Universidade Estadual de Campinas

Faculdade de Engenharia Elétrica e de Computação

Departamento de Telemática

Códigos Convolucionais Quânticos Concatenados

Tese apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Doutor em Engenharia Elétrica.

por

Antonio Carlos Aido de Almeida

Orientador: **Prof. Dr. Reginaldo Palazzo Jr. (FEEC/UNICAMP)**

Banca Examinadora:

Prof. Dr. Celso de Almeida (FEEC/UNICAMP)
Prof. Dr. Jaime Portugheis (FEEC/UNICAMP)
Prof. Dr. Carlile Campos Lavor (IME/UERJ)
Prof. Dr. Bartolomeu Ferreira Uchôa Filho (DEE/UFSC)
Prof. Dr. José Roberto Rios Leite (DF/UFPE)

Códigos Convolucionais Quânticos Concatenados

Este exemplar corresponde à redação final da tese devidamente corrigida e defendida por Antonio Carlos Aido de Almeida e aprovada pela banca examinadora.

Campinas, 14 de outubro de 2004.
Prof. Dr. Reginaldo Palazzo Jr.

Banca Examinadora:

Prof. Dr. Celso de Almeida (FEEC/UNICAMP)
Prof. Dr. Jaime Portugheis (FEEC/UNICAMP)
Prof. Dr. Carlile Campos Lavor (IME/UERJ)
Prof. Dr. Bartolomeu Ferreira Uchôa Filho (DEE/UFSC)
Prof. Dr. José Roberto Rios Leite (DF/UFPE)

Tese apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Doutor em Engenharia Elétrica.

Abstract

Decoherence is one of the major challenges facing the field of quantum computation. The field of quantum error correction has developed to meet this challenge. A group-theoretical structure and associated class of quantum codes, the stabilizer codes, has proved particularly fruitful in producing codes and in understanding the structure of both specified codes and classes of codes. All stabilizer codes discovered so far are block codes. In this thesis, we will construct a class of concatenated quantum convolutional codes. We will introduce the concept of quantum convolutional memory and some simple techniques to produce good quantum convolutional codes from classes of classical convolutional codes.

Key-words: Quantum Error Correction Codes, Convolutional Codes, Stabilizer Codes, Concatenated Codes.

Resumo

A decoerência é um dos maiores desafios obstrutivos da computação quântica. Os códigos corretores de erros quânticos têm sido desenvolvidos com o intuito de enfrentar este desafio. Uma estrutura de grupos e uma classe associada de códigos, a classe dos códigos estabilizadores, têm-se mostrado úteis na produção de códigos e no entendimento da estrutura de classes de códigos. Todos os códigos estabilizadores descobertos até o momento são códigos de bloco. Nesta tese, construiremos uma classe de códigos convolucionais quânticos concatenados. Introduziremos o conceito de memória convolucional quântica e algumas técnicas simples para produzir bons códigos convolucionais quânticos a partir de classes de códigos convolucionais clássicos.

Palavras-Chave: Códigos Corretores de Erros Quânticos, Códigos Convolucionais, Códigos Estabilizadores, Códigos Concatenados.

Agradecimentos

À minha família, que me deu todas as condições necessárias para que eu pudesse seguir em frente.

Ao Prof. Dr. Reginaldo Palazzo Jr., pela excelente orientação e convivência pessoal. Seu incentivo e constantes elogios ao meu trabalho foram de fundamental importância para o progresso da tese.

Aos meus colegas do Departamento de Telemática: Wanessa, Mário, Rodrigo, Edson, Andréia e Luzinete. Os momentos em que convivemos juntos ficarão em minha lembrança para sempre.

Aos funcionários da FEEC: Noêmia, Fernando e Míriam.

Ao CNPq, pelo suporte financeiro.

A meus pais,
Álvaro e Maria da Natividade

Conteúdo

Lista de Figuras

Lista de Tabelas

Parte I

Conceitos Fundamentais

Capítulo 1

Introdução Geral

Anybody who is not shocked by quantum theory has not understood it.

– Niels Bohr.

O primeiro capítulo desta tese apresenta o "estado da arte" da teoria da computação e da informação quântica com o intuito de contextualizar o problema da correção de erros em canais quânticos ruidosos. Dado o caráter interdisciplinar desta área de pesquisa, faz-se necessária a apresentação de um espectro amplo, variado e por vezes aprofundado de material preliminar. Tal material pode ser abordado de diferentes pontos de vista complementares. Este capítulo foi escrito sob a perspectiva de um físico teórico preocupado em trazer a física da informação para o contexto da engenharia quântica. Talvez, em um primeiro momento, alguns conceitos físicos sofisticados pareçam incompreensíveis para um engenheiro, porém, ao longo da tese, os tópicos fundamentais para a correção de erros quânticos serão abordados com a devida necessidade. Para uma abordagem mais ampla e aprofundada, sugerimos a consulta das referências [48, 67, 55].

1.1 A Física da Informação

Information is physical.

– Rolf Landauer.

A física da informação e a teoria da computação são disciplinas conhecidas por serem dependentes uma da outra há muitas décadas. Afinal de contas, informação é algo que é codificado no estado de um sistema físico e computação é algo que pode ser executado sobre um objeto fisicamente realizável. Portanto o estudo da informação e da computação devem estar relacionados ao estudo dos processos físicos fundamentais. Certamente, sob uma perspectiva da engenharia, o domínio dos princípios da física e da ciência dos materiais é necessário para o desenvolvimento do hardware computacional.

Sob uma perspectiva teórica mais abstrata, temos assitido a notáveis avanços em nosso entendimento sobre como a física nos obriga a considerar a informação, para além da matéria e da energia, na descrição de fenômenos físicos. A seguir, faremos uma breve apresentação de três resultados relativamente recentes de como a física e a teoria da informação estão cada vez mais dependentes uma da outra.

1.1.1 Princípio de Landauer

Em 1961, Rolf Landauer chamou a atenção para o fato de que o apagamento de informação é necessariamente um processo dissipativo e, portanto, irreversível [38].

Por exemplo, podemos armazenar um bit de informação ao colocarmos uma molécula dentro de uma caixa em um dos lados de uma partição que a divide ao meio. O apagamento significa que podemos mover a molécula para, digamos, o lado esquerdo, independente de qual lado começou a ação. Podemos de repente remover a partição, e então lentamente comprimir o "gás" de uma molécula com um pistão até que a molécula esteja definitivamente do lado esquerdo. Este procedimento reduz a entropia do gás por $\Delta S = k \ln 2$ (onde k é a *constante de Boltzman*), e cria um fluxo de calor associado da caixa para o ambiente. Se o processo é isotérmico à temperatura T, o trabalho $W = kT \ln 2$ é realizado sobre a caixa, trabalho este que temos que fornecer. Portanto, se tivermos que apagar informação teremos que gastar energia.

1.1.2 Computação Reversível

As portas lógicas usadas para executar uma computação são tipicamente *irreversíveis*, *e.g.*, a porta NAND:

$$(a,b) \longrightarrow \neg(a \wedge b), \tag{1.1}$$

onde $\neg$ denota a operação "not" e $\wedge$ denota a operação "and". Como se vê, esta porta tem dois bits de entrada e um bit de saída. Não podemos recuperar uma única entrada a partir do bit de saída. De acordo com o princípio de Landauer, como cerca de um bit é apagado por porta (medida sobre todas as suas possíveis entradas), é necessário pelo menos o trabalho $W = kT\ln 2$ para operar a porta. Se temos um estoque finito de baterias, esta condição parece impor um limite teórico de quão longa pode ser a computação.

Porém, Charles Bennett mostrou em 1973 que qualquer computação pode ser realizada usando somente passos *reversíveis* e portanto, em princípio, não exigiria dissipação [3]. Podemos construir uma versão reversível da porta NAND que preserva todas as informações sobre a entrada. Por exemplo, a porta de Toffoli:

$$(a,b,c) \longrightarrow (a,b,c \oplus a \wedge b), \tag{1.2}$$

onde $\oplus$ denota a operação "ou-exclusivo", é uma porta reversível de 3 bits que troca o terceiro bit se os dois primeiros forem "1" e não faz nada caso contrário. O terceiro bit da saída torna-se o NAND de a e b se $c = 1$. Podemos transformar uma computação irreversível em uma reversível trocando as portas NAND por portas Toffoli. Esta computação poderia, em princípio, ser feita com uma dissipação desprezível.

No entanto, durante este processo geramos uma quantidade de "lixo" extra. Será então que apenas adiamos o custo de energia ? Precisaremos pagar por este custo quando precisarmos apagar todo o "lixo". Bennett tratou esta questão afirmando que um computador reversível pode realizar toda a computação, imprimir uma cópia da resposta (uma operação reversível) e então *reverter* todos os seus passos para retornar à sua configuração original. Este procedimento remove o "lixo" sem qualquer custo de energia.

Em princípio, então, não precisamos pagar qualquer custo de energia para computar. Na prática, os computadores em uso atualmente (irreversíveis) dissipam ordens de grandeza maiores que $kT\ln 2$ por porta. Portanto, o limite de Landauer não é tão importante para a engenharia. Mas à medida que o hardware computacional diminuir de tamanho, pode se tornar importante atingir o limite de Landauer para impedir que os componentes dos

computadores se fundam. E, neste caso, a computação reversível pode ser a única opção para a realização de uma computação eficiente.

1.1.3 Demônio de Maxwell

As descobertas de Landauer levaram Bennett a propor em 1982 uma nova interpretação para o paradoxo da física conhecido como *demônio de Maxwell* [4][1].

Segundo Bennett, o demônio de Maxwell mostrou que os observadores não desempenham um papel passivo nos experimentos de física. O demônio coleta e armazena informação sobre as moléculas. Se o demônio tem uma capacidade de memória finita, não pode continuar a esfriar o gás indefinidamente; eventualmente, alguma informação terá que ser *apagada*. Neste ponto, finalmente paga-se a conta pelo esfriamento obtido (se o demônio não apagar informações, então devemos associar alguma entropia com a infomação armazenada).

Esta interpretação foi, em parte, antecipada em 1929 por um pioneiro da física da informação − Leo Szilard [71]. Szilard, em sua análise do paradoxo, inventou o conceito de um *bit* de informação (o nome bit foi introduzido mais tarde, por Tukey) e associou a entropia $\Delta S = k \ln 2$ com a aquisição de um bit (embora Szilard não tenha compreendido totalmente o princípio de Landauer, de que é o apagamento do bit que leva a um inevitável custo de energia).

[1] Uma descrição simplificada deste paradoxo: Imagine duas salas ligadas por um corredor (pode também ser uma caixa dividida ao meio por uma veneziana que abre e fecha − o princípio é o mesmo). Se a temperatura do ar for a mesma em ambas as salas, a velocidade média das moléculas de ar nos dois ambientes também será a mesma, já que a temperatura de qualquer gás é função da velocidade média com a qual as moléculas estão se movendo. Note que a velocidade varia consideravelmente de molécula para molécula; a velocidade de uma molécula em particular pode ser muito maior ou muito menor que a velocidade média. Imagine agora que um ser, um *demônio*, seja capaz de controlar a porta que separa as duas salas. Se o demônio conhece a velocidade das moléculas que estão se aproximando da porta, pode permitir que apenas as mais velozes passem para uma das salas e que apenas as mais lentas entrem na outra. Depois de algum tempo, a sala onde se acumularam as moléculas rápidas estará cheia de ar quente e a outra sala ficará cheia de ar frio. Este proceso ocorre com um gasto desprezível de calor. Ou seja, o calor flui de um local para outro sem nenhum custo, em aparente violação da segunda lei da termodinâmica.

1.2 Informação Quântica

Os três exemplos apresentados na seção anterior não deixam dúvidas de que a pesquisa de interface entre a física e a teoria da informação vem produzindo valiosos resultados, de interesse tanto para os físicos quanto para os cientistas da computação. Dadas as evidências já conhecidas e as suas previsíveis (e, talvez, imprevisíveis) consequências teóricas e tecnológicas, é imperioso que tal linha de pesquisa seja ampliada e aprofundada.

A lição que extraímos destes exemplos é que, de fato, como prevera Landauer, a *informação é física* e que é instrutivo considerar o que a física tem a nos dizer a respeito da informação. Mais do que isso, estas descobertas nos fizeram perceber (ou talvez lembrar) que a física não pode ser descrita apenas em termos de energia e matéria. Deve incluir também a informação e esta não deve ser vista como uma idéia abstrata, mas sim como uma entidade concreta, ou seja, algo com algum significado físico[2].

Sabemos que o nosso universo é regido fundamentalmente pelas leis da mecânica quântica. Mas como a física quântica pode iluminar nossas idéias a respeito da natureza da informação ? Desde os primórdios da teoria quântica ficou claro que as idéias clássicas sobre a informação precisariam ser revistas sob a nova física. Por exemplo, os cliques registrados em um detector que monitora uma fonte radioativa são descritos por um processo de Poisson *verdadeiramente aleatório*. Em contraste com esta situação quântica, não existe verdadeira aleatoriedade na dinâmica clássica determinística (embora, é claro, um sistema clássico complexo (caótico) possa exibir um comportamento que na prática é indistinguível da aleatoriedade). Além disso, como veremos mais adiante, na teoria quântica os observáveis que não comutam entre si não podem ter simultaneamente valores precisos definidos (princípio da incerteza de Heisenberg), e a realização da medida de um observável A irá necessariamente influenciar o resultado de uma subsequente medida de um observável B, se A e B não comutam entre si. Portanto, o ato de adquirir informação a respeito de um sistema físico inevitavelmente destruirá o estado do sistema. Não há contra-partida desta limitação na física clássica.

[2] John Wheeler, considerado o pai dos buracos negros e de algumas teorias modernas da fissão nuclear, da física quântica e da teoria da relatividade geral, resumiu em poucas palavras a física que prevê para o século XXI: "O *it* vem do *bit*".

O fato do ato de adquirir informação implicar na destruição do estado quântico está relacionado à aleatoriedade quântica. Isto porque o resultado de uma medida tem um elemento aleatório que faz com que sejamos incapazes de inferir o estado inicial do sistema a partir do resultado da medida.

Que adquirir informação causa a destruição do estado quântico observado também está conectado a outra distinção essencial entre a informação clássica e a quântica: a informação quântica não pode ser copiada com perfeita fidelidade. Este resultado, conhecido como *teorema da não clonagem*, foi anunciado por Wootters, Zurek [75] e Dieks [15] em 1982 e será provado na próxima seção. Se pudéssemos fazer uma cópia perfeita do estado quântico, poderíamos medir um observável da cópia sem perturbar o estado original e assim derrubar o princípio da destruição do estado observado. Por outro lado, nada nos impede de copiar informação clássica perfeitamente.

Estas propriedades da informação quântica são importantes, mas a propriedade mais profunda que faz com que a informação quântica difira da informação clássica emergiu do trabalho de John Bell [2] em 1964, que mostrou que as predições da mecânica quântica não podem ser reproduzidas por nenhuma teoria de variável escondida local. Bell mostrou que a informação quântica pode ser (e tipicamente é) codificada em *correlações não locais* entre as diferentes partes de um sistema físico, correlações estas que não têm contra-partida clássica. Algumas implicações do teorema de Bell serão discutidas mais adiante.

O estudo da informação quântica como disciplina emergiu no início dos anos 80, e "explodiu" na década de 90. Muitos dos resultados centrais da teoria da informação clássica têm análogos quânticos que foram descobertos e desenvolvidos recentemente. Alguns destes resultados serão discutidos ao longo deste capítulo.

1.3 Mecânica Quântica

Hilbert space is a big place.

– Carlton Caves.

A unidade fundamental da informação clássica é o *bit*, uma variável aleatória que pode assumir dois valores: 0 ou 1. A unidade de informação quântica correspondente é o bit

quântico ou *qubit*. O qubit é um vetor em um espaço vetorial complexo bidimensional com produto interno − ou seja, um vetor no espaço de Hilbert. De acordo com a notação introduzida por Dirac, podemos chamar os elementos de uma base ortonormal deste espaço de $|0\rangle$ e $|1\rangle$ [3]. Esta base é chamada de *base computacional*. Desta forma, o estado de um qubit arbitrário normalizado pode ser escrito como

$$|\psi\rangle = a|0\rangle + b|1\rangle, \tag{1.3}$$

onde $|a|^2 + |b|^2 = 1$, com $a, b \in \mathbb{C}$ (o corpo dos números complexos). A *base conjugada* é definida como $|\pm\rangle = \frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle)$.

O estado de um qubit também pode ser representado geometricamente. Podemos reescrever o estado quântico (1.3) como

$$|\psi\rangle = e^{i\gamma}\left[\cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\varphi}\sin\left(\frac{\theta}{2}\right)|1\rangle\right], \tag{1.4}$$

onde θ, φ e γ são números reais. O fator de fase global $e^{i\gamma}$ é *fisicamente* irrelevante e, portanto, pode ser ignorado[4]. Os números θ e φ definem um ponto em uma esfera, como mostrado na Figura 1.1. Esta esfera, conhecida como *esfera de Bloch*, oferece uma visualização gráfica do estado de um qubit e freqüentemente serve de referência para a discussão de idéias sobre a teoria da computação e da informação quântica. No entanto, tais discussões costumam ser limitadas, já que não há uma generalização simples conhecida da esfera de Bloch para múltiplos qubits.

Muitos sitemas físicos diferentes podem ser usados para produzir um qubit, como por exemplo o alinhamento do spin de um elétron orbitando um átomo em relação a um campo magnético externo, as duas diferentes polarizações de um fóton e o alinhamento de um spin nuclear em relação a um campo magnético nuclear.

O estado quântico de n qubits pode ser expresso como um vetor em um espaço de 2^n dimensões (ou seja, o produto tensorial de n espaços bidimensionais). Podemos escolher como uma base ortonormal para este espaço os estados nos quais cada qubit tem um valor

[3] Para o leitor não familiarizado com os postulados da mecânica quântica, sugerimos a consulta das referências [59, 16, 17].

[4] Entenda-se, *sem efeitos observáveis.*

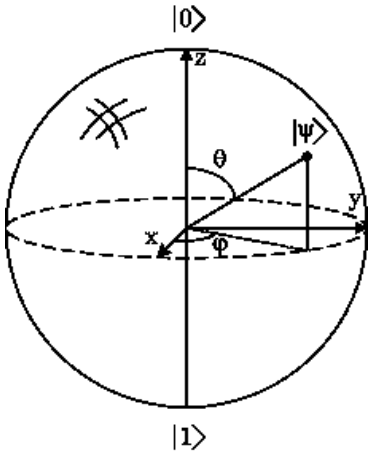

Figura 1.1: Representação da esfera de Bloch para um qubit.

definido, $|0\rangle$ ou $|1\rangle$. Desta forma, um vetor geral normalizado pode ser escrito nesta base como

$$|\psi\rangle = \sum_{x=0}^{2^n-1} a_x|x\rangle, \tag{1.5}$$

onde associamos com cada *string* o número que ela representa na notação binária, um valor entre 0 e 2^{n-1}. Os coeficientes a_x são números complexos satisfazendo a relação $\sum_x |a_x|^2 = 1$.

Com múltiplos qubits, podemos ter estados que podem não ser escritos como o produto tensorial de estados de um qubit. Por exemplo, o estado de dois qubits

$$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \tag{1.6}$$

não pode ser decomposto desta forma. Diz-se então que tal estado é *emaranhado*. Como veremos mais adiante, os estados emaranhados são os responsáveis pela grande capacidade computacional dos computadores quânticos e desempenham uma função crucial na correção de erros quânticos. Em particular, o estado (1.6) é conhecido como *estado de Bell* ou *par EPR* (Einstein-Podolsky-Rosen, [53, 2])[5]. Este estado serve como unidade básica de emaranhamento em muitas aplicações de criptografia quântica [15, 20].

[5]Na verdade, o estado (1.6) é apenas *um* dos quatro estados de Bell ou pares EPR, matematicamente descritos como $|\beta_{x,y}\rangle \equiv \frac{|0,y\rangle + (-1)^x|0,\bar{y}\rangle}{\sqrt{2}}$, onde $x, y = \{0, 1\}$ [48].

O resultado da medida do estado (1.3) é *não* determinístico − a probabilidade de obtermos o resultado $|0\rangle$ é $|a|^2$ e a probabilidade de obtermos $|1\rangle$ é $|b|^2$. A normalização assegura que a probabilidade de obter algum resultado seja exatamente 1.

Esta medida implementa um de dois operadores de projeção, as projeções sobre a base $\{|0\rangle, |1\rangle\}$. Esta não é a única medida que podemos fazer sobre um qubit. Na verdade, podemos projetar sobre qualquer base do espaço de Hilbert de um qubit. Se tivermos múltiplos qubits, podemos medir um número de diferentes qubits independentemente, ou podemos medir alguma propriedade comum dos qubits, o que corresponde a fazermos a projeção sobre alguma base emaranhada do sistema. Assim, se medirmos todos os n qubits do estado (1.5) através da projeção de cada um deles sobre a base $\{|0\rangle, |1\rangle\}$, a probabilidade de obter o resultado $|x\rangle$ é $|a_x|^2$. Em particular, o par EPR (1.6) tem uma importante propriedade: a medida de um qubit *sempre* dá o mesmo resultado da medida do outro qubit. Isto é, os resultados das medidas estão *correlacionados*.

Uma forma de entender um sistema quântico é observar o comportamento de vários operadores atuando sobre o estado do sistema. Por exemplo, um conjunto de operadores a considerar para um qubit é o conjunto das *matrizes de Pauli*. Estas matrizes na base $\{|0\rangle, |1\rangle\}$ são:

$$\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \sigma_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \quad \text{e} \quad \sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{1.7}$$

Uma outra notação para estas mesmas matrizes é:

$$X = \sigma_x, \quad Y = XZ = -i\sigma_y = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \quad \text{e} \quad Z = \sigma_z. \tag{1.8}$$

Além das matrizes de Pauli e da matriz identidade I com duas dimensões, um outro operador importante para o sistema de um qubit é o operador de Hadamard, que promove a mudança da base computacional para a base conjugada. Ou seja, o *operador de Hadamard* é uma rotação de π radianos em torno do eixo $\frac{1}{\sqrt{2}}(x+z)$:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{1.9}$$

Para um sistema com n qubits, σ_i^j denota uma matriz σ_i atuando sobre o qubit j e uma matriz I atuando sobre todos os outros qubits. O *grupo de Pauli* sobre n qubits, denotado por $\mathcal{G}_n$, é gerado por σ_i^j para $i = x, y, z$ e $j = 1, \ldots, n$. Poderemos também nos referir ao grupo de Pauli como sendo o menor *subgrupo real* gerado por X e Z.

A medida que fizemos para um qubit corresponde à medida do autovalor de σ_z. Os correspondentes operadores de projeção são $\frac{1}{2}(I \pm \sigma_z)$. Para uma partícula de spin-1/2, esta medida é realizada através da medida do spin da partícula ao longo do eixo z. Também poderíamos medir ao longo do eixo x ou y, o que corresponde a medir o autovalor de σ_x ou σ_y. Neste caso, os operadores de projeção são $\frac{1}{2}(I \pm \sigma_x)$ e $\frac{1}{2}(I \pm \sigma_y)$, respectivamente.

Podemos também fazer medidas de operadores mais gerais, desde que tais operadores tenham autovalores reais. Uma matriz A tem autovalores reais se, e somente se, for hermitiana, ou seja: $A^\dagger = A$, onde $A^\dagger$ é o adjunto hermitiano (ou simplesmente adjunto), igual ao transposto do conjugado complexo. Ao operador hermitiano que realiza medidas projetivas dá-se o nome de *observável*.

Note que todas as matrizes de Pauli são hermitianas. Além disso, estas matrizes satisfazem uma importante propriedade algébrica − *anticomutam* umas com as outras. Ou seja,

$$\{\sigma_i, \sigma_j\} = \sigma_i\sigma_j + \sigma_j\sigma_i = 0, \tag{1.10}$$

sempre que $i \neq j$ (com $i, j \in \{x, y, z\}$). Outra possível relação entre dois operadores A e B é *comutarem* entre si. Ou seja,

$$[A, B] = AB - BA = 0. \tag{1.11}$$

É possível que duas matrizes não comutem nem anticomutem entre si. Duas matrizes que comutam entre si podem ser simultaneamente diagonalizadas. Isto significa que podemos medir o autovalor de uma delas sem destruir os autovetores da outra. No caso contrário, quando dois operadores não comutam entre si, a medida provoca a destruição dos autovetores da outra, e portanto não podemos medir simultaneamente operadores que não comutam entre si.

Existe um produto interno complexo natural para estados quânticos. Dada uma base

ortonormal $|\psi_i\rangle$, o produto interno entre $|\alpha\rangle = \sum c_i|\psi_i\rangle$ e $|\beta\rangle = \sum d_i|\psi_i\rangle$ é:

$$\langle\alpha|\beta\rangle = \sum c_i^\star d_j \langle\psi_i|\psi_j\rangle = \sum c_i^\star d_i. \tag{1.12}$$

A cada *ket* $|\psi\rangle$ corresponde um *bra* $\langle\psi|$. O adjunto hermitiano é com respeito a este produto interno, portanto $U|\psi\rangle$ corresponde a $\langle\psi|U^\dagger$. O operador $\sum|\psi\rangle\langle\phi|$ atua sobre o espaço de Hilbert como:

$$(\sum|\psi\rangle\langle\phi|)|\alpha\rangle = \sum\langle\phi|\alpha\rangle|\psi\rangle. \tag{1.13}$$

O produto interno pode revelar uma grande quantidade de informação sobre a estrutura de um conjunto de estados. Por exemplo, $\langle\psi|\phi\rangle = 1$ se, e somente se, $|\psi\rangle = |\phi\rangle$.

Autovetores de um operador hermitiano A com diferentes autovalores são automaticamente ortogonais:

$$\begin{aligned}\langle\psi|A|\phi\rangle &= \langle\psi|(A|\phi\rangle) = \lambda_\phi\langle\psi|\phi\rangle \\ &= (\langle\psi|A)|\phi\rangle = \lambda_\psi^\star\langle\psi|\phi\rangle.\end{aligned} \tag{1.14}$$

Como os autovalores de A são reais, segue que $\langle\psi|\phi\rangle = 0$ sempre que $\lambda_\psi \neq \lambda_\phi$. Caso contrário, se $\langle\psi|\phi\rangle = 0$, existe um operador hermitiano para o qual $|\psi\rangle$ e $|\phi\rangle$ são autovetores com diferentes autovalores.

Uma importante propriedade dos estados quânticos tem profundas implicações para a computação quântica e a correção de erros quânticos: o teorema da não clonagem, que afirma que é *impossível* fazer uma cópia perfeita de um estado quântico *arbitrário desconhecido* [75, 15]. Teremos agora a oportunidade de prová-lo. Este teorema é uma consequência da linearidade da mecânica quântica.

Suponha que desejamos realizar uma operação que mapeia um estado arbitrário

$$|\psi\rangle \rightarrow |\psi\rangle \otimes |\psi\rangle. \tag{1.15}$$

Então um outro estado arbitrário $|\phi\rangle$ seria mapeado por

$$|\phi\rangle \rightarrow |\phi\rangle \otimes |\phi\rangle. \tag{1.16}$$

Como a transformação deve ser linear, segue que

$$|\psi\rangle + |\phi\rangle \rightarrow |\psi\rangle \otimes |\psi\rangle + |\phi\rangle \otimes |\phi\rangle. \tag{1.17}$$

No entanto,

$$|\psi\rangle \otimes |\psi\rangle + |\phi\rangle \otimes |\phi\rangle \neq (|\psi\rangle + |\phi\rangle) \otimes (|\psi\rangle + |\phi\rangle), \tag{1.18}$$

e portanto não é possível copiar $|\psi\rangle + |\phi\rangle$. Em geral, dada uma base ortonormal, podemos copiar os estados da base, porém não podemos copiar corretamente as *superposições* daqueles estados da base. Isto significa que para realizar a correção de erros quânticos, não podemos simplesmente fazer cópias de reserva do estado quântico a ser preservado.

1.4 Complexidade Computacional Quântica

Can physics be simulated by a universal computer ? [...] the physical world is quantum mechanical, and therefore the proper problem is the simulation of quantum physics [...] the full description of quantum mechanics with R *particles [...] has too many variables, it cannot be simulated with a normal computer with a number of elements proportional to* R *[... but it can be simulated with quantum computer elements]. [...] Can a quantum system be probabilistically simulated by a classical (probabilistic, I'd assume) universal computer ? [...] If you take the computer to be the classical kind I've described so far [...] the answer is certainly, No!*

−Richard Feynman (1982) [21].

Vimos que o estado de um computador clássico é um vetor sobre $\mathbb{Z}_2$ (o corpo dos números inteiros módulo-2). O estado de um computador quântico (ou de qualquer sistema físico) é um vetor sobre o espaço de Hilbert. Um computador clássico com n bits tem 2^n possíveis estados, mas é somente um espaço vetorial de n dimensões sobre $\mathbb{Z}_2$. Um computador quântico com n qubits é um estado em um espaço vetorial complexo com 2^n dimensões.

Estamos agora em condições de descrever uma computação quântica. Inicialmente reunimos n qubits e preparamos estes qubits em um estado inicial padrão tal como $|x = 0\rangle$. Aplicamos então uma transformação unitária U aos n qubits (a transformação U é construída como um produto de *portas quânticas padrão* − transformações unitárias que agem em alguns qubits em um instante de tempo. Por exemplo, para um qubit, as matrizes de Pauli são as portas quânticas padrão). Após U ser aplicado, medimos todos os qubits projetando-os sobre a base $\{|0\rangle, |1\rangle\}$. Vimos na seção anterior que o resultado final desta medida é uma *informação clássica* que pode ser impressa, por exemplo, em uma tese de doutorado.

Note que o algoritmo executado pelo computador quântico é um algoritmo intrinsicamente *probabilístico*. Isto é, por causa da aleatoriedade do processo de medida quântica, podemos executar exatamente o mesmo algoritmo duas vezes e obter resultados diferentes. Na verdade, o algoritmo quântico gera uma *distribuição de probabilidades* das possíveis saídas.

Embora um computador quântico opere de acordo com princípios físicos distintos dos princípios de um computador clássico, um computador quântico não pode fazer nenhuma coisa que um computador clássico não seja capaz de fazer. Computadores clássicos podem armazenar vetores, rotacionar vetores e modelar o processo de medida quântica pela projeção do vetor sobre eixos mutuamente ortogonais. Portanto, um computador clássico pode, certamente, *simular* um computador quântico com boa precisão. Nossa noção do que é *computável* será a mesma, seja com um computador clássico ou com um computador quântico. No entanto, devemos também levar em consideração o quão longa será esta simulação. Suponha que tenhamos um computador que opera com um número modesto de qubits, digamos n=100. Então para representar o estado quântico típico do computador, precisaríamos computar $2^n = 2^{100} \sim 10^{30}$ números complexos! Nenhum computador clássico existente é capaz de fazer isto em um tempo razoável. Além disso, executar uma rotação geral de um vetor em um espaço com 10^{30} dimensões está muito além da capacidade computacional de qualquer computador clássico previsto para o futuro.

Portanto, é verdade que um computador clássico pode simular um computador quântico, mas a simulação torna-se extremamente ineficiente à medida que o número de qubits

n aumenta. Sistemas quânticos são difíceis de simular classicamente porque genericamente utilizam o espaço de Hilbert total de 2^n dimensões à medida que evoluem, exigindo *fontes clássicas exponenciais* (tais como tempo e memória). Esta observação levou Feynamn a especular que um computador quântico seria capaz de realizar tarefas que estão longe de serem realizadas por qualquer computador clássico concebível. Os algoritmos quânticos que veremos mais adiante parecem corroborar esta visão.

Mas será que esta visão sobre a simulação clássica da capacidade computacional dos computadores quânticos está realmente correta ? A simulação clássica deve oferecer um meio de determinar probabilidades para todas as possíveis saídas da medida final. Poderíamos então supor que não é *realmente* necessário ter uma descrição completa do estado quântico de n qubits. Bastaria procurar por um *algoritmo clássico probabilístico* no qual a saída não é unicamente determinada pela entrada, mas sim por uma distribuição de probabilidades que coincida com o que é gerado pela computação quântica. Neste caso, poderíamos realizar uma simulação *local*, na qual cada qubit tem um valor definido para cada instante de tempo, e cada porta quântica pode atuar sobre os qubits de várias formas possíveis, cada uma das quais a ser selecionada por um gerador numérico (pseudo)-aleatório. Esta evolução seria muito mais fácil que a evolução de um vetor em um espaço exponencialmente grande. Porém, a conclusão do teorema de John Bell [2] é, *precisamente*, que esta simulação nunca será possível: *não existe algoritmo probabilístico local que possa reproduzir as conclusões da mecânica quântica.* Portanto, a simulação é um problema muito difícil para qualquer computador clássico.

Para entender melhor porque a descrição matemática da informação quântica é tão complexa, imagine que tenhamos um sistema quântico de $3n$ qubits ($n \gg 1$), dividido em três subsistemas de n qubits cada (chamados subsistemas (1), (2) e (3)). Escolhemos aleatoriamente um sistema quântico dos $3n$ qubits, e então separamos os três subsistemas, enviando o subsistema (1) para São Paulo e o subsistema (3) para o Rio de Janeiro, deixando o subsistema (2) em Campinas. Gostaríamos de fazer algumas medidas para saber o quanto podemos saber sobre o sistema quântico. Para tornar esta tarefa mais simples, imagine que tenhamos muitas cópias do estado do sistema de maneira que podemos

medir quaisquer observáveis que quisermos[6], mas estamos restritos a realizar cada medida dentro de um dos subsistemas − medidas coletivas abrangendo os limites dos subsistemas não são permitidas. Então para um estado *típico* do sistema de $3n$ qubits, nossas medidas não revelam quase nada sobre o estado. Quase toda a informação que distingue um estado de outro está nas *correlações não locais* entre os resultados das medidas nos subsistemas (1), (2) e (3). São estas as correlações não locais a que Bell se referiu como sendo uma parte essencial da descrição física.

O conteúdo da informação pode ser quantificado pela entropia (entropia grande significa pouca informação). Se escolhemos um estado dos $3n$ qubits aleatoriamente, quase sempre encontramos que a entropia de cada subsistema é muito próxima de[7]

$$S \cong n - 2^{-(n+1)}, \tag{1.19}$$

onde n é o valor máximo de entropia, correspondendo ao caso em que o sistema não carrega nenhuma informação acessível. Portanto, para n grande podemos acessar somente uma quantidade exponencialmente pequena de informação através da observação de cada subsistema *separadamente*. Isto é, as medidas revelam muito pouca informação se não considerarmos como os resultados das medidas obtidas em São Paulo, Rio de Janeiro e Campinas estão correlacionados uns com os outros − Na linguagem que estamos usando, uma medida de uma correlação é uma medida *coletiva* (embora tal medida pudesse ser realizada por físicos experimentais que observam as partes separadas da mesma cópia do estado e então trocam informações entre si para comparar seus resultados). Assim, através da medida das correlações podemos aprender muito mais; *em princípio*, podemos reconstruir o estado completamente.

Qualquer descrição satisfatória do estado de $3n$ qubits deve caracterizar estas correlações não locais, as quais são excessivamente complexas. É por esta razão que uma simulação clássica de um sistema quântico grande requer fontes imensas (quando tais correlações não locais existirem entre as partes de um sistema, dizemos que as partes es-

[6]Não podemos fazer cópias de um estado quântico desconhecido, mas podemos convidar um amigo para preparar muitas cópias idênticas do estado (ele pode fazer isso porque ele sabe qual é o estado), mas sem nos dizer qual estado.

[7]Este resultado foi encontrado por Don Page [55].

tão emaranhadas, significando que *não* podemos decifrar totalmente o estado do sistema dividindo-o em partes e estudando as partes separadamente).

1.5 Paralelismo Quântico

The theory of computation has traditionally been studied almost entirely in the abstract, as a topic in pure mathematics. This is to miss the point of it. Computers are physical objects, and computations are physical processes. What computers can or cannot compute is determined by the laws of physics alone, and not by pure mathematics.

– David Deutsch.

Em 1985, David Deutsch publicou um importante artigo no qual tratava de forma mais concreta as idéias lançadas por Feynman sobre o computador quântico universal. Neste trabalho, Deutsch enfatizou que podemos melhor nos dar conta do poder computacional de um computador quântico se invocarmos o que ele chamou de *paralelismo quântico* [12]. Heuristicamente, e sob o risco de excessiva simplificação, podemos afirmar que o paralelismo quântico permite que os computadores quânticos avaliem uma função $f(x)$ *simultaneamente* para muitos valores *diferentes* de x. Para entender melhor o que isto significa, consideremos o exemplo descrito por Deutsch.

Imagine uma caixa preta que computa uma função que mapeia um *bit* x em um bit $f(x)$. Não sabemos o que está acontecendo dentro da caixa, mas deve ser algo complicado porque a computação demora 24 horas. Existem quatro possíveis funções $f(x)$ (porque $f(0)$ e $f(1)$ podem assumir, cada uma, dois possíveis valores: 0 ou 1). Gostaríamos de saber o que a caixa está computando. Levaríamos 48 horas para descobrir $f(0)$ e $f(1)$. Mas gostaríamos de obter a resposta em 24 horas, não em 48 horas. Mesmo que quiséssemos saber apenas se $f(x)$ é uma função *constante* ($f(0) = f(1)$) ou *balanceada* ($f(0) \neq f(1)$), ainda assim levaríamos 48 horas para obter a resposta.

Suponha agora que tenhamos uma caixa preta quântica que computa $f(x)$. A ação do computador quântico é unitária e deve ser inversível, portanto precisamos de uma transformação U_f que mapeia dois qubits em dois qubits:

$$U_f : |x\rangle|y\rangle \rightarrow |x\rangle|y \oplus f(x)\rangle. \tag{1.20}$$

(Esta máquina troca o segundo qubit se $f(x) = 1$ e preserva o segundo qubit se $f(x) = 0$). Poderíamos determinar se $f(x)$ é uma função constante ou balanceada usando a caixa preta quântica duas vezes, em 48 horas. Mas, como veremos a seguir, podemos obter a resposta em apenas 24 horas, usando a caixa preta quântica *somente uma vez* − Este é o *Problema de Deutsch* [48].

Como a caixa preta é um computador quântico, podemos escolher o estado de entrada como sendo uma superposição de $|0\rangle$ e $|1\rangle$. Se o segundo qubit é inicialmente preparado no estado $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, então

$$\begin{aligned} U_f : |x\rangle\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) &\rightarrow |x\rangle\frac{1}{\sqrt{2}}(|f(x)\rangle - |f(x) \oplus 1\rangle) \\ &= |x\rangle(-1)^{f(x)}\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle), \end{aligned} \tag{1.21}$$

portanto isolamos a função f em uma fase dependente de x. Suponha agora que preparamos o primeiro qubit como $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. Então, a caixa preta atua como

$$\begin{aligned} U_f : \quad & \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \rightarrow \\ & \frac{1}{\sqrt{2}}\left[(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle\right]\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle). \end{aligned} \tag{1.22}$$

Finalmente, podemos fazer uma medida que projeta o primeiro qubit sobre a base conjugada

$$|\pm\rangle = \frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle). \tag{1.23}$$

Evidentemente, sempre obteremos $|+\rangle$ se a função for balanceada, e $|-\rangle$ se a função for constante[8].

[8]Em nossa descrição anterior de computação quântica, afirmamos que a medida final projetaria cada qubit sobre a base $\{|0\rangle, |1\rangle\}$, mas aqui estamos permitindo uma medida em uma base diferente. Para descrever o procedimento sobre a base $\{|0\rangle, |1\rangle\}$ deveríamos aplicar uma mudança unitária de base para cada qubit antes de realizar a medida final.

Portanto, resolvemos o problema de Deutsch, e tivemos uma idéia da extraordinária capacidade computacional de um computador quântico. O computador clássico levará o dobro do tempo para distinguir uma função balanceada de uma função constante, enquanto que o computador quântico faz o mesmo trabalho de uma só vez.

Isto é possível porque o computador quântico não é limitado a computar somente $f(0)$ ou $f(1)$. Pode atuar sobre uma *superposição* de $|0\rangle$ e $|1\rangle$, e deste modo extrair informação *global* sobre a função, informação que depende de $f(0)$ *e* de $f(1)$. Isto é o que chamamos de paralelismo quântico.

Suponha agora que estejamos interessados nas propriedades globais de uma função que atua sobre n qubits, uma função com 2^n possíveis argumentos. Para computar uma tabela completa dos valores de $f(x)$, teríamos que calcular $f(x)$ 2^n vezes, algo completamente inviável para $n \gg 1$ (*e.g.* 10^{30} vezes para $n = 100$). Mas com um computador quântico que atua de acordo com

$$U_f : |x\rangle|x\rangle \rightarrow |x\rangle|f(x)\rangle, \tag{1.24}$$

poderíamos escolher o registro de entrada como estando no estado

$$\left[\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right]^{\otimes n} = \frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle, \tag{1.25}$$

e através de uma única computação de $f(x)$, podemos gerar um estado

$$\frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle|f(x)\rangle. \tag{1.26}$$

As propriedades globais de f estão codificadas neste estado, e somos capazes de extrair algumas destas propriedades se pudermos pensar em um meio eficiente de fazer isto.

Esta computação quântica exibe um *paralelismo quântico massivo*; uma simulação da preparação deste estado em um computador clássico exigiria a computação de f um número inimaginavelmente grande de vezes (para $n \gg 1$). Com o computador quântico conseguimos fazer isto de uma única vez. É exatamente este paralelismo massivo que Shor usou em seu algoritmo de fatoração de números grandes, a ser descrito mais adiante.

Como observado anteriormente, uma das principais característica da mecânica quântica é que a informação pode ser codificada em correlações não locais entre as diferentes partes de um sistema físico. Na verdade, este é o caso do estado (1.26); as propriedades da função f estão armazenadas nas correlações entre o *registro de entrada* e o *registro de saída* do computador quântico. Esta correlação não local, no entanto, não é fácil de ser decifrada.

Por exemplo, se tivéssemos que medir o registro de entrada obteríamos o estado $|x_0\rangle$, onde x_0 é escolhido aleatoriamente dentre um conjunto de 2^n possíveis valores. Este procedimento prepararia um estado

$$|x_0\rangle|f(x_0)\rangle. \tag{1.27}$$

Poderíamos continuar a medir o registro de saída para descobrir $f(x_0)$. Mas como o estado (1.26) foi destruído pela medida, as intricadas correlações entre os registros foram perdidas, e não teríamos a oportunidade de determinar $f(y_0)$ para qualquer $y_0 \neq x_0$ através de medida futuras. Neste caso então, a computação quântica não oferece vantagem sobre a computação clássica.

A lição que tiramos da solução para o problema de Deutsch é que podemos algumas vezes ser mais inteligentes em explorar as correlações codificadas no estado (1.26). Muito da estrutura dos algoritmos quânticos envolve descobrir caminhos para fazer um uso mais inteligente das correlações não locais.

1.6 A Nova Teoria da Complexidade

It should not come as a surprise that our choice of polynomial algorithms as the mathematical concept that is supposed to capture the informal notion of "practically efficient computation" is open to criticism from all sides. [...] Ultimately, our argument for our choice must be this: Adopting polynomial worst-case performance as our criterion of efficiency results in an elegant and useful theory that says something meaningful about practical computation, and would be impossible without this simplification.

– Christos Papadimitriou.

O computador que usamos em casa ou no trabalho não é um computador quântico, mas ainda assim é uma máquina fantástica. Em princípio, é capaz de realizar qualquer computação imaginável. Porém, na prática, existem computações que não é capaz de realizar por exigirem uma quantidade extraordinariamente grande de tempo e/ou memória. Mas se fornecermos uma quantidade ilimitada de memória, e se estivermos dispostos a esperar quanto tempo for necessário, então qualquer coisa que mereça ser chamada de uma computação pode ser feita em nosso modesto PC. Dizemos, portanto, que nosso computador é um *computador universal*.

A teoria da complexidade clássica tem se desenvolvido ao longo dos anos para classificar os problemas de acordo com a sua dificuldade. Usualmente, a classificação de um problema entre "difícil" e "fácil" é feita em termos do quanto de tempo e memória é necessário para resolvê-lo. Mas como fazer distinções claras entre "difícil" e "fácil" sem especificar o hardaware que iremos usar ? Afinal, um problema pode ser difícil em um PC, mas talvez exista uma outra máquina que o resolva rapidamente. Ou talvez no futuro um computador muito melhor seja capaz de resolver o problema com mais eficiência. Portanto, as distinções entre um problema "difícil" e um problema "fácil" devem ser *universais*, ou seja, não podem depender da máquina que estamos usando.

A teoria clássica da complexidade está focada sobretudo na distinção entre os algoritmos de *tempo polinomial* e os algoritmos de *tempo exponencial*. Para qualquer algoritmo A atuando sobre um dado de entrada de comprimento variável, podemos associar uma *função de complexidade* $T_A(n)$, onde n é o comprimento do dado de entrada em bits. A função $T_A(n)$ é o tempo (isto é o número de passos elementares) mais longo necessário para o algoritmo executar uma tarefa por completo (por exemplo, se A é um algoritmo de fatoração, $T_A(n)$ é o tempo necessário para fatorar um número de n bits no pior caso possível). É dito que A é um algoritmo *polinomial* se

$$T_A(n) \leq Pol(n), \tag{1.28}$$

onde $Pol(n)$ denota um polinômio de n. Portanto, quando se diz que um problema necessita de um algoritmo de tempo polinomial para ser resolvido, estamos querendo dizer que o tempo necessário para resolver aquele problema não cresce mais rápido que uma potência

do número de bits do dado de entrada.

Se o problema não é de tempo polinomial, diz-se que é de tempo exponencial (embora isto seja uma designação incorreta, pois existem funções superpolinomiais como $n^{\log n}$ que aumentam muito mais lentamente que uma função exponencial). Este é um meio *razoável* de traçar uma linha que distingue problemas "difíceis" de problemas "fáceis". Mas a principal razão para fazer a distinção desta forma é que o critério em questão *independe* da máquina: não importa qual computador estamos usando, a classificação será sempre a mesma. A universalidade da distinção entre polinomial e exponencial segue de um dos resultados centrais da ciência da computação: um computador (clássico) universal pode simular outro com, no pior dos casos, um "limitante superior polinomial". Isto significa que, se somos capazes de executar um algoritmo em nosso computador em tempo polinomial, então poderemos sempre executá-lo em tempo polinomial em qualquer outro computador. Portanto, sempre concordaremos sobre quais algoritmos são de tempo polinomial.

À classe de problemas que pode ser resolvida com algoritmos que se utilizam de fontes (tais como tempo e memória) polinomiais com o tamanho do dado de entrada dá-se o nome de *classe P*. Tais problemas computacionais podem ser facilmente resolvidos em um computador clássico. Dentre os problemas de tempo não polinomial, ou seja, de tempo exponencial, alguns pertencem à *classe NP* (de *non-deterministic polynomial*). Esta classe engloba todos os problemas que, dada uma caixa preta que forneça a solução (um oráculo), pode-se *verificar* em tempo polinomial se a solução é correta. Um problema NP bastante conhecido é o problema de encontrar os fatores primos de um número com n bits. Até hoje não é conhecido um algoritmo rápido que resolva este problema em um computador clássico. Mas dados os fatores primos, é fácil de verificar se são a solução verdadeira. Outro problema classicamente intratável, e também bastante conhecido, que pertence à classe NP, é o problema do caixeiro viajante (encontrar o caminho cíclico mínimo conectando n cidades com distâncias especificadas uma da outra). Em particular, o problema do caixeiro viajante é um problema *NP-completo*; isto é, qualquer problema da classe NP pode ser transformado em um exemplo de problema do caixeiro viajante em tempo polinomial. Isto significa que se conseguirmos resolver o problema do caixeiro viajante em tempo polinomial poderemos resolver *qualquer* problema NP em tempo polinomial.

Com o advento da teoria da computação quântica, percebeu-se que o extraordinário poder computacional dos computadores quânticos poderia ser usado para tentar resolver problemas que classicamente são intratáveis. Isto ficou evidente quando o algoritmo quântico de Shor (a ser apresentado na próxima seção) provou que a simulação de um problema de tempo não polinomial é possível em um computador quântico. Neste caso, a teoria que se conhece hoje da complexidade computacional deve ser repensada. Por exemplo, podemos definir um nova *classe BQP* (de *bounded quantum polynomial*) como sendo a classe de todos os problemas computacionais que podem ser resolvidos eficientemente em um computador quântico com um determinado limitante para a probabilidade de erro. A Figura 1.2 mostra a suposta relação entre as classes computacionais clássicas P e NP e a classe computacional quântica BQP. Exatamente onde BQP está com respeito a P e NP ainda é algo desconhecido.

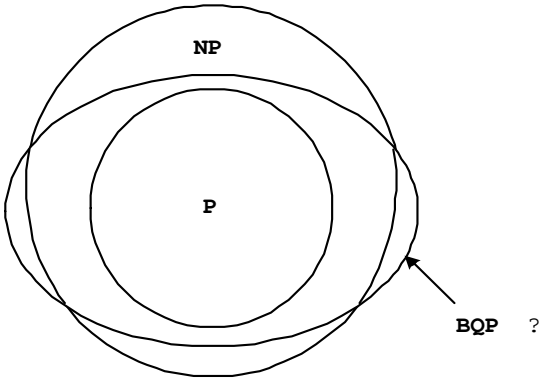

Figura 1.2: Suposta relação entre as classes computacionais clássicas P e NP e a classe computacional quântica BQP.

Se a classificação quântica da complexidade é realmente diferente da classificação clássica (como suspeitado, embora não provado), então este resultado irá "estremecer" os fundamentos da ciência da computação. A longo prazo, poderíamos esperar por novas revoluções tecnológicas, com consequências (certamente boas e más) imprevisíveis.

Mas qual seria o significado desta nova teoria da complexidade para a física ?

1.7 Algoritmos Quânticos

If computers that you build are quantum,
Then spies everywhere will all want'em.
Our codes will fail,
And they'll read our email,
Till we get crypto that's quantum, and daunt'em.

−Peter Shor.

Vimos nas seções anteriores que um computador quântico pode, de fato, ser muito mais poderoso que um computador clássico. Isto abre a possibilidade de alguns problemas intratáveis classicamente poderem se tornar tratáveis quanticamente. O exemplo mais espetacular disto foi apresentado por Peter Shor em 1994 [65].

Shor demonstrou que, pelo menos em princípio, um computador quântico pode *fatorar* eficientemente um número grande. O problema da fatoração (ou seja, da decomposição de um número em fatores primos) é um exemplo de problema possivelmente intratável classicamente com as seguintes propriedades:
− Pode-se *verificar facilmente* se a solução encontrada é verdadeira.
− Mas a solução é *difícil* de ser encontrada.

Isto é, se p e q são números primos grandes, o produto $n = pq$ pode ser computado em pouco tempo (o número de operações elementares de bit é aproximadamente $\log_2 p \cdot \log_2 q$). Mas dado n, é difícil encontrar p e q.

Acredita-se que o tempo exigido para encontrar os fatores primos seja *superpolinomial* em $\log n$ (embora isto nunca tenha sido provado). Ou seja, à medida que n aumenta, o tempo necessário no pior dos casos cresce mais rápido que qualquer potência de $\log n$. O melhor algoritmo de fatoração conhecido requer

$$t \simeq \exp[c(\ln n)^{1/3}(\ln \ln n)^{2/3}], \tag{1.29}$$

onde $c = (64/9)^{1/3} \sim 1.9$. A tecnologia atual permite que uma rede de computadores encontre os fatores primos de 65 dígitos de um número de 130 dígitos em cerca de um mês. Usando este resultado para estimar o pré-fator de (1.29), podemos estimar que a fatoração

de um número com 400 dígitos levaria cerca de 10^{10} anos, a idade estimada do universo! Portanto, mesmo com todos os avanços tecnológicos, a fatoração de um número com 400 dígitos está fora de questão pelas próximas décadas.

O problema da fatoração é interessante sob a perspectiva da teoria da complexidade, como um exemplo de problema presumidamente intratável; isto é, um problema que não pode ser resolvido em um tempo limitado por um polinômio função do tamanho da entrada, neste caso $\log n$. Também é um problema de importância prática, porque a dificuldade de fatoração é a base para os esquemas de criptografia de chave pública, tal como o esquema RSA (Rivest, Shamir, Adlemam).

O resultado apresentado por Shor permite que um computador quântico possa fatorar em um tempo polinomial, *e.g.*, $O[(\ln n)^3]$. Portanto, se tivéssemos um computador quântico que pudesse fatorar um número com 130 dígitos em um mês, o algoritmo de Shor poderia fatorar um número com 400 dígitos em menos de 3 anos. E quanto mais difícil o problema, maior é a vantagem oferecida pelo computador quântico que executa este algoritmo.

Lembremo-nos de que um algoritmo quântico gera uma *distribuição de probabilidades* das possíveis soluções. Assim, não se pode garantir que o algoritmo de fatoração de Shor tenha sucesso ao encontrar os fatores primos. Apenas é um algoritmo que obtém sucesso com uma probabilidade razoável. Mas continua sendo um algoritmo muito válido porque é fácil de verificar se os fatores encontrados são realmente a solução do problema.

Outro algoritmo quântico impressionante é o algoritmo proposto por Grover em 1996 [27]. Este algoritmo pode encontrar um elemento em um banco de dados não ordenado de n elementos em um tempo $O(\sqrt{n})$. Em um computador clássico, a mesma tarefa exigiria uma procura exaustiva de tempo $O(n)$. Foi mostrado que o tempo $O(\sqrt{n})$ é o menor possível para esta tarefa [5].

O algoritmo de Grover poderá ser utilizado para atacar o sistema criptográfico DES (*Data Encryption Standard*) que apresenta $2^{56} = 7 \times 10^{16}$ caminhos possíveis. Se um computador clássico executando um algoritmo puder verificar 1 milhão de caminhos por segundo, levará 100 anos para descobrir o caminho correto, enquanto que o algoritmo de Grover pode obter tal resultado em menos de 4 minutos. Por esta razão, o algoritmo de Grover também terá grande aplicação em protocolos de criptografia quântica [13].

Os resultados encontrados por Shor e Grover estimularam o interesse da comunidade científica pela teoria da computação quântica. É realmente espantoso imaginar as implicações destes resultados para a teoria da complexidade, para a teoria quântica e para a tecnologia em geral.

1.8 Hardware Quântico

Computers in the future may weight no more than 1.5 tons.
– Thomas Watson, chairman of IBM, 1943.

... quantum phenomena do not occur in a Hilbert space, they occur in a laboratory.
– Asher Peres.

Os desenvolvimentos da teoria da complexidade quântica e da correção de erros quânticos têm sido acompanhados por numerosos esforços de físicos em busca de técnicas experimentais que permitam o processamento de informação quântica.

A descrição de tais experimentos está fora do escopo desta tese, dada a sua extensão e complexidade. Não obstante, faremos a seguir um breve resumo das condições necessárias para a construção física de um computador quântico. Maiores detalhes podem ser encontrados em textos (veja, por exemplo, o capítulo 7 de [48]) ou em numerosos artigos citados nestes mesmos textos.

Para construirmos um hardware para um computador quântico, precisamos de uma tecnologia que nos permita manipular os qubits. O hardware necessita obrigatoriamente de algumas especificações, a saber:

1. **Armazenamento:** Precisamos armazenar qubits por um longo período de tempo, longo o suficiente para completar uma computação interessante.

2. **Isolamento:** Os qubits devem ser bem isolados do meio, para minimizar erros de decoerência.

3. **Leitura:** Precisamos medir os qubits eficientemente e corretamente.

4. **Portas:** Precisamos manipular os estados quânticos de qubits individuais, e induzir interações controladas entre os qubits, de modo a permitir a implementação de portas quânticas.

5. **Precisão:** As portas quânticas devem ser implementadas com alta precisão para que o sistema possa computar corretamente.

Existem muitas propostas de como implementar um computador quântico na prática. Três das principais realizações físicas de qubits nos últimos 10 anos foram: a polarização de fótons, com interações via cavidade óptica [42]; o estado fundamental e excitado de íons armazenados em uma armadilha de íons linear, com interações fornecidas através de um modo de vibração comum [9, 29]; e os estados do spin nuclear em polímeros, com interações dadas por técnicas de ressonância magnética nuclear [23].

1.9 Ruído Quântico

When I hear about Schrödinger's cat, I reach for my gun.

−Stephen Hawking.

Todas as implementações físicas acima citadas para os computadores quânticos possuem uma susceptibilidade para erros muito mais alta do que qualquer computador clássico moderno. Futuros desenvolvimentos tecnológicos podem reduzir os erros de várias ordens de grandeza, mas é improvável que os computadores quânticos alcancem a inacreditável confiabilidade dos computadores clássicos.

Como temos notado, a propriedade essencial da informação quântica que um computador quântico explora é a existência de correlações não-locais entre as diferentes partes de um sistema físico. Se olharmos somente para uma parte do sistema em um instante de tempo, podemos decifrar muito pouco da informação codificada no sistema.

Infelizmente, estas correlações não-locais são extremamente frágeis e tendem a desaparecer muito rapidamente quando implementadas na prática. O problema é que o sistema quântico está inevitavelmente em contato com um sistema muito maior, o meio ambiente circundante. É impossível isolar perfeitamente um grande sistema quântico do ambiente

que o cerca, mesmo que façamos um esforço heróico para isto. Interações entre um sistema quântico e o meio circundante estabelecem correlações não-locais entre os dois. Assim, eventualmente, a informação quântica que inicialmente codificamos no sistema é codificada em correlações entre o sistema e o meio circundante. Neste estágio, não podemos mais acessar a informação, somente através da observação do computador. Na prática, a informação é irreversivelmente perdida. Mesmo que o acoplamento entre o computador e o meio seja fraco, isto ocorrerá muito rapidamente.

Em 1935, Erwin Schrödinger lançou dúvidas sobre as interpretações dadas àquela época à mecânica quântica. Ele percebeu que existia um aparente paradoxo em um experimento hipotético envolvendo um *gato* dentro de uma caixa fechada [74][9]. Ele observou que a teoria quântica permite o estado quântico de um gato da forma

$$|\text{gato}\rangle = \frac{1}{\sqrt{2}}(|\text{morto}\rangle + |\text{vivo}\rangle). \tag{1.30}$$

Para Schrödinger, isto mostrava que a mecânica quântica não podia ser uma teoria completa, já que não faz sentido afirmar que um gato está vivo e morto ao mesmo tempo.

Um dos avanços mais importantes da teoria quântica nos últimos vinte anos é que aprendemos a responder com maior segurança à pergunta de Schrödinger. O estado $|\text{gato}\rangle$ é possível em princípio, mas raramente é visto porque é *extremamente* instável. Os gatos que Schrodinger observou nunca foram bem isolados do meio. Se alguém fosse preparar o estado $|\text{gato}\rangle$, a informação codificada na superposição de $|\text{morto}\rangle$ e $|\text{vivo}\rangle$ seria *imediatamente*

[9]Uma breve descrição deste experimento: Imagine um gato trancado em uma caixa com um vidro de cianeto. O vidro de cianeto está ligado a um aparelho que contém uma pequena amostra de uma substância radioativa, um contador Geiger e um martelo. Schrödinger ressalta o fato de que o aparelho deve estar "protegido contra uma intervenção direta por parte do gato". A quantidade de material radioativo deve ser suficientemente pequena para que em um intervalo de uma hora a probabilidade de que um dos átomos da amostra se desintegre seja de 50%. Se um dos átomos se desintegrar, o contador Geiger será disparado, ativando um relé e liberando o martelo. O martelo, por sua vez, quebrará o vidro, liberando o veneno, que matará o gato. Depois que o gato é trancado na caixa, ninguém sabe em que momento um dos átomos da amostra se desintegrará. A questão é a seguinte: o gato estará vivo ou morto depois de uma hora ? A resposta correta é a seguinte: é impossível saber, a não ser abrindo a caixa e examinando o interior. De acordo com a descrição quântica, o gato se encontra em uma espécie de limbo, meio morto e meio vivo, até alguém abrir a caixa.

transferida para correlações entre o gato e o meio, e tornar-se-ía completamente inacessível. Na verdade, o meio continuamente "mede" o gato, projetando-o sobre o estado $|\text{morto}\rangle$ ou sobre o estado $|\text{vivo}\rangle$. Este processo é chamado de *decoerência*.

Para realizar uma computação quântica complexa, precisamos preparar uma delicada superposição de estados de um sistema quântico relativamente grande (embora, talvez, não tão grande quanto um gato!). Infelizmente, este sistema não pode ser perfeitamente isolado do meio, portanto esta superposição, assim como o estado $|\text{gato}\rangle$, decai muito rapidamente. A informação quântica é rapidamente perdida, e nosso computador quântico é destruído.

Dito de outra forma, o contato entre o computador e o meio (decoerência) causa *erros* que degradam a informação quântica. Para operar um computador quântico com confiança, devemos encontrar algum meio de impedir ou corrigir estes erros.

Na verdade, a decoerência não é o nosso único problema. Mesmo se conseguíssemos um isolamento perfeito do sistema, não poderíamos esperar operar um computador quântico com grande precisão. Isto porque também temos o problema dos erros intrínsecos às portas quânticas que realizam a computação. As portas quânticas são transformações unitárias que operam sobre poucos qubits por unidade de tempo, como por exemplo matrizes unitárias atuando sobre dois qubits. Estas matrizes unitárias formam um *continuum*. Podemos, por exemplo, construir um protocolo para a aplicação de U_0 sobre dois qubits, mas nossa execução nunca será perfeita. A transformação real

$$U = U_0(1 + O(\varepsilon)) \tag{1.31}$$

irá diferir da transformação pretendida U_0 por alguma quantidade da ordem de ε. Após cerca de $1/\varepsilon$ portas serem aplicadas, estes erros irão se acumular e induzir uma falha que comprometerá a computação.

No caso dos computadores clássicos, foi desenvolvido ao longo das últimas décadas um sofisticado mecanismo para corrigir erros durante a transmissão de informação, *a teoria de códigos corretores de erros*. Na década de 50, John Von Neumann mostrou que um computador clássico com componentes ruidosos pode trabalhar com confiabilidade empregando suficiente redundância. Ele apontou que, se necessário, podemos computar cada porta lógica muitas vezes, e aceitar o resultado majoritário.

Os computadores clássicos atuais protegem-se de erros sobretudo por serem máquinas digitais ao invés de analógicas - ao invés de permitir que cada bit do computador varie continuamente entre 0 e 1, a cada unidade temporal o hardware "chuta" o bit de volta para o mais perto de 0 e 1. Isto impede que pequenos erros se tornem grandes erros. Desta forma, os erros são drasticamente reduzidos.

A mesma técnica não pode ser usada em um computador quântico porque continuamente a medida de cada qubit destrói os estados emaranhados que distinguem um computador quântico de um computador clássico. Os estados emaranhados são em geral muito delicados, e ao fazermos uma medida em um deles iremos tipicamente colapsá-lo para um estado menos emaranhado. Pequenas interações com o ambiente fornecem uma espécie de medida contínua do sistema, e à medida que o sistema cresce em tamanho, os erros tornam-se mais e mais difíceis de serem ignorarados. O sistema sofre então decoerência e começa a se comportar como um computador clássico. A decoerência é a razão pela qual o mundo parece clássico à escala humana. Reduzir as interações com o meio pode reduzir os efeitos da decoerência, mas não eliminá-los inteiramente.

Mesmo se a taxa de erro em um computador quântico puder ser reduzida para algum valor ε por unidade de tempo, após n unidades temporais, a probabilidade de sobrevivência sem erro é de somente $(1-\varepsilon)^n$, a qual decresce exponencialmente com n. Mesmo que um algoritmo seja executado em tempo polinomial em um computador livre de erros, precisaremos executá-lo exponencialmente muitas vezes em um computador "real" a menos que algo possa ser feito para controlar os erros.

O mesmo problema ocorre com os computadores clássicos. Em princípio, erros clássicos podem ser corrigidos com o uso de códigos corretores de erros. Na prática, estes códigos não são usualmente necessários para operações normais de um computador, mas são essenciais para corrigir erros em canais de comunicação. Apresentaremos uma introdução básica à teoria dos *códigos corretores de erros clássicos* (CCECs) nos Capítulos 2 e 3.

As técnicas de correção de erros clássicos não podem ser diretamente adaptadas para os computadores quânticos. As técnicas clássicas assumem que medimos todos os bits dos computadores. Para os computadores quânticos, isto destruiria qualquer emaranhamento entre os qubits. Além disso, um computador clássico precisa somente preservar os valores

dos bits 0 e 1. Um computador quântico precisa também manter a fase da informação nos estados emaranhados. Apesar das dificuldades aparentemente intransponíveis, é possível, como veremos no próximo capítulo, construir *códigos corretores de erros quânticos* (CCEQs).

Tendo em vista tudo o que foi apresentado ao longo deste capítulo, estamos agora em condições de enumerar explicitamente os desafios a serem superados durante a construção de CCEQs. A saber:

1. **Erros de fase**. Com a informação quântica mais coisas podem dar erradas. Além dos erros *bit flip*,

$$\begin{cases} |0\rangle \rightarrow |1\rangle \\ |1\rangle \rightarrow |0\rangle \end{cases} \tag{1.32}$$

   podem existir erros *phase flip*:

$$\begin{cases} |0\rangle \rightarrow |0\rangle \\ |1\rangle \rightarrow -|1\rangle \end{cases} \tag{1.33}$$

   Um erro phase flip é grave, porque faz o estado $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ transformar-se no estado ortogonal $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. A codificação clássica não oferece proteção contra erros desta natureza.

2. **Erros pequenos**. Como observado anteriormente, a informação quântica é contínua. Se pretendemos que um qubit esteja no estado $a|0\rangle + b|1\rangle$, um erro pode mudar a e b por uma quantidade da ordem de ε, e estes pequenos erros podem se acumular ao longo do tempo. Os esquemas clássicos são designados para corrigir apenas erros (bit flip) grandes.

3. **Medida causa destruição do estado original**. No esquema clássico de votação majoritária, precisamos medir os bits no código para detectar e corrigir os erros. Mas não podemos medir qubits sem destruir a informação quântica que codificamos.

4. **Não-Clonagem**. Com a codificação clássica protegemos a informação fazendo cópias extras dela. Mas sabemos que a informação quântica não pode ser copiada com perfeita fidelidade.

1.10 Sumário

Isto conclui nossa introdução geral à teoria da computação e da informação quântica. Vimos que três fatores convergentes foram combinados para tornar este tópico de pesquisa extremamente excitante. A saber:

1. Os computadores quânticos podem resolver problemas difíceis. Parece que uma nova classificação para a complexidade emergiu, uma classificação melhor fundada nas leis fundamentais da física do que a teoria da complexidade tradicional (embora seja necessário caracterizar mais precisamente a classe de problemas para os quais os computadores quânticos apresentam uma grande vantagem sobre os computadores clássicos).

2. O ruído quântico decorrente da decoerência e da imperfeição das portas lógicas que executam uma computação constitui um grave problema para a conclusão de uma computação longa e eficiente. Veremos no próximo capítulo que é possível construir esquemas de codificação para proteger um sistema quântico dos efeitos debilitantes da decoerência. Nunca poderemos ver se um gato está metade vivo e metade morto, mas talvez, possamos preparar e preservar um "gato codificado" que é metade morto e metade vivo!

3. O hardware quântico pode ser construído. Nossa geração de pesquisadores verá o florescimento da manipulação da informação quântica coerente em laboratório.

1.11 Conteúdo e Estrutura da Tese

A primeira parte desta tese descreve os conceitos fundamentais necessários ao desenvolvimento de CCEQs. Por ser uma área de pesquisa de forte caráter interdisciplinar,

faz-se necessária uma apresentação de material preliminar mais longa e didática que o usual.

O **Capítulo 1** apresentou uma introdução geral à teoria da computação e da informação quântica. Julgamos que esta introdução foi útil para que pudéssemos contextualizar o problema da correção de erros em canais quânticos ruidosos dentro da engenharia quântica.

O **Capítulo 2** apresentará uma introdução à teoria dos CCEQs com destaque para uma classe de códigos cuja construção é análoga à dos códigos lineares clássicos − a *classe dos códigos estabilizadores*.

Sabemos que a classe dos códigos lineares clássicos possui duas grandes subclasses: a classe dos códigos de bloco clássicos (CBCs) e a classe dos códigos convolucionais clássicos (CCCs). Na codificação em bloco, cada estado é primeiramente dividido em blocos *finitos* de mesmo comprimento e então é codificado separadamente usando um código que é *independente* do estado dos outros blocos. Já na codificação convolucional, a operação de codificação depende do estado corrente e de alguns estados passados, ou seja, a codificação envolve *memória*. Sabe-se que a classe dos CCCs possui pelo menos duas grandes vantagens em relação à classe dos CBCs: permite a codificação de uma sequência contínua de informação (em princípio, infinita) com uma complexidade exponencial de menor taxa e apresenta melhor performance de correção. Estes conceitos serão explorados em detalhes ao longo dos próximos capítulos.

A proposta desta tese é apresentar uma classe de *códigos convolucionais quânticos* (CCQs) estabilizadores, explicitando o esquema de *geração* e de *decodificação* destes códigos. Os CCQs pertencentes a esta classe são construídos a partir da *concatenação* de um CCQ que corrige erros phase flip com um CCQ que corrige erros bit flip, e são capazes de corrigir erros quânticos arbitrários. Esta técnica de construção é análoga à usada para a construção do código de Shor, o mais simples dos *códigos de bloco quânticos* (CBQs). Um dos objetivos da construção de CCQs é verificar se apresentam uma performance de correção superior aos CBQs.

O **Capítulo 3** apresenta uma revisão geral da classe dos CCCs. Os conceitos clássicos embutidos neste capítulo são necessários para a compreensão dos esquemas de geração e de

decodificação dos CCQs. Ao final deste capítulo terminamos a primeira parte da tese, que é a de exposição dos conceitos fundamentais, e damos início à segunda parte, que inclui os capítulos que apresentam as contribuições à teoria conhecida.

O **Capítulo 4** expõe detalhadamente o processo de geração e de decodificação do CCQ concatenado com construção mais simples: um CCQ com taxa 1/4 e três memórias quânticas, capaz de corrigir um erro quântico arbitrário. Este CCQ é construído a partir de um único CCC de taxa 1/2 e duas memórias.

O **Capítulo 5** estende os resultados obtidos no Capítulo 4 para outras taxas, memórias e distâncias quânticas. Também descreve outras técnicas de construção de bons CCQs concatenados a partir de CCs conhecidos. Dentre as técnicas discutidas estão a construção a partir de dois CCCs distintos, a construção a partir de CCCs puncionados e a construção a partir de CCCs ótimos.

O **Capítulo 6** apresenta as conclusões e considerações finais. Também são apresentadas futuras linhas de pesquisa a serem exploradas dentro da classe dos CCQs.

O **Apêndice A** apresenta as soluções da equação de Diofanto para duas variáveis, necessárias ao algoritmo de decodificação por síndromes para CCQs construídos a partir de CCCs com taxa 1/2. O **Apêndice B** apresenta uma lista de endereços eletrônicos para consulta de referências na Internet.

Ao final, apresentamos a **Bibliografia**.

Capítulo 2

Correção de Erros Quânticos

Anything that can go wrong, will.

– "Lei" atribuída a Edward A. Murphy, Jr.

Afinal, os computadores quânticos podem funcionar na prática ?

Vimos no capítulo anterior que o processamento de informação quântica é freqüentemente descrito como uma série de operações unitárias e de medidas em algum sistema físico. Imperfeições nestas operações e interações com o meio ambiente circundante (decoerência) são inevitáveis tanto no processamento de informação clássica quanto quântica. O acúmulo de erros será prejudicial em qualquer processamento de informação de larga escala.

Portanto, para que um computador quântico funcione na prática, temos um grande obstáculo a superar: proteger a informação quântica de erros. Sem esta proteção, nosso computador quântico certamente irá falhar. Qualquer estratégia efetiva para impedir que erros ocorram em um computador quântico deve proteger contra pequenos erros unitários em um circuito quântico, bem como contra erros de decoerência.

Contra os erros de decoerência foi desenvolvida a teoria dos CCEQs, a ser introduzida neste capítulo junto com a principal classe de códigos, a classe dos códigos estabilizadores, e alguns dos principais códigos desta classe. Quanto aos erros operacionais, intrínsecos às portas lógicas, foi desenvolvida a *teoria quântica de tolerância a falhas*, que não será abordada aqui (ver, por exemplo, [48]).

Um CCEQ pode ser visto como um mapeamento de k qubits (um espaço de Hilbert com 2^k dimensões) em n qubits (um espaço de Hilbert com 2^n dimensões), onde $n > k$. Os k qubits são os *qubits lógicos* que desejamos proteger de erros. Os n qubits são os *qubits físicos* resultantes da codificação. Os $n-k$ qubits adicionais permitem-nos armazenar os k qubits lógicos de uma forma redundante tal que os n qubits físicos não sejam facilmente alterados.

Antes de iniciar o estudo da teoria dos CCEQs e de alguns de seus exemplos, é imperioso que se faça uma revisão de elementos básicos da teoria clássica de codificação, já que os esquemas de codificação e de decodificação dos códigos estabilizadores tomam os CCECs como base para a sua construção.

2.1 Elementos da Teoria Clássica de Codificação

2.1.1 Conceitos Fundamentais

Quando a informação é armazenada ou enviada através de um canal não confiável, é possível representá-la com *redundância* para que possamos aumentar a probabilidade de recuperação da informação original. O conjunto de todas as possíveis mensagens codificadas (palavras-código) formam um *código corretor de erros*.

Vamos ilustrar os conceitos básicos da teoria clássica de codificação através de um exemplo simples, o mesmo que será posteriormente utilizado para a construção do mais simples dos códigos quânticos, o código de Shor, e para a construção de toda uma classe de CCQs.

Considere o *canal binário simétrico* (BSC), no qual um bit b é trocado por $\overline{b}$ com probabilidade p. Este canal é simétrico com respeito a $b = 0$ e 1. Veja a Figura 2.1.

Erros em diferentes usos deste canal são *independentes*, ou seja, este é um canal *sem memória*. Suponha que p seja pequeno, de forma a ser improvável que ocorra mais do que um erro na transmissão. Suponha também que enviamos b três vezes e decodificamos a sequência na saída do canal por votação majoritária. Esta técnica de decodificação falha se ocorrerem dois ou mais erros durante a transmissão. É fácil de verificar que a probabilidade disto ocorrer é $p_e = 3p^2(1-p) + p^3 = 3p^2 - 2p^3$. Sem esta codificação, a probabilidade de

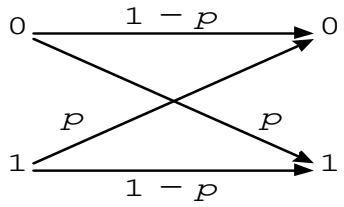

Figura 2.1: O canal binário simétrico.

ocorrer um erro seria p. Portanto, o uso do código permite uma transmissão mais segura já que sempre $p_e < p$ para $p < 0.5$. Este código é chamado de *código de repetição de três bits*.

Podemos melhorar ainda mais a confiabilidade se usarmos um código mais longo. Um código deste tipo (embora longe de ser o mais eficiente) é um código de repetição de n bits. A distribuição de probabilidade para o valor médio do bit, pelo teorema central do limite, aproxima-se de uma curva de Gauss com largura $1/\sqrt{n}$ à medida que $n \to \infty$. Se $p = 1/2 + \varepsilon$ é a probabilidade de cada bit ter o valor correto, então a probabilidade que a decodificação por votação majoritária falhe (para n grande) é

$$p_e \sim e^{-n\varepsilon^2}. \tag{2.1}$$

Portanto, com a introdução de redundância suficiente, podemos obter confiabilidade arbitrariamente boa para qualquer $\varepsilon > 0$. Mesmo para $\varepsilon < 0$, será vantajoso se assumirmos sempre que a votação majoritária dá o resultado *errado*. Somente para $p = 1/2$ é que teremos problemas, pois nosso bloco de n bits será *aleatório*, e não codificará informação.

O código de repetição ilustra alguns conceitos gerais da teoria de codificação, a saber:

1. Devemos assumir certas características a respeito do ruído do canal. No exemplo acima, assumimos que o canal é simétrico, binário e sem memória.

2. Dado um canal, as palavras-código são escolhidas dentro de um espaço *maior* que o requerido para um uso do canal, tal que os prováveis erros levem as palavras-código originais a conjuntos disjuntos de mensagens recebidas para assegurar a correta decodificação destas mensagens. Para o código de repetição de três bits, isto é melhor

ilustrado através da Figura 2.2.

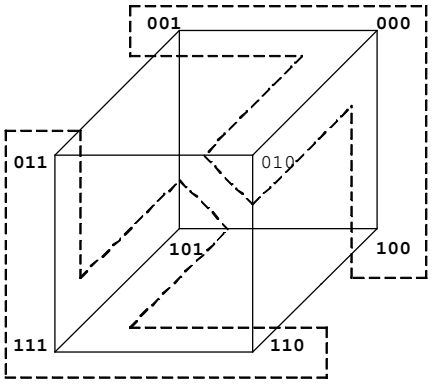

Figura 2.2: A geometria do código de repetição de três bits. Os erros prováveis levam as palavras-código 000 e 111 a dois conjuntos disjuntos de mensagens definidos pelas linhas tracejadas.

3. A codificação reduz a probabilidade de erro à custa de mais usos do canal. Se M palavras-código são codificadas em n bits, a *taxa* do código é $(\log_2 M)/n$. Por exemplo, o código de repetição de três bits tem taxa 1/3.

Na maior parte dos casos estaremos interessados em tratar apenas de erros independentes com pequena probabilidade p de ocorrência. Além disso, a codificação será sempre destinada a tratar até um certo número de erros. Por exemplo, o código de repetição de três bits é um código que corrige somente um erro. Similarmente, um código que corrige até t erros pode falhar se $t+1$ ou mais erros ocorrerem. A probabilidade total de erro é melhorada de p para $O(p^{t+1})$.

O esquema de codificação envolve uma relação entre a taxa do código e a confiabilidade da transmissão ou do armazenamento de informação, a qual pode ser melhor compreendida se estudarmos alguns parâmetros definidos na próxima seção.

2.1.2 A Geometria da Correção de Erros

Considere que o conjunto de símbolos seja um corpo F, $|F| = q$, e que as mensagens de comprimento n sejam vetores em F^n. No caso geral, F é um corpo arbitrário, mas

ocasionalmente restringiremo-nos ao corpo binário a fim de que haja simplificação na notação.

Definição 1. *A distância de Hamming entre dois vetores* $\mathbf{a}$ *e* $\mathbf{b}$*,* $d_H(\mathbf{a},\mathbf{b})$*, é o número de coordenadas em que os dois vetores diferem entre si.*

Note que a distância de Hamming é uma *métrica*. Chamamos o conjunto de pontos $\{\mathbf{b} : d_H(\mathbf{a},\mathbf{b}) \leq r\}$ de esfera de Hamming de raio r centrada em $\mathbf{a}$.

Definição 2. *Um código* (n,M,d) *é um conjunto de* M *vetores (palavras-código) em* F^n *com distância* d*. O parâmetro* n *é chamado de comprimento do bloco do código.*

Definição 3. *A distância de um código* $\mathcal{C}$ *é a distância de Hamming mínima entre quaisquer duas palavras-código,* $d = \min\{d_H(\mathbf{a},\mathbf{b}) : \mathbf{a} \neq \mathbf{b};\ \mathbf{a},\mathbf{b} \in C\}$.

Suponha que não ocorram mais do que r erros sobre a palavra-código durante a transmissão pelo canal. Então, cada palavra-código recebida à saída do canal será um ponto dentro de sua própria esfera de Hamming de raio r. Estas esferas de Hamming não se interceptam se $2r < d$ e, portanto, a palavra-código pode ser decodificada corretamente. Veja a ilustração desta situação na Figura 2.3.

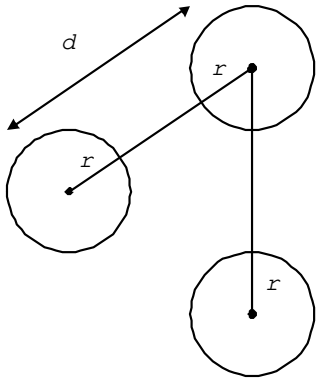

Figura 2.3: Esferas de Hamming de raio r para $2r < d$.

Teorema 1. *Um código com distância* d *é capaz de corrigir até* t *erros, onde* $t = \lfloor \frac{d-1}{2} \rfloor$[1].

[1] Ou seja, o maior inteiro menor ou igual a $\frac{d-1}{2}$.

Se imaginarmos um código (n, M, d) como um conjunto de pontos em F^n, o problema da codificação transforma-se no problema de empacotar tantos pontos quanto possíveis mantendo uma certa distância entre os pontos.

O *Limitante de Hamming* é um importante limitante que relaciona a distância e a taxa de um código. Para uma mensagem de comprimento n, uma esfera de Hamming de raio r tem $V(r) = \sum_{i=0}^{r} \binom{n}{i} (q-1)^i$ elementos em seu interior. Para um código corretor de t erros, as esferas de Hamming de raio t ao redor das palavras-código não se interceptam.

Lema 1. *(Limitante de Hamming) Seja C um código (n, M, d), então*

$$n - \log_q M \geq \log_q V\left(\lfloor \frac{d-1}{2} \rfloor\right). \tag{2.2}$$

Códigos que saturam o limitante de Hamming são chamados de códigos perfeitos.

Códigos corretores de erros também podem ser usados para detectar erros. O decodificador verificará se a mensagem recebida é ou não uma palavra-código válida. Para um código (n, M, d), são necessários d erros para se evitar a detecção. Portanto, um código (n, M, d) pode detectar $d-1$ erros. Em geral, um código (n, M, d) pode corrigir t erros e detectar até $t+t'$ erros para $2t+t' < d$. Isto é melhor compreendido com o auxílio da Figura 2.4.

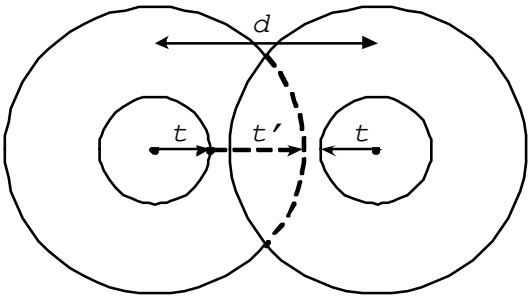

Figura 2.4: Esferas de Hamming e detecção de erros: um código pode corrigir t erros e detectar até $t+t'$ erros para $2t+t' < d$.

Na próxima seção, descreveremos uma classe especial de códigos clássicos, os códigos lineares, que possuem muitas propriedades simples. Veremos também que existe uma elegante teoria que relaciona esta classe de códigos clássicos aos códigos quânticos.

2.1.3 Códigos Lineares Clássicos

Qualquer código corretor de erros $\mathcal{C}$ é um *subconjunto* de F^n, o espaço vetorial de n dimensões sobre F. $\mathcal{C}$ é chamado de *linear* se for um *subespaço* de F^n. Em outras palavras,

1. $\mathcal{C} \neq \phi$ (o conjunto vazio).
2. $\mathbf{a}, \mathbf{b} \in \mathcal{C};\ \alpha, \beta \in F \Rightarrow \alpha\mathbf{a} + \beta\mathbf{b} \in \mathcal{C}$.

