriť nepodarí. Teda dva qubity môžu byť spolu akýmsi spôsobom **previazané**[6] a to dokonca tak, že dohromady tvoria systém, na ktorého popis nám nestačí informácia o jednej a druhej zložke samostatne. V klasickom systéme takáto situácia nastať nemôže.

No a teraz ten sľúbený dôkaz:

Veta 1 *Neexistujú také stavy* $|\phi\rangle$ *a* $|\psi\rangle$*, že* $|\phi\rangle \otimes |\psi\rangle = \frac{1}{\sqrt{2}}(|0,0\rangle + |1,1\rangle)$.

Dôkaz Predpokladajme, že také stavy existujú, potom sa dajú napísať ako

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

a

$$|\psi\rangle = \gamma\,|0\rangle + \delta\,|1\rangle$$

a musí platiť $\alpha\gamma = \beta\delta = 1$, $\alpha\delta = \beta\gamma = 0$. Potom však máme $0 = 0.0 = \alpha\delta\beta\gamma = \alpha\gamma\beta\delta = 1.1 = 1$, čo je spor.

Kvantový register je teda o čosi zložitejší, ako jeho klasický náprotivok.

3.6 Zápis – evolúcia kvantového stavu

Ako sme povedali, každý kvantový stav má tvar $\sum_i \alpha_i\,|\phi_i\rangle$, kde $\{|\phi_i\rangle\}_i$ je báza príslušného priestoru. Komplexné čísla α_i navyše spĺňajú podmienku $\sum_i |\alpha_i|^2 = 1$. Každá operácia, ktorá má transformovať kvantové stavy na kvantové stavy (nad tou istou bázou) musí preto zachovávať normovanosť vektorov. Ukážeme, že táto podmienka je pre lineárne zobrazenie A ekvivalentná podmienke $A^\dagger A = I$, kde I je jednotková matica. Dôkaz spravíme pre matice 2×2.

[5]Ešte raz je treba upozorniť, že stav $|0\rangle$ *neoznačuje* nulový vektor.

[6]Z anglického "entangled".

Veta 2 *Nech A je matica 2×2. Potom pre A platí $||u|| = 1 \Rightarrow ||Au|| = 1$ práve vtedy, keď $A^\dagger A = I$.*

Dôkaz Najvšeobecnejšia matica 2×2 má tvar

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}.$$

Aplikujme ju postupne na vektory

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ i \end{pmatrix}.$$

Dostaneme vektory

$$\begin{pmatrix} b \\ d \end{pmatrix}, \begin{pmatrix} a \\ c \end{pmatrix}, \frac{1}{\sqrt{2}} \begin{pmatrix} a+b \\ c+d \end{pmatrix}, \frac{1}{\sqrt{2}} \begin{pmatrix} a+bi \\ c+di \end{pmatrix}.$$

Keďže vstupné vektory boli normované, aj výsledné vektory také musia byť. Teda platí:

$$\begin{aligned} bb^* + dd^* &= 1 \\ aa^* + cc^* &= 1 \\ aa^* + bb^* + cc^* + dd^* + a^*b + b^*a + c^*d + d^*c &= 2 \\ aa^* + bb^* + cc^* + dd^* + a^*bi - b^*ai + c^*di - d^*ci &= 2 \end{aligned}$$

Po dosadení z prvej a druhej rovnice do tretej a štvrtej dostávame

$$\begin{aligned} a^*b + b^*a + c^*d + d^*c &= 0 \\ a^*b - b^*a + c^*d - d^*c &= 0 \end{aligned}$$

skadiaľ potom

$$\begin{aligned} a^*b + c^*d &= 0 \\ b^*a + d^*c &= 0 \end{aligned}$$

Vypočítajme teraz súčin $A^\dagger A$:

$$A^\dagger A = \begin{pmatrix} a^* & c^* \\ b^* & d^* \end{pmatrix} \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} aa^* + cc^* & a^*b + c^*d \\ b^*a + d^*c & bb^* + dd^* \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Dôkaz opačným smerom je možné spraviť analogicky, stačí obrátiť tento postup.

Keďže inverzné zobrazenie k unitárnemu zobrazeniu je opäť unitárne, ku každej transformácii kvantového stavu musí existovať transformácia inverzná. Z pohľadu dlhodobejšej evolúcie kvantového systému to znamená, že systém si "musí" pri evolúcii niesť "so sebou" informáciu o tom, ako sa vrátiť naspäť.

