Pozwólcie, że wytłumaczę, jak to działa. Przypuśćmy, że rejestr zaczyna w dowolnym stanie początkowym ψ_{in} oraz że miejsce 0 licznika programu jest zajęte. Wtedy jedynym wyrazem w całym hamiltonianie, który może na początku działać, ponieważ hamiltonian działa w kolejnych chwilach, jest pierwszy wyraz $q_1^* q_0 A_1$. q_0 zmieni numer miejsca 0 na miejsce niezajęte, podczas gdy q_1^* zmieni numer miejsca 1 na zajęte. Stąd czynnik $q_1^* q_0$ jest tym, który po prostu przesuwa zajęte miejsce z położenia 0 do położenia 1. Ale jest to mnożone przez macierz A_1, która działa tylko na n atomach rejestru, a więc mnoży stan początkowy n atomów rejestru przez A_1. Jeśli teraz hamiltonian zadziała po raz drugi, to ten pierwszy czynnik nie utworzy niczego, gdyż q_0 daje 0 dla miejsca numer 0, bo jest ono teraz wolne. Wyraz, który może teraz działać, to drugi wyraz, $q_2^* q_1 A_2$, bo może on przesunąć zajęty punkt, który nazwę „kursorem". Kursor może przesunąć się z miejsca 1 na miejsce 2, ale macierz A_2 działa teraz na rejestrze, a więc działa teraz na nim macierz $A_2 A_1$. Tak więc, patrząc na pierwszy wiersz hamiltonianu: jeśli to jest całe potrzebne działanie, ponieważ hamiltonian działa według kolejnych rozkazów, kursor będzie się kolejno przesuwał od 0 do k, a my uzyskamy, jedno po drugim, działania na n atomach rejestru, czyli macierzach A, w takiej kolejności, w jakiej chcemy zbudować całą M.

Jednak hamiltonian musi być hermitowski, a zatem musi istnieć sprzężenie zespolone wszystkich tych operatorów. Przypuśćmy, że na danym etapie mamy kursor w miejscu 2 i mamy macierz $A_2 A_1$ działającą na rejestrze. Teraz q_2, które chce przesunąć to położenie na nową pozycję, nie musi pochodzić z pierwszego wiersza, ale może być w drugim wierszu. W rzeczywistości może nawet pochodzić z $q_1^* q_2 A_2^*$, co przesunie kursor z powrotem z pozycji 2 ma pozycję 1. Ale zauważmy, że gdy to następuje, operator A_2^* działa na rejestrze i dlatego całkowity operator dla rejestru to w tym przypadku $A_2^* A_2 A_1$. Ale $A_2^* A_2$ to 1 i dlatego operatorem jest tylko A_1. Widzimy więc, że gdy kursor powraca na pozycję 1, wynik netto jest taki, że tylko operator A_1 naprawdę działał na rejestrze. W ten sposób różne wyrazy hamiltonianu przesuwają kursor w tył i w przód, A akumulują się lub są ponownie redukowane. Na przykład, jeśli na dowolnym etapie kursor znajdował się na miejscu j, to na rejestr n działały macierze od A_1 do A_j. Nie ma znaczenia, czy kursor w miejscu j przybył tam bezpośrednio z 0, czy idąc dalej i wracając lub poruszając się w tył i w przód według jakiegoś wzorca, o ile w końcu dotarł do stanu j. Dlatego prawdą jest, że jeśli kursor zostanie znaleziony w miejscu k, to mamy wynik netto dla n atomów rejestru, na których stan początkowy zgodnie z oczekiwaniem działała macierz M.

Jak więc możemy działać na tym komputerze? Zaczynamy od wprowadzenia bitów wejściowych do rejestru i umieszczenia kursora w miejscu 0. Następnie sprawdzamy na przykład w miejscu k, rozpraszając elektrony, czy miejsce to jest puste czy też znajduje się w nim kursor. W momencie znalezienia kursora w miejscu k usuwamy go, aby nie mógł wrócić w dół programu, i wtedy wiemy, że rejestr

zawiera informacje wyjściowe. Możemy je wtedy spokojnie zmierzyć. Oczywiście przy pomiarach i określaniu tego wszystkiego wchodzą w grę elementy zewnętrzne, które nie są częścią naszego komputera. Z pewnością komputer musi być w interakcji ze światem zewnętrznym, zarówno w celu podania danych, jak i pobrania wyników.

Matematycznie okazuje się, że propagacja kursora w górę i w dół tego wiersza programu jest dokładnie taka sama, jaka byłaby, gdyby operatory A nie były w hamiltonianie. Innymi słowy, hamiltonian reprezentuje tylko fale, które są znane z propagacji silnie związanych elektronów lub fal spinowych w jednym wymiarze i wiemy o nich dużo. Są to fale, które wędrują w górę i w dół linii i możemy mieć pakiety fal i tak dalej. Moglibyśmy ulepszyć działanie tego komputera i sprawić, że podejmie działania balistyczne w następujący sposób: należy utworzyć ciąg miejsc poza tymi wewnątrz, których faktycznie używamy do obliczeń, powiedzmy ciąg wielu miejsc przed nimi i za nimi. Wygląda to tak, jakbyśmy mieli wartości indeksu i dla q_i, które są mniejsze od 0 i większe od k, i żadna nie jest mnożona przez macierz A, a tylko przez 1. Wtedy mielibyśmy dłuższy łańcuch spinów i moglibyśmy zacząć, umieszczając kursor z różnymi amplitudami w różnych miejscach reprezentujących początkową wejściową falę spinową, szeroki pakiet o prawie określonym pędzie, zamiast umieszczać kursor dokładnie na początku miejsca 0. Fala spinowa będzie potem przechodzić przez cały komputer w sposób balistyczny i wychodzić z drugiej strony w końcówce wyjściowej, dodanej przez nas do ciągu miejsc programu, a tam łatwiej byłoby określić jego obecność i pokierować go w inne miejsce oraz przechwycić kursor. Dlatego jednostka logiczna może działać w sposób balistyczny.

Jest to istotny punkt, który wskazuje, przynajmniej dla specjalisty od komputerów, że możemy zbudować uniwersalny komputer, gdyż wie on, że jeśli możemy zbudować dowolną jednostkę logiczną, to możemy zrobić uniwersalny komputer. Mogłoby to reprezentować uniwersalny komputer, dla którego można wykonać kompozycję elementów i rozgałęzień, co nie jest całkiem oczywiste, jeśli nie mamy pewnego doświadczenia, ale to omówię szerzej później.

6.4. Niedoskonałości i nieodwracalna strata energii swobodnej

Jest jednak wiele zagadnień, które chciałbym omówić bardziej szczegółowo, takie jak kwestia niedoskonałości. W tej maszynie jest wiele źródeł niedoskonałości, ale pierwszym, które rozważymy, jest możliwość, że współczynniki w sprzężeniach wzdłuż wiersza programu nie są dokładnie równe. Wiersz jest tak długi, że w rzeczywistych obliczeniach niewielkie nieregularności mogą spowodować pewne prawdopodobieństwo rozpraszania, a fale nie będą wędrować dokładnie balistycznie, lecz będą iść w tę i z powrotem. Jeśli przykładowo układ jest tak zbudowany, że te