Se $\dim(\mathcal{C}) = k$, $\mathcal{C}$ é um código (n, q^k, d). Quando $q = 2$, k bits são codificados em n bits. O código de repetição de três bits descrito na seção anterior é o código linear mais simples que pode ser construído, embora não seja o mais eficiente para a maioria das aplicações práticas.

Definição 4. *O peso de um vetor* $\mathbf{a}$ *é o número de componentes não nulas. Esta quantidade é denotada por* $w_H(\mathbf{a})$ *e é igual a* $d_H(\mathbf{a}, \mathbf{0})$ *(onde* $\mathbf{0}$ *é o vetor nulo).*

Teorema 2. *A distância de um código linear* $\mathcal{C}$ *é igual ao seu peso mínimo, ou seja, o mínimo dos pesos de suas palavras-código não nulas.*

Um código linear corretor de erros pode ser definido por sua *matriz geradora* ou por sua *matriz verificação de paridade*. Seja $\mathcal{C}$ um código (n, k, d). Uma matriz $\mathbf{G}$ com dimensões $k \times n$ é uma matriz geradora de $\mathcal{C}$ se as linhas de $\mathbf{G}$ formam uma base para $\mathcal{C}$ (ou seja, as palavras-código são os vetores-linha de $\mathbf{G}$). Uma mensagem original $\mathbf{u}$ é codificada como $\mathbf{v}$ onde $\mathbf{v} = \mathbf{uG}$. Uma matriz $\mathbf{H}$, com dimensões $(n-k) \times n$, é uma matriz verificação de paridade para $\mathcal{C}$ se, e somente se, para qualquer $\mathbf{v} \in \mathcal{C}$, $\mathbf{vH}^T = 0$. Como consequência, as linhas de $\mathbf{H}$ formam uma base para o complemento ortogonal de $\mathcal{C}$. Note que $\mathbf{GH}^T = 0$.

Por exemplo, para o código de repetição de três bits temos:

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \quad \text{e} \quad \mathbf{H} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}. \tag{2.3}$$

Assim, $\mathbf{G}$ mapeia as possíveis mensagens, 0 e 1, em $\mathbf{v} = (0) \cdot (1,1,1) = (0,0,0)$ e $\mathbf{v} = (1) \cdot (1,1,1) = (1,1,1)$. Para construirmos $\mathbf{H}$ basta escolhermos $3 - 1 = 2$ vetores linearmente independentes e ortogonais à linha de $\mathbf{G}$, como por exemplo $(1,1,0)$ e $(0,1,1)$. Note que, de fato, $\mathbf{vH}^T = 0$ *somente* para as palavras-código $\mathbf{v} = (0,0,0)$ e $\mathbf{c} = (1,1,1)$.

O código dual de $\mathcal{C}$, denotado por $\mathcal{C}^{\perp}$, é o complemento ortogonal de $\mathcal{C}$ em F^n. $\mathcal{C}^{\perp}$ é um código $(n,\, n-k,\, d^{\perp})$ com matriz geradora $\mathbf{H}$ e matriz verificação de paridade $\mathbf{G}$. Em geral, não existe uma relação simples entre d e $d^{\perp}$. Se um código é auto-dual, $d = d^{\perp}$. Se um código contém seu dual, $d^{\perp} \geq d$. Contra-intuitivamente, um vetor *binário* é auto-dual se tiver peso par, e é possível que haja intersecção entre $\mathcal{C}$ e $\mathcal{C}^{\perp}$.

O limitante de Hamming para um código $(n,\, k,\, d)$ é dado por

$$n-k \geq \log_q V\left(\lfloor \frac{d-1}{2} \rfloor\right). \tag{2.4}$$

Para um código definido sobre $GF(2)$ (o corpo de Galois com dois elementos) que corrija t erros, o limitante (2.4) reduz-se a

$$\sum_{j=0}^{t} \binom{n}{j} 2^k \leq 2^n. \tag{2.5}$$

Quando n, k e t são grandes, este limitante aproxima-se da forma assintótica

$$\frac{k}{n} \leq 1 - H\left(\frac{t}{n}\right), \tag{2.6}$$

onde $H(x)$ é a *entropia binária de Hamming*

$$H(x) = -x\log_2 x - (1-x)\log_2(1-x). \tag{2.7}$$

Outro conceito que será útil para a correção de erros quânticos é o conceito de *síndrome*. Considere $\mathcal{C}$ um código $(n,\, k,\, d)$ com matriz verificação de paridade $\mathbf{H}$. Seja $\mathbf{v}$ a mensagem codificada enviada ao canal e $\mathbf{r}$ a mensagem possivelmente corrompida, que é recebida na saída do canal. A síndrome de $\mathbf{r}$ é definida como sendo $\mathbf{s} = \mathbf{r}\mathbf{H}^T$. Se $\mathbf{e}$ é o vetor erro, $\mathbf{r} = \mathbf{v} \oplus \mathbf{e}$ e

$$\mathbf{s} = \mathbf{r}\mathbf{H}^T = (\mathbf{v}\mathbf{H}^T) \oplus (\mathbf{e}\mathbf{H}^T) = \mathbf{e}\mathbf{H}^T. \tag{2.8}$$

Portanto, a síndrome *não* depende de $\mathbf{v}$ e depende *somente* de $\mathbf{e}$. Existe uma correspondência um-a-um entre os erros de peso $\leq (d-1)/2$ e as síndromes destes mesmos erros, o que possibilita uma *decodificação por máxima verossimilhança*.

Uma *classe lateral* de um código $\mathcal{C}$ é o conjunto de elementos $\mathbf{w} \oplus \mathcal{C}$ para $\mathbf{w} \in F^n$. Um erro $\mathbf{e}$ mapeia o espaço de palavras-código original na classe lateral $\mathbf{e} \oplus \mathcal{C}$. Duas classes laterais $\mathbf{w}_1 \oplus \mathcal{C}$ e $\mathbf{w}_2 \oplus \mathcal{C}$ são iguais se, e somente se, $\mathbf{w}_1$ e $\mathbf{w}_2$ estão na mesma classe lateral. As classes laterais formam uma partição de F^n. Em um código $(n,\ k,\ d)$, existe uma correspondência um-a-um entre síndromes e classes laterais. O elemento de peso mínimo em cada classe lateral (um único elemento de peso $\leq (d-1)/2$) corresponde ao erro mais provável ocorrido quando a classe lateral é identificada. Isto pode ser melhor entendido com o auxílio da tabela de *arranjo padrão* de $\mathcal{C}$, que é uma matriz de dimensões $q^{n-k} \times q^k$ com o formato da Tabela 2.1.

	$\mathbf{u}_0$	$\mathbf{u}_1$	$\mathbf{u}_2$	$\cdots$	$\mathbf{u}_{q^k-1}$
$\mathcal{C}$	$\mathbf{0}$	$\mathbf{v}_1$	$\mathbf{v}_2$	$\cdots$	$\mathbf{v}_{q^k-1}$
$\mathbf{e}_1 \oplus \mathcal{C}$	$\mathbf{e}_1$	$\mathbf{e}_1 \oplus \mathbf{v}_1$	$\mathbf{e}_1 \oplus \mathbf{v}_2$	$\cdots$	$\mathbf{e}_1 \oplus \mathbf{v}_{q^k-1}$
$\mathbf{e}_2 \oplus \mathcal{C}$	$\mathbf{e}_2$	$\mathbf{e}_2 \oplus \mathbf{v}_1$	$\mathbf{e}_2 \oplus \mathbf{v}_2$	$\cdots$	$\mathbf{e}_2 \oplus \mathbf{v}_{q^k-1}$
.	.	.	.	$\cdots$	.
$\mathbf{e}_l \oplus \mathcal{C}$	$\mathbf{e}_l$	$\mathbf{e}_l \oplus \mathbf{v}_1$	$\mathbf{e}_l \oplus \mathbf{v}_2$	$\cdots$	$\mathbf{e}_l \oplus \mathbf{v}_{q^k-1}$

Tabela 2.1: Arranjo padrão para um código (n, k, d). O parâmetro l é dado por $l = q^{n-k} - 1$.

Neste arranjo padrão, as linhas são as classes laterais. Cada erro $\mathbf{e}_i$ faz um mapeamento de $\mathcal{C}$ em $\mathbf{e}_i \oplus \mathcal{C}$, e a i-ésima linha é a imagem de $\mathcal{C}$ sob $\mathbf{e}_i$. A j-ésima coluna pode ser imaginada como a esfera de Hamming ao redor da palavra-código $\mathbf{v}_j$ (aqui os elementos em uma coluna podem não formar um esfera exata). A primeira coluna em cada linha é o erro mais provável referente à síndrome correspondente. Quando $\mathbf{r}$ é recebido na saída do canal, a decodificação por máxima verossimilhança pode ser feita através da leitura do índice da coluna. Equivalentemente, a síndrome pode ser obtida para encontrar o índice da linha de $\mathbf{r}$ e o erro mais provável $\mathbf{e}$. A mensagem enviada, $\mathbf{v}$, pode então ser recuperada através da soma de $\mathbf{e}$ a $\mathbf{r}$, *sem* o conhecimento de $\mathbf{v}$. Um exemplo concreto de arranjo padrão pode ser visto na Tabela 2.2.

A interpretação do arranjo padrão e o segundo método de decodificação, que não requer o conhecimento de $\mathbf{v}$ será importante para a teoria de codificação quântica, a ser discutida

mensagem	0	1	**síndrome**
código	000	111	00
classe lateral	100	011	10
classe lateral	010	101	11
classe lateral	001	110	01

Tabela 2.2: Arranjo padrão para o código de repetição de três bits. Veja a correspondência entre os erros mais prováveis e suas classes laterais e síndromes.

na próxima seção.

No final do Capítulo 1 antecipamos que a classe dos códigos lineares clássicos possui duas subclasses importantes: a classe dos códigos de bloco clássicos (CBCs), em que cada bloco de informação é codificado independentemente dos outros blocos, como é o caso do código de repetição de três bits, e a classe dos códigos convolucionais clássicos (CCCs), em que a codificação de um bloco de informação depende de blocos de informação anteriores, ou seja, o processo de codificação envolve memória. Todo o desenvolvimento teórico apresentado nesta seção foi feito para os CBCs. Uma teoria semelhante envolvendo memória no processo de codificação será apresentada no Capítulo 3 para os CCCs.

2.2 Introdução à Codificação Quântica

Para proteger os estados quânticos dos efeitos do ruído, seria desejável desenvolver CCEQs baseados em princípios similares aos dos CCECs.

Mas, como vimos no Capítulo 1, existem algumas diferenças importantes entre a informação clássica e a informação quântica que exigem a introdução de novas idéias para que a construção dos CCEQs seja possível. Vale a pena relembrar as quatro grandes dificuldades que devem ser superadas:

1. Os *erros de fase* não têm análogo clássico.

2. *Os erros são contínuos*: Um *continuum* de diferentes erros pode ocorrer em um qubit. Determinar qual erro ocorreu para que se possa fazer a correção parece requerer

precisão infinita, e portanto fontes infinitas.

3. *Não é possível clonar a informação quântica*: Poderíamos tentar implementar o código de repetição no contexto quântico através da duplicação do estado quântico três ou mais vezes. Mas isto é proibido pelo teorema da não clonagem. Mesmo se a clonagem fosse possível, não seria possível medir e comparar os três estados quânticos da saída do canal.

4. *A medida destrói a informação quântica*: Na correção de erros clássicos observamos a saída do canal e decidimos qual procedimento de decodificação devemos adotar. Sabemos que as observações da mecânica quântica em geral destroem o estado quântico sob observação e tornam impossível a recuperação do estado original.

Felizmente, como veremos ao longo deste capítulo, nenhum destes problemas é capaz de impedir a construção de CCEQs a partir de CCECs. Em particular, um código será descrito em detalhes: o código de Shor [66]. É o código quântico mais simples que conhecemos e foi também o primeiro a ser descoberto (1995). O código de Shor é um código de bloco e é construído a partir do código de repetição de três bits. O método de construção deste código servirá de suporte para a construção de uma classe de CCQs.

A seguir, faremos a apresentação da classe dos códigos quânticos estabilizadores [26, 25]. Os códigos desta classe são construídos a partir de códigos lineares clássicos. Os exemplos mais importantes são: o próprio código de Shor, a subclasse dos códigos CSS [6] (com destaque para o código de Steane [69]) e o código que satura o limitante quântico de Hamming: o código perfeito [51, 68]. Portanto, vale a pena iniciar com o formalismo estabilizador e posteriormente revisitar os códigos quânticos conhecidos sob o ponto de vista deste formalismo. Apresentaremos também as condições *necessárias* e *suficientes* para a construção de *qualquer* CCEQ [26].

Assumiremos ao longo deste capítulo que as operações lógicas são *perfeitas*, ou seja, que os códigos são projetados de forma a proteger a informação de erros de armazenamento ou de transmissão mas não de erros *operacionais*.

O melhor meio de começar a entender como um CCEQ funciona é apresentando o código de Shor.

2.3 Construção do Código de Shor

2.3.1 O Código Bit Flip

Suponha que desejamos enviar qubits através de um canal que troca os estados-base de um qubit de $|0\rangle$ para $|1\rangle$ e vice-versa com probabilidade p e preserva o qubit de erros com probabilidade $1-p$. Ou seja, com probabilidade p o estado $|\psi\rangle$ é levado ao estado $X|\psi\rangle$, onde X é o *operador bit flip*. Assim se $|\psi\rangle = a|0\rangle + b|1\rangle$, temos $X|\psi\rangle = a|1\rangle + b|\ 0\rangle$. Este canal é chamado de *canal bit flip* e é equivalente aos canais binários clássicos sem memória e com erros ocorrendo aleatoriamente. Portanto, para proteger os qubits dos efeitos do ruído deste canal, é necessário construir um *código corretor de erros bit flip*.

Suponha que codifiquemos cada estado-base do qubit $a|0\rangle + b|1\rangle$ em um estado de três qubits. Esta operação de codificação é convenientemente escrita como:

$$|u\rangle \rightarrow |u_L\rangle \equiv |u,\, u,\, u\rangle, \tag{2.9}$$

onde $u = \{0, 1\}$. Ou, mais explicitamente,

$$\begin{aligned} |0\rangle &\rightarrow |0_L\rangle \equiv |000\rangle \\ |1\rangle &\rightarrow |1_L\rangle \equiv |111\rangle \end{aligned} \tag{2.10}$$

Assim, temos um mapeamento da superposição dos estados-base do qubit em uma superposição dos correspondentes estados-base codificados. A notação $|0_L\rangle$ e $|1_L\rangle$ indica, respectivamente, que estes são os estados-base *lógicos* $|0\rangle$ e $|1\rangle$.

Suponha que o estado inicial $a|0\rangle + b|1\rangle$ tenha sido perfeitamente codificado como $a|000\rangle + b|111\rangle$ e que cada um dos três qubits seja passado através de uma cópia idêntica do canal bit flip. Suponha também que possa ter ocorrido um erro em um dos três qubits.

Gostaríamos agora de corrigir um eventual erro bit flip sem destruir a superposição $a|000\rangle + b|111\rangle$. É claro, não podemos medir um qubit. Por exemplo, se medirmos o primeiro qubit e obtermos o resultado $|0\rangle$, teremos preparado o estado $|0_L\rangle$ e perdido a informação quântica codificada nos coeficientes a e b.

Porém, existe um procedimento simples de detectar qual erro ocorreu (se de fato ocorreu) através de duas medidas do sistema físico. Identificado o erro, é possível então fazer a correção e recuperar o estado quântico original que foi enviado ao canal. Duas possíveis medidas são as dos observáveis Z_1Z_2 ($Z \otimes Z \otimes I$) e Z_2Z_3 ($I \otimes Z \otimes Z$). Cada um destes observáveis tem autovalores ± 1, portanto cada medida fornece um bit de informação de um total de dois bits de informação. Ou seja, no total tem-se quatro possíveis *síndromes de erro*. A realização da primeira medida, de Z_1Z_2, é na verdade uma forma de *comparar* os dois primeiros qubits para ver se são iguais. E a realização da segunda medida, de Z_2Z_3, uma forma de comparar os dois últimos qubits para ver se são iguais. Para melhor entender o porquê disto, observe a decomposição espectral destes dois observáveis:

$$\begin{aligned} Z_1Z_2 &= (|00\rangle\langle 00| + |11\rangle\langle 11|) \otimes I - (|01\rangle\langle 01| + |10\rangle\langle 10|) \otimes I \\ Z_2Z_3 &= I \otimes (|00\rangle\langle 00| + |11\rangle\langle 11|) - I \otimes (|01\rangle\langle 01| + |10\rangle\langle 10|) \end{aligned} \tag{2.11}$$

O resultado da medida de Z_1Z_2 será $+1$ se os qubits forem iguais, e -1 se forem diferentes. Similarmente, o resultado da medida de Z_2Z_3 será $+1$ se os qubits forem iguais e -1 se forem diferentes. Ao combinarmos os resultados destas duas medidas, podemos determinar se um erro bit flip ocorreu em algum qubit ou não, e se ocorreu, em qual dos qubits ocorreu o erro: se ambos os resultados das medidas derem $+1$, então com grande probabilidade pode-se afirmar que nenhum erro bit flip ocorreu; se a medida de Z_1Z_2 der $+1$ e a medida de Z_2Z_3 der -1, então com grande probabilidade ocorreu um erro bit flip no terceiro qubit; se a medida de Z_1Z_2 der -1 e a medida de Z_2Z_3 der $+1$, então com grande probabilidade ocorreu um erro bit flip no primeiro qubit; e finalmente se ambas as medidas derem -1, então com grande probabilidade ocorreu um erro bit flip no segundo qubit. Note que estas síndromes são semelhantes àquelas encontradas na Tabela 2.2 para o caso clássico. A única diferença é que há um mapeamento entre as síndromes clássicas e quânticas: $0 \rightarrow +1$ e $1 \rightarrow -1$. Os outros dois possíveis conjuntos de observáveis que podem ser usados para a medida das síndromes são: $\{Z_1Z_3, Z_2Z_3\}$ e $\{Z_1Z_2, Z_1Z_3\}$. Repare que os três possíveis conjuntos de observáveis são compostos de operadores Z atuando sobre os qubits das posições dadas pelo número 1 na matriz verificação de paridade da eq. (2.3) ou em uma de suas duas formas *equivalentes*. Isto será discutido com mais detalhes mais adiante.

O que é crucial para o sucesso deste procedimento é que nenhuma das medidas dá qualquer informação sobre as amplitudes a e b do estado quântico codificado, e portanto nenhuma medida destrói as superposições dos estados quânticos que desejamos preservar ao usarmos este código.

A etapa seguinte é usar o valor das síndromes para determinar qual procedimento adotar para recuperar o estado original. Se as síndromes indicarem que nenhum erro ocorreu, nada temos a fazer. Mas se indicarem que ocorreu um erro de bit no qubit i (i=1, 2 ou 3), basta aplicar o operador X sobre o qubit i (X_i) para fazer a correção. Portanto, dadas as síndromes de erros, é possível recuperar o estado original com total perfeição.

O grupo de erros a corrigir é composto por um único gerador, $\langle X \rangle$. Sendo a *distância* do código de repetição de três bits igual a três, podemos afirmar que o código quântico associado é capaz de corrigir até $\lfloor (3-1)/2 \rfloor = 1$ erro X (ver definição 3 e teorema 1).

Portanto, o procedimento de correção descrito acima funciona perfeitamente desde que ocorra no máximo *um* erro bit flip. Isto ocorre com probabilidade $(1-p)^3 + 3p(1-p)^2 = 1 - 3p^2 + 2p^3$. A probabilidade de um erro não ser corrigido é portanto $3p^2 - 2p^3$, exatamente como o código de repetição clássico estudado anteriormente. Novamente, desde que $p < 1/2$, a codificação melhora a confiabilidade de armazenagem do estado quântico.

Fidelidade Quântica

A análise de erros que fizemos acima não é completamente adequada. O problema é que nem todos os erros e estados na mecânica quântica são criados de modo igual: vimos que estados quânticos estão em um espaço *contínuo*, portanto é possível que alguns erros corrompam um estado por uma quantidade muito pequena, enquanto que outros destruam o estado por completo. Um exemplo extremo disto é o fato que o *erro* bit flip X não afeta o estado $(|0\rangle + |1\rangle)/\sqrt{2}$ como um todo, mas troca o estado $|0\rangle$ pelo estado $|1\rangle$ e vice-versa. Para o estado $(|0\rangle + |1\rangle)/\sqrt{2}$, não precisaríamos nos preocupar se um erro bit flip ocorreu, enquanto que para os estados $|0\rangle$ ou $|1\rangle$, teríamos, obviamente, motivos para ficar muito preocupados.

Para tratar deste problema faz-se uso do conceito de *fidelidade quântica*[2]. Deixe-nos comparar a fidelidade *mínima* alcançada pelo código de três qubits com a fidelidade quando nenhuma correção de erros é feita. Suponha que o estado quântico de interesse seja $|\psi\rangle$. Sem o uso do código corretor de erros, o estado do qubit após ser enviado através do canal é:

$$\rho = (1-p)|\psi\rangle\langle\psi| + pX|\psi\rangle\langle\psi|X. \tag{2.12}$$

A fidelidade é dada por:

$$F = \sqrt{\langle\psi|\rho|\psi\rangle} = \sqrt{(1-p) + p\langle\psi|X|\psi\rangle\langle\psi|X|\psi\rangle}. \tag{2.13}$$

O segundo termo sob a raiz quadrada é não negativo, e igual a zero quando $|\psi\rangle = |0\rangle$. Portanto, vemos que a fidelidade mínima é $F = \sqrt{1-p}$. Suponha que o código corretor de três qubits seja usado para proteger o estado $|\psi\rangle = a|0_L\rangle + b|1_L\rangle$. O estado quântico corrompido após a correção é:

$$\rho = [(1-p)^3 + 3p(1-p)^2]|\psi\rangle\langle\psi| + \ldots. \tag{2.14}$$

Os termos omitidos representam contribuições de bit flips em dois ou três qubits. Todos os termos omitidos são operadores positivos, portanto a fidelidade que calculamos será um *limitante inferior* para a fidelidade verdadeira. Temos $\sqrt{\langle\psi|\rho|\psi\rangle} \geq \sqrt{(1-p)^3 + 3p(1-p)^2}$. Assim, a fidelidade é de pelo menos $\sqrt{1-3p^2+2p^3}$ e, portanto, a fidelidade para o estado quântico é aumentada desde que $p < 1/2$. Esta é a mesma conclusão a que chegamos anteriormente com uma análise bem menos refinada.

2.3.2 O Código Phase Flip

O código bit flip é interessante, mas não representa nenhuma inovação significante em relação aos CCECs, e não corrige os tipos de erros diferentes do erro bit flip que podem ocorrer sobre os qubits. Um canal quântico mais interessante é o *canal phase flip*,

[2]A fidelidade entre um estado puro $|\psi\rangle$ e um misto ρ é dada por $F(|\psi\rangle, \rho) = \sqrt{\langle\psi|\rho|\psi\rangle}$. (veja, por exemplo, o Capítulo 9 de [48]). O objetivo da correção de erros quânticos é aumentar a fidelidade até próximo da fidelidade máxima possível.

que pode corromper um estado quântico com erros phase flip. Neste modelo de erro, o qubit é preservado com probabilidade $1-p$, e com probabilidade p a *fase relativa* dos estados $|0\rangle$ para $|1\rangle$ é trocada. Ou seja, *o operador phase flip* Z é aplicado ao qubit com probabilidade $p>0$ e, portanto, quando o estado $a|0\rangle+b|1\rangle$ é transmitido, teremos à saida deste canal o estado $a|0\rangle-b|1\rangle$. Não existe equivalente clássico para o canal phase flip, pois canais clássicos não têm nenhuma propriedade equivalente à fase. No entanto, existe um meio fácil de tratar o canal phase flip como um canal bit flip. Suponha que ao invés de trabalharmos com a base computacional $\{|0\rangle, |1\rangle\}$, trabalhemos com a base conjugada $\{|+\rangle \equiv (|0\rangle+|1\rangle)/\sqrt{2},\ |-\rangle \equiv (|0\rangle-|1\rangle)/\sqrt{2}\}$ para o qubit. Com respeito a esta base, o operador Z leva o estado $|+\rangle$ para o estado $|-\rangle$ e vice-versa, isto é, este operador atua como se fosse um operador bit flip com respeito aos símbolos $+$ e $-$. Isto sugere os estados:

$$\begin{aligned} |0\rangle &\rightarrow |0_L\rangle \equiv |{+}{+}{+}\rangle \\ |1\rangle &\rightarrow |1_L\rangle \equiv |{-}{-}{-}\rangle \end{aligned} \tag{2.15}$$

como os estados $|0\rangle$ e $|1\rangle$ *lógicos* para a proteção de erros phase flip. Todas as operações necessárias ao processo de correção deste tipo de erro − o esquema de codificação, a determinação das síndromes de erros e a recuperação do estado original − são executadas da mesma forma que para o canal bit flip, mas agora com respeito à base $\{|+\rangle, |-\rangle\}$, ao invés da base $\{|0\rangle, |1\rangle\}$. Para realizarmos esta mudança de base, basta aplicar a porta de Hadamard e sua inversa (também a porta de Hadamard) nos pontos apropriados do processo, pois a porta de Hadamard é a operação unitária que realiza a troca entre a base computacional e a conjugada.

Na base computacional, a codificação para o canal phase flip é realizada em duas etapas: primeiro, codificamos em três qubits exatamente como foi feito para o canal bit flip; segundo, aplicamos uma porta de Hadamard sobre cada um dos qubits. Assim, a operação de codificação (2.15) é escrita na base $\{|0\rangle, |1\rangle\}$ como:

$$|u\rangle \rightarrow |u_L\rangle \equiv \frac{1}{2\sqrt{2}} \sum_{p,q,r=0}^{1} (-1)^{(p+q+r)u} |p,\, q,\, r\rangle, \tag{2.16}$$

onde $u = \{0,1\}$. Ou, mais explicitamente,

$$\begin{aligned} |0\rangle \to |0_L\rangle &\equiv \frac{|000\rangle + |001\rangle + |010\rangle + |011\rangle + |100\rangle + |101\rangle + |110\rangle + |111\rangle}{2\sqrt{2}} \\ |1\rangle \to |1_L\rangle &\equiv \frac{|000\rangle - |001\rangle - |010\rangle + |011\rangle - |100\rangle + |101\rangle + |110\rangle - |111\rangle}{2\sqrt{2}} \end{aligned} \tag{2.17}$$

Repare que aqui o código de repetição de três *bits* não aparece explícito dentro dos kets (como no código para o canal bit flip), mas aparece de forma sutil na distribuição dos *sinais* em frente de cada um dos oito kets: para $u = 0$, todos os sinais são positivos, mas para $u = 1$ os kets com número ímpar de 1s tem fase negativa.

Da mesma forma que para o canal bit flip, a detecção de um eventual erro phase flip pode ser feita com duas medidas. Dois possíveis observáveis são $H^{\otimes 3} Z_1 Z_2 H^{\otimes 3} = X_1 X_2$ e $H^{\otimes 3} Z_2 Z_3 H^{\otimes 3} = X_2 X_3$. A medida dos observáveis $X_1 X_2$ e $X_2 X_3$ permite comparar os *sinais* dos dois primeiros qubits e dos dois últimos qubits, respectivamente. Isto porque a medida de $X_1 X_2$ dá resultado $+1$ para estados do tipo $|+\rangle|+\rangle \otimes |\cdot\rangle$ ou $|-\rangle|-\rangle \otimes |\cdot\rangle$, e -1 para estados do tipo $|+\rangle|-\rangle \otimes |\cdot\rangle$ ou $|-\rangle|+\rangle \otimes |\cdot\rangle$.

Finalmente, a correção de erros phase flip pode ser completada aplicando-se os mesmos operadores utilizados no canal bit flip, mas conjugados pela matriz de Hadamard. Se as síndromes indicarem que nenhum erro ocorreu, nada temos a fazer. Mas se indicarem que ocorreu um erro de fase no qubit i, basta aplicar o operador $HXH = Z$ sobre o qubit i (Z_i) para fazer a correção. Portanto, dadas as síndromes de erros, é possível recuperar o estado original com total perfeição.

Obviamente, o código para o canal phase flip tem as *mesmas* características do código para o canal bit flip. Dizemos que estes dois canais são *unitariamente equivalentes*, pois existe um operador unitário U (neste caso a porta de Hadamard) tal que a ação de um canal é a mesma do outro canal, determinando que o primeiro canal seja precedido por U e seguido por $U^\dagger$. Além disso, estas operações podem ser trivialmente incorporadas às operações de codificação e correção de erros.

O grupo de erros a corrigir é composto por um único gerador, $\langle Z \rangle$. Sendo a *distância* do código de repetição de três bits igual a três, podemos afirmar que o código quântico associado é capaz de corrigir até $\lfloor (3-1)/2 \rfloor = 1$ erro Z.

2.3.3 O Código de Shor

O código de Shor é capaz de corrigir um erro quântico arbitrário em um qubit. Para construí-lo basta fazer uma combinação dos códigos de três qubits para os canais bit flip e phase flip. Primeiro codificamos o qubit usando o código phase flip: $|0\rangle \rightarrow |+++\rangle$ e $|1\rangle \rightarrow |---\rangle$. A seguir, codificamos *cada* um destes qubits usando o código bit flip de três qubits: $|+\rangle$ é codificado como $(|000\rangle + |111\rangle)/\sqrt{2}$ e $|-\rangle$ é codificado como $(|000\rangle - |111\rangle)/\sqrt{2}$. O resultado é um código de nove qubits, com palavras-código dadas por

$$|u\rangle \rightarrow |u_L\rangle \equiv \frac{1}{2\sqrt{2}} \sum_{p,q,r=0}^{1} (-1)^{(p+q+r)u} |p,\, p,\, p,\, q,\, q,\, q,\, r,\, r,\, r\rangle, \tag{2.18}$$

onde $u = \{0,1\}$. Ou, mais explicitamente,

$$\begin{aligned} |0\rangle \rightarrow |0_L\rangle &\equiv \frac{(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)}{2\sqrt{2}} \\ |1\rangle \rightarrow |1_L\rangle &\equiv \frac{(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)}{2\sqrt{2}} \end{aligned} \tag{2.19}$$

Uma superposição original $a|0\rangle + b|1\rangle$ é codificada como $a|0_L\rangle + b|1_L\rangle$, *deslocalizando* a informação a ser codificada entre os nove qubits.

Este código quântico corrigirá um erro quântico geral em qualquer dos qubits pelas seguintes razões. Primeiro devemos mostrar que este código é capaz de corrigir erros X e Z em qualquer dos qubits. Sem perda de generalidade, suponha que tenha ocorrido um erro X no primeiro qubit. Da mesma forma que para o código bit flip, este erro pode ser detectado sem ambiguidade quando realizamos a medida dos observáveis Z_1Z_2 e Z_2Z_3. Neste caso, os resultados das medidas serão, respectivamente -1 e $+1$, indicando que ocorreu um erro no primeiro qubit. Para corrigi-lo basta então aplicar o operador X ao primeiro qubit. Similarmente, podemos detectar e corrigir um erro bit flip em qualquer um dos nove qubits do código. Como em geral não sabemos em qual dos blocos um erro X eventualmente ocorreu, é necessário realizar a medida de *seis* observáveis $\{Z_1Z_2, Z_2Z_3, Z_4Z_5, Z_5Z_6, Z_7Z_8, Z_8Z_9\}$.

Similarmente, um erro Z pode ser facilmente detectado quando comparamos os sinais do primeiro e do segundo *blocos*, e do segundo e do terceiro *blocos*, da mesma forma que comparamos o sinal do primeiro e do segundo qubits para o código phase flip. Por exemplo,

suponha, sem perda de generalidade, que um erro de fase tenha ocorrido no primeiro qubit. Este erro troca o sinal do primeiro bloco de qubits, de $|000\rangle + |111\rangle$ para $|000\rangle - |111\rangle$ (para $|0_L\rangle$), e de $|000\rangle - |111\rangle$ para $|000\rangle + |111\rangle$ (para $|1_L\rangle$). Na verdade, um erro Z em *qualquer* um dos *três primeiros* qubits tem este efeito e, portanto, o procedimento que vamos descrever serve para um erro Z em qualquer um dos três qubits. É possível demonstrar com um pouco de álgebra que os dois possíveis observáveis que comparam os sinais dos blocos são $X_1X_2X_3X_4X_5X_6$, $X_4X_5X_6X_7X_8X_9$. O primeiro destes observáveis compara o sinal do primeiro e do segundo blocos e o segundo observável compara o sinal do segundo e do terceiro blocos. Quando um erro Z ocorre em qualquer um dos três primeiros qubits, os sinais do primeiro e do segundo blocos são diferentes e os sinais do segundo e do terceiro blocos são iguais. Para recuperar o estado original, basta aplicar o operador Z sobre *qualquer* um dos três primeiros qubits.

O código de Shor também pode corrigir um erro X e Z ocorrendo no *mesmo* qubit. Para que isto seja possível é necessário aplicarmos ambos os procedimentos acima descritos de detecção e correção de erros X e Z, os quais são *independentes*. Finalmente, um qubit com um erro *arbitrário* é *projetado* para ter erros I, X, Z, ou XZ durante as medidas de síndrome dos erros X e Z, e assim fazer com que estes erros possam ser corrigidos.

A ordem dos códigos bit flip e phase flip na concatenação é importante. É fácil de verificar que se tivéssemos codificado o qubit primeiro com o código bit flip, ou seja, $|0\rangle \to |000\rangle$ e $|1\rangle \to |111\rangle$, e em seguida codificado cada um destes três qubits com o código phase flip, ou seja, $|0\rangle \to |{+}{+}{+}\rangle$ e $|1\rangle \to |{-}{-}{-}\rangle$, teríamos, na base $\{|0\rangle, |1\rangle\}$, o seguinte código:

$$\begin{aligned} |0\rangle &\to \frac{(|0\rangle+|1\rangle)(|0\rangle+|1\rangle)(|0\rangle+|1\rangle)(|0\rangle+|1\rangle)(|0\rangle+|1\rangle)(|0\rangle+|1\rangle)(|0\rangle+|1\rangle)(|0\rangle+|1\rangle)}{16\sqrt{2}} \\ |1\rangle &\to \frac{(|0\rangle-|1\rangle)(|0\rangle-|1\rangle)(|0\rangle-|1\rangle)(|0\rangle-|1\rangle)(|0\rangle-|1\rangle)(|0\rangle-|1\rangle)(|0\rangle-|1\rangle)(|0\rangle-|1\rangle)}{16\sqrt{2}} \end{aligned} \tag{2.20}$$

Mas este é um "código universal" do tipo $|000\ldots000\rangle$, $|000\ldots001\rangle$, $|000\ldots010\rangle$, $\ldots$, $|111\ldots101\rangle$, $|111\ldots110\rangle$, $|111\ldots111\rangle$, incapaz de garantir a correção de qualquer erro.

Muitos conceitos importantes sobre a teoria de correção de erros quânticos estão ilustra-

dos neste código:

1. **Medir as síndromes não implica em medir os qubits.** As síndromes podem ser encontradas sem o ganho de qualquer informação sobre os estados codificados. Os erros unitários (X e Z) podem ser invertidos sem o conhecimento da informação codificada. Isto é análogo ao método de decodificação clássica que projeta o estado recebido do canal sobre uma classe lateral do arranjo padrão e inverte o erro mais provável (de peso mínimo).

2. A **redundância** é usada para inserir o espaço das palavras-código em um espaço maior, de forma a fazer com que *os erros corrigíveis mapeiem o espaço do código em subespaços ortogonais* (sem intersecção). Isto é o análogo quântico das esferas de Hamming.

3. **Os erros são locais e a informação codificada é não local**. É importante enfatizar a hipótese central que serve de base para a construção de um código quântico − erros afetando diferentes qubits são, em boa aproximação, *não correlacionados*. Assumimos que um evento que causa um erro em dois qubits é muito menos provável que um evento que causa um erro em um qubit. É claro que uma questão física é saber se esta hipótese é justificada ou não − podemos facilmente imaginar processos que causam erros em dois qubits de uma só vez. Se tais erros correlacionados forem comuns, a codificação falhará.

4. **O código de Shor é um código *concatenado***. Este código concatena dois códigos de repetição de três qubits. A concatenação segue uma hierarquia de níveis. O primeiro código, que corrige erros Z, é concatenado ao segundo código, que corrige erros X. Portanto, a matriz geradora do código clássico associado ao código de Shor é

$$\begin{aligned} \mathbf{G} = \mathbf{G}_Z \cdot \mathbf{G}_X &= \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}. \end{aligned} \tag{2.21}$$

ou seja, um código de repetição de taxa 1/9 (e, obviamente, distância igual a nove). Portanto, este código clássico é capaz de corrigir $\lfloor (9-1)/2 \rfloor = 4$ erros.

5. **O código de Shor pode corrigir um erro arbitrário ocorrendo em um qubit**. O código de Shor pode corrigir um erro quântico geral porque os códigos clássicos associados aos códigos bit flip, phase flip e bit-phase flip têm distâncias três (o que garante a correção de um erro X), três (o que garante a correção de um erro Z) **e** nove (o que garante a correção de um erro cuja *base* é gerada pelos erros X, Z, XZ e I), respectivamente. O grupo de erros que este código corrige é composto por dois geradores, $\langle X, Z \rangle$. Isto é aparentemente impossível, já que o espaço de síndromes é finito e o número possível de erros é infinito. Porém, o *continuum* de erros pode ser *discretizado*. Para ver isto, suponha que um erro arbitrário E tenha ocorrido em um qubit. Como E pode ser sempre escrito como $E = c_i I + c_x X + c_y Y + c_z Z$, o estado corrompido é a superposição $E|\psi_{in}\rangle = c_i|\psi_{in}\rangle + c_x X|\psi_{in}\rangle + c_y Y|\psi_{in}\rangle + c_z Z|\psi_{in}\rangle$. As medidas de síndrome, as quais identificam os erros I, X, Y e Z, *projetam* o estado corrompido sobre um dos quatro termos, todos eles corrigíveis. Em outras palavras, *quantizamos* os erros. Embora os erros na informação quântica possam ser pequenos, fazemos medidas que projetam nosso estado sobre um estado sem nenhum erro ou sobre um estado com um erro que pertence a um conjunto discreto de erros que sabemos como corrigir.

6. **Erros Z atuando em diferentes qubits dentro do mesmo bloco têm efeitos idênticos**. Não é possível, nem necessário, distinguir tais erros. Um código quântico é *não degenerado* se todos os erros corrigíveis podem ser identificados sem ambiguidade; caso contrário, é degenerado. O código de Shor é portanto um código degenerado.

   Esta propriedade é mais facilmente entendida através de um exemplo. Considere o efeito dos erros Z_1, Z_2 e Z_3 (primeiro bloco) sobre a palavra-código (2.19). É fácil de perceber que o efeito destes erros sobre esta palavra-código é o *mesmo*. O processo de decodificação nos permite identificar um erro de fase no primeiro bloco, mas não nos permite identificar em qual dos três primeiros qubits o erro de fase ocorreu. Para

corrigir *qualquer* um destes três erros, basta aplicar *qualquer* um dos operadores inversos, ou seja, Z_1, Z_2 ou Z_3.

A informação foi armazenada em *correlações* envolvendo vários qubits. Presumimos uma natureza local para os erros e codificamos a informação por um método não local. Não existe meio de distinguir $|0_L\rangle$ de $|1_L\rangle$ através da medida de nenhum dos nove qubits. Se medirmos um qubit, encontraremos $|0_L\rangle$ com probabilidade $1/2$ e $|1_L\rangle$ com probabilidade $1/2$ independente do valor do qubit codificado.

O meio circundante pode ocasionalmente "medir" um dos qubits. Mas, neste caso, a informação codificada não pode ser prejudicada através da perturbação daquele qubit, porque um qubit, por si só, não carrega informação pelo todo. A informação codificada não localmente é invulnerável a influências locais − este é o princípio central sobre o qual os CCEQs estão fundamentados.

2.4 Critérios para a Correção de Erros Quânticos

Nesta seção apresentamos as condições algébricas para que um subespaço $\mathcal{C}$ seja um código quântico que corrige certos erros atuando sobre o espaço de Hilbert $\mathcal{H}$. Estas condições são freqüentemente chamadas de *critérios para a correção de erros quânticos*, e são de grande valor para o entendimento e a construção de códigos quânticos. Tais condições também permitem que as noções de correção de erros quânticos sejam tratadas de modo mais rigoroso.

Para codificar k qubits em n qubits, um código terá 2^k palavras-código-base. Qualquer combinação linear destas palavras-código-base também é uma palavra-código. O espaço $\mathcal{C}$ de palavras-código (o *espaço de codificação*) é portanto um espaço de Hilbert de 2^k dimensões e um subespaço de um espaço de Hilbert com 2^n dimensões.

Vimos através do código de Shor que se pudermos corrigir erros E e F, também poderemos corrigir um erro arbitrário $aE + bF$. Logo, precisamos somente nos preocupar se o código pode corrigir uma *base de erros*. Uma base conveniente para se usar é o conjunto de produtos tensoriais de X, Y, Z e I. O *peso* de um operador desta forma é o número de qubits que diferem da identidade. O conjunto de todos estes produtos tensoriais com

uma possível fase global de -1 ou $\pm i$ forma um grupo $\mathcal{G}$ sob a *operação de multiplicação*. Usamos a notação $\mathcal{G}_n$ para distinguir os grupos referentes a diferentes números de qubits; $\mathcal{G}_n$ é o produto direto de n cópias do grupo dos quatérnios a menos de um fator de fase global[3]. Note que normalmente incluimos a identidade no conjunto dos possíveis *erros* porque não queremos confundir um erro ocorrendo em um qubit com *nenhum* erro ocorrendo em outro qubit. Se tivermos um canal em que estamos *certos* de que algum erro ocorreu, então não precisaremos incluir a identidade como um possível erro.

Para um código corrigir dois erros E_a e E_b, devemos sempre ser capazes de distinguir um erro E_a atuando sobre uma palavra-código-base $|i_L\rangle$ de um erro E_b atuando sobre uma palavra-código-base *diferente* $|j_L\rangle$. E somente estaremos certos disto se $E_a|i_L\rangle$ for *ortogonal* a $E_b|j_L\rangle$; caso contrário, haverá alguma chance de confundí-los. Também, quando fazemos uma medida para determinar o erro, não devemos obter nenhuma informação a respeito do estado do código dentro do espaço de codificação. Se obtivermos alguma informação, estaremos destruindo as superposições dos estados da base, e não seremos mais capazes de corrigir uma palavra-código arbitrária.

Knill e Laflamme [37] e Bennett *et al.* [68] provaram que para um código funcionar como um CCEQ a seguinte condição deve ser satisfeita:

$$\langle i_L|E_a^\dagger E_b|j_L\rangle = \mathbf{C}_{ab}\delta_{ij}, \tag{2.22}$$

onde $|i_L\rangle$ e $|j_L\rangle$ são todas as possíveis combinações de palavras-código-base, E_a e E_b são todas as possíveis combinações de erros e $\mathbf{C}_{ab}$ é um elemento de uma matriz independente de i e j. Eles provaram que a condição (2.22) é uma condição *necessária* e *suficiente* para um código corrigir os erros $\{E_a\}$.

O limitante quântico de Hamming, análogo ao limitante clássico de Hamming (2.5), pode ser facilmente encontrado para códigos *não degenerados*. Suponha que um código quântico não degenerado seja usado para codificar k qubits de informação em n qubits de forma a poder corrigir erros sobre qualquer subconjunto de t ou menos qubits. Suponha que j erros ocorreram, onde $j \leq t$. Existem $\binom{n}{j}$ possíveis conjuntos de localizações onde os erros podem ocorrer. A cada um destes conjuntos associamos três erros possíveis (X, Y e

[3]Em particular, $\mathcal{G}_1$ é conhecido como grupo dos *quatérnios* [54].

Z) que podem corromper cada qubit, gerando um total de 3^j possíveis erros. Para codificar k qubits de forma não degenerada, cada um destes erros deve corresponder a um subespaço 2^k-dimensional ortogonal. Todos estes subespaços devem estar contidos dentro do espaço total de 2^n dimensões gerado pelos n qubits das palavras-código. Portanto, conclui-se que

$$\sum_{j=0}^{t} \binom{n}{j} 3^j 2^k \leq 2^n \tag{2.23}$$

é o limitante quântico de Hamming para códigos não degenerados. Para $k = 1$ e $t = 1$, temos:

$$2(1+3n) \leq 2^n. \tag{2.24}$$

Esta condição só é satisfeita para $n \geq 5$. Assim, o menor código quântico não degenerado que pode corrigir um erro arbitrário em qualquer qubit tem comprimento cinco. O código que satura este limitante é conhecido como *código perfeito* e será apresentado mais adiante. Note que dizer que um código é degenerado ou não depende do conjunto de erros que pretendemos corrigir. Por exemplo, um código degenerado corretor de dois erros pode não ser degenerado quando considerado como um código corretor de um erro.

Na equação (2.22), se E_a e E_b pertencem ao grupo $\mathcal{G}$, $E = E_a^\dagger E_b$ também pertence ao grupo $\mathcal{G}$. O menor peso de E em $\mathcal{G}$ para o qual (2.22) *não* é satisfeita é chamado de *distância* do código. Para um código quântico corrigir até t erros, deve ter distância de pelo menos $2t+1$. Todo código tem distância pelo menos um. Um código quântico de distância d codificando k qubits em n qubits é descrito como um código $[n, k, d]$. O colchete serve para enfatizar a similaridade com a teoria clássica.

Podemos também considerar variações do problema usual de correção de erros. Por exemplo, suponha que queremos somente detectar se um erro ocorreu e não corrigi-lo. Neste caso, não precisamos distinguir erros E_a de E_b, somente precisamos distingui-los da identidade. Podemos usar a condição (2.22), só que agora $E_b = I$ sempre. Isto significa que para um código detectar s erros deve ter distância pelo menos $s+1$. Outra variação do problema é quando sabemos em qual qubit (ou em quais qubits) um erro ocorreu, como em alguns casos específicos de canal quântico. Neste caso, somente precisamos distinguir os erros E_a de E_b que afetam os mesmos qubits. Isto significa que $E_a^\dagger E_b$ tem o mesmo peso de

E_a, e para corrigir r erros "localizáveis", precisamos de um código de distância pelo menos $r+1$. Podemos também imaginar a combinação de todas estas tarefas. Ou seja, para um código corrigir t erros arbitrários, r erros "localizáveis" adicionais e ainda detectar s erros, deve ter distância de pelo menos $r+s+2t+1$.

Assumiremos que os erros ocorrem independentemente em diferentes qubits, e que quando um erro ocorre em um qubit, é igualmente provável de ser um erro X, Y ou Z. Sabemos que, em sistemas reais, a hipótese de que erros são igualmente prováveis é pobre. Na prática, algumas combinações lineares de X, Y ou Z são mais prováveis que outras. Se a probabilidade de erro por qubit ε é muito pequena, é freqüentemente útil para a simplificação do processo ignorar a probabilidade de mais de t erros ocorrerem, pois isto somente ocorre com probabilidade $O(\varepsilon^{t+1})$. Portanto, trabalharemos tipicamente com códigos que corrigem até t erros arbitrários. Outra possível dificuldade surge quando podem ocorrer erros correlacionados em múltiplos qubits. Em princípio isto pode ser um grave problema, mas este caso pode ser considerado sem uma mudança do formalismo se a probabilidade de um erro correlacionado cair rápido o suficiente com o tamanho dos blocos de erros.

A verificação direta das condições necessárias e suficientes para a correção de erros quânticos é uma tarefa exaustiva, inglória. Na próxima seção descreveremos um formalismo teórico que usa estas condições para a correção de erros quânticos como um ponto de partida para a construção de muitas classes interessantes de códigos, e que contorna muito da dificuldade associada com a verificação direta destas condições.

2.5 Códigos Estabilizadores

We cannot clone, perforce; instead, we split
Coherence to protect it from that wrong
That would destroy our valued quantum bit
And make our computation take too long.

Correct a flip and phase — that will suffice.
If in our code another error's bred,

We simply measure it, then God plays dice,
Collapsing it to X or Y or zed.

We start with noisy seven, nine, or five
And end with perfect one. To better spot
Those flaws we must avoid, we first must strive
To find which ones commute and which do not.

With group and eigenstate, we're learned to fix
Your quantum errors with our quantum tricks.

—"Quantum Error Correction Sonnet", by Daniel Gottesman.

A teoria geral de correção de erros quânticos descrita na seção anterior fornece critérios para a correção de erros. No entanto, não fornece métodos de construção. Vimos que o código de Shor está relacionado com o código de repetição clássico. Uma conexão útil a fazer é adaptar códigos clássicos conhecidos para construir uma classe mais geral de códigos quânticos.

Steane [69] e independentemente Shor e Calderbank [6], desenvolveram uma classe de códigos (hoje conhecida como classe CSS) derivada de alguns códigos lineares clássicos especiais sobre $GF(2)$. Gottesman [26, 25] subsequentemente desenvolveu uma teoria de grupos para códigos quânticos que é muito útil para a construção e o entendimento de códigos quânticos. Resultados similares, mas mais gerais, devido a Calderbank *et al* [63, 64] conectaram a codificação quântica com a geometria ortogonal e códigos clássicos sobre $GF(4)$. Vamos revisar o formalismo de código estabilizador devido a Gottesman, pois será o mais relevante para a construção da classe de CCQs que apresentaremos nos Capítulos 4 e 5.

Os códigos estabilizadores (ou *aditivos*) são uma classe importante de códigos quânticos cuja construção é análoga à dos códigos lineares clássicos [11]. Para melhor entendermos esta classe de códigos é necessário primeiro desenvolver o *formalismo estabilizador*, um método algébrico poderoso necessário para o entendimento de uma classe mais ampla de operações da mecância quântica. As aplicações do formalismo estabilizador estendem-se para além da correção de erros quânticos (são cruciais, por exemplo, para a teoria

quântica de tolerância a falhas), no entanto limitaremo-nos a esta aplicação específica. Após definirmos o formalismo estabilizador, apresentaremos um importante teorema que quantifica as limitações das operações de estabilizadores. A seguir, explicaremos como construir códigos estabilizadores através de alguns exemplos.

2.5.1 O Formalismo Estabilizador

Considere o estado EPR de dois qubits:

$$|\psi\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}. \tag{2.25}$$

É fácil de verificar que este estado satisfaz as identidades $X_1X_2|\psi\rangle = |\psi\rangle$ e $Z_1Z_2|\psi\rangle = |\psi\rangle$; assim, dizemos que o estado $|\psi\rangle$ é estabilizado pelos operadores X_1X_2 e Z_1Z_2. Um pouco menos óbvio é verificar que o estado $|\psi\rangle$ é o *único* estado quântico (a menos de uma fase global) que é estabilizado pelos operadores X_1X_2 e Z_1Z_2.

A idéia básica do formalismo estabilizador é que muitos estados quânticos podem ser descritos mais facilmente pelos operadores que o estabilizam do que pelo próprio estado quântico. Muitos códigos quânticos (incluindo o código de Shor e os códigos CSS) podem ser descritos de forma muito mais compacta usando estabilizadores do que a descrição por vetores de estado. Além disso, as operações quânticas tais como os erros nos qubits e porta de Hadamard sobre uma base computacional são mais facilmente descritas se usarmos o formalismo estabilizador. O poder do formalismo estabilizador está no uso inteligente da teoria de grupos.

Geradores do Grupo Estabilizador

Considere o espaço de Hilbert $\mathcal{H}$ para n qubits (portanto um espaço com 2^n dimensões) e o grupo de Pauli $\mathcal{G}_n$ gerado por $X^{(i)}$ e $Z^{(i)}$ (i=1,..., n) atuando sobre $\mathcal{H}$. Nesta definição, $\sigma_y \notin \mathcal{G}_n$, embora $Y = XZ = -i\sigma_y \in \mathcal{G}_n$. Cada elemento $M \in \mathcal{G}_n$ tem autovalores ± 1 ou $\pm i$, e $M^2 = \pm I$ ($M^\dagger = \pm M$). Quaisquer dois elementos em $\mathcal{G}_n$ comutam ou anticomutam. Seja S o subgrupo *abeliano* de $\mathcal{G}_n$ com $n-k$ geradores *hermitianos*. Estes geradores, que comutam entre si, definem 2^{n-k} autoespaços simultâneos. O autoespaço $\mathcal{C}$ correspondente

ao autovalor $+1$ para todos os geradores de S é chamado de código estabilizador com estabilizador S [4]. Qualquer estado $|\psi\rangle$ em $\mathcal{C}$ é estabilizado por qualquer elemento M em S, isto é, $M|\psi\rangle = |\psi\rangle$. $\mathcal{C}$ tem 2^k dimensões e, portanto, codifica k qubits.

Seja $\{|i_L\rangle\}$ uma base para $\mathcal{C}$ e $\{E_a\}$ o conjunto de erros a ser corrigidos. Vamos nos concentrar em códigos que corrigem até t erros, portanto $\{E_a\} \subset \mathcal{G}_n$ contém elementos em $\mathcal{G}_n$ com peso de Hamming menor ou igual a t.

Teorema 3. *[48] $\mathcal{C}$ é um código que corrige os erros $\{E_a\}$ se $\forall$ a e b, $E_a^\dagger E_b$ anticomuta com algum $M \in S$ ou $E_a^\dagger E_b \in S$.*

Prova: *Se $E_a^\dagger E_b$ anticomuta com algum $M \in S$, então*

$$\langle i_L|E_a^\dagger E_b|j_L\rangle = \langle i_L|E_a^\dagger E_b M|j_L\rangle = -\langle i_L|ME_a^\dagger E_b|j_L\rangle = -\langle i_L|E_a^\dagger E_b|j_L\rangle = 0. \tag{2.26}$$

Por outro lado, se $E_a^\dagger E_b \in S$, então

$$\langle i_L|E_a^\dagger E_b|j_L\rangle = \langle i_L|j_L\rangle = \delta_{ij}. \tag{2.27}$$

Além disso, se $E_a^\dagger E_b \in S$, então $(E_a^\dagger E_b)^{-1} = E_b^\dagger E_a \in S$, e portanto a equação (2.27) vale quando a e b são trocados. Combinando as equações (2.26) e (2.27), temos que $\langle i_L|E_a^\dagger E_b|j_L\rangle = \mathbf{C}_{ab}\delta_{ij}$, onde $\mathbf{C}_{ab} = 0$ ou 1 é independente de i e j, e como $\mathbf{C}_{aa} = 1$, $\mathbf{C}_{ab}$ forma uma matriz positiva. Portanto, $\mathcal{C}$ satisfaz os critérios de correção de erros dados pela equação (2.22). ■

Um código estabilizador é não-degenerado se para $\forall\, a \neq b$, $E_a^\dagger E_b$ anticomuta com algum $M \in S$. Neste caso, $\mathbf{C}_{ab}$ é diagonal. Um código estabilizador degenerado tem alguns erros corrigíveis E_a e E_b tais que $E_a^\dagger E_b \in S$, significando que E_a e E_b atuam identicamente sobre $\mathcal{C}$. Portanto, para um código estabilizador, erros corrigíveis são exatamente distinguíveis ou idênticos em $\mathcal{C}$.

[4] A existência de tal subespaço é garantida pela comutatividade e hermiticidade de seus geradores. Note que todo elemento $M \in S$ é hermitiano com autovalores ± 1, e $M^2 = I$. Ao fazer isto, excluimos elementos de S com número ímpar de Ys.

Grupo de Pauli Lógico para Códigos Estabilizadores

Vimos que o estabilizador S é um subgrupo abeliano de $\mathcal{G}_n$. Os operadores que não comutam com todos os $M \in S$ mapeiam $\mathcal{C}$ em seu complemento ortogonal. Tais operadores são erros detectáveis. No entanto, podem existir operadores em $\mathcal{G}_n$ que comutam com todo $M \in S$ mas *não* estão em S. Estes operadores são os erros que mudam os qubits codificados *sem serem detectados*. No entanto, tais operações são operações lógicas legítimas para os códigos estabilizadores porque *mapeiam palavras-código em palavras-código*.

Matematicamente, o *centralizador* de S consiste em operações que comutam com todo $M \in S$. Para um estabilizador, o centralizador é exatamente o *normalizador* de S em $\mathcal{G}_n$, denotado por $N(S)$, que é o conjunto de todas as operações que permutam os elementos de S por conjugação[5].

A distância do código, d, é o peso mínimo de $E \in N(S) - S \equiv N(S)/S$ [6]. Como $M \in S$ atua trivialmente sobre $\mathcal{C}$, as operações lógicas podem ser tomadas como elementos em $N(S)/S$.

Considere um código estabilizador $[n, k, d]$. S tem $n-k$ geradores (e 2^{n-k} elementos). Os geradores podem ser estendidos a um conjunto *independente* máximo de n observáveis que comutam mutuamente (gerando 2^n observáveis que comutam entre si) através de k elementos extras $\overline{Z}_1$, $\overline{Z}_2$, ..., $\overline{Z}_k$ em $N(S)/S$. Então, os autoestados simultâneos de $\overline{Z}_i$ em $\mathcal{C}$ podem ser escolhidos como os estados lógicos codificados. Em particular, o estado codificado $|c_1, c_2, \ldots, c_k\rangle$ é o autoestado $(-1)^{c_i}$ de $\overline{Z}_i$. Isto transforma o $\overline{Z}_i$ no σ_z lógico para o i-ésimo qubit lógico.

Os elementos restantes em $N(S)/S$ não comutarão com todos os $\overline{Z}_i$, e cada $\overline{X}_i$ pode ser escolhido em $N(S)/S$ desde que comutem com todos os $\overline{Z}_j$ para $j \neq i$ e anticomutem com $\overline{Z}_i$. Analogamente, $\overline{X}_i$ atua como o σ_x lógico para o i-ésimo qubit lógico.

[5] Seja $A \in \mathcal{G}_n$ e $M \in S$, então $AMA^\dagger = M \in S$ se $[M,A] = 0$. $AMA^\dagger = -M \notin S$ se $\{M,A\} = 0$. Portanto, $[A,M] \in S$ para $\forall\, M \in S$ se, e somente se, $AMA^\dagger \in S$ para $\forall\, M \in S$, significando que A permuta os elementos de S.

[6] Representação matemática para grupo quociente.

2.5.2 Sumário: Formalismo Estabilizador

O formalismo estabilizador oferece um meio fácil de caracterizar o subespaço do código quântico. Isto se dá através de dois grupos de operadores:

1. **Geradores do Grupo Estabilizador:**

   - O grupo estabilizador S é um subgrupo abeliano do grupo multiplicativo de Pauli, $\mathcal{G}_n = \{I, X, Y, Z\}^{\otimes n}$.
   - O subespaço do código $\mathcal{C}$ é o maior subespaço de $\mathcal{H}^{\otimes n}$ estabilizado por S:

   $$|\psi\rangle \in \mathcal{C} \Longleftrightarrow S|\psi\rangle = |\psi\rangle. \tag{2.28}$$

   - Equivalentemente, se os M_i's são $n-k$ geradores independentes de S, então:

   $$|\psi\rangle \in \mathcal{C} \Longleftrightarrow \forall\, i,\, M_i|\psi\rangle = |\psi\rangle. \tag{2.29}$$

   Estas equações, chamadas de síndromes, definem o subespaço do código.

2. **Grupo de Pauli Lógico:** Os operadores lógicos deixam o subespaço do código $\mathcal{C}$ globalmente invariante, mas possuem uma ação não trivial sobre este espaço. É possível exigir que tais operadores reproduzam exatamente as relações de comutação do grupo de Pauli para os qubits lógicos. Isto é matematicamente expresso por:

$$\begin{cases} \overline{X}_i, \overline{Z}_i \in N(S)/S \\ \{\overline{X}_i, \overline{Z}_i\} = 0, \\ \forall\ i \neq j,\ [\overline{X}_i, \overline{X}_j] = [\overline{Z}_i, \overline{Z}_j] = [\overline{X}_i, \overline{Z}_j] = 0. \end{cases} \tag{2.30}$$

A aplicação do formalismo estabilizador ficará muito mais clara com a apresentação dos exemplos abaixo.

2.5.3 Os Códigos Bit Flip e Phase Flip Revisitados

Revisitemos os códigos bit flip e phase flip usados na construção do código de Shor sob a ótica do formalismo estabilizador.

O código bit flip de três qubits é um código quântico [3, 1, 3] em que o estabilizador tem dois geradores: $S = \langle Z_1Z_2,\, Z_2Z_3\rangle$. É fácil de verificar que as palavras-código $|0_L\rangle$ e $|1_L\rangle$ apresentadas em (2.10) são autovetores com autovalor $+1$ dos geradores Z_1Z_2 e Z_2Z_3. Ou seja, $M_i|u_L\rangle = |u_L\rangle$ para $u = \{0,1\}$ e $i = 1,2$.

Por inspeção, vemos que todo produto tensorial possível de dois elementos do conjunto de erros $\{I,\, X_1,\, X_2,\, X_3\}$ $-$ $I,\, X_1,\, X_2,\, X_3,\, X_1X_2,\, X_1X_3,\, X_2X_3$ $-$ anticomuta com pelo menos um dos geradores do estabilizador (exceto para I, que está em S), e portanto o conjunto $E = \{I,\, X_1,\, X_2,\, X_3\}$ forma um conjunto corrigível para este código.

A detecção de erros para este código é efetuada através da medida dos autovalores dos geradores do estabilizador, Z_1Z_2 e Z_2Z_3. Se, por exemplo, o erro X_1 ocorreu, a medida da síndrome fornece como resultados -1 e $+1$. Similarmente, o erro X_2 fornece como resultados -1 e -1, o erro X_3 fornece como resultados $+1$ e -1, e o "erro" I fornece como resultados $+1$ e $+1$. Em cada caso, a correção é efetuada aplicando-se a operação inversa ao erro indicado pela síndrome.

Os geradores de um código quântico estabilizador têm uma função similar à função desempenhada pela matriz verificação de paridade na codificação clássica. No código estabilizador, medimos o autovalor (± 1) de cada gerador no estado recebido possivelmente corrompido, o que é análogo a uma verificação de paridade. Os autovalores medidos formam a síndrome no código quântico. Como cada erro distinguível mapeia $\mathcal{C}$ em um autoespaço simultâneo diferente dos geradores de S correspondendo a autovalores diferentes, cada síndrome corresponderá a um único erro em $\{E_a\}$.

Para sermos mais claros, note que a matriz verificação de paridade da matriz geradora do código de repetição de três bits é

$$\mathbf{H}_X = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}, \tag{2.31}$$

e que as posições dos 1s nesta matriz indicam exatamente as posições dos qubits com

operador Z nos dois geradores do estabilizador. Como $\mathbf{G}_X\mathbf{H}_X^T = 0$ e as posições dos 1s das linhas de $\mathbf{H}_X$ determinam os geradores Z do estabilizador, podemos usar a posições dos 1s da linha de $\mathbf{G}_X$ para determinar a operação lógica de erro $\overline{X}$, já que $\overline{X}$ deve ser independente dos dois geradores do estabilizador e comutar com cada um deles. Assim, $\overline{X} = X_1X_2X_3$. É fácil de verificar que, para as palavras-código $|u_L\rangle$ apresentadas em (2.10), $\overline{X}|u_L\rangle = |(u+1)_L\rangle$ é satisfeita para $u = \{0,1\}$.

Para o código phase flip dado pelas palavras-código (2.17), a situação é análoga. Basta fazer a permuta $Z \leftrightarrow X$ para os geradores do estabilizador, os erros corrigíveis e o operador lógico de erro. Portanto, para o canal phase flip, temos $S = \langle X_1X_2,\ X_2X_3\rangle$, $E = \{I,\ Z_1,\ Z_2,\ Z_3\}$ e $\overline{Z} = X_1X_2X_3$. É fácil de verificar que, para as palavras-código $|u_L\rangle$ apresentadas em (2.17) com $u = \{0,1\}$, as relações $M_i|u_L\rangle = |u_L\rangle$ para $i = 1,2$ e $\overline{Z}|u_L\rangle = (-1)^u|u_L\rangle$ são satisfeitas.

Para canais como estes, em que o grupo de erros possui um único gerador, o tratamento é simples e o uso do formalismo estabilizador pode ser dispensado. A verdadeira utilidade do formalismo estabilizador começa a aparecer quando tratamos de exemplos mais complexos.

2.5.4 O Código de Shor Revisitado

Revisitemos o código de Shor sob a ótica do formalismo estabilizador. O código de Shor é um código quântico estabilizador [9, 1, 3] com oito geradores. Veja a Tabela 2.3. Nesta tabela, as linhas correspondem aos geradores, e as colunas aos qubits. O fato do código de Shor poder corrigir todos os erros de um qubit pode ser verificado para todos os operadores de Pauli com peso menor ou igual a dois. Portanto, a distância deste código é de fato três. Considere, por exemplo, erros de um qubit como X_1 e Y_4. O produto X_1Y_4 anticomuta com Z_1Z_2, e portanto não está em $N(S)$. Similarmente, todos os outros produtos de dois erros do conjunto de erros ou estão em S ou senão anticomutam com pelo menos um elemento de S (e, portanto, não estão em $N(S)$), implicando que o código de Shor pode ser usado para corrigir um erro arbitrário em um qubit.

É fácil de verificar que as palavras-código $|0_L\rangle$ e $|1_L\rangle$ apresentadas em (2.19) são autovetores com autovalor $+1$ de todos os oito geradores da Tabela 2.3. Ou seja, $M_i|u_L\rangle = |u_L\rangle$

$M_{X,1}$	Z	Z	I	I	I	I	I	I	I
$M_{X,2}$	I	Z	Z	I	I	I	I	I	I
$M_{X,3}$	I	I	I	Z	Z	I	I	I	I
$M_{X,4}$	I	I	I	I	Z	Z	I	I	I
$M_{X,5}$	I	I	I	I	I	I	Z	Z	I
$M_{X,6}$	I	I	I	I	I	I	I	Z	Z
$M_{Z,1}$	X	X	X	X	X	X	I	I	I
$M_{Z,2}$	I	I	I	X	X	X	X	X	X

Tabela 2.3: Os geradores do estabilizador do código de Shor [9, 1, 3].

para $u = \{0,1\}$ e $i = 1,\ldots,8$. Todos os operadores em $\mathcal{G}_1$ que fixam ambos $|0_L\rangle$ e $|1_L\rangle$ podem ser escritos como o produto dos oito geradores acima. O conjunto de operadores que fixa $|0_L\rangle$ e $|1_L\rangle$ forma um grupo S, chamado de *estabilizador* do código, e M_1 até M_8 são os geradores deste grupo. As medidas de síndrome que comparam os qubits ou os sinais de blocos correspondem à medida dos autovalores de M_i.

Vimos na seção anterior que os geradores de um código estabilizador têm uma função similar à matriz verificação de paridade da codificação clássica. No caso do código de Shor a matriz verificação de paridade da matriz $\mathbf{G}_X$ da equação (2.21) é:

$$\mathbf{H}_X = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}. \tag{2.32}$$

As posições dos 1s na matriz (2.32) indicam exatamente as posições dos qubits com operador Z nos seis primeiros geradores da Tabela 2.3. Similarmente, note que a matriz de verificação de paridade da matriz $\mathbf{G}_Z$ da equação (2.21), expandida para nove qubits (basta tomar cada uma das duas linhas como um bloco de informação a ser codificado por $\mathbf{G}_X$), é:

$$\mathbf{H}_Z = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}, \tag{2.33}$$

e que as posições dos 1s na matriz (2.33) indicam exatamente as posições dos qubits com operador X nos dois últimos geradores da Tabela 2.3. Estes dois conjuntos de geradores Zs e Xs são *independentes* e a medida dos autovalores de ambos permite determinar a síndrome de um erro geral com geradores X e Z. Isto é possível porque, classicamente, a matriz

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{2.34}$$

é uma matriz verificação de paridade do código associado à concatenação, que como vimos tem matriz geradora $\mathbf{G} = [1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1]$ e, portanto, é capaz de corrigir quatro erros. Quanticamente, estes quatro erros clássicos correspondem aos erros X, Y, Z e I que constituem uma base de um erro quântico geral.

Da mesma forma que foi feito com o canal bit flip, como $\mathbf{G}\mathbf{H}^T = 0$ e as posições dos 1s das linhas de $\mathbf{H}$ determinam os geradores Z e X do estabilizador, podemos usar as posições dos 1s da linha de $\mathbf{G}$ para determinar as operações lógicas de erro $\overline{Z}$ e $\overline{X}$, já que $\overline{Z}$ e $\overline{X}$ são independentes dos oito geradores do estabilizador, comutam com cada um deles e anticomutam entre si. Assim, $\overline{Z} = X_1X_2X_3X_4X_5X_6X_7X_8X_9$ e $\overline{X} = Z_1Z_2Z_3Z_4Z_5Z_6Z_7Z_8Z_9$. É fácil de verificar que, para as palavras-código $|u_L\rangle$ apresentadas em (2.19), $\overline{Z}|u_L\rangle = (-1)^u|u_L\rangle$ e $\overline{X}|u_L\rangle = |(u+1)_L\rangle$ são satisfeitas para $u = \{0,1\}$.

Vimos que o código de Shor é construído a partir do código clássico de repetição de três bits. Em uma situação mais geral, podemos construir uma *classe de CBQs concatenados* construídos a partir de outros códigos lineares de bloco clássicos. Isto permitiria obter

CBQs capazes de corrigir mais de um erro quântico geral.

As duas próximas seções serão de exemplos de códigos quânticos estabilizadores mais complexos. Será feita apenas uma breve descrição com o intuito de apontar linhas de pesquisa futuras ao final deste trabalho. O entendimento do código de Shor já é suficiente para o entendimento da classe de CCQs que iremos apresentar.

2.5.5 Códigos CSS

Vimos que o código de Shor é construído a partir de um código linear clássico (o código de repetição de três bits). Outros códigos lineares clássicos podem ser usados para construir uma classe mais geral de códigos quânticos. Nesta seção faremos uma breve apresentação da classe de CCEQs proposta por Calderbank, Shor e Steane (hoje conhecida como classe CSS), que explora o conceito de código dual de um código linear clássico para construir códigos quânticos. A classe CSS, é uma importante subclasse da classe dos códigos estabilizadores. Faremos uma breve descrição de como construir um código quântico CSS. Uma demonstração lógico-formal destes resultados pode ser encontrada nas referências [6, 70, 69].

Suponha que $\mathcal{C}_1$ e $\mathcal{C}_2$ sejam códigos lineares clássicos (n, k_1) e (n, k_2), respectivamente, tal que $\mathcal{C}_2 \subset \mathcal{C}_1$, e ambos $\mathcal{C}_1$ e $\mathcal{C}_2^{\perp}$ podem corrigir erros em até t bits. Então, é possível construir um código quântico $\mathrm{CSS}(\mathcal{C}_1, \mathcal{C}_2)$ $[n, k_1 - k_2]$ que pode corrigir erros arbitrários em até t qubits [6]. O código de Shor é um código concatenado e, portanto, é um caso particular de um código CSS com $\mathcal{C}_1$ (9, 3) e $\mathcal{C}_2$ (9, 2). Os códigos CSS são códigos estabilizadores tais que cada gerador pode ser escolhido como sendo o produto tensorial de I com X *ou* Z, mas não de I com X *e* Z. Os geradores X correspondem à matriz verificação de paridade do código clássico $\mathcal{C}_2^{\perp}$ (ou seja, correspondem à matriz geradora de $\mathcal{C}_2$), que corrige os erros Z, e os geradores Z correspondem à matriz verificação de paridade de $\mathcal{C}_1$, que corrige os erros X (usando a notação definida acima).

O Código de Steane

Um exemplo importante de código CSS é o código de Steane [70, 69]. Este código é construído a partir do código de Hamming (7, 4, 3), cuja matriz verificação de paridade é

dada por:

$$\mathbf{H} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}. \tag{2.35}$$

Suponha que chamemos o código de Hamming de $\mathcal{C}$ e definamos $\mathcal{C}_1 \equiv \mathcal{C}$ e $\mathcal{C}_2 \equiv \mathcal{C}^{\perp}$. Para que estes códigos possam ser usados na definição de um código CSS, precisamos antes verificar se $\mathcal{C}_2 \subset \mathcal{C}_1$. A matriz verificação de paridade de $\mathcal{C}_2 = \mathcal{C}^{\perp}$, é igual à transposta da matriz geradora de $\mathcal{C}_1 = \mathcal{C}$:

$$\mathbf{H}[\mathcal{C}_2] = \mathbf{G}[\mathcal{C}_1]^T = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}. \tag{2.36}$$

Comparando (2.36) com (2.35), vemos que o espaço gerado pelas linhas de $\mathbf{H}[\mathcal{C}_2]$ contém estritamente o espaço gerado pelas linhas de $\mathbf{H}[\mathcal{C}_1]$, e como os correspondentes códigos são os *núcleos* de $\mathbf{H}[\mathcal{C}_2]$ e $\mathbf{H}[\mathcal{C}_1]$, concluímos que $\mathcal{C}_2 \subset \mathcal{C}_1$. Além disso, $\mathcal{C}_2^{\perp} = (\mathcal{C}^{\perp})^{\perp} = \mathcal{C}$, portanto ambos $\mathcal{C}_1$ e $\mathcal{C}_2^{\perp}$ são códigos de distância três que podem corrigir um erro de bit. Como $\mathcal{C}_1$ é um código (7, 4) e $\mathcal{C}_2$ é um código (7, 3), segue que o código CSS$(\mathcal{C}_1, \mathcal{C}_2)$ é um código quântico [7, 1] que pode corrigir erros em um qubit.

Os geradores do estabilizador deste código, apresentados na Tabela 2.4, são obtidos a partir das linhas da matriz (2.35). Note que este código é simétrico com respeito à troca dos geradores X e Z. ($\mathcal{C}_2 = \mathcal{C}_1^{\perp}$, portanto $\mathcal{C}_1$ tem que conter seu dual.). Este código tem muitas características interessantes que não serão abordadas aqui [48].

A palavra-código $|0_L\rangle$ é composta pelas oito palavras-código do código $\mathcal{C}_2$ e a palavra-código $|1_L\rangle$ é composta pelas oito palavras-código do código $\mathcal{C}_1$ *distintas* das oito palavras-código de $\mathcal{C}_2$ [48]:

$M_{X,1}$	I	I	I	Z	Z	Z	Z
$M_{X,2}$	I	Z	Z	I	I	Z	Z
$M_{X,3}$	Z	I	Z	I	Z	I	Z
$M_{Z,4}$	I	I	I	X	X	X	X
$M_{Z,5}$	I	X	X	I	I	X	X
$M_{Z,6}$	X	I	X	I	X	I	X
$\overline{Z}$	Z	Z	Z	Z	Z	Z	Z
$\overline{X}$	X	X	X	X	X	X	X

Tabela 2.4: Os geradores do estabilizador e as operações lógicas $\overline{Z}$ e $\overline{X}$ do código de Steane [7, 1, 3].

$$
\begin{aligned}
|0_L\rangle = \frac{1}{\sqrt{8}}\Big[&|0000000\rangle + |1010101\rangle + |0110011\rangle + |1100110\rangle \\
&+|0001111\rangle + |1011010\rangle + |0111100\rangle + |1101001\rangle\Big] \\
|1_L\rangle = \frac{1}{\sqrt{8}}\Big[&|1111111\rangle + |0101010\rangle + |1001100\rangle + |0011001\rangle \\
&+|1110000\rangle + |0100101\rangle + |1000011\rangle + |0010110\rangle\Big].
\end{aligned}
\tag{2.37}
$$

2.5.6 O Código Perfeito [5, 1, 3]

O último exemplo de código estabilizador é o do menor código não degenerado que pode ser construído para codificar um qubit e corrigir um erro quântico arbitrário em um qubit qualquer da palavra-código. O código perfeito [5, 1, 3] é cíclico e satura o limitante quântico de Hamming da equação (2.23). O método de construção deste código está fora do escopo desta tese (veja, por exemplo, [51, 68]).

Os geradores do estabilizador deste código são apresentados na Tabela 2.5 [48]. Este código é um exemplo de código estabilizador que *não* pertence à classe CSS. Nenhum código CSS pode codificar um qubit em cinco e corrigir um erro quântico arbitrário. Isto pode ser mostrado por eliminação. Suponha que exista um código CSS com quatro geradores. A

M_1	X	Z	Z	X	I
M_2	I	X	Z	Z	X
M_3	X	I	X	Z	Z
M_4	Z	X	I	X	Z
$\overline{Z}$	Z	Z	Z	Z	Z
$\overline{X}$	X	X	X	X	X

Tabela 2.5: Os geradores do estabilizador e as operações lógicas $\overline{Z}$ e $\overline{X}$ do código perfeito [5, 1, 3].

única possibilidade é ter dois geradores com operadores Xs. Estes geradores não podem ter ambos I em nenhuma coordenada. Portanto, cada gerador tem peso maior ou igual a três. Reordenando os qubits, o primeiro gerador é da forma $X\,X\,X\,I\,I$. Agora, qualquer escolha do segundo gerador comuta com algum operador de Pauli com dois Zs nas primeiras três coordenadas, completando a prova.

As palavras-código deste código são [48]:

$$\begin{aligned}|0_L\rangle = \frac{1}{\sqrt{4}}\Big\{&|00000\rangle + |10010\rangle + |01001\rangle + |10100\rangle \\ &+|01010\rangle - |11011\rangle - |00110\rangle - |11000\rangle \\ &-|11101\rangle - |00011\rangle - |11110\rangle - |01111\rangle \\ &-|10001\rangle - |01100\rangle - |10111\rangle + |00101\rangle\Big\} \\ |1_L\rangle = \frac{1}{\sqrt{4}}\Big\{&|11111\rangle + |01101\rangle + |10110\rangle + |01011\rangle \\ &+|10101\rangle - |00100\rangle - |11001\rangle - |00111\rangle \\ &-|00010\rangle - |11100\rangle - |00001\rangle - |10000\rangle \\ &-|01110\rangle - |10011\rangle - |01000\rangle + |11010\rangle\Big\}.\end{aligned} \tag{2.38}$$

2.6 Uma Nova Classe de Códigos Quânticos

A Figura 2.5 é uma ilustração das classes conhecidas de CCEQs.

Com excessão dos exemplos apresentados nos artigos de Chau [7, 8] e de Ollivier/Tillich [50, 49], todos os códigos quânticos construídos até hoje, dos quais os exemplos descritos neste capítulo são os mais conhecidos e simples, são códigos de bloco. Isto inclui até mesmo os códigos que não pertencem à classe dos códigos estabilizadores, que não foram discutidos aqui. A proposta desta tese é apresentar uma *classe* de CCQs estabilizadores. Como os códigos estabilizadores são construídos a partir de códigos lineares clássicos e estes possuem duas grandes subclasses, a classe dos CBCs e a classe dos CCCs, é instrutivo estudar uma classe de CCQs estabilizadores e verificar se sua performance é superior à da classe dos CBQs estabilizadores.

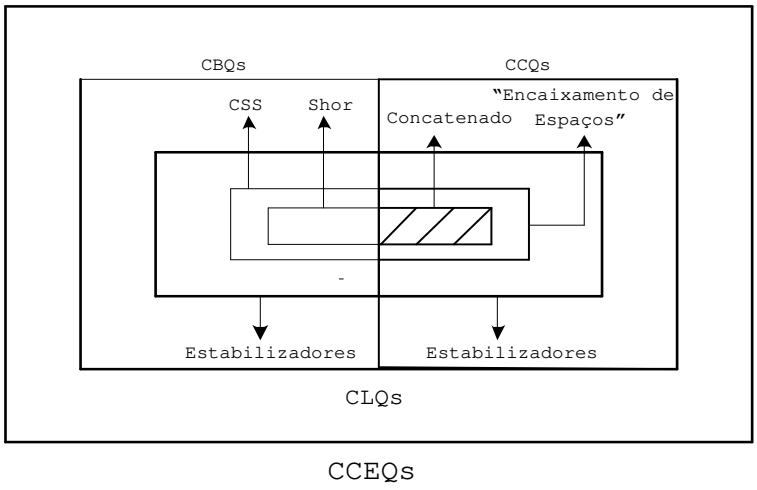

Figura 2.5: Classes conhecidas de CCEQs.

A classe preenchida por linhas transversais é a classe que será apresentada nos Capítulos 4 e 5 desta tese. Desenvolveremos o método de codificação e de decodificação de uma classe de CCQs concatenados. Este método é análogo ao utilizado para construir o código de bloco de Shor. É realmente estimulante imaginar as perspectivas de pesquisa dentro da classe dos CCQs, haja visto que ainda não se sabe nada sobre as classes mais gerais do que a classe dos CCQs concatenados, como as análogas convolucionais das classes CSS ("encaixamento de espaços") e não-CSS.

Para uma introdução à classe de CCQs, é necessário uma revisão dos CCCs. Isto será feito no Capítulo 3.

Capítulo 3

Códigos Convolucionais Clássicos

3.1 Introdução

Considere o circuito de codificação mostrado na Figura 3.1. Este codificador consiste de um registro de deslocamento contendo duas memórias e de dois somadores módulo-2. A sequência de informação $\mathbf{u}$ é deslocada da esquerda para a direita de um bit por unidade de tempo, e duas sequências codificadas $\mathbf{v}^{(1)}$ e $\mathbf{v}^{(2)}$ são geradas pelos somadores módulo-2. Portanto, a cada unidade de tempo i, o bit de informação corrente (primeira célula) é u_i, os bits de informação passados são u_{i-1} e u_{i-2} (respectivamente, a segunda e terceira células) e as duas saídas são $\mathbf{v}_i^{(1)}$ e $\mathbf{v}_i^{(2)}$. A primeira saída, $\mathbf{v}_i^{(1)}$, é a soma módulo-2 de u_i e u_{i-2}, enquanto que a segunda saída, $\mathbf{v}_i^{(2)}$, é a soma módulo-2 de u_i, u_{i-1} e u_{i-2}. A sequência de bits codificados é então multiplexada e transmitida pelo canal.

Considere a sequência de informação $\mathbf{u} = 1\,1\,0\,1\,0\,0\,1$. Assumimos que as duas últimas células inicialmente contém 0s. Então, no instante $i = 0$, o bit de informação $u_0 = 1$ é codificado em dois bits $v_0^{(1)} = 1$ e $v_0^{(2)} = 1$. No instante $i = 1$, $u_0 = 1$ é deslocado para dentro da segunda célula, e o bit 0 que havia nesta célula é deslocado para dentro da terceira célula. O bit de informação corrente $u_1 = 1$ gera dois bits codificados $v_1^{(1)} = 1$ e $v_1^{(2)} = 0$. No instante $i = 2$, $u_1 = 1$ é deslocado para dentro da segunda célula, e $u_0 = 1$ é deslocado para dentro da terceira célula. O bit de informação corrente $u_2 = 0$ gera dois bits codificados $v_2^{(1)} = 1$ e $v_2^{(2)} = 0$. Continuando com este processo, a sequência codificada inteira é $\mathbf{v} = 11\,10\,10\,00\,01\,11\,11\,01\,11$. Note que os últimos dois pares de bits codificados

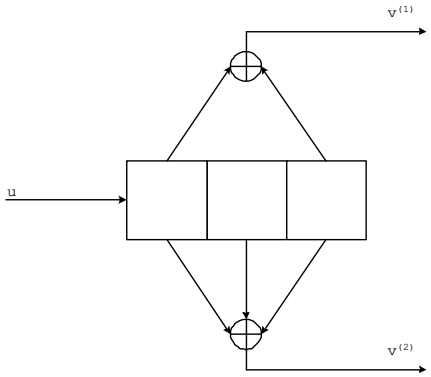

Figura 3.1: Um codificador convolucional.

são obtidos através do deslocamento de 0s após u_6, nos instantes $i = 7$ e $i = 8$.