Z hľadiska kvantových výpočtov by to znamenalo, že ak by jediný povolený krok výpočtu bol aplikácia niektorej unitárnej operácie, takýto systém by zákonite musel byť reverzibilný. Ak chceme túto podmienku porušiť, musíme povoliť aj inú operáciu – a s jednou takou operáciou sa zoznámime v nasledujúcej časti.

3.7 Čítanie – meranie

Zatiaľ vieme, akým spôsobom môžeme meniť obsah kvantového registra. Otázkou je, nakoľko tento stav vieme čítať? Keby sme totiž vedeli priamo zistiť koeficienty α a β v stave $\alpha|0\rangle + \beta|1\rangle$, vedeli by sme v jednom qubite uložiť potenciálne nekonečnú informáciu (keďže α a β sú ľubovoľné komplexné čísla), čo však odporuje tradičným predstavám o ukladaní informácie.

Žiaľ, tu nás kvantová mechanika príliš nepustí – zisťovať stav kvantového systému možno len *meraním*, a merania majú určité nepríjemné obmedzenia. Prv než sa s nimi zoznámime, zavedieme si niekoľko pojmov:

Definícia 21 *Lineárny operátor P sa nazýva* ***projektor****, ak je rovný svojmu Hermitovskému duálu sa a má tvar $P = \sum_i |i\rangle\langle i|$ pre niektorú množinu ortonormálnych vektorov $\{|i\rangle\}$.*

Je ľahké vidieť, že pre projektory platí rovnosť $P^2 = P$:

$$\begin{aligned} P^2(|x\rangle) = (\sum_i |i\rangle\langle i|)(\sum_j |j\rangle\langle j|)\,|x\rangle &= \\ &= (\sum_i |i\rangle\langle i|)(\sum_j |j\rangle\langle j|x\rangle) \\ &= \sum_j \langle j|x\rangle (\sum_i |i\rangle\langle i|j\rangle) \\ &= \sum_j \langle j|x\rangle (\sum_{i,i=j} |i\rangle) \\ &= \sum_j \langle j|x\rangle\,|j\rangle \\ &= P(j) \end{aligned}$$

Definícia 22 ***Ortogonálnym (alebo von Neumannovským)*** *meraním nazývame množinu projektorov $\{P_i\}_{\{}i \in I\}$ takých, že pre všetky $j \neq i$ platí $P_iP_j = 0$. Okrem toho ešte musí platiť $\sum_{i\in I} P_i = Id$, kde Id je identický operátor. Uvedenej nožine operátorov sa hovorí* ***pozorovateľná***[7] *prípadne pozorovanie.*

Postuluje sa, že v prípade, že na kvantový stav $|\phi\rangle$ použijeme takéto meranie, udeje sa nasledovné:

1. Ako výsledok merania dostaneme index z množiny I – výsledok merania.
2. Pravdepodobnosť, že dostaneme práve výsledok k je rovná $||P_k|\phi\rangle||^2$

[7]Z anglického ”observable”.

3. Systém sa pretransformuje do stavu $\frac{1}{||P_j|\phi\rangle||}P_j|\phi\rangle$ – t.j. pri opakovanom meraní budeme dostávať tie isté výsledky, keďže $P_j^2 = P_j$ a $P_iP_j = 0$ ak $i \neq j$.

Inak povedané, operácia *čítania* kvantového stavu tento stav *zmení*! Na jednej strane sa toto správanie môže javiť ako strašné – veď takto si môžeme uprostred výpočtu niečo pokaziť!

Na druhej strane, práve toto správanie nám umožňuje robiť nereverzibilné – operácie napríklad, nastaviť daný qubit na určitú hodnotu. Stačí zrealizovať meranie, čím qubit dostaneme do jedného zo stavov, ktoré poznáme (keďže sme si pomocou nich vytvárali meranie) a potom aplikovať príslušnú unitárnu operáciu, ktorá stav prevedie do žiadanej podoby.

Inou možnou aplikáciou je generovanie náhodných hodnôt – ak vieme opakovane vytvoriť vhodný kvantový stav, môžeme pomocou merania generovať náhodné hodnoty s určenými pravdepodobnosťami.

Samozrejme, hlavným účelom merania je skutočne zistenie stavu systému; na pri výpočtových modeloch sa často meraním zisťuje, či automat už prišiel do akceptačného alebo zamietacieho stavu. Na ten účel sa vytvorí pozorovateľná, v ktorej jeden projektor zodpovedá akceptačným stavom, druhý zamietacím a tretí zvyšným.