Em geral, no instante de tempo i, um bloco de informação com k bits é deslocado para dentro do codificador e um bloco de n bits é gerado na saída do codificador. A taxa de codificação é portanto $R = k/n$. A diferença básica entre a codificação de bloco e a codificação convolucional é que na codificação de bloco, o bloco codificado no instante de tempo i depende *somente* do bloco de informação do instante i, enquanto que na codificação convolucional, o bloco codificado no instante de tempo i depende não somente do bloco de informação do instante i mas também de m blocos de informação anteriores. Portanto, um codificador convolucional requer *memória*. Com este exemplo simples, estamos prontos para estudar a codificação convolucional com mais detalhes e formalismo.

3.2 Codificadores e Códigos Convolucionais

A codificação convolucional envolve memória. Portanto, um codificador convolucional é implementado com o uso de lógica sequencial. Em um *circuito sequencial linear* (CSL) [24], os sinais escolhidos de um corpo finito $GF(q)$ (na prática, o corpo binário $GF(2)$) são aplicados simultaneamente a todos os terminais de entrada em instantes discretos de tempo. Este circuito consiste de uma rede de interconexões entre um número finito de componentes primitivos. Dois tipos de componentes primitivos são considerados em codificadores convolucionais: os *somadores* para implementar adição módulo-q e os e *elementos de memória* para atrasar (ou armazenar) um sinal de entrada por uma unidade de tempo.

As variáveis de circuito básicas de um CSL são os terminais de entrada, os terminais de saída e os estados (conteúdos dos registros de deslocamento). Neste tipo de circuito, a sequência de saída é uma função linear da sequência de entrada e dos estados, bem como a sequência de entrada também é uma função linear da sequência de saída e dos estados.

Os conceitos de codificador e de código convolucional estão intimamente relacionados, mas são distintos. Na literatura existem duas interpretações distintas, mas complementares. Uma destas interpretações define o código primeiro, como um subespaço $\mathcal{C}$ k-dimensional de um espaço vetorial n-dimensional sobre um corpo apropriado, e então define o codificador como uma matriz de dimensões $k \times n$ cujas linhas são uma base para o código. Esta interpretação foi concebida por Massey [44], e McEliece e Onyszchuk [46]. A outra interpretação define o codificador como um CSL com k entradas e n saídas, e então define o código como o conjunto de sequências de saída geradas pelo codificador para todas as possíveis sequências de entrada. Esta interpretação foi concebida por Forney em seu histórico artigo sobre os fundamentos da teoria de codificação convolucional [34], e posteriormente seguida por Piret [52], e Johannesson e Wan [30]. De acordo com a primeira interpretação, o código é o espaço gerado pelas linhas do codificador. Na segunda interpretação, o CSL realiza um mapeamento entre um espaço vetorial k-dimensional e um espaço vetorial n-dimensional, cujo conjunto imagem é o código. Quando consideradas em justaposição, estas duas interpretações descrevem os mesmos objetos, embora em uma ordem diferente.

O nível de generalização empregado na definição de codificadores e códigos convolucionais é ditado pela estrutura dos conjuntos de sequências de informação e de sequências codificadas. Elementos destes conjuntos podem ser extraídos dos seguintes corpos infinitos: o corpo $F((D))$ de séries formais de Laurent em D (onde D é um *operador de retardo*) sobre F, ou o corpo $F(D)$ de funções racionais em D sobre F, onde $F = GF(q)$. Também podem ser elementos do anel $F[[D]]$ de séries formais de potência em D sobre F, ou do anel $F[D]$ de polinômios em D sobre F. Na prática, o corpo F é usualmente o corpo binário $GF(2)$. O conjunto F^n de n-uplas de elementos de F também é um corpo. O corpo de séries formais de Laurent de elementos de F^n é denotado por $F^n((D))$. Por outro lado, o conjunto $\{F((D))\}^n$ de todas as n-uplas de elementos de $F((D))$ é também um corpo.

Aqui, adotamos a convenção de que $\{F((D))\}^n = F^n((D))$.

Considere que a sequência de informação de k-uplas no instante de tempo i seja $\mathbf{u}_i = (u_i^{(1)}, u_i^{(2)}, \ldots, u_i^{(k)})$. A sequência de informação de k-uplas pode ser representada como $\mathbf{u}(D) = \sum_{i=r}^{\infty} \mathbf{u}_i D^i$, começando em algum instante de tempo arbitrário. As entradas são zero para $i < r$. A sequência $\mathbf{u}$ é um elemento de $F^k((D))$. Similarmente, considere que a sequência codificada de n-uplas no instante de tempo i seja $\mathbf{v}_i = (v_i^{(1)}, v_i^{(2)}, \ldots, v_i^{(n)})$. A sequência codificada de n-uplas pode ser representada como $\mathbf{v}(D) = \sum_{i=r}^{\infty} \mathbf{v}_i D^i$. A sequência $\mathbf{v}$ é um elemento de $F^n((D))$. Por outro lado, se as sequências de informação e de codificação são consideradas separadamente, as k sequências de informação são $\mathbf{u}^{(i)}$, $i = 1, \ldots, k$, e podem ser representadas como o vetor $\mathbf{u} = (\mathbf{u}^{(1)}, \mathbf{u}^{(2)}, \ldots, \mathbf{u}^{(k)})$. Similarmente, as n sequências codificadas são $\mathbf{v}^{(j)}$, $i = 1, \ldots, n$, e podem ser representadas como o vetor $\mathbf{v} = (\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \ldots, \mathbf{v}^{(n)})$. Finalmente, $\mathbf{u}(D) = (\mathbf{u}^{(1)}(D), \ldots, \mathbf{u}^{(k)}(D))$, e $\mathbf{v}(D) = (\mathbf{v}^{(1)}(D), \ldots, \mathbf{v}^{(n)}(D))$.

Definição 5. *Um código convolucional $\mathcal{C}$ (n, k) sobre um corpo finito F é um subespaço k-dimensional de um espaço n-dimensional $\{F((D))\}^n$.*

Definição 6. *Um codificador é um codificador convolucional $k \times n$ sobre F se o mapeamento $F^k((D)) \rightarrow F^n((D))$ realizado pelo codificador puder ser representado por $\mathbf{v}(D) = \mathbf{u}(D)\mathbf{G}(D)$, onde $\mathbf{G}(D)$ é uma matriz $k \times n$ de posto k com entradas extraídas no subconjunto $F[D]$ de $F((D))$.*

É comum chamar $\mathbf{G}(D)$ de codificador. O codificador da Definição 6 tem taxa de codificação $R = k/n$. A Definição 6 foi proposta por Piret [52], que definiu um código convolucional como o espaço linha-$F((D))$ do codificador $\mathbf{G}(D)$. As definições de Piret são essencialmente as mesmas de Forney [34].

Outra definição que será muito útil para a correção de erros quânticos por códigos convolucionais é a de código convolucional dual.

Um subespaço V de um espaço vetorial W tem associado a si um espaço dual $V^{\perp}$. Um vetor em V é ortogonal a todo vetor em $V^{\perp}$. Além disso, se $dim(V)$ é a dimensão de V como um espaço vetorial, então $dim(V) + dim(V^{\perp}) = dim(W)$. Portanto, associado a um código convolucional $\mathcal{C}$, existe um código dual $\mathcal{C}^{\perp}$.

Definição 7. *Seja $\mathcal{C}$ um código convolucional (n, k). Então, seu código dual $\mathcal{C}^{\perp}$ é um*

subespaço $(n-k)$*-dimensional de* $\{F((D))\}^n$*, consistindo de todas as sequências* $\mathbf{v}^{\perp}(D)$ *ortogonais a todas as sequências codificadas* $\mathbf{v}(D) \in \mathcal{C}$*.*

O código dual $\mathcal{C}^{\perp}$ é um código convolucional $(n,\ n-k)$. Este código é gerado por qualquer codificador $\mathbf{H}$ de taxa $(n-k)/n$ tal que $\mathbf{G}\mathbf{H}^T = \mathbf{0}$. A matriz $\mathbf{H}$ é chamada de *matriz verificação de paridade do código* $\mathcal{C}$ ou de *matriz geradora do código dual* $\mathcal{C}^{\perp}$.

Forney [35] chamou a transposta da matriz verificação de paridade de *decodificador de síndromes*:

Definição 8. *O circuito sequencial linear com* n *entradas e* $n-k$ *saídas, cuja matriz função de transferência é* $\mathbf{H}^T(D)$*, é chamado de decodificador de síndromes, e tem a propriedade de satisfazer* $\mathbf{v}(D)\mathbf{H}^T(D) = \mathbf{0}$ *se, e somente se,* $\mathbf{v}(D) \in \mathcal{C}$*.*

Isto é, quando as n saídas de um codificador $\mathbf{G}(D)$ estão diretamente conectadas com as n entradas do correspondente decodificador de síndromes $\mathbf{H}^T(D)$, as $n-k$ saídas do decodificador de síndromes são zero para todo instante de tempo.

Seja $\mathbf{r}(D)$ a sequência recebida quando a palavra-código $\mathbf{v}(D)$ é transmitida através do canal. A sequência $\mathbf{r}(D) \in \{F((D))\}^n$ pode ser diferente da palavra-código transmitida $\mathbf{v}(D)$ pois o canal pode introduzir erros. Seja $\mathbf{e}(D)$ a sequência de erros. Então,

$$\mathbf{r}(D) = \mathbf{v}(D) + \mathbf{e}(D). \tag{3.1}$$

O decodificador de síndromes $\mathbf{H}^T(D)$ é útil na geração da sequência de síndromes correspondente à sequência recebida $\mathbf{r}(D)$.

Definição 9. *O vetor de síndromes* $\mathbf{s}(D)$ *é definido como*

$$\mathbf{s}(D) = \mathbf{r}(D)\mathbf{H}(D)^T. \tag{3.2}$$

Substituindo (3.1) em (3.2), temos que

$$\mathbf{s}(D) = \mathbf{v}(D)\mathbf{H}^T(D) + \mathbf{e}(D)\mathbf{H}^T(D). \tag{3.3}$$

Mas, $\mathbf{v}(D)\mathbf{H}^T(D) = \mathbf{0}$*, e portanto*

$$\mathbf{s} = \mathbf{e}\mathbf{H}^T. \tag{3.4}$$

Portanto, a síndrome depende somente dos erros introduzidos pelo canal e não depende da palavra-código transmitida. Vimos no Capítulo 2 que esta propriedade é de fundamental importância para a correção de erros quânticos.

3.3 Representações de Codificadores Convolucionais

3.3.1 Representação Discreta

Um codificador convolucional de taxa k/n é representado por nk sequências de geradores $\mathbf{g}_i^{(j)} = (g_{i,0}^{(j)}, g_{i,1}^{(j)}, \ldots, g_{i,m}^{(j)})$ para $i = 1, \ldots, k$ e $j = 1, \ldots, n$. A operação de codificação convolucional, como o próprio nome indica, é a convolução discreta da sequência de informação com as sequências de geradores, e é expressa como

$$\mathbf{v}^{(j)} = \sum_{i=1}^{k} \mathbf{u}^{(i)} * \mathbf{g}_i^{(j)}, \tag{3.5}$$

para $j = 1, \ldots, n$, onde $*$ é a *operação de convolução*. Assim, cada uma das n sequências codificadas $\mathbf{v}^{(j)} = (v_0^{(j)}, v_1^{(j)}, v_2^{(j)}, \ldots)$, para $j = 1, \ldots, n$ pode depender de cada uma das k sequências de informação $\mathbf{u}^{(i)} = (u_0^{(i)}, u_1^{(i)}, u_2^{(i)}, \ldots)$, para $i = 1, \ldots, k$. As sequências de geradores definem a natureza desta dependência. A sequência de gerador composta para a i-ésima entrada do codificador é definida como

$$\mathbf{g}_i = (g_{i,0}^{(1)}, g_{i,0}^{(2)}, \ldots, g_{i,0}^{(n)}, g_{i,1}^{(1)}, \ldots, g_{i,1}^{(n)}, \ldots, g_{i,m}^{(1)}, \ldots, g_{i,m}^{(n)}). \tag{3.6}$$

Para $k = 1$, a sequência de gerador composta é simplesmente

$$\mathbf{g} = (g_0^{(1)}, g_0^{(2)}, \ldots, g_0^{(n)}, g_1^{(1)}, \ldots, g_1^{(n)}, \ldots, g_m^{(1)}, \ldots, g_m^{(n)}). \tag{3.7}$$

Um codificador com parâmetros n, k, e m é chamado de um codificador (n, k, m).

Por exemplo, para o codificador convolucional (2, 1, 2) da Figura 3.1, a operação de codificação pode ser escrita com as seguintes equações:

$$\begin{aligned} v_t^{(1)} &= u_t + u_{t-2} \\ v_t^{(2)} &= u_t + u_{t-1} + u_{t-2}, \end{aligned} \tag{3.8}$$

onde definimos $u_0 = u_{-1} = 0$.

A convolução discreta (3.5) pode ser mais compactamente expressa como uma multiplicação de matrizes. As k sequências de informação podem ser escritas como a sequência de informação composta $\mathbf{u} = (\mathbf{u}_0, \mathbf{u}_1, \ldots) = (u_0^{(1)}, u_0^{(2)}, \ldots, u_0^{(k)}, u_1^{(1)}, \ldots, u_1^{(k)}, \ldots)$, onde $\mathbf{u}_t = (u_t^{(1)}, \ldots, u_t^{(k)})$ é o bloco de informação no instante de tempo t. Similarmente, a sequência codificada composta (ou palavra-código) que é obtida a partir das n sequências codificadas é $\mathbf{v} = (\mathbf{v}_0, \mathbf{v}_1, \ldots) = (v_0^{(1)}, v_0^{(2)}, \ldots, v_0^{(n)}, v_1^{(1)}, \ldots, v_1^{(n)}, \ldots)$, onde $\mathbf{v}_t = (v_t^{(1)}, \ldots, v_t^{(n)})$ é o bloco codificado no instante de tempo t. Agora, a codificação convolucional pode ser escrita como

$$\mathbf{v} = \mathbf{u}\mathbf{G}. \tag{3.9}$$

Aqui, a matriz $\mathbf{G}$ é uma matriz geradora semi-infinita da forma

$$\mathbf{G} = \begin{bmatrix} \mathbf{G}_0 & \mathbf{G}_1 & \mathbf{G}_2 & \cdots & \mathbf{G}_m & & & \\ & \mathbf{G}_0 & \mathbf{G}_1 & \cdots & \mathbf{G}_{m-1} & \mathbf{G}_m & & \\ & & \mathbf{G}_0 & \cdots & \mathbf{G}_{m-2} & \mathbf{G}_{m-1} & \mathbf{G}_m & \\ & & & \ddots & \ddots & \ddots & \ddots & \ddots \end{bmatrix}, \tag{3.10}$$

onde os espaços em branco indicam zeros, e

$$\mathbf{G}_l = \begin{bmatrix} g_{1,l}^{(1)} & g_{1,l}^{(2)} & \cdots & g_{1,l}^{(n)} \\ g_{2,l}^{(1)} & g_{2,l}^{(2)} & \cdots & g_{2,l}^{(n)} \\ \vdots & \vdots & & \vdots \\ g_{k,l}^{(1)} & g_{k,l}^{(2)} & \cdots & g_{k,l}^{(n)} \end{bmatrix}. \tag{3.11}$$

A matriz geradora $\mathbf{G}$ é uma matriz semi-infinita pois a sequência de informação pode ser infinita.

Para um codificador $(2,1,m)$, temos $\mathbf{G}_l = [g_l^{(1)}\ g_l^{(2)}]$, e a matriz geradora é

$$\mathbf{G} = \begin{bmatrix} g_0^{(1)}g_0^{(2)} & g_1^{(1)}g_1^{(2)} & g_2^{(1)}g_2^{(2)} & \cdots & g_m^{(1)}g_m^{(2)} & \\ & g_0^{(1)}g_0^{(2)} & g_1^{(1)}g_1^{(2)} & \cdots & g_{m-1}^{(1)}g_{m-1}^{(2)} & g_m^{(1)}g_m^{(2)} \\ & & & \ddots & \ddots & \ddots \end{bmatrix}. \tag{3.12}$$

Exemplo 1. *A matriz geradora semi-infinita* $\mathbf{G}$ *para o codificador (2, 1, 2) da Figura 3.1 é:*

$$\mathbf{G} = \begin{bmatrix} 11 & 01 & 11 & & & & & \\ & 11 & 01 & 11 & & & & \\ & & 11 & 01 & 11 & & & \\ & & & 11 & 01 & 11 & & \\ & & & & \ddots & \ddots & \ddots & \end{bmatrix} \tag{3.13}$$

e a sequência gerada pela sequência de informação $\mathbf{u} = 1\,1\,0\,1\,0\,0\,1$ *é obtida pela seguinte multiplicação matricial:*

$$(1101001) \begin{bmatrix} 11 & 01 & 11 & & & & & & \\ & 11 & 01 & 11 & & & & & \\ & & 11 & 01 & 11 & & & & \\ & & & 11 & 01 & 11 & & & \\ & & & & 11 & 01 & 11 & & \\ & & & & & 11 & 01 & 11 & \\ & & & & & & 11 & 01 & 11 \end{bmatrix}, \tag{3.14}$$

a qual produz $\mathbf{v} = 11\ 10\ 10\ 00\ 01\ 11\ 11\ 01\ 11$. *Note que esta sequência é a mesma que foi obtida na seção 3.1.*

No exemplo 1, uma sequência de informação $\mathbf{u}$ de comprimento $N = 7$ bits foi codificada em uma palavra-código $\mathbf{v}$ de comprimento $M = 18$ bits. Os últimos quatro bits na palavra-código correspondem a $m = 2$ zeros que são usados para retornar ao estado inicial do codificador. A primeira linha de $\mathbf{G}$ é a sequência de gerador composta $\mathbf{g} = 11\ 01\ 11$. A segunda linha é obtida através do deslocamento da primeira linha de duas posições para a direita. A terceira linha é obtida através do deslocamento da segunda linha de duas posições para a direita. E assim sucessivamente.

Em geral, ao usarmos um codificador (n, k, m), uma sequência de informação com N blocos de comprimento (isto é, com kN bits) é codificada em uma palavra-código com comprimento $M = (N + m)$ blocos, ou seja, com $n(N + m)$ bits, onde os últimos nm bits são gerados por m blocos de informação com todos os bits 0s anexados à sequência de

informação. Portanto, a matriz geradora $\mathbf{G}$ é uma matriz com kN linhas e $n(m+N)$ colunas. A cada unidade de tempo, um conjunto sucessivo de k linhas é deslocado para a direita de n posições. Portanto, a matriz geradora captura o deslocamento da sequência de informação na operação de convolução implementada com o uso de registros de deslocamento.

3.3.2 Representação Polinomial

As sequências de geradores $\mathbf{g}_i(j)$, com $i = 1,\ldots,k$ e $j = 1,\ldots,n$ são finitas e podem ser representadas como polinômios de grau finito em um operador de retardo D (também chamado de transformada-D). Os polinômios geradores para um codificador (n, k, m) são $\mathbf{g}_i^j(D) = g_{i,0}^{(j)} + g_{i,1}^{(j)}D + \cdots + g_{i,m}^{(j)}D^m$ para $i = 1,\ldots,k$ e $j = 1,\ldots,n$.

As sequências de entrada e saída do codificador podem ser escritas com o uso do operador D como $\mathbf{u}^{(i)}(D) = u_0^{(i)} + u_1^{(i)}D + u_2^{(i)}D^2 + \ldots$ e $\mathbf{v}^{(j)}(D) = v_0^{(j)} + v_1^{(j)}D + v_2^{(j)}D^2 + \ldots$, respectivamente. Como a convolução no domínio temporal é equivalente à multiplicação no domínio das transformadas, a operação de codificação pode ser escrita como

$$\mathbf{v}^{(j)}(D) = \sum_{i=1}^{k} \mathbf{u}^{(i)}(D)\mathbf{g}_i^{(j)}(D). \tag{3.15}$$

No domínio das transformadas, um codificador (n, k, m) pode ser representado por uma matriz $\mathbf{G}$ de dimensões $k \times n$, chamada de *matriz geradora polinomial* dada por

$$\mathbf{G}(D) = \begin{bmatrix} \mathbf{g}_1^{(1)}(D) & \mathbf{g}_1^{(2)}(D) & \cdots & \mathbf{g}_1^{(n)}(D) \\ \mathbf{g}_2^{(1)}(D) & \mathbf{g}_2^{(2)}(D) & \cdots & \mathbf{g}_2^{(n)}(D) \\ \vdots & \vdots & & \vdots \\ \mathbf{g}_k^{(1)}(D) & \mathbf{g}_k^{(2)}(D) & \cdots & \mathbf{g}_k^{(n)}(D) \end{bmatrix}. \tag{3.16}$$

Aqui, cada entrada $\mathbf{g}_i^{(j)}(D)$ é um polinômio. Os polinômios geradores $\mathbf{g}_i^{(j)}(D)$ para um dado i e para todo $j = 1,\ldots,n$ capturam a influência do i-ésimo registro de deslocamento sobre todas as n saídas.

Em termos da matriz geradora polinomial, a operação de codificação pode ser escrita como

$$\mathbf{v}(D) = \mathbf{u}(D)\mathbf{G}(D). \tag{3.17}$$

O vetor $\mathbf{u}(D)$ tem k componentes $\mathbf{u}^{(i)}(D)$, para $i = 1, \ldots, k$. Similarmente, o vetor $\mathbf{v}(D)$ tem n componentes $\mathbf{v}^{(j)}(D)$, para $j = 1, \ldots, n$.

3.3.3 Parâmetros de Codificadores Convolucionais

Alguns parâmetros do codificador convolucional são importantes para o estudo da capacidade de correção de erros. A seguir, apresentamos algumas definições de *comprimento* de um codificador (n, k, m) que serão úteis nas próximas seções e capítulos.

Definição 10. *O comprimento do i-ésimo registro de deslocamento é*

$$\nu_i = \max_{1 \leq j \leq n} [grau(\mathbf{g}_i^{(j)}(D))]. \tag{3.18}$$

Definição 11. *O comprimento de restrição total é*

$$\nu = \sum_{i=1}^{k} \nu_i. \tag{3.19}$$

Definição 12. *A ordem da memória do codificador é*

$$m = \max_{1 \leq i \leq k} \nu_i. \tag{3.20}$$

Definição 13. *O comprimento de restrição de saída é*

$$n_A = n(m+1). \tag{3.21}$$

Definição 14. *O comprimento de restrição de entrada é*

$$K = \nu + k. \tag{3.22}$$

O comprimento de restrição total é o número total de memórias do codificador. A ordem da memória do codificador é o grau máximo de um polinômio gerador do codificador. Para um codificador $(n, 1, m)$, temos $k = 1$ e $\nu = \nu_1 = m$. O comprimento de restrição de saída é igual ao número de bits da palavra-código afetados por um único bit de informação. O comprimento de restrição de entrada é igual ao número de bits de informação que afetam qualquer saída, $K = \nu + k$. O comprimento de restrição é um importante parâmetro para a determinação da capacidade de correção de erros de um CCC.

Exemplo 2. *Os polinômios geradores do codificador (2, 1, 2) da Figura 3.1 são* $\mathbf{g}^{(1)}(D) = 1+D^2$ *e* $\mathbf{g}^{(2)}(D) = 1+D+D^2$. *Logo, a matriz geradora é* $\mathbf{G}(D) = [\mathbf{g}^{(1)}(D),\ \mathbf{g}^{(2)}(D)] = [1+D^2,\ 1+D+D^2]$. *Este codificador tem* $\nu = 2$. *Cada bit de informação afeta três blocos de bits de saída e portanto* $n_A = 6$. *A ordem de memória é* $m = \nu = 2$ *e* $K = \nu + k = 3$.

3.3.4 Representações Gráficas

Um melhor entendimento da operação de codificação convolucional pode ser obtido com representações gráficas. Nesta seção descrevemos duas representações gráficas que são úteis para o estudo de CCCs.

Diagrama de Estados

Um codificador convolucional é um CSL e é implementado com o uso de registros de deslocamento. O conteúdo dos registros de deslocamento é chamado de *estado* do codificador. O estado de um codificador (n, k, m), no instante de tempo i, e com uma sequência de informação $\mathbf{u}$, é dado por

$$(u_{i-1}^{(1)}, u_{i-2}^{(1)}, \ldots, u_{i-\nu_1}^{(1)}, u_{i-1}^{(2)}, \ldots, u_{i-\nu_2}^{(2)}, \ldots, u_{i-1}^{(k)}, \ldots, u_{i-\nu_k}^{(k)}). \tag{3.23}$$

O próximo bloco de informação $\mathbf{u}_{i+1}$ pode alterar o estado do codificador. Todos os possíveis estados do codificador e as entradas que causam as transições entre estes estados podem ser compactamente representados pelo *diagrama de estados*. A Figura 3.2 mostra o diagrama de estados para o codificador (2, 1, 2) da Figura 3.1. Cada elipse é um estado. Usaremos a convenção em todos os diagramas de estado de que o bit mais recente é o menos significativo no estado. Assim, o estado deste codificador no instante de tempo i é formado pelos bits contidos na segunda e na primeira células e o estado no instante de tempo $i-1$ é formado pelos bits contidos na terceira e na segunda células. Os arcos representam as transições de estados causadas por uma entrada. Cada bit que entra no codificador produz uma saída (dois bits neste caso), a qual depende do bit de entrada e dos bits armazenados na memória (ou seja, do estado do codificador). Os arcos do diagrama de estados estão acompanhados dos bits de saída e o bit que entra no codificador em cada transição está indicado através de uma legenda.

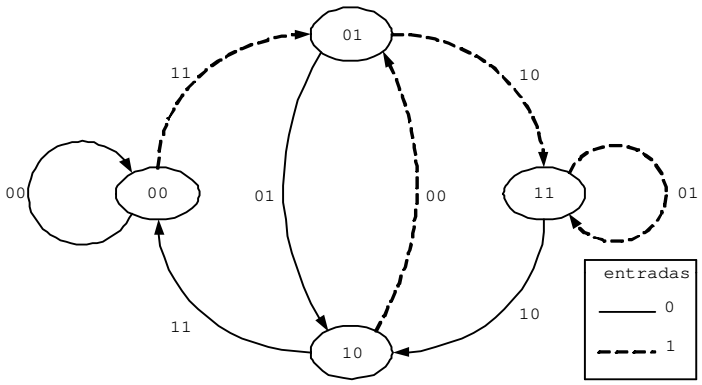

Figura 3.2: Diagrama de estados para o CCC (2, 1, 2).

A operação de codificação corresponde a uma sequência de transições de estados, começando por um estado conhecido. Em geral, existem 2^k transições saindo de cada estado, correspondendo à sequência de informação $\mathbf{u}_i$ de comprimento k. Para uma dada sequência de informação, a correspondente sequência codificada é obtida percorrendo-se o diagrama de estados a começar pelo estado todo nulo. A escolha do estado todo nulo como o estado inicial é equivalente a inicializar o codificador com todos os elementos de memória dos registros de deslocamento preenchidos por zeros. Observe pelo diagrama de estados da Figura 3.2 que a sequência gerada pelo codificador para a sequência de informação $\mathbf{u} = 1\,1\,0\,1\,0\,0\,1$ é $\mathbf{v} = 11\;10\;10\;00\;01\;11\;11\;01\;11$. Esta sequência é a mesma que foi obtida no Exemplo 1.

Diagrama de Treliça

Uma outra representação muito comum para o codificador convolucional é o *diagrama de treliça*. O diagrama de treliça para o codificador (2, 1, 2) da Figura 3.1 é mostrado na Figura 3.3. Existem quatro possíveis estados. Assume-se que o estado inicial é o estado todo nulo $a = 00$. Após $m = 2$ unidades de tempo, a treliça expande-se para incluir todos os quatro estados. Duas transições saem de cada estado; as linhas contínuas e tracejadas da treliça correspondem, respectivamente, à entrada dos bits 0 e 1 no codificador. Cada linha é acompanhada pelos correspondentes bits de saída. Os últimos m estágios da treliça são para o retorno ao estado inicial $a = 00$ e, portanto, correspondem à entrada de m blocos de

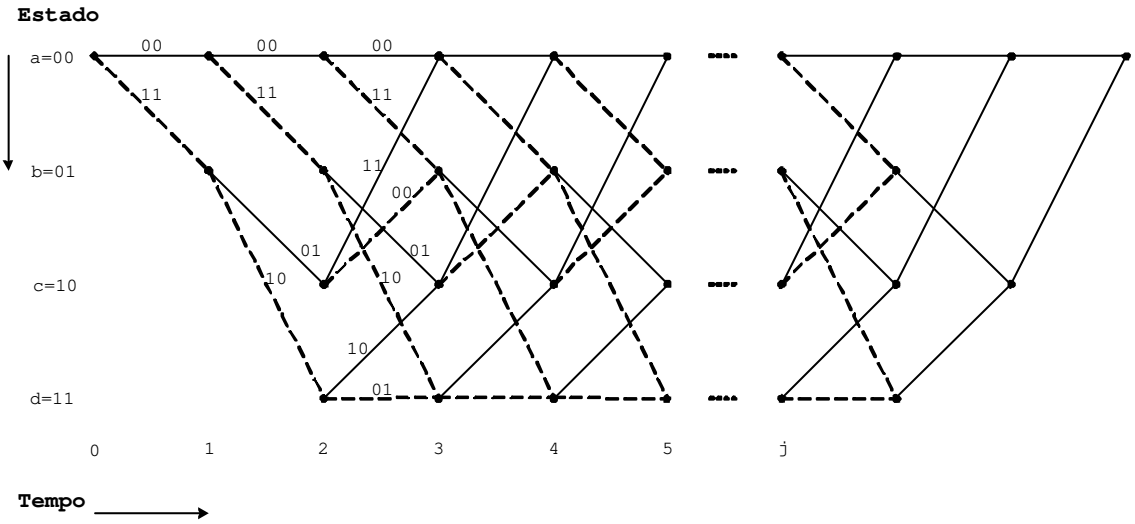

Figura 3.3: Diagrama de treliça para o CCC (2, 1, 2).

bits zeros. Cada palavra-código é um caminho que começa no estado $a = 00$ no instante de tempo $t = 0$ e termina no estado $a = 00$.

3.3.5 CCCs Equivalentes

Um código convolucional pode ser gerado por muitos codificadores. Portanto, uma noção de equivalência de codificadores pode ser introduzida. Diz-se que dois codificadores convolucionais são *equivalentes* se eles geram o mesmo código. A questão óbvia é: entre todos os codificadores equivalentes, qual deve ser escolhido para gerar um código ? Como veremos nos próximos capítulos, a resposta para esta questão é importante para a construção de códigos quânticos de menor complexidade.

Exemplo 3. *Vamos construir um codificador equivalente ao codificador da Figura 3.1 com taxa* $2/4$ *e com o mesmo número total de memórias (duas). A opção trivial é utilizarmos a mesma matriz geradora discreta semi-infinita (3.13). Esta matriz, quando associada a um codificador* $(4, 2, 1)$ *pode ser escrita como:*

$$\mathbf{G} = \begin{bmatrix} 1101 & 1100 & & & \\ 0011 & 0111 & & & \\ & 1101 & 1100 & & \\ & 0011 & 0111 & & \\ & & 1101 & 1100 & \\ & & 0011 & 0111 & \\ & & & \ddots & \ddots \\ & & & \ddots & \ddots \end{bmatrix}. \tag{3.24}$$

ou na forma de matriz geradora polinomial,

$$\mathbf{G}(D) = \begin{bmatrix} 1+D & 1+D & 0 & 1 \\ 0 & D & 1+D & 1+D \end{bmatrix}. \tag{3.25}$$

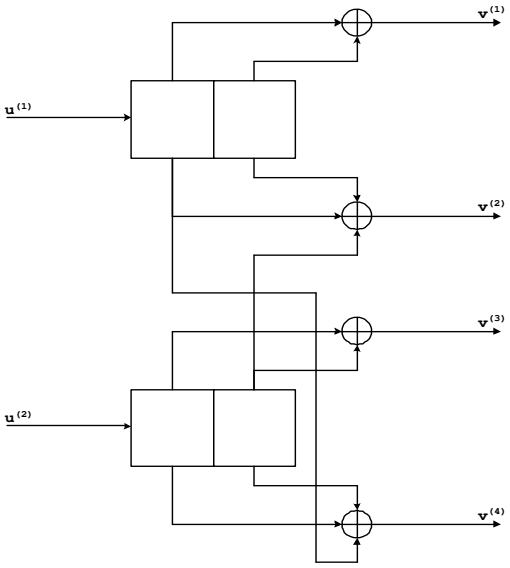

Figura 3.4: Codificador para o CCC (4, 2, 1).

Podemos construir o CSL deste codificador a partir de qualquer uma destas duas matrizes. Veja a Figura 3.4. Obviamente, o diagrama de estados para este codificador (4, 2, 1) também é o equivalente trivial do diagrama de estados da Figura 3.2. Veja a Figura 3.5.

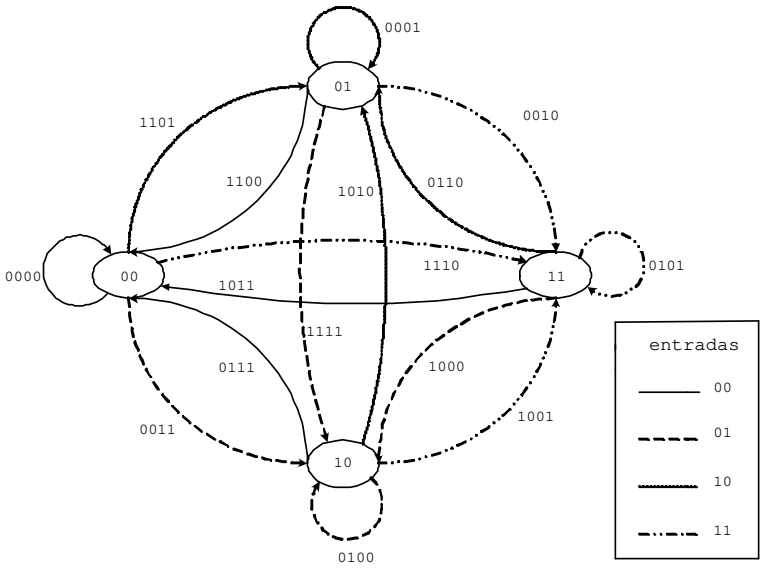

Figura 3.5: Diagrama de estados para o CCC (4, 2, 1).

Também podemos obter códigos convolucionais equivalentes a partir de codificadores não triviais, ou seja, a partir de codificadores com *diferentes* matrizes geradoras. Como veremos no Capítulo 5, esta possibilidade poderá ser útil para a construção de bons CCQs concatenados.

3.4 Distância Livre

É interessante considerar a distância entre quaisquer duas palavras-código em um CCC. Costello [31, 32] definiu uma medida de distância chamada de *distância livre*. Sejam $\mathbf{u}$ e $\mathbf{u}'$ duas sequências de informação, e $\mathbf{v}$ e $\mathbf{v}'$ as palavras-código correspondentes a $\mathbf{u}$ e $\mathbf{u}'$, respectivamente. Cabe então a definição:

Definição 15. *A distância livre de um código convolucional é*

$$\begin{aligned} d_{free} &= \min\{d_H(\mathbf{v},\mathbf{v}') : \mathbf{u} \neq \mathbf{u}'\} \\ &= \min\{w_H(\mathbf{v}) : \mathbf{u} \neq \mathbf{0}\} \\ &= \min\{w_H(\mathbf{uG}) : \mathbf{u} \neq \mathbf{0}\}. \end{aligned} \tag{3.26}$$

Portanto, a distância livre é a distância de Hamming mínima entre quaisquer duas

palavras-código distintas. Por causa da linearidade dos códigos convolucionais, a distância livre é também o menor peso da palavra-código não-nula. Em um diagrama de estados ou de treliça, o *caminho de* d_{free} é o caminho não nulo de menor peso de Hamming saindo do estado $a = 00$ e retornando ao estado $a = 00$. Na maioria dos casos, o caminho de d_{free} também é o caminho não nulo mais curto saindo do estado $a = 00$ e retornando ao estado $a = 00$, mas há exceções.

Exemplo 4. *Através do diagrama de estados da Figura 3.2 ou do diagrama de treliça da Figura 3.3, pode-se facilmente verificar que a distância livre do código gerado pelo codificador da Figura 3.1 é* $d_{free} = 5$. *O caminho de* d_{free} *é constituído de três transições de estados, a saber:* $a = 00 \rightarrow b = 01 \rightarrow c = 10 \rightarrow a = 00$, *referentes à palavra-código* 11 01 11 *e à sequência de informação* $\mathbf{u} = 100$.

Diz-se que um código convolucional tem *distância livre ótima* (DLO) se sua distância livre é igual ou superior a qualquer outro código convolucional de mesma taxa e mesmo comprimento de restrição de saída.

A partir da definição de distância livre, cabe também uma definição para a capacidade máxima de correção de erros destes códigos:

Definição 16. *A máxima capacidade de correção de erros* t_{free} *de um código convolucional é*

$$t_{free} = \lfloor \frac{d_{free} - 1}{2} \rfloor. \tag{3.27}$$

Obviamente, os códigos gerados por codificadores equivalentes possuem o mesmo d_{free} e, consequentemente, a mesma capacidade de correção.

O parâmetro t_{free} é usado para analizar o desempenho de algoritmos de decodificação baseados no princípio de máxima verossimilhança, como o algoritmo de Viterbi e o algoritmo de decodificação por síndromes.

3.4.1 Limitantes da Distância Livre

Um conhecido limitante inferior para a distância livre d_{free} é o limitante de Gilbert [33], dado por:

$$\frac{d_{free}}{n_A} \geq H^{-1}(1-R), \tag{3.28}$$

onde $R = k/n$ é a taxa de codificação, $n_A = n(m+1)$ é o comprimento de restrição de saída, e $H(x)$ é a função de entropia para o alfabeto binário, dada por:

$$H(x) = -xlog_2x - (1-x)log_2(1-x), \ 0 \geq x \geq 1. \tag{3.29}$$

Heller [28] encontrou um limitante superior para a distância livre d_{free} dos códigos convolucionais de taxa $1/n$:

$$d_{free} \leq \min_{j \leq 1} \lfloor \frac{n}{2} \frac{2^j}{2^j - 1} (\nu + j) \rfloor. \tag{3.30}$$

3.5 Classes de CCCs

3.5.1 Codificadores Catastróficos e Não Catastróficos

Massey [43] denominou os codificadores que geram uma palavra-código de peso finito para uma sequência de informação de peso infinito de codificadores *catastróficos*. Nestes casos, existe uma probabilidade não nula de que quando esta palavra-código é transmitida através de um canal ruidoso, um número finito de erros introduzidos pelo canal possa trocar todos os dígitos codificados não nulos por zeros, fazendo com que o decodificador gere uma saída toda nula no receptor. Portanto, um número finito de erros de canal pode causar um número infinito de erros de decodificação. Esta situação, claramente indesejável, é conhecida como *propagação catastrófica de erros*.

Massey e Sain propuseram um teorema para determinar se um codificador é ou não catastrófico [45]. Segundo este teorema, para evitar a propagação catastrófica de erros os codificadores de taxa $1/n$ devem satisfazer a condição:

$$\text{mdc}\left[\mathbf{g}^{(j)}(D), j = 1, 2, \ldots, n\right] = 1, \tag{3.31}$$

onde mdc é o *máximo divisor comum*. Discutiremos o caso de um codificador mais geral, de taxa k/n, na seção mais adiante onde introduziremos o teorema do fator invariante.

O diagrama de estados de um codificador catastrófico tem a seguinte propriedade:

Propriedade 1. *Um codificador é catastrófico se, e somente se, seu diagrama de estados tem um ciclo de peso zero que não seja a auto-malha em torno do estado todo nulo.*

Exemplo 5. *Para o codificador (2, 1, 2) com polinômios geradores* $\mathbf{g}^{(1)}(D) = 1+D$ *e* $\mathbf{g}^{(2)}(D) = 1+D^2$*, temos mdc* $[\mathbf{g}^{(1)}(D), \mathbf{g}^{(2)}(D)] = 1+D$*, e portanto o codificador é catastrófico.*

Considere a sequência de informação $\mathbf{u}(D) = 1/(1+D)$*, isto é, a sequência* $\mathbf{u} = 1\,1\,1\ldots$*. Neste caso, as sequências codificadas são* $\mathbf{v}^{(1)} = \mathbf{u}(D)\mathbf{g}^{(1)}(D) = 1$ *e* $\mathbf{v}^{(2)} = \mathbf{u}(D)\mathbf{g}^{(2)}(D) = 1+D$*. Isto corresponde à sequência codificada* $\mathbf{v} = 11\,01\,00\,00\ldots$*. Se erros de canal afetarem os três bits não nulos, a sequência recebida será* $\mathbf{r} = 00\,00\,00\ldots$*. Neste caso, o decodificador irá produzir a sequência toda nula como estimativa da sequência de informação original, resultando portanto em um número infinito de erros de decodificação. O diagrama de estados é mostrado na Figura 3.6. Observe que existe uma auto-malha de peso zero em torno do estado* 11*.*

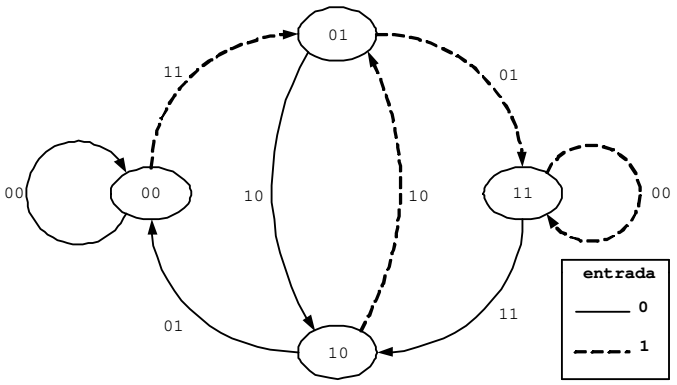

Figura 3.6: Diagrama de estados para o CCC (2, 1, 2) catastrófico.

3.5.2 Codificadores e Códigos Puncionados

Um código convolucional de taxa $R = k'/n'$ pode ser gerado por muitos codificadores de taxa $R = k/n$, onde $k \geq k'$ e $n > n'$. Nesta subseção descreveremos um processo conhecido como *puncionamento* que nos permite gerar códigos de diferentes taxas através do uso de

um único codificador. Um código puncionado é obtido através do apagamento periódico (ou puncionamento) de dígitos das sequências codificadas. O puncionamento tem o efeito de reduzir o número de dígitos codificados correspondentes aos dígitos de informação, ou seja, tem o efeito de aumentar a taxa do código. Portanto, um codificador de taxa baixa pode ser usado para gerar muitos códigos de taxa alta através da seleção apropriada do padrão de puncionamento (também conhecido como mapa de apagamento ou padrão de perfuração). O codificador de taxa baixa é chamado de codificador *original*, enquanto que o codificador de taxa alta obtido após o puncionamento é o codificador *puncionado*. Se um codificador original de taxa $1/n$ é puncionado através do apagamento de alguns dos nP bits codificados correspondentes a P bits de informação, então P é chamado de *período de puncionamento*. Isto ficará mais claro com o exemplo a seguir.

Exemplo 6. *Considere que os bits codificados gerados pelo codificador (2, 1, 2) com matriz geradora* $\mathbf{G} = [1+D^2,\ 1+D+D^2]$ *sejam agrupados em blocos de quatro bits. Cada um destes blocos corresponde a dois bits de informação. Se deletarmos um bit (digamos, o terceiro) de cada bloco de quatro, ficaremos com três bits correspondendo a dois bits de informação. Ou seja, estaremos gerando um codificador puncionado de taxa* $R = 2/3$.

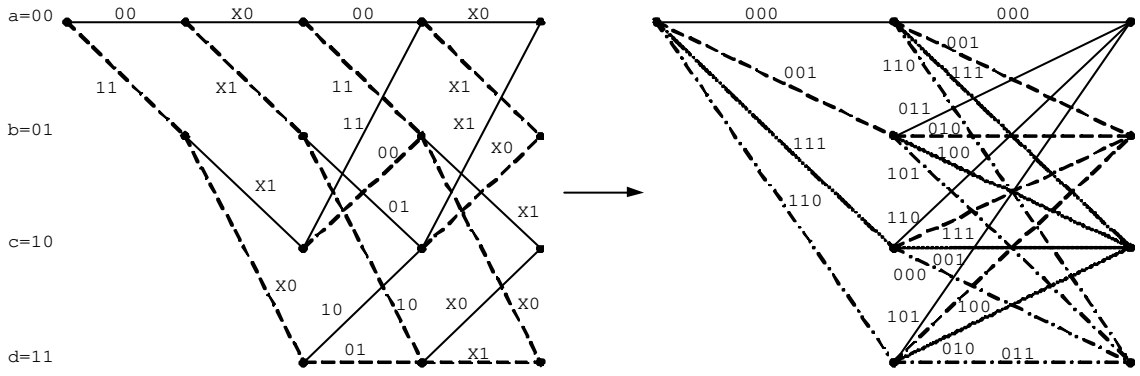

Figura 3.7: Ação de Puncionamento sobre o terceiro bit e com $P = 2$.

A ação de puncionamento pode ser melhor descrita através de um diagrama de treliça. A Figura 3.7 mostra os quatro estágios iniciais da treliça do codificador original com os bits puncionados (representado pela letra "X") e os dois primeiros estágios da treliça do correspondente codificador puncionado. A treliça do codificador puncionado é obtida

através da "fusão" (dois a dois) dos quatro estágios da treliça do codificador original, com a remoção dos bits puncionados.

A matriz geradora discreta semi-infinita do codificador puncionado $R = 2/3$ *pode ser obtida através do apagamento de todas as terceiras colunas dentro de cada grupo de quatro colunas da matriz geradora (3.13). Ou seja, a matriz geradora do codificador puncionado será:*

$$\mathbf{G} = \begin{bmatrix} 111 & 110 & & \\ 001 & 011 & & \\ & 111 & 110 & \\ & 001 & 011 & \\ & & \ddots & \ddots \\ & & \ddots & \ddots \end{bmatrix}. \tag{3.32}$$

Repare que o número de estados permanece inalterado. Assim, o codificador puncionado é de memória unitária. Veja-o na Figura 3.8.

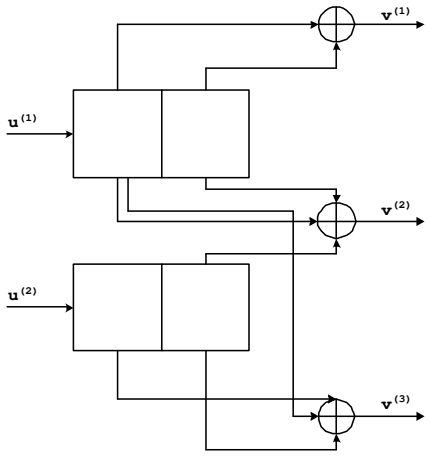

Figura 3.8: Codificador puncionado (3, 2, 1) gerado de um codificador (2, 1, 2).

Este código puncionado também poderia ser obtido a partir de um código de taxa $R = 1/3$ *com geradores* $\mathbf{G}(D) = [1+D+D^2,\ 1+D^2,\ 1+D+D^2]$ *que tivesse o primeiro e o segundo bits apagados da primeira seção da treliça e o terceiro bit apagado da segunda seção da treliça.*

O puncionamento reduz a distância livre do código. Se o d_{free} do código puncionado de uma dada taxa e comprimento de restrição for comparável ao d_{free} de um código ordinário de mesma taxa e comprimento de restrição, o puncionamento será útil pois poderemos usar a estrutura de treliça do codificador original (de menor complexidade) no algoritmo de decodificação do código puncionado.

No exemplo acima, o d_{free} caiu de cinco para três (palavra-código 001 011). No entanto, este d_{free} é o mesmo alcançado pelo melhor codificador (3, 2, 1). A vantagem do codificador puncionado é que poderemos usar a treliça do codificador original (de menor complexidade, já que chegam a cada estado apenas dois caminhos ao invés dos quatro caminhos da treliça de um codificador ordinário (3, 2, 1)) no algoritmo de decodificação.

Como veremos no Capítulo 5, o puncionamento será uma opção simples de construção de bons CCQs, podendo ser útil tanto em termos de distância quanto de complexidade. Mas não será abordado em profundidade, pois desconhecemos um procedimento sistemático de obtenção do código dual puncionado a partir do código original (ou de seu dual). Um estudo mais abrangente de codificadores e códigos puncionados pode ser encontrado em referências especializadas [14, 10].

3.5.3 Codificadores de Memória Unitária

Lee [40] definiu uma classe de codificadores convolucionais de memória unitária que têm a maior distância livre entre todos os codificadores de mesma taxa e complexidade de estados.

Considere um codificador de taxa k_0/n_0 com ordem de memória m e com equações de codificação no instante t dadas por

$$\mathbf{v}_t = \mathbf{u}_t\mathbf{G}_0 + \mathbf{u}_{t-1}\mathbf{G}_1 + \ldots + \mathbf{u}_{t-m}\mathbf{G}_m, \tag{3.33}$$

O codificador (n_0, k_0, m) é equivalente a um codificador ($n' = mn_0$, $k' = mk_0$, $m' = 1$) com

$$\mathbf{G}'_0 = \begin{bmatrix} \mathbf{G}_0 & \mathbf{G}_1 & \cdots & \mathbf{G}_{m-1} \\ 0 & \mathbf{G}_0 & \cdots & \mathbf{G}_{m-2} \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \mathbf{G}_0 \end{bmatrix} \quad \text{e} \quad \mathbf{G}'_1 = \begin{bmatrix} \mathbf{G}_m & 0 & \cdots & 0 \\ \mathbf{G}_{m-1} & \mathbf{G}_m & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ \mathbf{G}_1 & \mathbf{G}_2 & \cdots & \mathbf{G}_m \end{bmatrix}. \tag{3.34}$$

Codificadores com ordem de memória igual a um são chamados de *codificadores de memória unitária*. Lee definiu a *complexidade de estado* de um codificador binário (n_0, k_0, m) como 2^{mk_0}, porque existem 2^{mk_0} estados distintos do codificador. É claro que a complexidade dos dois codificadores equivalentes acima é a mesma. Portanto, o valor máximo de d_{free} é alcançado dentro do subconjunto de codificadores convolucionais com $m = 1$. Lauer [39] definiu a complexidade de estado μ de um codificador de memória unitária como sendo o posto de $\mathbf{G}'_1$, a qual pode ser diferente de mk_0. Os codificadores com $m' = 1$ e $\mu < k'$ são chamados de codificadores de *memória unitária parcial* (MUP).

Como veremos no Capítulo 5, a classe dos codificadores com memória unitária completa e parcial serão muito importantes para a construção de bons CCQs concatenados.

3.6 Decodificação e Correção de Erros para CCCs

We are motivated to take a closer look at dual codes than our cursory survey ... This paper develops some initial results on the analysis of convolutional codes via their dual codes; however, it seems that we have not exhausted the productiveness of this viewpoint, and that it may eventually be useful in exploring the "terra incognita" of weight and distance properties of convolutional codes.

– G. D. Forney, em [35].

O melhor, e também o mais conhecido, algoritmo de decodificação de CCCs é o algoritmo de Viterbi [14, 73, 72, 36]. Dada a sequência recebida à saída do canal, este algoritmo utiliza o critério de máxima verossimilhança[1] para estimar a sequência transmitida. Porém, para que um algoritmo clássico possa ser aproveitado no contexto quântico

[1] Por este critério, o algoritmo estima a sequência transmitida $\hat{\mathbf{v}}$, dada a sequência recebida $\mathbf{r}$, através da *minimização da probabilidade* em que $\hat{\mathbf{v}}$ e $\mathbf{v}$ diferem, isto é, $\hat{\mathbf{v}}$ é sequência mais provável dado que $\mathbf{r}$ foi recebido.

necessitamos de um algoritmo que estime os erros do canal sem uma medida direta da sequência recebida. Isto só é possível se utilizarmos o conceito de síndrome de erro, da mesma forma que foi utilizada para os CBQs no Capítulo 2.

Reed e Truong [56, 57, 58] desenvolveram um algoritmo de decodificação por síndromes (ADS) baseado no princípio de máxima verossimilhança semelhante ao algoritmo de Viterbi. Vale a pena descrever em detalhes este algoritmo de decodificação já que é pouco conhecido mesmo entre especialistas da área de códigos. Dedicaremos as duas próximas subseções a esta tarefa. A primeira subseção descreve o algoritmo para o caso simples de um codificador de taxa $1/2$. A segunda subseção descreve o algoritmo para o caso geral de codificadores de taxa k/n. No Capítulo 4, adaptaremos este algoritmo para a identificação e correção de erros em CCQs.

3.6.1 ADS para um CCC (2, 1, m)

Para um CCC (2, 1, m), a matriz geradora polinomial é uma matriz de dimensões 1×2 da forma $\mathbf{G}(D) = [\mathbf{g}^{(1)}(D), \mathbf{g}^{(2)}(D)]$. Para evitar a possibilidade de propagação catastrófica de erros é necessário que $\text{mdc}[\mathbf{g}^{(1)}(D), \mathbf{g}^{(2)}(D)] = 1$.

A matriz verificação de paridade $\mathbf{H}(D)$, associada à matriz geradora $\mathbf{G}(D)$, é uma matriz polinomial de dimensões 1×2 sobre $GF(2)$ em que $\mathbf{G}(D)\mathbf{H}^T(D) = 0$ é satisfeita. Para um CCC (2, 1, m) não sistemático[2] com $\mathbf{G}(D)$, é fácil de verificar que

$$\mathbf{H}(D) = [\mathbf{g}^{(2)}(D), \mathbf{g}^{(1)}(D)]. \tag{3.35}$$

Exemplo 7. *A matriz* $\mathbf{H}$ *para o codificador (2, 1, 2) com* $\mathbf{G}(D) = [1+D^2, 1+D+D^2]$ *é* $\mathbf{H}(D) = [1+D+D^2, 1+D^2]$. *É fácil de verificar que a relação* $\mathbf{G}\mathbf{H}^T = 0$ *é satisfeita. Na forma de matriz discreta semi-infinita, esta* $\mathbf{H}$ *escreve-se como:*

[2]Um codificador convolucional (n, k, m) é chamado de *sistemático* se $\mathbf{v}^{(j)} = \mathbf{u}^{(j)}$ para $j = 1, \ldots, k$.

$$\mathbf{H} = \begin{bmatrix} 11 & & & & & & \\ 10 & 11 & & & & & \\ 11 & 10 & 11 & & & & \\ & 11 & 10 & 11 & & & \\ & & 11 & 10 & 11 & & \\ & & & \ddots & \ddots & \ddots \end{bmatrix}. \tag{3.36}$$

Portanto, pelas equações (3.35) e (3.4), a síndrome para um CCC (2, 1, m) é

$$\mathbf{s}(D) = \mathbf{e}^{(1)}(D)\mathbf{g}^{(2)}(D) + \mathbf{e}^{(2)}(D)\mathbf{g}^{(1)}(D). \tag{3.37}$$

A equação (3.37) e, mais geralmente a equação (3.4), são equações de Diofanto lineares sobre o espaço $F[D]$ de polinômios em D. Usando técnicas similares àquelas usadas na teoria dos números (veja o Apêndice A), a solução geral da equação (3.37) pode ser facilmente encontrada. Como o mdc$(\mathbf{g}^{(1)}(D), \mathbf{g}^{(2)}(D)) = 1$, o algoritmo de Euclides pode ser usado para encontrar polinômios $\alpha_1(D)$ e $\alpha_2(D)$ tais que $\alpha_1(D)\mathbf{g}^{(2)}(D) + \alpha_2(D)\mathbf{g}^{(1)}(D) = 1$. Em termos de $\alpha_k(D)$, as solução gerais da equação (3.37) são:

$$\begin{aligned} \mathbf{e}^{(1)}(D) &= \alpha_1(D)\mathbf{s}(D) + \mathbf{g}^{(1)}(D)\mathbf{t}(D) \\ \mathbf{e}^{(2)}(D) &= \alpha_2(D)\mathbf{s}(D) - \mathbf{g}^{(2)}(D)\mathbf{t}(D), \end{aligned} \tag{3.38}$$

onde $\mathbf{t}(D)$ é um polinômio arbitrário em $F[D]$.

Para um CCC (2, 1, m), uma possível sequência de erro $\mathbf{e}(D)$ tem a forma $\mathbf{e}(D) = [\mathbf{e}^{(1)}(D), \mathbf{e}^{(2)}(D)]$, onde $\mathbf{e}^{(1)}(D)$ e $\mathbf{e}^{(2)}(D)$ são polinômios de grau finito sobre $GF(2)$. O peso de Hamming de $\mathbf{e}(D)$ é escrito como $W_H[\mathbf{e}(D)] = W_H[\mathbf{e}^{(1)}(D)] + W_H[\mathbf{e}^{(2)}(D)]$, onde $W_H[\mathbf{e}^{(1)}(D)]$ e $W_H[\mathbf{e}^{(2)}(D)]$ são, respectivamente, os números de coeficientes não nulos de $\mathbf{e}^{(1)}(D)$ e $\mathbf{e}^{(2)}(D)$.

Estamos agora prontos para apresentar um ADS que eficientemente encontre $\hat{\mathbf{e}}(D)$, a estimativa da sequência de erro.

Implementação do ADS para um CCC (2, 1, m) - Exemplo

Apresentamos a seguir, através de um exemplo, um algoritmo recursivo de decodificação por síndromes. Considere novamente o CCC não sistemático (2, 1, 2) do Exemplo 7. Vimos que para este CCC temos $\mathbf{G}(D) = [\mathbf{g}^{(1)}(D),\, \mathbf{g}^{(2)}(D)] = [1+D^2,\, 1+D+D^2]$ e $\mathbf{H}(D) = [\mathbf{g}^{(2)}(D),\, \mathbf{g}^{(1)}(D)] = [1+D+D^2,\, 1+D^2]$. Substituindo $\mathbf{g}^{(i)}(D)$ para $i = 1,2$ na equação (3.37) temos:

$$\mathbf{s}(D) = \mathbf{r}^{(1)}(D) + D\mathbf{r}^{(1)}(D) + D^2\mathbf{r}^{(1)}(D) + \mathbf{r}^{(2)}(D) + D^2\mathbf{r}^{(2)}(D). \tag{3.39}$$

Finalmente, as soluções da equação diofantina 3.37 são:

$$\begin{aligned} \mathbf{e}^{(1)}(D) &= D\mathbf{s}(D) + \mathbf{t}(D) + D^2\mathbf{t}(D) \\ \mathbf{e}^{(2)}(D) &= \mathbf{s}(D) + D\mathbf{s}(D) + \mathbf{t}(D) + D\mathbf{t}(D) + D^2\mathbf{t}(D), \end{aligned} \tag{3.40}$$

pois $\alpha_1(D) = D$ e $\alpha_2(D) = 1+D$ constituem uma solução particular de $\alpha_1(D)\mathbf{g}^{(2)}(D) + \alpha_2(D)\mathbf{g}^{(1)}(D) = 1$.

O CSL gerado pela matriz $\mathbf{H}^T$ relacionando as sequências $\mathbf{v}(D)$, $\mathbf{e}(D)$, $\mathbf{r}(D)$ e $\mathbf{s}(D)$ é apresentado na Figura 3.9. Schalkwijk e Vinck [60, 61, 62] usaram os estados deste CSL como os estados de seu diagrama de treliça.

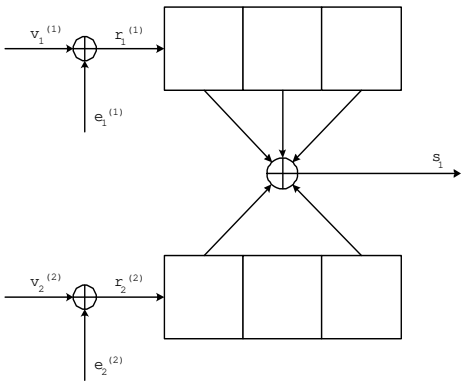

Figura 3.9: Decodificador de síndromes para o CCC (2, 1, 2).

Já no algoritmo que vamos descrever abaixo, os estados do diagrama de treliça são equivalentes aos estados do registro de deslocamento necessários nas equações (3.40) para armazenar sequencialmente as versões atrasadas de $\mathbf{t}(D)$. Ou seja, os estados do diagrama de treliça são da forma $(D\mathbf{t}(D), D^2\mathbf{t}(D))$. A Tabela 3.1 apresenta as transições permitidas entre estes estados.

$D\mathbf{t}(D)$, $D^2\mathbf{t}(D)$	$\mathbf{t}(D)$=0	$\mathbf{t}(D)$=1
$a = 00$	$a = 00$	$c = 10$
$b = 01$	$a = 00$	$c = 10$
$c = 10$	$b = 01$	$d = 11$
$d = 11$	$b = 01$	$d = 11$

Tabela 3.1: Tabela de estados do registro de deslocamento para $\mathbf{t}(D)$.

Ilustramos a aplicação do ADS através do exemplo da Figura 3.10. Nesta figura, uma transição de linha sólida corresponde à entrada $\mathbf{t}(D) = 0$ e uma transição de linha tracejada corresponde à entrada $\mathbf{t}(D) = 1$. Vamos assumir que a síndrome seja 0 antes do estágio 0. Em outras palavras, isto significa dizer que assumimos que o estado inicial do decodificador da Figura 3.9 seja o estado todo nulo. Da mesma forma, é assumido que o estado inicial do registro de deslocamento de $\mathbf{t}(D)$ seja $(D\mathbf{t}(D), D^2\mathbf{t}(D)) = (0,0)$. Assim, no estágio 0, o algoritmo está no estado $a = 00$.

Para este exemplo, assuma que a mensagem a ser codificada tem comprimento de apenas dois bits (mensagens de comprimento maior inviabilizariam a reprodução da treliça de decodificação nesta página) $\mathbf{u}(D) = (1\ 0)$. As sequências $\mathbf{v}^{(1)}(D) = (1\ 0\ 1\ 0)$ e $\mathbf{v}^{(2)}(D) = (1\ 1\ 1\ 0)$ são as duas componentes de $\mathbf{v}(D) = (11\ 01\ 11\ 00)$. Considere que a mensagem recebida à saída do canal seja $\mathbf{r}(D) = (01\ 00\ 11\ 00)$. Então $\mathbf{r}^{(1)}(D) = (0\ 0\ 1\ 0)$ e $\mathbf{r}^{(2)}(D) = (1\ 0\ 1\ 0)$ são as duas componentes de $\mathbf{r}(D)$. Com o uso da equação (3.39), é fácil de verificar que a sequência de síndrome é $\mathbf{s}(D) = 1 + D^2 + D^3 = (1\ 0\ 1\ 1\ 0\ 0)$. Estes dígitos estão colocados sobre as correspondentes seções da treliça da Figura 3.10.

O ADS é semelhante ao algoritmo de Viterbi [73, 72, 36], só que aqui como não se conhece a sequência recebida $\mathbf{r}(D)$, devemos usar as equações (3.40) para o cálculo da sequência de erro $\mathbf{e}(D)$. Para cada seção da treliça temos associado um par de síndromes

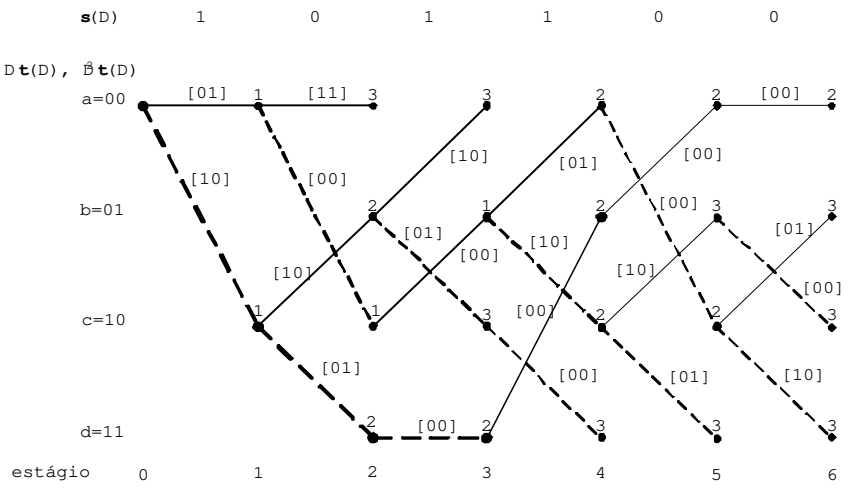

Figura 3.10: ADS para o CCC (2, 1, 2) com $\mathbf{s}(D) = (1\,0\,1\,1\,0\,0)$.

($\mathbf{s}(D)$ e $D\mathbf{s}(D)$) a ser usado nas equações (3.40) para as diferentes transições de estados ($D\mathbf{t}(D)$, $D^2\mathbf{t}(D)$) permitidas para aquela seção. Os pares $[\mathbf{e}^{(1)}(D), \mathbf{e}^{(2)}(D)]$ para diferentes seções de treliça e transições de estados, bem como a métrica (peso de Hamming) acumulado em cada nó da treliça, estão colocados na Figura 3.10.

A partir do estágio 3, chegam dois caminhos distintos a cada nó da treliça. Deve-se calcular a métrica acumulada de ambos os caminhos e selecionar o de *menor* métrica. Se ocorrer empate das métricas, podemos selecionar *qualquer um* dos dois caminhos possíveis, exatamente como é feito no algoritmo de Viterbi. Assim, continuaremos a ter apenas um caminho chegando a cada nó e, consequentemente, quatro caminhos chegando a cada estágio da treliça. Este procedimento deve ser repetido até o final da treliça. Finalmente, no último estágio, devemos selecionar entre os quatro caminhos que sobreviveram, o caminho com a menor métrica, também conhecido como *caminho sobrevivente*.

Note que no estágio 6 da Figura 3.10, o caminho de menor peso é $a \rightarrow c \rightarrow d \rightarrow d \rightarrow b \rightarrow a \rightarrow a$. Os ramos deste caminho produzem $\hat{\mathbf{e}}(D) = [\hat{\mathbf{e}}^{(1)}(D), \hat{\mathbf{e}}^{(2)}(D)] = [1, D] = [10\,01\,00\,00]$ como estimativa do vetor erro. Se subtrairmos estes vetores de $\mathbf{r}^{(1)}(D)$ e $\mathbf{r}^{(2)}(D)$, teremos $\hat{\mathbf{v}}^{(1)}(D) = [1 + D^2]$ e $\hat{\mathbf{v}}^{(2)}(D) = [1 + D + D^2]$ como estimativas das sequências codificadas. Finalmente, usa-se o circuito sequencial linear inverso de Massey e Sain [45] com equação $\hat{\mathbf{u}}(D) = (1 + D)\hat{\mathbf{v}}^{(1)}(D) + D\hat{\mathbf{v}}^{(2)}(D)$ para produzir a sequência de informação $\hat{\mathbf{u}}(D) = (1\ 0)$

como uma estimativa (correta) da mensagem original.

Este exemplo será usado no Capítulo 4 para entendermos o processo de identificação e correção de erros quânticos.

3.6.2 Generalização: ADS para CCC (n, k, m)

O método de decodificação descrito na subseção 3.6.1 para CCCs (2, 1, m) pode ser generalizado para CCCs (n, k, m). Vimos que o algoritmo de decodificação consiste em encontrar a solução geral das equações de síndrome para o vetor polinomial de erro $\mathbf{e}(D)$. Estas equações de síndrome são equações diofantinas lineares sobre o anel de polinômios $F[D]$ com coeficientes pertencentes a um corpo finito, no caso o $GF(2)$. O conjunto de soluções de Diofanto para as equações de síndrome constituem uma *classe lateral* do CCC. O problema da decodificação por síndromes é encontrar o vetor polinomial de menor peso $\hat{\mathbf{e}}(D)$ desta classe lateral e subtraí-lo do vetor polinomial recebido $\mathbf{r}(D)$ para produzir uma estimativa $\hat{\mathbf{v}}(D)$ do vetor polinomial transmitido. Para encontrar $\hat{\mathbf{e}}(D)$ eficientemente, desenvolveremos uma generalização do ADS apresentado acima. Assim como na subseção anterior, apresentaremos um exemplo para descrever o ADS para taxas $R = k/n$. Este exemplo será utilizado no Capítulo 5 para a decodificação de um CCQ concatenado de taxa 1/9.

Alguns dos resultados exibidos a seguir não serão demonstrados explicitamente. Uma análise detalhada deste material requeriria a descrição de conceitos sofisticados de álgebra fora do escopo principal desta tese. As referências citadas junto ao texto devem ser consultadas em caso de interesse por um tópico específico[3].

O Teorema do Fator Invariante

Se $\mathbf{G}(D)$ é uma matriz $k \times n$ sobre um anel Euclidiano $F[D]$, então $\mathbf{G}(D)$ pode ser decomposta na forma normal de Smith[4]

[3]Para uma introdução de conceitos básicos de álgebra, veja, por exemplo, [19].

[4]O teorema do fator invariante foi desenvolvido pelo matemático inglês H. J. S. Smith em meados do século XIX.

$$\mathbf{G}(D) = \mathbf{A}(D)\Gamma(D)\mathbf{B}(D), \tag{3.41}$$

onde $\mathbf{A}(D)$ é uma matriz $k \times k$ sobre $F[D]$ com uma inversa $\mathbf{A}^{-1}(D)$ com elementos em $F[D]$; $\mathbf{B}(D)$ é uma matriz $n \times n$ sobre $F[D]$ também com uma inversa $\mathbf{B}^{-1}(D)$ com elementos em $F[D]$; e $\Gamma(D)$ é uma matriz $k \times n$ sobre $F[D]$ da forma:

$$\Gamma(D) = [\Gamma_1(D), \mathbf{0}], \tag{3.42}$$

onde $\mathbf{0}$ é uma matriz $k \times (n-k)$ de zeros e $\Gamma_1(D)$ é uma matriz diagonal

$$\Gamma_1(D) = \text{diag}\,[\gamma_1(D), \gamma_2(D), \ldots, \gamma_k(D)]. \tag{3.43}$$

Os elementos diagonais $\gamma_i(D)$ $(1 \leq i \leq k)$ da equação (3.43) são elementos de $F[D]$ e são chamados de *fatores invariantes* de $\mathbf{G}(D)$. Os fatores invariantes são únicos e podem ser calculados da seguinte forma: Seja Δ_i o máximo divisor comum (mdc) de todos os subdeterminantes $i \times i$ de $\mathbf{G}$. Considere $\Delta_0 = 1$. Então $\gamma_i(D) = \Delta_i/\Delta_{i-1}$. Se $\gamma_{i+1}(D) \neq 0$, então $\gamma_i(D)$ divide $\gamma_{i+1}(D)$ para $i = 1, 2, \ldots, k-1$.

Forney esboçou a prova deste teorema [34]. Uma demonstração completa e formal deste teorema pode ser encontrada em alguns textos clássicos de álgebra moderna[5].

Considere que a matriz geradora de um CCC tenha a forma normal de Smith da equação (3.41). Então a saída do codificador pode ser expressa como

$$\mathbf{v}(D) = \mathbf{u}(D)\mathbf{G}(D) = \mathbf{u}(D)\mathbf{A}(D)\Gamma(D)B(D), \tag{3.44}$$

onde $\mathbf{A}(D)$ e $\mathbf{B}(D)$ têm as inversas $\mathbf{A}^{-1}(D)$ e $\mathbf{B}^{-1}(D)$, respectivamente, e $\Gamma(D)$ é a matriz $k \times n$ definida pela equação (3.42).

Para a operação de codificação da equação (3.44) ser útil no contexto de um sistema de comunicação, é desejável que o mapeamento de $\mathbf{u}(D)$ em $\mathbf{v}(D)$ sobre $F[D]$ seja um mapeamento um-a-um e reversível. E para que isto seja verdade, deve existir uma matriz inversa à direita $\mathbf{G}^{-1}(D)$ da matriz $\mathbf{G}(D)$. Se $\mathbf{G}^{-1}(D)$ existir, então

$$\mathbf{v}(D)\mathbf{G}^{-1}(D) = \mathbf{u}(D)\mathbf{G}(D)\mathbf{G}^{-1}(D) = \mathbf{u}(D)\mathbf{I}_k = \mathbf{u}(D), \tag{3.45}$$

[5]Veja, por exemplo: Seção 10, Capítulo III da Referência [1]

onde $\mathbf{I}_k$ é a matriz identidade $k \times k$ e $\mathbf{u}(D)$ é unicamente recuperável a partir da mensagem codificada $\mathbf{v}(D)$.

Massey e Sain provaram que uma inversa $\mathbf{G}^{-1}(D)$ deve ser livre de realimentação (*feedback free*)[6] para evitar que CCCs dêem origem à propagação de erros catastróficos [45]. Um CCC com uma inversa à direita e livre de realimentação é chamado de um codificador *básico* $\mathbf{G}(D)$. Forney demonstrou em detalhes que um codificador é básico se, e somente se, é um polinômio sobre $F[D]$ e tem todos os seus fatores invariantes iguais a 1 [34]. Daqui em diante trataremos somente de codificadores básicos para um CCC. Para estes, a forma normal de Smith da matriz geradora tem a forma

$$\mathbf{G}(D) = \mathbf{A}(D)[\mathbf{I}_k, 0]\mathbf{B}(D), \tag{3.46}$$

onde $\mathbf{I}_k$ é a matriz identidade $k \times k$. A inversa de $\mathbf{G}(D)$ tem a forma normal de Smith

$$\mathbf{G}^{-1}(D) = \mathbf{B}^{-1}(D)[\mathbf{I}_k, 0]^T\mathbf{A}^{-1}(D). \tag{3.47}$$

Forney [34, 35] deduziu uma matriz verificação de paridade $\mathbf{H}(D)$ sobre $F[D]$ para uma matriz geradora $\mathbf{G}(D)$ da forma da equação (3.46). Para verificar isto, considere $\mathbf{B}^{-1}(D)$ a inversa da matriz $\mathbf{B}(D)$ de dimensões $n \times n$ na forma normal de Smith da equação (3.46) e seja $\mathbf{B}_1(D)$ a matriz composta das últimas $(n-k)$ colunas de $\mathbf{B}^{-1}(D)$. Então,

$$\mathbf{H}(D) = \mathbf{B}_1^T(D). \tag{3.48}$$

Forney mostrou que a matriz $\mathbf{B}_1(D)$ é a matriz verificação de paridade de $\mathbf{G}(D)$ da equação (3.46). Isto é, $\mathbf{G}(D)\mathbf{H}^T(D) = 0$.

Como as $(n-k)$ linhas de $\mathbf{H}(D)$ são as $(n-k)$ colunas da matriz inversível $\mathbf{B}^{-1}(D)$, estas linhas devem ser linearmente independentes. Portanto, $\mathbf{H}(D)$, da forma definida na equação (3.48), tem posto $(n-k)$ e é uma matriz verificação de paridade válida para a matriz geradora $\mathbf{G}(D)$ dada pela equação (3.46).

A seguir, usaremos a matriz verificação de paridade para obter a síndrome do vetor recebido à saída do canal. Um algoritmo de decodificação será então apresentado através de um exemplo.

[6] Ou seja, as saídas do codificador não podem servir como entradas para o próprio codificador.

Soluções Gerais da Equação de Síndromes para CCCs (n, k, m)

Vimos que a equação geral de síndromes é

$$\mathbf{s}(D) = \mathbf{r}(D)\mathbf{H}^T(D), \tag{3.49}$$

ou, em termos da sequência de erro,

$$\mathbf{s}(D) = \mathbf{e}(D)\mathbf{H}^T(D). \tag{3.50}$$

Para encontrar a solução geral desta equação faremos uso do teorema do fator invariante. Este teorema será aplicado à matriz $\mathbf{H}^T(D)$. Pelo teorema do fator invariante e um lema devido a Forney ([34], Apêndice I), $\mathbf{H}^T(D)$ tem a forma normal de Smith para um codificador básico

$$\mathbf{H}^T = \mathbf{L}(D)[\mathbf{I}_{n-k}, \mathbf{0}]^T\mathbf{Z}(D), \tag{3.51}$$

onde $\mathbf{L}(D)$ e $\mathbf{Z}(D)$ são matrizes inversíveis $n \times n$ e $(n-k)\times(n-k)$, respectivamente, sobre $F(D)$, $\mathbf{I}_{n-k}$ é a matriz identidade $(n-k)\times(n-k)$ e $\mathbf{0}$ denota uma matriz $(n-k)\times k$ de zeros.

Para encontrarmos a solução para $\mathbf{e}(D)$, primeiro substituímos a equação (3.51) na equação (3.50). Isto produz

$$\mathbf{s}(D) = \mathbf{e}(D)\mathbf{L}(D)[\mathbf{I}_{n-k}, \mathbf{0}]^T\mathbf{Z}(D). \tag{3.52}$$

Após multiplicar ambos os lados da equação (3.52) por $\mathbf{Z}^{-1}(D)$, obtemos

$$\begin{aligned} \sigma(D) &= \mathbf{s}(D)\mathbf{Z}^{-1}(D) \\ &= [\mathbf{e}(D)\mathbf{L}(D)][\mathbf{I}_{n-k}, \mathbf{0}]^T \\ &= \varepsilon(D)[\mathbf{I}_{n-k}, \mathbf{0}]^T, \end{aligned} \tag{3.53}$$

onde

$$\sigma(D) = \mathbf{s}(D)\mathbf{Z}^{-1}(D) = [\sigma^{(1)}(D), \ldots, \sigma^{(n-k)}(D)] \tag{3.54}$$

é um vetor transformado de $\mathbf{s}(D)$ com $(n-k)$ componentes e

$$\boldsymbol{\varepsilon}(D) = \mathbf{e}(D)\mathbf{L}(D) = [\varepsilon^{(1)}(D), \ldots, \varepsilon^{(n)}(D)] \tag{3.55}$$

é um vetor transformado do vetor erro polinomial desconhecido $\mathbf{e}(D)$ com n componentes.

A solução componente-a-componente da equação (3.53) é obtida igualando-se as componentes das equações

$$\varepsilon^{(j)}(D) = \sigma^{(j)}(D), \;\; 1 \leq j \leq n-k \tag{3.56}$$

$$\varepsilon^{(j)}(D) = \mathbf{t}^{(j-n+k)}(D), \;\; n-k+1 \leq j \leq n, \tag{3.57}$$

onde $\mathbf{t}^{(i)}(D)$ para $1 \leq i \leq k$ é um polinômio arbitrário no anel Euclideano $F[D]$.

Substituindo as equações (3.56) e (3.57) no lado direito da equação (3.55) e resolvendo para $\mathbf{e}(D)$ temos finalmente

$$\begin{aligned} \mathbf{e}(D) &= [\mathbf{e}^{(1)}(D), \ldots, \mathbf{e}^{(n)}(D)] \\ &= [\sigma^{(1)}(D), \ldots, \sigma^{(n-k)}(D), \mathbf{t}^{(1)}(D), \ldots, \mathbf{t}^{(k)}(D)]\mathbf{L}^{-1}(D) \end{aligned} \tag{3.58}$$

como a solução geral da equação (3.50) em termos das $n-k$ componentes da síndrome transformada $\boldsymbol{\sigma}(D)$ da equação (3.54) e k parâmetros polinomiais arbitrários $\mathbf{t}^{(j)}(D)$ de $F[D]$ para $1 \leq j \leq k$.

Exemplo 8. *Considere a matriz geradora*

$$\mathbf{G}(D) = [1+D^2,\ 1+D+D^2,\ 1+D+D^2] \tag{3.59}$$

de um CCC (3, 1, 2). Em notação de matriz discreta semi-infinita, $\mathbf{G}(D)$ *é escrita como:*

$$\mathbf{G}(D) = \begin{bmatrix} 111 & 011 & 111 & & & \\ & 111 & 011 & 111 & & \\ & & 111 & 011 & 111 & \\ & & & \ddots & \ddots & \ddots \end{bmatrix}. \tag{3.60}$$

As Figuras 3.11 e 3.12 mostram, respectivamente, o codificador e o diagrama de estados com esta $\mathbf{G}(D)$*. É fácil de verificar que este codificador tem* $d_{free} = 8$ *e, portanto, é capaz de corrigir até três erros.*

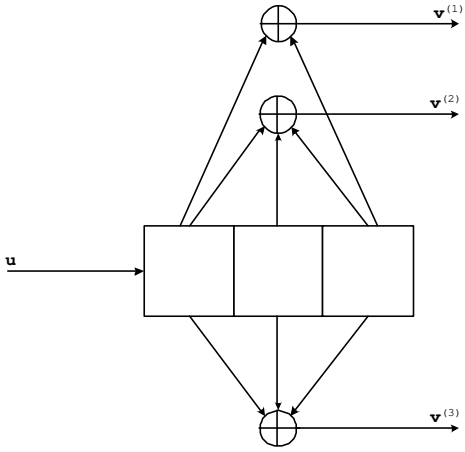

Figura 3.11: Codificador para o CCC (3, 1, 2).

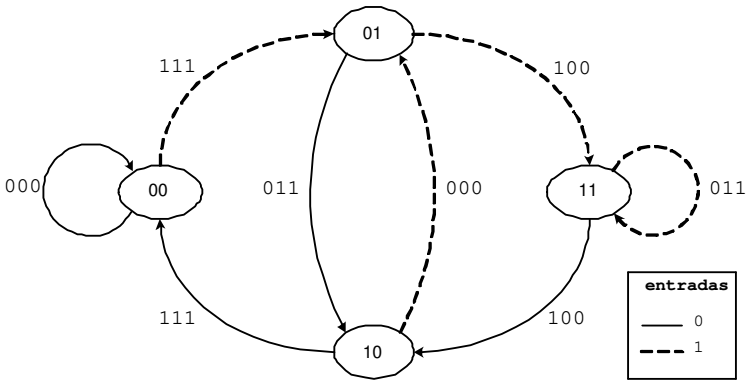

Figura 3.12: Diagrama de estados para o CCC (3, 1, 2).

Usando o método de Forney desenvolvido acima, uma matriz verificação de paridade para esta $\mathbf{G}(D)$ *é*

$$\mathbf{H}(D) = \begin{bmatrix} 1+D+D^2 & 1+D^2 & 0 \\ 1+D+D^2 & D^2 & 1 \end{bmatrix}, \tag{3.61}$$

Ou, mais explicitamente, na forma de matriz discreta semi-infinita:

$$\mathbf{H} = \begin{bmatrix} 110 & & & & \\ 101 & & & & \\ 100 & 110 & & & \\ 100 & 101 & & & \\ 110 & 100 & 110 & & \\ 110 & 100 & 101 & & \\ & 110 & 100 & 110 & \\ & 110 & 100 & 101 & \\ & & \ddots & \ddots & \ddots \\ & & \ddots & \ddots & \ddots \end{bmatrix}. \tag{3.62}$$

A Figura 3.13 mostra o decodificador de síndromes para esta $\mathbf{H}(D)$.

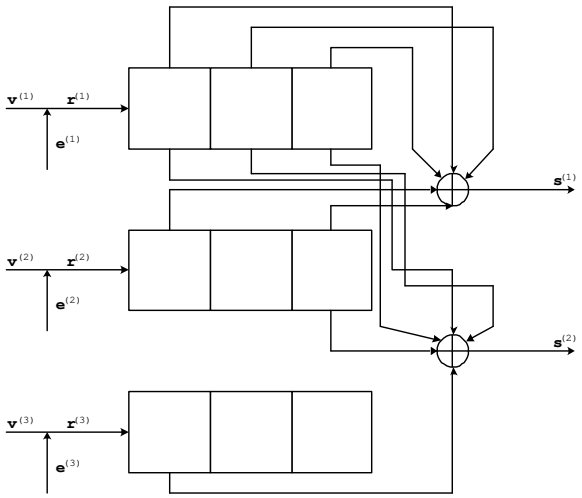

Figura 3.13: Decodificador de síndromes para o CCC (3, 1, 2).

A forma normal de Smith para $\mathbf{H}^T(D)$ *é dada por*

$$\mathbf{H}^T(D) = \mathbf{L}(D) \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}^T, \tag{3.63}$$

onde a inversa de $\mathbf{L}(D)$ *é*

$$\mathbf{L}^{-1}(D) = \begin{bmatrix} D & 1+D & D \\ 0 & 0 & 1 \\ 1+D^2 & 1+D+D^2 & 1+D+D^2 \end{bmatrix}. \tag{3.64}$$

Substituindo a equação (3.63) na equação (3.50), ficamos com a seguinte equação de síndrome:

$$\mathbf{s}(D) = [\mathbf{s}^{(1)}(D), \mathbf{s}^{(2)}(D)] = [\boldsymbol{\varepsilon}^{(1)}(D), \boldsymbol{\varepsilon}^{(2)}(D), \boldsymbol{\varepsilon}^{(3)}(D)] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}^T, \tag{3.65}$$

onde $\boldsymbol{\varepsilon}^{(i)}(D) = \mathbf{e}^{(i)}(D)\mathbf{L}(D)$, *para* $1 \leq i \leq 3$.

Usando as equações (3.56) e (3.57), a solução da equação (3.65) para $\boldsymbol{\varepsilon}^{(j)}(D)$ *é*

$$\boldsymbol{\varepsilon}^{(1)}(D) = \mathbf{s}^{(1)}(D),\ \boldsymbol{\varepsilon}^{(2)}(D) = \mathbf{s}^{(2)}(D),\ \boldsymbol{\varepsilon}^{(3)}(D) = \mathbf{t}(D), \tag{3.66}$$

onde $\mathbf{t}(D)$ *é um polinômio arbitrário de* $F[D]$. *Finalmente os vetores de erro* $\mathbf{e}^{(i)}(D)$ *em termos das soluções dadas na equação (3.66) são, usando a equação (3.64),*

$$[\mathbf{e}^{(1)}(D), \mathbf{e}^{(2)}(D), \mathbf{e}^{(3)}(D)] = [\mathbf{s}^{(1)}(D), \mathbf{s}^{(2)}(D), \mathbf{t}(D)]\mathbf{L}^{-1}(D). \tag{3.67}$$

Isto produz

$$\begin{aligned} \mathbf{e}^{(1)}(D) &= D\mathbf{s}^{(1)}(D) + \mathbf{t}(D) + D^2\mathbf{t}(D), \\ \mathbf{e}^{(2)}(D) &= \mathbf{s}^{(1)}(D) + D\mathbf{s}^{(1)}(D) + \mathbf{t}(D) + D\mathbf{t}(D) + D^2\mathbf{t}(D), \\ \mathbf{e}^{(3)}(D) &= D\mathbf{s}^{(1)}(D) + \mathbf{s}^{(2)}(D) + \mathbf{t}(D) + D\mathbf{t}(D) + D^2\mathbf{t}(D) \end{aligned} \tag{3.68}$$

como solução geral da equação de síndrome (3.49) para o codificador (3, 1, 2) da equação (3.59).

Mostraremos a seguir como usar as soluções da equação de síndromes para executar o ADS.

Implementação do ADS para um CCC (n, k, m) - Exemplo

A decodificação por síndromes de um CCC (n, k, m) tem como objetivo encontrar uma estimativa de máxima verossimilhança (EMV) $\hat{\mathbf{e}}(D)$ da sequência de erro dentro da classe lateral determinada pela equação (3.58), dentre todas as possíveis soluções da equação de síndrome (3.49). A melhor forma de entender como isto se dá na prática é através da apresentação de um exemplo.

Considere o CCC (3, 1, 2) com matriz geradora dada pela equação (3.59). A saída do codificador é:

$$\begin{aligned}\mathbf{v}(D) &= \mathbf{u}(D)\mathbf{G}(D) = [\mathbf{v}^{(1)}(D),\, \mathbf{v}^{(2)}(D),\, \mathbf{v}^{(3)}(D)] \\ &= [\mathbf{u}(D) + D^2\mathbf{u}(D),\, \mathbf{u}(D) + D\mathbf{u}(D) + D^2\mathbf{u}(D),\, \mathbf{u}(D) + D\mathbf{u}(D) + D^2\mathbf{u}(D)]. \qquad (3.69)\end{aligned}$$

Assuma que a sequência de entrada seja $\mathbf{u}(D) = [1\ 1]$ (sequências de informação mais longas inviabilizariam a reprodução de um exemplo em uma folha deste tamanho).

Então, da equação (3.69), concluímos que $\mathbf{v}^{(1)}(D) = [1\ 1\ 1\ 1]$, $\mathbf{v}^{(2)}(D) = [1\ 0\ 0\ 1]$ e $\mathbf{v}^{(3)}(D) = [1\ 0\ 0\ 1]$ são as três componentes da sequência transmitida. Considere que as correspondentes três sequências recebidas sejam:

$$\begin{aligned}\mathbf{r}^{(1)}(D) &= [1\ 1\ 1\ 0], \\ \mathbf{r}^{(2)}(D) &= [0\ 1\ 0\ 1], \qquad (3.70) \\ \mathbf{r}^{(3)}(D) &= [1\ 0\ 0\ 1].\end{aligned}$$

Agora, substitua a matriz verificação de paridade da equação (3.61) na equação (3.49) para obter

$$\begin{aligned}\mathbf{s}^{(1)}(D) &= (1+D+D^2)\mathbf{r}^{(1)}(D)+(1+D^2)\mathbf{r}^{(2)}(D),\\ \mathbf{s}^{(2)}(D) &= (1+D+D^2)\mathbf{r}^{(1)}(D)+D^2\mathbf{r}^{(2)}(D)+\mathbf{r}^{(3)}(D),\end{aligned} \tag{3.71}$$

como as síndrome do código. Um cálculo das síndromes da equação (3.71) em termos da sequência recebida $[\mathbf{r}^{(1)}(D), \mathbf{r}^{(2)}(D), \mathbf{r}^{(3)}(D)]$ da equação (3.70) produz as sequências de síndromes

$$\begin{aligned}\mathbf{s}^{(1)}(D) &= [1\ 1\ 1\ 0\ 1\ 1],\\ \mathbf{s}^{(2)}(D) &= [0\ 0\ 1\ 0\ 1\ 1],\end{aligned} \tag{3.72}$$

para este exemplo. Dadas as sequências de síndromes da equação (3.72), é necessário agora resolver a equação de síndrome (3.50) da sequência de erro $\mathbf{e}(D)$.

As soluções gerais explícitas da equação de síndrome (3.50) para as componentes de $\mathbf{e}(D)$ foram encontradas na equação (3.68). O problema agora é encontrar, a partir da equação (3.68) e das sequências de síndromes dadas pela equações (3.72), a EMV $\hat{\mathbf{e}}(D)$ de $\mathbf{e}(D) = [\mathbf{e}^{(1)}(D), \mathbf{e}^{(2)}(D), \mathbf{e}^{(3)}(D)]$.

Da mesma forma que foi feito para um CCC (2, 1, m), $\hat{\mathbf{e}}(D)$ é encontrado iterativamente com a ajuda de um diagrama de treliça. Neste exemplo, os estados do diagrama de treliça são equivalentes aos estados do registro de deslocamento necessários na equação (3.68) para armazenar sequencialmente as versões atrasadas $D\mathbf{t}(D)$ e $D^2\mathbf{t}(D)$ da função arbitrária $\mathbf{t}(D)$ em F. A tabela de estados do registro de deslocamento para $\mathbf{t}(D)$ é a *mesma* tabela apresentada na subseção 3.6.1 (Tabela 3.1) para um CCC (2, 1, m), já que a ordem de memória do codificador é a mesma, $m = 2$.

A Figura 3.14 apresenta o diagrama de treliça para o nosso exemplo.

O procedimento deste ADS é semelhante ao que foi desenvolvido para o caso de um CCC (2, 1, m). Os dígitos das sequências de síndromes $\mathbf{s}^{(1)}(D)$ e $\mathbf{s}^{(2)}(D)$ computados na equação (3.72) estão colocados sobre as correspondentes seções da treliça. Os vetores

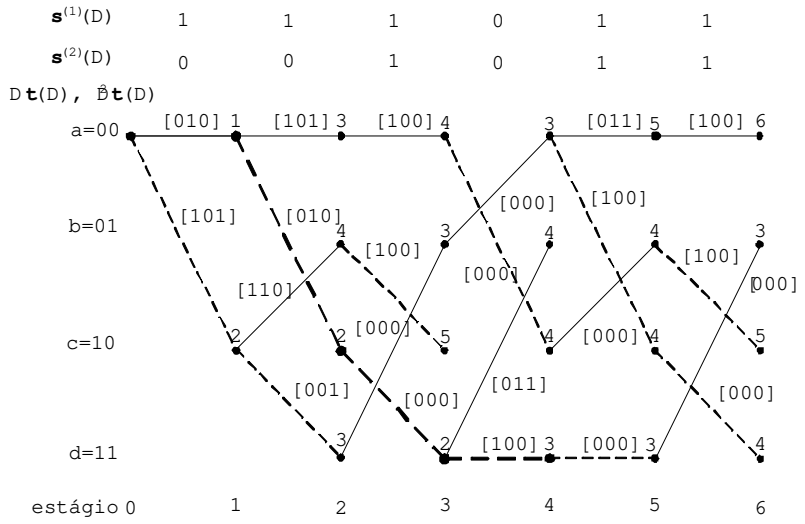

Figura 3.14: ADS para o CCC (3, 1, 2) com $\mathbf{s}^{(1)}(D) = (1\ 1\ 1\ 0\ 1\ 1)$ e $\mathbf{s}^{(2)}(D) = (0\ 0\ 1\ 0\ 1\ 1)$.

$[\mathbf{e}^{(1)}(D), \mathbf{e}^{(2)}(D), \mathbf{e}^{(3)}(D)]$ são computados em cada ramo da treliça a partir da equação (3.68), usando as síndromes $\mathbf{s}^{(1)}(D)$ e $\mathbf{s}^{(2)}(D)$ e os valores de $\mathbf{t}(D)$, $D\mathbf{t}(D)$ e $D^2\mathbf{t}(D)$, para o ramo considerado. Além disso, cada nó é classificado com o peso de Hamming acumulado pelo caminho que chega a esse nó (ou seja, com a métrica acumulada). No estágio 6, o caminho de menor peso no diagrama de treliça é $a \to a \to c \to d \to d \to d \to b$. Os ramos deste caminho produzem $\hat{\mathbf{e}}(D) = [\hat{\mathbf{e}}^{(1)}(D), \hat{\mathbf{e}}^{(2)}(D), \hat{\mathbf{e}}^{(3)}(D)] = [0001, 1100, 0000] = [D^3, 1+D, 0]$ como a estimativa do vetor erro $\mathbf{e}(D)$. Subtraindo estas estimativas de erro das componentes de $\mathbf{r}$ na equação (3.70), temos

$$\hat{\mathbf{v}}^{(1)}(D) = [1\ 1\ 1\ 1],$$

$$\hat{\mathbf{v}}^{(2)}(D) = [1\ 0\ 0\ 1], \tag{3.73}$$

$$\hat{\mathbf{v}}^{(3)}(D) = [1\ 0\ 0\ 1].$$

A forma normal de Smith do CCC (3, 1, 2) com $\mathbf{G}(D)$ dado pela equação (3.59) é $\mathbf{G}(D) = [1\ 0\ 0]\mathbf{B}(D)$, onde

$$\mathbf{B}^{-1}(D) = \begin{bmatrix} 1+D & 1+D+D^2 & 1+D+D^2 \\ D & 1+D^2 & D^2 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.74}$$

Portanto, a estimativa $\hat{\mathbf{u}}(D)$ da mensagem em termos da estimativa da palavra transmitida $\hat{\mathbf{v}}(D)$ é $\hat{\mathbf{v}}(D) = \hat{\mathbf{u}}(D)\mathbf{G}(D) = \hat{\mathbf{u}}(D)[1\,0\,0]\mathbf{B}(D)$. Resolvendo esta relação para $\hat{\mathbf{u}}(D)$:

$$\begin{aligned} \hat{\mathbf{u}}(D) &= \hat{\mathbf{v}}(D)\mathbf{G}^{-1}(D) = \hat{\mathbf{v}}(D)\mathbf{B}^{-1}(D)[1,\,0,\,0]^T \\ &= [\hat{\mathbf{v}}^{(1)}(D),\,\hat{\mathbf{v}}^{(2)}(D),\,\hat{\mathbf{v}}^{(3)}(D)] \\ &\times \begin{bmatrix} 1+D & 1+D+D^2 & 1+D+D^2 \\ D & 1+D^2 & 1+D^2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \\ &= (1+D)\hat{\mathbf{v}}^{(1)}(D) + D\hat{\mathbf{v}}^{(2)}(D). \end{aligned} \tag{3.75}$$

Uma substituição das estimativas de $\hat{\mathbf{v}}^{(1)}(D)$ e $\hat{\mathbf{v}}^{(2)}(D)$ dadas em (3.73) na equação (3.75) produz finalmente $\hat{\mathbf{u}}(D) = (1\ 1)$ como a estimativa (correta) da mensagem original.

No exemplo acima, o ADS produziu a mensagem original. No entanto, se o número de erros exceder o valor permitido pelo d_{free}, poderão existir dois ou mais caminhos com o mesmo peso mínimo. Neste caso, o ADS falha.

Finalmente, estamos agora em condições de apresentar alguns procedimentos para a geração e a decodificação de CCQs.

Parte II

Códigos Convolucionais Quânticos (CCQs)

Capítulo 4

Construção de um CCQ

We have learned that it is possible to fight entanglement with entanglement.

– John Preskill.

Neste capítulo construiremos um CCQ concatenado de taxa $R = k/n = 1/4$ e memória $m = 3$, denotado por $[n, k, m]$=[4, 1, 3], capaz de corrigir um erro quântico geral. Faremos uso do formalismo estabilizador para a descrição do subespaço do código quântico e adaptaremos o algoritmo clássico de decodificação por síndromes (ADS) para o contexto quântico, preservando a otimalidade da estimativa de erros segundo o critério de máxima verossimilhança.

Em 1998 e 1999, Chau [7, 8] propôs a construção de CCQs a partir de CBQs e de CCCs de forma a permitir a codificação de sequências de registros quânticos com um desempenho melhor e uma complexidade menor do que o oferecido pelos CBQs conhecidos na época. Sua análise, no entanto, não apresentou um procedimento sistemático claro de codificação e de decodificação. Já em 2003 e 2004, Ollivier e Tillich [50, 49] apresentaram os fundamentos de uma teoria para CCQs e deram um exemplo de CCQ com taxa 1/5. No entanto, estes trabalhos também não apresentaram métodos sistemáticos de construção e outros exemplos concretos de CCQs.

Em termos de complexidade, os CCQs têm pelo menos duas grandes vantagens em relação aos CBQs: a complexidade de codificação cresce exponencialmente, porém com um expoente menor, e a complexidade do algoritmo de decodificação cresce linearmente,

porém com uma inclinação menor, com o número de qubits de informação para todos os canais sem memória. Além disso, bons CCQs costumam ter capacidade de correção maior do que os melhores CBQs para uma mesma taxa de transmissão.

O CCQ [4 ,1, 3] que proporemos será construído a partir da concatenação de dois CCCs, a saber, um (2, 1, 2) e um (4, 2, 1), gerados a partir de uma *mesma* matriz geradora. A técnica de construção é semelhante à utilizada por Shor para a construção de seu código de bloco de nove qubits. Mais precisamente, construiremos dois CCQs específicos $\mathcal{C}_1$ e $\mathcal{C}_2$, capazes de corrigir um erro de fase Z e de bit X, respectivamente, para um mesmo conjunto de registros quânticos. A seguir, faremos a concatenação destes dois códigos: primeiro usaremos $\mathcal{C}_1$ para codificar o estado quântico de informação e então usaremos $\mathcal{C}_2$ para codificar o estado quântico resultante da codificação por $\mathcal{C}_1$. Veremos que o código $\mathcal{C}$ resultante desta concatenação será capaz de corrigir um erro quântico geral com geradores Z e X para o mesmo conjunto de registros quânticos protegido por $\mathcal{C}_1$ e $\mathcal{C}_2$. Até o ano de 2003, este era o código quântico com a taxa mais alta capaz de corrigir um erro quântico geral. Posteriormente, como será visto no Capítulo 5, conseguimos construir CCQs com taxas ainda mais altas e com maior capacidade de correção.

Antes de iniciarmos a construção de um CCQ, introduziremos algumas observações referentes à notação que será utilizada para o estudo de CCQs.

Por se tratar de uma introdução aos CCQs, não faremos uso da notação polinomial para as operações de codificação, do formalismo estabilizador e da decodificação. A notação polinomial será citada apenas quando fizermos uso de CCCs. Apesar de mais concisa e elegante, a notação polinomial no domínio quântico é demasiadamente complexa para uma primeira análise de CCQs. A notação de matriz discreta semi-infinita é a mais simples para os CCQs que iremos estudar ao longo desta tese. Como vimos no Capítulo 3, ambas as notações são equivalentes e o uso de uma não traz qualquer perda de informação em relação à outra.

A sequência de informação quântica que desejamos proteger é, a princípio, composta por infinitos registros quânticos. Suponha que cada registro quântico seja um qubit e que, portanto, tenha dois estados-base ortogonais, $|0\rangle$ e $|1\rangle$. Então, a base de um estado quântico geral constituído de (possivelmente) infinitos registros quânticos pode ser escolhida

como $\{|\mathbf{u}\rangle\} \equiv \{|u_0, u_1, \ldots, u_m, \ldots\rangle\}$, para $u_m \in \{0,1\}$. Além disso, definimos $u_m = 0$ para $m < 0$. Todas as adições dentro dos kets serão módulo-2.

4.1 Um CCQ para o Canal Bit Flip

Considere que façamos uso do CCC (2, 1, 2) com matriz geradora polinomial $\mathbf{G}(D) = [1+D^2,\ 1+D+D^2]$ na operação de codificação dos bits contidos dentro de cada um dos kets da base de uma sequência de registros quânticos que deve ser protegida da ação dos erros de um canal bit flip. Desta operação de codificação clássica, podemos obter a correspondente operação de codificação quântica:

$$\bigotimes_{t=0}^{+\infty} |u_t\rangle \longmapsto \bigotimes_{t=0}^{+\infty} |u_t + u_{t-2},\ u_t + u_{t-1} + u_{t-2}\rangle, \tag{4.1}$$

onde $u_{-1} = u_{-2} = 0$.

Na prática, estamos sempre interessados em codificar uma sequência *finita* de qubits. Isto implica que ao final de cada sequência de bits dentro dos kets da base sejam acrescentados mais dois bits 0s para serem codificados. Esta condição garantirá que tenhamos sempre palavras-código clássicas válidas dentro dos kets e, consequentemente, garantirá que tenhamos sempre uma palavra-código quântica válida.

Este CCQ herda todas as propriedades do CCC associado. Ou seja, é um CCQ [2, 1, 2] não-catastrófico[1], capaz de garantir a correção de até dois erros bit flip X em quaisquer dois qubits da palavra-código[2]. Por causa da natureza convolucional da operação de codificação, algumas palavras-código quânticas com mais de dois erros X também podem ser corrigidas.

Não discutiremos ao longo desta tese, para nenhum dos canais, o *codificador quântico* de um CCQ, ou seja, a "máquina quântica" que realiza as operações de codificação convolucional quântica. A construção de um circuito sequencial linear (CSL) quântico está fora do escopo desta tese.

[1] Ou seja, o CCC associado não gera palavras-código de peso finito para *nenhuma* sequência de bits de informação de peso infinito.

[2] A capacidade de correção deste CCQ será limitada pelo d_{free} (cinco) do CCC associado, pois este não gera *nenhuma* palavra-código válida não-nula com peso menor do que o d_{free}, *independente* de quantos bits dentro dos kets da base do estado quântico de informação codifiquemos.

4.1.1 Formalismo Estabilizador

Vimos no Capítulo 3 que a matriz discreta semi-infinita associada à matriz polinomial $\mathbf{G}(D) = [1+D^2,\ 1+D+D^2]$ tem a seguinte forma:

$$\mathbf{G} = \begin{bmatrix} 11 & 01 & 11 & & & & & \\ & 11 & 01 & 11 & & & & \\ & & 11 & 01 & 11 & & & \\ & & & 11 & 01 & 11 & & \\ & & & & 11 & 01 & 11 & \\ & & & & & \ddots & \ddots & \ddots \end{bmatrix}. \tag{4.2}$$

Vimos também que a matriz verificação de paridade associada a $\mathbf{G}(D) = [\mathbf{g}^{(1)},\ \mathbf{g}^{(2)}] = [1+D^2,\ 1+D+D^2]$ é $\mathbf{H}(D) = [\mathbf{g}^{(2)},\ \mathbf{g}^{(1)}] = [1+D+D^2,\ 1+D^2]$, que na notação de matriz discreta semi-infinita tem a seguinte forma:

$$\mathbf{H} = \begin{bmatrix} 11 & & & & & \\ 10 & 11 & & & & \\ 11 & 10 & 11 & & & \\ & 11 & 10 & 11 & & \\ & & 11 & 10 & 11 & \\ & & & \ddots & \ddots & \ddots \end{bmatrix}. \tag{4.3}$$

As linhas desta matriz permitem-nos escrever os geradores do grupo estabilizador S_X do CCQ [2, 1, 2] descrito pela operação de codificação (4.1). Estes geradores são mostrados na Tabela 4.1.

Todos os geradores da Tabela 4.1 comutam e são independentes entre si. Portanto, o subespaço do código (isto é, o maior subespaço comum dos geradores com autovalor $+1$) é não-trivial. É fácil de verificar que qualquer M_t do conjunto de geradores da Tabela 4.1 satisfaz a relação $M_t|\psi\rangle = |\psi\rangle$, onde $|\psi\rangle$ é a palavra-código gerada pela operação de codificação (4.1).

Como, a princípio, o CCQ será usado para codificar uma sequência infinita de qubits de informação, o número de geradores do grupo estabilizador será também infinito. Na

M_0	Z	Z	I	I	I	I	I	I	I	I	I	I	$\cdots$	$\cdots$
M_1	Z	I	Z	Z	I	I	I	I	I	I	I	I	$\cdots$	$\cdots$
M_2	Z	Z	Z	I	Z	Z	I	I	I	I	I	I	$\cdots$	$\cdots$
M_3	I	I	Z	Z	Z	I	Z	Z	I	I	I	I	$\cdots$	$\cdots$
M_4	I	I	I	I	Z	Z	Z	I	Z	Z	I	I	$\cdots$	$\cdots$
$\vdots$						$\ddots$		$\ddots$		$\ddots$				
M_∞	I	I	I	I	I	I	I	I	I	I	I	I	Z	Z.

Tabela 4.1: Geradores do CCQ [2, 1, 2] descrito pela operação de codificação (4.1).

prática, estamos sempre interessados em codificar um número finito de qubits de informação. Isto transforma o CCQ em um CBQ gerado através de uma operação de convolução. Portanto, na prática, devemos somente considerar um número finito de geradores.

Para uma sequência de informação com N qubits, precisamos considerar somente $2(N+2)-N = N+4$ geradores para descrever o subespaço do código (o número de geradores deste código será sempre a diferença entre o comprimento do código e o número de qubits de informação). Isto ficará mais claro com os exemplos que serão apresentados para $N = 1$ e $N = 2$.

Outro importante ponto a considerar quando estudamos códigos estabilizadores é a habilidade de manipular informação codificada através de operações lógicas. Queremos encontrar o operador de Pauli lógico $\overline{X}$ correspondente a cada um dos qubits de informação (os qubits lógicos). Vimos no Capítulo 2 que estes operadores devem satisfazer as seguintes relações:

$$\begin{cases} \overline{X}_i \in N(S_X)/S_X \\ \\ \forall\ i \neq j,\ [\overline{X}_i, \overline{X}_j] = 0, \end{cases} \tag{4.4}$$

onde $N(S_X)$ é o normalizador de S_X. A primeira destas relações impõe que os operadores de Pauli deixem o subespaço do código globalmente invariante, mas tenham uma ação não trivial sobre seus elementos, enquanto que a segunda condição assegura que a manipulação do qubit i não afeta os outros qubits.

As linhas da matriz geradora (4.2) podem ser usadas para determinar as operações lógicas $\overline{X}_i$, já que estas devem ser independentes dos geradores do estabilizador e comutar com cada um deles. Estes operadores estão dispostos na Tabela 4.2.

$\overline{X}_1$	X	X	I	X	X	X	I	I	I	I	I	I	$\cdots$	$\cdots$
$\overline{X}_2$	I	I	X	X	I	X	X	X	I	I	I	I	$\cdots$	$\cdots$
$\overline{X}_3$	I	I	I	I	X	X	I	X	X	X	I	I	$\cdots$	$\cdots$
$\vdots$							$\ddots$		$\ddots$		$\ddots$			
$\overline{X}_\infty$	I	I	I	I	I	I	I	I	X	X	I	X	X	X.

Tabela 4.2: Operações lógicas do CCQ [2, 1, 2] descrito pela operação de codificação (4.1).

Como o CCC (2, 1, 2) para uma sequência de informação de comprimento N é uma sequência de comprimento $2(N+2)$, podemos afirmar para o correspondente CCQ [2, 1, 2] que uma transformação unitária sobre N qubits *lógicos* será implementada através de uma transformação unitária sobre os $2(N+2)$ qubits *físicos*.

4.1.2 Identificação e Correção de Erros

A cada instante de tempo cada um dos dois qubits gerados do CCQ [2, 1, 2] acima é passado através de uma cópia idêntica de um canal bit flip sem memória. Nosso problema agora é detectar e corrigir os erros que eventualmente possam ter ocorrido na palavra-código durante a transmissão pelo canal. Uma ótima solução é adaptar o ADS clássico ao contexto quântico.

Com a medida da palavra-código recebida à saída do canal quântico pelo conjunto de observáveis da Tabela 4.1 é possível determinar qual erro eventualmente ocorreu sobre a palavra-código original. Os resultados destas medidas são um conjunto de autovalores $\{\alpha_t\}$, que podem assumir os valores $+1$ ou -1, já que são autovalores de operadores constituídos de produtos tensoriais de matrizes de Pauli. É fácil de verificar que existe, para cada instante de tempo t, um homomorfismo entre as síndromes clássicas $s_t = \{0,1\}$ e os autovalores $\alpha_t = \{+1,-1\}$ estabelecido pela relação $s_t = (1-\alpha_t)/2$ (t= 0, 1, 2, ...). Esta

relação permite que as soluções da equação de síndromes (3.40) para o CCC com matriz geradora $\mathbf{G}(D) = [1+D^2,\ 1+D+D^2]$ possam ser adaptadas para o domínio quântico como:

$$\begin{aligned} \mathbf{e}^{(1)}(D) &= D\mathbf{s}(D) + \mathbf{t}(D) + D^2\mathbf{t}(D) \\ \mathbf{e}^{(2)}(D) &= \mathbf{s}(D) + D\mathbf{s}(D) + \mathbf{t}(D) + D\mathbf{t}(D) + D^2\mathbf{t}(D), \end{aligned} \tag{4.5}$$

onde:

$$\mathbf{s}(D) = \frac{1-\alpha_0}{2} + \frac{1-\alpha_1}{2}D + \frac{1-\alpha_2}{2}D^2 + \ldots \tag{4.6}$$

Os valores de $\mathbf{t}(D)$, $D\mathbf{t}(D)$ e $D^2\mathbf{t}(D)$ ao longo das transições de tempo podem ser obtidos através da Tabela 3.1. Definimos que o estado inicial é $(D\mathbf{t}(D), D^2\mathbf{t}(D)) = (0,0)$ e que $\mathbf{s}(D) = 0$ (ou $\alpha = +1$) antes do estágio 0. O ADS então seleciona o caminho no diagrama de treliça com o menor peso de Hamming, exatamente como foi feito classicamente (veja a subseção 3.6.1).

4.1.3 Exemplos: $N = 1$ e $N = 2$

A seguir, apresentamos o CCQ descrito pela operação de codificação (4.1) para uma sequência de informação composta de um e de dois registros quânticos. Os códigos para sequências de informação mais longas podem ser obtidos de maneira similar.

Exemplo 9. *Suponha que desejamos codificar apenas um registro quântico. O estado do sistema quântico a ser codificado é, portanto, o estado de um único qubit,* $|\psi\rangle = a|0\rangle + b|1\rangle$. *A operação de codificação quântica é realizada através da codificação do bit dentro de cada um dos kets da base pelo CCC com matriz geradora dada por:*

$$\mathbf{G} = \begin{bmatrix} 11 & 01 & 11 \end{bmatrix}. \tag{4.7}$$

Assim, temos a seguinte operação de codificação quântica:

$$|0_L\rangle \rightarrow |00\,00\,00\rangle$$
$$|1_L\rangle \rightarrow |11\,01\,11\rangle \tag{4.8}$$

Considerando que o estado incial $a|0\rangle + b|1\rangle$ *tenha sido perfeitamente codificado, podemos garantir a correção de até dois erros* X *em quaisquer dos seis qubits das palavras-código.*

Suponha que desejamos detectar e corrigir erros X *que eventualmente possam ter ocorrido durante a transmissão da palavra-código (4.8). A matriz verificação de paridade correspondente à matriz geradora (4.7) é:*

$$\mathbf{H} = \begin{bmatrix} 11 & & & & & \\ 10 & 11 & & & & \\ 11 & 10 & 11 & & & \\ & 11 & 10 & & & \\ & & 11 & & & \end{bmatrix}. \tag{4.9}$$

As linhas desta matriz fornecem-nos os geradores do estabilizador do CCQ. Veja a Tabela 4.3.

M_0	Z	Z	I	I	I	I
M_1	Z	I	Z	Z	I	I
M_2	Z	Z	Z	I	Z	Z
M_3	I	I	Z	Z	Z	I
M_4	I	I	I	I	Z	Z
$\overline{X}$	X	X	I	X	X	X

Tabela 4.3: Geradores e operação lógica do CCQ [2, 1, 2] descrito pela operação de codificação (4.1) para um qubit.

É fácil de verificar que cada um dos geradores M_t *da Tabela 4.3 satisfaz a relação* $M_t|\psi\rangle = |\psi\rangle$*, onde* $|\psi\rangle$ *é palavra-código (4.8). Na Tabela 4.3 também é apresentada a operação lógica do código,* $\overline{X}$*. Note que, de fato, as seguintes relações são satisfeitas:*

$$\begin{aligned} \overline{X}|0_L\rangle &= |1_L\rangle \\ \overline{X}|1_L\rangle &= |0_L\rangle \end{aligned} \tag{4.10}$$

Usaremos os cinco observáveis da Tabela 4.3 para realizar medidas da palavra-código recebida à saída do canal. Os resultados destas medidas permitem-nos detectar sem ambiguidade qualquer grupo de até dois erros X que possa ter ocorrido sobre os seis qubits da palavra-código original. Por exemplo, suponha que o resultado das medidas dos cinco observáveis seja dado pelo conjunto de autovalores $(-1,-1,+1,-1,+1)$*. O resultado da aplicação do ADS para este conjunto de autovalores é mostrado na Figura 4.1.*

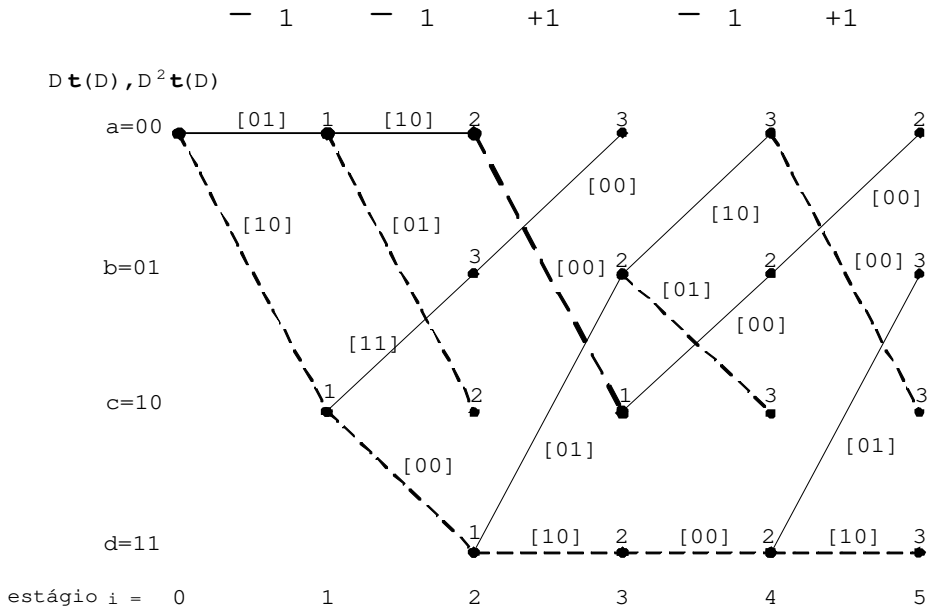

Figura 4.1: ADS para o conjunto de autovalores $(-1, -1, +1, -1, +1)$. O caminho selecionado pelo algoritmo é: $a \to a \to a \to c \to b \to a$.

Note que no estágio 5 da Figura 4.1, o caminho de menor peso é o caminho $a \to a \to a \to c \to b \to a$*. Os ramos deste caminho produzem* $\hat{\mathbf{e}} = [01\ 10\ 00]$ *como estimativa da sequência de erro. Isto significa que a palavra-código recebida à saída do canal foi*

$a|01\ 10\ 00\rangle + b|10\ 11\ 11\rangle$. *Para corrigirmos os dois erros, basta aplicarmos os operadores* X_2 *e* X_3 *sobre esta palavra-código.*

Exemplo 10. *Suponha agora que desejamos codificar dois registros quânticos. O estado do sistema quântico é, portanto, o estado de dois qubits,* $|\psi\rangle = a_1a_2|00\rangle + a_1b_2|01\rangle + b_1a_2|10\rangle + b_1b_2|11\rangle$. *A operação de codificação quântica é realizada através da codificação dos dois bits contidos dentro de cada um dos quatro kets da base pelo CCC com matriz geradora dada por:*

$$\mathbf{G} = \begin{bmatrix} 11 & 01 & 11 & \\ & 11 & 01 & 11 \end{bmatrix}. \tag{4.11}$$

Assim, temos:

$$\begin{aligned} |00_L\rangle &\rightarrow |00\ 00\ 00\ 00\rangle \\ |01_L\rangle &\rightarrow |00\ 11\ 01\ 11\rangle \\ |10_L\rangle &\rightarrow |11\ 01\ 11\ 00\rangle \\ |11_L\rangle &\rightarrow |11\ 10\ 10\ 11\rangle \end{aligned} \tag{4.12}$$

Considerando novamente que o estado inicial tenha sido perfeitamente codificado, podemos garantir a correção de até dois erros X *em quaisquer dos oito qubits da palavra-código.*

Suponha que desejamos detectar e corrigir erros X *que eventualmente possam ter ocorrido durante a transmissão da palavra-código (4.12). Sendo a matriz geradora do CCC associado ao CCQ a matriz (4.11), a correspondente matriz verificação de paridade é:*

$$\mathbf{H} = \begin{bmatrix} 11 & & & & \\ 10 & 11 & & & \\ 11 & 10 & 11 & & \\ & 11 & 10 & 11 & \\ & & 11 & 10 & \\ & & & 11 & \end{bmatrix}. \tag{4.13}$$

As linhas desta matriz fornecem-nos os geradores do estabilizador do CCQ. Veja a Tabela 4.4.

M_0	Z	Z	I	I	I	I	I	I
M_1	Z	I	Z	Z	I	I	I	I
M_2	Z	Z	Z	I	Z	Z	I	I
M_3	I	I	Z	Z	Z	I	Z	Z
M_4	I	I	I	I	Z	Z	Z	I
M_5	I	I	I	I	I	I	Z	Z
$\overline{X}_1$	X	X	I	X	X	X	I	I
$\overline{X}_2$	I	I	X	X	I	X	I	I

Tabela 4.4: Geradores e operações lógicas do CCQ [2, 1, 2] descrito pela operação de codificação (4.1) para dois qubits.

É fácil de verificar que cada um dos geradores M_t da Tabela 4.4 satisfaz a relação $M_t|\psi\rangle = |\psi\rangle$, onde $|\psi\rangle$ é palavra-código (4.12). Nesta tabela também são apresentadas as operações lógicas do código, $\overline{X}_1$ e $\overline{X}_2$. Note que, de fato, as seguintes relações são satisfeitas:

$$\begin{aligned}
\overline{X}_1|00_L\rangle = |10_L\rangle \;\; ; \;\; \overline{X}_2|00_L\rangle = |01_L\rangle \\
\overline{X}_1|01_L\rangle = |11_L\rangle \;\; ; \;\; \overline{X}_2|01_L\rangle = |00_L\rangle \\
\overline{X}_1|10_L\rangle = |00_L\rangle \;\; ; \;\; \overline{X}_2|10_L\rangle = |11_L\rangle \\
\overline{X}_1|11_L\rangle = |01_L\rangle \;\; ; \;\; \overline{X}_2|11_L\rangle = |10_L\rangle
\end{aligned} \tag{4.14}$$

Usaremos os seis observáveis da Tabela 4.4 para realizar medidas da palavra-código recebida à saída do canal. Os resultados destas medidas permitem-nos detectar sem ambiguidade qualquer grupo de até dois erros X que possa ter ocorrido sobre os oito qubits da palavra-código original. Por exemplo, suponha que o resultado das medidas dos seis observáveis seja dado pelo conjunto de autovalores $(-1,+1,-1,-1,+1,+1)$*. O resultado da aplicação do ADS para este conjunto de autovalores é mostrado na Figura 4.2.*

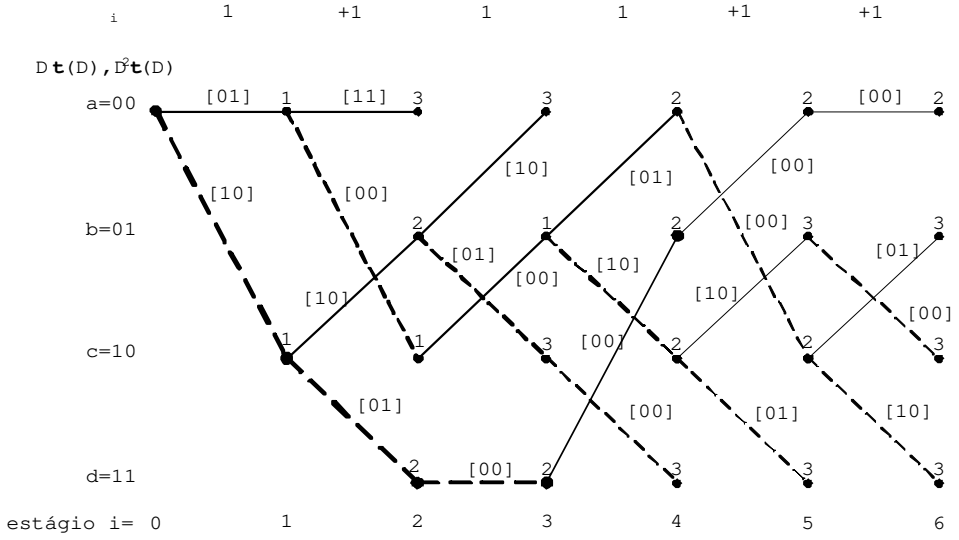

Figura 4.2: ADS para o conjunto de autovalores $(-1, +1, -1, -1, +1, +1)$. O caminho selecionado pelo algoritmo é: $a \to c \to d \to d \to b \to a \to a$.

Note que no estágio 6, o caminho de menor peso é o caminho $a \to c \to d \to d \to b \to$

$a \to a$. Os ramos deste caminho produzem $\hat{\mathbf{e}} = [10\ 01\ 00\ 00]$ como estimativa do vetor erro. Isto significa que a palavra-código recebida à saída do canal foi $a_1a_2|10\ 01\ 00\ 00\rangle + a_1b_2|10\ 10\ 01\ 11\rangle + b_1a_2|01\ 00\ 11\ 00\rangle + b_1b_2|01\ 11\ 10\ 11\rangle$. Para corrigirmos os dois erros, basta aplicarmos os operadores X_1 e X_4 sobre esta palavra-código.

Podemos continuar com este procedimento sistemático para codificar quantos qubits desejarmos. Se quisermos codificar N qubits, teremos 2^N sequências com N bits de informação a serem codificados pelo CCC (2, 1, 2). Portanto, cada uma das 2^N palavras-código da base será um estado com $2(N+2)$ qubits.

4.2 Um CCQ para o Canal Phase Flip

Vimos no Capítulo 2, durante a discussão do código de Shor, que existe um meio fácil de tratar o canal phase flip como um canal bit flip. Suponha que ao invés de trabalharmos com a base computacional $\{|0\rangle, |1\rangle\}$ para o qubit, trabalhemos com a base conjugada $\{|+\rangle \equiv (|0\rangle + |1\rangle)/\sqrt{2}, |-\rangle \equiv (|0\rangle - |1\rangle)/\sqrt{2}\}$. Com respeito a esta base, o operador Z leva o estado $|+\rangle$ ao estado $|-\rangle$ e vice-versa, isto é, este operador atua como se fosse um operador bit flip com respeito aos símbolos + e −. Assim, todas as operações necessárias − codificação, detecção e a correção de erros − podem ser executadas exatamente como no canal bit flip, mas com respeito à base $\{|+\rangle, |-\rangle\}$, ao invés da base $\{|0\rangle, |1\rangle\}$. Usando este mesmo procedimento para os CCQs, a operação de codificação (4.1) descrita para um canal bit flip pode ser escrita para um canal phase flip na base $\{|0\rangle, |1\rangle\}$ como:

$$\bigotimes_{t=0}^{+\infty} |u_t\rangle \longmapsto \bigotimes_{t=0}^{+\infty} \left\{ \frac{1}{2}(|0\rangle + (-1)^{(u_t+u_{t-2})}|1\rangle)\ (|0\rangle + (-1)^{(u_t+u_{t-1}+u_{t-2})}|1\rangle) \right\}, \tag{4.15}$$

onde $u_{-1} = u_{-2} = 0$. Ou, mais explicitamente, como[3]:

[3]Chamaremos a operação (4.15) de forma *compacta* (a palavra-código sempre pode ser escrita como um produto tensorial de *blocos* como estes) e a operação (4.16) de forma *explícita*. Ambas as formas são equivalentes, mas para um CCQ de canal quântico com erro geral, a forma compacta será a mais útil para o entendimento da operação de decodificação e a forma explícita a mais útil para o entendimento da operação de geração.

$$\bigotimes_{t=0}^{+\infty} |u_t\rangle \longmapsto \bigotimes_{t=0}^{+\infty} \left\{ \frac{1}{2} \sum_{(p_t,\, q_t)=(0,\, 0)}^{(1,\, 1)} (-1)^{(u_t+u_{t-2})p_t+(u_t+u_{t-1}+u_{t-2})q_t} |p_t,\, q_t\rangle \right\}. \tag{4.16}$$

Evidentemente, este CCQ também está associado ao CCC com matriz geradora $\mathbf{G}(D) = [1+D^2,\, 1+D+D^2]$. Portanto, é um CCQ [2, 1, 2] não-catastrófico, capaz de garantir a correção de qualquer grupo de até dois erros Z sobre a palavra-código quântica (4.16). Algumas palavras-código quânticas com mais de dois erros Z também podem ser corrigidas.

No CCQ para o canal bit flip, a natureza convolucional estava explícita nos *bits* presentes dentro dos kets do código. No CCQ para o canal phase flip, a natureza convolucional aparece de forma sutil na distribuição de *sinais* em frente aos kets do código.

4.2.1 Formalismo Estabilizador

Apresentamos acima um CCQ para o canal phase flip construído a partir da mesma matriz geradora utilizada na construção de um CCQ para o canal bit flip. Por esta razão, o CCQ para o canal phase flip também está associado à mesma matriz verificação de paridade do CCQ para o canal bit flip. Isto nos permite utilizar as linhas da matriz verificação de paridade (4.3) para escrever os geradores do grupo estabilizador S_Z do CCQ para o canal phase flip. Estes geradores estão dispostos na Tabela 4.5.

M_0	X	X	I	I	I	I	I	I	I	I	I	I	$\cdots$	$\cdots$
M_1	X	I	X	X	I	I	I	I	I	I	I	I	$\cdots$	$\cdots$
M_2	X	X	X	I	X	X	I	I	I	I	I	I	$\cdots$	$\cdots$
M_3	I	I	X	X	X	I	X	X	I	I	I	I	$\cdots$	$\cdots$
M_4	I	I	I	I	X	X	X	I	X	X	I	I	$\cdots$	$\cdots$
$\vdots$							$\ddots$		$\ddots$		$\ddots$			
M_∞	I	I	I	I	I	I	I	I	I	I	I	I	X	X.

Tabela 4.5: Geradores do CCQ [2, 1, 2] descrito pela operação de codificação (4.16).

É fácil de verificar que todos os geradores comutam e são independentes entre si. Qualquer M_t do conjunto de geradores satisfaz a relação $M_t|\psi\rangle = |\psi\rangle$, onde $|\psi\rangle$ é a palavra-código gerada pela operação de codificação (4.16). Na prática, estamos sempre interessados

em codificar um número finito de qubits de informação e neste caso o número de geradores do grupo estabilizador é um número finito. Da mesma forma que o CCQ para o canal bit flip, para uma sequência de informação com N qubits, precisamos considerar somente $2(N+2)-N=N+4$ geradores para descrever o subespaço do código.

Se desejarmos manipular o código através de operações lógicas, devemos encontrar o operador de Pauli $\overline{Z}$ correspondente a cada um dos qubits lógicos. Vimos no Capítulo 2 que estes operadores devem satisfazer as seguintes relações:

$$\begin{cases} \overline{Z}_i \in N(S_Z)/S_Z \\ \\ \forall\ i \neq j,\ [\overline{Z}_i, \overline{Z}_j] = 0, \end{cases} \tag{4.17}$$

onde $N(S_Z)$ é o normalizador de S_Z. A primeira destas condições impõe que os operadores de Pauli deixem o subespaço do código globalmente invariante, mas tenham uma ação não trivial sobre seus elementos, enquanto que a segunda condição assegura que a manipulação do qubit i não afeta os outros qubits. Novamente, podemos usar as linhas da matriz geradora (4.2) para determinar as operações lógicas $\overline{Z}_i$, já que estas devem ser independentes dos geradores do estabilizador e comutar com cada um deles. Estes operadores estão dispostos na Tabela 4.6.

$\overline{Z}_1$	Z	Z	I	Z	Z	Z	I	I	I	I	I	I	$\cdots$	$\cdots$
$\overline{Z}_2$	I	I	Z	Z	I	Z	Z	Z	I	I	I	I	$\cdots$	$\cdots$
$\overline{Z}_3$	I	I	I	I	Z	Z	I	Z	Z	Z	I	I	$\cdots$	$\cdots$
$\vdots$							$\ddots$		$\ddots$		$\ddots$			
$\overline{Z}_\infty$	I	I	I	I	I	I	I	I	Z	Z	I	Z	Z	Z.

Tabela 4.6: Operações lógicas do CCQ [2, 1, 2] descrito pela operação de codificação (4.16).

Da mesma forma que o CCQ para o canal bit flip, uma transformação unitária sobre N qubits *lógicos* será implementada através de uma transformação unitária sobre $2(N+2)$ qubits *físicos*.

4.2.2 Identificação e Correção de Erros

Como a matriz verificação de paridade do CCQ para o canal phase flip é exatamente a mesma do CCQ para o canal bit flip, as soluções (4.5) da equação de síndromes continuam válidas para, com o uso do ADS, identificar possíveis erros Z sobre a palavra-código (4.16). O procedimento de identificação de erros Z através do ADS é análogo ao que foi apresentado na seção 4.1.2 para a identificação de erros X. Isto ficará mais claro com os exemplos a seguir.

4.2.3 Exemplos: $N = 1$ e $N = 2$

A seguir, apresentamos o CCQ descrito pela operação de codificação (4.16) para uma sequência de informação composta de um e de dois registros quânticos. Os códigos para sequências de informação mais longas podem ser obtidos de maneira similar.

Exemplo 11. *Suponha que desejamos codificar apenas um registro quântico. De acordo com a operação de codificação (4.16), o CCQ pode ser convenientemente escrito na base* $\{|+\rangle, |-\rangle\}$ *como:*

$$\begin{aligned} |0_L\rangle &\rightarrow |{+}{+}\ {+}{+}\ {+}{+}\rangle \\ |1_L\rangle &\rightarrow |{-}{-}\ {+}{-}\ {-}{-}\rangle \end{aligned} \tag{4.18}$$

Ou, na base $\{|0\rangle, |1\rangle\}$,

$$\begin{aligned} |0_L\rangle \rightarrow \frac{1}{8}\Big\{ &(|0\rangle + |1\rangle)(|0\rangle + |1\rangle)(|0\rangle + |1\rangle) \\ &(|0\rangle + |1\rangle)(|0\rangle + |1\rangle)(|0\rangle + |1\rangle)\Big\} \\ |1_L\rangle \rightarrow \frac{1}{8}\Big\{ &(|0\rangle - |1\rangle)(|0\rangle - |1\rangle)(|0\rangle + |1\rangle) \\ &(|0\rangle - |1\rangle)(|0\rangle - |1\rangle)(|0\rangle - |1\rangle)\Big\} \end{aligned} \tag{4.19}$$

Ou seja, cada estado da base foi codificado em uma superposição de sessenta e quatro estados, cada um contendo seis qubits. Considerando que o estado inicial $a|0\rangle + b|1\rangle$ *tenha sido perfeitamente codificado, podemos garantir a correção de até dois erros* Z *em quaisquer dos seis qubits da palavra-código. Dito de outra forma, este código é capaz de corrigir dois erros de sinal em quaisquer dos seis blocos acima.*

Suponha que desejamos detectar e corrigir erros Z *que eventualmente possam ter ocorrido durante a transmissão da palavra-código (4.19). As linhas da matriz (4.9) fornecem-nos os geradores do grupo estabilizador. Estes geradores são apresentados na Tabela 4.7.*

M_0	X	X	I	I	I	I
M_1	X	I	X	X	I	I
M_2	X	X	X	I	X	X
M_3	I	I	X	X	X	I
M_4	I	I	I	I	X	X
$\overline{Z}$	Z	Z	I	Z	Z	Z

Tabela 4.7: Geradores e operação lógica do CCQ [2, 1, 2] descrito pela operação de codificação (4.16) para um qubit.

É fácil de verificar que cada um dos geradores M_t *da Tabela 4.7 satisfaz a relação* $M_t|\psi\rangle = |\psi\rangle$*, onde* $|\psi\rangle$ *é palavra-código (4.19). Nesta tabela também é apresentada a operação lógica do código,* $\overline{Z}$*. Note que, de fato, as seguintes relações são satisfeitas:*

$$\begin{aligned} \overline{Z}|0_L\rangle &= |0_L\rangle \\ \overline{Z}|1_L\rangle &= -|1_L\rangle \end{aligned} \tag{4.20}$$

Usaremos os cinco observáveis da Tabela 4.7 para realizar medidas da palavra-código recebida à saída do canal. Os resultados destas medidas nos permitem detectar sem ambiguidade qualquer grupo de até dois erros Z *que possa ter ocorrido nos seis qubits da palavra-código original.*

Por exemplo, suponha que o resultado da medida dos cinco observáveis seja novamente dado pelo conjunto de autovalores $(-1,-1,+1,-1,+1)$*, visto no Exemplo 9. O resultado da aplicação do ADS para este conjunto de autovalores foi mostrado na Figura 4.1. Vimos que o vetor erro detectado pelo ADS foi* $\mathbf{e} = (01\,10\,00)$*. Isto significa, para o CCQ de canal phase flip, que a palavra-código recebida foi:*

$$\begin{aligned} |0_L\rangle \rightarrow \frac{1}{8}\Big\{ & (|0\rangle + |1\rangle)(|0\rangle - |1\rangle)(|0\rangle - |1\rangle) \\ & (|0\rangle + |1\rangle)(|0\rangle + |1\rangle)(|0\rangle + |1\rangle)\Big\} \\ |1_L\rangle \rightarrow \frac{1}{8}\Big\{ & (|0\rangle - |1\rangle)(|0\rangle + |1\rangle)(|0\rangle - |1\rangle) \\ & (|0\rangle - |1\rangle)(|0\rangle - |1\rangle)(|0\rangle - |1\rangle)\Big\} \end{aligned} \tag{4.21}$$

Para corrigirmos os erros de sinal no segundo e terceiro blocos, basta aplicarmos os operadores Z_2 *e* Z_3 *sobre esta palavra-código. Como o processo de correção é análogo ao do Exemplo 9, algumas palavras-código com mais de dois erros* Z *também podem ser corrigidas.*

Exemplo 12. *Suponha agora que desejamos codificar dois registros quânticos. De acordo com a operação de codificação (4.16), o CCQ pode ser convenientemente escrito como:*

$$\begin{aligned} |00_L\rangle &\rightarrow |{+}{+}\ {+}{+}\ {+}{+}\ {+}{+}\rangle \\ |01_L\rangle &\rightarrow |{+}{+}\ {-}{-}\ {+}{-}\ {-}{-}\rangle \\ |10_L\rangle &\rightarrow |{-}{-}\ {+}{-}\ {-}{-}\ {+}{+}\rangle \\ |11_L\rangle &\rightarrow |{-}{-}\ {-}{+}\ {-}{+}\ {-}{-}\rangle \end{aligned} \tag{4.22}$$

Ou, na base $\{|0\rangle, |1\rangle\}$*,*

$$
\begin{aligned}
|00_L\rangle \rightarrow \frac{1}{16}\Big\{ & (|0\rangle+|1\rangle)(|0\rangle+|1\rangle)(|0\rangle+|1\rangle)(|0\rangle+|1\rangle) \\
& (|0\rangle+|1\rangle)(|0\rangle+|1\rangle)(|0\rangle+|1\rangle)(|0\rangle+|1\rangle)\Big\} \\
|01_L\rangle \rightarrow \frac{1}{16}\Big\{ & (|0\rangle+|1\rangle)(|0\rangle+|1\rangle)(|0\rangle-|1\rangle)(|0\rangle-|1\rangle) \\
& (|0\rangle+|1\rangle)(|0\rangle-|1\rangle)(|0\rangle-|1\rangle)(|0\rangle-|1\rangle)\Big\} \\
|10_L\rangle \rightarrow \frac{1}{16}\Big\{ & (|0\rangle-|1\rangle)(|0\rangle-|1\rangle)(|0\rangle+|1\rangle)(|0\rangle-|1\rangle) \\
& (|0\rangle-|1\rangle)(|0\rangle-|1\rangle)(|0\rangle+|1\rangle)(|0\rangle+|1\rangle)\Big\} \\
|11_L\rangle \rightarrow \frac{1}{16}\Big\{ & (|0\rangle-|1\rangle)(|0\rangle-|1\rangle)(|0\rangle-|1\rangle)(|0\rangle+|1\rangle) \\
& (|0\rangle-|1\rangle)(|0\rangle+|1\rangle)(|0\rangle-|1\rangle)(|0\rangle-|1\rangle)\Big\}
\end{aligned}
\tag{4.23}
$$

Ou seja, cada estado da base foi codificado em uma superposição de duzentos e cinquenta e seis estados, cada um com oito qubits. Considerando que o estado inicial $|\psi\rangle = a_1a_2|00\rangle + a_1b_2|01\rangle + b_1a_2|10\rangle + b_1b_2|11\rangle$ *tenha sido perfeitamente codificado, pode-se garantir a correção de até dois erros* Z *em quaisquer dos oito qubits da palavra-código. Dito de outra forma, este código é capaz de corrigir dois erros de sinal em quaisquer dos oito blocos acima.*

Suponha que desejamos detectar e corrigir erros Z *que eventualmente possam ter ocorrido durante a transmissão da palavra-código (4.23). As linhas da matriz (4.13) fornecem-nos os geradores do estabilizador. Estes geradores estão dispostos na Tabela 4.8.*

É fácil de verificar que cada um dos geradores M_t *da Tabela 4.8 satisfaz a relação* $M_t|\psi\rangle = |\psi\rangle$*, onde* $|\psi\rangle$ *é palavra-código (4.23). Nesta tabela também são apresentadas as operações lógicas do código,* $\overline{Z}_1$ *e* $\overline{Z}_2$*. Note que, de fato, as seguintes relações são satisfeitas:*

$$\begin{aligned}
\overline{Z}_1|00_L\rangle = |00_L\rangle \quad &; \quad \overline{Z}_2|00_L\rangle = |00_L\rangle \\
\overline{Z}_1|01_L\rangle = |01_L\rangle \quad &; \quad \overline{Z}_2|01_L\rangle = -|01_L\rangle \\
\overline{Z}_1|10_L\rangle = -|10_L\rangle \quad &; \quad \overline{Z}_2|01_L\rangle = -|01_L\rangle \\
\overline{Z}_1|11_L\rangle = -|11_L\rangle \quad &; \quad \overline{Z}_2|11_L\rangle = -|11_L\rangle
\end{aligned} \tag{4.24}$$

M_0	X	X	I	I	I	I	I	I
M_1	X	I	X	X	I	I	I	I
M_2	X	X	X	I	X	X	I	I
M_3	I	I	X	X	X	I	X	X
M_4	I	I	I	I	X	X	X	I
M_5	I	I	I	I	I	I	X	X
$\overline{Z}_1$	Z	Z	I	Z	Z	Z	I	I
$\overline{Z}_2$	I	I	Z	Z	I	Z	Z	Z

Tabela 4.8: Geradores e operações lógicas do CCQ [2, 1, 2] descrito pela operação de codificação (4.16) para dois qubits.

Usaremos os seis observáveis da Tabela 4.8 para realizar medidas da palavra-código recebida à saída do canal. Os resultados destas medidas nos permitem detectar sem ambiguidade qualquer grupo de até dois erros Z que possa ter ocorrido nos oito qubits da palavra-código original.

Por exemplo, suponha que o resultado da medida dos seis observáveis seja novamente dado pelo conjunto de autovalores $(-1,+1,-1,-1,+1,+1)$, considerado no Exemplo 10. O resultado da aplicação do ADS para este conjunto de autovalores foi mostrado na Figura 4.2. Vimos que o vetor erro detectado pelo ADS é $\mathbf{e} = (10\,01\,00\,00)$. Isto significa que a palavra-código recebida foi:

$$
\begin{aligned}
|00_L\rangle \rightarrow \frac{1}{16}\Big\{ & (|0\rangle - |1\rangle)(|0\rangle + |1\rangle)(|0\rangle + |1\rangle)(|0\rangle - |1\rangle) \\
& (|0\rangle + |1\rangle)(|0\rangle + |1\rangle)(|0\rangle + |1\rangle)(|0\rangle + |1\rangle)\Big\} \\
|01_L\rangle \rightarrow \frac{1}{16}\Big\{ & (|0\rangle - |1\rangle)(|0\rangle + |1\rangle)(|0\rangle - |1\rangle)(|0\rangle + |1\rangle) \\
& (|0\rangle + |1\rangle)(|0\rangle - |1\rangle)(|0\rangle - |1\rangle)(|0\rangle - |1\rangle)\Big\} \\
|10_L\rangle \rightarrow \frac{1}{16}\Big\{ & (|0\rangle + |1\rangle)(|0\rangle - |1\rangle)(|0\rangle + |1\rangle)(|0\rangle + |1\rangle) \\
& (|0\rangle - |1\rangle)(|0\rangle - |1\rangle)(|0\rangle + |1\rangle)(|0\rangle + |1\rangle)\Big\} \\
|11_L\rangle \rightarrow \frac{1}{16}\Big\{ & (|0\rangle + |1\rangle)(|0\rangle - |1\rangle)(|0\rangle - |1\rangle)(|0\rangle - |1\rangle) \\
& (|0\rangle - |1\rangle)(|0\rangle + |1\rangle)(|0\rangle - |1\rangle)(|0\rangle - |1\rangle)\Big\}
\end{aligned}
\tag{4.25}
$$

Para corrigirmos os erros de sinal no primeiro e quarto blocos, basta aplicarmos os operadores Z_1 e Z_4 sobre esta palavra-código. Como o processo de correção é análogo ao do Exemplo 10, algumas palavras-código com mais de dois erros Z também podem ser corrigidas.

Podemos continuar com este procedimento sistemático para codificar quantos qubits desejarmos. Se quisermos codificar N qubits, cada uma das 2^N palavras-código da base será, na base $\{|0\rangle, |1\rangle\}$, uma superposição de $2^{2(N+2)}$ estados, cada um com $2(N+2)$ qubits.

4.3 CCQ para o Canal Quântico com Erro Geral

Até o momento construímos dois códigos quânticos, $\mathcal{C}_1$ e $\mathcal{C}_2$, capazes de corrigir erros de fase Z e de bit X, respectivamente, para um mesmo conjunto de registros quânticos. Os códigos $\mathcal{C}_1$ e $\mathcal{C}_2$ são CCQs [2, 1, 2]. Para que $\mathcal{C}_2$ possa ser concatenado a $\mathcal{C}_1$, devemos antes construir um código *equivalente* a $\mathcal{C}_2$ com taxa 2/4. A opção *trivial* para esta tarefa

é usar a *mesma* matriz geradora (4.2) usada na construção de $\mathcal{C}_2$, mas agora com dois bits entrando no codificador a cada instante de tempo e com cada um dos dois registros de deslocamento contendo apenas uma memória de modo que o número total de memórias permaneça o mesmo de $\mathcal{C}_1$. Assim, o código equivalente a $\mathcal{C}_2$ com taxa $2/4$ será construído a partir da matriz geradora (4.2) escrita na forma[4]:

$$\mathbf{G} = \begin{bmatrix} 1101 & 1100 & & & & \\ 0011 & 0111 & & & & \\ & 1101 & 1100 & & & \\ & 0011 & 0111 & & & \\ & & 1101 & 1100 & & \\ & & 0011 & 0111 & & \\ & & & \ddots & \ddots & \\ & & & \ddots & \ddots & \end{bmatrix}. \tag{4.26}$$

Desta forma, a operação de codificação quântica para o código $\mathcal{C}_2$ [4, 2, 1] será:

$$\bigotimes_{t=0}^{+\infty} |u_t^{(1)},\, u_t^{(2)}\rangle \longmapsto \bigotimes_{t=0}^{+\infty} |u_t^{(1)} + u_{t-1}^{(1)},\, u_t^{(1)} + u_{t-1}^{(1)} + u_{t-1}^{(2)},\, u_t^{(2)} + u_{t-1}^{(2)},\, u_t^{(2)} + u_{t-1}^{(2)} + u_t^{(1)}\rangle, \tag{4.27}$$

onde $u_t^{(1)},\, u_t^{(2)} \in \{0,1\}$ e $u_{-1} = u_{-2} = 0$. Esta operação de codificação quântica é a equivalente trivial com taxa $2/4$ e memória unitária da operação (4.1). Por serem equivalentes, a capacidade de correção do novo código é a mesma, ou seja, podemos garantir com $\mathcal{C}_2$ [4, 2, 1] a correção de qualquer grupo com até dois erros X.

A matriz verificação de paridade associada ao CCQ $\mathcal{C}_2$ [4, 2, 1] é a matriz (4.3) escrita na forma:

[4]Veja o Exemplo 3 no Capítulo 3.

$$\mathbf{H} = \begin{bmatrix} 1100 & & & & & \\ 1011 & & & & & \\ 1110 & 1100 & & & & \\ 0011 & 1011 & & & & \\ & 1110 & 1100 & & & \\ & 0011 & 1011 & & & \\ & & 1110 & 1100 & & \\ & & 0011 & 1011 & & \\ & & & \ddots & \ddots & \\ & & & \ddots & \ddots & \end{bmatrix}. \tag{4.28}$$

Como as matrizes (4.3) e (4.28) são exatamente a mesma matriz, podemos usar as soluções (3.40) da equação de síndromes (3.39) para determinar a sequência de erros $\mathbf{e} = (\mathbf{e}^{(1)}, \mathbf{e}^{(2)}, \mathbf{e}^{(3)}, \mathbf{e}^{(4)})$ da equação de síndromes $\mathbf{s} = \mathbf{e}\mathbf{H}^T$, onde $\mathbf{s} = (\mathbf{s}^{(1)}, \mathbf{s}^{(2)})$ e $\mathbf{e} = (\mathbf{e}^{(1)}, \mathbf{e}^{(2)}, \mathbf{e}^{(3)}, \mathbf{e}^{(4)})$.

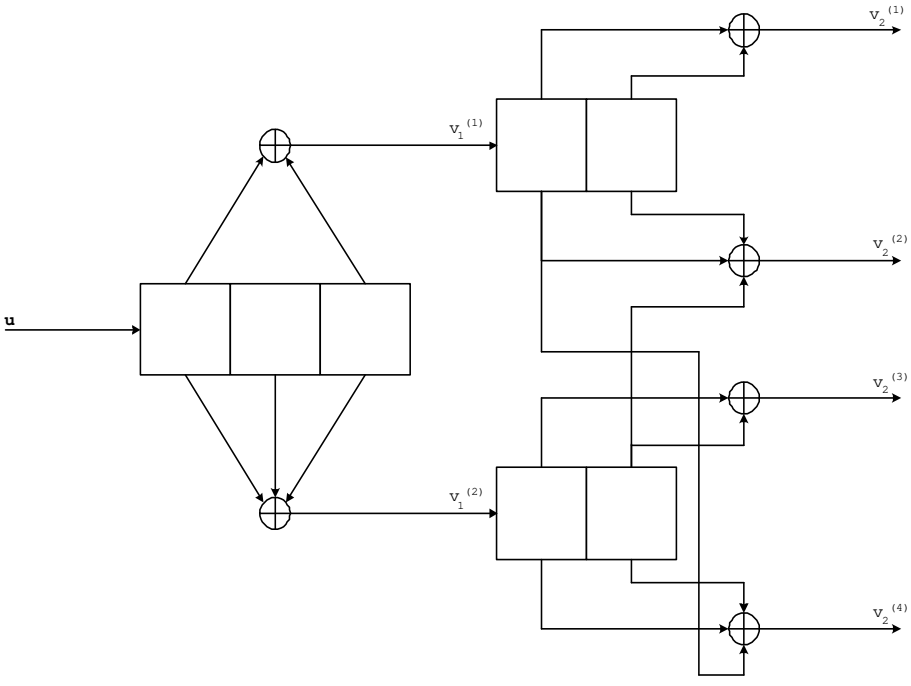

Figura 4.3: Concatenação dos codificadores (2, 1, 2) e (4, 2, 1).

Quando fazemos a concatenação dos CCCs (2, 1, 2) e (4, 2, 1), respectivamente, associados aos CCQs $\mathcal{C}_1$ [2, 1, 2] e $\mathcal{C}_2$ [4, 2, 1], obtemos um novo CCC, de taxa 1/4. Veja o CSL deste CCC concatenado na Figura 4.3.

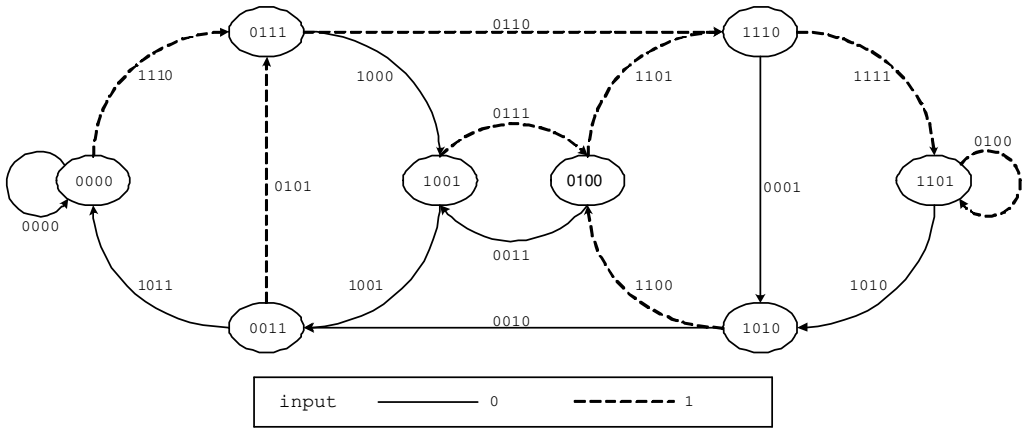

Figura 4.4: Diagrama de estados para o CCC concatenado (4, 1, 3).

Este codificador concatenado não-catastrófico possui três memórias *efetivas*, duas do primeiro codificador e uma do segundo. Assim, apenas oito estados (e não dezesseis, como poderia sugerir o número total de memórias do codificador) são gerados por este codificador. Isto pode ser facilmente verificado através de seu diagrama de estados, apresentado na Figura 4.4. Outra possível representação gráfica para este codificador, equivalente à primeira, é o diagrama de treliça da Figura 4.5.

É fácil de verificar em qualquer uma destas duas representações gráficas que são necessárias pelo menos quatro transições para sair do estado inicial 0000 e retornar ao mesmo estado ($0000 \rightarrow 0111 \rightarrow 1001 \rightarrow 0011 \rightarrow 0000$). E o caminho mais curto é também o caminho de d_{free}. Ou seja, não há nenhum outro caminho não nulo que saia do estado inicial 0000 e retorne ao mesmo estado com peso menor do que nove (palavra-código $1110 \rightarrow 1000 \rightarrow 1001 \rightarrow 1011$). Logo o d_{free} deste CCC é nove e, portanto, é capaz de corrigir qualquer grupo com até quatro erros sobre a palavra-código.

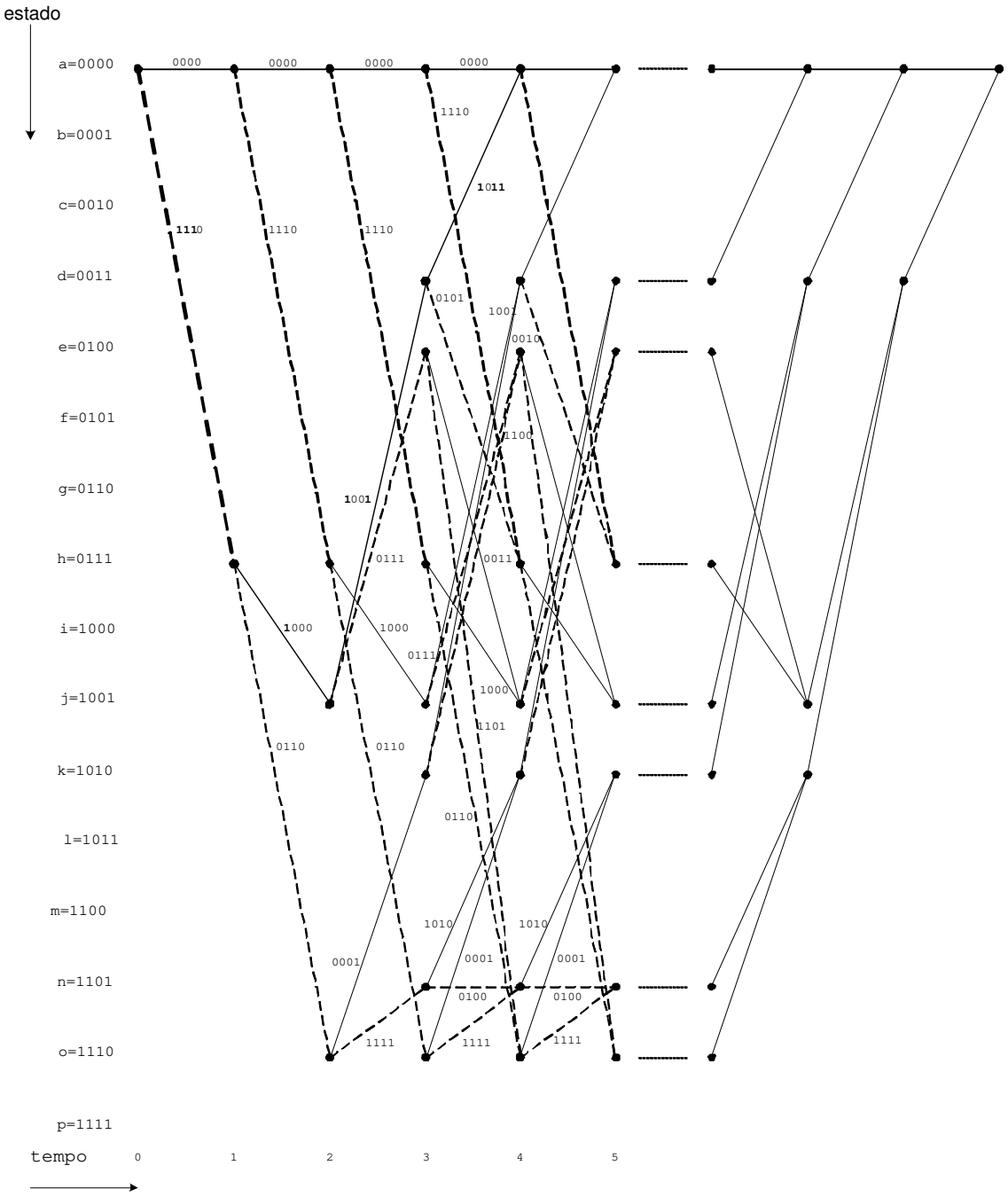

Figura 4.5: Diagrama de treliça para o CCC concatenado (4, 1, 3).

As Figuras 4.6 e 4.7 mostram os decodificadores de síndromes para os CCCs (2, 1, 2) e (4, 2, 1). Estes dois decodificadores são equivalentes: o decodificador da Figura 4.6 produz as *mesmas* síndromes do decodificador da Figura 4.7, embora com o dobro do tempo.

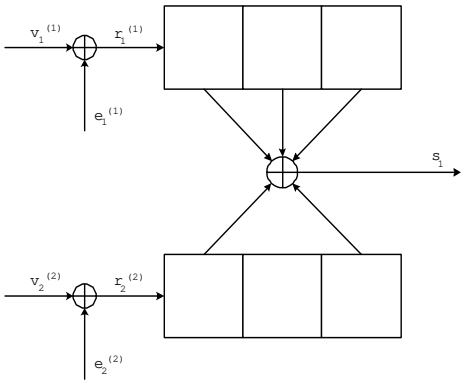

Figura 4.6: Decodificador de síndromes para o CCC (2, 1, 2).

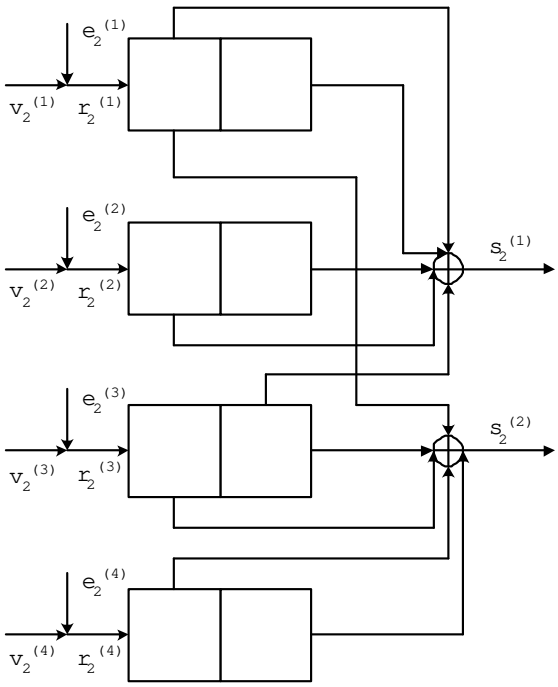

Figura 4.7: Decodificador de síndromes para o CCC (4, 2, 1).

O CCC (4, 1, 3) pode ser usado na contrução de um CCQ capaz de garantir a correção de um erro quântico geral com geradores Z e X em qualquer dos qubits da palavra-código quântica. Isto porque três condições são satisfeitas:

1. O CCC (2, 1, 2), associado a $\mathcal{C}_1$ [2, 1, 2], garante a correção de um erro (no caso estudado garante a correção de até mesmo dois erros). Este erro clássico está associado ao erro quântico Z;

2. O CCC (4, 2, 1), associado a $\mathcal{C}_2$ [4, 2, 1], garante a correção de um erro (no caso estudado garante a correção de até mesmo dois erros). Este erro clássico está associado ao erro quântico X;

3. O CCC (4, 1, 3), associado à concatenação de $\mathcal{C}_2$ [4, 2, 1] a $\mathcal{C}_1$ [2, 1, 2], garante a correção de quatro erros. Estes erros clássicos estão associados aos erros quânticos Z, X, Y $(=XZ)$ e I, que constituem a base para um erro quântico geral.

Portanto, $\mathcal{C}_2$ [4, 2, 1] codifica o estado quântico resultante da codificação por $\mathcal{C}_1$ [2, 1, 2]. O código resultante desta concatenação, $\mathcal{C}$, é um CCQ com taxa 1/4. Definindo o número de *memórias convolucionais quânticas* como $m = T - 1$, onde T é o número de transições de unidades de tempo necessárias para se obter uma palavra-código quântica válida[5], teremos que a memória do CCQ concatenado é $m = 3$. Pode-se então dizer que $\mathcal{C}$ é um CCQ [4, 1, 3].

A operação de codificação do CCQ para um canal quântico com erro geral pode agora ser convenientemente escrita na base $\{|0\rangle, |1\rangle\}$ como:

$$\bigotimes_{t=0}^{+\infty} |u_t\rangle \longmapsto \bigotimes_{t=0}^{+\infty} \left\{ \sum_{(p_t,\, q_t)=(0,0)}^{(1,1)} \frac{1}{2}(-1)^{(u_t+u_{t-2})p_t+(u_t+u_{t-1}+u_{t-2})q_t} \right. \\ \left. |p_t + p_{t-1}, p_t + p_{t-1} + q_{t-1}, q_t + q_{t-1}, q_t + q_{t-1} + p_t\rangle \right\}, \tag{4.29}$$

onde $u_{-1} = u_{-2} = 0$ e $p_{-1} = q_{-1} = 0$.

[5]Esta definição é análoga à definição clássica de memória.

Houve a necessidade de se usar dois codificadores clássicos concatenados porque o erro deste canal quântico possui dois geradores, Z e X.

Como o d_{free} do CCC associado é exatamente nove, este CCQ não é capaz de corrigir nenhum erro além do erro geral, o que nos leva a concluir que a distância quântica é três. Além disso, como os CCCs (2, 1, 2), (4, 2, 1) e (4, 1, 3) são não-catastróficos, segue que o CCQ [4, 1, 3] associado também é um código não-catastrófico.

É importante ressaltar que se quiséssemos construir um CCQ que corrigisse apenas um erro X ou apenas um erro Z, poderíamos ter usado um CCC mais simples, de memória unitária, como por exemplo $\mathbf{G}(D) = [1+D, D]$, que tem $d_{free} = 3$ e que, portanto, garante ao CCQ associado a correção de até um erro X ou Z. Porém, este CCC quando concatenado a um equivalente de taxa $2/4$ geraria um CCC com $d_{free} < 9$ e, assim, o CCQ $\mathcal{C}$, associado à concatenação, não garantiria a correção de um erro quântico geral. Para evitar esta situação, achamos mais prudente iniciar o estudo de CCQs concatenados com o exemplo discutido neste capítulo. Outras situações serão estudadas com mais detalhes no próximo capítulo.

4.3.1 Formalismo Estabilizador

A matriz geradora discreta semi-infinita do codificador convolucional concatenado da Figura 4.3 tem a seguinte forma:

$$\mathbf{G}_C = \begin{bmatrix} 1110 & 1000 & 1001 & 1011 & & & & \\ & 1110 & 1000 & 1001 & 1011 & & & \\ & & 1110 & 1000 & 1001 & 1011 & & \\ & & & 1110 & 1000 & 1001 & 1011 & \\ & & & & \ddots & \ddots & \ddots & \ddots \end{bmatrix}. \tag{4.30}$$

Chamemos a matriz (4.28) de $\mathbf{H}_X$, e a matriz (4.3), expandida para o código $\mathcal{C}$ (basta tomar cada uma das linhas como um bloco de informação a ser codificado pela matriz (4.26)($\equiv \mathbf{G}_X$)), de $\mathbf{H}_Z$. Com as matrizes $\mathbf{H}_X$ e $\mathbf{H}_Z$ podemos construir a matriz $\mathbf{H}_C = [\mathbf{H}_X; \mathbf{H}_Z]$:

$$\mathbf{H}_C = \left[\begin{array}{ccccccc}
1100 & & & & & & \\
1011 & & & & & & \\
1110 & 1100 & & & & & \\
0011 & 1011 & & & & & \\
 & 1110 & 1100 & & & & \\
 & 0011 & 1011 & & & & \\
 & & 1110 & 1100 & & & \\
 & & 0011 & 1011 & & & \\
 & & & 1110 & 1100 & & \\
 & & & 0011 & 1011 & & \\
 & & & & \ddots & \ddots & \\
 & & & & \ddots & \ddots & \\
1110 & 1011 & & & & & \\
1101 & 0010 & 1011 & & & & \\
1110 & 0110 & 0010 & 1011 & & & \\
 & 1110 & 0110 & 0010 & 1011 & & \\
 & & 1110 & 0110 & 0010 & 1011 & \\
 & & & \ddots & \ddots & \ddots &
\end{array}\right]. \tag{4.31}$$

É fácil de verificar que, quando convenientemente truncadas, temos $\mathbf{G}_C\mathbf{H}_C^T = 0$. As linhas da matriz (4.31) permitem-nos escrever os geradores do grupo estabilizador S do CCQ [4, 1, 3] descrito pela operação de codificação (4.29). Usamos as linhas da matriz $\mathbf{H}_X$, para obter os geradores Z, e as linhas da matriz $\mathbf{H}_Z$, para obter os geradores X. Estes geradores são mostrados na Tabela 4.9.

Todos os geradores da Tabela 4.9 comutam e são independentes entre si. Portanto, o subespaço do código (isto é, o maior subespaço comum dos geradores com autovalor $+1$) é não-trivial. É fácil de verificar que qualquer M_t do conjunto de geradores da Tabela 4.9 satisfaz a relação $M_t|\psi\rangle = |\psi\rangle$, onde $|\psi\rangle$ é a palavra-código gerada pela operação de codificação (4.29).