Napríklad, ak máme kvantový stav $\frac{1}{2}|0\rangle + \frac{\sqrt{3}}{2}|1\rangle$ a zmeriame ho vzhľadom na pozorovateľnú $\{P_1, P_2\}$, kde $P_1 = |0\rangle\langle 0|$ a $P_2 = |1\rangle\langle 1|$, dostaneme výsledok 1 s pravdepodobnosťou $(\frac{1}{2})^2 = \frac{1}{4}$ a výsledok 2 s pravdepodobnosťou $(\frac{3}{2})^2 = \frac{3}{4}$. V prvom prípade bude nový stav qubitu rovný $|0\rangle$, v druhom $|1\rangle$.

Ako je zrejmé z tohto príkladu, pri špecifikovaní merania vlastne netreba určovať samotné projektory, stačia napríklad iba ket-vektory, ktoré ich tvoria. V tomto prípade by sme napríklad povedali, že P_1 meria stav $|0\rangle$ a P_2 stav $|1\rangle$.

O čosi zaujímavejší stav dosiahneme, keď skúsime zmerať starý-známy previazaný stav $|\phi\rangle = \frac{1}{\sqrt{2}}(|0,0\rangle + |1,1\rangle)$ vzhľadom na pozorovanie udané projektormi $P_1 = |0,0\rangle + |0,1\rangle$ a $P_2 = |1,0\rangle + |1,1\rangle$. Je ľahké vidieť, že dostaneme výsledok 1 alebo 2 s pravdepodobnosťou $\frac{1}{2}$. Zaujímavejší je však stav, v akom bude systém po takomto meraní – ak dostaneme výsledok 1, systém bude v stave $\frac{1}{||P_1|\phi\rangle||}P_1|\phi\rangle = |0,0\rangle$. Ak dostaneme výsledok 2, vyjde nám stav $\frac{1}{||P_2|\phi\rangle||}P_2|\phi\rangle = |1,1\rangle$. Použité meranie však fungovalo tak, že meralo hodnotu iba prvého qubitu (ako je ľahko vidieť z jeho zápisu), no napriek tomu sa zmenil súčasne aj druhý qubit. Teda pri previazaných stavoch môže zmeraním jednej časti systému dôjsť ku zmene úplne inej časti, a treba si na takéto stavy dávať špeciálny pozor.

Poznámka 2 *Existujú aj všeobecnejšie druhy meraní – napríklad POTM merania, no tie nie sú pre ďalší výklad potrebné. Viac o nich sa možno dozvedieť v [NC00], časti 2.2.3-2.2.6.*

Nepríjemným dôsledkom princípu fungovania meraní je tiež známa "no-cloning theorem" – veta o nemožnosti dokonalého klonovania kvantových stavov. Ako sme videli, meranie nám pri klonovaní príliš pomôcť nevie, keďže ono samotné stav takmer iste zmení (detailný dôkaz je však nad rámec tejto práce). Zostávajú nám preto evolučné operácie – t.j. operácie unitárne. Predpokladajme teda, že existuje unitárna operácia U taká, že

$U|0\rangle|x\rangle|y\rangle = |0\rangle|0\rangle|z_0\rangle$, $U|1\rangle|x\rangle|y\rangle = |1\rangle|1\rangle|z_1\rangle$, kde x je počiatočný obsah qubitu, v ktorom chceme vytvoriť kópiu stavu, y je počiatočné pomocné miesto a z_0 resp. z_1 sú koncové obsahy tohoto pomocného miesta.

Potom musí platiť aj $U\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)|x\rangle|y\rangle = \frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)|z\rangle$, kde, tak ako predtým $|z\rangle$ je výsledný obsah pomocného miesta. Problémom je lineárnosť zobrazenia U, ktorá hovorí, že dostaneme výsledok:

$U\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)|x\rangle|y\rangle = U\frac{1}{\sqrt{2}}|0\rangle|x\rangle|y\rangle + U\frac{1}{\sqrt{2}}|1\rangle|x\rangle|y\rangle = \frac{1}{\sqrt{2}}(|0\rangle|0\rangle|z_0\rangle + |1\rangle|1\rangle|z_1\rangle)$. Je ľahké vidieť, že tento stav nie je rovnaký, ako ten, ktorý mal vzniknúť – dokonca to, čo sme dostali, je, na rozdiel od požadovaného výsledku, previazané. Dokonalé kopírovanie kvantových stavov teda nie je možné. Na druhej strane, existujú metódy umožňujúce kopírovanie s istou nedokonalou úspešnosťou/chybou.