Como, a princípio, o CCQ será usado para codificar uma sequência infinita de qubits de informação, o número de geradores do grupo estabilizador será também infinito. Na

$M_{X,0}$	Z	Z	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	$\cdots$	$\cdots$
$M_{X,1}$	Z	I	Z	Z	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	$\cdots$	$\cdots$
$M_{X,2}$	Z	Z	Z	I	Z	Z	I	I	I	I	I	I	I	I	I	I	I	I	I	I	$\cdots$	$\cdots$
$M_{X,3}$	I	I	Z	Z	Z	I	Z	Z	I	I	I	I	I	I	I	I	I	I	I	I	$\cdots$	$\cdots$
$M_{X,4}$	I	I	I	I	Z	Z	Z	I	Z	Z	I	I	I	I	I	I	I	I	I	I	$\cdots$	$\cdots$
$M_{X,5}$	I	I	I	I	I	I	Z	Z	Z	I	Z	Z	I	I	I	I	I	I	I	I	$\cdots$	$\cdots$
$M_{X,6}$	I	I	I	I	I	I	I	I	Z	Z	Z	I	Z	Z	I	I	I	I	I	I	$\cdots$	$\cdots$
$M_{X,7}$	I	I	I	I	I	I	I	I	I	I	Z	Z	Z	I	Z	Z	I	I	I	I	$\cdots$	$\cdots$
$M_{X,8}$	I	I	I	I	I	I	I	I	I	I	I	I	Z	Z	Z	I	Z	Z	I	I	$\cdots$	$\cdots$
$M_{X,9}$	I	I	I	I	I	I	I	I	I	I	I	I	I	I	Z	Z	Z	I	Z	Z	$\cdots$	$\cdots$
$\vdots$																	$\ddots$		$\ddots$		$\ddots$	
$M_{X,\infty}$	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	Z	Z.
$M_{Z,0}$	X	X	X	I	X	I	X	X	I	I	I	I	I	I	I	I	I	I	I	I	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$M_{Z,1}$	X	X	I	X	I	I	X	I	X	I	X	X	I	I	I	I	I	I	I	I	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$M_{Z,2}$	X	X	X	I	I	X	X	I	I	I	X	I	X	I	X	X	I	I	I	I	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$M_{Z,3}$	I	I	I	I	X	X	X	I	I	X	X	I	I	I	X	I	X	I	X	X	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$M_{Z,4}$	I	I	I	I	I	I	I	I	X	X	X	I	I	X	X	I	I	I	X	I	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$\vdots$													$\ddots$				$\ddots$					
$M_{Z,\infty}$	I	I	I	I	I	I	I	I	X	X	X	I	I	X	X	I	I	I	X	I	X	I	X	X.

Tabela 4.9: Geradores do CCQ [4, 1, 3] descrito pela operação de codificação (4.29).

prática, estamos sempre interessados em codificar um número finito de qubits de informação. Isto transforma o CCQ em um CBQ gerado através de uma operação de convolução. Portanto, na prática, devemos somente considerar um número finito de geradores.

Para uma sequência de informação com N qubits, precisamos considerar somente $2(N+2)-N = N+4$ geradores X e $2(N+4) = 2N+8$ geradores Z geradores para descrever o subespaço do código (o número total de geradores, $N+4+2(N+4) = 3N+12$, será sempre a diferença entre o comprimento do código e o número de qubits de informação). Com este conjunto de geradores seremos capazes de garantir a identificação de um erro geral com geradores Z e X que eventualmente possa ter ocorrido sobre qualquer qubit da palavra-código (4.29).

Para manipularmos este código estabilizador através de operações lógicas, devemos encontrar operadores de Pauli lógicos $\overline{X}$ e $\overline{Z}$ correspondentes a cada um dos qubits lógicos.

Vimos no Capítulo 2 que estes operadores devem satisfazer as seguintes relações:

$$\begin{cases} \overline{X}_i, \overline{Z}_i \in N(S)/S \\ \\ \forall\, i \neq j,\ [\overline{X}_i, \overline{X}_j] = [\overline{Z}_i, \overline{Z}_j] = [\overline{X}_i, \overline{Z}_j] = 0 \\ \\ \{\overline{X}_i, \overline{Z}_i\} = 0, \end{cases} \tag{4.32}$$

onde $N(S)$ é o normalizador de S. A primeira destas condições impõe que os operadores de Pauli deixem o subespaço do código globalmente invariante, mas tenham uma ação não trivial sobre seus elementos, enquanto que a segunda condição assegura que a manipulação do qubit i não afeta os outros qubits. Podemos usar as linhas da matriz geradora (4.30) do CCC associado (4, 1, 3) para determinar as operações lógicas tanto de $\overline{X}_i$ quanto de $\overline{Z}_i$, já que estas devem ser independentes dos geradores do estabilizador e comutar com cada um deles. Estes operadores estão dispostos na Tabela 4.10.

$\overline{X}_1$	Z	Z	Z	I	Z	I	I	I	Z	I	I	Z	Z	I	Z	Z	I	I	I	I	I	I	I	I	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$\overline{X}_2$	I	I	I	I	Z	Z	Z	I	Z	I	I	I	Z	I	I	Z	Z	I	Z	Z	I	I	I	I	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$\vdots$									$\ddots$				$\ddots$				$\ddots$				$\ddots$							
$\overline{X}_\infty$	I	I	I	I	I	I	I	I	I	I	I	I	Z	Z	Z	I	Z	I	I	I	Z	I	I	Z	Z	I	Z	Z.
$\overline{Z}_1$	X	X	X	I	X	I	I	I	X	I	I	X	X	I	X	X	I	I	I	I	I	I	I	I	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$\overline{Z}_2$	I	I	I	I	X	X	X	I	X	I	I	I	X	I	I	X	X	I	X	X	I	I	I	I	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$\vdots$									$\ddots$				$\ddots$				$\ddots$				$\ddots$							
$\overline{Z}_\infty$	I	I	I	I	I	I	I	I	I	I	I	I	X	X	X	I	X	I	I	I	X	I	I	X	X	I	X	X.

Tabela 4.10: Operações lógicas do CCQ [4, 1, 3] descrito pela operação de codificação (4.29).

Como o CCC (2, 1, 2) para uma sequência de informação de comprimento N é uma sequência de comprimento $2(N+2)$ e o CCC (4, 2, 1) para um sequência de informação de comprimento $2(N+2)$ é uma sequência de comprimento $2(2(N+2)+2) = 4N+12$, podemos afirmar para o CCQ [4, 1, 3] associado à concatenação dos CCCs (2, 1, 2) e (4, 2, 1) que uma transformação unitária sobre N qubits *lógicos* será implementada através

de uma transformação unitária sobre $4N+12$ qubits *físicos*. Isto ficará mais claro com os exemplos que serão apresentados para $N=1$ e $N=2$ qubits de informação.

4.3.2 Identificação e Correção de Erros

A cada instante de tempo cada um dos quatro qubits gerados pelo CCQ [4, 1, 3] é passado através de uma cópia idêntica de um canal bit-phase flip sem memória. Nosso problema agora é detectar e corrigir os erros de bit e de fase que eventualmente possam ter ocorrido sobre a palavra-código durante a transmissão pelo canal.

Com a medida da palavra-código recebida à saída do canal pelo conjunto de observáveis da Tabela 4.9, é possível determinar qual erro eventualmente ocorreu sobre a palavra-código original. Os resultados destas medidas são um conjunto de $2N+8$ autovalores $\{\alpha_{M_{X,t}}\}$ e um conjunto de $N+4$ autovalores $\{\alpha_{M_{Z,t}}\}$, ambos podendo assumir os valores $+1$ ou -1. Usaremos os autovalores $\{\alpha_{M_{X,t}}\}$ e $\{\alpha_{M_{Z,t}}\}$ nas soluções (4.5) da equação de síndromes para detectar, através de *dois* diagramas de treliça, os qubits onde eventualmente ocorreram erros de bit e os *blocos* do código phase-flip associado onde eventualmente ocorreram erros de fase. Aqui, como no código de Shor, os geradores X não detectam as posições dos qubits, mas sim os blocos do código phase flip associado, onde eventualmente ocorreram erros de fase.

As medidas dos geradores Z nos permitem detectar sem ambiguidade qualquer grupo com até dois erros de bit sobre os qubits do código bit-phase flip. Assim, podemos corrigir quaisquer dois qubits com erros de bit. Já as medidas dos geradores X nos permitem detectar sem ambiguidade qualquer grupo com até dois erros de fase sobre os blocos do código phase flip. Porém, só nos permite detectar sem ambiguidade apenas um erro de fase no código bit-phase flip correspondente[6].

No caso do código de Shor, detectado o erro de fase no bloco, é fácil de verificar quais os possíveis qubits onde ocorreu o erro de fase. No caso de um CCQ concatenado, esta tarefa é mais complexa porque o código bit-phase flip não pode ser compactado como

[6]Uma simulação computacional que considere até dois erros Z atuando sobre o código bit-phase flip nos permite mostrar que não conseguimos associar *síndromes distintas* para padrões de erros com efeitos globais distintos sobre a palavra-código.

um produto tensorial de blocos como é feito no código de Shor. No caso de um CCQ concatenado, isto só pode ser feito com o código phase flip associado.

Para poucos qubits de informação, é relativamente fácil verificar para qual qubit do código bit-phase flip os erros de fase nos bloco do código phase flip se propagam, mas não é óbvio. Para muitos qubits de informação, será necessário criar um algoritmo que permita descobrir como erros de fase em blocos de um código phase flip se propagam para os qubits de um código bit-phase flip. De qualquer forma, o uso do ADS, aliado a um algoritmo de verificação de propagação de erros de fase (ambos de complexidade linear com o número de qubits), é uma solução muito melhor do que o uso de um algoritmo que resolve "heuristicamente" a equação de síndromes $\mathbf{s} = \mathbf{r}\mathbf{H}^T$ (de complexidade quadrática com o número de qubits de informação).

Podemos nos deparar com a situação em que detectamos erros de fase em dois blocos do código phase flip se propagando para um erro de fase em um qubit do código bit-phase flip. Isto ocorre devido às características do produto tensorial do código phase flip em questão. Esta situação será apresentada em um dos exemplos abaixo.

O CCQ concatenado [4, 1, 3] tem outra importante propriedade: assim como o código concatenado de Shor de nove qubits, o CCQ [4, 1, 3] é um código degenerado. Ou seja, teremos erros de fase atuando em qubits diferentes e associados a um mesmo conjunto de autovalores (ou de síndromes) dos geradores X. Assim como no código de Shor, não é possível, nem necessário, distinguir tais erros. Basta fazer a correção de *qualquer* um dos possíveis qubits com aquele erro de fase para fazer a correção do código bit-phase flip.

Um erro Z atuando sobre o primeiro qubit tem o mesmo efeito global sobre a palavra-código de um erro Z atuando sobre o segundo qubit, *independente* do tamanho da palavra-código (ou seja, independente de quantos qubits codifiquemos). O mesmo vale para o penúltimo e o último qubits da palavra-código. Isto porque os dois primeiros e os dois últimos qubits da palavra-código serão sempre iguais[7].

Nos exemplos a seguir para $N = 1$ e $N = 2$, mostraremos como detectar e corrigir o pior dos casos: quando ocorre um erro de bit e um erro de fase sobre o mesmo qubit. As situações mais simples, em que ocorrem apenas erros de bit ou apenas erros de fase, são

[7]Para verificar isto, repare que os dois primeiros e os dois últimos bits da palavra-código do diagrama de estados da Figura 3.5 são sempre idênticos (00 ou 11), independente de quantos bits codifiquemos.

análogas às abordadas nas seções anteriores e não serão consideradas. Se somos capazes de detectar e corrigir um erro de bit e de fase sobre um mesmo qubit para todos os qubits do código bit-phase flip, podemos afirmar que somos capazes também de corrigir um erro geral sobre qualquer qubit do código bit-phase flip [63].

4.3.3 Exemplos: $N = 1$ e $N = 2$

A seguir, apresentamos o CCQ descrito pela operação de codificação (4.29) para uma sequência de informação composta de um e de dois registros quânticos. Os códigos para sequências de informação mais longas podem ser obtidos de maneira similar.

Exemplo 13. *Suponha que desejamos codificar um registro quântico, ou seja, um qubit. Nossa operação de codificação é realizada através da codificação pelo CCC (4, 2, 1) de cada uma das sessenta e quatro sequências de seis bits do CCQ para o canal phase flip (4.19). Veja o código gerado por esta operação na Tabela 4.14, que é exibida no final deste capítulo.*

Para melhor entendermos a dinâmica desta operação de codificação: para cada um dos estados da base do qubit foi gerada uma superposição de quatro estados de comprimento quatro no estágio 0, uma superposição de dezesseis estados de comprimento oito no estágio 1, uma superposição de sessenta e quatro estados de comprimento doze no estágio 2, e, finalmente, uma superposição de sessenta e quatro estados de comprimento dezesseis no estágio 3. Portanto, o número de memórias quânticas é, de fato, três.

Considerando que o estado inicial $a|0\rangle + b|1\rangle$ *tenha sido perfeitamente codificado, podemos garantir a correção de um erro quântico geral em qualquer um dos dezesseis qubits das palavras-código.*

Os quinze geradores deste CCQ podem ser obtidos a partir da matriz verificação de paridade $\mathbf{H}_C = [\mathbf{H}_X; \mathbf{H}_Z]$ *(4.31) truncada:*

$$\mathbf{H}_C = \begin{bmatrix}
1100 & & & \\
1011 & & & \\
1110 & 1100 & & \\
0011 & 1011 & & \\
 & 1110 & 1100 & \\
 & 0011 & 1011 & \\
 & & 1110 & 1100 \\
 & & 0011 & 1011 \\
 & & & 1110 \\
 & & & 0011 \\
1110 & 1011 & & \\
1101 & 0010 & 1011 & \\
1110 & 0110 & 0010 & 1011 \\
 & 1110 & 0110 & 1100 \\
 & & 1110 & 1011
\end{bmatrix}. \tag{4.33}$$

Repare que a matriz (4.33) é uma matriz verificação de paridade da matriz geradora (4.30) truncada:

$$\mathbf{G}_C = \begin{bmatrix} 1110 & 1000 & 1001 & 1011 \end{bmatrix}. \tag{4.34}$$

As linhas da matriz (4.33) e a linha da matriz (4.34) fornecem-nos, respectivamente, os geradores do estabilizador e as operações lógicas $\overline{X}$ e $\overline{Z}$ do CCQ. Veja a Tabela 4.11. É fácil de verificar que cada um dos geradores desta tabela satisfaz a relação $M_t|\psi\rangle = |\psi\rangle$, onde $|\psi\rangle$ é palavra-código da Tabela 4.14. Observe também que as seguintes relações são satisfeitas pelos operadores lógicos $\overline{X}$ e $\overline{Z}$:

$$\begin{aligned}
\overline{X}|0_L\rangle = |1_L\rangle \quad &; \quad \overline{X}|1_L\rangle = |0_L\rangle \\
\overline{Z}|0_L\rangle = |0_L\rangle \quad &; \quad \overline{Z}|1_L\rangle = -|1_L\rangle
\end{aligned} \tag{4.35}$$

Usamos os quinze observáveis da Tabela 4.11 para realizar medidas da palavra-código recebida à saída do canal. Os resultados destas medidas permitem-nos detectar sem am-

M_0	Z	Z	I	I	I	I	I	I	I	I	I	I	I	I	I	I
M_1	Z	I	Z	Z	I	I	I	I	I	I	I	I	I	I	I	I
M_2	Z	Z	Z	I	Z	Z	I	I	I	I	I	I	I	I	I	I
M_3	I	I	Z	Z	Z	I	Z	Z	I	I	I	I	I	I	I	I
M_4	I	I	I	I	Z	Z	Z	I	Z	Z	I	I	I	I	I	I
M_5	I	I	I	I	I	I	Z	Z	Z	I	Z	Z	I	I	I	I
M_6	I	I	I	I	I	I	I	I	Z	Z	Z	I	Z	Z	I	I
M_7	I	I	I	I	I	I	I	I	I	I	Z	Z	Z	I	Z	Z
M_8	I	I	I	I	I	I	I	I	I	I	I	I	Z	Z	Z	I
M_9	I	I	I	I	I	I	I	I	I	I	I	I	I	I	Z	Z
M_{10}	X	X	X	I	X	I	X	X	I	I	I	I	I	I	I	I
M_{11}	X	X	I	X	I	I	X	I	X	I	X	X	I	I	I	I
M_{12}	X	X	X	I	I	X	X	I	I	I	X	I	X	I	X	X
M_{13}	I	I	I	I	X	X	X	I	I	X	X	I	X	X	I	I
M_{14}	I	I	I	I	I	I	I	I	X	X	X	I	X	I	X	X
$\overline{X}$	Z	Z	Z	I	Z	I	I	I	Z	I	I	Z	Z	I	Z	Z
$\overline{Z}$	X	X	X	I	X	I	I	I	X	I	I	X	X	I	X	X

Tabela 4.11: Geradores e operações lógicas do CCQ [4, 1, 3] descrito pela operação de codificação (4.29) para um qubit.

biguidade um erro geral que eventualmente possa ter ocorrido em qualquer um dos dezesseis qubits da palavra-código original.

Por exemplo, suponha que o resultado das medidas dos dez observáveis $\mathbf{Z}$ *seja dado pelo conjunto de autovalores* $(+1, +1, -1, -1, -1, +1, +1, +1, +1, +1)$ *e o resultado das medidas dos cinco observáveis* $\mathbf{X}$ *seja dado pelo conjunto de autovalores* $(-1, +1, +1, -1, +1)$. *Os resultados da aplicação do ADS para estes dois conjunto de autovalores são mostrados na Figura 4.8.*

O caminho de menor peso das duas primeiras treliças da Figura 4.8 é o caminho $a \to a \to a \to c \to d \to b \to a \to a \to a \to a$. Os ramos deste caminho produzem $\hat{\mathbf{e}} = [00\ 00\ 10\ 00\ 00\ 00\ 00\ 00]$ como estimativa da sequência de erro de bit sobre a palavra-código da Tabela 4.14, ou seja, o ADS detectou um erro de bit no quinto qubit. O caminho de menor peso da terceira treliça da Figura 4.8 é o caminho $a \to c \to b \to c \to b \to a$. Os ramos deste caminho produzem $\hat{\mathbf{e}} = [10\ 10\ 00]$ como estimativa da sequência de erro de

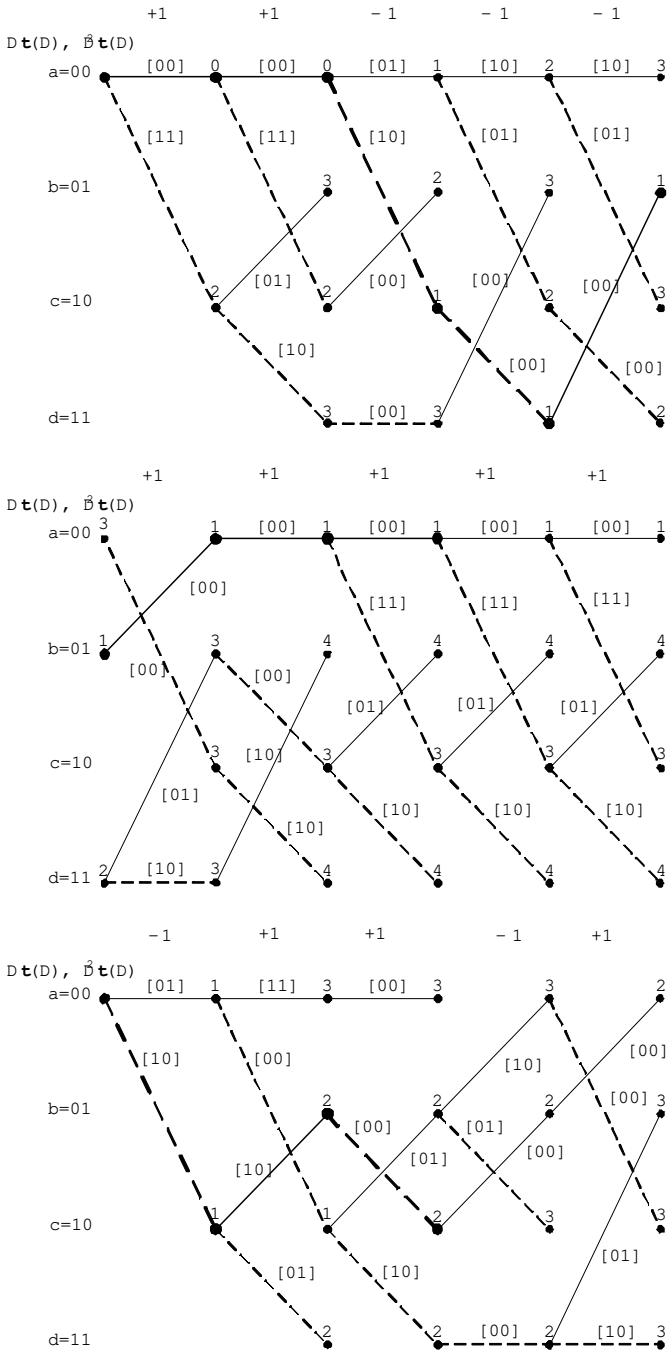

Figura 4.8: Aplicação do ADS para o conjunto de autovalores $\{\alpha_{M_{X,t}}\}$ (primeira e segunda treliças) e $\{\alpha_{M_{Z,t}}\}$ (terceira treliça).

fase para o código phase flip associado (4.19), ou seja, o ADS detectou erros de fase no

primeiro e terceiro blocos. Com um pouco de álgebra, é possível mostrar que estes dois erros de fase no código phase-flip vão se propagar para um erro de fase sobre o quinto qubit do código bit-phase flip da Tabela 4.14. Para corrigirmos o quinto qubit deste código dos erros de bit e de fase, basta aplicarmos, respectivamente, os operadores X_5 e Z_5 sobre a palavra-código corrompida. Portanto, fomos capazes de detectar e corrigir um erro de bit e um erro de fase atuando sobre o quinto qubit. O mesmo poderá ser feito com qualquer um dos outros quinze qubits. Isto nos leva a concluir que também seremos capazes de corrigir um erro geral sobre qualquer um dos dezesseis qubits.

Se o resultado das medidas dos cinco observáveis X fosse, por exemplo, o conjunto de autovalores $(-1,-1,-1,+1,+1)$, o ADS detectaria um erro de fase no primeiro bloco do código phase flip. No código bit-phase flip, este erro de fase poderia se propagar tanto para o primeiro quanto para o segundo qubit. Ficaríamos em dúvida em saber qual dos dois qubits realmente sofreu um erro de fase. Ou seja, teríamos um erro de fase degenerado. Para corrigi-lo, bastaria aplicarmos Z_1 ou Z_2.

Exemplo 14. *Suponha agora que desejamos codificar dois registros quânticos, ou seja, um sistema quântico composto por dois qubits. Nossa operação de codificação realiza-se através da codificação pelo CCC (4, 2, 1) de cada uma das duzentas e cinquenta e seis sequências de oito bits do CCQ para o canal phase flip (4.23). Veja o código gerado por esta operação na Tabela 4.15, que é exibida no final deste capítulo.*

Para melhor entendermos a dinâmica desta operação de codificação: para cada um dos estados da base do sistema de dois qubits foi gerada uma superposição de quatro estados de comprimento quatro no estágio 0, uma superposição de dezesseis estados de comprimento oito no estágio 1, uma superposição de sessenta e quatro estados de comprimento doze no estágio 2, uma superposição de duzentos e cinquenta e seis estados de comprimento dezesseis no estágio 3 e finalmente uma superposição de duzentos e cinquenta e seis estados de comprimento vinte no estágio 4.

Considerando que o estado inicial $|\psi\rangle = a_1a_2|00\rangle + a_1b_2|01\rangle + b_1a_2|10\rangle + b_1b_2|11\rangle$ *tenha sido perfeitamente codificado, podemos garantir a correção de um erro quântico geral em qualquer um dos vinte qubits das palavras-código.*

Os dezoito geradores deste CCQ podem ser obtidos a partir da matriz verificação de

paridade $\mathbf{H}_C = [\mathbf{H}_X; \mathbf{H}_Z]$ *(4.31) truncada:*

$$\mathbf{H}_C = \left[\begin{array}{ccccccccccccccc}
1100 & & & & & & & & & & & & & & \\
1011 & & & & & & & & & & & & & & \\
1110 & 1100 & & & & & & & & & & & & & \\
0011 & 1011 & & & & & & & & & & & & & \\
 & 1110 & 1100 & & & & & & & & & & & & \\
 & 0011 & 1011 & & & & & & & & & & & & \\
 & & 1110 & 1100 & & & & & & & & & & & \\
 & & 0011 & 1011 & & & & & & & & & & & \\
 & & & 1110 & 1100 & & & & & & & & & & \\
 & & & 0011 & 1011 & & & & & & & & & & \\
 & & & & 1110 & 1100 & & & & & & & & & \\
 & & & & 0011 & 1011 & & & & & & & & & \\
 & & & & & 1110 & & & & & & & & & \\
 & & & & & 0011 & & & & & & & & & \\
1110 & 1011 & & & & & & & & & & & & & \\
1101 & 0010 & 1011 & & & & & & & & & & & & \\
1110 & 0110 & 0010 & 1011 & & & & & & & & & & & \\
 & 1110 & 0110 & 0010 & 1011 & & & & & & & & & & \\
 & & 1110 & 0110 & 1100 & & & & & & & & & & \\
 & & & 1110 & 1011 & & & & & & & & & &
\end{array}\right]. \tag{4.36}$$

Repare que a matriz (4.36) é uma matriz verificação de paridade da matriz geradora (4.30) truncada:

$$\mathbf{G}_C = \left[\begin{array}{ccccccccccccccc}
1110 & 1000 & 1001 & 1011 & & & & & & & & & & & \\
 & 1110 & 1000 & 1001 & 1011 & & & & & & & & & &
\end{array}\right]. \tag{4.37}$$

As linhas das matrizes (4.36) e (4.37) fornecem-nos, respectivamente, os geradores do estabilizador e as operações lógicas $\overline{X}_1$, $\overline{Z}_1$, $\overline{X}_2$ *e* $\overline{Z}_2$ *do CCQ. Veja a Tabela 4.12.*

É fácil de verificar que cada um dos geradores da Tabela 4.12 satisfaz a relação $M_t|\psi\rangle = |\psi\rangle$*, onde* $|\psi\rangle$ *é palavra-código da Tabela 4.15. Observe também que as seguintes relações são satisfeitas pelos operadores lógicos* $\overline{X}_1$, $\overline{Z}_1$, $\overline{X}_2$ *e* $\overline{Z}_2$*:*

M_0	Z	Z	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
M_1	Z	I	Z	Z	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
M_2	Z	Z	Z	I	Z	Z	I	I	I	I	I	I	I	I	I	I	I	I	I
M_3	I	I	Z	Z	Z	I	Z	Z	I	I	I	I	I	I	I	I	I	I	I
M_4	I	I	I	I	Z	Z	Z	I	Z	Z	I	I	I	I	I	I	I	I	I
M_5	I	I	I	I	I	I	Z	Z	Z	I	Z	Z	I	I	I	I	I	I	I
M_6	I	I	I	I	I	I	I	I	Z	Z	Z	I	Z	Z	I	I	I	I	I
M_7	I	I	I	I	I	I	I	I	I	I	Z	Z	Z	I	Z	Z	I	I	I
M_8	I	I	I	I	I	I	I	I	I	I	I	I	Z	Z	Z	I	Z	Z	I
M_9	I	I	I	I	I	I	I	I	I	I	I	I	I	I	Z	Z	Z	I	Z
M_{10}	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	Z	Z	Z
M_{11}	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	Z	Z
M_{12}	X	X	X	I	X	I	X	X	I	I	I	I	I	I	I	I	I	I	I
M_{13}	X	X	I	X	I	I	X	I	X	I	X	X	I	I	I	I	I	I	I
M_{14}	X	X	X	I	I	X	X	I	I	I	X	I	X	I	X	X	I	I	I
M_{15}	I	I	I	I	X	X	X	I	I	X	X	I	I	I	X	I	X	I	X
M_{16}	I	I	I	I	I	I	I	I	X	X	X	I	I	X	X	I	X	X	I
M_{17}	I	I	I	I	I	I	I	I	I	I	I	I	X	X	X	I	X	I	X
$\overline{X}_1$	Z	Z	Z	I	Z	I	I	I	Z	I	I	Z	Z	I	Z	Z	I	I	I
$\overline{Z}_1$	X	X	X	I	X	I	I	I	X	I	I	X	X	I	X	X	I	I	I
$\overline{X}_2$	I	I	I	I	Z	Z	Z	I	Z	I	I	I	Z	I	I	Z	Z	I	Z
$\overline{Z}_2$	I	I	I	I	X	X	X	I	X	I	I	I	X	I	I	X	X	I	X

Tabela 4.12: Geradores e operações lógicas do CCQ [4, 1, 3] descrito pela operação de codificação (4.29) para dois qubits.

$$\overline{X}_1|00_L\rangle = |10_L\rangle \quad ; \quad \overline{X}_2|00_L\rangle = |01_L\rangle$$
$$\overline{Z}_1|00_L\rangle = |00_L\rangle \quad ; \quad \overline{Z}_2|00_L\rangle = |00_L\rangle$$
$$\overline{X}_1|01_L\rangle = |11_L\rangle \quad ; \quad \overline{X}_2|01_L\rangle = |00_L\rangle$$
$$\overline{Z}_1|01_L\rangle = |01_L\rangle \quad ; \quad \overline{Z}_2|01_L\rangle = -|01_L\rangle$$
$$\overline{X}_1|10_L\rangle = |00_L\rangle \quad ; \quad \overline{X}_2|10_L\rangle = |11_L\rangle$$
$$\overline{Z}_1|10_L\rangle = -|10_L\rangle \quad ; \quad \overline{Z}_2|10_L\rangle = |10_L\rangle$$
$$\overline{X}_1|11_L\rangle = |01_L\rangle \quad ; \quad \overline{X}_2|11_L\rangle = |10_L\rangle$$
$$\overline{Z}_1|11_L\rangle = -|11_L\rangle \quad ; \quad \overline{Z}_2|11_L\rangle = -|11_L\rangle$$

Usamos os dezoito observáveis da Tabela 4.12 para realizar medidas da palavra-código recebida à saída do canal. Os resultados destas medidas permitem-nos detectar sem ambiguidade um erro geral que eventualmente possa ter ocorrido em qualquer um dos vinte qubits da palavra-código original.

Por exemplo, suponha que o resultado das medidas dos doze observáveis Z seja dado pelo conjunto de autovalores $(+1, -1, -1, -1, +1, +1, +1, +1, +1, +1, +1, +1)$ *e o resultado das medidas dos cinco observáveis X seja dado pelo conjunto de autovalores* $(-1, +1, -1, +1, +1, +1)$. *Os resultados da aplicação do ADS para estes dois conjunto de autovalores são mostrados na Figura 4.9.*

O caminho de menor peso das duas primeiras treliças da Figura 4.9 é o caminho $a \rightarrow a \rightarrow c \rightarrow d \rightarrow b \rightarrow a \rightarrow a \rightarrow a \rightarrow a \rightarrow a \rightarrow a \rightarrow a$. *Os ramos deste caminho produzem* $\hat{\mathbf{e}} = [00\ 10\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00]$ *como estimativa da sequência de erro de bit sobre a palavra-código da Tabela 4.15, ou seja, o ADS detectou um erro de bit no terceiro qubit. O caminho de menor peso da terceira treliça da Figura 4.9 é o caminho* $a \rightarrow a \rightarrow c \rightarrow b \rightarrow a \rightarrow a \rightarrow a$. *Os ramos deste caminho produzem* $\hat{\mathbf{e}} = [01\ 00\ 00\ 00]$ *como estimativa da sequência de erro de fase para o código phase flip (4.23), ou seja, o ADS detectou um erro de fase no segundo bloco. Com um pouco de álgebra, é possível mostrar que este erro de fase no código phase flip vai se propagar para um erro de fase sobre o terceiro qubit do código bit-phase flip da Tabela 4.15. Para corrigirmos o terceiro qubit deste código dos erros de bit e de fase, basta aplicarmos, respectivamente, os operadores* X_3 *e* Z_3 *sobre a palavra-código corrompida. Portanto, fomos capazes de detectar e corrigir um erro de bit e um erro de fase atuando sobre o terceiro qubit. O mesmo poderá ser feito com qualquer um dos outros dezenove qubits. Isto nos leva a concluir que o código também será capaz de corrigir um erro geral sobre qualquer um dos vinte qubits.*

Se o resultado das medidas dos seis observáveis X fosse, por exemplo, o conjunto de autovalores $(+1,+1,+1,-1,+1,-1)$, *o ADS detectaria um erro de fase no oitavo bloco do código phase flip. No código bit-phase flip, este erro de fase poderia se propagar tanto para o décimo nono quanto para o vigésimo qubit. Ficaríamos em dúvida em saber qual dos dois qubits realmente sofreu um erro de fase. Ou seja, teríamos um erro de fase degenerado. Para corrigi-lo, bastaria aplicarmos* Z_{19} *ou* Z_{20}.

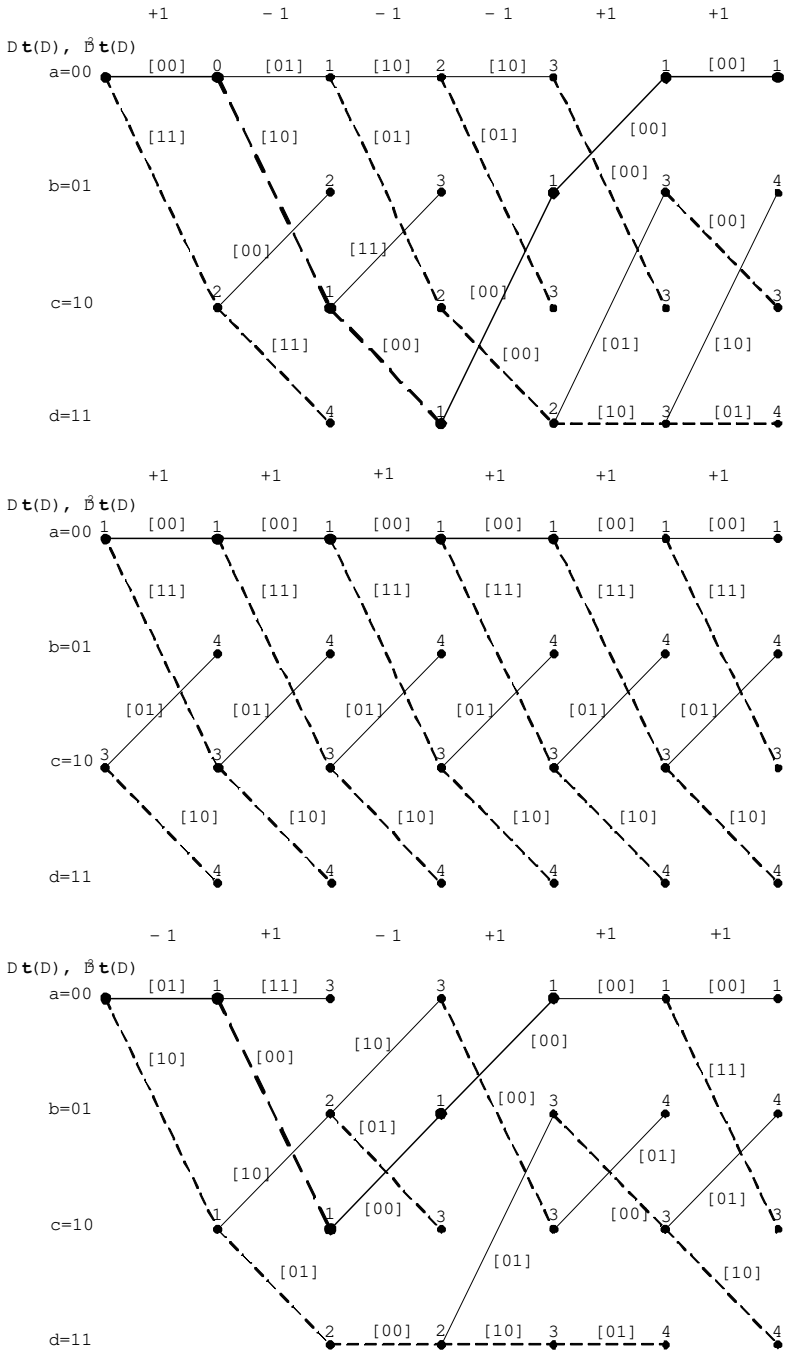

Figura 4.9: Aplicação do ADS para o conjunto de autovalores $\{\alpha_{M_{X,t}}\}$ (primeira e segunda treliças) e $\{\alpha_{M_{Z,t}}\}$ (terceira treliça).

Podemos continuar com este procedimento sistemático para codificar quantos qubits

desejarmos. Se quisermos codificar N qubits de informação, cada um dos 2^N estados da base será uma superposição de $2^{2(N+2)}$ estados com $2(2(N+2)+2) = 4N+12$ qubits cada um. Vimos, através da dinâmica da operação de codificação dos exemplos acima, que o código phase flip é responsável pelo número de estados na superposição do código e o código bit flip é responsável pelo número de qubits dentro de cada um destes estados da superposição. Em ambos os casos, o crescimento é exponencial com o número de transições de tempo. O comprimento do código cresce uma unidade de tempo além do número de estados, porque o código bit flip é o segundo da ordem de concatenação e possui memória unitária.

Estamos agora em condições de comparar a complexidade do CCQ apresentado nesta seção, com a complexidade de código de bloco de nove qubits de Shor, ambos códigos quânticos concatenados. Se usarmos o código de Shor para codificar uma sequência de N qubits de informação, cada um dos 2^N estados da base será uma superposição de 2^{3N} estados de comprimento $9N$. Portanto, quando comparamos com o código de Shor, o CCQ [4, 1, 3] já tem comprimento menor para $N = 3$ e menos estados em sua superposição para $N = 5$. Além disso, a diferença de complexidade entre estes dois códigos concatenados cresce muito rapidamente mesmo para valores moderados de N. Veja a Tabela 4.13. Todas as evidências indicam que, em termos de complexidade, o CCQ [4, 1, 3] também é superior quando comparado com outros CBQs.

Devido ao *paralelismo* quântico, o estudo da complexidade em termos do comprimento do código quântico é o que realmente importa para a construção de sistemas de codificação, transmissão, decodificação e armazenamento de informação quântica. Porém, o estudo da complexidade em termos do número de estados na superposição do código quântico é importante do ponto de vista algébrico, pois está relacionado ao *particionamento* da geometria do código em classes laterais.

A propósito, qual seria a geometria para o CCQ [4, 1, 3] ?

n° de qubits	CBQ Shor	CCQ [4, 1, 3]
1	(8)[9]	(64)[16]
2	(64)[18]	(256)[20]
3	(512)[27]	(1024)[24]
4	(4096)[36]	(4096)[28]
5	(32768)[45]	(16384)[32]
⋮	⋮	⋮

Tabela 4.13: Comparação entre a complexidade do código de Shor e o CCQ [4, 1, 3]: (número de estados na superposição)[comprimento do estado].

++	$\|0000\ 0000\ 0000\ 0000\rangle$	+−	$\|1101\ 1100\ 0000\ 0000\rangle$
+−	$\|0000\ 0000\ 0011\ 0111\rangle$	++	$\|1101\ 1100\ 0011\ 0111\rangle$
+−	$\|0000\ 0000\ 1101\ 1100\rangle$	++	$\|1101\ 1100\ 1101\ 1100\rangle$
++	$\|0000\ 0000\ 1110\ 1011\rangle$	+−	$\|1101\ 1100\ 1110\ 1011\rangle$
+−	$\|0000\ 0011\ 0111\ 0000\rangle$	++	$\|1101\ 1111\ 0111\ 0000\rangle$
++	$\|0000\ 0011\ 0100\ 0111\rangle$	+−	$\|1101\ 1111\ 0100\ 0111\rangle$
++	$\|0000\ 0011\ 1010\ 1100\rangle$	+−	$\|1101\ 1111\ 1010\ 1100\rangle$
+−	$\|0000\ 0011\ 1001\ 1011\rangle$	++	$\|1101\ 1111\ 1001\ 1011\rangle$
++	$\|0000\ 1101\ 1100\ 0000\rangle$	+−	$\|1101\ 0001\ 1100\ 0000\rangle$
+−	$\|0000\ 1101\ 1111\ 0111\rangle$	++	$\|1101\ 0001\ 1111\ 0111\rangle$
+−	$\|0000\ 1101\ 0001\ 1100\rangle$	++	$\|1101\ 0001\ 0001\ 1100\rangle$
++	$\|0000\ 1101\ 0010\ 1011\rangle$	+−	$\|1101\ 0001\ 0010\ 1011\rangle$
+−	$\|0000\ 1110\ 1011\ 0000\rangle$	++	$\|1101\ 0010\ 1011\ 0000\rangle$
++	$\|0000\ 1110\ 1000\ 0111\rangle$	+−	$\|1101\ 0010\ 1000\ 0111\rangle$
++	$\|0000\ 1110\ 0110\ 1100\rangle$	+−	$\|1101\ 0010\ 0110\ 1100\rangle$
+−	$\|0000\ 1110\ 0101\ 1011\rangle$	++	$\|1101\ 0010\ 0101\ 1011\rangle$
+−	$\|0011\ 0111\ 0000\ 0000\rangle$	++	$\|1110\ 1011\ 0000\ 0000\rangle$
++	$\|0011\ 0111\ 0011\ 0111\rangle$	+−	$\|1110\ 1011\ 0011\ 0111\rangle$
++	$\|0011\ 0111\ 1101\ 1100\rangle$	+−	$\|1110\ 1011\ 1101\ 1100\rangle$
+−	$\|0011\ 0111\ 1110\ 1011\rangle$	++	$\|1110\ 1011\ 1110\ 1011\rangle$
++	$\|0011\ 0100\ 0111\ 0000\rangle$	+−	$\|1110\ 1000\ 0111\ 0000\rangle$
+−	$\|0011\ 0100\ 0100\ 0111\rangle$	++	$\|1110\ 1000\ 0100\ 0111\rangle$
+−	$\|0011\ 0100\ 1010\ 1100\rangle$	++	$\|1110\ 1000\ 1010\ 1100\rangle$
++	$\|0011\ 0100\ 1001\ 1011\rangle$	+−	$\|1110\ 1000\ 1001\ 1011\rangle$
+−	$\|0011\ 1010\ 1100\ 0000\rangle$	++	$\|1110\ 0110\ 1100\ 0000\rangle$
++	$\|0011\ 1010\ 1111\ 0111\rangle$	+−	$\|1110\ 0110\ 1111\ 0111\rangle$
++	$\|0011\ 1010\ 0001\ 1100\rangle$	+−	$\|1110\ 0110\ 0001\ 1100\rangle$
+−	$\|0011\ 1010\ 0010\ 1011\rangle$	++	$\|1110\ 0110\ 0010\ 1011\rangle$
++	$\|0011\ 1001\ 1011\ 0000\rangle$	+−	$\|1110\ 0101\ 1011\ 0000\rangle$
+−	$\|0011\ 1001\ 1000\ 0111\rangle$	++	$\|1110\ 0101\ 1000\ 0111\rangle$
+−	$\|0011\ 1001\ 0110\ 1100\rangle$	++	$\|1110\ 0101\ 0110\ 1100\rangle$
++	$\|0011\ 1001\ 0101\ 1011\rangle$	+−	$\|1110\ 0101\ 0101\ 1011\rangle$

Tabela 4.14: CCQ para um qubit em canal quântico com erro geral. Cada estado da base do qubit é codificado em uma superposição de sessenta e quatro estados, cada um com dezesseis qubits. A primeira coluna de sinais refere-se aos sinais que estão à frente dos estados da superposição de $|0_L\rangle$. A segunda coluna de sinais refere-se aos sinais que estão à frente dos estados da superposição de $|1_L\rangle$.

Tabela 4.15: CCQ para dois qubits em canal quântico com erro geral. Cada estado da base do sistema de dois qubits é codificado em uma superposição de duzentos e cinquenta e seis estados, cada um com vinte qubits. A primeira coluna de sinais refere-se aos sinais que estão à frente dos estados da superposição de $|00_L\rangle$. A segunda coluna de sinais refere-se aos sinais que estão à frente dos estados da superposição de $|01_L\rangle$. A terceira coluna de sinais refere-se aos sinais que estão à frente dos estados da superposição de $|10_L\rangle$. A quarta coluna de sinais refere-se aos sinais que estão à frente dos estados da superposição de $|11_L\rangle$.

$++++$	$\|0000\ 0000\ 0000\ 0000\ 0000\rangle$	$++--$	$\|1101\ 1100\ 0000\ 0000\ 0000\rangle$
$+-+-$	$\|0000\ 0000\ 0000\ 0011\ 0111\rangle$	$+--+$	$\|1101\ 1100\ 0000\ 0011\ 0111\rangle$
$+-+-$	$\|0000\ 0000\ 0000\ 1101\ 1100\rangle$	$+--+$	$\|1101\ 1100\ 0000\ 1101\ 1100\rangle$
$++++$	$\|0000\ 0000\ 0000\ 1110\ 1011\rangle$	$++--$	$\|1101\ 1100\ 0000\ 1110\ 1011\rangle$
$+--+$	$\|0000\ 0000\ 0011\ 0111\ 0000\rangle$	$+-+-$	$\|1101\ 1100\ 0011\ 0111\ 0000\rangle$
$++--$	$\|0000\ 0000\ 0011\ 0100\ 0111\rangle$	$++++$	$\|1101\ 1100\ 0011\ 0100\ 0111\rangle$
$++--$	$\|0000\ 0000\ 0011\ 1010\ 1100\rangle$	$++++$	$\|1101\ 1100\ 0011\ 1010\ 1100\rangle$
$+--+$	$\|0000\ 0000\ 0011\ 1001\ 1011\rangle$	$+-+-$	$\|1001\ 1100\ 0011\ 1001\ 1011\rangle$
$++--$	$\|0000\ 0000\ 1101\ 1100\ 0000\rangle$	$++++$	$\|1101\ 1100\ 1101\ 1100\ 0000\rangle$
$+--+$	$\|0000\ 0000\ 1101\ 1111\ 0111\rangle$	$+-+-$	$\|1101\ 1100\ 1101\ 1111\ 0111\rangle$
$+--+$	$\|0000\ 0000\ 1101\ 0001\ 1100\rangle$	$+-+-$	$\|1101\ 1100\ 1101\ 0001\ 1100\rangle$
$++--$	$\|0000\ 0000\ 1101\ 0010\ 1011\rangle$	$++++$	$\|1101\ 1100\ 1101\ 0010\ 1011\rangle$
$+-+-$	$\|0000\ 0000\ 1110\ 1011\ 0000\rangle$	$+--+$	$\|1101\ 1100\ 1110\ 1011\ 0000\rangle$
$++++$	$\|0000\ 0000\ 1110\ 1000\ 0111\rangle$	$++--$	$\|1101\ 1100\ 1110\ 1000\ 0111\rangle$
$++++$	$\|0000\ 0000\ 1110\ 0110\ 1100\rangle$	$++--$	$\|1101\ 1100\ 1110\ 0110\ 1100\rangle$
$+-+-$	$\|0000\ 0000\ 1110\ 0101\ 1011\rangle$	$+--+$	$\|1101\ 1100\ 1110\ 0101\ 1011\rangle$
$+--+$	$\|0000\ 0011\ 0111\ 0000\ 0000\rangle$	$+-+-$	$\|1101\ 1111\ 0111\ 0000\ 0000\rangle$
$++--$	$\|0000\ 0011\ 0111\ 0011\ 0111\rangle$	$++++$	$\|1101\ 1111\ 0111\ 0011\ 0111\rangle$
$++--$	$\|0000\ 0011\ 0111\ 1101\ 1100\rangle$	$++++$	$\|1101\ 1111\ 0111\ 1101\ 1100\rangle$
$+--+$	$\|0000\ 0011\ 0111\ 1110\ 1011\rangle$	$+-+-$	$\|1101\ 1111\ 0111\ 1110\ 1011\rangle$
$++++$	$\|0000\ 0011\ 0100\ 0111\ 0000\rangle$	$++--$	$\|1101\ 1111\ 0100\ 0111\ 0000\rangle$
$+-+-$	$\|0000\ 0011\ 0100\ 0100\ 0111\rangle$	$+--+$	$\|1101\ 1111\ 0100\ 0100\ 0111\rangle$
$+-+-$	$\|0000\ 0011\ 0100\ 1010\ 1100\rangle$	$+--+$	$\|1101\ 1111\ 0100\ 1010\ 1100\rangle$
$++++$	$\|0000\ 0011\ 0100\ 1001\ 1011\rangle$	$++--$	$\|1101\ 1111\ 0100\ 1001\ 1011\rangle$

$+-+-$	$\|0000\ 0011\ 1010\ 1100\ 0000\rangle$	$+--+$	$\|1101\ 1111\ 1010\ 1100\ 0000\rangle$
$++++$	$\|0000\ 0011\ 1010\ 1111\ 0111\rangle$	$++--$	$\|1101\ 1111\ 1010\ 1111\ 0111\rangle$
$++++$	$\|0000\ 0011\ 1010\ 0001\ 1100\rangle$	$++--$	$\|1101\ 1111\ 1010\ 0001\ 1100\rangle$
$+-+-$	$\|0000\ 0011\ 1010\ 0010\ 1011\rangle$	$+--+$	$\|1101\ 1111\ 1010\ 0010\ 1011\rangle$
$++--$	$\|0000\ 0011\ 1001\ 1011\ 0000\rangle$	$++++$	$\|1101\ 1111\ 1001\ 1011\ 0000\rangle$
$+--+$	$\|0000\ 0011\ 1001\ 1000\ 0111\rangle$	$+-+-$	$\|1101\ 1111\ 1001\ 1000\ 0111\rangle$
$+--+$	$\|0000\ 0011\ 1001\ 0110\ 1100\rangle$	$+-+-$	$\|1101\ 1111\ 1001\ 0110\ 1100\rangle$
$++--$	$\|0000\ 0011\ 1001\ 0101\ 1011\rangle$	$++++$	$\|1101\ 1111\ 1001\ 0101\ 1011\rangle$
$+-+-$	$\|0000\ 1101\ 1100\ 0000\ 0000\rangle$	$+--+$	$\|1101\ 0001\ 1100\ 0000\ 0000\rangle$
$++++$	$\|0000\ 1101\ 1100\ 0011\ 0111\rangle$	$++--$	$\|1101\ 0001\ 1100\ 0011\ 0111\rangle$
$++++$	$\|0000\ 1101\ 1100\ 1101\ 1100\rangle$	$++--$	$\|1101\ 0001\ 1100\ 1101\ 1100\rangle$
$+-+-$	$\|0000\ 1101\ 1100\ 1110\ 1011\rangle$	$+--+$	$\|1101\ 0001\ 1100\ 1110\ 1011\rangle$
$++--$	$\|0000\ 1101\ 1111\ 0111\ 0000\rangle$	$++++$	$\|1101\ 0001\ 1111\ 0111\ 0000\rangle$
$+--+$	$\|0000\ 1101\ 1111\ 0100\ 0111\rangle$	$+-+-$	$\|1101\ 0001\ 1111\ 0100\ 0111\rangle$
$+--+$	$\|0000\ 1101\ 1111\ 1010\ 1100\rangle$	$+-+-$	$\|1101\ 0001\ 1111\ 1010\ 1100\rangle$
$++--$	$\|0000\ 1101\ 1111\ 1001\ 1011\rangle$	$++++$	$\|1001\ 0001\ 1111\ 1001\ 1011\rangle$
$+--+$	$\|0000\ 1101\ 0001\ 1100\ 0000\rangle$	$+-+-$	$\|1101\ 0001\ 0001\ 1100\ 0000\rangle$
$++--$	$\|0000\ 1101\ 0001\ 1111\ 0111\rangle$	$++++$	$\|1101\ 0001\ 0001\ 1111\ 0111\rangle$
$++--$	$\|0000\ 1101\ 0001\ 0001\ 1100\rangle$	$++++$	$\|1101\ 0001\ 0001\ 0001\ 1100\rangle$
$+--+$	$\|0000\ 1101\ 0001\ 0010\ 1011\rangle$	$+-+-$	$\|1101\ 0001\ 0001\ 0010\ 1011\rangle$
$++++$	$\|0000\ 1101\ 0010\ 1011\ 0000\rangle$	$++--$	$\|1101\ 0001\ 0010\ 1011\ 0000\rangle$
$+-+-$	$\|0000\ 1101\ 0010\ 1000\ 0111\rangle$	$+--+$	$\|1101\ 0001\ 0010\ 1000\ 0111\rangle$
$+-+-$	$\|0000\ 1101\ 0010\ 0110\ 1100\rangle$	$+--+$	$\|1101\ 0001\ 0010\ 0110\ 1100\rangle$
$++++$	$\|0000\ 1101\ 0010\ 0101\ 1011\rangle$	$++--$	$\|1101\ 0001\ 0010\ 0101\ 1011\rangle$
$++--$	$\|0000\ 1110\ 1011\ 0000\ 0000\rangle$	$++++$	$\|1101\ 0010\ 1011\ 0000\ 0000\rangle$
$+--+$	$\|0000\ 1110\ 1011\ 0011\ 0111\rangle$	$+-+-$	$\|1101\ 0010\ 1011\ 0011\ 0111\rangle$
$+--+$	$\|0000\ 1110\ 1011\ 1101\ 1100\rangle$	$+-+-$	$\|1101\ 0010\ 1011\ 1101\ 1100\rangle$
$++--$	$\|0000\ 1110\ 1011\ 1110\ 1011\rangle$	$++++$	$\|1101\ 0010\ 1011\ 1110\ 1011\rangle$
$+-+-$	$\|0000\ 1110\ 1000\ 0111\ 0000\rangle$	$+--+$	$\|1101\ 0010\ 1000\ 0111\ 0000\rangle$
$++++$	$\|0000\ 1110\ 1000\ 0100\ 0111\rangle$	$++--$	$\|1101\ 0010\ 1000\ 0100\ 0111\rangle$
$++++$	$\|0000\ 1110\ 1000\ 1010\ 1100\rangle$	$++--$	$\|1101\ 0010\ 1000\ 1010\ 1100\rangle$
$+-+-$	$\|0000\ 1110\ 1000\ 1001\ 1011\rangle$	$+--+$	$\|1101\ 0010\ 1000\ 1001\ 1011\rangle$
$++++$	$\|0000\ 1110\ 0110\ 1100\ 0000\rangle$	$++--$	$\|1101\ 0010\ 0110\ 1100\ 0000\rangle$
$+-+-$	$\|0000\ 1110\ 0110\ 1111\ 0111\rangle$	$+--+$	$\|1101\ 0010\ 0110\ 1111\ 0111\rangle$
$+-+-$	$\|0000\ 1110\ 0110\ 0001\ 1100\rangle$	$+--+$	$\|1101\ 0010\ 0110\ 0001\ 1100\rangle$
$++++$	$\|0000\ 1110\ 0110\ 0010\ 1011\rangle$	$++--$	$\|1101\ 0010\ 0110\ 0010\ 1011\rangle$

$+--+$	$\|0000\ 1110\ 0101\ 1011\ 0000\rangle$	$+-+-$	$\|1101\ 0010\ 0101\ 1011\ 0000\rangle$
$++--$	$\|0000\ 1110\ 0101\ 1000\ 0111\rangle$	$++++$	$\|1101\ 0010\ 0101\ 1000\ 0111\rangle$
$++--$	$\|0000\ 1110\ 0101\ 0110\ 1100\rangle$	$++++$	$\|1101\ 0010\ 0101\ 0110\ 1100\rangle$
$+--+$	$\|0000\ 1110\ 0101\ 0101\ 1011\rangle$	$+-+-$	$\|1101\ 0010\ 0101\ 0101\ 1011\rangle$
$++--$	$\|0011\ 0111\ 0000\ 0000\ 0000\rangle$	$++++$	$\|1110\ 1011\ 0000\ 0000\ 0000\rangle$
$+--+$	$\|0011\ 0111\ 0000\ 0011\ 0111\rangle$	$+-+-$	$\|1110\ 1011\ 0000\ 0011\ 0111\rangle$
$+--+$	$\|0011\ 0111\ 0000\ 1101\ 1100\rangle$	$+-+-$	$\|1110\ 1011\ 0000\ 1101\ 1100\rangle$
$++--$	$\|0011\ 0111\ 0000\ 1110\ 1011\rangle$	$++++$	$\|1110\ 1011\ 0000\ 1110\ 1011\rangle$
$+-+-$	$\|0011\ 0111\ 0011\ 0111\ 0000\rangle$	$+--+$	$\|1110\ 1011\ 0011\ 0111\ 0000\rangle$
$++++$	$\|0011\ 0111\ 0011\ 0100\ 0111\rangle$	$++--$	$\|1110\ 1011\ 0011\ 0100\ 0111\rangle$
$++++$	$\|0011\ 0111\ 0011\ 1010\ 1100\rangle$	$++--$	$\|1110\ 1011\ 0011\ 1010\ 1100\rangle$
$+-+-$	$\|0011\ 0111\ 0011\ 1001\ 1011\rangle$	$+--+$	$\|1110\ 1011\ 0011\ 1001\ 1011\rangle$
$++++$	$\|0011\ 0111\ 1101\ 1100\ 0000\rangle$	$++--$	$\|1110\ 1011\ 1101\ 1100\ 0000\rangle$
$+-+-$	$\|0011\ 0111\ 1101\ 1111\ 0111\rangle$	$+--+$	$\|1110\ 1011\ 1101\ 1111\ 0111\rangle$
$+-+-$	$\|0011\ 0111\ 1101\ 0001\ 1100\rangle$	$+--+$	$\|1110\ 1011\ 1101\ 0001\ 1100\rangle$
$++++$	$\|0011\ 0111\ 1101\ 0010\ 1011\rangle$	$++--$	$\|1110\ 1011\ 1101\ 0010\ 1011\rangle$
$+--+$	$\|0011\ 0111\ 1110\ 1011\ 0000\rangle$	$+-+-$	$\|1110\ 1011\ 1110\ 1011\ 0000\rangle$
$++--$	$\|0011\ 0111\ 1110\ 1000\ 0111\rangle$	$++++$	$\|1110\ 1011\ 1110\ 1000\ 0111\rangle$
$++--$	$\|0011\ 0111\ 1110\ 0110\ 1100\rangle$	$++++$	$\|1110\ 1011\ 1110\ 0110\ 1100\rangle$
$+--+$	$\|0011\ 0111\ 1110\ 0101\ 1011\rangle$	$+-+-$	$\|1110\ 1011\ 1110\ 0101\ 1011\rangle$
$+-+-$	$\|0011\ 0100\ 0111\ 0000\ 0000\rangle$	$+--+$	$\|1110\ 1000\ 0111\ 0000\ 0000\rangle$
$++++$	$\|0011\ 0100\ 0111\ 0011\ 0111\rangle$	$++--$	$\|1110\ 1000\ 0111\ 0011\ 0111\rangle$
$++++$	$\|0011\ 0100\ 0111\ 1101\ 1100\rangle$	$++--$	$\|1110\ 1000\ 0111\ 1101\ 1100\rangle$
$+-+-$	$\|0011\ 0100\ 0111\ 1110\ 1011\rangle$	$+--+$	$\|1110\ 1000\ 0111\ 1110\ 1011\rangle$
$++--$	$\|0011\ 0100\ 0100\ 0111\ 0000\rangle$	$++++$	$\|1110\ 1000\ 0100\ 0111\ 0000\rangle$
$+--+$	$\|0011\ 0100\ 0100\ 0100\ 0111\rangle$	$+-+-$	$\|1110\ 1000\ 0100\ 0100\ 0111\rangle$
$+--+$	$\|0011\ 0100\ 0100\ 1010\ 1100\rangle$	$+-+-$	$\|1110\ 1000\ 0100\ 1010\ 1100\rangle$
$++--$	$\|0011\ 0100\ 0100\ 1001\ 1011\rangle$	$++++$	$\|1110\ 1000\ 0100\ 1001\ 1011\rangle$
$+--+$	$\|0011\ 0100\ 1010\ 1100\ 0000\rangle$	$+-+-$	$\|1110\ 1000\ 1010\ 1100\ 0000\rangle$
$++--$	$\|0011\ 0100\ 1010\ 1111\ 0111\rangle$	$++++$	$\|1110\ 1000\ 1010\ 1111\ 0111\rangle$
$++--$	$\|0011\ 0100\ 1010\ 0001\ 1100\rangle$	$++++$	$\|1110\ 1000\ 1010\ 0001\ 1100\rangle$
$+--+$	$\|0011\ 0100\ 1010\ 0010\ 1011\rangle$	$+-+-$	$\|1110\ 1000\ 1010\ 0010\ 1011\rangle$
$++++$	$\|0011\ 0100\ 1001\ 1011\ 0000\rangle$	$++--$	$\|1110\ 1000\ 1001\ 1011\ 0000\rangle$
$+-+-$	$\|0011\ 0100\ 1001\ 1000\ 0111\rangle$	$+--+$	$\|1110\ 1000\ 1001\ 1000\ 0111\rangle$
$+-+-$	$\|0011\ 0100\ 1001\ 0110\ 1100\rangle$	$+--+$	$\|1110\ 1000\ 1001\ 0110\ 1100\rangle$
$++++$	$\|0011\ 0100\ 1001\ 0101\ 1011\rangle$	$++--$	$\|1110\ 1000\ 1001\ 0101\ 1011\rangle$

$+--+$	$\|0011\ 1010\ 1100\ 0000\ 0000\rangle$	$+-+-$	$\|1110\ 0110\ 1100\ 0000\ 0000\rangle$
$++--$	$\|0011\ 1010\ 1100\ 0011\ 0111\rangle$	$+++-$	$\|1110\ 0110\ 1100\ 0011\ 0111\rangle$
$++--$	$\|0011\ 1010\ 1100\ 1101\ 1100\rangle$	$+++-$	$\|1110\ 0110\ 1100\ 1101\ 1100\rangle$
$+--+$	$\|0011\ 1010\ 1100\ 1110\ 1011\rangle$	$+-+-$	$\|1110\ 0110\ 1100\ 1110\ 1011\rangle$
$++++$	$\|0011\ 1010\ 1111\ 0111\ 0000\rangle$	$++--$	$\|1110\ 0110\ 1111\ 0111\ 0000\rangle$
$+-+-$	$\|0011\ 1010\ 1111\ 0100\ 0111\rangle$	$+--+$	$\|1110\ 0110\ 1111\ 0100\ 0111\rangle$
$+-+-$	$\|0011\ 1010\ 1111\ 1010\ 1100\rangle$	$+--+$	$\|1110\ 0110\ 1111\ 1010\ 1100\rangle$
$++++$	$\|0011\ 1010\ 1111\ 1001\ 1011\rangle$	$++--$	$\|1110\ 0110\ 1111\ 1001\ 1011\rangle$
$+-+-$	$\|0011\ 1010\ 0001\ 1100\ 0000\rangle$	$+--+$	$\|1110\ 0110\ 0001\ 1100\ 0000\rangle$
$++++$	$\|0011\ 1010\ 0001\ 1111\ 0111\rangle$	$++--$	$\|1110\ 0110\ 0001\ 1111\ 0111\rangle$
$++++$	$\|0011\ 1010\ 0001\ 0001\ 1100\rangle$	$++--$	$\|1110\ 0110\ 0001\ 0001\ 1100\rangle$
$+-+-$	$\|0011\ 1010\ 0001\ 0010\ 1011\rangle$	$+--+$	$\|1110\ 0110\ 0001\ 0010\ 1011\rangle$
$++--$	$\|0011\ 1010\ 0010\ 1011\ 0000\rangle$	$++++$	$\|1110\ 0110\ 0010\ 1011\ 0000\rangle$
$+--+$	$\|0011\ 1010\ 0010\ 1000\ 0111\rangle$	$+-+-$	$\|1110\ 0110\ 0010\ 1000\ 0111\rangle$
$+--+$	$\|0011\ 1010\ 0010\ 0110\ 1100\rangle$	$+-+-$	$\|1110\ 0110\ 0010\ 0110\ 1100\rangle$
$++--$	$\|0011\ 1010\ 0010\ 0101\ 1011\rangle$	$++++$	$\|1110\ 0110\ 0010\ 0101\ 1011\rangle$
$++++$	$\|0011\ 1001\ 1011\ 0000\ 0000\rangle$	$++--$	$\|1110\ 0101\ 1011\ 0000\ 0000\rangle$
$+-+-$	$\|0011\ 1001\ 1011\ 0011\ 0111\rangle$	$+--+$	$\|1110\ 0101\ 1011\ 0011\ 0111\rangle$
$+-+-$	$\|0011\ 1001\ 1011\ 1101\ 1100\rangle$	$+--+$	$\|1110\ 0101\ 1011\ 1101\ 1100\rangle$
$++++$	$\|0011\ 1001\ 1011\ 1110\ 1011\rangle$	$++--$	$\|1110\ 0101\ 1011\ 1110\ 1011\rangle$
$+--+$	$\|0011\ 1001\ 1000\ 0111\ 0000\rangle$	$+-+-$	$\|1110\ 0101\ 1000\ 0111\ 0000\rangle$
$++--$	$\|0011\ 1001\ 1000\ 0100\ 0111\rangle$	$++++$	$\|1110\ 0101\ 1000\ 0100\ 0111\rangle$
$++--$	$\|0011\ 1001\ 1000\ 1010\ 1100\rangle$	$++++$	$\|1110\ 0101\ 1000\ 1010\ 1100\rangle$
$+--+$	$\|0011\ 1001\ 1000\ 1001\ 1011\rangle$	$+-+-$	$\|1110\ 0101\ 1000\ 1001\ 1011\rangle$
$++--$	$\|0011\ 1001\ 0110\ 1100\ 0000\rangle$	$++++$	$\|1110\ 0101\ 0110\ 1100\ 0000\rangle$
$+--+$	$\|0011\ 1001\ 0110\ 1111\ 0111\rangle$	$+-+-$	$\|1110\ 0101\ 0110\ 1111\ 0111\rangle$
$+--+$	$\|0011\ 1001\ 0110\ 0001\ 1100\rangle$	$+-+-$	$\|1110\ 0101\ 0110\ 0001\ 1100\rangle$
$++--$	$\|0011\ 1001\ 0110\ 0010\ 1011\rangle$	$++++$	$\|1110\ 0101\ 0110\ 0010\ 1011\rangle$
$+++-$	$\|0011\ 1001\ 0101\ 1011\ 0000\rangle$	$+--+$	$\|1110\ 0101\ 0101\ 1011\ 0000\rangle$
$+-++$	$\|0011\ 1001\ 0101\ 1000\ 0111\rangle$	$++--$	$\|1110\ 0101\ 0101\ 1000\ 0111\rangle$
$+-++$	$\|0011\ 1001\ 0101\ 0110\ 1100\rangle$	$++--$	$\|1110\ 0101\ 0101\ 0110\ 1100\rangle$
$+++-$	$\|0011\ 1001\ 0101\ 0101\ 1011\rangle$	$+--+$	$\|1110\ 0101\ 0101\ 0101\ 1011\rangle$

Capítulo 5

Uma Classe de CCQs Concatenados

Our field is still in its embryonic stage. It's great that we haven't been around for 2000 years. We are still at a stage where very, very important results occur in front of our eyes.

– Michael Rabin, on quantum computer science.

No Capítulo 4, apresentamos um CCQ [4, 1, 3] concatenado capaz de corrigir até um erro quântico geral. Neste capítulo, mostraremos algumas outras técnicas de concatenação que nos permitem construir CCQs com outras taxas, memórias e capacidades de correção. A seleção das técnicas e dos códigos baseou-se no princípio da diversidade e da simplicidade do processo de geração e de decodificação.

5.1 As Técnicas de Concatenação

Para construirmos um CCQ capaz de corrigir até t erros quânticos gerais, é necessário que os CCCs associados a $\mathcal{C}_1$ (código phase flip) e $\mathcal{C}_2$ (código bit flip) assegurem a correção de pelo menos t erros ($d_{free} \geq 2t+1$), e que o CCC associado a $\mathcal{C}$ (código bit-phase flip) assegure a correção de pelo menos $4t$ erros ($d_{free} \geq 8t+1$). Esta condição é válida se considerarmos que os erros nos qubits são *independentes* entre si[1].

O número de codificadores na cadeia de concatenação depende exclusivamente do número de geradores do erro quântico geral com "naturezas" distintas. Como estamos in-

[1]Não trataremos aqui do caso em que os erros dos qubits podem estar correlacionados entre si ou do caso em que o canal pode produzir uma "rajada" (*burst*) de erros.

teressados em corrigir apenas erros quânticos gerais com geradores Z e X, são necessários apenas dois codificadores na cadeia de concatenação, um para os erros do tipo Z e outro para os erros do tipo X. A ordem dos codificadores na concatenação é importante. Vimos na seção 2.3.3 que quando o primeiro codificador é destinado à correção de erros X e o segundo codificador é destinado à correção de erros Z, apenas códigos universais são produzidos, os quais são inúteis para assegurar a correção de erros de qualquer natureza.

Ao longo deste capítulo usaremos apenas CCCs não-catastróficos na cadeia de concatenação, pois temos que garantir que os erros Z e X sejam corrigidos localmente. A concatenação de dois CCCs não-catastróficos produz necessariamente um CCC concatenado não-catastrófico. Estando garantida a não-catastroficidade dos CCCs associados a $\mathcal{C}_1$, $\mathcal{C}_2$ e $\mathcal{C}$, podemos garantir que o correspondente CCQ concatenado também será não-catastrófico.

Quando o d_{free} dos CCCs da cadeia de concatenação for excedente para o número de erros quânticos gerais que se pretende corrigir, *algumas* palavras-código com padrões de erros envolvendo mais erros Z e X do que o exigido para a correção daquele número de erros quânticos gerais também poderão ser corrigidas pelo CCQ concatenado.

Para a classe dos CCQs, o maior problema é alcançar a distância do código concatenado, ao contrário da classe dos CBQs, em que o maior problema reside em alcançar a distância dos códigos da cadeia de concatenação. Por exemplo, para o CBQ de taxa $1/16$ gerado a partir do código de repetição com quatro bits, a distância do código concatenado ($d = 16$) é mais do que suficiente para corrigir dois erros, porém os códigos da concatenação possuem apenas distância para corrigir um erro ($d = 4$). Para corrigir dois erros, temos que ir além e usar o CBQ de taxa $1/25$ gerado a partir do código de repetição com cinco bits. Este fato ressalta que a complementariedade existente em alguns aspectos entre a classe dos códigos de bloco e a classe dos códigos convolucionais no domínio clássico reflete-se também no domínio quântico.

Em geral, a concatenação de bons CCCs gera um bom CCC concatenado, não necessariamente o de máximo d_{free} e muito menos o ótimo (ou seja, o de *menor* número de caminhos com máximo d_{free} para aquela taxa e memória). Encontrar uma fórmula para o d_{free} de um CCC concatenado, ou, para ser mais modesto, um limitante inferior ou supe-

rior, é um problema desafiador. Lembremo-nos de que não há nem mesmo uma fórmula para se determinar o d_{free} de um CCC isolado, embora existam alguns limitantes inferiores e superiores[2].

Quando consultamos uma tabela de bons CCCs, percebemos que o d_{free} do CCC equivalente ao CCC concatenado nos dá apenas uma idéia muito grosseira do limitante superior do d_{free} do CCC concatenado. A concatenação tem o efeito de *diminuir* o d_{free} do código. Por exemplo, se formos a uma tabela de bons CCCs, veremos que o CCC (4, 1, 3) tem $d_{free} = 13$. Entretanto, em nossa busca computacional, não conseguimos obter nenhum CCC (4, 1, 3) gerado da concatenação de um CCC (2, 1, 2) com um CCC (4, 2, 1) com $d_{free} > 11$.

É necessário um pouco de experiência prática para perceber que gerar bons CCCs concatenados com baixa complexidade não é uma tarefa tão simples como pode parecer à primeira vista. Veremos mais adiante, que quando transportamos CCCs com muitos registros de entrada, memórias e saídas, a complexidade do processo de geração e de decodificação dos correspondentes CCQs cresce muito rapidamente. Daí a importância de buscarmos formas de gerar bons CCCs concatenados com o mínimo de complexidade.

O objetivo deste capítulo é introduzir algumas técnicas simples de se alcançar um d_{free} alto para o CCC concatenado com o mínimo de complexidade, tanto em termos de número de memórias quanto de taxa. Vimos no Capítulo 4 que o primeiro codificador é responsável pelo número de estados da superposição da palavra-código quântica e que o segundo codificador é responsável pelo comprimento de cada um destes estados da superposição. Dependendo da aplicação prática do código, reduzir a complexidade de pelo menos uma dimensão do código pode ser extremamente valioso.

5.1.1 A Memória Convolucional Quântica

Vimos no Capítulo 4 que o conceito de memória convolucional no domínio quântico é fundamental para entendermos o processo de geração e de decodificação de CCQs. Isto ficará ainda mais explícito com os exemplos apresentados ao longo deste capítulo.

Por convenção, falaremos sempre do número de memórias por *registro de deslocamento*,

[2]Veja a seção 3.4.1.

e não do número total de memórias. Chamaremos o número de memórias do primeiro codificador da cadeia (associado à correção de erros de fase) de *memória convolucional de fase* e o número de memórias do segundo codificador da cadeia (associado à correção de erros de bit) de *memória convolucional de bit*. O número de memórias do codificador concatenado é a *soma* do número de memórias dos codificadores da cadeia de concatenação. Chamaremos o número de memórias do codificador concatenado de *memória convolucional quântica*. Como vimos no Capítulo 4, trata-se do número $T-1$, onde T é o número de transições necessárias para se obter uma palavra-código quântica válida.

O número total de memórias do primeiro e do segundo codificador pode ser diferente. Em geral, buscaremos sempre um segundo codificador com memória unitária completa (MUC) ou com memória unitária parcial (MUP) com o objetivo de diminuirmos a complexidade do código, já que o segundo codificador terá sempre mais de um registro de entrada. Veremos mais adiante que até mesmo a opção por codificadores com MUC ou com MUP (em que o d_{free} é menor) pode fazer uma diferença importante em termos de complexidade.

Construir o diagrama de estados de um CCC concatenado nem sempre é uma tarefa fácil. Quando não for possível desenhá-lo para a análise do caminho de d_{free}, resta-nos fazer uma simulação computacional até uma determinada profundidade ou usar algum algoritmo que calcule o d_{free}. O número de estados e o número de transições do CCC concatenado será, respectivamente, $2^{k_1.(m_1+m_2)}$ e $2^{k_1.(m_1+m_2+1)}$.

5.1.2 Estrutura dos Geradores e dos Operadores Lógicos

A matriz geradora do CCC concatenado é o produto das matrizes geradoras dos CCCs da concatenação. Uma matriz verificação de paridade para o CCC concatenado pode ser encontrada se fizermos uma *justaposição* da matriz verificação de paridade do segundo codificador, a matriz $\mathbf{H}_X$, com a matriz verificação de paridade do primeiro codificador projetada sobre o segundo codificador (basta tomar as linhas da matriz verificação de paridade do primeiro codificador como sequências de informação a serem codificadas pelo segundo codificador), a matriz $\mathbf{H}_Z$. As linhas da matriz verificação de paridade e da matriz geradora do CCC concatenado podem nos fornecer, respectivamente, os geradores do grupo

estabilizador do código quântico e as operações lógicas para os qubits de informação. Isto porque os operadores lógicos devem ser independentes dos geradores do estabilizador e comutar com cada um deles.

5.1.3 As Degenerescências de um CCQ Concatenado

Todos os CCQs concatenados são degenerados ? Não podemos ainda responder a esta questão. Todos os CCQs concatenados que estudamos são degenerados, mas por enquanto temos apenas uma conjectura. Não faremos um estudo das degenerescências dos CCQs deste capítulo. Isto requer um estudo à parte, pois a identificação completa de todas as degenerescências é uma tarefa complexa. Deixe-nos colocar com mais profundidade algumas destas dificuldades.

As degenerescências podem ocorrer dentro de um mesmo padrão de erros (como no caso do CCQ [4, 1, 3]) *e/ou*, para quando o CCQ corrigir mais de um erro quântico geral, entre padrões de erros distintos. No caso de CBQs concatenados gerados a partir de um código de bloco de repetição (como o código de Shor), a identificação das degenerescências é simples. Neste caso, *todos* os padrões de peso par de erros Z ocorrendo dentro do mesmo bloco têm efeito nulo (padrão de peso zero) sobre o bloco. Além disso, *todos* os padrões de peso ímpar de erros Z ocorrendo dentro do mesmo bloco têm o mesmo efeito dos padrões de peso um sobre o bloco. Identificado o efeito local efetivo dos padrões de erros Z, é fácil identificar o efeito global dos padrões de erros Z sobre todos os blocos do código. Deste modo, torna-se fácil identificar todas as degenerescências do código. Para CCQs concatenados, em que não se pode compactar a palavra-código em um produto tensorial de blocos, nada disto vale. Não se pode diferenciar entre padrões de peso par e ímpar de erros Z e nem afirmar que *todos* os padrões e *todos* os grupos destes padrões seguem uma regra de degenerescência.

O aparecimento da classe dos códigos quânticos degenerados trouxe boas e más notícias para a teoria dos códigos corretores de erros quânticos. Uma das más notícias é a de que algumas das técnicas usadas classicamente para provar limitantes de correção de erros deixam de ser válidas para os códigos degenerados[3]. Por outro lado, uma das boas notícias

[3]Veja a seção 2.4.

é a de que os códigos quânticos degenerados parecem estar entre os códigos quânticos mais interessantes. Estes códigos são capazes de "empacotar" mais informação que os códigos clássicos porque erros distintos não necessariamente mapeiam o espaço do código em espaços ortogonais[4], e é possível (embora ainda não tenha sido provado) que esta habilidade extra possa levar os códigos degenerados a armazenar informação quântica mais eficientemente que qualquer código não degenerado.

Com o método de construção de códigos quânticos a partir de códigos clássicos, sejam eles de bloco ou convolucionais, não precisamos nos preocupar com as degenerescências dos códigos que servem como geradores. Para todo efeito, estes códigos são códigos clássicos escritos sob a notação quântica e sabemos que, classicamente, não existe degenerescência de erros. O mesmo não se pode dizer para alguns códigos quânticos que corrigem padrões mais gerais de erros quânticos e que já são construídos a partir de códigos geradores "puramente quânticos" (por exemplo, a partir de códigos que já corrigem um erro quântico geral).

5.1.4 O Uso de Codificadores Primitivos

No Capítulo 4, produzimos um CCQ [4, 1, 3] a partir de um único CCC (2, 1, 2). O mesmo pode ser feito com outros CCCs de diferentes taxas e memórias. O uso de um mesmo codificador primitivo para gerar os codificadores da cadeia de concatenação é uma técnica importante para a simplificação do processo de geração e de decodificação de CCQs. Além de preservar a matriz geradora, de verificação de paridade, a distância e a não-catastroficidade para o segundo codificador, evita que resolvamos duas equações de síndromes distintas, algo que pode ser extremamente complexo para alguns codificadores com mais de um registro de entrada. Nem sempre o segundo codificador tem o d_{free} máximo (e muito menos o ótimo) para aquela taxa e memória, mas quando isto não ocorre será sempre um valor muito próximo.

Dentre os codificadores primitivos, os CCCs (n, 1, n) merecem especial atenção. A concatenação de CCCs (n, 1, n) com CCCs (n^2, n, 1) pode ser usada na construção de

[4]Lembremo-nos de que, na teoria dos códigos corretores de erros clássicos, erros em diferentes bits necessariamente levam a palavras-código corrompidas diferentes.

CCQs [n^2, 1, $n+1$] desde que a distância requerida seja alcançada. Fizemos isto para os valores n=1,2,3 e 4 e obtivemos as distâncias d_{free}=7, 9, 24 e 54. Haverá alguma "lei de formação", ainda que aproximada, por trás desta série ? Esta classe de CCCs é uma subclasse de duas classes mais gerais de CCCs (n, k, n) e (n, 1, m).

Os CCCs (2, 1, m) e (3, 1, m) merecem especial atenção pela simplicidade e distâncias alcançadas e por isso serão tratados à parte. Os bons CCCs (2, 1, m) e (3, 1, m) estão tabelados para várias memórias. Tomaremos sempre os CCCs ótimos como os CCCs primitivos de nossa construção. Nada nos impede de usar os CCCs que não têm máximo d_{free} ou que não são ótimos, mas como sabemos que a concatenação de bons CCCs gera um bom CCC concatenado, tomaremos sempre os CCCs ótimo como referência-padrão. Por vezes, o uso de dois CCCs ótimos é importante para se construir um bom CCQ concatenado. Este caso será discutido em uma seção mais adiante.

5.2 Uso de Codificadores Primitivos (2, 1, m)

A grande vantagem de construirmos CCQs a partir de CCCs de taxa 1/2 é o fato de ser fácil encontrar a matriz verificação de paridade, já que para $\mathbf{G}(D) = [\mathbf{g}^{(1)}(D),\ \mathbf{g}^{(2)}(D)]$, temos que $\mathbf{H}(D) = [\mathbf{g}^{(2)}(D),\ \mathbf{g}^{(1)}(D)]$. Além disso, as soluções da equação de síndromes para um codificador de taxa 1/2 são facilmente encontradas. Vimos na seção 3.6.1 que as soluções da equação $\mathbf{s}(D) = \mathbf{e}(D)\mathbf{H}^T(D)$ para um CCC de taxa 1/2, onde $\mathbf{e}(D) = [\mathbf{e}^{(1)}(D),\ \mathbf{e}^{(2)}(D)]$, são:

$$\begin{aligned} \mathbf{e}^{(1)}(D) &= \alpha_1(D)\mathbf{s}(D) + \mathbf{g}^{(1)}(D)\mathbf{t}(D), \\ \mathbf{e}^{(2)}(D) &= \alpha_2(D)\mathbf{s}(D) - \mathbf{g}^{(2)}(D)\mathbf{t}(D), \end{aligned} \tag{5.1}$$

onde $\alpha_1(D)\mathbf{g}^{(2)}(D) + \alpha_2(D)\mathbf{g}^{(1)}(D) = 1$ e $\mathbf{t}(D)$ é um polinômio arbitrário em $F[D]$. Nosso interesse em construir CCQs a partir de CCCs de taxa 1/2 torna-se ainda mais relevante quando nos lembramos de que encontrar a matriz verificação de paridade e as soluções da equação de síndromes para um codificador de taxa k/n (para $k > 1$) nem sempre é uma

tarefa simples[5].

Podemos usar codificadores de taxa $1/2$ de diferentes memórias como codificadores primitivos de codificadores de maior taxa, por vezes necessários à cadeia de concatenação de construção de CCQs. Foi o que fizemos no Capítulo 4, quando usamos um CCC (2, 1, 2) para gerar um CCC (4, 2, 1). Generalizaremos esta técnica de construção.

Vimos que, na forma de matriz discreta semi-infinita, a matriz $\mathbf{G}(D)$ de um CCC (2, 1, m) escreve-se como:

$$\mathbf{G} = \begin{bmatrix} g_0^{(1)}g_0^{(2)} & g_1^{(1)}g_1^{(2)} & g_2^{(1)}g_2^{(2)} & \cdots & g_m^{(1)}g_m^{(2)} & & \\ & g_0^{(1)}g_0^{(2)} & g_1^{(1)}g_1^{(2)} & \cdots & g_{m-1}^{(1)}g_{m-1}^{(2)} & g_m^{(1)}g_m^{(2)} & \\ & & g_0^{(1)}g_0^{(2)} & \cdots & g_{m-2}^{(1)}g_{m-2}^{(2)} & g_{m-1}^{(1)}g_{m-1}^{(2)} & g_m^{(1)}g_m^{(2)} \\ & & & & \ddots & \ddots & \ddots \end{bmatrix}. \tag{5.2}$$

Com o auxílio desta matriz, pode-se mostrar que a partir da matriz geradora de um CCC (2, 1, m) sempre é possível construir um CCC ($2m$, m, 1) com $K = m$ (número total de memórias). Por exemplo, pode-se usar um CCC (2, 1, 2) para gerar um CCC (4, 2, 1) com $K = 2$, como foi feito no Capítulo 4, ou usar um CCC (2, 1, 3) para gerar um CCC (6, 3, 1) com $K = 3$, como faremos em um exemplo mais adiante.

Além disso, para $m = 2c$, onde $c = 2, 3\ldots$, será sempre possível construir um CCC (m, $m/2$, 2). Esta situação é particularmente interessante, pois um único CCC (2, 1, m) pode dar origem à concatenação de um CCC (m, $m/2$, 2) com um CCC ($2m$, m, 1), gerando um CCC ($2m$, $m/2$, 3). Esta situação será exemplificada com a construção de um CCQ [12, 3, 3] gerado a partir de um único CCC (2, 1, 6). Apesar da taxa ser "idêntica" ao do CCQ [4, 1, 3], este CCQ [12, 3, 3] é capaz de corrigir até dois erros quânticos gerais.

Tentamos antes construir um CCQ mais simples, um CCQ [8, 2, 3] gerado a partir de um único CCC (2, 1, 4), mas o CCC (8, 2, 3), oriundo da concatenação de um CCC (4, 2, 2) com um CCC (8, 4, 1) tem $d_{free} < 17$, não sendo, portanto, útil para a correção de dois erros quânticos gerais.

Uma outra técnica de geração dos CCCs da cadeia de concatenação a partir de CCCs (2, 1, m) é o uso de dois codificadores de taxa $1/2$ com memórias distintas, um para cada codificador da cadeia de concatenação. Por exemplo, podemos tentar gerar um CCC (8,

[5]Veja a seção 3.6.2.

2, 3) a partir da concatenação de um CCC (4, 2, 2), obtido de um CCC (2, 1, 4), com um CCC (8, 4, 1), obtido de um CCC (2, 1, 8). Um outro exemplo, que será discutido com mais detalhes adiante, é a concatenação de um CCC (2, 1, 4) com um CCC (3, 2, 1) obtido a partir de um CCC (2, 1, 2) puncionado.

Para cada uma das técnicas descritas acima, a matriz verificação de paridade na forma discreta semi-infinita pode ser obtida de maneira análoga. A matriz verificação de paridade da matriz (5.2) tem a seguinte forma transposta:

$$\mathbf{H}^T = \begin{bmatrix} g_0^{(2)} & g_1^{(2)} & \cdots & g_m^{(2)} & & \\ g_0^{(1)} & g_1^{(1)} & \cdots & g_m^{(1)} & & \\ & g_0^{(2)} & \cdots & g_{m-1}^{(2)} & g_m^{(2)} & \\ & g_0^{(1)} & \cdots & g_{m-1}^{(1)} & g_m^{(1)} & \\ & & & \vdots & \vdots & \ddots \\ & & & g_0^{(2)} & g_1^{(2)} & \\ & & & g_0^{(1)} & g_1^{(1)} & \\ & & & & g_0^{(2)} & \\ & & & & g_0^{(1)} & \\ & & & & & \ddots \end{bmatrix}. \tag{5.3}$$

Para N bits de informação, a matriz (5.2) é uma matriz de dimensões $N \times 2(N+2)$, e a matriz (5.3), necessária para a identificação inambígua dos padrões de erro corrigíveis do código, é uma matriz de dimensões $(N+4) \times 2(N+2)$. A complementariedade das dimensões destas matrizes é uma característica dos CCCs auto-duais, como os CCCs de taxa $1/2$.

Em suma, o uso de codificadores primitivos (2, 1, m) pode ser extremamente vantajoso na construção de um cadeia de concatenação de codificadores, tendo em vista a facilidade para se calcular a matriz verificação de paridade e as soluções da equação de síndromes. É fácil de perceber a infinidade de opções de combinações de técnicas de geração, de taxas e de memórias disponíveis para se alcançar a distância necessária à correção do conjunto de possíveis erros do canal.

A escolha da técnica deve ser ditada pela taxa, capacidade de correção, grau de simplicidade e tipo de canal almejados. Nem sempre é uma tarefa simples, como pode parecer à

primeira vista. É necessário um pouco de experiência prática para que o bom senso impere na escolha da técnica de concatenação.

5.2.1 O CCQ [4, 1, 2] e o CCQ [4, 1, 3] Revisitados

Considere o CCC (2, 1, 1) com matriz geradora $\mathbf{G}(D) = [1+D,\, D]$. Este CCC tem $d_{free} = 3$ e, portanto, é capaz de corrigir um erro. Em notação quântica, teremos a seguinte operação de codificação:

$$\bigotimes_{t=0}^{+\infty} |u_t\rangle \longmapsto \bigotimes_{t=0}^{+\infty} |u_t + u_{t-1},\, u_{t-1}\rangle, \tag{5.4}$$

onde $u_{-1} = 0$. Este é um CCQ [2, 1, 1] capaz de corrigir um erro X em qualquer qubit da palavra-código. Uma simples transformação de Hadamard nos possibilita também construir um CCQ [2, 1, 1] capaz de corrigir um erro Z em qualquer qubit da palavra-código.

Suponha agora que geremos um CCC (4, 2, 1) a partir da *mesma* matriz geradora do CCC (2, 1, 1). Assim, a matriz geradora do CCC (4, 2, 1) equivalente será então:

$$\mathbf{G}(D) = \begin{bmatrix} 1 & 0 & 1 & 1 \\ D & D & 1 & 0 \end{bmatrix}. \tag{5.5}$$

Analogamente ao que foi feito para o CCC (2, 1, 1), este CCC (4, 2, 1) pode ser usado para construir um CCQ [4, 2, 1] capaz de corrigir um erro X (ou Z) sobre qualquer qubit da palavra-código.

Veja agora a Figura 5.1. A concatenação dos CCCs (2, 1, 1) e (4, 2, 1) produz um CCC (4, 1, 2) com matriz geradora

$$\mathbf{G}(D) = \begin{bmatrix} 1+D+D^2 & D^2 & 1 & 1+D \end{bmatrix}. \tag{5.6}$$

Veja o diagrama de estados deste CCC na Figura 5.2. Este CCC tem $d_{free} = 7$ e portanto pode corrigir até três erros clássicos.

Podemos usar este CCC (4, 1, 2) para construir um CCQ [4, 1, 2] com a seguinte operação de codificação:

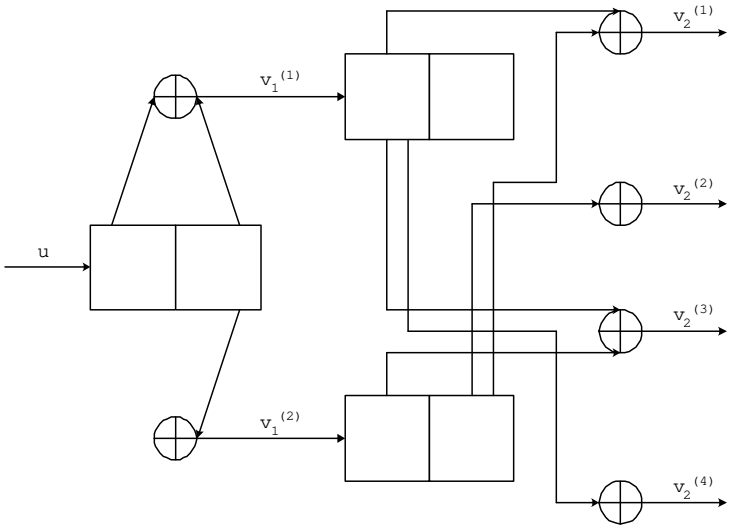

Figura 5.1: Concatenação dos codificadores (2, 1, 1) e (4, 2, 1).

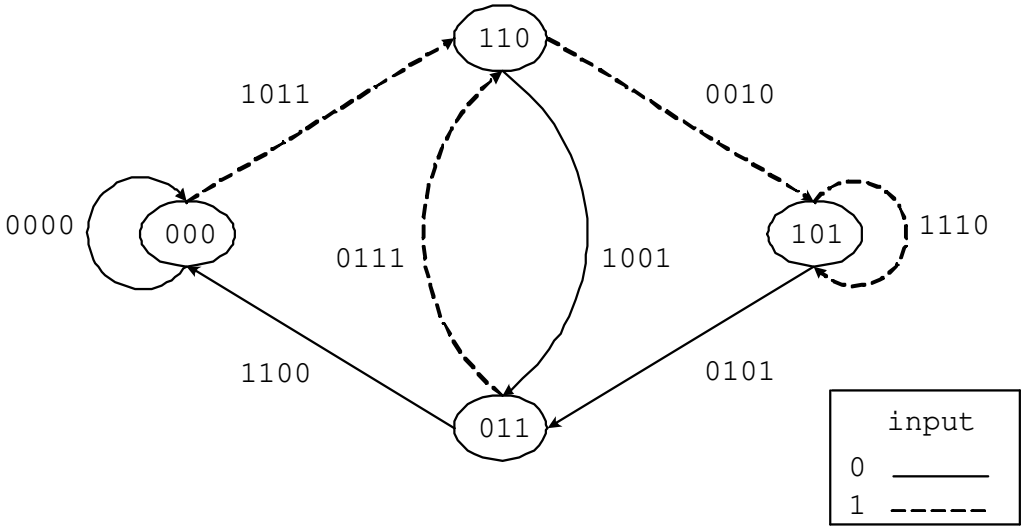

Figura 5.2: Diagrama de estados do CCC (4, 1, 2).

$$\bigotimes_{t=0}^{+\infty} |u_t\rangle \longmapsto \bigotimes_{t=0}^{+\infty} \left\{ \sum_{(p_t,q_t)=(0,0)}^{(1,1)} \frac{1}{2}(-1)^{(u_t+u_{t-1})p_t+(u_{t-1})q_t} |p_t+q_{t-1}, q_{t-1}, p_t+q_t, p_t\rangle \right\}, \quad (5.7)$$

onde $u_{-1} = 0$ e $p_{-1} = q_{-1} = 0$. Este código, porém, *não* é capaz de corrigir um erro quântico geral, pois para isto o d_{free} do CCC (4, 1, 2) associado teria que ser pelo menos

igual a nove. Pode-se mostrar que nenhum CCC (2, 1, 1) é capaz de gerar um CCC (4, 1, 2) que alcance $d_{free} = 9$.

Ou seja, a concatenação de um CCQ capaz de corrigir um erro de fase com um CCQ capaz de corrigir um erro de bit não produz *necessariamente* um CCQ capaz de corrigir um erro geral. Além disso, um CCC de taxa $R = 1/n$ capaz de corrigir t erros, nem sempre pode ser estendido para produzir um CCQ de taxa $R^2 = 1/n^2$ capaz de corrigir t erros quânticos gerais. O CCQ [4, 1, 3] apresentado no Capítulo 4 é apenas um caso particular de CCQ concatenado capaz de satisfazer estas duas condições. Mesmo com alguns CCCs (2, 1, 2) não é possível produzir um CCC (4, 1, 3) capaz de alcançar $d_{free} = 9$. Isto nos faz perceber o quanto o CCC (2, 1, 2) com $\mathbf{G}(D) = [1 + D^2,\ 1 + D + D^2]$ é especial.

O CCQ [4, 1, 2] apresentado nesta seção é fundamental para entendermos a importância do conceito de memória convolucional quântica no processo de geração de CCQs. Obviamente, este conceito deve também ser levado em consideração no processo de decodificação.

5.2.2 Um CCQ [12, 3, 3]

Considere o CCC (2, 1, 6) com matriz geradora polinomial $\mathbf{G}(D) = [1 + D^2 + D^3 + D^5 + D^6,\ 1 + D + D^2 + D^3 + D^6]$ [6]. Em termos de notação de matriz discreta semi-infinita, esta matriz escreve-se como:

$$\mathbf{G} = \begin{bmatrix} 11 & 01 & 11 & 11 & 00 & 10 & 11 & & & \\ & 11 & 01 & 11 & 11 & 00 & 10 & 11 & & \\ & & 11 & 01 & 11 & 11 & 00 & 10 & 11 & \\ & & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \end{bmatrix}. \tag{5.8}$$

Este CCC tem $d_{free} = 10$ e, portanto, pode corrigir até quatro erros clássicos. Por ser um CCC de taxa 1/2, temos que $\mathbf{H}(D) = [1 + D + D^2 + D^3 + D^6,\ 1 + D^2 + D^3 + D^5 + D^6]$. Em termos de notação de matriz discreta semi-infinita, teremos esta matriz verificação de paridade escrita como:

[6] Referência [41], páginas 330-331.

$$\mathbf{H} = \begin{bmatrix} 11 & & & & & & & & \\ 10 & 11 & & & & & & & \\ 11 & 10 & 11 & & & & & & \\ 11 & 11 & 10 & 11 & & & & & \\ 00 & 11 & 11 & 10 & 11 & & & & \\ 01 & 00 & 11 & 11 & 10 & 11 & & & \\ 11 & 01 & 00 & 11 & 11 & 10 & 11 & & \\ & 11 & 01 & 00 & 11 & 11 & 10 & 11 & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \end{bmatrix}. \tag{5.9}$$

De acordo com (5.1), as soluções da equação de síndromes para este codificador são:

$$\begin{aligned} \mathbf{e}^{(1)}(D) &= \mathbf{s}(D) + D\mathbf{s}(D) + D^2\mathbf{s}(D) + D^3\mathbf{s}(D) + D^4\mathbf{s}(D) + \\ &\quad \mathbf{t}(D) + D^2\mathbf{t}(D) + D^3\mathbf{t}(D) + D^5\mathbf{t}(D) + D^6\mathbf{t}(D), \\ \mathbf{e}^{(2)}(D) &= D^2\mathbf{s}(D) + D^4\mathbf{s}(D) + \mathbf{t}(D) + D\mathbf{t}(D) + D^2\mathbf{t}(D) + \\ &\quad D^3\mathbf{t}(D) + D^6\mathbf{t}(D), \end{aligned} \tag{5.10}$$

onde aqui usamos $\alpha_1(D) = 1 + D + D^2 + D^3 + D^4$ e $\alpha_2(D) = D^2 + D^4$.

Os CCQs [6, 3, 2] e [12, 6, 1] para os canais bit flip e phase flip

Faremos uso da matriz (5.8) para construir um CCC (6, 3, 2). Assim, esta matriz pode ser reescrita como:

$$\mathbf{G} = \begin{bmatrix}
110111 & 110010 & 110000 & & \\
001101 & 111100 & 101100 & & \\
000011 & 011111 & 001011 & & \\
 & 110111 & 110010 & 110000 & \\
 & 001101 & 111100 & 101100 & \\
 & 000011 & 011111 & 001011 & \\
 & & \ddots\ddots\ddots & \ddots\ddots\ddots & \ddots\ddots\ddots \\
 & & \ddots\ddots\ddots & \ddots\ddots\ddots & \ddots\ddots\ddots \\
 & & \ddots\ddots\ddots & \ddots\ddots\ddots & \ddots\ddots\ddots
\end{bmatrix}. \tag{5.11}$$

Estamos, agora, em condições de gerar o CCQ [6, 3, 2] para os canais bit flip e phase flip. Para o canal phase flip (que será o mais interessante neste contexto), a operação de codificação na base $\{|0\rangle, |1\rangle\}$ será escrita como:

$$\bigotimes_{t=0}^{+\infty} |u_t^{(1)}, u_t^{(2)}, u_t^{(3)}\rangle \longmapsto \bigotimes_{t=0}^{+\infty} \left\{ \frac{1}{8}(|0\rangle + (-1)^{v_t^{(1)}}|1\rangle)(|0\rangle + (-1)^{v_t^{(2)}}|1\rangle) \right.$$

$$\left. (|0\rangle + (-1)^{v_t^{(3)}}|1\rangle)(|0\rangle + (-1)^{v_t^{(4)}}|1\rangle)(|0\rangle + (-1)^{v_t^{(5)}}|1\rangle)(|0\rangle + (-1)^{v_t^{(6)}}|1\rangle) \right\}, \tag{5.12}$$

onde $v_t^{(1)} = u_t^{(1)} + u_{t-1}^{(1)} + u_{t-2}^{(1)} + u_{t-1}^{(2)} + u_{t-2}^{(2)}$, $v_t^{(2)} = u_t^{(1)} + u_{t-1}^{(1)} + u_{t-2}^{(1)} + u_{t-1}^{(2)} + u_{t-1}^{(3)}$, $v_t^{(3)} = u_t^{(2)} + u_{t-1}^{(2)} + u_{t-2}^{(2)} + u_{t-1}^{(3)} + u_{t-2}^{(3)}$, $v_t^{(4)} = u_t^{(1)} + u_t^{(2)} + u_{t-1}^{(2)} + u_{t-2}^{(2)} + u_{t-1}^{(3)}$, $v_t^{(5)} = u_t^{(1)} + u_{t-1}^{(1)} + u_t^{(3)} + u_{t-1}^{(3)} + u_{t-2}^{(3)}$ e $v_t^{(6)} = u_t^{(1)} + u_t^{(2)} + u_t^{(3)} + u_{t-1}^{(3)} + u_{t-2}^{(3)}$. Aqui, definimos $u_{-1}^{(1)} = u_{-2}^{(1)} = u_{-1}^{(2)} = u_{-2}^{(2)} = u_{-1}^{(3)} = u_{-2}^{(3)} = 0$. Cada um dos 2^{3N} estados da base de uma sequência de informação com N *blocos* de qubits será codificado em uma superposição de $2^{6(N+2)}$ estados, cada um dos quais com $6(N+2)$ qubits. Naturalmente, trata-se de um CCQ [6, 3, 2] capaz de garantir a correção de até quatro erros Z sobre a palavra-código.

A matriz verificação de paridade (5.9) pode agora ser reescrita como:

$$\mathbf{H} = \begin{bmatrix}
110000 & & & & & \\
101100 & & & & & \\
111011 & & & & & \\
111110 & 110000 & & & & \\
001111 & 101100 & & & & \\
010011 & 111011 & & & & \\
110100 & 111110 & 110000 & & & \\
001101 & 001111 & 101100 & & & \\
000011 & 010011 & 111011 & & & \\
 & 110100 & 111110 & 110000 & & \\
 & 001101 & 001111 & 101100 & & \\
 & 000011 & 010011 & 111011 & & \\
 & & \ddots\ddots\ddots & \ddots\ddots\ddots & \ddots\ddots\ddots & \\
 & & \ddots\ddots\ddots & \ddots\ddots\ddots & \ddots\ddots\ddots & \\
 & & \ddots\ddots\ddots & \ddots\ddots\ddots & \ddots\ddots\ddots &
\end{bmatrix}. \tag{5.13}$$

As linhas da matriz de paridade (5.13) e da matriz geradora (5.11) permitem-nos escrever, respectivamente, os geradores do grupo estabilizador e as operações lógicas $\overline{Z}$ atuando sobre os qubits de informação. No caso de um canal phase flip, os 0s e 1s da matriz de paridade (5.13) devem ser substituídos por operadores I e X e os 0s e 1s da matriz geradora (5.11) por operadores I e Z.

O CSL de decodificação de síndromes para o CCC (6, 3, 2) é apresentado na Figura 5.3.

Ao entrarem no circuito da Figura 5.3 os $(N+2)$ blocos recebidos do canal ruidoso, são necessários mais dois blocos de zeros para retornar ao estado inicial (todo nulo) do decodificador de síndromes. Assim, para uma sequência de informação com $3N$ qubits, precisaremos considerar a medida de $3((N+2)+2) = 3N+12$ observáveis para poder aplicar o ADS clássico. Quanto à manipulação lógica deste código, pode-se afirmar que uma transformação unitária sobre $3N$ qubits *lógicos* será implementada através de uma transformação unitária sobre os $6(N+2)$ qubits *físicos*.

Como as matrizes (5.9) e (5.13) são exatamente as mesmas, podemos usar as soluções

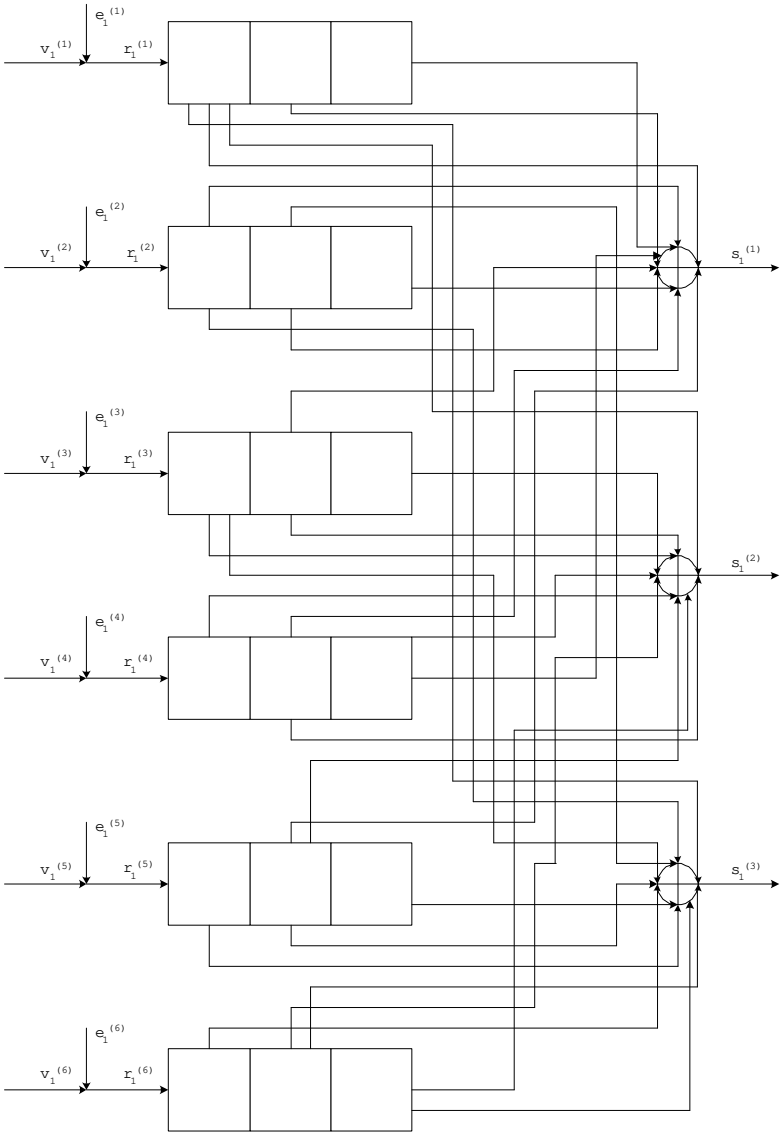

Figura 5.3: Decodificador de síndromes para o codificador (6, 3, 2).

(5.10) para determinar a sequência de erros $\mathbf{e} = (\mathbf{e}^{(1)}, \mathbf{e}^{(2)}, \mathbf{e}^{(3)}, \mathbf{e}^{(4)}, \mathbf{e}^{(5)}, \mathbf{e}^{(6)})$ da equação de síndromes $\mathbf{s} = \mathbf{eH}^T$, com $\mathbf{s} = (\mathbf{s}^{(1)}, \mathbf{s}^{(2)}, \mathbf{s}^{(3)})$ e $\mathbf{r} = (\mathbf{r}^{(1)}, \mathbf{r}^{(2)}, \mathbf{r}^{(3)}, \mathbf{r}^{(4)}, \mathbf{r}^{(5)}, \mathbf{r}^{(6)})$. Aqui, porém, precisaremos de três vezes mais unidades de tempo em relação ao CCC (2, 1, 6) para realizar a decodificação de uma sequência de bits.

Analogamente, podemos usar a matriz (5.8) para construir um CCC (12, 6, 1). Assim, a matriz (5.8) pode ser reescrita como:

$$\mathbf{G} = \left[\begin{array}{cccc}
110111110010 & 110000000000 & & \\
001101111100 & 101100000000 & & \\
000011011111 & 001011000000 & & \\
000000110111 & 110010110000 & & \\
000000001101 & 111100101100 & & \\
000000000011 & 011111001011 & & \\
 & 110111110010 & 110000000000 & \\
 & 001101111100 & 101100000000 & \\
 & 000011011111 & 001011000000 & \\
 & 000000110111 & 110010110000 & \\
 & 000000001101 & 111100101100 & \\
 & 000000000011 & 011111001011 & \\
 & & \ddots & \ddots \\
 & & \ddots & \ddots \\
 & & \ddots & \ddots \\
 & & \ddots & \ddots \\
 & & \ddots & \ddots \\
 & & \ddots & \ddots
\end{array} \right]. \tag{5.14}$$

Estamos agora em condições de gerar o CCQ [12, 6, 1] para os canais bit flip e phase flip. Para o canal bit flip (que será o mais interessante neste contexto), a operação de codificação será escrita como:

$$\bigotimes_{t=0}^{+\infty} |u_t^{(1)}, u_t^{(2)}, u_t^{(3)}, u_t^{(4)}, u_t^{(5)}, u_t^{(6)}\rangle \longmapsto \bigotimes_{t=0}^{+\infty} |v_t^{(1)}, v_t^{(2)}, v_t^{(3)}, v_t^{(4)}, v_t^{(5)}, v_t^{(6)}, v_t^{(7)}, v_t^{(8)}, v_t^{(9)}, v_t^{(10)}, v_t^{(11)}, v_t^{(12)}\rangle, \tag{5.15}$$

onde $v_t^{(1)} = u_t^{(1)} + u_{t-1}^{(1)} + u_{t-1}^{(2)} + u_{t-1}^{(4)} + u_{t-1}^{(5)}$, $v_t^{(2)} = u_t^{(1)} + u_{t-1}^{(1)} + u_{t-1}^{(4)} + u_{t-1}^{(5)} + u_{t-1}^{(6)}$, $v_t^{(3)} = u_t^{(2)} + u_{t-1}^{(2)} + u_{t-1}^{(3)} + u_{t-1}^{(5)} + u_{t-1}^{(6)}$, $v_t^{(4)} = u_t^{(1)} + u_t^{(2)} + u_{t-1}^{(2)} + u_{t-1}^{(5)} + u_{t-1}^{(6)}$, $v_t^{(5)} = u_t^{(1)} + u_t^{(3)} + u_{t-1}^{(3)} + u_{t-1}^{(4)} + u_{t-1}^{(6)}$, $v_t^{(6)} = u_t^{(1)} + u_t^{(2)} + u_t^{(3)} + u_{t-1}^{(3)} + u_{t-1}^{(6)}$, $v_t^{(7)} = u_t^{(1)} + u_t^{(2)} + u_t^{(4)} + u_{t-1}^{(4)} + u_{t-1}^{(5)}$, $v_t^{(8)} = u_t^{(1)} + u_t^{(2)} + u_t^{(3)} + u_t^{(4)} + u_{t-1}^{(4)}$, $v_t^{(9)} = u_t^{(2)} + u_t^{(3)} + u_t^{(5)} + u_{t-1}^{(5)} + u_{t-1}^{(6)}$, $v_t^{(10)} = u_t^{(2)} + u_t^{(3)} + u_t^{(4)} + u_t^{(5)} + u_{t-1}^{(5)}$, $v_t^{(11)} = u_t^{(1)} + u_t^{(3)} + u_t^{(4)} + u_t^{(6)} + u_{t-1}^{(6)}$ e $v_t^{(12)} = u_t^{(3)} + u_t^{(4)} + u_t^{(5)} + u_t^{(6)} + u_{t-1}^{(6)}$.

Aqui, definimos $u_{-1}^{(1)} = u_{-1}^{(2)} = u_{-1}^{(3)} = u_{-1}^{(4)} = u_{-1}^{(5)} = u_{-1}^{(6)} = 0$. Cada um dos 2^{6N} estados da base de uma sequência de informação com N blocos de qubits será codificado em um estado com $12(N+1)$ qubits. Naturalmente, trata-se de um CCQ [12, 6, 1] capaz de garantir a correção de até quatro erros X sobre a palavra-código.

A matriz verificação de paridade (5.9) pode agora ser reescrita como:

$$
\mathbf{H} = \left[\begin{array}{lll}
110000000000 & & \\
101100000000 & & \\
111011000000 & & \\
111110110000 & & \\
001111101100 & & \\
010011111011 & & \\
110100111110 & 110000000000 & \\
001101001111 & 101100000000 & \\
000011010011 & 111011000000 & \\
000000110100 & 111110110000 & \\
000000001101 & 001111101100 & \\
000000000011 & 010011111011 & \\
 & \ddots\ddots\ddots\ddots\ddots\ddots & \ddots\ddots\ddots\ddots\ddots\ddots \\
 & \ddots\ddots\ddots\ddots\ddots\ddots & \ddots\ddots\ddots\ddots\ddots\ddots \\
 & \ddots\ddots\ddots\ddots\ddots\ddots & \ddots\ddots\ddots\ddots\ddots\ddots \\
 & \ddots\ddots\ddots\ddots\ddots\ddots & \ddots\ddots\ddots\ddots\ddots\ddots \\
 & \ddots\ddots\ddots\ddots\ddots\ddots & \ddots\ddots\ddots\ddots\ddots\ddots \\
 & \ddots\ddots\ddots\ddots\ddots\ddots & \ddots\ddots\ddots\ddots\ddots\ddots
\end{array} \right]. \qquad (5.16)
$$

As linhas da matriz de paridade (5.16) e da matriz geradora (5.14) permitem-nos escrever, respectivamente, os geradores do grupo estabilizador e as operações lógicas $\overline{X}$ atuando sobre os qubits de informação. No caso de um canal phase flip, os 0s e 1s da matriz (5.16) devem ser substituídos por operadores I e Z e os 0s e 1s da matriz (5.14) por operadores I e X.

O CSL de decodificação de síndromes para o CCC (12, 6, 1) é apresentado na Figura 5.4.

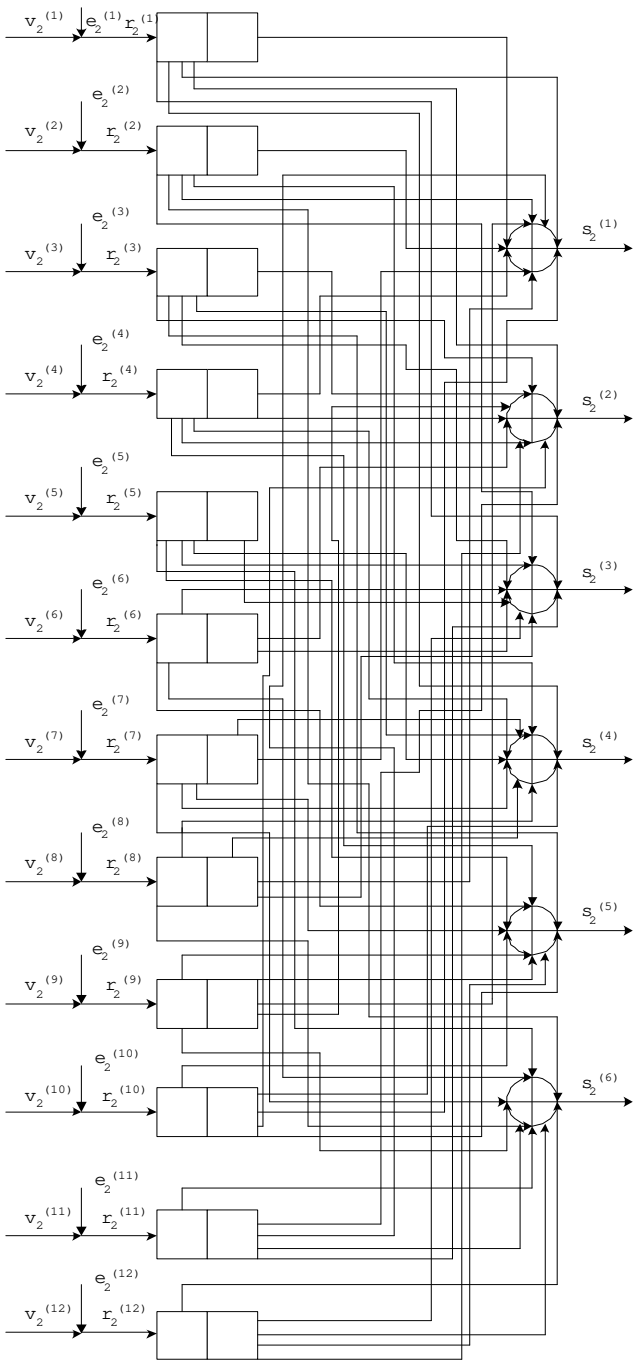

Figura 5.4: Decodificador de síndromes para o codificador (12, 6, 1).

Ao entrarem no circuito da Figura 5.4 os $(N+1)$ blocos recebidos do canal ruidoso, é necessário mais um bloco de zeros para retornar ao estado inicial (todo nulo) do decodificador de síndromes. Assim, para uma sequência de informação com $6N$ qubits, precisare-

mos considerar a medida de $6((N+1)+1) = 6N+12$ observáveis para poder aplicar o ADS clássico. Quanto à manipulação lógica deste código, pode-se afirmar que uma transformação unitária sobre $6N$ qubits *lógicos* será implementada através de uma transformação unitária sobre os $12(N+1)$ qubits *físicos*.

Como as matrizes (5.9) e (5.16) são exatamente as mesmas, podemos usar as soluções (5.10) para determinar a sequência de erros $\mathbf{e} = (\mathbf{e}^{(1)}, \mathbf{e}^{(2)}, \mathbf{e}^{(3)}, \mathbf{e}^{(4)}, \mathbf{e}^{(5)}, \mathbf{e}^{(6)}, \mathbf{e}^{(7)}, \mathbf{e}^{(8)}, \mathbf{e}^{(9)}, \mathbf{e}^{(10)}\ \mathbf{e}^{(11)}, \mathbf{e}^{(12)})$ da equação de síndromes $\mathbf{s} = \mathbf{e}\mathbf{H}^T$, com $\mathbf{s} = (\mathbf{s}^{(1)}, \mathbf{s}^{(2)}, \mathbf{s}^{(3)}, \mathbf{s}^{(4)}, \mathbf{s}^{(5)}, \mathbf{s}^{(6)})$ e $\mathbf{r} = (\mathbf{r}^{(1)}, \mathbf{r}^{(2)}, \mathbf{r}^{(3)}, \mathbf{r}^{(4)}, \mathbf{r}^{(5)}, \mathbf{r}^{(6)}, \mathbf{r}^{(7)}, \mathbf{r}^{(8)}, \mathbf{r}^{(9)}, \mathbf{r}^{(10)}, \mathbf{r}^{(11)}, \mathbf{r}^{(12)})$. Aqui, no entanto, precisaremos de seis vezes mais unidades de tempo em relação ao CCC (2, 1, 6) para realizar a decodificação de uma sequência de bits.

O CCQ [12, 3, 3] para o canal com erro quântico geral

Quando fazemos a concatenação dos CCCs (6, 3, 2) e (12, 6, 1), obtemos um novo CCC, de taxa 3/12 e três memórias efetivas. Veja o CSL do CCC (12, 3, 3) na Figura 5.5.

Não foi possível reproduzir nesta folha o diagrama de estados deste CCC. Trata-se de um diagrama com $2^{3.(2+1)} = 512$ estados e $2^{3.(2+1+1)} = 4096$ transições entre os estados. Através de uma simulação computacional foi possível verificar que este CCC tem $d_{free} = 20$ e, portanto, é capaz de corrigir até nove erros sobre a palavra-código.

O CCC (12, 3, 3) pode ser usado na contrução de um CCQ capaz de garantir a correção de até dois erros quânticos gerais com geradores Z e X sobre a palavra-código quântica. Isto porque os CCCs (6, 3, 2) e (12, 6, 1), associados respectivamente à correção de erros de fase e de bit, são capazes de garantir a correção de dois erros e o CCC (12, 3, 3), associado à correção de erros quânticos gerais, é capaz de garantir a correção de oito erros. Todos estes CCCs tem capacidade de correção além da exigida para a correção de dois erros quânticos gerais, o que nos permite afirmar que alguns padrões com mais de dois erros X e Z também poderão ser corrigidos por este código.

A operação de codificação para este CCQ [12, 3, 3] pode ser escrita como:

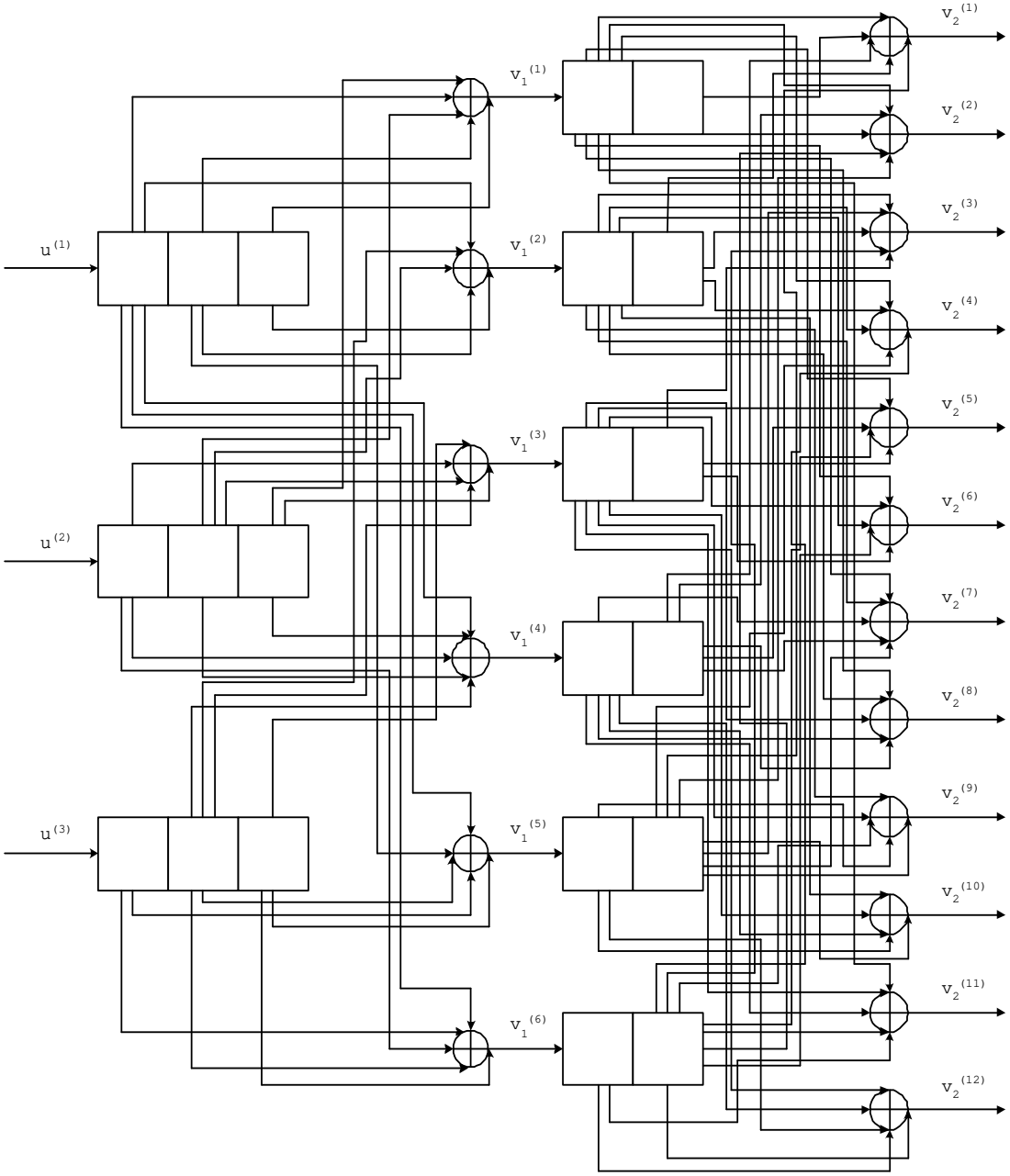

Figura 5.5: Concatenação dos codificadores (6, 3, 2) e (12, 6, 1).

$$\bigotimes_{t=0}^{+\infty} |u_t^{(1)}, u_t^{(2)}, u_t^{(3)}\rangle \longmapsto \bigotimes_{t=0}^{+\infty} \left\{ \sum_{(p_t,q_t,r_t,x_t,y_t,z_t)=(0,0,0,0,0,0)}^{(1,1,1,1,1,1)} \frac{1}{8}(-1)^{v_t^{(1)}p_t+v_t^{(2)}q_t+v_t^{(3)}r_t+v_t^{(4)}x_t+v_t^{(5)}y_t+v_t^{(6)}z_t} \right.$$

$$\left| p_t + p_{t-1} + q_{t-1} + x_{t-1} + y_{t-1},\ \ p_t + p_{t-1} + x_{t-1} + y_{t-1} + z_{t-1},\ \ q_t + q_{t-1} + r_{t-1} + y_{t-1} + z_{t-1}, \right.$$

$$p_t + q_t + q_{t-1} + y_{t-1} + z_{t-1}, p_t + r_t + r_{t-1} + x_{t-1} + z_{t-1}, p_t + q_t + r_t + r_{t-1} + z_{t-1},$$

$$p_t + q_t + x_t + x_{t-1} + y_{t-1}, p_t + q_t + r_t + x_t + x_{t-1}, q_t + r_t + y_t + y_{t-1} + z_{t-1},$$

$$q_t + r_t + x_t + y_t + y_{t-1}, p_t + r_t + x_t + z_t + z_{t-1}, r_t + x_t + y_t + z_t + z_{t-1}\rangle \Bigg\}, \tag{5.17}$$

onde $v_t^{(1)} = u_t^{(1)} + u_{t-1}^{(1)} + u_{t-2}^{(1)} + u_{t-1}^{(2)} + u_{t-2}^{(2)}$, $v_t^{(2)} = u_t^{(1)} + u_{t-1}^{(1)} + u_{t-2}^{(1)} + u_{t-1}^{(2)} + u_{t-1}^{(3)}$, $v_t^{(3)} = u_t^{(2)} + u_{t-1}^{(2)} + u_{t-2}^{(2)} + u_{t-1}^{(3)} + u_{t-2}^{(3)}$, $v_t^{(4)} = u_t^{(1)} + u_t^{(2)} + u_{t-1}^{(2)} + u_{t-2}^{(2)} + u_{t-1}^{(3)}$, $v_t^{(5)} = u_t^{(1)} + u_{t-1}^{(1)} + u_t^{(3)} + u_{t-1}^{(3)} + u_{t-2}^{(3)}$ e $v_t^{(6)} = u_t^{(1)} + u_t^{(2)} + u_t^{(3)} + u_{t-1}^{(3)} + u_{t-2}^{(3)}$. Aqui, definimos $u_{-1}^{(1)} = u_{-2}^{(1)} = u_{-1}^{(2)} = u_{-2}^{(2)} = u_{-1}^{(3)} = u_{-2}^{(3)} = 0$ e $p_{-1} = q_{-1} = r_{-1} = x_{-1} = y_{-1} = z_{-1} = 0$. Cada um dos 2^{3N} estados da base de uma sequência de informação com N blocos de qubits será codificado em uma superposição de $2^{6(N+2)}$ estados, cada um dos quais com $12((N+2)+1) = 12N+36$ qubits.

A matriz geradora polinomial do CCC (12, 3, 3) pode ser obtida através do produto das matrizes geradoras polinomiais dos CCCs (6, 3, 2) e (12, 6, 1). Esta matriz tem a seguinte forma[7]:

$$\mathbf{G}^T(D) = \begin{bmatrix} 1+D & 0 & D+D^2+D^3 \\ 1+D+D^2+D^3 & D+D^2 & D^2 \\ 1+D^2+D^3 & D+D^3 & D+D^3 \\ D+D^3 & D & D+D^2 \\ 1+D+D^2 & 1+D & 0 \\ D & 1+D+D^2+D^3 & D+D^2 \\ 1+D^2 & 1+D^2+D^3 & D+D^3 \\ 1+D & D+D^3 & D \\ 0 & 1+D+D^2 & 1+D \\ 1+D & D & 1+D+D^2+D^3 \\ 1+D^2 & 1+D^2 & 1+D^2+D^3 \\ 1 & 1+D & D+D^3 \end{bmatrix}, \tag{5.18}$$

[7]Esta matriz foi colocada na forma transposta para que fosse possível reproduzi-la aqui.

ou, na forma de matriz discreta semi-infinita,

$$\mathbf{G}_C = \begin{bmatrix} 111010110111 & 110111010100 & 011010100010 & 011100000000 & & \\ 000011101011 & 011111011101 & 010001101010 & 001001110000 & & \\ 000000001110 & 101101111101 & 110101000110 & 101000100111 & & \\ & 111010110111 & 110111010100 & 011010100010 & 011100000000 & \\ & 000011101011 & 011111011101 & 010001101010 & 001001110000 & \\ & 000000001110 & 101101111101 & 110101000110 & 101000100111 & \\ & & \ddots\ddots\ddots & \ddots\ddots\ddots & \ddots\ddots\ddots & \ddots\ddots\ddots \\ & & \ddots\ddots\ddots & \ddots\ddots\ddots & \ddots\ddots\ddots & \ddots\ddots\ddots \\ & & \ddots\ddots\ddots & \ddots\ddots\ddots & \ddots\ddots\ddots & \ddots\ddots\ddots \end{bmatrix}. \tag{5.19}$$

Chamemos a matriz (5.16) de $\mathbf{H}_X$, e a matriz (5.13), expandida para o CCQ [12, 3, 3] (basta tomar cada uma das linhas como um bloco de informação a ser codificado pela matriz (5.14) ($\equiv \mathbf{G}_X$)), de $\mathbf{H}_Z$. Com as matrizes $\mathbf{H}_X$ e $\mathbf{H}_Z$ podemos construir a matriz $\mathbf{H}_C = [\mathbf{H}_X; \mathbf{H}_Z]$. Veja a matriz (5.20).

As linhas da matriz (5.20) permitem-nos escrever os geradores do grupo estabilizador do CCQ [12, 3, 3] descrito pela operação de codificação (5.17). Usamos as linhas da matriz $\mathbf{H}_X$ para obter os geradores Z e as linhas da matriz $\mathbf{H}_Z$ para obter os geradores X. Os 0s e 1s da matriz $\mathbf{H}_X$ devem ser substituídos por operadores I e Z e os 0s e 1s da matriz $\mathbf{H}_Z$ por operadores I e X. Usaremos também as linhas da matriz $\mathbf{G}_C$ para determinar as operações lógicas tanto de $\overline{X}$ quanto de $\overline{Z}$ atuando sobre os qubits de informação. Para determinar $\overline{X}$, os 0s e 1s da matriz (5.19) devem ser substituídos por operadores I e Z, e para determinar $\overline{Z}$, os 0s e 1s da matriz (5.19) devem ser substituídos por operadores I e X. Pode-se afirmar que uma transformação unitária sobre $3N$ qubits *lógicos* será implementada através de uma transformação unitária sobre $12N+36$ qubits *físicos*.

Ao entrarem no circuito da Figura 5.4 os $(N+3)$ blocos recebidos do canal ruidoso, é necessário mais um bloco de zeros para retornar ao estado inicial (todo nulo) do decodificador de síndromes. Assim, para uma sequência de informação com $3N$ qubits, precisaremos considerar a medida de $6((N+3)+1) = 6N+24$ observáveis Z e de $3N+12$ observáveis X para poder aplicar o ADS clássico. Usamos os autovalores $\{\alpha_{M_X}\}$ e $\{\alpha_{M_Z}\}$

(onde M_X e M_Z são os observáveis Z e X, respectivamente) nas soluções (5.10) da equação de síndromes para detectar através de *dois* diagramas de treliça os qubits onde eventualmente ocorreram erros de bit e os *blocos* do código phase flip associado onde eventualmente ocorreram erros de fase.

$$
\mathbf{H}_C = \left[\begin{array}{cccccc}
110000000000 & & & & & \\
101100000000 & & & & & \\
111011000000 & & & & & \\
111110110000 & & & & & \\
001111101100 & & & & & \\
010011111011 & & & & & \\
110100111110 & 110000000000 & & & & \\
001101001111 & 101100000000 & & & & \\
000011010011 & 111011000000 & & & & \\
000000110100 & 111110110000 & & & & \\
000000001101 & 001111101100 & & & & \\
000000000011 & 010011111011 & & & & \\
 & \ddots\ddots\ddots\ddots\ddots\ddots & \ddots\ddots\ddots\ddots\ddots\ddots & & & \\
 & & \ddots\ddots\ddots\ddots\ddots\ddots & \ddots\ddots\ddots\ddots\ddots\ddots & & \\
 & & & \ddots\ddots\ddots\ddots\ddots\ddots & \ddots\ddots\ddots\ddots\ddots\ddots & \\
111010001110 & 011100000000 & & & & \\
110100011010 & 001001110000 & & & & \\
111001011111 & 110100100111 & & & & \\
111001101011 & 100011010010 & 011100000000 & & & \\
000011100110 & 101110001101 & 001001110000 & & & \\
001101110010 & 110110111000 & 110100100111 & & & \\
111010111001 & 010111011011 & 100011010010 & 011100000000 & & \\
000011101011 & 100101011101 & 101110001101 & 001001110000 & & \\
000000001110 & 101110010101 & 110110111000 & 110100100111 & & \\
 & \ddots\ddots\ddots & \ddots\ddots\ddots & \ddots\ddots\ddots & \ddots\ddots\ddots & \\
 & \ddots\ddots\ddots & \ddots\ddots\ddots & \ddots\ddots\ddots & \ddots\ddots\ddots & \\
 & \ddots\ddots\ddots & \ddots\ddots\ddots & \ddots\ddots\ddots & \ddots\ddots\ddots &
\end{array}\right]. \tag{5.20}
$$

5.3 Uso de Codificadores Primitivos (3, 1, m)

Nesta seção usaremos CCCs ótimos de taxa 1/3 com duas e três memórias para gerar bons CCQs de taxa 1/9 (Referência [14], páginas 88-89.). Três casos serão considerados. No primeiro caso, vamos concatenar um CCC (3, 1, 3) ótimo a um CCC (9, 3, 1) com matriz geradora derivada do primeiro (e, portanto, de MUC) para gerar um CCC (9, 1, 4). No segundo caso, vamos concatenar um CCC (3, 1, 2) ótimo a um CCC (9, 3, 1) com matriz geradora derivada do primeiro (e, portanto, de MUP) para gerar um CCC (9, 1, 3). Em ambos os casos, os CCCs concatenados possuem $d_{free} > 17$, o que possibilita a construção de CCQs de taxa 1/9 capazes de corrigir até dois erros quânticos gerais. Finalmente, no terceiro caso, vamos concatenar um CCC (3, 1, 3) ótimo a um CCC (9, 3, 1) também ótimo para gerar um CCC (9, 1, 4) com $d_{free} > 25$, o que possibilita a construção de um CCQ de taxa 1/9 capaz de corrigir até três erros quânticos gerais.

5.3.1 Um CCQ [9, 1, 4]

Os CCQs [3, 1, 3] e [9, 3, 1] para os canais bit flip e phase flip

Considere que façamos uso do CCC (3, 1, 3) com matriz geradora polinomial

$$\mathbf{G}(D) = [1+D^2+D^3,\ 1+D+D^3,\ 1+D+D^2+D^3] \tag{5.21}$$

na construção de um CCQ para o canal bit flip. Este código tem $d_{free} = 10$ e, portanto, pode corrigir até quatro erros clássicos. A correspondente operação de codificação quântica será escrita como:

$$\bigotimes_{t=0}^{+\infty} |u_t\rangle \longmapsto \bigotimes_{t=0}^{+\infty} |v_t^{(1)},\ v_t^{(2)},\ v_t^{(3)}\rangle, \tag{5.22}$$

onde $v_t^{(1)} = u_t + u_{t-2} + u_{t-3}$, $v_t^{(2)} = u_t + u_{t-1} + u_{t-3}$ e $v_t^{(3)} = u_t + u_{t-1} + u_{t-2} + u_{t-3}$. Aqui, definimos $u_{-1} = u_{-2} = u_{-3} = 0$. Cada um dos 2^N estados da base de uma sequência de informação com N qubits será codificado em um estado de comprimento $3(N+3)$. Este CCQ herda todas as propriedades do CCC associado. Ou seja, é um CCQ [3, 1, 3] capaz de garantir a correção de até quatro erros X em quaisquer quatro qubits da palavra-código.

A matriz discreta semi-infinita associada à matriz (5.21) tem a seguinte forma:

$$\mathbf{G} = \begin{bmatrix} 111 & 011 & 101 & 111 & & \\ & 111 & 011 & 101 & 111 & \\ & & \ddots & \ddots & \ddots & \ddots \end{bmatrix}. \tag{5.23}$$

Com um pouco de álgebra é possível demonstrar que uma matriz verificação de paridade associada à matriz (5.21) é:

$$\mathbf{H}(D) = \begin{bmatrix} 1+D+D^3 & 1+D^2+D^3 & 0 \\ 1+D+D^2 & D^2 & 1 \end{bmatrix}, \tag{5.24}$$

que na notação de matriz discreta semi-infinita tem a seguinte forma:

$$\mathbf{H} = \begin{bmatrix} 110 & & & & & & & \\ 101 & & & & & & & \\ 100 & 110 & & & & & & \\ 100 & 101 & & & & & & \\ 010 & 100 & 110 & & & & & \\ 110 & 100 & 101 & & & & & \\ 110 & 010 & 100 & 110 & & & & \\ 000 & 110 & 100 & 101 & & & & \\ & 110 & 010 & 100 & 110 & & & \\ & 000 & 110 & 100 & 101 & & & \\ & & 110 & 010 & 100 & 110 & & \\ & & 000 & 110 & 100 & 101 & & \\ & & & 110 & 010 & 100 & 110 & \\ & & & 000 & 110 & 100 & 101 & \\ & & & & \ddots & \ddots & \ddots & \ddots \\ & & & & \ddots & \ddots & \ddots & \ddots \end{bmatrix}. \tag{5.25}$$

O CSL de codificação e o CSL de decodificação de síndromes são apresentados na Figura 5.6.

As linhas da matriz de paridade (5.25) e da matriz geradora (5.23) permitem-nos

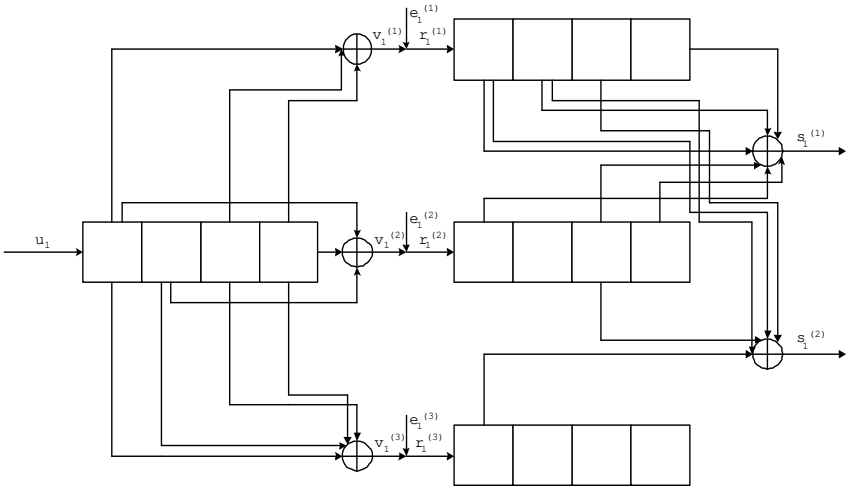

Figura 5.6: CSL de codificação e CSL de decodificação de síndromes para o CCC (3, 1, 3).

escrever, respectivamente, os geradores do grupo estabilizador S_X e as operações lógicas $\overline{X}_i$ atuando sobre os qubits de informação. Estes operadores estão dispostos na Tabela 5.1.

Ao entrarem no circuito da Figura 5.6 os $(N+3)$ blocos recebidos do canal ruidoso, são necessários mais três blocos de zeros para retornar ao estado inicial (todo nulo) do decodificador de síndromes. Assim, para uma sequência de informação com N qubits, precisaremos considerar a medida de $2((N+3)+3) = 2N+12$ observáveis para poder aplicar o ADS clássico. Quanto à manipulação lógica deste código, pode-se afirmar que uma transformação unitária sobre N qubits *lógicos* será implementada através de uma transformação unitária sobre os $3(N+3)$ qubits *físicos*.

Analogamente, podemos estender a operação de codificação (5.22) para um canal phase flip. Na base $\{|0\rangle, |1\rangle\}$, teremos a seguinte operação de codificação:

$$\bigotimes_{t=0}^{+\infty} |u_t\rangle \longmapsto \bigotimes_{t=0}^{+\infty} \left\{ \frac{1}{2\sqrt{2}} (|0\rangle + (-1)^{v_t^{(1)}}|1\rangle)(|0\rangle + (-1)^{v_t^{(2)}}|1\rangle)(|0\rangle + (-1)^{v_t^{(3)}}|1\rangle) \right\}, \quad (5.26)$$

onde $v_t^{(1)} = u_t + u_{t-2} + u_{t-3}$, $v_t^{(2)} = u_t + u_{t-1} + u_{t-3}$, $v_t^{(3)} = u_t + u_{t-1} + u_{t-2} + u_{t-3}$ e $u_{-1} = u_{-2} = u_{-3} = 0$. Ou, mais explicitamente,

M_0	Z	Z	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	$\cdots$	$\cdots$	$\cdots$
M_1	Z	I	Z	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	$\cdots$	$\cdots$	$\cdots$
M_2	Z	I	I	Z	Z	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	$\cdots$	$\cdots$	$\cdots$
M_3	Z	I	I	Z	I	Z	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	$\cdots$	$\cdots$	$\cdots$
M_4	I	Z	I	Z	I	I	Z	Z	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	$\cdots$	$\cdots$	$\cdots$
M_5	Z	Z	I	Z	I	I	Z	I	Z	I	I	I	I	I	I	I	I	I	I	I	I	I	I	$\cdots$	$\cdots$	$\cdots$
M_6	Z	Z	I	I	Z	I	Z	I	I	Z	Z	I	I	I	I	I	I	I	I	I	I	I	I	$\cdots$	$\cdots$	$\cdots$
M_7	I	I	I	Z	Z	I	Z	I	I	Z	I	Z	I	I	I	I	I	I	I	I	I	I	I	$\cdots$	$\cdots$	$\cdots$
M_8	I	I	I	Z	Z	I	I	Z	I	Z	I	I	Z	Z	I	I	I	I	I	I	I	I	I	$\cdots$	$\cdots$	$\cdots$
M_9	I	I	I	I	I	I	Z	Z	I	Z	I	I	Z	I	Z	I	I	I	I	I	I	I	I	$\cdots$	$\cdots$	$\cdots$
M_{10}	I	I	I	I	I	I	Z	Z	I	I	Z	I	Z	I	I	Z	Z	I	I	I	I	I	I	$\cdots$	$\cdots$	$\cdots$
M_{11}	I	I	I	I	I	I	I	I	Z	Z	I	Z	I	I	Z	I	Z	I	I	I	I	I	I	$\cdots$	$\cdots$	$\cdots$
M_{12}	I	I	I	I	I	I	I	I	Z	Z	I	I	Z	I	Z	I	I	Z	Z	I	I	I	I	$\cdots$	$\cdots$	$\cdots$
M_{13}	I	I	I	I	I	I	I	I	I	I	I	Z	Z	I	Z	I	I	Z	I	Z	I	I	I	$\cdots$	$\cdots$	$\cdots$
$\vdots$												$\vdots$		$\vdots$		$\vdots$		$\vdots$								
$\vdots$												$\vdots$		$\vdots$		$\vdots$		$\vdots$								
$M_{\infty-1}$	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	Z	Z	$I.$
M_∞	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	$I.$
$\overline{X}_1$	X	X	X	I	X	X	X	I	X	X	X	X	I	I	I	I	I	I	I	I	I	I	I	$\cdots$	$\cdots$	$\cdots$
$\overline{X}_2$	I	I	I	X	X	X	I	X	X	X	I	X	X	X	X	I	I	I	I	I	I	I	I	$\cdots$	$\cdots$	$\cdots$
$\overline{X}_3$	I	I	I	I	I	I	X	X	X	I	X	X	X	I	X	X	X	X	I	I	I	I	I	$\cdots$	$\cdots$	$\cdots$
$\vdots$										$\vdots$			$\vdots$		$\vdots$		$\vdots$									
$\overline{X}_\infty$	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	X	X	X	I	X	X	X	I	X	X	X	$X.$

Tabela 5.1: Geradores e operações lógicas do CCQ [3, 1, 3] descrito pela operação de codificação (5.22).

$$\bigotimes_{t=0}^{+\infty} |u_t\rangle \longmapsto \bigotimes_{t=0}^{+\infty} \left\{ \frac{1}{2\sqrt{2}} \sum_{(p_t,\, q_t,\, r_t)=(0,\,0,\,0)}^{(1,\,1,\,1)} (-1)^{v_t^{(1)} p_t + v_t^{(2)} q_t + v_t^{(3)} r_t} |p_t,\, q_t,\, r_t\rangle \right\}. \tag{5.27}$$

Assim, cada um dos 2^N estados da base de uma sequência de informação com N qubits será codificado em uma superposição de $2^{3(N+3)}$ estados, cada um dos quais com $3(N+3)$ qubits. Trata-se de um CCQ [3, 1, 3] capaz de garantir a correção de até quatro erros Z em quaisquer quatro qubits da palavra-código.

Os geradores e operações lógicas para o CCQ de canal phase flip podem ser obtidos a

partir da mesma Tabela 5.1, bastando fazer a permutação $Z \leftrightarrow X$.

Com a medida da palavra-código recebida à saída do canal quântico pelo conjunto de observáveis da Tabela 5.1 é possível determinar qual erro eventualmente ocorreu sobre a palavra-código original. A cada instante de tempo t, fazemos a medida com dois novos observáveis. Os resultados destas medidas são um conjunto de autovalores $\{\alpha_t^{(1)}, \alpha_t^{(2)}\}$. Para cada instante de tempo t, existe um homomorfismo entre as síndromes clássicas $\{s_t^{(1)}, s_t^{(2)}\} = \{0,1\}$ e os autovalores $\{\alpha_t^{(1)}, \alpha_t^{(2)}\} = \{+1,-1\}$ estabelecido pelas relações $s_t^{(1)} = (1-\alpha_t^{(1)})/2$ e $s_t^{(2)} = (1-\alpha_t^{(2)})/2$ (t= 0, 1, 2, ...). Vamos agora encontrar as soluções da equação de síndromes para o codificador (5.21).

A forma normal de Smith para a matriz transposta de (5.24) é escrita como:

$$\mathbf{H}^T(D) = \mathbf{L} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}^T, \tag{5.28}$$

onde a inversa de $\mathbf{L}(D)$ é

$$\mathbf{L}^{-1}(D) = \begin{bmatrix} D & 1+D & D \\ 0 & 0 & 1 \\ 1+D^2+D^3 & 1+D+D^3 & 1+D+D^2+D^3 \end{bmatrix}, \tag{5.29}$$

Substituindo a equação (5.28) na equação $\mathbf{s}(D) = \mathbf{e}(D)\mathbf{H}^T(D)$, teremos a seguinte equação de síndromes a resolver:

$$\mathbf{s}(D) = [\mathbf{s}^{(1)}(D), \mathbf{s}^{(2)}(D)] = [\boldsymbol{\varepsilon}^{(1)}(D), \boldsymbol{\varepsilon}^{(2)}(D), \boldsymbol{\varepsilon}^{(3)}(D)] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}^T, \tag{5.30}$$

onde $\boldsymbol{\varepsilon}^{(i)}(D) = \mathbf{e}^{(i)}(D)\mathbf{L}(D)$ para $1 \leq i \leq n = 3$.

Usando as equações $\boldsymbol{\varepsilon}^{(j)}(D) = \boldsymbol{\sigma}^{(j)}(D)$, para $1 \leq j \leq n-k = 2$, e $\boldsymbol{\varepsilon}^{(j)}(D) = \mathbf{t}^{(j-n+k)}(D)$, para $n-k+1 = 3-1+1 = 3 \leq j \leq n = 3$ (onde $\mathbf{t}^{(i)}(D) \equiv \mathbf{t}(D)$ para $1 \leq i \leq k = 1$ é um polinômio arbitrário no anel euclidiano $F[D]$), a solução da equação (5.30) para $\boldsymbol{\varepsilon}^{(j)}(D)$ será:

$$\boldsymbol{\varepsilon}^{(1)}(D) = \mathbf{s}^{(1)}(D), \ \boldsymbol{\varepsilon}^{(2)}(D) = \mathbf{s}^{(2)}(D), \ \boldsymbol{\varepsilon}^{(3)}(D) = \mathbf{t}(D). \tag{5.31}$$

Finalmente os vetores de erro $\mathbf{e}^{(i)}(D)$ em termos das soluções dadas em (5.31) são, usando (5.29), $[\mathbf{e}^{(1)}(D), \mathbf{e}^{(2)}(D), \mathbf{e}^{(3)}(D)] = [\mathbf{s}^{(1)}(D), \mathbf{s}^{(2)}(D), \mathbf{t}(D)]\mathbf{L}^{-1}(D)$. Isto produz:

$$\begin{aligned}
\mathbf{e}^{(1)}(D) &= D\mathbf{s}^{(1)}(D) + \mathbf{t}(D) + D^2\mathbf{t}(D) + D^3\mathbf{t}(D), \\
\mathbf{e}^{(2)}(D) &= \mathbf{s}^{(1)}(D) + D\mathbf{s}^{(1)}(D) + \mathbf{t}(D) + D\mathbf{t}(D) + D^3\mathbf{t}(D), \\
\mathbf{e}^{(3)}(D) &= D\mathbf{s}^{(1)}(D) + \mathbf{s}^{(2)}(D) + \mathbf{t}(D) + D\mathbf{t}(D) + D^2\mathbf{t}(D) + D^3\mathbf{t}(D),
\end{aligned} \tag{5.32}$$

onde:

$$\begin{aligned}
\mathbf{s}^{(1)}(D) &= \frac{1-\alpha_0^{(1)}}{2} + \frac{1-\alpha_1^{(1)}}{2}D + \frac{1-\alpha_2^{(1)}}{2}D^2 + \ldots, \\
\mathbf{s}^{(2)}(D) &= \frac{1-\alpha_0^{(2)}}{2} + \frac{1-\alpha_1^{(2)}}{2}D + \frac{1-\alpha_2^{(2)}}{2}D^2 + \ldots,
\end{aligned} \tag{5.33}$$

são as soluções gerais da equação de síndromes $\mathbf{s}(D) = \mathbf{e}(D)\mathbf{H}^T(D)$ para o codificador (3, 1, 3) com matriz geradora (5.21).

$D\mathbf{t}(D)$, $D^2\mathbf{t}(D)$, $D^3\mathbf{t}(D)$	$\mathbf{t}(D)$=0	$\mathbf{t}(D)$=1
$a = 000$	$a = 000$	$c = 100$
$b = 001$	$a = 000$	$c = 100$
$c = 010$	$b = 001$	$d = 101$
$d = 011$	$b = 001$	$d = 101$
$a = 100$	$a = 010$	$c = 110$
$b = 101$	$a = 010$	$c = 110$
$c = 110$	$b = 011$	$d = 111$
$d = 111$	$b = 011$	$d = 111$

Tabela 5.2: Tabela de estados do registro de deslocamento para $\mathbf{t}(D)$.

Os valores de $\mathbf{t}(D)$, $D\mathbf{t}(D)$, $D^2\mathbf{t}(D)$ e $D^3\mathbf{t}(D)$ ao longo da treliça podem ser obtidos através da Tabela 5.2. Definimos o estado inicial como $(D\mathbf{t}(D), D^2\mathbf{t}(D), D^3\mathbf{t}(D)) = (0,0,0)$

e $\mathbf{s}^{(1)}(D) = \mathbf{s}^{(2)}(D) = 0$ (ou $\alpha^{(1)} = \alpha^{(2)} = +1$) antes do estágio 0. O ADS então seleciona o caminho no diagrama de treliça com o menor peso de Hamming.

Com este algoritmo clássico podemos detectar e corrigir, sem ambiguidade, qualquer grupo com até quatro erros X sobre a palavra-código (5.22) ou qualquer grupo com até quatro erros Z sobre a palavra-código (5.26).

Considere agora o CCC (9, 3, 1) construído a partir da matriz geradora (5.23). Neste caso, esta matriz pode ser reescrita da seguinte forma:

$$\mathbf{G} = \begin{bmatrix} 111011101 & 111000000 & & \\ 000111011 & 101111000 & & \\ 000000111 & 011101111 & & \\ & 111011101 & 111000000 & \\ & 000111011 & 101111000 & \\ & 000000111 & 011101111 & \\ & & \ddots\ddots\ddots & \ddots\ddots\ddots \\ & & \ddots\ddots\ddots & \ddots\ddots\ddots \\ & & \ddots\ddots\ddots & \ddots\ddots\ddots \end{bmatrix}. \tag{5.34}$$

A operação de codificação para o CCQ [9, 3, 1] será escrita como:

$$\bigotimes_{t=0}^{+\infty} |u_t^{(1)}, u_t^{(2)}, u_t^{(3)}\rangle \longmapsto \bigotimes_{t=0}^{+\infty} |v_t^{(1)}, v_t^{(2)}, v_t^{(3)}, v_t^{(4)}, v_t^{(5)}, v_t^{(6)}, v_t^{(7)}, v_t^{(8)}, v_t^{(9)}\rangle \tag{5.35}$$

onde $v_t^{(1)} = u_t^{(1)} + u_{t-1}^{(1)} + u_{t-1}^{(2)}$, $v_t^{(2)} = u_t^{(1)} + u_{t-1}^{(1)} + u_{t-1}^{(3)}$, $v_t^{(3)} = u_t^{(1)} + u_{t-1}^{(1)} + u_{t-1}^{(2)} + u_{t-1}^{(3)}$, $v_t^{(4)} = u_t^{(2)} + u_{t-1}^{(2)} + u_{t-1}^{(3)}$, $v_t^{(5)} = u_t^{(1)} + u_t^{(2)} + u_{t-1}^{(2)}$, $v_t^{(6)} = u_t^{(1)} + u_t^{(2)} + u_{t-1}^{(2)} + u_{t-1}^{(3)}$, $v_t^{(7)} = u_t^{(1)} + u_t^{(3)} + u_{t-1}^{(3)}$, $v_t^{(8)} = u_t^{(2)} + u_t^{(3)} + u_{t-1}^{(3)}$ e $v_t^{(9)} = u_t^{(1)} + u_t^{(2)} + u_t^{(3)} + u_{t-1}^{(3)}$. Aqui, definimos $u_{-1}^{(1)} = u_{-1}^{(2)} = u_{-1}^{(3)} = 0$. Esta operação de codificação quântica é a equivalente trivial com taxa $3/9$ e memória unitária da operação (5.22). Por serem equivalentes, a capacidade de correção do novo código é a mesma, ou seja, podemos garantir com o CCQ [9, 3, 1] a correção de qualquer grupo com até quatro erros X.

A matriz verificação de paridade (5.25) pode agora ser reescrita como:

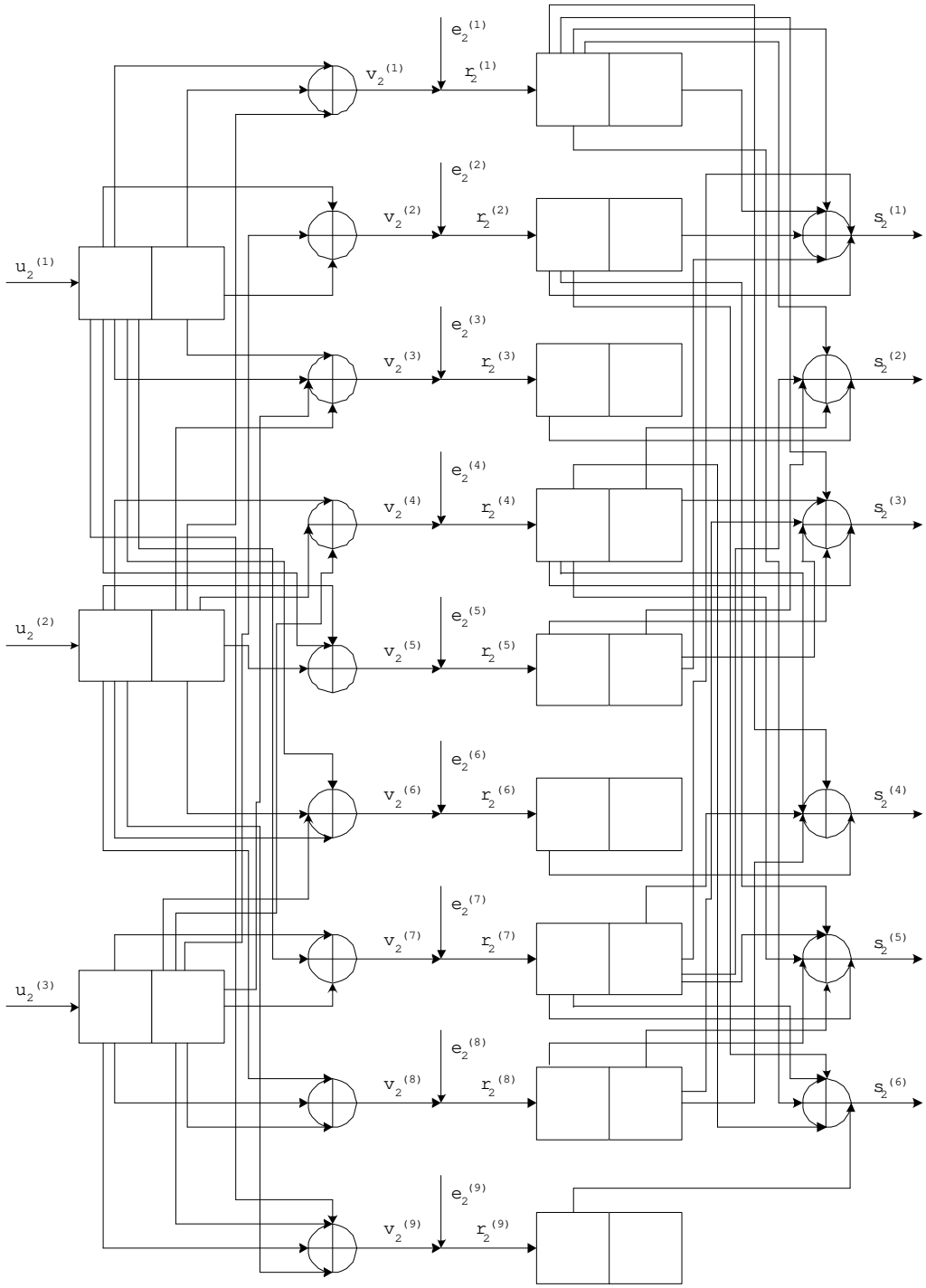

Figura 5.7: CSL de codificação e CSL de decodificação de síndromes para o CCC (9, 3, 1).

$$\mathbf{H} = \begin{bmatrix} 110000000 & & & \\ 101000000 & & & \\ 100110000 & & & \\ 100101000 & & & \\ 010100110 & & & \\ 110100101 & & & \\ 110010100 & 110000000 & & \\ 000110100 & 101000000 & & \\ 000110010 & 100110000 & & \\ 000000110 & 100101000 & & \\ 000000110 & 010100110 & & \\ 000000000 & 110100101 & & \\ & \ddots\ddots\ddots & \ddots\ddots\ddots & \\ & \ddots\ddots\ddots & \ddots\ddots\ddots & \\ & \ddots\ddots\ddots & \ddots\ddots\ddots & \\ & \ddots\ddots\ddots & \ddots\ddots\ddots & \\ & \ddots\ddots\ddots & \ddots\ddots\ddots & \\ & \ddots\ddots\ddots & \ddots\ddots\ddots & \end{bmatrix}. \tag{5.36}$$

Como as matrizes (5.25) e (5.36) são exatamente a mesma matriz, podemos usar as soluções (5.32) para determinar a sequência de erros $\mathbf{e} = (\mathbf{e}^{(1)}, \mathbf{e}^{(2)}, \mathbf{e}^{(3)}, \mathbf{e}^{(4)}, \mathbf{e}^{(5)}, \mathbf{e}^{(6)}, \mathbf{e}^{(7)}, \mathbf{e}^{(8)}, \mathbf{e}^{(9)})$ da equação de síndromes $\mathbf{s} = \mathbf{e}\mathbf{H}^T$, onde agora temos $\mathbf{s} = (\mathbf{s}^{(1)}, \mathbf{s}^{(2)}, \mathbf{s}^{(3)}, \mathbf{s}^{(4)}, \mathbf{s}^{(5)}, \mathbf{s}^{(6)})$ e $\mathbf{r} = (\mathbf{r}^{(1)}, \mathbf{r}^{(2)}, \mathbf{r}^{(3)}, \mathbf{r}^{(4)}, \mathbf{r}^{(5)}, \mathbf{r}^{(6)}, \mathbf{r}^{(7)}, \mathbf{r}^{(8)}, \mathbf{r}^{(9)})$. Aqui, porém, precisaremos de três vezes mais unidades de tempo em relação ao CCC (3, 1, 3) para realizar a decodificação de uma sequência de bits. Veja o CSL de codificação e o CSL de decodificação de síndromes para o CCC (9, 3, 1) na Figura 5.7.

O CCQ [9, 1, 4] para o canal com erro quântico geral

Quando fazemos a concatenação dos CCCs (3, 1, 3) e (9, 3, 1), obtemos um novo CCC, de taxa 1/9 e quatro memórias efetivas. Veja o CSL do CCC concatenado (9, 1, 4) e o seu respectivo diagrama de estados nas Figuras 5.8 e 5.9.

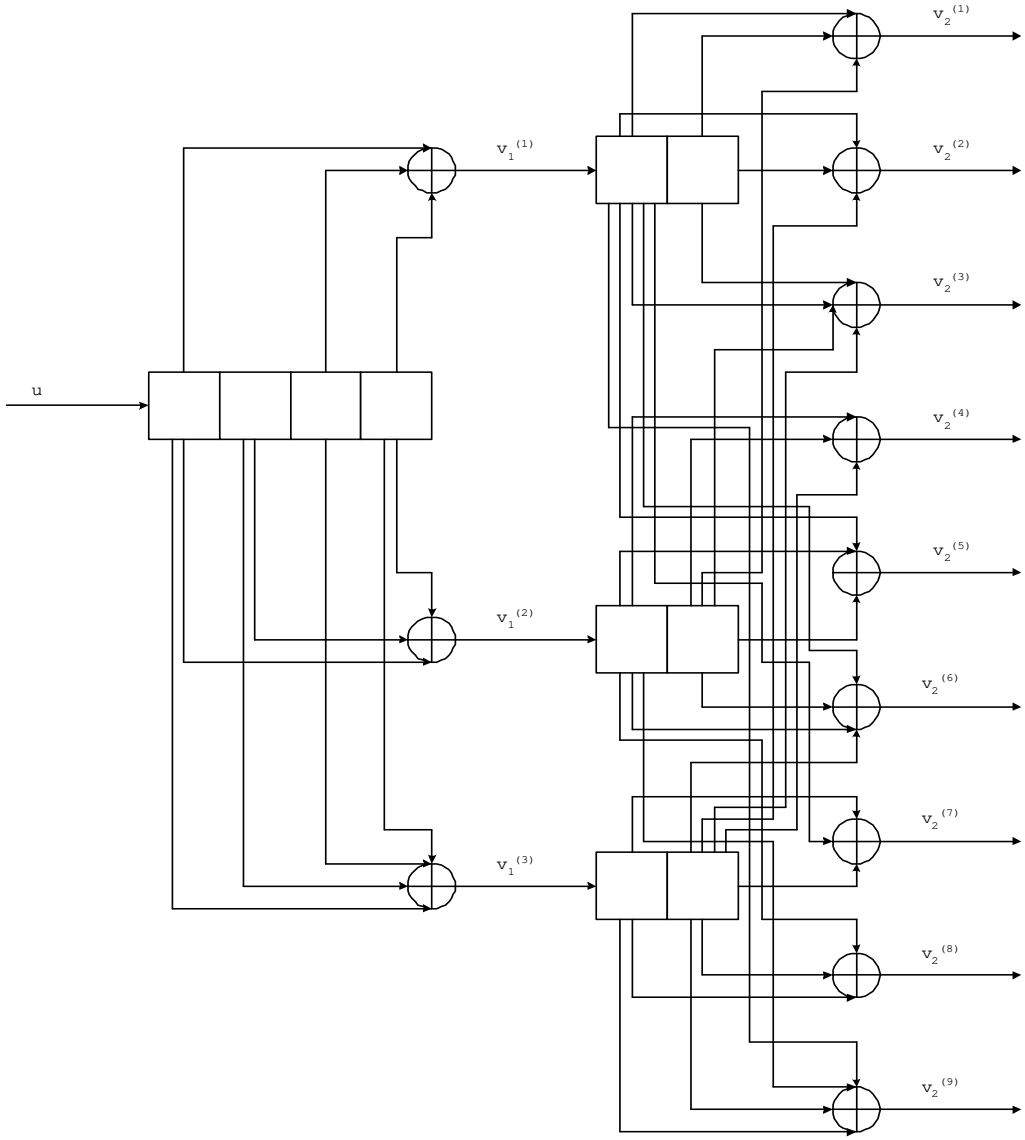

Figura 5.8: Concatenação dos codificadores (3, 1, 3) e (9, 3, 1).

É fácil de verificar que são necessárias pelo menos cinco transições para sair do estado inicial 000000 e retornar ao mesmo estado ($000000 \rightarrow 001111 \rightarrow 010011 \rightarrow 100101 \rightarrow 000111 \rightarrow 000000$). Este caminho também é o caminho de d_{free}. Ou seja, não é possível encontrar outro caminho não nulo que saia do estado inicial 000000 e retorne ao mesmo estado com peso menor do que vinte e quatro (palavra-código $111100001 \rightarrow 001101011 \rightarrow 001001101 \rightarrow 011001110 \rightarrow 001010111$). Logo, este CCC (9, 1, 4) é capaz de corrigir qualquer grupo com até onze erros clássicos sobre a palavra-código.