Kapitola 4

Konečné kvantové výpočtové modely

V tejto kapitole si všimneme dva zaujímavé kvantové výpočtové modely, ktoré vznikli ako zovšeobecnenie klasických konečných automatov, a pritom si zachovali vlastnosť konečnosti, ktorú rozoberieme v prvej časti. Ako zdroj informácií o iných modeloch môže poslúžiť článok [Gru00], diplomová práca [Pet98] a kniha [Gru99].

4.1 Konečnosť

Konečnosť budeme chápať ako fyzikálnu vlastnosť – aby bol konečný, systém by mal byť "zostrojiteľný" z konečného množstva surovín. Konečný automat napríklad konečný je, no taký zásobníkový automat už nie – jeho zásobník totiž môže rásť cez všetky medze v závislosti od toho, čo automat dostane ako vstup.

Zatiaľčo v prípade klasických modelov je k dispozícii surovín veľké množstvo, pri kvantových to zatiaľ neplatí – pokusy udržať i veľmi malé (v porovnaní s klascikým prípadom) počty kvantových stavov pohromade a izolovane od okolia zatiaľ zlyhávajú. Preto, je užitočné hľadať modely, ktoré sú menej náročné na využívanie kvantových fenoménov, no napriek tomu sú iné ako ich klasické náprotivky.

4.2 1QFA

Tento model je priamočiarym prekladom konečného automatu do kvantovej podoby. Zavedený bol v článku [KW97] ako špeciálny prípad dvojsmerného kvantového "konečného" automatu (2QFA), ktorý však nespĺňa našu podmienku konečnosti v pravom zmysle slova (môže byť v superpozícii nad všetkými písmenkami vstupu, a teda veľkosť jeho kvantovej časti by musela byť úmerná dĺžke vstup), preto sa ním podrobnejšie nezaoberáme.

Namiesto konečnej klasickej pamäte má 1QFA konečnú kvantovú pamäť. Postup výpočtu automatu je veľmi jednoduchý – prakticky totožný s klasickým konečným automatom. V jednom kroku prečíta písmenko a na jeho základe si upraví stav svojej pamäte. Keďže jeho pamäť je kvantová, táto úprava musí byť unitárna. Po každej takejto úprave sa vykoná meranie vzhľadom na pozorovateľnú, o ktorej sme už hovorili – pomocou nej sa rozhodne, či automat slovo zakceptuje, odmientne, alebo pokračuje ďalej v čítaní vstupu. Takémuto módu činnosti sa hovorí aj "many-measurements"; niektorí autori uvažovali aj o prípade, že by sa meranie uskutočňovalo len raz, a to na konci výpočtu ("measure-once").

V našom abstraktnom modeli sa dá zadefinovať napríklad nasledovne:

Definícia 23 *Nech Σ je abeceda, K je konečná neprázdna množina stavov, $\hat{K}$ je množina normovaných vektorov v $|K|$-rozmernom unitárnom priestore (označíme ich tak, že $K \subseteq \hat{K}$), $F \subseteq K$ je množina akceptačných stavov, $R \subseteq K$ množina zamietacích stavov, $q_0 \in K$ je počiatočný obsah pamäte a $\{V_a : \hat{K} \to \hat{K} \mid a \in \Sigma\}$ je množina prechodových unitárnych operácii pre jednotlivé písmenká. Budeme používať projektory $P_{acc} = \sum_{i \in F} |i\rangle \langle i|$, $P_{rej} = \sum_{i \in R} |i\rangle \langle i|$, $P_{non} = \sum_{i \in K-(R \cup F)} |i\rangle \langle i|$. Položme:*

1. $I := \Sigma^*$
2. $S := \hat{K} \times \Sigma^*$ *– prvá zložka reprezentuje obsah pamäte, druhá doposiaľ neprečítaný úsek vstupného slova*
3. $T := \hat{K} \times \{\varepsilon\}$ *– výpočet skončí, keď je celý vstup spracovaný*
4. $O := \{\top, \bot\}$
5. $i(w) := (|q_0\rangle, w)$
6. $(|\phi\rangle, a \cdot w) \vdash (|\psi\rangle, w, p_a, p_r)$, *ak* $a \in \Sigma$, $w \in \Sigma^*$, $p_a = ||P_{acc}|\phi\rangle||$, $p_r = ||P_{rej}|\phi\rangle||$ *a* $|\psi\rangle = \frac{1}{||P_{non}|\phi\rangle||} P_{non}|\phi\rangle$ *je výsledok merania vzhľadom na projektor* P_{non}.

Keďže takýto automat je zo svojej podstaty pravdepodobnostný, na akceptovanie jazyka je potrebné vziať niektorú z definícií pravdepodobnostného akceptovania. V tomto prípade je to ešte navyše skompilkované možnosťou, že automat môže na danom vstupe skončiť bez toho, aby vstup akceptoval alebo zamietol.

Dá sa však demonštrovať, že podmienka unitárnosti vývoja obsahu pamäte je omnoho tvrdšia, ako by sa mohlo zdať. Takýto model nedokáže akceptovať ani všetky regulárne jazyky (a pritom všetky jazyky, ktoré akceptuje, regulárne sú). Ukazuje sa teda, že priamočiara konverzia z klasického automatu na kvantový je v tomto prípade veľmi nevýhodná – kvantové fenomény nám skôr škodia ako pomáhajú.

Tento nedostatok sa snaží riešiť druhý model, ktorý spomenieme v nasledujúcej časti.

4.3 2QCFA

Model bol navrhnutý v článku [AW02] ako istý medzistupeň medzi 1QFA, ktoré sú príliš slabé a 2QFA, ktoré, vzhľadom na to, že veľkosť ich kvantového stavu je úmerná dĺžke vstupu, sú príliš "nepraktické".

Samotný automat je definovaný veľmi jednoducho – jeho pamäť pozostáva z dvoch častí – klasickej a kvantovej. V každom kroku môže urobiť jednu z dvoch vecí – podľa písmenka na vstupe buď upraviť svoju klasickú a kvantovú časť, alebo obsah kvantovej časti zmerať vzhľadom na určenú pozorovateľnú (a podľa výsledku merania upraviť obe svoje pamäte). Principiálne možno tieto dve aktivity spojiť do jednej – stačí merať vzhľadom na pozorovateľnú, ktorá obsahuje iba identický projektor. Teda môžeme predpokladať, že automat vždy spraví meranie a potom sa rozdhodne podľa obsahuje pásky a výsledku merania.

Vzhľadom na to, že merania sú pravdepodobnostné, aj akceptovanie musí byť pravdepodobnostné. V originálnom článku autori ukázali, že pokiaľ sa akceptovanie definuje ako akceptovanie s jednostrannou chybou – t.j. na každom vstupe automat skončí v akceptačnom alebo zamietacom stave s pravdepodobnosťou 1, pričom na slovách patriacich do jazyka je pravdepodobnosť akceptovania rovná 1 a na ostatných je pravdepodobnosť zamietnutia aspoň $1 - \epsilon$, tak tento model má veľmi zaujímavé výpočtové vlastnosti.

Čiastočne sformalizovaná definícia takéhoto automatu (originálni autori volili radšej neformálnejší prístup):

Definícia 24 *Nech Σ je abeceda, K, Q sú konečné neprázdne množiny stavov – klasických a kvantových, $\hat{Q}$ je množina normovaných vektorov v $|Q|$-rozmernom unitárnom priestore (kde $Q \subseteq \hat{Q}$), $F \subseteq K$ je množina akceptačných stavov, $R \subseteq K$ množina zamietacích stavov, $k_0 \in K, q_0 \in Q$ sú počiatočné obsahy pamätí, m je funkcia priraďujúca dvojici (aktuálny stav, čítané písmenko) meranie, ktoré sa má v príslušnom stave previesť, a $\delta : K \times \Sigma \times M \to K \times U \times \{-1, 0, 1\}$ je prechodová funkcia, ktorá priraďuje trojici (klasický stav, písmenko, výsledok merania) trojicu (nový klasický stav, unitárna operácia na $\hat{Q}$, posun automatu).*

Budeme používať projektory $P_{acc} = \sum_{i \in F} |i\rangle \langle i|$, $P_{rej} = \sum_{i \in R} |i\rangle \langle i|$, $P_{non} = \sum_{i \in K-(R \cup F)} |i\rangle \langle i|$.