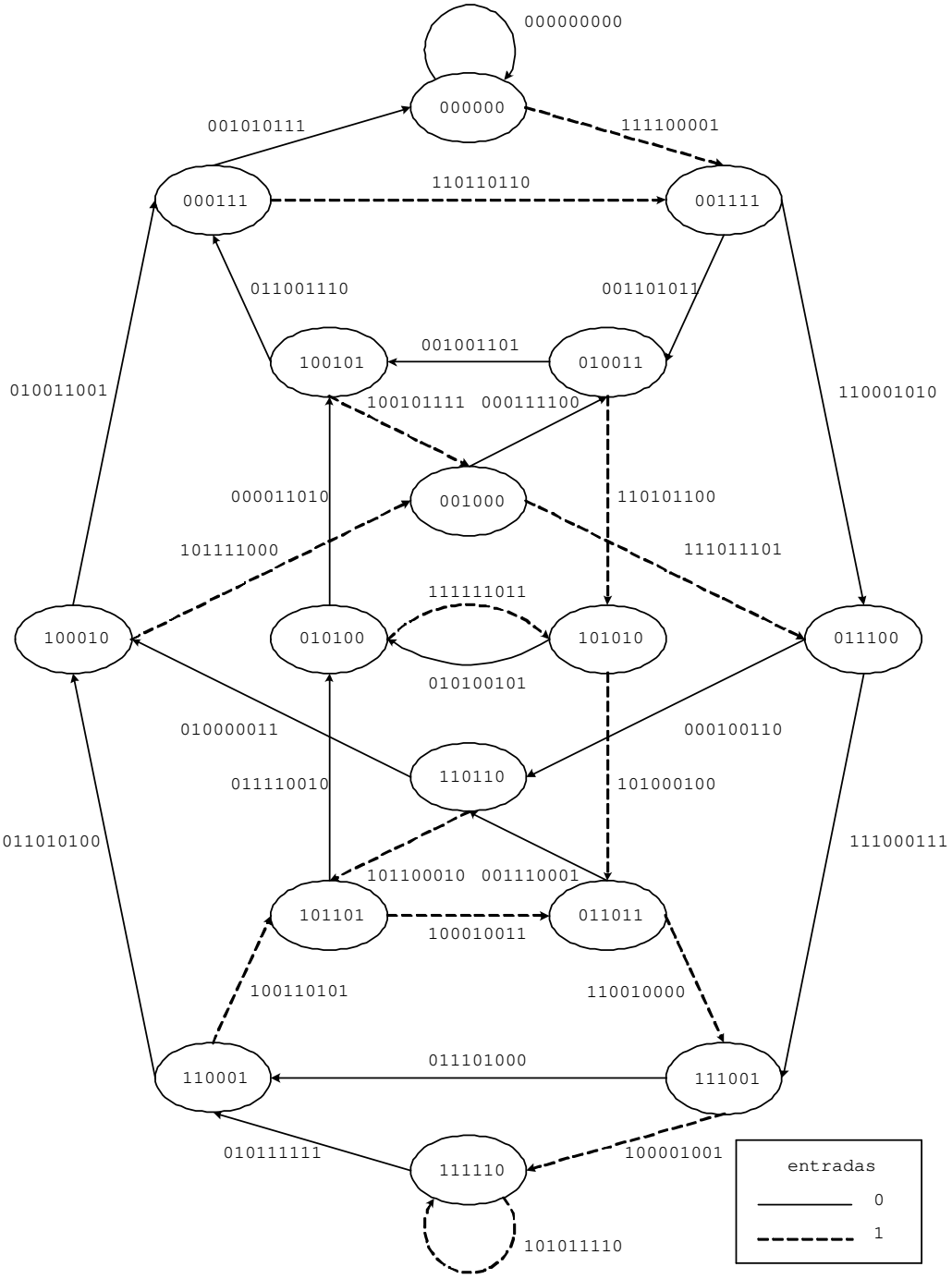

Figura 5.9: Diagrama de estados do CCC concatenado (9, 1, 4).

O CCC (9, 1, 4) pode ser usado na contrução de um CCQ capaz de garantir a correção de até dois erros quânticos gerais com geradores Z e X sobre a palavra-código quântica.

Isto porque os CCCs (3, 1, 3) e (9, 3, 1), associados, respectivamente, à correção de erros de fase e de bit, são capazes de garantir a correção de dois erros e o CCC (9, 1, 4), associado à correção de erros quânticos gerais, é capaz de garantir a correção de oito erros. Todos estes CCCs tem capacidade de correção além da exigida para a correção de dois erros quânticos gerais, o que nos permite afirmar que alguns padrões com mais de dois erros X e Z também poderão ser corrigidos por este código.

A operação de codificação para o CCQ [9, 1, 4] pode ser escrita como:

$$\bigotimes_{t=0}^{+\infty}|u_t\rangle \longmapsto \bigotimes_{t=0}^{+\infty}\left\{\sum_{(p_t,\,q_t,\,r_t)=(0,\,0,\,0)}^{(1,\,1,\,1)} \frac{1}{2\sqrt{2}}(-1)^{v_t^{(1)}p_t+v_t^{(2)}q_t+v_t^{(3)}r_t}|p_t+p_{t-1}+r_{t-1},\right.$$

$$p_t+p_{t-1}+r_{t-1},\ p_t+p_{t-1}+q_{t-1}+r_{t-1},\ q_t+q_{t-1}+r_{t-1},\ p_t+q_t+q_{t-1},$$

$$\left. p_t+q_t+q_{t-1}+r_{t-1},\ p_t+r_t+r_{t-1},\ r_t+q_t+q_{t-1},\ p_t+r_t+q_t+q_{t-1}\rangle\right\}, \quad (5.37)$$

onde $v_t^{(1)}=u_t+u_{t-2}+u_{t-3}$, $v_t^{(2)}=u_t+u_{t-1}+u_{t-3}$ e $v_t^{(3)}=u_t+u_{t-1}+u_{t-2}+u_{t-3}$. Aqui, definimos $u_{-1}=u_{-2}=u_{-3}=0$ e $p_{-1}=q_{-1}=r_{-1}=0$. Cada um dos 2^N estados da base de uma sequência de informação com N qubits será codificado em uma superposição de $2^{3(N+3)}$ estados, cada um dos quais com $3(3(N+3)+3)=9N+36$ qubits.

A matriz geradora polinomial do CCC (9, 1, 4) pode ser obtida através do produto das matrizes geradoras polinomiais dos CCCs (3, 1, 3) e (9, 3, 1). Esta matriz tem a seguinte forma: $\mathbf{G}(D)=[1,\ 1+D^3,\ 1+D+D^2+D^3+D^4,\ 1+D,\ D^4,\ D+D^2+D^3,\ D^2+D^3+D^4,\ D+D^3+D^4,\ 1+D+D^2+D^4]$. Ou, na forma de matriz discreta semi-infinita:

$$\mathbf{G}_C=\begin{bmatrix} 111100001 & 001101011 & 001001101 & 011001110 & 001010111 & & \\ & 111100001 & 001101011 & 001001101 & 011001110 & 001010111 & \\ & & \ddots\,\ddots\,\ddots & \ddots\,\ddots\,\ddots & \ddots\,\ddots\,\ddots & \ddots\,\ddots\,\ddots & \end{bmatrix}. \quad (5.38)$$

Chamemos a matriz (5.36) de $\mathbf{H}_X$, e a matriz (5.25), expandida para o CCQ [9, 1, 4] (basta tomar cada uma das linhas como um bloco de informação a ser codificado pela

matriz (5.34) ($\equiv \mathbf{G}_X$)), de $\mathbf{H}_Z$. Com as matrizes $\mathbf{H}_X$ e $\mathbf{H}_Z$ podemos construir a matriz $\mathbf{H}_C = [\mathbf{H}_X; \mathbf{H}_Z]$:

$$\mathbf{H}_C = \left[\begin{array}{ccccccccc}
110000000 & & & & & & & & \\
101000000 & & & & & & & & \\
100110000 & & & & & & & & \\
100101000 & & & & & & & & \\
010100110 & & & & & & & & \\
110100101 & & & & & & & & \\
110010100 & 110000000 & & & & & & & \\
000110100 & 101000000 & & & & & & & \\
000110010 & 100110000 & & & & & & & \\
000000110 & 100101000 & & & & & & & \\
000000110 & 010100110 & & & & & & & \\
000000000 & 110100101 & & & & & & & \\
 & \ddots\ddots\ddots & \ddots\ddots\ddots & & & & & & \\
 & & \ddots\ddots\ddots & \ddots\ddots\ddots & & & & & \\
 & & & \ddots\ddots\ddots & \ddots\ddots\ddots & & & & \\
 & & & & \ddots\ddots\ddots & \ddots\ddots\ddots & & & \\
 & & & & & \ddots\ddots\ddots & \ddots\ddots\ddots & & \\
111100110 & 010111000 & & & & & & & \\
111011010 & 100101111 & & & & & & & \\
111011101 & 000100110 & 010111000 & & & & & & \\
111011101 & 000011010 & 100101111 & & & & & & \\
000111011 & 010100101 & 000100110 & 010111000 & & & & & \\
111100110 & 101100101 & 000011010 & 100101111 & & & & & \\
111100110 & 010000011 & 010100101 & 000100110 & 010111000 & & & & \\
000000000 & 111100110 & 101100101 & 000011010 & 100101111 & & & & \\
 & \ddots\ddots\ddots & \ddots\ddots\ddots & \ddots\ddots\ddots & \ddots\ddots\ddots & \ddots\ddots\ddots & \ddots\ddots\ddots & &
\end{array}\right]. \quad (5.39)$$

As linhas da matriz (5.39) permitem-nos escrever os geradores do grupo estabilizador do CCQ [9, 1, 4] descrito pela operação de codificação (5.37). Usaremos as linhas da matriz $\mathbf{H}_X$ para obter os geradores Z e as linhas da matriz $\mathbf{H}_Z$ para obter os geradores X.

Os 0s e 1s da matriz $\mathbf{H}_X$ devem ser substituídos por operadores I e Z e os 0s e 1s da matriz $\mathbf{H}_Z$ por operadores I e X. Usaremos também as linhas da matriz $\mathbf{G}_C$ para determinar as operações lógicas tanto de $\overline{X}$ quanto de $\overline{Z}$ atuando sobre os qubits de informação. Para determinar $\overline{X}$, os 0s e 1s da matriz (5.38) devem ser substituídos por operadores I e Z, e para determinar $\overline{Z}$, os 0s e 1s da matriz (5.38) devem ser substituídos por operadores I e X. Pode-se afirmar que uma transformação unitária sobre N qubits *lógicos* será implementada através de uma transformação unitária sobre $9N+36$ qubits *físicos*. Devido às limitações de espaço, não foi possível reproduzir aqui a tabela com os geradores e operações lógicas do CCQ [9, 1, 4].

Ao entrarem no circuito da Figura 5.7 os $(N+4)$ blocos recebidos do canal ruidoso, é necessário mais um bloco de zeros para retornar ao estado inicial (todo nulo) do decodificador de síndromes. Assim, para uma sequência de informação com N qubits, precisaremos considerar a medida de $6((N+4)+1) = 6N+30$ observáveis Z e de $2N+12$ observáveis X para poder aplicar o ADS clássico. Usamos os autovalores $\{\alpha_{M_X}\}$ e $\{\alpha_{M_Z}\}$ (onde M_X e M_Z são os observáveis Z e X, respectivamente) nas soluções (5.32) da equação de síndromes para detectar através de *dois* diagramas de treliça os qubits onde eventualmente ocorreram erros de bit e os *blocos* do código phase flip associado onde eventualmente ocorreram erros de fase.

5.3.2 Um CCQ [9, 1, 3]

Os CCQs [3, 1, 2] e [9, 3, 1] para os canais bit flip e phase flip

Considere que façamos uso do CCC (3, 1, 2) com matriz geradora polinomial

$$\mathbf{G}(D) = [1+D^2,\ 1+D+D^2,\ 1+D+D^2] \tag{5.40}$$

na construção de um CCQ para o canal bit flip. Este código tem $d_{free} = 8$ e, portanto, pode corrigir até três erros clássicos[8]. A correspondente operação de codificação quântica será escrita como:

[8] Lembremo-nos de que já estudamos este codificador na seção 3.6.2.

$$\bigotimes_{t=0}^{+\infty} |u_t\rangle \longmapsto \bigotimes_{t=0}^{+\infty} |v_t^{(1)},\, v_t^{(2)},\, v_t^{(3)}\rangle, \tag{5.41}$$

onde $v_t^{(1)} = u_t + u_{t-2}$, $v_t^{(2)} = u_t + u_{t-1} + u_{t-2}$ e $v_t^{(3)} = u_t + u_{t-1} + u_{t-2}$. Aqui, definimos $u_{-1} = u_{-2} = 0$. Cada um dos 2^N estados da base de uma sequência de informação com N qubits será codificado em um estado de comprimento $3(N+2)$. Este CCQ herda todas as propriedades do CCC associado. Ou seja, é um CCQ [3, 1, 2] capaz de garantir a correção de até três erros X em quaisquer três qubits da palavra-código.

A matriz discreta semi-infinita associada à matriz (5.40) tem a seguinte forma:

$$\mathbf{G} = \begin{bmatrix} 111 & 011 & 111 & & \\ & 111 & 011 & 111 & \\ & & \ddots & \ddots & \ddots \end{bmatrix}. \tag{5.42}$$

Com um pouco de álgebra é possível demonstrar que uma matriz verificação de paridade associada à matriz (5.40) é:

$$\mathbf{H}(D) = \begin{bmatrix} 1+D+D^2 & 1+D^2 & 0 \\ 1+D+D^2 & D^2 & 1 \end{bmatrix}, \tag{5.43}$$

que na notação de matriz discreta semi-infinita tem a seguinte forma:

$$\mathbf{H} = \begin{bmatrix} 110 & & & & & \\ 101 & & & & & \\ 100 & 110 & & & & \\ 100 & 101 & & & & \\ 110 & 100 & 110 & & & \\ 110 & 100 & 101 & & & \\ & 110 & 100 & 110 & & \\ & 110 & 100 & 101 & & \\ & & 110 & 100 & 110 & \\ & & 110 & 100 & 101 & \\ & & & \ddots & \ddots & \ddots \\ & & & \ddots & \ddots & \ddots \end{bmatrix}. \tag{5.44}$$

O CSL de codificação e o CSL de decodificação de síndromes são apresentados na Figura 5.10.

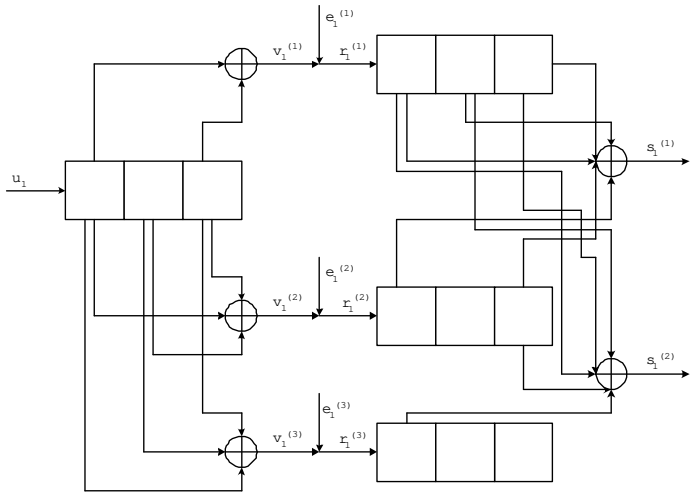

Figura 5.10: CSL de codificação e CSL de decodificação de síndromes para o CCC (3, 1, 2).

As linhas da matriz de paridade (5.44) e da matriz geradora (5.42) permitem-nos escrever, respectivamente, os geradores do grupo estabilizador S_X e as operações lógicas $\overline{X}_i$ atuando sobre os qubits de informação. Estes operadores estão dispostos na Tabela 5.3.

Ao entrarem no circuito da Figura 5.10 os $(N+2)$ blocos recebidos do canal ruidoso, são necessários mais dois blocos de zeros para retornar ao estado inicial (todo nulo) do decodificador de síndromes. Assim, para uma sequência de informação com N qubits, precisaremos considerar a medida de $2((N+2)+2) = 2N+8$ observáveis para poder aplicar o ADS clássico. Quanto à manipulação lógica deste código, pode-se afirmar que uma transformação unitária sobre N qubits *lógicos* será implementada através de uma transformação unitária sobre os $3(N+2)$ qubits *físicos*.

Analogamente, podemos estender a operação de codificação (5.41) para um canal phase flip. Na base $\{|0\rangle, |1\rangle\}$, teremos a seguinte operação de codificação:

$$\bigotimes_{t=0}^{+\infty} |u_t\rangle \longmapsto \bigotimes_{t=0}^{+\infty} \left\{ \frac{1}{2\sqrt{2}} (|0\rangle + (-1)^{v_t^{(1)}} |1\rangle)(|0\rangle + (-1)^{v_t^{(2)}} |1\rangle)(|0\rangle + (-1)^{v_t^{(3)}} |1\rangle) \right\}, \quad (5.45)$$

M_0	Z	Z	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	$\cdots$	$\cdots$	$\cdots$
M_1	Z	I	Z	I	I	I	I	I	I	I	I	I	I	I	I	I	I	$\cdots$	$\cdots$	$\cdots$
M_2	Z	I	I	Z	Z	I	I	I	I	I	I	I	I	I	I	I	I	$\cdots$	$\cdots$	$\cdots$
M_3	Z	I	I	Z	I	Z	I	I	I	I	I	I	I	I	I	I	I	$\cdots$	$\cdots$	$\cdots$
M_4	Z	Z	I	Z	I	I	Z	Z	I	I	I	I	I	I	I	I	I	$\cdots$	$\cdots$	$\cdots$
M_5	Z	Z	I	Z	I	I	Z	I	Z	I	I	I	I	I	I	I	I	$\cdots$	$\cdots$	$\cdots$
M_6	I	I	I	Z	Z	I	Z	I	I	Z	Z	I	I	I	I	I	I	$\cdots$	$\cdots$	$\cdots$
M_7	I	I	I	Z	Z	I	Z	I	I	Z	I	Z	I	I	I	I	I	$\cdots$	$\cdots$	$\cdots$
M_8	I	I	I	I	I	I	Z	Z	I	Z	I	I	Z	Z	I	I	I	$\cdots$	$\cdots$	$\cdots$
M_9	I	I	I	I	I	I	Z	Z	I	Z	I	I	Z	I	Z	I	I	$\cdots$	$\cdots$	$\cdots$
$\vdots$										$\vdots$			$\vdots$			$\vdots$				
$\vdots$										$\vdots$			$\vdots$			$\vdots$				
$M_{\infty-1}$	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	Z	Z	$I.$
M_∞	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	Z	Z	$I.$
$\overline{X}_1$	X	X	X	I	X	X	X	X	X	I	I	I	I	I	I	I	I	$\cdots$	$\cdots$	$\cdots$
$\overline{X}_2$	I	I	I	X	X	X	I	X	X	X	X	X	I	I	I	I	I	$\cdots$	$\cdots$	$\cdots$
$\overline{X}_3$	I	I	I	I	I	I	X	X	X	I	X	X	X	X	X	I	I	I	$\cdots$	$\cdots$	$\cdots$
$\vdots$										$\vdots$			$\vdots$			$\vdots$				
$\overline{X}_\infty$	I	I	I	I	I	I	I	I	I	I	I	I	X	X	X	I	X	X	X	X	$X.$

Tabela 5.3: Geradores e operações lógicas do CCQ [3, 1, 2] descrito pela operação de codificação (5.41).

onde $v_t^{(1)} = u_t + u_{t-2}$, $v_t^{(2)} = u_t + u_{t-1} + u_{t-2}$, $v_t^{(3)} = u_t + u_{t-1} + u_{t-2}$ e $u_{-1} = u_{-2} = 0$. Ou, mais explicitamente,

$$\bigotimes_{t=0}^{+\infty} |u_t\rangle \longmapsto \bigotimes_{t=0}^{+\infty} \left\{ \frac{1}{2\sqrt{2}} \sum_{(p_t,\, q_t,\, r_t)=(0,\,0,\,0)}^{(1,\,1,\,1)} (-1)^{v_t^{(1)} p_t + v_t^{(2)} q_t + v_t^{(3)} r_t} |p_t,\, q_t,\, r_t\rangle \right\}. \tag{5.46}$$

Assim, cada um dos 2^N estados da base de uma sequência de informação com N qubits será codificado em uma superposição de $2^{3(N+2)}$ estados, cada um dos quais com $3(N+2)$ qubits. Trata-se de um CCQ [3, 1, 2] capaz de garantir a correção de até três erros Z em quaisquer três qubits da palavra-código.

Os geradores e operações lógicas para o CCQ de canal phase flip podem ser obtidos a partir da mesma Tabela 5.3, bastando fazer a permutação $Z \leftrightarrow X$.

Com a medida da palavra-código recebida à saída do canal quântico pelo conjunto de observáveis da Tabela 5.3 é possível determinar qual erro eventualmente ocorreu sobre a palavra-código original.

As soluções gerais da equação de síndromes $\mathbf{s}(D) = \mathbf{e}(D)\mathbf{H}^T(D)$ para o codificador (3, 1, 2) com matriz geradora (5.40), apresentadas na seção 3.6.2, podem agora ser escritas para o contexto quântico como:

$$\begin{aligned}
\mathbf{e}^{(1)}(D) &= D\mathbf{s}^{(1)}(D) + \mathbf{t}(D) + D^2\mathbf{t}(D), \\
\mathbf{e}^{(2)}(D) &= \mathbf{s}^{(1)}(D) + D\mathbf{s}^{(1)}(D) + \mathbf{t}(D) + D\mathbf{t}(D) + D^2\mathbf{t}(D), \\
\mathbf{e}^{(3)}(D) &= D\mathbf{s}^{(1)}(D) + \mathbf{s}^{(2)}(D) + \mathbf{t}(D) + D\mathbf{t}(D) + D^2\mathbf{t}(D),
\end{aligned} \tag{5.47}$$

onde:

$$\begin{aligned}
\mathbf{s}^{(1)}(D) &= \frac{1-\alpha_0^{(1)}}{2} + \frac{1-\alpha_1^{(1)}}{2}D + \frac{1-\alpha_2^{(1)}}{2}D^2 + \ldots, \\
\mathbf{s}^{(2)}(D) &= \frac{1-\alpha_0^{(2)}}{2} + \frac{1-\alpha_1^{(2)}}{2}D + \frac{1-\alpha_2^{(2)}}{2}D^2 + \ldots,
\end{aligned} \tag{5.48}$$

Os valores de $\mathbf{t}(D)$, $D\mathbf{t}(D)$ e $D^2\mathbf{t}(D)$ ao longo da treliça podem ser obtidos através da Tabela 3.1. Definimos o estado inicial como $(D\mathbf{t}(D), D^2\mathbf{t}(D)) = (0,0)$ e $\mathbf{s}^{(1)}(D) = \mathbf{s}^{(2)}(D) = 0$ (ou $\alpha^{(1)} = \alpha^{(2)} = +1$) antes do estágio 0. O ADS então seleciona o caminho na treliça com o menor peso de Hamming.

Com este algoritmo clássico podemos detectar e corrigir, sem ambiguidade, qualquer grupo com até três erros X sobre a palavra-código (5.41) ou qualquer grupo com até três erros Z sobre a palavra-código (5.45).

Considere agora o CCC (9, 3, 1) construído a partir da matriz geradora (5.42). Neste caso, esta matriz geradora pode ser reescrita da seguinte forma:

$$\mathbf{G} = \begin{bmatrix} 111011111 & 000000000 & & \\ 000111011 & 111000000 & & \\ 000000111 & 011111000 & & \\ & 111011111 & 000000000 & \\ & 000111011 & 111000000 & \\ & 000000111 & 011111000 & \\ & & \ddots\ddots\ddots & \ddots\ddots\ddots \\ & & \ddots\ddots\ddots & \ddots\ddots\ddots \\ & & \ddots\ddots\ddots & \ddots\ddots\ddots \end{bmatrix}. \tag{5.49}$$

Note que este CCC (9, 3, 1) é de *memória unitária parcial* (MUP), ao contrário do CCC (9, 3, 1) com matriz geradora (5.34), que é de memória unitária completa (MUC).

A matriz verificação de paridade (5.44) pode agora ser reescrita como:

$$\mathbf{H} = \begin{bmatrix} 110000000 & & \\ 101000000 & & \\ 100110000 & & \\ 100101000 & & \\ 110100110 & & \\ 110100101 & & \\ 000110100 & 110000000 & \\ 000110100 & 101000000 & \\ 000000110 & 100110000 & \\ 000000110 & 100101000 & \\ 000000000 & 110100110 & \\ 000000000 & 110100101 & \\ & \ddots\ddots\ddots & \ddots\ddots\ddots \\ & \ddots\ddots\ddots & \ddots\ddots\ddots \\ & \ddots\ddots\ddots & \ddots\ddots\ddots \\ & \ddots\ddots\ddots & \ddots\ddots\ddots \\ & \ddots\ddots\ddots & \ddots\ddots\ddots \\ & \ddots\ddots\ddots & \ddots\ddots\ddots \end{bmatrix}. \tag{5.50}$$

Veja o CSL de codificação e o CSL de decodificação de síndromes para este CCC (9, 3, 1) na Figura 5.11.

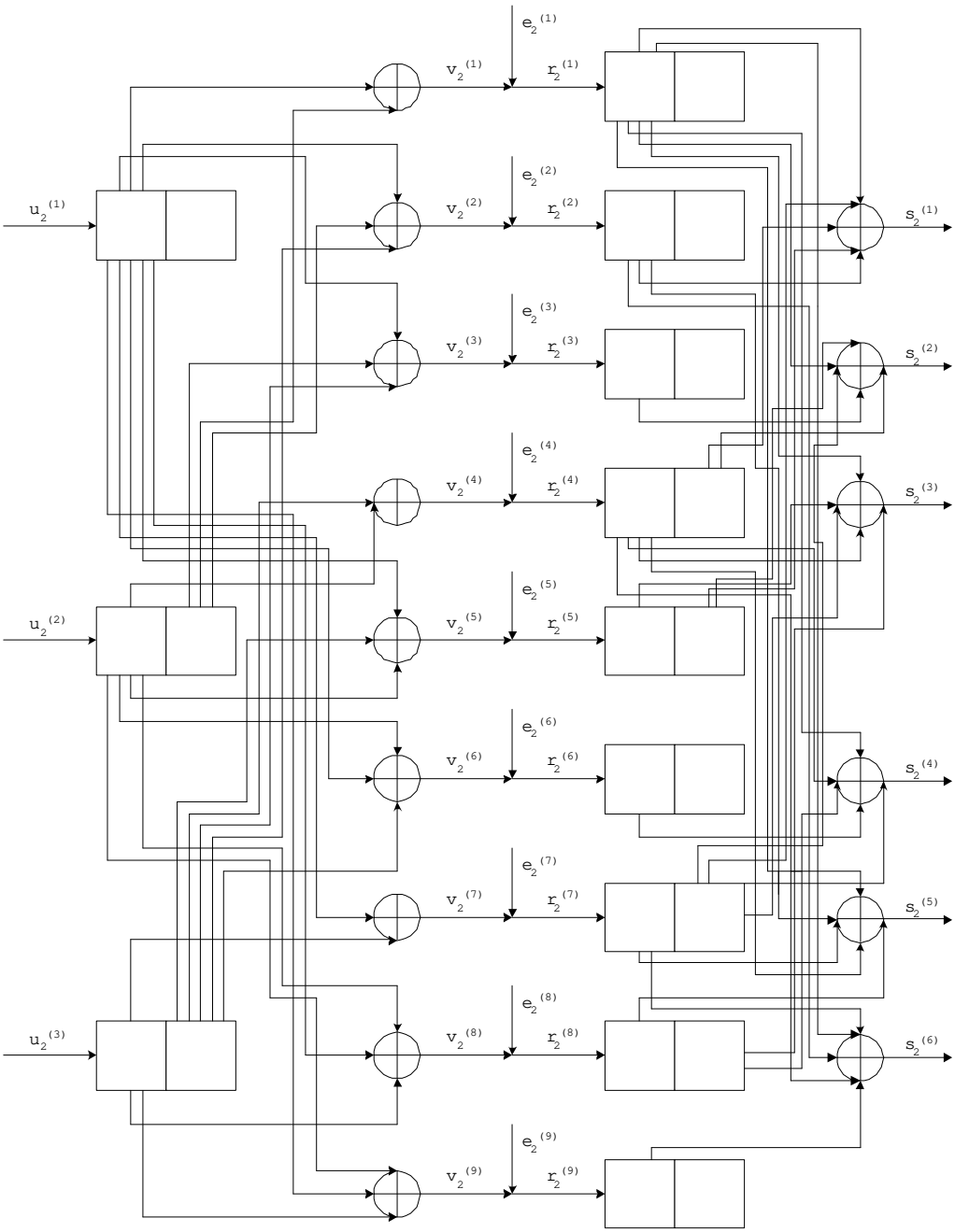

Figura 5.11: CSL de codificação e CSL de decodificação de síndromes para o CCC (9, 3, 1).

A operação de codificação para este CCQ [9, 3, 1] será escrita como:

$$\bigotimes_{t=0}^{+\infty} |u_t^{(1)}, u_t^{(2)}, u_t^{(3)}\rangle \longmapsto \bigotimes_{t=0}^{+\infty} |v_t^{(1)}, v_t^{(2)}, v_t^{(3)}, v_t^{(4)}, v_t^{(5)}, v_t^{(6)}, v_t^{(7)}, v_t^{(8)}, v_t^{(9)}\rangle, \tag{5.51}$$

onde $v_t^{(1)} = u_t^{(1)} + u_{t-1}^{(2)}$, $v_t^{(2)} = u_t^{(1)} + u_{t-1}^{(2)} + u_{t-1}^{(3)}$, $v_t^{(3)} = u_t^{(1)} + u_{t-1}^{(2)} + u_{t-1}^{(3)}$, $v_t^{(4)} = u_t^{(2)} + u_{t-1}^{(3)}$, $v_t^{(5)} = u_t^{(1)} + u_t^{(2)} + u_{t-1}^{(3)}$, $v_t^{(6)} = u_t^{(1)} + u_t^{(2)} + u_{t-1}^{(3)}$, $v_t^{(7)} = u_t^{(1)} + u_t^{(3)}$, $v_t^{(8)} = u_t^{(1)} + u_t^{(2)} + u_t^{(3)}$ e $v_t^{(9)} = u_t^{(1)} + u_t^{(2)} + u_t^{(3)}$. Aqui, definimos $u_{-1}^{(1)} = u_{-1}^{(2)} = u_{-1}^{(3)} = 0$. Esta operação de codificação quântica é a equivalente trivial com taxa $3/9$ e memória unitária da operação (5.41). Por serem equivalentes, a capacidade de correção do novo código é a mesma, ou seja, podemos garantir com este CCQ [9, 3, 1] a correção de qualquer grupo com até três erros X.

O CCQ [9, 1, 3] para o canal com erro quântico geral

Quando fazemos a concatenação dos CCCs (3, 1, 2) e (9, 3, 1), obtemos um novo CCC, de taxa 1/9 e três memórias efetivas. Veja o CSL do CCC concatenado (9, 1, 3) e o seu respectivo diagrama de estados nas Figuras 5.12 e 5.13.

É fácil de verificar que são necessárias pelo menos quatro transições para sair do estado inicial 00000 e retornar ao mesmo estado ($00000 \to 01111 \to 10011 \to 00111 \to 00000$). Este caminho também é o caminho de d_{free}. Ou seja, não há nenhum outro caminho não nulo que saia do estado inicial 00000 e retorne ao mesmo estado com peso menor do que dezoito (palavra-código $111100011 \to 100000100 \to 011011011 \to 100111000$). Logo, este CCC (9, 1, 3) é capaz de corrigir qualquer grupo com até oito erros sobre a palavra-código.

O CCC (9, 1, 3) pode ser usado na contrução de um CCQ capaz de garantir a correção de até dois erros quânticos gerais com geradores Z e X sobre a palavra-código quântica. Isto porque os CCCs (3, 1, 2) e (9, 3, 1), associados respectivamente à correção de erros de fase e de bit, são capazes de garantir a correção de dois erros e o CCC (9, 1, 3), associado à correção de erros quânticos gerais, é capaz de garantir a correção de oito erros. Todos estes CCCs tem capacidade de correção além da exigida para a correção de dois erros quânticos gerais, o que nos permite afirmar que alguns padrões com mais de dois erros X e Z também poderão ser corrigidos por este código.

A operação de codificação para o CCQ [9, 1, 3] pode ser escrita como:

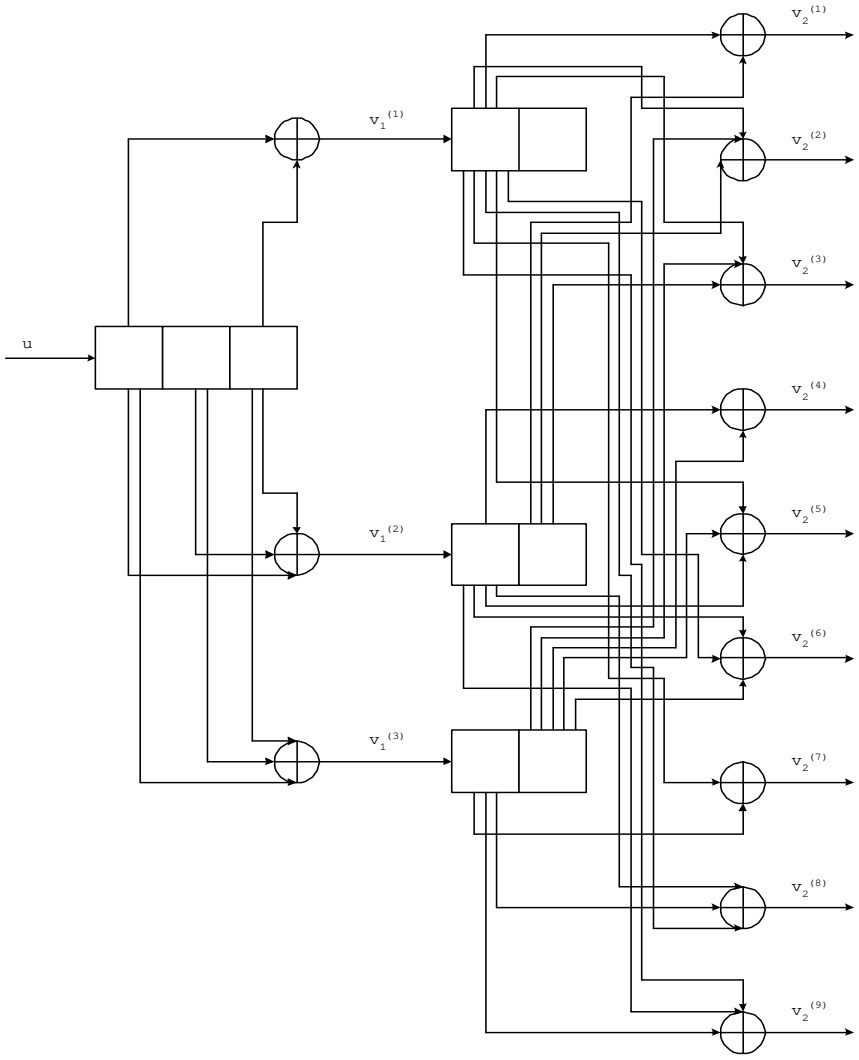

Figura 5.12: Concatenação dos codificadores (3, 1, 2) e (9, 3, 1).

$$\bigotimes_{t=0}^{+\infty}|u_t\rangle \longmapsto \bigotimes_{t=0}^{+\infty}\left\{\sum_{(p_t,q_t,r_t)=(0,0,0)}^{(1,1,1)} \frac{1}{2\sqrt{2}}(-1)^{v_t^{(1)}p_t+v_t^{(2)}q_t+v_t^{(3)}r_t}|p_t+q_{t-1},\ p_t+q_{t-1}+r_{t-1},\right.$$

$$\left.p_t+q_{t-1}+r_{t-1},\ q_t+r_{t-1},\ p_t+q_t+r_{t-1},\ p_t+q_t+r_{t-1},\ p_t+r_t,\ p_t+q_t+r_t,\ p_t+q_t+r_t\rangle\right\},$$

onde $v_t^{(1)} = u_t + u_{t-2}$, $v_t^{(2)} = u_t + u_{t-1} + u_{t-2}$ e $v_t^{(3)} = u_t + u_{t-1} + u_{t-2}$. Aqui, definimos $u_{-1} = u_{-2} = 0$ e $p_{-1} = q_{-1} = r_{-1} = 0$. Cada um dos 2^N estados da base de uma sequência de informação com N qubits será codificado em uma superposição de $2^{3(N+2)}$ estados, cada

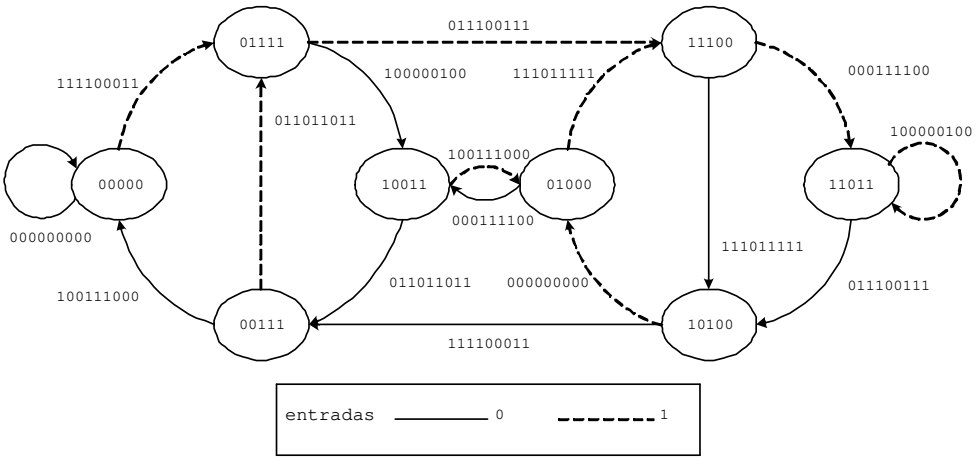

Figura 5.13: Diagrama de estados do CCC concatenado (9, 1, 3).

um dos quais com $3(3(N+2)+3) = 9N+27$ qubits.

A matriz geradora polinomial do CCC (9, 1, 3) pode ser obtida através do produto das matrizes geradoras polinomiais dos CCCs (3, 1, 2) e (9, 3, 1). Esta matriz tem a seguinte forma: $\mathbf{G}(D) = [1+D+D^3,\ 1+D^2,\ 1+D^2,\ 1+D^3,\ D^2+D^3,\ D^2+D^3,\ D,\ 1+D^2,\ 1+D^2]$, ou, na forma discreta semi-infinita:

$$\mathbf{G}_C = \begin{bmatrix} 111100011 & 100000100 & 011011011 & 100111000 & & \\ & 111100011 & 100000100 & 011011011 & 100111000 & \\ & & \ddots\ddots\ddots & \ddots\ddots\ddots & \ddots\ddots\ddots & \end{bmatrix}. \qquad (5.52)$$

Chamemos a matriz (5.50) de $\mathbf{H}_X$, e a matriz (5.44), expandida para o CCQ [9, 1, 3] (basta tomar cada uma das linhas como um bloco de informação a ser codificado pela matriz (5.49) ($\equiv \mathbf{G}_X$)), de $\mathbf{H}_Z$. Com as matrizes $\mathbf{H}_X$ e $\mathbf{H}_Z$ podemos construir a matriz $\mathbf{H}_C = [\mathbf{H}_X;\ \mathbf{H}_Z]$:

$$
\mathbf{H}_C = \begin{bmatrix}
110000000 & & & & \\
101000000 & & & & \\
100110000 & & & & \\
100101000 & & & & \\
110100110 & & & & \\
110100101 & & & & \\
000110100 & 110000000 & & & \\
000110100 & 101000000 & & & \\
000000110 & 100110000 & & & \\
000000110 & 100101000 & & & \\
000000000 & 110100110 & & & \\
000000000 & 110100101 & & & \\
 & \ddots\ddots\ddots & \ddots\ddots\ddots & & \\
 & & \ddots\ddots\ddots & \ddots\ddots\ddots & \\
 & & & \ddots\ddots\ddots & \ddots\ddots\ddots \\
111100100 & 111000000 & & & \\
111011000 & 011111000 & & & \\
111011111 & 111100100 & 111000000 & & \\
111011111 & 111011000 & 011111000 & & \\
111100100 & 000011111 & 111100100 & 111000000 & \\
111100100 & 000011111 & 111011000 & 011111000 & \\
 & \ddots\ddots\ddots & \ddots\ddots\ddots & \ddots\ddots\ddots & \ddots\ddots\ddots
\end{bmatrix}. \tag{5.53}
$$

As linhas da matriz (5.53) permitem-nos escrever os geradores do grupo estabilizador do CCQ [9, 1, 3] descrito pela operação de codificação (5.52). Usaremos as linhas da matriz $\mathbf{H}_X$ para obter os geradores Z e as linhas da matriz $\mathbf{H}_Z$ para obter os geradores X. Os 0s e 1s da matriz $\mathbf{H}_X$ devem ser substituídos por operadores I e Z e os 0s e 1s da matriz $\mathbf{H}_Z$ por operadores I e X. Usaremos também as linhas da matriz $\mathbf{G}_C$ para determinar as operações lógicas tanto de $\overline{X}$ quanto de $\overline{Z}$ atuando sobre os qubits de informação. Para determinar $\overline{X}$, os 0s e 1s da matriz (5.52) devem ser substituídos por operadores I e Z, e para determinar $\overline{Z}$, os 0s e 1s da matriz (5.52) devem ser substituídos por operadores I e X. Pode-se afirmar que uma transformação unitária sobre N qubits *lógicos* será implementada

através de uma transformação unitária sobre $9N+27$ qubits *físicos*. Devido às limitações de espaço, não foi possível reproduzir aqui a tabela com os geradores e operações lógicas do CCQ [9, 1, 3].

Ao entrarem no circuito da Figura 5.11 os $(N+3)$ blocos recebidos do canal ruidoso, é necessário mais um bloco de zeros para retornar ao estado inicial (todo nulo) do decodificador de síndromes. Assim, para uma sequência de informação com N qubits, precisaremos considerar a medida de $6((N+3)+1) = 6N+24$ observáveis Z e de $2N+10$ observáveis X para poder aplicar o ADS clássico. Usamos os autovalores $\{\alpha_{M_X}\}$ e $\{\alpha_{M_Z}\}$ (onde M_X e M_Z são os observáveis Z e X, respectivamente) nas soluções (5.47) da equação de síndromes para detectar através de *dois* diagramas de treliça os qubits onde eventualmente ocorreram erros de bit e os *blocos* do código phase flip associado onde eventualmente ocorreram erros de fase.

N	CCQ [9, 1, 4]	CCQ [9, 1, 3]
1	(4096)[45]	(512)[36]
2	(32768)[54]	(4096)[45]
3	(262144)[63]	(32768)[54]
4	(2097152)[72]	(262144)[63]
5	(16777216)[81]	(2097152)[72]
⋮	⋮	⋮

Tabela 5.4: Comparação entre as complexidades dos CCQs [9, 1, 4] e [9, 1, 3]: (número de estados na superposição)[comprimento do estado].

Estamos agora em condições de comparar a complexidade dos CCQs [9, 1, 4] e [9, 1, 3], ambos códigos com capacidade de correção de até dois erros quânticos gerais sobre a palavra-código. Veja a Tabela 5.4. A complexidade do CCQ [9, 1, 3] para N qubits é exatamente a mesma complexidade do CCQ [9, 1, 4] para $N-1$ qubits. Naturalmente, esta vantagem do CCQ [9, 1, 3] sobre o CCQ [9, 1, 4] torna-se importante para valores grandes de N. A performance destes dois CCQs torna evidente a importância que o uso de codificadores com MUP pode ter na simplificação do processo de geração e de decodificação de CCQs.

5.3.3 Um CCQ [9, 1, 4] de Alta Performance

Os CCQs [9, 1, 4] e [9, 1, 3] estudados nas duas últimas seções foram gerados a partir de um único CCC, respectivamente os CCCs (3, 1, 3) e (3, 1, 2) ótimos. Ambos são de mesma taxa e capacidade de correção, mas de diferentes complexidades. Nesta seção vamos apresentar um CCQ [9, 1, 4] com capacidade de correção de até três erros quânticos gerais, gerado a partir da concatenação do CCC (3, 1, 3) ótimo com um CCC (9, 3, 1) também ótimo, que não pode ser obtido a partir de *nenhum* CCC (3, 1, 3). Há, porém, um preço alto a pagar pela excepcional performance deste código. Deixe-nos expor com mais detalhes tais obstáculos .

O CCQ [9, 3, 1] ótimo para o canal bit flip

Considere o CCC (9, 3, 1) ótimo com matriz geradora polinomial

$$\mathbf{G}(D) = \begin{bmatrix} 1 & 1+D & 1+D & 1+D & 1+D & D & 0 & 0 & 0 \\ D & D & D & 1+D & 1 & 1 & 1 & 1+D & 0 \\ 1 & 1 & D & 1+D & 0 & 1 & D & D & 1+D \end{bmatrix} \tag{5.54}$$

na construção de um CCQ para o canal bit flip. Este código tem $d_{free} = 10$ e é ótimo dentre os CCCs (9, 3, 1).

A matriz discreta semi-infinita associada à matriz (5.54) tem a seguinte forma:

$$\mathbf{G} = \begin{bmatrix} 111 & 110 & 000 & 011 & 111 & 000 & & & \\ 000 & 111 & 110 & 111 & 100 & 010 & & & \\ 110 & 101 & 001 & 001 & 100 & 111 & & & \\ & & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ & & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ & & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \end{bmatrix}. \tag{5.55}$$

Uma matriz verificação de paridade associada à matriz geradora (5.54) é a matriz

$$\mathbf{H}(D) = \begin{bmatrix} 1+D & D & 0 & 1 & D & 0 & 0 & 1 & 0 \\ 0 & 1 & D & 1 & D & 0 & 1 & D & 0 \\ 1 & 0 & 0 & 1+D & 0 & D & 0 & 1+D & 0 \\ 0 & 1 & 0 & D & 1 & 1+D & 0 & D & 0 \\ 0 & D & 1 & D & 1 & 0 & D & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}, \tag{5.56}$$

que na notação de matriz discreta semi-infinita tem a seguinte forma:

$$\mathbf{H} = \begin{bmatrix} 100100010 & & & \\ 010100100 & & & \\ 100100010 & & & \\ 010011000 & & & \\ 001010010 & & & \\ 110001101 & & & \\ 110010000 & 100100010 & & \\ 001010010 & 010100100 & & \\ 000101010 & 100100010 & & \\ 000101010 & 010011000 & & \\ 010100100 & 001010010 & & \\ 000000000 & 110001101 & & \\ & \ddots\ddots\ddots & \ddots\ddots\ddots & \\ & \ddots\ddots\ddots & \ddots\ddots\ddots & \\ & \ddots\ddots\ddots & \ddots\ddots\ddots & \\ & \ddots\ddots\ddots & \ddots\ddots\ddots & \\ & \ddots\ddots\ddots & \ddots\ddots\ddots & \\ & \ddots\ddots\ddots & \ddots\ddots\ddots & \end{bmatrix}. \tag{5.57}$$

O CSL de codificação e o CSL de decodificação de síndromes para o CCC (9, 3, 1) ótimo são apresentados na Figura 5.14.

Como o CCC (9, 3, 1) ótimo não pode ser obtido a partir de nenhum CCC (3, 1, 3), não podemos nos aproveitar das soluções da equação de síndromes de um CCC de taxa

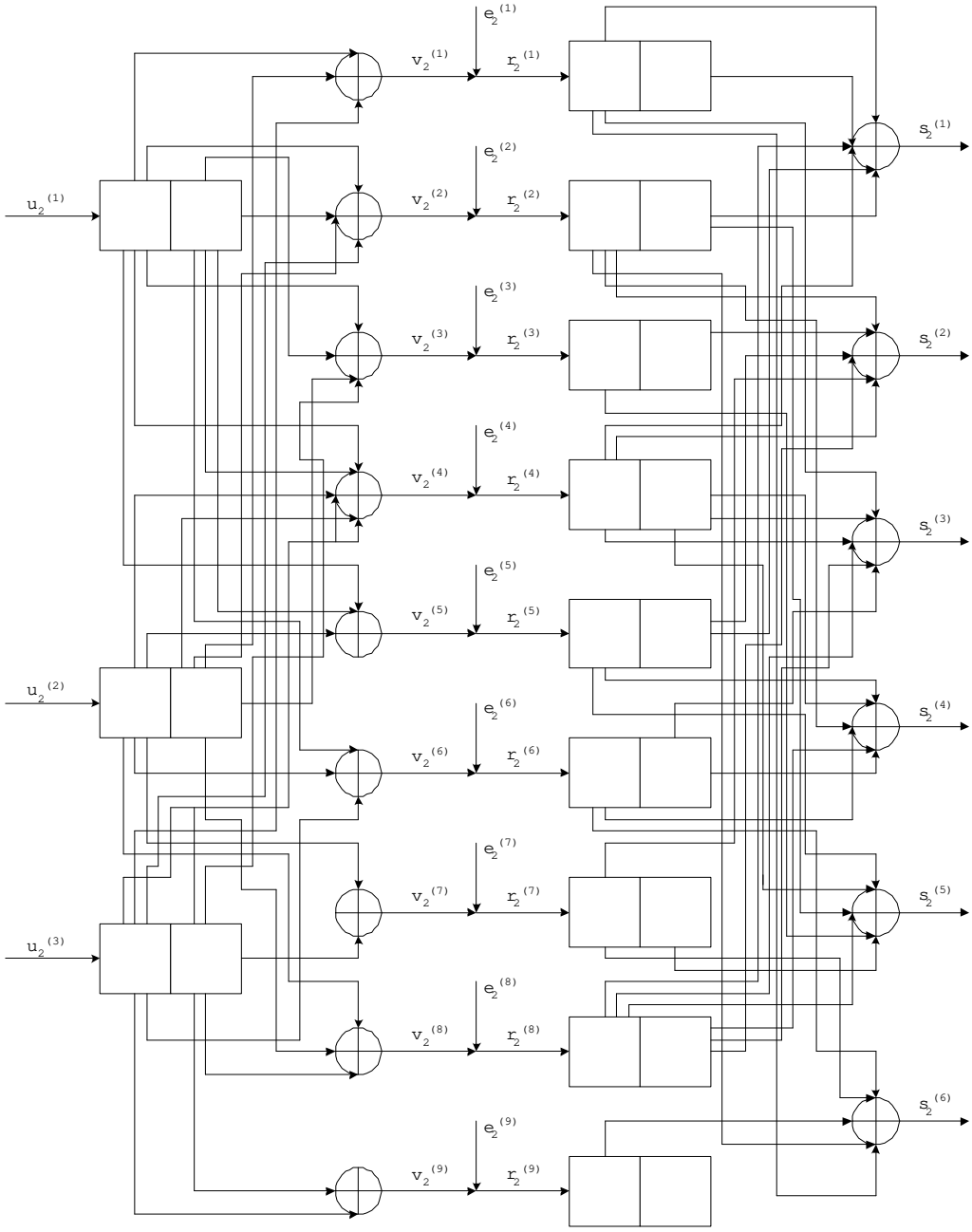

Figura 5.14: CSL de codificação e CSL de decodificação de síndromes para o CCC (9, 3, 1) ótimo.

1/3, que, como vimos na seção 3.6.2, são muito mais fáceis de serem obtidas do que as soluções da equação de síndromes de um CCC de taxa k/n, para $k > 1$ [9]. É a dificuldade

[9]De fato, os elementos (i, j) da i-ésima linha e j-ésima coluna da matriz (5.55), onde $i > 1$ e $j < 3i$, não

de aplicação do ADS clássico para codificadores com taxa k/n que torna a construção do CCQ [9, 1, 4] desta seção uma opção interessante apenas para casos específicos de canais quânticos ruidosos.

Os obstáculos a serem superados são desafiadores, mesmo em termos computacionais. Inicialmente, deve-se colocar a matriz (5.56), de dimensões 6×9, na forma de Smith. Se desejarmos que as soluções das equações de síndromes sejam as mais compactas e simples possíveis, devemos ainda exigir que as matrizes da decomposição de Smith tenham determinante igual a um[10]. Por fim, obtidas as soluções das equações de síndromes, serão estas dependentes de $k = 3$ polinômios arbitrários $t_k(D)$ (k=1, 2, 3) do anel euclidiano $F[D]$ ao longo de uma treliça com $2^{km} = 2^{3.1} = 8$ estados e com $2^k = 2^3 = 8$ ramos de comprimento $n = 9$ saindo de cada estado. Como se tudo isto já não fosse desafiador, poder-se-á chegar a conclusão de que a matriz (5.56), obtida também ao custo da decomposição de Smith da matriz (5.54), pode não ter sido a melhor escolha para a matriz verificação de paridade, já que pode levar a soluções da equação de síndromes extremamente complexas do ponto de vista computacional. Um melhor entendimento das dificuldades envolvidas nestes cálculos pode ser obtido através de uma revisão da seção 3.6.2. Devido a estas dificuldades, não obtivemos as soluções da equação de síndromes para o CCC (9, 3, 1) ótimo. Isto, porém, não nos impede de expor um CCQ [9, 1, 4] gerado a partir de CCCs (3, 1, 3) e (9, 3, 1) ótimos. Caso a aplicação prática deste código se revele, de fato, interessante, não haverá escolha. Dever-se-á buscar as soluções da equação de síndromes para o CCC (9, 3, 1) ótimo.

A operação de codificação para este CCQ [9, 3, 1] pode então ser escrita como:

$$\bigotimes_{t=0}^{+\infty} |u_t^{(1)}, u_t^{(2)}, u_t^{(3)}\rangle \longmapsto \bigotimes_{t=0}^{+\infty} |v_t^{(1)}, v_t^{(2)}, v_t^{(3)}, v_t^{(4)}, v_t^{(5)}, v_t^{(6)}, v_t^{(7)}, v_t^{(8)}, v_t^{(9)}\rangle, \tag{5.58}$$

onde $v_t^{(1)} = u_t^{(1)} + u_{t-1}^{(2)} + u_t^{(3)}$, $v_t^{(2)} = u_t^{(1)} + u_{t-1}^{(1)} + u_{t-1}^{(2)} + u_t^{(3)}$, $v_t^{(3)} = u_t^{(1)} + u_{t-1}^{(1)} + u_{t-1}^{(2)} + u_{t-1}^{(3)}$, $v_t^{(4)} = u_t^{(1)} + u_{t-1}^{(1)} + u_t^{(2)} + u_{t-1}^{(2)} + u_t^{(3)} + u_{t-1}^{(3)}$, $v_t^{(5)} = u_t^{(1)} + u_{t-1}^{(1)} + u_t^{(2)}$, $v_t^{(6)} = u_{t-1}^{(1)} + u_t^{(2)} + u_t^{(3)}$, $v_t^{(7)} = u_t^{(2)} + u_{t-1}^{(3)}$, $v_t^{(8)} = u_t^{(2)} + u_{t-1}^{(2)} + u_{t-1}^{(3)}$ e $v_t^{(9)} = u_t^{(3)} + u_{t-1}^{(3)}$. Aqui, definimos $u_{-1}^{(1)} = u_{-1}^{(2)} =$

são todos nulos.

[10]Isto porque as soluções da equação de síndromes são obtidas a partir da inversa de uma das matrizes da decomposição de Smith de $\mathbf{H}(D)$.

$u_{-1}^{(3)} = 0.$

O CCQ [9, 1, 4] para o canal com erro quântico geral

Deixe-nos descrever como construir o CCQ [9, 1, 4] a partir dos CCCs (3, 1, 3) e (9, 3, 1) ótimos. Quando fazemos a concatenação destes CCCs, obtemos um novo CCC, de taxa 1/9 e com quatro memórias efetivas. Veja o CSL do CCC concatenado (9, 1, 4) e o seu respectivo diagrama de estados nas Figuras 5.15 e 5.16.

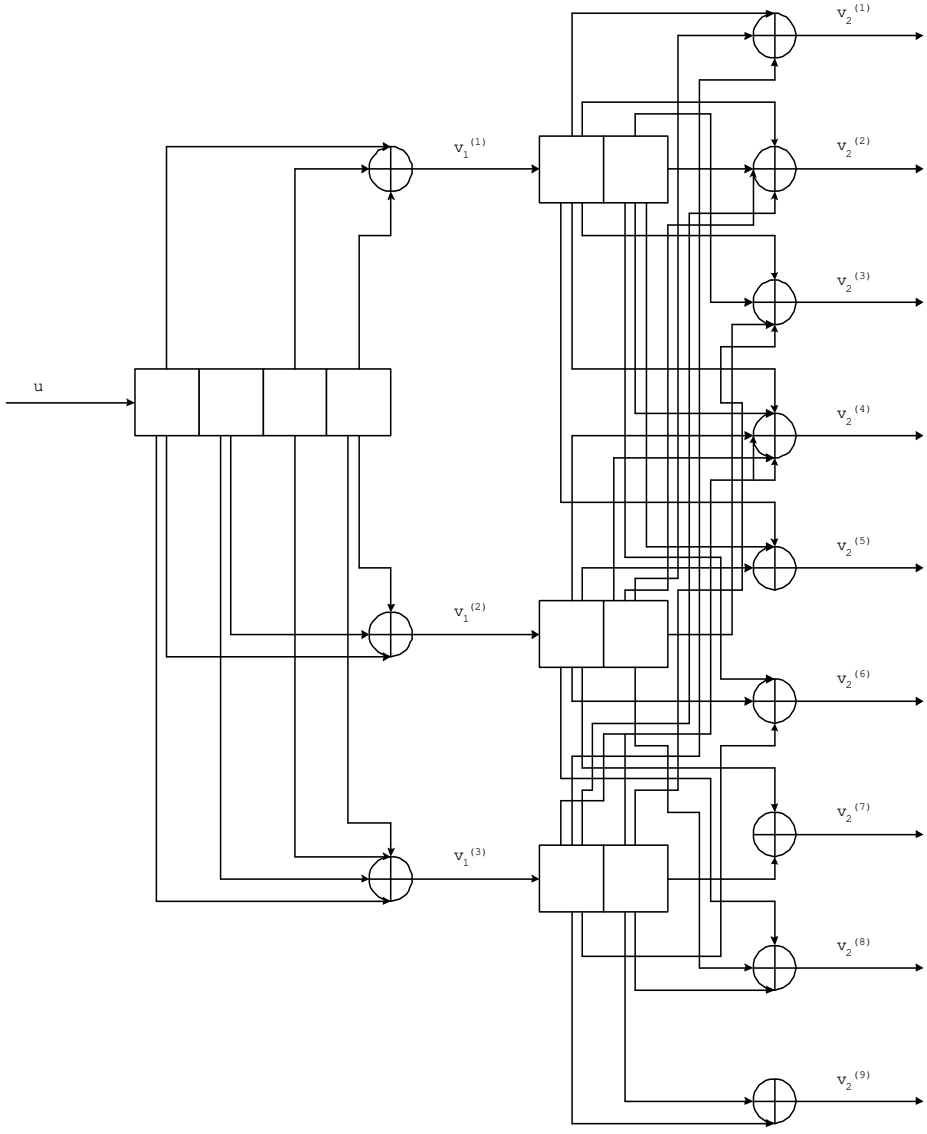

Figura 5.15: Concatenação dos codificadores (3, 1, 3) e (9, 3, 1) ótimos.

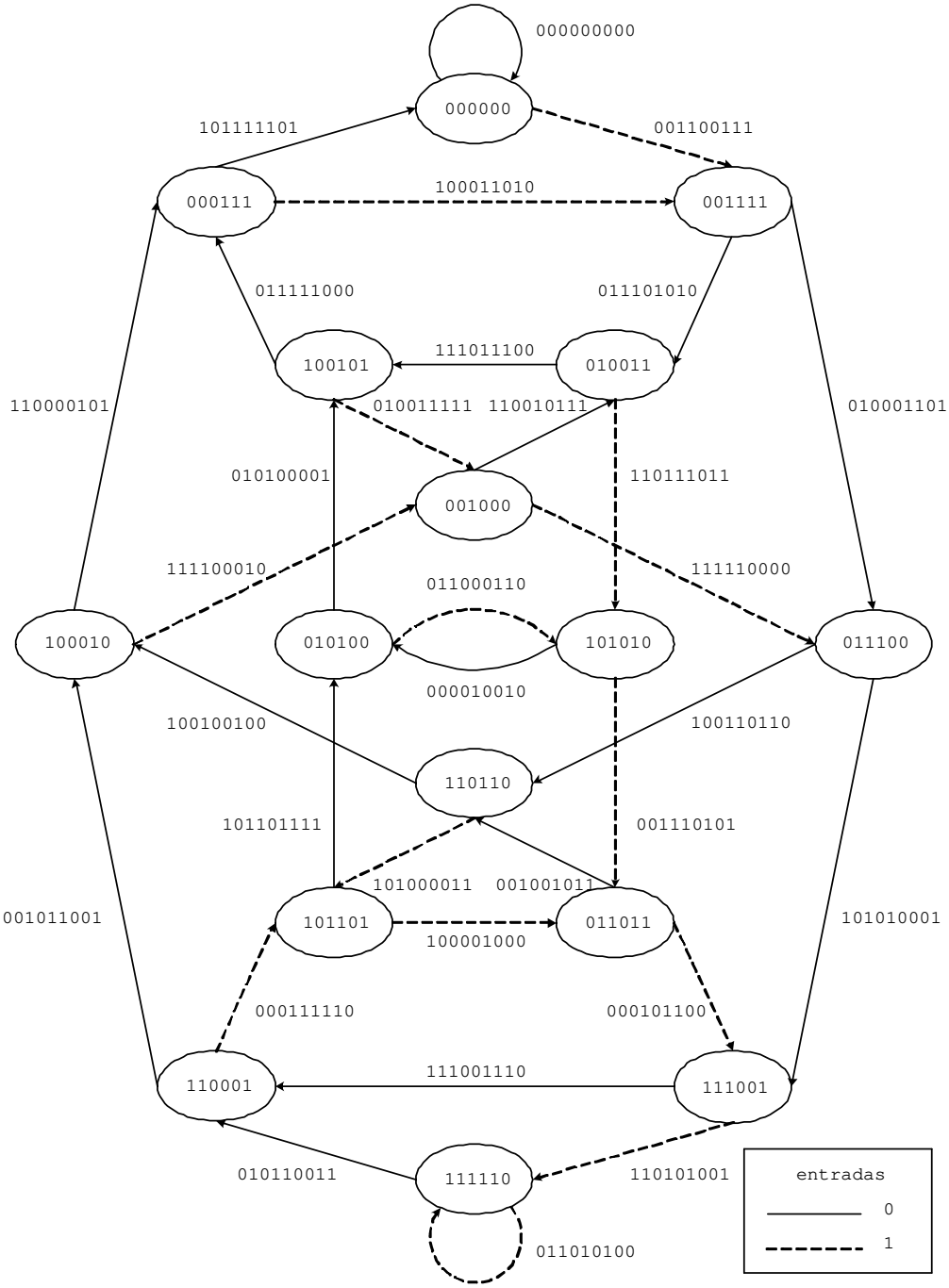

Figura 5.16: Diagrama de estados do CCC concatenado (9, 1, 4).

É fácil de verificar que são necessárias pelo menos cinco transições para sair do estado inicial 000000 e retornar ao mesmo estado ($000000 \rightarrow 001111 \rightarrow 010011 \rightarrow 100101 \rightarrow 000111 \rightarrow 000000$). Este caminho também é o caminho de d_{free}. Não há nenhum outro

caminho não nulo que saia do estado inicial 000000 e retorne ao mesmo estado com peso menor do que vinte e oito (palavra-código $001100111 \rightarrow 011101010 \rightarrow 111011100 \rightarrow 011111000 \rightarrow 101111101$). Logo, este CCC é capaz de corrigir qualquer grupo com até treze erros sobre a palavra-código.

Este CCC (9, 1, 4) pode ser usado na contrução de um CCQ capaz de garantir a correção de até três erros quânticos gerais com geradores Z e X sobre a palavra-código quântica. Isto porque os CCCs (3, 1, 3) e (9, 3, 1), associados respectivamente à correção de erros de fase e de bit, são capazes de garantir a correção de três erros e o CCC (9, 1, 4), associado à correção de erros quânticos gerais, é capaz de garantir a correção de doze erros. Todos estes CCCs tem capacidade de correção além da exigida para a correção de três erros quânticos gerais, o que nos permite afirmar que alguns padrões com mais de três erros X e Z também poderão ser corrigidos por este código.

A operação de codificação para este CCQ [9, 1, 4] pode ser escrita como:

$$\bigotimes_{t=0}^{+\infty} |u_t\rangle \longmapsto \bigotimes_{t=0}^{+\infty} \left\{ \sum_{(p_t,\, q_t,\, r_t)=(0,\,0,\,0)}^{(1,\,1,\,1)} \frac{1}{2\sqrt{2}} (-1)^{v_t^{(1)} p_t + v_t^{(2)} q_t + v_t^{(3)} r_t} | p_t + q_{t-1} + r_t,$$

$$p_t + p_{t-1} + q_{t-1} + r_t,\ p_t + p_{t-1} + q_{t-1} + r_{t-1},\ p_t + p_{t-1} + q_t + q_{t-1} + r_t + r_{t-1},$$

$$p_t + p_{t-1} + q_t,\ p_{t-1} + q_t + r_t,\ q_t + r_{t-1},\ q_t + q_{t-1} + r_{t-1},\ r_t + r_{t-1} \rangle \right\}, \qquad (5.59)$$

onde $v_t^{(1)} = u_t + u_{t-2} + u_{t-3}$, $v_t^{(2)} = u_t + u_{t-1} + u_{t-3}$ e $v_t^{(3)} = u_t + u_{t-1} + u_{t-2} + u_{t-3}$. Aqui, definimos $u_{-1} = u_{-2} = u_{-3} = 0$ e $p_{-1} = q_{-1} = r_{-1} = 0$. Cada um dos 2^N estados da base de uma sequência de informação com N qubits será codificado em uma superposição de $2^{3(N+3)}$ estados, cada um dos quais com $3(3(N+3)+3) = 9N+36$ qubits.

A matriz geradora polinomial do CCC (9, 1, 4) pode ser obtida através do produto das matrizes geradoras polinomiais dos CCCs (3, 1, 3) e (9, 3, 1). Esta matriz tem a seguinte forma: $\mathbf{G}(D) = [D^2+D^4,\ D+D^2+D^3,\ 1+D+D^2+D^3+D^4\ 1+D+D^3+D^4,\ D^2+D^3+D^4,\ D+D^2+D^3+D^4,\ 1+D^2+D^4,\ 1+D,\ 1+D^4]$, ou, na forma de matriz discreta semi-infinita:

$$\mathbf{G}_C = \begin{bmatrix} 001100111 & 011101010 & 111011100 & 011111000 & 101111101 & \\ & 001100111 & 011101010 & 111011100 & 011111000 & 101111101 \\ & & \ddots\ddots\ddots & \ddots\ddots\ddots & \ddots\ddots\ddots & \ddots\ddots\ddots \end{bmatrix}. \quad (5.60)$$

Chamemos a matriz (5.57) de $\mathbf{H}_X$, e a matriz (5.25), expandida para este CCQ [9, 1, 4] (basta tomar cada uma das linhas como um bloco de informação a ser codificado pela matriz (5.55) ($\equiv \mathbf{G}_X$)), de $\mathbf{H}_Z$. Com as matrizes $\mathbf{H}_X$ e $\mathbf{H}_Z$ podemos construir a matriz $\mathbf{H}_C = [\mathbf{H}_X; \mathbf{H}_Z]$. Veja a matriz (5.61).

As linhas da matriz (5.61) permitem-nos escrever os geradores do grupo estabilizador do CCQ [9, 1, 4] descrito pela operação de codificação (5.59). Usaremos as linhas da matriz $\mathbf{H}_X$ para obter os geradores Z e as linhas da matriz $\mathbf{H}_Z$ para obter os geradores X. Os 0s e 1s da matriz $\mathbf{H}_X$ devem ser substituídos por operadores I e Z e os 0s e 1s da matriz $\mathbf{H}_Z$ por operadores I e X. Usaremos também as linhas da matriz $\mathbf{G}_C$ para determinar as operações lógicas tanto de $\overline{X}$ quanto de $\overline{Z}$ atuando sobre os qubits de informação. Para determinar $\overline{X}$, os 0s e 1s da matriz (5.60) devem ser substituídos por operadores I e Z, e para determinar $\overline{Z}$, os 0s e 1s da matriz (5.60) devem ser substituídos por operadores I e X. Uma transformação unitária sobre N qubits *lógicos* será implementada através de uma transformação unitária sobre $9N+36$ qubits *físicos*.

Ao entrarem no circuito da Figura 5.14 os $(N+4)$ blocos recebidos do canal ruidoso, é necessário mais um bloco de zeros para retornar ao estado inicial (todo nulo) do decodificador de síndromes. Assim, para uma sequência de informação com N qubits, precisaremos considerar a medida de $6((N+4)+1) = 6N+30$ observáveis Z e de $2N+12$ observáveis X para poder aplicar o ADS clássico. Os autovalores $\{\alpha_{M_X}\}$, referentes aos observáveis Z, devem ser usados nas soluções da equação de síndromes para o CCC (9, 3, 1) ótimo (que não foram calculadas aqui) para detectar, através de um diagrama de treliça, possíveis erros X sobre os qubits da palavra-código original (5.59) e os autovalores $\{\alpha_{M_Z}\}$, referentes aos observáveis X, devem ser usados nas soluções (5.32) da equação de síndromes para o CCC (3, 1, 3) ótimo para detectar, através de um segundo diagrama de treliça, os blocos do código phase flip associado onde eventualmente ocorreram erros Z.

$$
\mathbf{H}_C = \begin{bmatrix}
100100010 & & & & & & \\
010100100 & & & & & & \\
100100010 & & & & & & \\
010011000 & & & & & & \\
001010010 & & & & & & \\
110001101 & & & & & & \\
110010000 & 100100010 & & & & & \\
001010010 & 010100100 & & & & & \\
000101010 & 100100010 & & & & & \\
000101010 & 010011000 & & & & & \\
010100100 & 001010010 & & & & & \\
000000000 & 110001101 & & & & & \\
 & \ddots\ddots\ddots & \ddots\ddots\ddots & & & & \\
 & & \ddots\ddots\ddots & \ddots\ddots\ddots & & & \\
 & & & \ddots\ddots\ddots & \ddots\ddots\ddots & & \\
 & & & & \ddots\ddots\ddots & \ddots\ddots\ddots & \\
111001110 & 100011010 & & & & & \\
001011001 & 010011111 & & & & & \\
111110000 & 100110110 & 100011010 & & & & \\
111110000 & 010100001 & 010011111 & & & & \\
000111110 & 000010010 & 100110110 & 100011010 & & & \\
111001110 & 011101010 & 010100001 & 010011111 & & & \\
111001110 & 100100100 & 000010010 & 100110110 & 100011010 & & \\
000000000 & 111001110 & 011101010 & 010100001 & 010011111 & & \\
 & \ddots\ddots\ddots & \ddots\ddots\ddots & \ddots\ddots\ddots & \ddots\ddots\ddots & \ddots\ddots\ddots &
\end{bmatrix}. \tag{5.61}
$$

A complexidade do CCQ [9, 1, 4] desta seção, em termos do número e do comprimento dos estados, é, obviamente, a mesma do CCQ [9, 1, 4] construído na seção 5.3.1. Porém, é fácil de perceber que as matrizes geradoras e de verificação de paridade do CCC (9, 1, 4) desta seção possuem maior quantidade de 1's. Isto se refletirá em uma maior quantidade de operadores Z e X nos geradores e operadores lógicos do código quântico, o que pode

acarretar em dificuldades de implementação física (ver, por exemplo, seção 10.6 de [48]). É o preço que se paga pela sua exuberante capacidade de correção.

5.4 Miscelânea de Primitivos (2, 1, m) e (3, 1, m)

Nesta seção mostraremos que a concatenação de codificadores (2, 1, m) e (3, 1, m) pode produzir CCQs com altas taxas e capacidade de correção.

5.4.1 Um CCQ [6, 1, 4]

Considere o CCC (2, 1, 3) com matriz geradora polinomial $\mathbf{G}(D) = [1+D+D^3,\ 1+D+D^2+D^3]$ [11]. Em termos de notação de matriz discreta semi-infinita, esta matriz escreve-se como:

$$\mathbf{G} = \begin{bmatrix} 11 & 11 & 01 & 11 & & & \\ & 11 & 11 & 01 & 11 & & \\ & & 11 & 11 & 01 & 11 & \\ & & & \ddots & \ddots & \ddots & \ddots \end{bmatrix}. \tag{5.62}$$

Este CCC tem $d_{free} = 6$ e, portanto, pode corrigir até dois erros clássicos. Por ser um CCC de taxa 1/2, temos que $\mathbf{H}(D) = [1+D+D^2+D^3,\ 1+D+D^3]$. Em termos de notação de matriz discreta semi-infinita, esta matriz escreve-se como:

$$\mathbf{H} = \begin{bmatrix} 11 & & & & & \\ 11 & 11 & & & & \\ 10 & 11 & 11 & & & \\ 11 & 10 & 11 & 11 & & \\ & 11 & 10 & 11 & 11 & \\ & & \ddots & \ddots & \ddots & \ddots \end{bmatrix}. \tag{5.63}$$

De acordo com (5.1), as soluções da equação de síndromes para este codificador são:

[11] Referência [41], páginas 330-331.

$$\begin{aligned}\mathbf{e}^{(1)}(D) &= \mathbf{s}(D)+D\mathbf{s}(D)+D^2\mathbf{s}(D)+\mathbf{t}(D)+D\mathbf{t}(D)+D^3\mathbf{t}(D),\\ \mathbf{e}^{(2)}(D) &= D^2\mathbf{s}(D)+\mathbf{t}(D)+D\mathbf{t}(D)+D^2\mathbf{t}(D)+D^3\mathbf{t}(D),\end{aligned} \tag{5.64}$$

onde aqui usamos $\alpha_1 = 1+D+D^2$ e $\alpha_2 = D^2$.

O CCQ [6, 3, 1] para o canal bit flip

Faremos uso da matriz (5.62) para construir um CCC (6, 3, 1). Assim, esta matriz pode ser reescrita como:

$$\mathbf{G} = \begin{bmatrix} 111101 & 110000 & & \\ 001111 & 011100 & & \\ 000011 & 110111 & & \\ & 111101 & 110000 & \\ & 001111 & 011100 & \\ & 000011 & 110111 & \\ & & \ddots\ddots\ddots & \ddots\ddots\ddots \\ & & \ddots\ddots\ddots & \ddots\ddots\ddots \\ & & \ddots\ddots\ddots & \ddots\ddots\ddots \end{bmatrix}. \tag{5.65}$$

Estamos agora em condições de gerar o CCQ [6, 3, 1] para um canal bit flip. Para este canal, a operação de codificação será escrita como:

$$\bigotimes_{t=0}^{+\infty} |u_t^{(1)},\, u_t^{(2)},\, u_t^{(3)}\rangle \longmapsto \bigotimes_{t=0}^{+\infty} |v_t^{(1)},\, v_t^{(2)},\, v_t^{(3)}\, v_t^{(4)},\, v_t^{(5)}\, v_t^{(6)}\rangle, \tag{5.66}$$

onde $v_t^{(1)} = u_t^{(1)} + u_{t-1}^{(1)} + u_{t-1}^{(3)}$, $v_t^{(2)} = u_t^{(1)} + u_{t-1}^{(1)} + u_{t-1}^{(2)} + u_{t-1}^{(3)}$, $v_t^{(3)} = u_t^{(1)} + u_t^{(2)} + u_{t-1}^{(2)}$, $v_t^{(4)} = u_t^{(1)} + u_t^{(2)} + u_{t-1}^{(2)} + u_{t-1}^{(3)}$, $v_t^{(5)} = u_t^{(2)} + u_t^{(3)} + u_{t-1}^{(3)}$ e $v_t^{(6)} = u_t^{(1)} + u_t^{(2)} + u_t^{(3)} + u_{t-1}^{(3)}$. Aqui, definimos $u_{-1}^{(1)} = u_{-1}^{(2)} = u_{-1}^{(3)} = 0$. Cada um dos 2^{3N} estados da base de uma sequência de informação com N blocos de qubits será codificado em um estado de comprimento $6(N+1)$. Naturalmente, trata-se de um CCQ [6, 3, 2] capaz de garantir a correção de até dois erros X sobre a palavra-código.

A matriz verificação de paridade (5.63) pode agora ser reescrita como:

$$\mathbf{H} = \begin{bmatrix} 110000 & & & & \\ 111100 & & & & \\ 101111 & & & & \\ 111011 & 110000 & & & \\ 001110 & 111100 & & & \\ 000011 & 101111 & & & \\ & 111011 & 110000 & & \\ & 001110 & 111100 & & \\ & 000011 & 101111 & & \\ & & \ddots\ddots\ddots & \ddots\ddots\ddots & \\ & & \ddots\ddots\ddots & \ddots\ddots\ddots & \\ & & \ddots\ddots\ddots & \ddots\ddots\ddots & \end{bmatrix}. \tag{5.67}$$

As linhas da matriz de paridade (5.67) e da matriz geradora (5.65) permitem-nos escrever, respectivamente, os geradores do grupo estabilizador e as operações lógicas $\overline{X}$ atuando sobre os qubits de informação. No caso de um canal bit flip, os 0s e 1s da matriz (5.67) devem ser substituídos por operadores I e Z e os 0s e 1s da matriz (5.65) por operadores I e X.

O CSL de codificação e o CSL de decodificação de síndromes para o CCC (6, 3, 1) são apresentados na Figura 5.17.

Ao entrarem no circuito da Figura 5.17 os $(N+1)$ blocos recebidos do canal ruidoso, é necessário mais um bloco de zeros para retornar ao estado inicial (todo nulo) do decodificador de síndromes. Assim, para uma sequência de informação com $3N$ qubits, precisaremos considerar a medida de $3((N+1)+1) = 3N+6$ observáveis para poder aplicar o ADS clássico. Quanto à manipulação lógica deste código, pode-se afirmar que uma transformação unitária sobre $3N$ qubits *lógicos* será implementada através de uma transformação unitária sobre os $6(N+1)$ qubits *físicos*.

Como as matrizes (5.63) e (5.67) são exatamente as mesmas, podemos usar as soluções (5.64) para determinar a sequência de erros $\mathbf{e} = (\mathbf{e}^{(1)}, \mathbf{e}^{(2)}, \mathbf{e}^{(3)}, \mathbf{e}^{(4)}, \mathbf{e}^{(5)}, \mathbf{e}^{(6)})$ da equação de síndromes $\mathbf{s} = \mathbf{e}\mathbf{H}^T$, com $\mathbf{s} = (\mathbf{s}^{(1)}, \mathbf{s}^{(2)}, \mathbf{s}^{(3)})$ e $\mathbf{r} = (\mathbf{r}^{(1)}, \mathbf{r}^{(2)}, \mathbf{r}^{(3)}, \mathbf{r}^{(4)}, \mathbf{r}^{(5)}, \mathbf{r}^{(6)})$. Aqui,

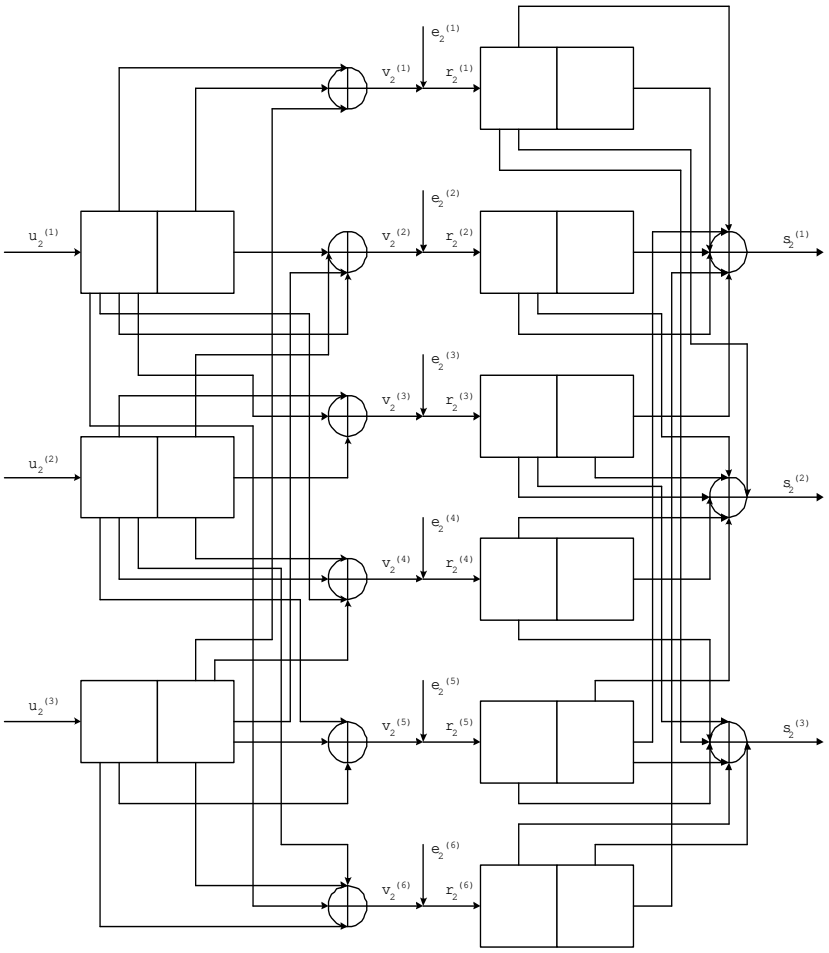

Figura 5.17: CSL de codificação e o CSL de decodificação de síndromes para o CCC (6, 3, 1).

porém, precisaremos de três vezes mais unidades de tempo em relação ao CCC (2, 1, 3) para realizar a decodificação de uma sequência de bits.

O CCQ [6, 1, 4] para o canal com erro quântico geral

Quando fazemos a concatenação do CCC (3, 1, 3) com matriz geradora (5.21) com o CCC (6, 3, 1) acima, obtemos um novo CCC, de taxa 1/6 e quatro memórias efetivas. Veja o CSL do CCC concatenado (6, 1, 4) e o seu respectivo diagrama de estados nas Figuras 5.18 e 5.19.

É fácil de verificar que são necessárias pelo menos cinco transições para sair do es-

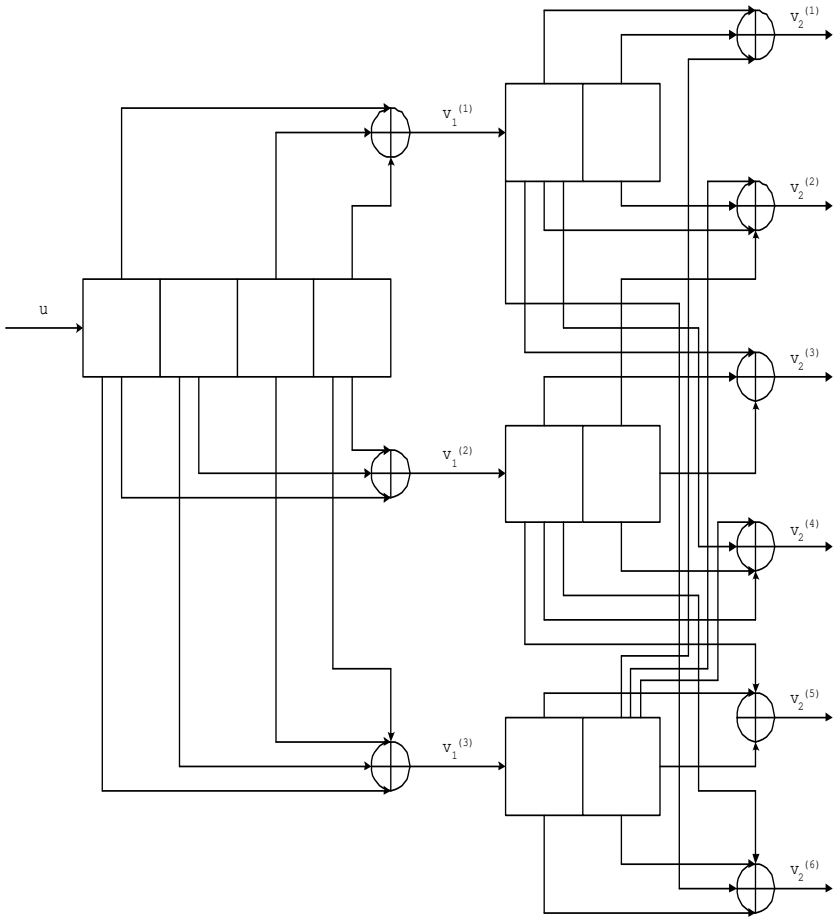

Figura 5.18: Concatenação dos codificadores (3, 1, 3) e (6, 3, 1).

tado inicial 000000 e retornar ao mesmo estado ($000000 \rightarrow 001111 \rightarrow 010011 \rightarrow 100101 \rightarrow 000111 \rightarrow 000000$). Este caminho também é o caminho de d_{free}. Não há nenhum outro caminho não nulo que saia do estado inicial 000000 e retorne ao mesmo estado com peso menor do que dezoito (palavra-código $110001 \rightarrow 010111 \rightarrow 010101 \rightarrow 110110 \rightarrow 011011$). Logo, este CCC é capaz de corrigir qualquer grupo com até oito erros sobre a palavra-código.