Položme:

1. $I := \Sigma^*$
2. $S := K \times \hat{Q} \times (\triangle \Sigma^* \Box) \times \mathbb{N}$ *– prvá a druhá zložka reprezentujú obsahy pamätí, tretia vstup (spolu s okrajovými značkami $\triangle$ a $\Box$), štvrtá miesto, ktoré je automatom práve čítané.*
3. T *nebudeme zapisovať formálne*
4. $O := \{\top, \bot\}$
5. $i(w) := (k_0, |q_0\rangle, \triangle w \Box, 0)$ *kde*
6. $(k, |\phi\rangle, w, n) \vdash (k', |\psi\rangle, w, n', p_a, p_r)$, *ak* $w \in \Sigma^*$, $\delta(k, a, m(k, a)) = (k', U, d)$ *a* $|\psi\rangle = \frac{1}{||P_{non}|\theta\rangle}||P_{non}|\theta\rangle$, *kde* $|\theta\rangle = U|\phi\rangle$. $p_a = ||P_{acc}|\theta\rangle||$, $p_r = ||P_{rej}|\theta\rangle||$.

Je zrejmé, že takýto model rozpoznáva všetky jazyky, ktoré sú rozpoznávateľné dvojsmernými pravdepodobnostnými automatmi, keďže, ako sme už povedali, kvantový automat si "môže hodiť kockou". Podobne, ako sme vysvetľovali pri konečných pravdepodobnostných automatoch, výpočet automatu si možno predstaviť ako priamočiare počítanie s maticami. V tomto prípade však matice nemusia byť stochastické[1], ale unitárne. Unitárnosť je vlastnosť, ktorá v tomto prípade automatu výrazne pomôže – vďaka tomu, že v matici môžu byť záporné čísla, je možné dosiahnuť napríklad istú operáciu aj jej navrátenie späť – čo pri pravdepodobnostných automatoch možné nebolo.

Ukážeme dva príklady, ktoré sú zaujímavé z hľadiska (oba príklady sú spracované podľa [AW02]):

Prvým jazykom je jazyk palindrómov – $L_{pal} = \{w \in \{a, b\}^* \mid w = w^R\}$. Tento jazyk nie je možné akceptovať žiadnym dvojsmerným pravdepodobnostným automatom. Na druhej strane, existuje 2QCFA, ktorý tento jazyk akceptuje a jeho kvantová časť má veľkosť jediného qubitu! Základom činnosti 2QCFA akceptujúceho tento jazyk je fakt, že v grupe unitárnych operácií na jednom qubite existuje podgrupa izomorfná s dvojgenerátorovou voľnou grupou:

Definícia 25 *Voľná grupa s dvoma generátormi je grupa tvorená slovami nad abecedou $\{x, y, x^{-1}, y^{-1}\}$, pričom ak slovo obsahuje dvojicu xx^{-1}, $x^{-1}x$,yy^{-1},$y^{-1}y$, táto sa považuje za ekvivalentnú s ϵ. Grupovou operáciou je skladanie slov.*

Pre zjednodušenie však budeme uvažovať namiesto jedného qubitu jeden qutrit. Napríklad je možné vziať rotácie Príklady unitárnych operácií, ktoré tvoria grupu izomorfnú s voľnou dvojgenerátorovou grupou sú:

$$U_a = \begin{pmatrix} \frac{4}{5} & \frac{3}{5} & 0 \\ -\frac{3}{5} & \frac{4}{5} & 0 \\ 0 & 0 & 1 \end{pmatrix} U_b = \begin{pmatrix} \frac{4}{5} & 0 & \frac{3}{5} \\ 0 & 1 & 0 \\ -\frac{3}{5} & 0 & \frac{4}{5} \end{pmatrix}$$

Máme teda voľnú grupu s dvoma generátormi – U_a a U_b. Zadaný vstup zinterpretujeme tak, že ho prečítame dvakrát, prvýkrát pri každom písmenku a použijeme operáciu U_a a pri každom b operáciu U_b, pri druhom prechode používame U_a^{-1} a U_b^{-1}. Ak výsledok nebude ekvivalentný s neutrálnym prvkom grupy (t.j. s prázdnym slovom), slovo zamietneme, inak zostane potenciálnym kandidátom na akceptáciu.

Zjavne, ak $w = w^R$, každá operácia U_a^{-1} vyruší príslušnú U_a, ktorá bola aplikovaná pri prvom prechode, podobne je to s U_b a U_b^{-1}. Ak však $w \neq w^R$, niekedy nastane prípad, keď bola aplikovaná operácia U_a a za ňou U_b^{-1} (alebo U_b a za ňou U_a^{-1}). Takéto slovo však nemôže byť ekvivalentné s prázdnym slovom.

Problémom je, že pravdepodobnosť zamietnutia zlého slova je síce nenulová, no veľmi nízka (exponenciálne klesá s dĺžkou vstupného slova). Preto je potrebné tento postup mnohokrát opakovať. Na tento účel sa využíva trik, ktorý by sme mohli nazvať "úmyselné zdržovanie" – stroj vykoná postupnosť krokov, ktorej pravdepodobnosť úspechu je exponenciálne nízka a ak sa táto činnosť skončí úspechom, automat skončí s akceptujúcou odpoveďou. Príkladom takej činnosti je "hádzanie mincou" na každom písmenku vstupného slova a pokiaľ nepadne všade hlava, tak sa opakuje horeuvedené overovanie príslušnosti slova do jazyka.

[1] Teda spĺňajúce podmienku celkovej pravdepodobnosti rovnej jednotke

Inak povedané, stroj sa vlastne snaží hľadať spôsob, ako slovo zamietnuť, a iba ak sa mu to dostatočne dlho nepodarí nájsť nepodarí, dá kladnú odpoveď. Tento prístup je aj zdrojom problémov pri pokusoch o rozšírenie modelu 2QCFA o istý druh nedeterminizmu – priamočiary nedeterminizmus sa totiž snaží práve o čo najskoršie dosiahnutie akceptačného stavu.

Zaujímavým problémom je otázka, či je možné dosiahnuť aj väčšie ako exponenciálne "zdržovanie" a či by takéto zdržovanie malo prínos pre výpočtovú silu automatu. Na základe doterajšieho skúmania je odpoveď záporná.

Druhým jazykom je jazyk $L_{eq} = \{a^n b^n \mid n \geq 0\}$. Tento je síce pravdepodobnostným automatom akceptovateľný, no takýto automat potrebuje exponenciálny čas, zatiaľčo príslušnému 2QCFA stačí čas polynomiálny.

Automat akceptujúci tento jazyk je jednoduchší ako predchádzajúci – za každé písmenko a aplikuje na svoj stav rotáciu o uhol, ktorý generuje istú nekonečnú podgrupu grupy všetkých unitárnych operácií na jednom qubite. Podobne, za každé b realizuje operáciu k tejto rotácii inverznú. Po prejdení slova overí, či výsledný stav zodpovedá neutrálnemu prvku. Opäť, pravdepodobnosť zamietnutia je relatívne nízka, no nie exponenciálne nízka. Aj keď by bolo možné použiť rovnaký spôsob "zdržovania" pomocou hádzania mince, zbytočne by sa tým zväčšila doba výpočtu. Preto je v tomto prípade použitá technika známa ako "random-walk", pri ktorej sa na základe hodu mincou automat presúva doľava alebo doprava a ak príde až po koncové písmenko, tak akceptuje. Vďaka tejto technike je možné zdržať chod automatu polynomiálne – a tým dosiahnuť očakávaný polynomiálny čas behu.

Podobne ako predtým, aj na činnosť tohto automatu sa možno dívať tak, že vstupné slovo je popisom prvku v istej grupe a automat tento prvok priamo konštruuje (vďaka faktu, že príslušná grupa je podgrupou grupy unitárnych transformácii jeho stavového priestoru) a na konci konštrukcie porovnáva s jedným špecifickým prvkom (bez ujmy na všeobecnosti s neutrálnym prvkom). Keďže však kvantové stavy nie je možné dokonale porovnávať, ide len o pravdepodobnostný prístup – a teda tento postup treba viackrát opakovať.

Pokiaľ je nám známe, takýto pohľad zatiaľ nebol skúmaný

Pomocou štandardných konštrukcií možno ukázať, že trieda jazykov akceptovaná takýmito automatmi je uzavretá na prienik aj inverzný homomorfizmus. Konštrukciou podobnou tej, ktorú sme ukázali, vieme ukázať aj to, že takéto automaty dokážu akceptovať jazyky $\{a^n b^n c^m \mid n, m \geq 0\}$ a $\{a^n b^n c^m \mid n, m \geq 0\}$, ktorých prienikom je jazyk $\{a^n b^n c^n \mid n \geq 0\}$, ktorý nie je ani bezkontextový.

Na jednej strane teda takýto automat dokáže rozpoznávať jazyky, ktoré nie sú ani bezkontextové (a teda nie sú akceptovateľné ani klasickým zásobníkovým automatom, ktorý nie je, na rozdiel od tohto modelu, konečný), no na druhej strane sa zatiaľ zdá, že nedokáže akceptovať ani niektoré jazyky kontextové. Jednou takouto ukážkou je jazyk, v 4.3 označený ako $L_{balanced} = \{x \in \{(,)\}^* \mid$ uzátvorkovanie je korektné, t.j. ku každej ľavej zátovorke existuje prislúchajúca pravá $\}$.