O CCC (6, 1, 4) pode ser usado na contrução de um CCQ capaz de garantir a correção de até dois erros quânticos gerais com geradores Z e X sobre a palavra-código quântica. Isto porque os CCCs (3, 1, 3) e (6, 3, 1), associados, respectivamente, à correção de erros de fase e de bit, são capazes de garantir a correção de dois erros e o CCC (6, 1, 4), associado

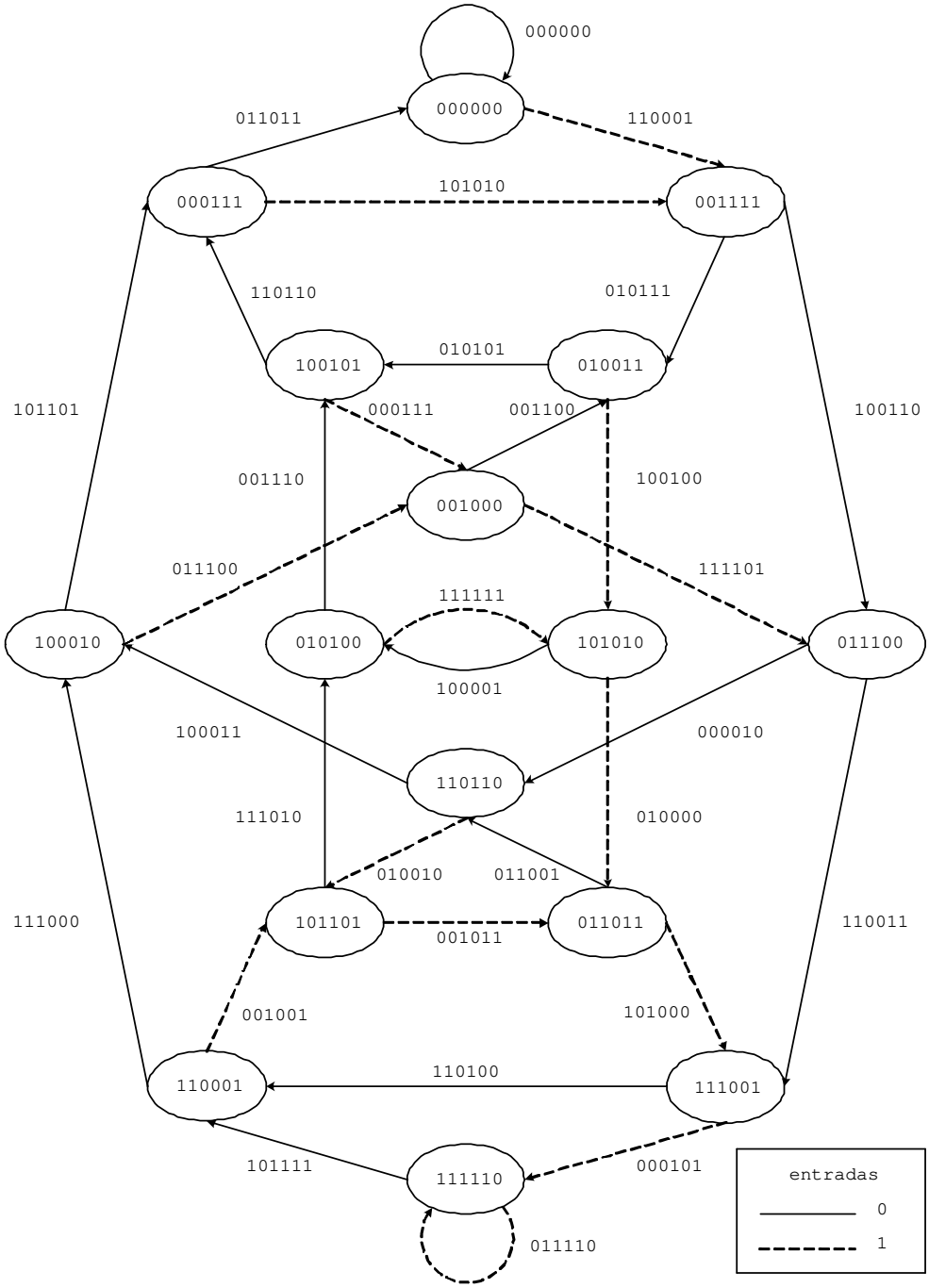

Figura 5.19: Diagrama de estados do CCC concatenado (6, 1, 4).

à correção de erros quânticos gerais, é capaz de garantir a correção de oito erros. Todos estes CCCs tem capacidade de correção além da exigida para a correção de dois erros quânticos gerais, o que nos permite afirmar que alguns padrões com mais de dois erros X

e Z também poderão ser corrigidos por este código.

A operação de codificação para o CCQ [6, 1, 4] pode ser escrita como:

$$\bigotimes_{t=0}^{+\infty}|u_t\rangle \longmapsto \bigotimes_{t=0}^{+\infty}\left\{\sum_{(p_t,\,q_t,\,r_t)=(0,\,0,\,0)}^{(1,\,1,\,1)} \frac{1}{2\sqrt{2}}(-1)^{v_t^{(1)}p_t+v_t^{(2)}q_t+v_t^{(3)}r_t}|p_t+p_{t-1}+r_{t-1},\ p_t+p_{t-1}+ \right.$$

$$\left. q_{t-1}+r_{t-1},\ p_t+q_t+q_{t-1},\ p_t+q_t+q_{t-1}+r_{t-1},\ q_t+r_t+r_{t-1},\ p_t+q_t+r_t+r_{t-1},\ \rangle\right\}, \quad (5.68)$$

onde $v_t^{(1)} = u_t + u_{t-2} + u_{t-3}$, $v_t^{(2)} = u_t + u_{t-1} + u_{t-3}$ e $v_t^{(3)} = u_t + u_{t-1} + u_{t-2} + u_{t-3}$. Aqui, definimos $u_{-1} = u_{-2} = u_{-3} = 0$ e $p_{-1} = q_{-1} = r_{-1} = 0$. Cada um dos 2^N estados da base de uma sequência de informação com N qubits será codificado em uma superposição de $2^{3(N+3)}$ estados, cada um dos quais com $6((N+3)+1) = 6N+24$ qubits.

A matriz geradora polinomial do CCC (6, 1, 4) pode ser obtida através do produto das matrizes geradoras polinomiais dos CCCs (3, 1, 3) e (6, 3, 1). Esta matriz tem a seguinte forma: $\mathbf{G}(D) = [1+D^2+D^3,\ 1+D+D^2+D^3+D^4,\ D+D^2+D^3,\ D+D^3+D^4,\ 1+D+D^2+D^4]$. Ou, na forma de matriz discreta semi-infinita,

$$\mathbf{G}_C = \begin{bmatrix} 110001 & 010111 & 110101 & 110110 & 011011 & & \\ & 110001 & 010111 & 110101 & 110110 & 011011 & \\ & & \ddots\ddots\ddots & \ddots\ddots\ddots & \ddots\ddots\ddots & \ddots\ddots\ddots & \ddots\ddots\ddots \end{bmatrix}. \quad (5.69)$$

Chamemos a matriz (5.67) de $\mathbf{H}_X$, e a matriz (5.25), expandida para o CCQ [6, 1, 4] (basta tomar cada uma das linhas como um bloco de informação a ser codificado pela matriz (5.65) ($\equiv \mathbf{G}_X$)), de $\mathbf{H}_Z$. Com as matrizes $\mathbf{H}_X$ e $\mathbf{H}_Z$ podemos construir a matriz $\mathbf{H}_C = [\mathbf{H}_X;\ \mathbf{H}_Z]$:

$$\mathbf{H}_C = \begin{bmatrix}
110000 & & & & & & \\
111100 & & & & & & \\
101111 & & & & & & \\
111011 & 110000 & & & & & \\
001110 & 111100 & & & & & \\
000011 & 101111 & & & & & \\
 & \ddots\ddots\ddots & \ddots\ddots\ddots & & & & \\
 & & \ddots\ddots\ddots & \ddots\ddots\ddots & & & \\
 & & & \ddots\ddots\ddots & \ddots\ddots\ddots & & \\
 & & & & \ddots\ddots\ddots & \ddots\ddots\ddots & \\
110010 & 101100 & & & & & \\
111110 & 000111 & & & & & \\
111101 & 000010 & 101100 & & & & \\
111101 & 001110 & 000111 & & & & \\
001111 & 100001 & 000010 & 101100 & & & \\
110010 & 010001 & 001110 & 000111 & & & \\
110010 & 100011 & 100001 & 000010 & 101100 & & \\
000000 & 110010 & 010001 & 001110 & 000111 & & \\
 & \ddots\ddots\ddots & \ddots\ddots\ddots & \ddots\ddots\ddots & \ddots\ddots\ddots & \ddots\ddots\ddots & \\
 & \ddots\ddots\ddots & \ddots\ddots\ddots & \ddots\ddots\ddots & \ddots\ddots\ddots & \ddots\ddots\ddots &
\end{bmatrix}. \tag{5.70}$$

As linhas da matriz (5.70) permitem-nos escrever os geradores do grupo estabilizador do CCQ [6, 1, 4] descrito pela operação de codificação (5.68). Usaremos as linhas da matriz $\mathbf{H}_X$ para obter os geradores Z e as linhas da matriz $\mathbf{H}_Z$ para obter os geradores X. Os 0s e 1s da matriz $\mathbf{H}_X$ devem ser substituídos por operadores I e Z e os 0s e 1s da matriz $\mathbf{H}_Z$ por operadores I e X. Usaremos também as linhas da matriz $\mathbf{G}_C$ para determinar as operações lógicas tanto de $\overline{X}$ quanto de $\overline{Z}$ atuando sobre os qubits de informação. Para determinar $\overline{X}$, os 0s e 1s da matriz (5.69) devem ser substituídos por operadores I e Z, e para determinar $\overline{Z}$, os 0s e 1s da matriz (5.69) devem ser substituídos por operadores I e X. Pode-se afirmar que uma transformação unitária sobre N qubits *lógicos* será implementada através de uma transformação unitária sobre $6N+24$ qubits *físicos*.

Ao entrarem no circuito da Figura 5.17 os $(N+4)$ blocos recebidos do canal ruidoso, é necessário mais um bloco de zeros para retornar ao estado inicial (todo nulo) do decodificador de síndromes. Assim, para uma sequência de informação com N qubits, precisaremos considerar a medida de $3((N+4)+1) = 3N+15$ observáveis Z e de $2N+12$ observáveis X para poder aplicar o ADS clássico. Usaremos os autovalores $\{\alpha_{M_X}\}$ (onde M_X são os observáveis Z) nas soluções (5.64) e os autovalores $\{\alpha_{M_Z}\}$ (onde M_Z são os observáveis X) nas soluções (5.32) da equação de síndromes para detectar através de *dois* diagramas de treliça os qubits onde eventualmente ocorreram erros de bit e os *blocos* do código phase flip associado onde eventualmente ocorreram erros de fase.

5.4.2 Um CCQ [3, 1, 5]

Um CCQ [2, 1, 4] para o canal phase flip

Considere o CCC (2, 1, 4) com matriz geradora polinomial $\mathbf{G}(D) = [1+D+D^4,\ 1+D^2+D^3+D^4]$ [12]. Em termos de notação de matriz discreta semi-infinita, esta matriz escreve-se como:

$$\mathbf{G} = \begin{bmatrix} 11 & 10 & 01 & 01 & 11 & & & \\ & 11 & 10 & 01 & 01 & 11 & & \\ & & 11 & 10 & 01 & 01 & 11 & \\ & & & \ddots & \ddots & \ddots & \ddots & \ddots \end{bmatrix}. \tag{5.71}$$

Este CCC tem $d_{free} = 7$ e, portanto, pode corrigir até três erros clássicos. Por ser um CCC de taxa 1/2, temos que $\mathbf{H}(D) = [1+D^2+D^3+D^4,\ 1+D+D^4]$. Em termos de notação de matriz discreta semi-infinita, esta matriz escreve-se como:

[12] Referência [14], páginas 81-94.

$$\mathbf{H} = \begin{bmatrix} 11 & & & & & & \\ 01 & 11 & & & & & \\ 10 & 01 & 11 & & & & \\ 10 & 10 & 01 & 11 & & & \\ 11 & 10 & 10 & 01 & 11 & & \\ & 11 & 10 & 10 & 01 & 11 & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots \end{bmatrix}. \tag{5.72}$$

De acordo com (5.1), as soluções da equação de síndromes para este codificador são:

$$\begin{aligned} \mathbf{e}^{(1)}(D) &= \mathbf{s}(D) + D\mathbf{s}(D) + \mathbf{t}(D) + D\mathbf{t}(D) + D^4\mathbf{t}(D), \\ \mathbf{e}^{(2)}(D) &= D\mathbf{s}(D) + \mathbf{t}(D) + D^2\mathbf{t}(D) + D^3\mathbf{t}(D) + D^4\mathbf{t}(D), \end{aligned} \tag{5.73}$$

onde aqui usamos $\alpha_1 = 1 + D$ e $\alpha_2 = D$.

Estamos agora em condições de gerar o CCQ [2, 1, 4] para os canais bit flip e phase flip. Para o canal phase flip (que será o mais interessante neste contexto), a operação de codificação na base $\{|0\rangle, |1\rangle\}$, será escrita como:

$$\bigotimes_{t=0}^{+\infty} |u_t\rangle \longmapsto \bigotimes_{t=0}^{+\infty} \left\{ \frac{1}{2}(|0\rangle + (-1)^{v_t^{(1)}}|1\rangle)\,(|0\rangle + (-1)^{v_t^{(2)}}|1\rangle) \right\}, \tag{5.74}$$

onde $v_t^{(1)} = u_t + u_{t-1} + u_{t-4}$ e $v_t^{(2)} = u_t + u_{t-2} + u_{t-3} + u_{t-4}$. Aqui, definimos $u_{-1} = u_{-2} = u_{-3} = u_{-4} = 0$. Cada um dos 2^N estados da base de uma sequência de informação com N qubits será codificado em uma superposição de $2^{2(N+4)}$ estados, cada um dos quais com comprimento $2(N+4)$. Naturalmente, trata-se de um CCQ [2, 1, 4] capaz de garantir a correção de até três erros Z sobre a palavra-código.

As linhas da matriz de paridade (5.72) e da matriz geradora (5.71) permitem-nos escrever, respectivamente, os geradores do grupo estabilizador e as operações lógicas $\overline{Z}$ atuando sobre os qubits de informação. No caso de um canal phase flip, os 0s e 1s da matriz (5.72) devem ser substituídos por operadores I e X e os 0s e 1s da matriz (5.71) por operadores I e Z.

O CSL de codificação e o CSL de decodificação de síndromes são apresentados na Figura 5.20.

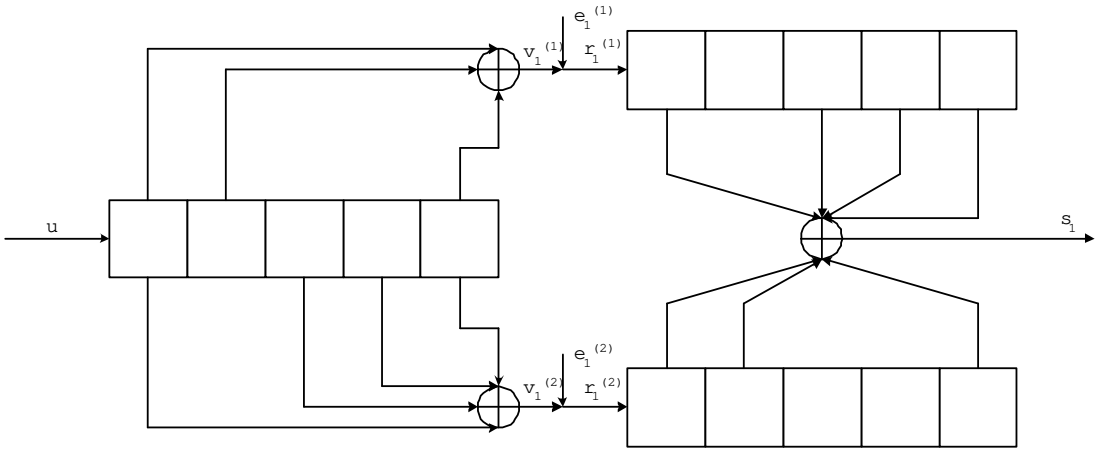

Figura 5.20: CSL de codificação e CSL de decodificação de síndromes para o CCC (2, 1, 4).

Ao entrarem no circuito da Figura 5.20 os $(N+4)$ blocos recebidos do canal ruidoso, são necessários mais quatro blocos de zeros para retornar ao estado inicial (todo nulo) do decodificador de síndromes. Assim, para uma sequência de informação com N qubits, precisaremos considerar a medida de $(N+4)+4 = N+8$ observáveis para poder aplicar o ADS clássico. Quanto à manipulação lógica deste código, pode-se afirmar que uma transformação unitária sobre N qubits *lógicos* será implementada através de uma transformação unitária sobre os $2(N+4)$ qubits *físicos*.

Um CCQ [3, 2, 1] para o canal bit flip

Considere que obtemos um CCC (3, 2, 1) através do puncionamento do terceiro bit do CCC (2, 1, 2) com matriz geradora $\mathbf{G}(D) = [1+D^2,\ 1+D+D^2]$ [13]. Assim, a matriz geradora polinomial do CCC (3, 2, 1) será:

$$\mathbf{G}(D) = \begin{bmatrix} 1+D & 1+D & 1 \\ 0 & D & 1+D \end{bmatrix}, \tag{5.75}$$

[13] Vimos este exemplo na seção 3.5.2.

ou, na forma de matriz discreta semi-infinita,

$$\mathbf{G} = \begin{bmatrix} 111 & 110 & & \\ 001 & 011 & & \\ & 111 & 110 & \\ & 001 & 011 & \\ & & \ddots & \ddots \\ & & \ddots & \ddots \end{bmatrix}. \tag{5.76}$$

Este CCC tem $d_{free} = 3$ e, portanto, pode corrigir um erro clássico. Estamos agora em condições de gerar o CCQ [3, 2, 1] para um canal bit flip. Para este canal, a operação de codificação será escrita como:

$$\bigotimes_{t=0}^{+\infty} |u_t^{(1)}, u_t^{(2)}\rangle \longmapsto \bigotimes_{t=0}^{+\infty} |v_t^{(1)}, v_t^{(2)}, v_t^{(3)}\rangle, \tag{5.77}$$

onde $v_t^{(1)} = u_t^{(1)} + u_{t-1}^{(1)}$, $v_t^{(2)} = u_t^{(1)} + u_{t-1}^{(1)} + u_{t-1}^{(2)}$ e $v_t^{(3)} = u_t^{(1)} + u_t^{(2)} + u_{t-1}^{(2)}$. Aqui, definimos $u_{-1}^{(1)} = u_{-1}^{(2)} = 0$. Cada um dos 2^{2N} estados da base de uma sequência de informação com N blocos de qubits será codificado em um estado com $3(N+1)$ qubits. Trata-se de um CCQ [3, 2, 1] capaz de garantir a correção de um erro X sobre qualquer qubit da palavra-código.

Uma matriz verificação de paridade para a matriz geradora (5.75) é a matriz:

$$\mathbf{H}(D) = \begin{bmatrix} 1+D+D^2 & 1+D^2 & D+D^2 \end{bmatrix}, \tag{5.78}$$

ou, na forma de matriz discreta semi-infinita,

$$\mathbf{H} = \begin{bmatrix} 110 & & & & & & & \\ 101 & 110 & & & & & & \\ 111 & 101 & 110 & & & & & \\ & 111 & 101 & 110 & & & & \\ & & 111 & 101 & 110 & & & \\ & & & 111 & 101 & 110 & & \\ & & & & 111 & 101 & 110 & \\ & & & & & \ddots & \ddots & \ddots \end{bmatrix}. \tag{5.79}$$

Para este CCC (3, 2, 1) não foi possível encontrar um matriz verificação de paridade com o mesmo número de memórias (uma) da matriz geradora. A matriz (5.78) é a matriz *mínima* (ou seja, a com menor número de memórias) para a matriz (5.76).

Repare que a matriz (5.78) é muito parecida com a matriz verificação de paridade do CCC (2, 1, 2) que deu origem a este puncionamento, a matriz $\mathbf{H}(D) = [1+D+D^2,\ 1+D^2]$. Este fato é uma *pista* para, em uma nova etapa de nossa pesquisa, tentarmos descobrir um meio de obter a matriz verificação de paridade do CCC puncionado a partir da matriz verificação de paridade do CCC original, algo que, até onde temos conhecimento, nunca foi conseguido. Obviamente, a transformação de uma matriz em outra seria uma função da posição do(s) bit(s) puncionado(s). É neste ponto que reside a complexidade do problema. O estudo da classe dos CCQs é um grande incentivo para que se busque uma resposta para este problema. Isto evitaria que tivéssemos que calcular a matriz verificação de paridade e as soluções da equação de síndromes para o CCC puncionado, que, em geral, são muito mais difíceis de serem obtidas do que para o CCC original.

A forma normal de Smith para a matriz transposta de (5.78) é escrita como:

$$\mathbf{H}^T(D) = \mathbf{L} \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T, \tag{5.80}$$

onde a inversa de $\mathbf{L}(D)$ é

$$\mathbf{L}^{-1}(D) = \begin{bmatrix} 1 & 0 & 1 \\ 1+D & 1 & D \\ D+D^2 & 0 & 1+D+D^2 \end{bmatrix}. \tag{5.81}$$

Substituindo a equação (5.80) na equação $\mathbf{s}(D) = \mathbf{e}(D)\mathbf{H}^T(D)$, teremos a seguinte equação de síndromes a resolver:

$$\mathbf{s}(D) = [\boldsymbol{\varepsilon}^{(1)}(D),\, \boldsymbol{\varepsilon}^{(2)}(D),\, \boldsymbol{\varepsilon}^{(3)}(D)] \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T, \tag{5.82}$$

onde $\boldsymbol{\varepsilon}^{(i)}(D) = \mathbf{e}^{(i)}(D)\mathbf{L}(D)$ para $1 \leq i \leq n = 3$.

Usando as equações $\boldsymbol{\varepsilon}^{(j)}(D) = \boldsymbol{\sigma}^{(j)}(D)$, para $1 \leq j \leq n-k = 1$ e $\boldsymbol{\varepsilon}^{(j)}(D) = \mathbf{t}^{(j-n+k)}(D)$, para $n-k+1 = 3-2+1 = 2 \leq j \leq n = 3$ (onde $\mathbf{t}^{(i)}(D)$ para $1 \leq i \leq k = 2$ é um polinômio arbitrário no anel euclidiano $F[D]$), a solução da equação (5.82) para $\boldsymbol{\varepsilon}^{(j)}(D)$ será:

$$\boldsymbol{\varepsilon}^{(1)}(D) = \mathbf{s}(D),\ \boldsymbol{\varepsilon}^{(2)}(D) = \mathbf{t}^{(1)}(D),\ \boldsymbol{\varepsilon}^{(3)}(D) = \mathbf{t}^{(2)}(D). \tag{5.83}$$

Finalmente os vetores de erro $\mathbf{e}^{(i)}(D)$ em termos das soluções dadas em(5.83) são, usando (5.81), $[\mathbf{e}^{(1)}(D),\ \mathbf{e}^{(2)}(D),\ \mathbf{e}^{(3)}(D)] = [\mathbf{s}(D),\ \mathbf{t}^{(1)}(D),\ \mathbf{t}^{(2)}(D)]\mathbf{L}^{-1}(D)$. Isto produz:

$$\begin{aligned}
\mathbf{e}^{(1)}(D) &= \mathbf{s}(D) + \mathbf{t}^{(1)}(D) + D\mathbf{t}^{(1)}(D) + D\mathbf{t}^{(2)}(D) + D^2\mathbf{t}^{(2)}(D), \\
\mathbf{e}^{(2)}(D) &= \mathbf{t}^{(1)}(D), \\
\mathbf{e}^{(3)}(D) &= \mathbf{s}(D) + D\mathbf{t}^{(1)}(D) + \mathbf{t}^{(2)}(D) + D\mathbf{t}^{(2)}(D) + D^2\mathbf{t}^{(2)}(D),
\end{aligned} \tag{5.84}$$

como as soluções gerais da equação de síndromes $\mathbf{s}(D) = \mathbf{e}(D)\mathbf{H}^T(D)$ para o codificador (3, 2, 1) com matriz geradora (5.75).

Os valores de $\mathbf{t}^{(1)}(D)$, $D\mathbf{t}^{(1)}(D)$,$D^2\mathbf{t}^{(1)}(D)$, $\mathbf{t}^{(2)}(D)$, $D\mathbf{t}^{(2)}(D)$ e $D^2\mathbf{t}^{(2)}(D)$ ao longo da treliça podem ser obtidos através da Tabela 5.5. Definimos o estado inicial como ($D\mathbf{t}^{(1)}(D)$, $D^2\mathbf{t}^{(1)}(D)$, $D\mathbf{t}^{(2)}(D)$, $D^2\mathbf{t}^{(2)}(D)$) = (0,0,0,0) e $\mathbf{s}(D) = 0$ antes do estágio 0. O ADS então seleciona o caminho no diagrama de treliça com o menor peso de Hamming.

As linhas da matriz de paridade (5.79) e da matriz geradora (5.76) permitem-nos escrever, respectivamente, os geradores do grupo estabilizador e as operações lógicas $\overline{X}$ atuando sobre os qubits de informação. No caso de um canal bit flip, os 0s e 1s da matriz (5.79) devem ser substituídos por operadores I e Z e os 0s e 1s da matriz (5.76) por operadores I e X.

O CSL de codificação e o CSL de decodificação de síndromes para o CCC (3, 2, 1) são apresentados na Figura 5.21.

Ao entrarem no circuito da Figura 5.21 os $(N+1)$ blocos recebidos do canal ruidoso, são necessários mais dois blocos de zeros para retornar ao estado inicial (todo nulo) do decodificador de síndromes. Assim, para uma sequência de informação com $2N$ qubits, precisaremos considerar a medida de $(N+1)+2 = N+3$ observáveis para poder aplicar o ADS clássico. Quanto à manipulação lógica deste código, pode-se afirmar que uma transformação unitária sobre $2N$ qubits *lógicos* será implementada através de uma transformação

$D\mathbf{t}^{(1)}(D)$, $D^2\mathbf{t}^{(1)}(D)$, $D\mathbf{t}^{(2)}(D)$, $D^2\mathbf{t}^{(2)}(D)$	$\mathbf{t}^{(1)}(D)$=0	$\mathbf{t}^{(1)}(D)$=1	$\mathbf{t}^{(2)}(D)$=0	$\mathbf{t}^{(2)}(D)$=1
$a = 0000$	0000	1000	0000	0010
$b = 0001$	0001	1001	0000	0010
$c = 0010$	0010	1010	0001	0011
$d = 0011$	0011	1011	0001	0011
$e = 0100$	0000	1000	0100	0110
$f = 0101$	0001	1001	0100	0110
$g = 0110$	0010	1010	0101	0111
$h = 0111$	0011	1011	0101	0111
$i = 1000$	0100	1100	1000	1010
$j = 1001$	0101	1101	1000	1010
$k = 1010$	0110	1110	1001	1011
$l = 1011$	0111	1111	1001	1011
$m = 1100$	0100	1100	1100	1110
$n = 1101$	0101	1101	1100	1110
$o = 1110$	0110	1110	1101	1111
$p = 1111$	0111	1111	1101	1111

Tabela 5.5: Tabela de estados do registro de deslocamento para $\mathbf{t}^{(1)}(D)$ e $\mathbf{t}^{(2)}(D)$.

unitária sobre os $3(N+1)$ qubits *físicos*.

Um CCQ [3, 1, 5] para o canal com erro quântico geral

Quando fazemos a concatenação dos CCCs (2, 1, 4) e (3, 2, 1), obtemos um novo CCC, de taxa 1/3 e cinco memórias efetivas. Veja o CSL do CCC concatenado (3, 1, 5) e o seu respectivo diagrama de estados nas Figuras 5.22 e 5.23.

É fácil de verificar que são necessárias pelo menos seis transições para sair do estado inicial 000000 e retornar ao mesmo estado ($000000 \rightarrow 000111 \rightarrow 001010 \rightarrow 010001 \rightarrow 100001 \rightarrow 000011 \rightarrow 000000$). Este caminho também é o caminho de d_{free}. Não há nenhum outro caminho não nulo que saia do estado inicial 000000 e retorne ao mesmo estado com peso menor do que onze (palavra-código $110 \rightarrow 010 \rightarrow 111 \rightarrow 010 \rightarrow 101 \rightarrow 101$). Logo,

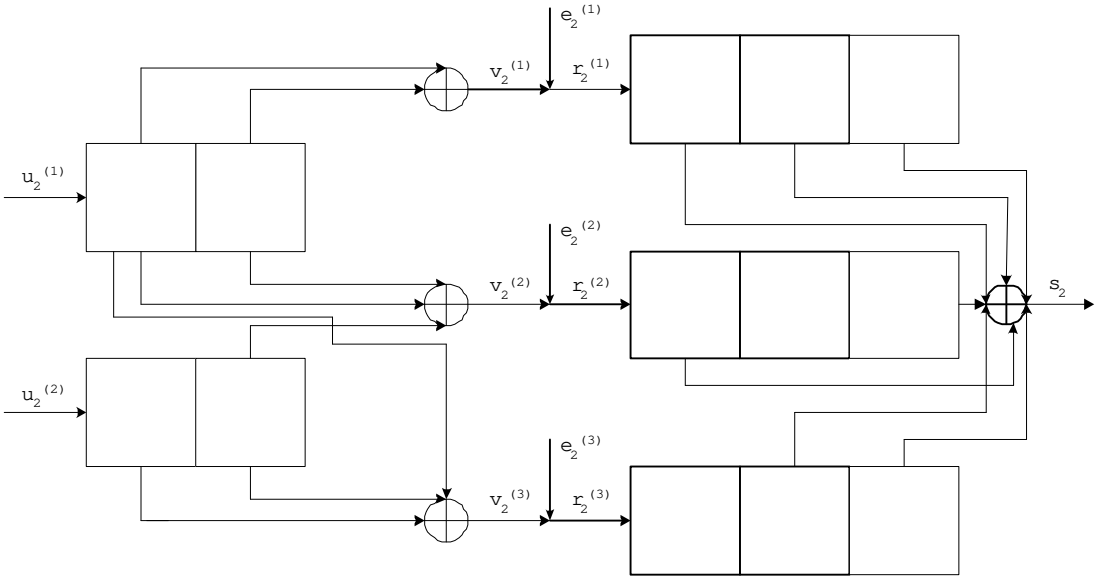

Figura 5.21: CSL de codificação e CSL de decodificação de síndromes para o CCC (3, 2, 1).

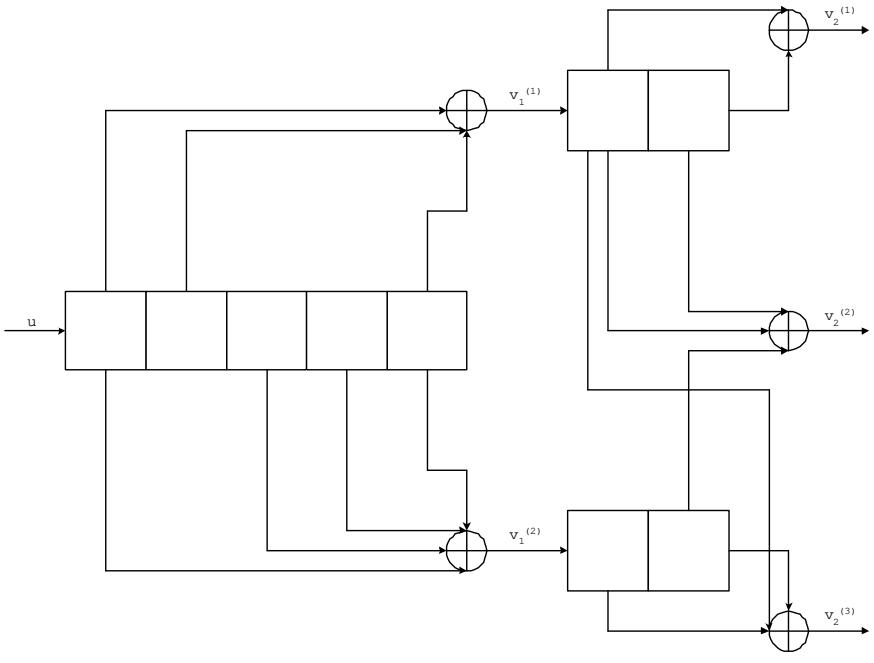

Figura 5.22: Concatenação dos codificadores (2, 1, 4) e (3, 2, 1).

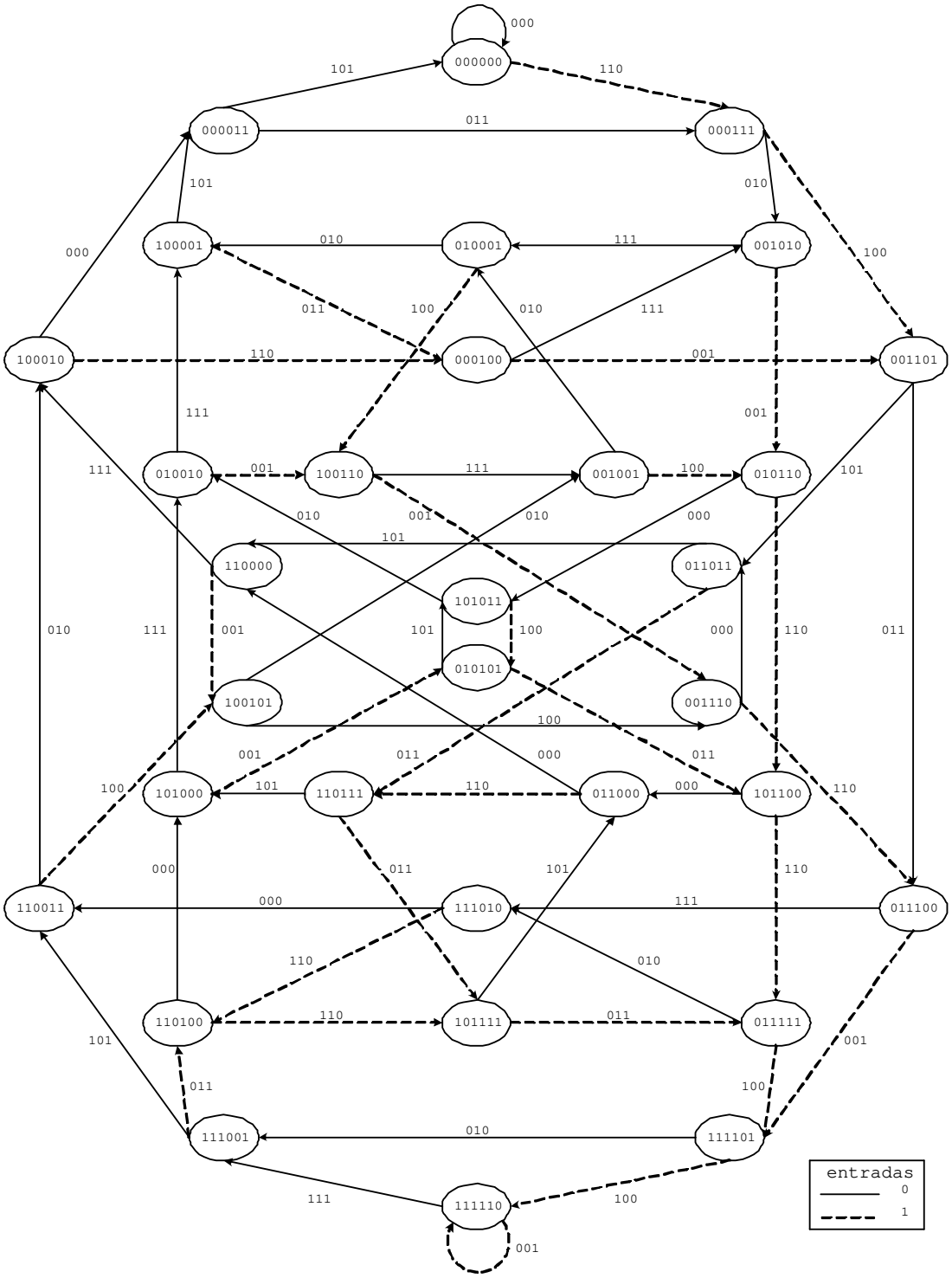

Figura 5.23: Diagrama de estados do CCC concatenado (3, 1, 5).

este CCC é capaz de corrigir qualquer grupo com até cinco erros sobre a palavra-código.

O CCC (3, 1, 5) pode ser usado na contrução de um CCQ capaz de garantir a correção de até um erro quântico geral com geradores Z e X sobre a palavra-código quântica. Isto porque os CCCs (2, 1, 4) e (3, 2, 1), associados respectivamente à correção de erros de fase e de bit, são capazes de garantir a correção de um erro e o CCC (3, 1, 5), associado à correção de erros quânticos gerais, é capaz de garantir a correção de quatro erros. Os CCCs (2, 1, 4) e (3, 1, 5) têm capacidade de correção além da exigida para a correção de um erro quântico geral, o que nos permite afirmar que alguns padrões com mais de um erro Z também poderão ser corrigidos por este código.

Repare que aqui conseguimos construir um CCQ de taxa 1/3 capaz de corrigir um erro quântico geral. Porém, o preço que pagamos foi o aumento do número de memórias convolucionais quânticas, que passou de três para cinco.

A operação de codificação para o CCQ [3, 1, 5] pode ser escrita como:

$$\bigotimes_{t=0}^{+\infty}|u_t\rangle \longmapsto \bigotimes_{t=0}^{+\infty}\left\{\sum_{(p_t,\,q_t)=(0,\,0)}^{(1,\,1)} \frac{1}{2}(-1)^{v_t^{(1)}p_t+v_t^{(2)}q_t}|p_t+p_{t-1},\ p_t+p_{t-1}+q_{t-1},\ p_t+q_t+q_{t-1}\rangle\right\}. \quad (5.85)$$

onde $v_t^{(1)} = u_t + u_{t-1} + u_{t-4}$ e $v_t^{(2)} = u_t + u_{t-2} + u_{t-3} + u_{t-4}$. Aqui, definimos $u_{-1} = u_{-2} = u_{-3} = u_{-4} = 0$ e $p_{-1} = q_{-1} = 0$. Cada um dos 2^N estados da base de uma sequência de informação com N qubits será codificado em uma superposição de $2^{2(N+4)}$ estados, cada um dos quais com $3((N+4)+1) = 3N+15$ qubits.

A matriz geradora polinomial do CCC (3, 1, 5) pode ser obtida através do produto das matrizes geradoras polinomiais dos CCCs (2, 1, 4) e (3, 2, 1). Esta matriz tem a seguinte forma: $\mathbf{G}(D) = [1+D^2+D^4+D^5,\ 1+D+D^2+D^3,\ D^2+D^4+D^5]$. Ou, na forma de matriz discreta semi-infinita,

$$\mathbf{G}_C = \begin{bmatrix} 110 & 010 & 111 & 010 & 101 & 101 & & \\ & 110 & 010 & 111 & 010 & 101 & 101 & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \end{bmatrix}. \quad (5.86)$$

Chamemos a matriz (5.79) de $\mathbf{H}_X$, e a matriz (5.72), expandida para o CCQ [3, 1, 5] (basta tomar cada uma das linhas como um bloco de informação a ser codificado pela

matriz (5.76) ($\equiv \mathbf{G}_X$)), de $\mathbf{H}_Z$. Com as matrizes $\mathbf{H}_X$ e $\mathbf{H}_Z$ podemos construir a matriz $\mathbf{H}_C = [\mathbf{H}_X; \mathbf{H}_Z]$:

$$\mathbf{H}_C = \left[\begin{array}{cccccccc}
110 & & & & & & & \\
101 & 110 & & & & & & \\
111 & 101 & 110 & & & & & \\
 & 111 & 101 & 110 & & & & \\
 & & 111 & 101 & 110 & & & \\
 & & & 111 & 101 & 110 & & \\
 & & & & 111 & 101 & 110 & \\
 & & & & & \ddots & \ddots & \ddots \\
110 & 101 & & & & & & \\
001 & 101 & 101 & & & & & \\
111 & 111 & 101 & 101 & & & & \\
111 & 001 & 111 & 101 & 101 & & & \\
110 & 010 & 001 & 111 & 101 & 101 & & \\
 & 110 & 010 & 001 & 111 & 101 & 101 & \\
 & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots
\end{array}\right]. \tag{5.87}$$

As linhas da matriz (5.87) permitem-nos escrever os geradores do grupo estabilizador do CCQ [3, 1, 5] descrito pela operação de codificação (5.85). Usaremos as linhas da matriz $\mathbf{H}_X$ para obter os geradores Z e as linhas da matriz $\mathbf{H}_Z$ para obter os geradores X. Os 0s e 1s da matriz $\mathbf{H}_X$ devem ser substituídos por operadores I e Z e os 0s e 1s da matriz $\mathbf{H}_Z$ por operadores I e X. Usaremos também as linhas da matriz $\mathbf{G}_C$ para determinar as operações lógicas tanto de $\overline{X}$ quanto de $\overline{Z}$ atuando sobre os qubits de informação. Para determinar $\overline{X}$, os 0s e 1s da matriz (5.86) devem ser substituídos por operadores I e Z, e para determinar $\overline{Z}$, os 0s e 1s da matriz (5.86) devem ser substituídos por operadores I e X. Pode-se afirmar que uma transformação unitária sobre N qubits *lógicos* será implementada através de uma transformação unitária sobre $3N+15$ qubits *físicos*.

Ao entrarem no circuito da Figura 5.21 os $(N+5)$ blocos recebidos do canal ruidoso, são necessários mais dois blocos de zeros para retornar ao estado inicial (todo nulo) do decodificador de síndromes. Assim, para uma sequência de informação com N qubits, pre-

cisaremos considerar a medida de $(N+5)+2 = N+7$ observáveis Z e de $N+8$ observáveis X para poder aplicar o ADS clássico. Usaremos os autovalores $\{\alpha_{M_X}\}$ (onde M_X são os observáveis Z) nas soluções (5.84) e os autovalores de $\{\alpha_{M_Z}\}$ (onde M_Z são os observáveis X) nas soluções (5.73) da equação de síndromes para detectar através de *dois* diagramas de treliça os qubits onde eventualmente ocorreram erros de bit e os *blocos* do código phase flip associado onde eventualmente ocorreram erros de fase.

Até agora construimos CCQs concatenados a partir de CCCs (2, 1, m) e (3, 1, m), para os quais encontrar a matriz verificação de paridade e as soluções da equação de síndromes é uma tarefa relativamente simples. O mesmo não se pode dizer para outras taxas. Apesar do ganho na capacidade de correção, a complexidade do problema exige grande trabalho computacional. Conscientes destas dificuldades, faremos apenas uma breve exposição para os dois próximos códigos.

5.5 Uso de Codificadores Primitivos (4, 1, m)

A seguir, um exemplo de CCQ [16, 1, 5] concatenado capaz de corrigir até seis erros quânticos gerais. Este CCQ é gerado a partir de um único CCC (4, 1, 4).

Considere que usemos o CCC (4, 1, 4) com matriz geradora

$$\mathbf{G}(D) = \left[\begin{array}{cccc} 1+D^2+D^4 & 1+D^2+D^3+D^4 & 1+D+D^3+D^4 & 1+D+D^2+D^3+D^4 \end{array} \right] \qquad (5.88)$$

na construção de um CCQ [4, 1, 4] para o canal phase flip. Este CCC (4, 1, 4) tem $d_{fee} = 16$ e, portanto, o correspondente CCQ [4, 1, 4] é capaz de corrigir até sete erros Z sobre a palavra-código. Suponha também que usemos o CCC (16, 4, 1) gerado a partir da mesma matriz geradora do CCC (4, 1, 4), ou seja, o CCC (16, 4, 1) com matriz geradora transposta[14]

[14]Colocamos a matriz na forma transposta para que pudesse ser reproduzida aqui.

$$\mathbf{G}^T(D) = \left[\begin{array}{cccc} 1+D & 0 & D & 0 \\ 1+D & D & D & 0 \\ 1+D & D & 0 & D \\ 1+D & D & D & D \\ 0 & 1+D & 0 & D \\ 0 & 1+D & D & D \\ 1 & 1+D & D & 0 \\ 1 & 1+D & D & D \\ 1 & 0 & 1+D & 0 \\ 1 & 0 & 1+D & D \\ 0 & 1 & 1+D & D \\ 1 & 1 & 1+D & D \\ 0 & 1 & 0 & 1+D \\ 1 & 1 & 0 & 1+D \\ 1 & 0 & 1 & 1+D \\ 1 & 1 & 1 & 1+D \end{array}\right] \tag{5.89}$$

na construção de um CCQ [16, 4, 1] para o canal bit flip. Este CCC (16, 4, 1) tem também $d_{free} = 16$ e, portanto, o correspondente CCQ [16, 4, 1] é capaz de corrigir até sete erros X sobre a palavra-código[15].

Quando fazemos a concatenação deste dois CCCs (4, 1, 4) e (16, 4, 1), obtemos um CCC concatenado (16, 1, 5). Veja a matriz geradora deste CCC concatenado em (5.90).

Veja agora a Figura 5.24. Este CCC tem $d_{fee} = 54$ e, portanto, é capaz de gerar um CCQ [16, 1, 5] capaz de corrigir até seis erros quânticos gerais sobre a palavra-código.

[15]Este CCC (4, 1, 4) é ótimo (Referência [14], páginas 81-94).

$$\mathbf{G}_C^T(D) = \begin{bmatrix} 1+D^3 \\ 1+D+D^4+D^5 \\ 1+D+D^3+D^4+D^5 \\ 1+D^2+D^3 \\ 1+D^3+D^4 \\ 1+D+D^2+D^3+D^5 \\ D^2 \\ D+D^3+D^4+D^5 \\ D^3+D^4+D^5 \\ D+D^2 \\ D+D^2+D^3 \\ 1+D+D^3+D^4 \\ D^2+D^3+D^4+D^5 \\ 1+D^3+D^5 \\ 1+D+D^2+D^3+D^5 \\ D+D^4+D^5 \end{bmatrix}. \tag{5.90}$$

Encontrar a matriz verificação de paridade e as soluções da equação de síndromes para um CCC (4, 1, m) não é uma tarefa tão simples como é para os CCCs (2, 1, m) e (3, 1, m). Além disso, as dimensões dos códigos envolvidos na concatenação acima são muito grandes para explicitarmos todos os cálculos. Deixaremos esta tarefa para os interessados na aplicação prática deste código quântico.

5.6 Um Super CCQ [4, 2, 4] Concatenado

Considere que usemos o CCC (3, 2, 2) com matriz geradora

$$\mathbf{G}(D) = \begin{bmatrix} 1+D+D^2 & D^2 & D^2 \\ D & 1+D^2 & 1+D+D^2 \end{bmatrix} \tag{5.91}$$

na construção de um CCQ [3, 2, 2] para o canal phase flip. Este CCC (3, 2, 2) tem $d_{fee} = 5$ e, portanto, o correspondente CCQ [3, 2, 2] é capaz de corrigir até dois erros Z sobre a

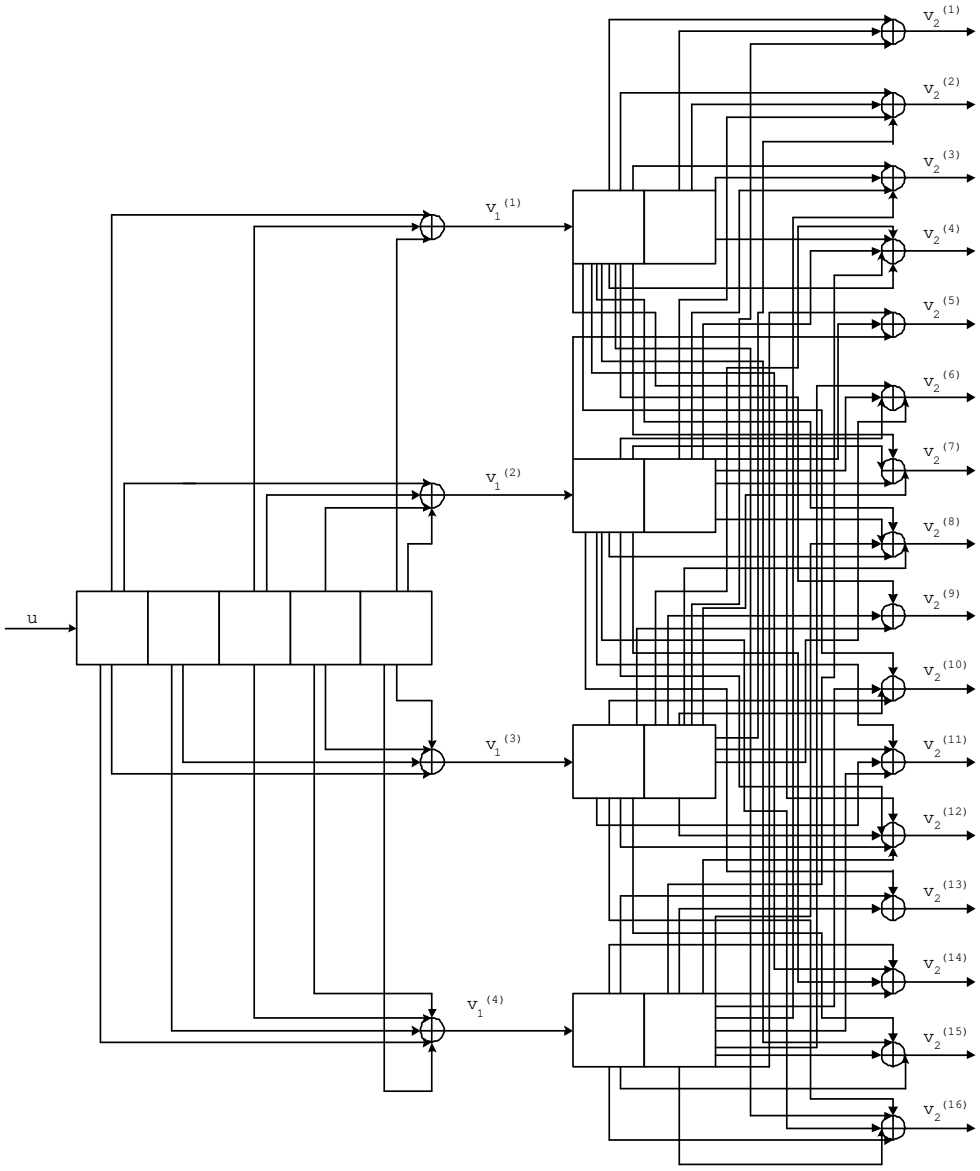

Figura 5.24: Concatenação dos codificadores (4, 1, 4) e (16, 4, 1).

palavra-código. Suponha também que usemos o CCC (4, 3, 2) com matriz geradora

$$\mathbf{G}(D) = \begin{bmatrix} 1+D & D^2 & 0 & 1+D+D^2 \\ D+D^2 & 1 & D^2 & 1+D \\ D & D+D^2 & 1+D+D^2 & 1 \end{bmatrix} \tag{5.92}$$

na construção de um CCQ [4, 3, 2] para o canal bit flip. Este CCC (4, 3, 2) tem $d_{free} = 6$ e, portanto, o correspondente CCQ [4, 3, 2] é capaz de corrigir até dois erros X sobre a palavra-código[16].

Quando fazemos a concatenação deste dois CCCs (3, 2, 2) e (4, 3, 2), obtemos um CCC concatenado (4, 2, 4) com matriz geradora

$$\mathbf{G}_C(D) = \begin{bmatrix} 1+D^3+D^4 & 0 & D^2+D^3 & 1+D^2+D^3+D^4 \\ D+D^2+D^4 & 1+D+D^2+D^3+D^4 & 1 & D+D^2 \end{bmatrix}. \tag{5.93}$$

Veja a Figura 5.25. Este CCC tem $d_{fee} = 9$ e, portanto, capaz de gerar um CCQ [4, 2, 4] capaz de corrigir até um erro quântico geral sobre qualquer qubit da palavra-código. Assim, construimos um CCQ com taxa efetiva $1/2$ capaz de corrigir um erro quântico geral, um código quântico de performance excepcional.

Encontrar a matriz verificação de paridade e as soluções das equações de síndrome para os CCCs acima não é uma tarefa fácil. A começar pelo fato de não ser possível obter para estes CCCs matrizes de verificação de paridade com o mesmo número de memórias da correspondente matriz geradora. É um esforço que certamente valerá a pena caso a aplicação prática deste código se mostre relevante.

[16]Estes dois CCCs (3, 2, 2) e (4, 3, 2) são ótimos (Referência [14], páginas 81-94).

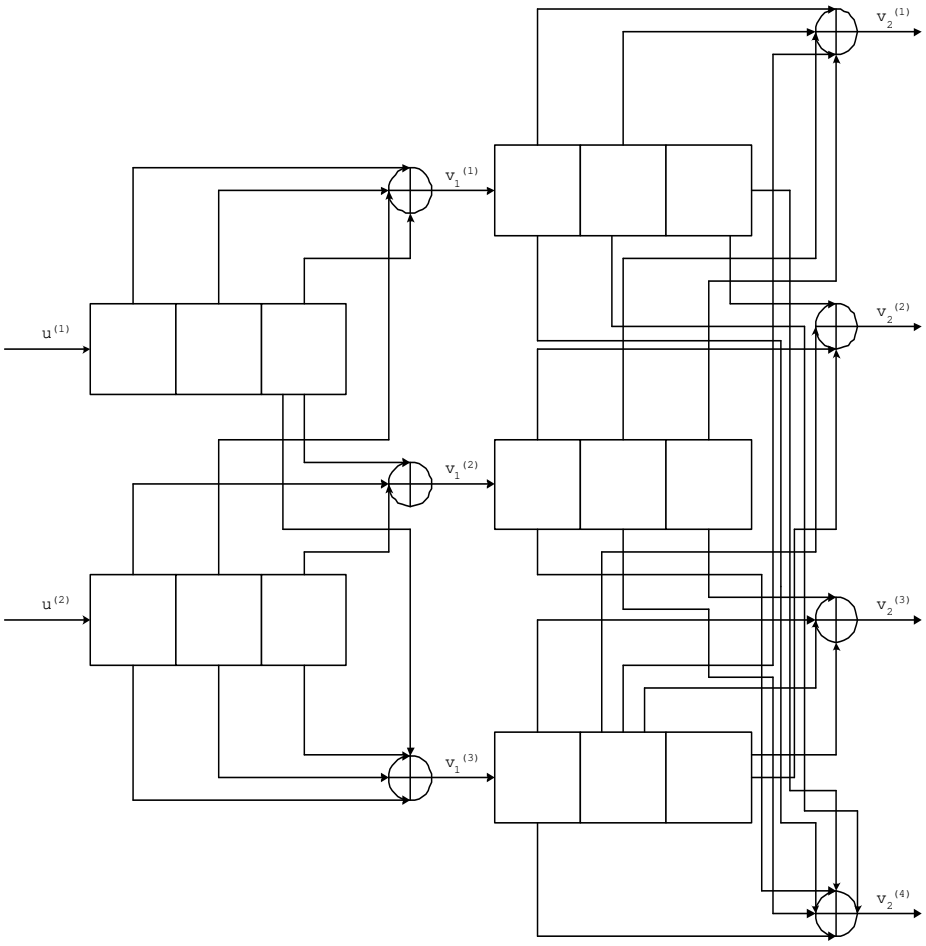

Figura 5.25: Concatenação dos codificadores (3, 2, 2) e (4, 3, 2).

Capítulo 6

Conclusões e Perspectivas para a Pesquisa em CCQs

6.1 Principais Conclusões

Esta tese introduziu uma nova classe de códigos corretores de erros quânticos: a classe dos CCQs concatenados.

Esta classe de CCQs foi construída a partir de um método clássico simples e geral: a concatenação de CCCs. Mais precisamente, usamos o mesmo método de codificação proposto por Shor para construir o seu CBQ de nove qubits: a concatenação de dois códigos, um para a proteção de erros de fase e o outro para a proteção de erros de bit.

O número de codificadores na cadeia de concatenação depende exclusivamente do número de geradores do erro quântico com "naturezas" distintas. Assim, o método de concatenação acima pode ser generalizado para permitir a construção de CCQs capazes de corrigir padrões de erros quânticos mais gerais. Além de ter gerado CCQs, o método que utilizamos serviu para explicitar a relação que existe entre códigos clássicos e quânticos dentro da classe dos códigos quânticos estabilizadores.

Os CCQs têm pelo menos duas grandes vantagens sobre os CBQs: a complexidade de codificação cresce exponencialmente, porém com um expoente menor, e a complexidade do algoritmo de decodificação cresce linearmente, porém com uma inclinação menor, com o número de qubits de informação. Em termos de performance, os CCQs concatenados são

superiores quando comparados aos CBQs concatenados. Há evidências de que o mesmo ocorre quando os comparamos com outras classes de CBQs.

A seguir, apontamos as principais conclusões de cada capítulo.

O Capítulo 1 introduziu o problema da correção de erros quânticos dentro do contexto da engenharia quântica. O Capítulo 2 apresentou as bases da teoria conhecida de códigos quânticos, que inclui somente a classe dos CBQs, e introduziu o formalismo estabilizador. O Capítulo 3 apresentou uma revisão geral da teoria clássica de códigos convolucionais.

No Capítulo 4 descrevemos com detalhes o processo de codificação e de decodificação do CCQ concatenado mais simples de ser construído, o CCQ [4, 1, 3]. Este CCQ é gerado a partir de um único CCC (2, 1, 2) e é capaz de corrigir até um erro quântico geral. Este capítulo introduziu também o conceito de memória convolucional quântica e enfatizou o papel fundamental que desempenha no processo de codificação e de decodificação.

No Capítulo 5 estendemos os resultados do Capítulo 4 para uma classe de CCQs com diferentes taxas, memórias e capacidades de correção. Foram apresentadas várias técnicas de concatenação, dentre as quais destacamos o uso de um ou de dois CCCs primitivos do tipo (n, 1, n), o uso de CCCs ótimos, o uso de CCCs puncionados e o uso de CCCs com MUC ou MUP. Neste capítulo apresentamos alguns exemplos que, apesar de semelhantes, possuem características próprias distintas. O CCQ [12, 3, 3] foi construído a partir da concatenação de dois CCCs, ambos gerados a partir de um único CCC (2, 1, 6). Os CCQs [9, 1, 4] e o CCQ [9, 1, 3] mostraram que o uso de codificadores com MUC ou com MUP pode ser decisivo para diminuir a complexidade ou aumentar a capacidade de correção do código. O CCQ [6, 1, 4] mostrou que a concatenação de CCCs gerados a partir de CCCs com taxa 1/2 e 1/3 pode aumentar a taxa e a capacidade de correção a um custo baixo de complexidade. O CCQ [3, 1, 5] foi construido exclusivamente a partir de CCCs de taxa 1/2, mas à custa de um puncionamento e do aumento do número de memórias. Por fim, fizemos uma breve descrição de dois CCQs de alta performance, o CCQ [4, 2, 4] ("equivalente" a um CCQ [2, 1, 4]), capaz de corrigir um erro quântico geral, e o CCQ [16, 1, 5], capaz de corrigir até seis erros quânticos gerais. A Tabela 6.1 resume os CCQs construídos ao longo desta tese.

$[n, k, m]$	d_{free} do CCC (n, k, m)	erros quânticos gerais corrigíveis
$[4,1,3]$	9	1
$[12,3,3]$	20	2
$[9,1,4]$	24	2
$[9,1,3]$	18	2
$[9,1,4]$	28	3
$[6,1,4]$	18	2
$[3,1,5]$	11	1
$[16,1,5]$	54	6
$[4,2,4]$	9	1
$\vdots$	$\vdots$	$\vdots$

Tabela 6.1: CCQs gerados ao longo da tese.

6.2 Perspectivas para a Pesquisa em CCQs

Dentre as possíveis linhas de pesquisa dentro da classe dos CCQs, sugerimos:

- A construção de novas classes de CCQs, como a equivalente à classe CSS dos CBQs;
- A extensão do grupo de erros e do alfabeto (sistemas com mais de dois níveis) do código;
- A comparação de performances entre CQQs e CBQs;
- O desenho da "máquina quântica": construção dos circuitos quânticos de codificação e de decodificação;
- A introdução do formalismo polinomial e de um formalismo estabilizador mais geral;
- A determinação de uma relação entre a distância quântica e a distância clássica que nos permita obter todos os padrões de erros corrigíveis;
- O uso de outros algoritmos clássicos de decodificação no processo quântico de decodificação: sequencial, lógica majoritária, etc...;

- O estudo da complexidade do processo de geração e de decodificação de CCQs;
- O estudo detalhado das degenerescências de um CCQ concatenado;
- O estudo da geometria de CCQs concatenados;
- A introdução de ruído nas portas quânticas de codificação e de decodificação: extensão da teoria dos CCQs concatenados à teoria da tolerância à falha;
- O estudo das subclasses dos CCQs concatenados; relação entre as sublcasses dos CCCs e dos CCQs;
- A determinação de um limitante inferior e de um limitante superior para o d_{free} dos CCCs concatenados que geram os CCQs concatenados;
- A construção de CCQs concatenados para canais quânticos que introduzem erros correlacionados ou "rajadas" de erros;
- A construção de CCQs concatenados a partir de CCCs puncionados; determinação de um método de obter a matriz verificação de paridade do CCC puncionado a partir da matriz verificação de paridade do CCC original;
- O estudo da minimalidade (ou seja, do menor número de memórias para aquela taxa e capacidade de correção) do codificador e do decodificador concatenado, tanto no processo de codificação, quanto de decodificação;

6.3 Comentários Finais

Esta tese de doutorado, juntamente com a tese de mestrado de Wanessa Carla Gazzoni [22], constituem os dois primeiros trabalhos em codificação quântica apresentados na Faculdade de Engenharia Elétrica e de Computação da UNICAMP. Acreditamos que estes dois trabalhos sejam também as primeiras teses em codificação quântica concluídas no Brasil. Esperamos que estas duas teses, ambas sob a orientação do Prof. Dr. Reginaldo Palazzo Jr., possam contribuir no futuro para aplicações tecnológicas em engenharia quântica.

Em relação à nossa tese de doutorado, julgamos que o método de codificação e de decodificação que apresentamos para a construção de CCQs é simples e geral. Temos certeza de que esta tese ajudará a introduzir a classe dos CCQs a um público maior de físicos, matemáticos e principalmente engenheiros brasileiros.

O interesse pela pesquisa em engenharia quântica é inevitável e irreversível. Muitos governos de países desenvolvidos, com especial destaque para os EUA, já perceberam o potencial teórico e tecnológico desta área de pesquisa e começam a investir maciçamente em projetos de construção de hardware quântico. Também surgem as primeiras iniciativas privadas, como a empresa *MagicQ*, sediada em Nova York, e voltada para a construção não apenas de hardware, mas também de software quântico.

Os resultados do Capítulo 4 desta tese foram resumidos em um artigo de seis páginas sob o título *"A Concatenated [(4, 1, 3)] Quantum Convolutional Code"*, o qual foi enviado para apresentação no *2004 IEEE Information Theory Workshop*, que decorrerá entre 24 e 29 de outubro na cidade de San Antonio, Texas (EUA). O artigo foi aceito com elogios pelos *referees*. Também enviamos para a revista *Physical Review A* um artigo com o título *"Comments on: Quantum Convolutional Error Correcting Codes"*, em que contradizemos, através de um exemplo muito simples (o mesmo da seção 5.2.1), dois teoremas apresentados por H. F. Chau em [7]. Nosso artigo mostra que os resultados da teoria de codificação de bloco quântica não podem estendidos para a teoria de codificação convolucional quântica sem se considerar o conceito de *memória* no processo de construção dos códigos. No momento, aguardamos a revisão deste artigo. Em breve, esperamos também publicar um artigo com os resultados do Capítulo 5. Este artigo tem o título provisório *"Good Concatenated Quantum Convolutional Codes"*.

Mesmo na classe de CCQs concatenados, ainda há muito por fazer. Vamos continuar a pesquisa nesta área no meu pós-doutorado, que estará atrelado ao projeto temático *"Códigos Geometricamente Uniformes em Espaços Homogêneos"*, apoiado pela FAPESP, e do qual já fazemos parte.

Mais do que apresentar alguns resultados, nosso objetivo foi despertar o interesse para esta área de pesquisa e lançar novas perguntas. Estamos vivendo em uma época em que toda a física começa a ser revista sob o ponto de vista da *conservação de informação*,

da mesma forma que foi feito em séculos passados para a energia e a massa. Dentro deste contexto, a proteção de informação quântica será crucial para tornar as aplicações tecnológicas da engenharia quântica viáveis. Acreditamos que este redescobrimento da física valorizará em muito os resultados apresentados nesta tese.

A pesquisa nesta área é muito promissora e merece ser continuada.

Apêndice A

Soluções da Equação Diofantina Linear com Duas Variáveis

A seguir, apresentamos as equações diofantinas lineares para apenas *duas* variáveis [18, 47]. Considere uma equação

$$ax + by = c, \tag{A.1}$$

onde a, $b \in \mathbb{Z}$ e suponha a e b não simultaneamente nulos. Uma solução de (A.1) é, neste contexto, um par $(x_0, y_0) \in \mathbb{Z} \times \mathbb{Z}$ para o qual a igualdade

$$ax_0 + by_0 = c \tag{A.2}$$

é verdadeira. Vejamos em que condições (A.1) admite soluções.

Proposição 1. *Uma equação diofantina $ax+by=c$, em que $a \neq 0$ ou $b \neq 0$, admite solução se, e somente se, $d = mdc(a,b)$ divide c.*

Demonstração: ($\Rightarrow$) Se $(x_0, y_0) \in \mathbb{Z} \times \mathbb{Z}$ é solução, vale a igualdade:

$$ax_0 + by_0 = c. \tag{A.3}$$

Como $d|a$ e $d|b$, então $d|c$.

($\Leftarrow$) Como $d = \text{mdc}(a,b)$, a Proposição 1 garante que $d = ax_0 + by_0$, para um par conveniente $(x_0, y_0) \in \mathbb{Z} \times \mathbb{Z}$. Mas da hipótese $d|c$ segue que $c = dt$, para algum $t \in \mathbb{Z}$. Assim,

$$c = dt = (ax_0 + by_0)t = a(x_0 t) + b(y_0 t), \tag{A.4}$$

o que mostra que $(x_0 t, y_0 t)$ é solução da equação considerada. ■

Proposição 2. *Seja (x_0, y_0) uma solução particular da equação diofantina $ax + by = c$, onde $a \neq 0$ e $b \neq 0$. Então essa equação admite infinitas soluções e o conjunto dessas soluções é:*

$$S = \left\{ \left(x_0 + \frac{b}{d}t,\ y_0 - \frac{a}{d}t \right),\ t \in \mathbb{Z} \right\}, \tag{A.5}$$

onde $d = mdc(a,b)$.

Demonstração: Se indicamos genericamente por (x', y') as soluções de $ax + by = c$, então

$$ax' + by' = c = ax_0 + by_0, \tag{A.6}$$

o que equivale a

$$a(x' - x_0) = b(y_0 - y'). \tag{A.7}$$

Daí, supondo que $a = dr$ e $b = ds$, vem que

$$r(x' - x_0) = s(y_0 - y'), \tag{A.8}$$

onde $\text{mdc}(r,s) = 1$. Como, pela igualdade anterior, r divide $s(y_0 - y')$, então $r|(y_0 - y')$ e, portanto, $y_0 - y' = rt$ para algum $t \in \mathbb{Z}$. De onde tiramos que

$$y' = y_0 - rt = y_0 - \frac{a}{d}t. \tag{A.9}$$

Observando agora que

$$r(x' - x_0) = s(y_0 - y') = srt, \tag{A.10}$$

obtêm-se

$$x' = x_0 + st = x_0 + \frac{b}{d}t. \tag{A.11}$$

É fácil de verificar que, para todo $t \in \mathbb{Z}$, o par

$$\left(x_0 + \frac{b}{d}t,\, y_0 - \frac{a}{d}t\right) \tag{A.12}$$

é solução da equação dada. Isto conclui a demonstração. ■

Corolário 1. *Se a e b não são nulos e $mdc(a,b) = 1$, então a equação diofantina $ax + by = c$ tem conjunto solução não vazio dado por*

$$S = \{x_0 + bt,\, y_0 - at) \mid t \in \mathbb{Z}\}, \tag{A.13}$$

onde (x_0, y_0) é uma de suas soluções particulares.

Apêndice B

Lista de Endereços Eletrônicos

A seguir, apresentamos alguns dos principais endereços na Internet referentes à pesquisa em Computação e Informação Quântica.

B.0.1 Páginas Pessoais e Grupos de Pesquisa

- John Preskill (Caltech, EUA)

  http://www.theory.caltech.edu/people/preskill/

  Notas de Aula: http://www.theory.caltech.edu/people/preskill/ph229

- Daniel Gottesman (Perimeter Institute, Canadá)

  http://perimeterinstitute.ca/people/researchers/dgottesman/

- The Home of the Home Pages Page (Quantum Computation and Information)

  http://web.mit.edu/vandam/www/homes.html

- The Physics of Quantum Information European Research Network

  http://www.uibk.ac.at/c/c7/c704/qinet/

B.0.2 Artigos

- Physical Review Online Archive

  http://prola.aps.prg/

- Archives - Cornell University

  http://arxiv.org/archive/physics/

- Quantum Physics Archives at the Los Alamos National Laboratory

  http://xxx.lanl.gov/archive/quant-ph

B.0.3 Tutoriais

- Oxford Centre for Quantum Computation

  http://www.qubit.org/

- Centre for Quantum Computation - Cambridge University

  http://qubit.damtp.cam.ac.uk

- Centre for Quantum Information and Communication (QuiC)

  Univesité Libre de Bruxelles

  www.quic.ulb.ac.be

- Introduction to Quantum Error Correction

  http://www.c3.lanl.gov/k̃nill/quip/ecprhtml/

B.0.4 Workshops

- 2004 IEEE INFORMATION THEORY WORKSHOP

  San Antonio, Texas, October 24-29, 2004

  San Antonio Marriott Riverwalk Hotel

  http://ee-wcl.tamu.edu/itw2004/

- Workshop "Computação Quântica"

  Brasília, DF - 30 de agosto a 3 de setembro de 2004

  Direção: Amir Caldeira (UNICAMP, Brasil), Gianni Blatter (ETHZ, Zurique, Suiça) e Luiz Davidovich (UFRJ, Brasil)

http://iccmp.unb.br/eventos/

- Congresso Nacional de Matemática Aplicada e Computacional (CNMAC)

  Porto Alegre, RS - 13 a 16 de setembro

  Minicurso: Computação Quântica

  Renato Portugal (LNCC), Carlile Campos Lavor e Luiz Mariano Carvalho (UERJ) e Nelson Maculan Filho (UFRJ)

- Quantum Computation and Information Conferences

  http://qubit.chem.utoronto.ca/qc-conferences.html

B.0.5 Empresas

- MagiQ - Quantum Information Solutions for the Real World

  http://www.magiqtech.com

- Quantum Computation and Information Physics at IBM Research Yorktown

  http://www.research.ibm.com/quantuminfo/

B.0.6 LateX

- "Introdução ao LateX"

  http://www.mat.ufmg.br/ regi/latex/

- *"The Not So Short Introduction to LateX 2e"*

  http://www.ctan.org

Referências Bibliográficas

[1] A. A. Albert. *Modern Higher Algebra*. University of Chicago Press, 1947.

[2] J. S. Bell. *On the Einstein-Podolsky-Rosen paradox*. Physics, 1, pages 195-200, 1964.

[3] C. H. Bennett. *Logical reversibility of computation*. IBM J. Res. Dev., 17, pages 525-532, 1973.

[4] C. H. Bennett. *The thermodynamics of computation - a review*. Int. J. Theor. Phys., 21, pages 905-940, 1982.

[5] C. B. Bennett; E. Bernstein; G. Brassard and U. Vazirani. *Strengths and weaknesses of quantum computing*. SIAM Journal on Computing, vol. 26, nº 5, pages 1411-1473.

[6] A. R. Calderbank and P. W. Shor. *Good quantum error-correcting codes exist*. Phys. Rev. A, 54(2), pages 1098-1105, 1996.

[7] H. F. Chau. *Quantum convolutional error correcting codes*. Phys. Rev. A, 58(2), pages 905-909, 1998.

[8] H. F. Chau. *Good quantum convolutional error correction codes and their decoding algorithm exist*. Phys. Rev. A, 60(3), pages 1966-1974, 1999.

[9] J. I. Cirac and P. Zoller. *Quantum computations with cold trapped ions*. Phys. Rev. Lett., 74(20), 4091-4094, 1995.

[10] J. B. Cain; G. C. Clark and J. M. Geist. *Punctured convolutional codes of rate (n-1)/n and simplified maximum-likelihood decoding*. IEEE Inf. Theory, IT-25(1), pages 97-100, 1979.

[11] R. Cleve. *Quantum stabilizer codes and classical linear codes.* Phys. Rev. A, 55(6), pages 4054-4059, 1997.

[12] D. Deutsch. *Quantum theory, the Church-Turing Principle and the universal quantum computer.* Proc. R. Soc. Lond. A, 400, page 97, 1985.

[13] D. Deutsch and A. K. Ekert. *Quantum Computation.* Disponível em www.qubit.org.

[14] A. Dholakia. *Introduction to Convolutional Codes with Applications.* Kluwer Academic Publishers, Norwell, MA, 1994.

[15] D. Dieks. *Communication by EPR devices.* Phys. Lett. A, 92(6), pages 271-272, 1982.

[16] C. C. Tannoudji; B. Diu and F. Laloë. *Quantum Mechanics - Volume One.* John Wiely and Sons, New York, 1977.

[17] C. C. Tannoudji; B. Diu and F. Laloë. *Quantum Mechanics - Volume Two.* John Wiely and Sons, New York, 1977.

[18] H. H. Domingues. *Fundamentos da Aritmética.* Atual Editora, São Paulo, 1998.

[19] H. H. Domingues e G. Iezzi. *Álgebra Moderna.* Atual Editora, São Paulo, 1979.

[20] A. K. Ekert. *Quantum cryptography based on Bell's theorem.* Phys. Rev. Lett., 67(6), pages 661-663, 1991.

[21] R. P. Feynman. *Simulating physics with computers.* Int. J. Theor. Phys., 21, page 467, 1982.

[22] W. C. Gazzoni. *Propriedades Algébricas e Geométricas dos Códigos de Bloco Quânticos.* Tese de Mestrado - FEEC, UNICAMP, Campinas, SP, Abril de 2004.

[23] N. Gershenfeld and I. Chuang. *Bulk spin resonance quantum computation.* Science, 275, page 350, 1997.

[24] A. Gill. *Linear Sequential Circuit - Analysis, Synthesis, and Applications.* McGraw-Hill, New York, 1966.

[25] D. Gottesman. *A class of quantum error-correcting codes saturing the quantum Hamming bound*. Phys. Rev. A, 54(3), pages 1862-1868, 1996.

[26] D. Gottesman. *Stabilizer codes and quantum error correction*. PhD Thesis - California Institute of Technology, Pasadena, CA, 1997.

[27] L. K. Grover. *A fast quantum mechanical algorithm for database search*. Proceedings of the 28th Annual ACM Symposium on Theory of Computation, pages 212-219, 1996.

[28] J. A. Heller. *Sequential decoding: short constraint length convolutional codes*. Space Program Summary, vol. 3, pages 37-54 - Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, 1968.

[29] C. Monroe; D. M. Meekhof; B. E. King; W. M. Itano and D. J. Wineland. *Demonstration of a fundamental quantum logic gate*. Phys. Rev. Lett., 75(25), pages 4714-4717, 1995.

[30] R. Johannesson and Z. Wan. *A linear algebra approach to minimal convolutional encoders*. IEEE Trans. Inf. Theory, 39(4), pages 1219-1233, 1993.

[31] D. J. Costello Jr. *Construction of Convolutional Codes for Sequential Decoding*. PhD Thesis - University of Notre Dame, Notre Dame, IN, 1969.

[32] D. J. Costello Jr. *A construction technique for random-error-correcting convolutional codes*. IEEE Trans. Inf. Theory, IT-15(5), pages 631-636, 1969.

[33] D. J. Costello Jr. *Free distance bounds for convolutional codes*. IEEE Trans. Inf. Theory, IT-20(3), pages 356-365, 1974.

[34] G. D. Forney Jr. *Convolutional codes I: algebraic structure*. IEEE Trans. Inf. Theory, IT-16(6), pages 720-738, 1970.

[35] G. D. Forney Jr. *Structural analysis of convolutional codes via dual codes*. IEEE Tran. Inf. Theory, IT-19(4), pages 512-518, 1973.

[36] G. D. Forney Jr. *Convolutional codes II: maximum-likelihood decoding.* Information and Control, 25, pages 222-266, 1974.

[37] E. Knill and R. Laflamme. *A theory of quantum error-correcting codes.* Phys. Rev. A, 55(2), pages 900-911, 1997.

[38] R. Landauer. *Irreversibility and heat generation in the computing process.* IBM J. Res. Dev., 5, page 183, 1961.

[39] G. S. Lauer. *Some optimal partial-unit-memory codes.* IEEE Trans. Inf. Theory, IT-25(2), pages 240-243, 1979.

[40] L. N. Lee. *Short unit-memory byte-oriented binary convolutional codes having maximal free distance.* IEEE Trans. Inf. Theory, IT-22(3), pages 349-352, 1976.

[41] S. Lin and D. J. Costello Jr. *Error Control Coding - Fundamentals and Applications.* Prentice-Hall, Englewood Cliffs, NJ, 1983.

[42] Q. A. Turchette; C. J. Hood; W. Lange; H. Mabuchi and H. J. Kimble. *Measurement of conditional phase shifts for quantum logic.* Phys. Rev. Lett., 75(25), pages 4710-4713, 1995.

[43] J. L. Massey. *Catastrophic error-propagation in convolutional codes.* Proc. of the 11th Midwest Symp. Cir. Th., pages 583-587, Notre Dame, IN, 1968.

[44] J. L. Massey. *Coding Theory.* Handbook of Applicable Mathematics: Vol. V, Part B, Chapter 16, pages 623-676, New York, 1985.

[45] J. L. Massey and M. K. Sain. *Inverses of linear sequential circuits.* IEEE Trans. Comp., C-17, pages 330-337, 1968.

[46] R. J. McEliece and I. Onyszchuk. *The extended invariant fator algorithm with application to the Fourier analysis of convolutional codes.* Proc. IEEE Int. Symp. Info. Th., page 142, 1993.

[47] T. Nagell. *Introduction to Number Theory.* John Wiley and Sons, New York, 1951.

[48] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, UK, 2000.

[49] H. Ollivier and J. P. Tillich. *Quantum convolutional codes: fundamentals*. Disponível em www.arxiv.org/abs/quant-ph/0401134.

[50] H. Ollivier and J. P. Tillich. *Description of a quantum convolutional code*. Phys. Rev. Let., 91(17), pages 1779021-1779024, 2003.

[51] R. Laflamme; C. Miquel; J. P. Paz and W. K. Zurek. *Perfect quantum error correction code*. Phys. Rev. Lett., 77(1), pages 198-201, 1996.

[52] P. Piret. *Convolutional Codes: An Algebraic Approach*. MIT Press, Cambridge, MA, 1988.

[53] A. Einstein; B. Podolsky and N. Rosen. *Can quantum-mechanical description of physical reality be considered complete ?* Phys. Rev., 47(10), pages 777-780, 1935.

[54] H. Pollatsek. *Quantum error correction: classic group theory meets a quantum challenge*. The Mathematical Association of America, 108, pages 932-962, 2001.

[55] J. Preskill. *Notas de Aulas - Informação e Computação Quântica*. Disponível em www.theory.caltech.edu/people/preskill/ph229.

[56] I. S. Reed and T. K. Truong. *New syndrome decoder for (n,1) convolutional codes*. Electronics Letters, 19(9), pages 344-346, 1983.

[57] I. S. Reed and T. K. Truong. *New syndrome decoding techiniques for the (n,k) convolutional codes*. IEE Proceedings, 131-F(4), pages 412-416, 1984.

[58] I. S. Reed and T. K. Truong. *Error-trellis syndrome decoding techniques for convolutional codes*. IEE Proceedings, 132-F(2), pages 77-83, 1985.

[59] J. J. Sakurai. *Modern Quantum Mechanics*. Addison-Wesley Publishing Company, USA, 1994.

[60] J. P. M. Schalkwijk and A. J. Vinck. *Syndrome decoding of convolutional codes*. IEEE Trans. Comm., COM-23(7), pages 789-792, 1975.

[61] J. P. M. Schalkwijk and A. J. Vinck. *Syndrome decoding of binary rate 1/2 convolutional codes*. IEEE Trans. Comm., COM-24(9), pages 977-985, 1976.

[62] J. P. M. Schalkwijk and A. J. Vinck. *Syndrome decoding of binary rate k/n convolutional codes*. IEEE Trans. Inf. Theory, IT-24(5), pages 553-562, 1978.

[63] A. R. Calderbank; E. M. Rains; P. W. Shor and N. J. A. Sloane. *Quantum error correction and orthogonal geometry*. Phys. Rev. Lett., 78(3), pages 405-408, 1997.

[64] A. R. Calderbank; E. M. Rains; P. W. Shor and N. J. A. Sloane. *Quantum error correction via codes over GF(4)*. IEEE Transactions on Information Theory, 44, pages 1369-1387, 1998.

[65] P. W. Shor. *Algorithms for quantum computation: discrete logarithms and factoring*. Proceedings of the 35th Annual Symposium on Fundamentals of Computer Science, pages 124-134, 1994.

[66] P. W. Shor. *Scheme for reducing decoherence in quantum computer memory*. Phys. Rev. A, 52 (4), pages 2493-2496, 1995.

[67] T. Siegfried. *O Bit e o Pêndulo*. Editora Campus, Rio de Janeiro, 2000.

[68] C. H. Bennett; D. P. DiVicenzo; J. A. Smolin and W. K. Wooters. *Mixed state entanglement and quantum error correction*. Phys. Rev. A, 54(5), pages 3824-3851, 1996.

[69] A. M. Steane. *Error correcting codes in quantum theory*. Phys. Rev. Lett., 77(5), pages 793-797, 1996.

[70] A. M. Steane. *Multiple particle interference and quantum error correction*. Proc. R. Soc. London A, 452, pages 2551-76, 1996.

[71] L. Szilard. *Uber die entropieverminderung in einen thermodynamischen system bei eingriffen intelligenter wesen*. Z. Phys., 53, pages 840-856, 1929.

[72] A. J. Viterbi. *Convolutional codes and their performance in communication systems.* IEEE Trans. Comm., COM-19(5), pages 751-771, 1971.

[73] A. J. Viterbi and J. K. Omura. *Principles of Digital Communication and Coding.* McGraw-Hill, New York, 1979.

[74] J. Wheeler and W. H. Zurek. *Quantum Theory and Measurement.* Princeton University Press, 1983.

[75] W. K. Wootters and W. H. Zurek. *A single quantum cannot be cloned.* Nature, 299, pages 802-803, 1982.