Okrem toho, popísané konštrukcie vyžadovali veľmi vysokú mieru presnosti, a tak ich praktická využiteľnosť nie je príliš vysoká. Zaujímavé by však mohlo byť pokúsiť sa o rozšírenie tohto modelu o ďalšie možnosti – napríklad, pridaním určitej nekvantovej operácie (konkrétne porovnávanie dvoch kvantových stavov) sa zdá, že by bolo možné tento model povýšiť na úroveň klasických počítadlových automatov, použitím tejto operácie v automate M_{equ} by malo byť možné simulovať klasické počítadlo. Tým by sa však tento model dostal mimo kategórie "praktických" modelov, keďže jeho fyzická realizácia by priamo odporovala kvantovej mechanike.

Kapitola 5

Záver

V predkladanej diplomovej práci sme sa zaoberali klasickými a kvantovými modelmi, uviedli sme aj prípady, keď rozšírenie automatu (nahradenie klasickej pamäte kvantovou) spôsobilo rozšírenie jeho možností, no analogické rozšírenie ekvivalentného modelu mu naopak výpočtovú silu ubralo.

Problém úplnej charakterizácie výpočtovej sily rôznych modifikácii modelov 1QFA i 2QCFA zostáva i naďalej otvorený. Zaujímavé by mohlo byť aj zistenie, akú výpočtovú silu má 2QCFA, pokiaľ mu pridáme (nekvantovú) možnosť overenia, či jeho kvantová časť je v niektorom pevne zvolenom stave. Takýto model by, intuitívne, mal byť aspoň taký silný, ako dvojsmerný počítadlový automat. Otázkou tiež zostáva požadovaná presnosť realizácie kvantových operácií.

Množstvo výsledkov z tejto oblasti možno nájsť v práci [Gru00].

Počas písania práce bolo vyskúšaných viacero prístupov k problematike vysvetľovania teórie kvantových výpočtov na úrovni zrozumiteľnej študentovi informatiky. Viacero modelov, ktoré zjednocovali klasické a kvantové výpočty, sa ukázalo byť nevhodnými, či už z hľadiska názornosti alebo všeobecnosti. Model, ktorý sme zaviedli v tejto práci, je dostatočne univerzálny, aby ho bolo možné použiť na viacero známych kvantových aj klasických modelov, a pritom sa snaží byť čo najnázornejší, aby umožnil urobiť prechod od klasického ku kvantovému čo najjednoduchšie.

Iný prístup k problematike modelov kvantových výpočtov bol zvolený v [Gud00], kde sa však autor nesnaží o vysvetlenie základov kvantových výpočtov, ale o ich zjednotenie a vzájomné porovnanie.

Literatúra

[AW02] A.Ambainis and J. Watrous. Two-way finite automata with quantum and classical states. *Theoretical Computer Science*, 287(1), 2002.

[BB84] C. H. Bennett and G. Brassard. Public key distribution and coin tossing. In *Proceedings of IEEE International Conference on Computers, Sysetms and Signal Processing*, pages 175–179. IEEE, New York, Bangalore, India, 1984.

[DS90] C. Dword and L. Stockmeyer. A time-complexity gap for two-way probabilistic finite state automata. *SIAM Journal of Computing*, 19:1011–1023, 1990.

[Gru99] Jozef Gruska. *Quantum Computing*. McGraw-Hill, 1999.

[Gru00] Jozef Gruska. Descriptional complexity issues in quantum computing. *International Workshop on Descriptional Complexity of Automata, Grammas and Related Structures*, 2000.

[Gud00] Stanley Gudder. Basic properties of quantum automata. Technical report, University of Denver, 2000.

[KW97] A. Kondacs and J. Watrous. On the power of quantum finite state automata. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, pages 66–75, 1997.

[LG68] S.Mac Lane and G.Birkhoff. *Algebra*. Alfa, Bratislava, 1968.

[NC00] M. A. Nielsen and I. L. Chuang. *Quantum computation and quantum information*. Cambridge Univ. Press, Cambridge, 2000.

[Pet98] Pavel Petrovič. Kvantové automaty. Diplomová práca, MFF UK, Bratislava, 1998.

[Sho94] Peter Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings, 35^{th} Annual Symposium on Foundations of Computer Science*. IEEE Press, Los Alamitos, CA, 1994.