量子ビット連結性制約検査のための依存型システム

脇坂 遼　五十嵐 淳

量子回路モデルに基づく量子プログラムでは，量子ゲートを量子ビットに適用することで計算を行う．量子ゲートのうち，CNOT ゲートは量子ビットペアに適用される特殊なゲートである．この特殊性から，実際の量子コンピュータ上では CNOT ゲートは特定の量子ビットペアにしか適用できない．この制約は qubit connectivity とよばれ，coupling graph という，量子ビットを頂点とし，CNOT ゲートが適用できる量子ビット対を辺とするグラフで表現される．本研究では，量子プログラムが qubit connectivity 制約を満たすかどうかを検証するために，Selinger らの量子ラムダ計算に依存型を導入した型システムを提案する．そして，与えられた coupling graph のもとで型付け可能なプログラムが，その実行中の CNOT ゲートの適用時に制約に違反しないことを証明する．

1　はじめに

量子ゲートを用いた量子計算においては，量子ビットに量子ゲートを作用させることで計算を行う．量子計算のためのプログラミング言語のモデルとして，ラムダ計算に量子ビットと量子ゲートを導入したような体系 [19] [14] [15] [1] [12] [4] [3] が研究されてきている．これらの体系では，線形型などを使って，量子複製不可能定理 (no-cloning theorem) [20] に由来する量子プログラム独特の制約を静的に検査するものが多い．

量子プログラムやアルゴリズムを設計する際には，量子ゲートを任意の量子ビットにたいして正確に適用できることを前提にすることが多いが，実際の量子コンピュータ上で動作させる際には量子誤りや qubit connectivity　といったハードウェアに由来する制約を考慮する必要がある．

Qubit connectivity 制約 (以下，単に連結性制約と呼ぶ) とは，量子ゲートのうち CNOT ゲートは特定の量子ビットペアにしか適用できないという制約である．この制約は，量子ビットを頂点とし，CNOT ゲートが適用できるペアを辺とする，coupling graph と呼ばれるグラフで表現される．Coupling graph の辺は有向辺であり，始点を control qubit，終点を target qubit とする CNOT ゲートが実行できる．Coupling graph は量子コンピュータアーキテクチャ毎に固有のものであり，例えば IBM の qx2 量子コンピューター [2] は図 1 に表される coupling graph をもつ．量子プログラムを実機上で動かす際には，対象のアーキテクチャの coupling graph に対して動かしたいプログラムが制約を満たしているか，すなわち，そのプログラムを実行した際に CNOT ゲートが適用されうる任意の量子ビットペアが，coupling graph 上で辺で直

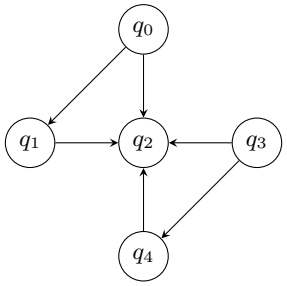

図 1　IBM qx2 量子コンピュータの coupling graph

---

* A Dependent Type System for Qubit Connectivity Verification
This is an unrefereed paper. Copyrights belong to the Author(s).

Ryo Wakizaka, Atsushi Igarashi, 京都大学大学院情報学研究科, Graduate School of Informatics, Kyoto University.

接隣接していることを検証する必要がある．しかしながら，プログラムが連結性制約に反していないことを静的に保証する枠組みはあまり研究されていない．

以上を踏まえ，本研究では関数型量子プログラムのための連結性制約を満たすことを保証するための静的型システムを提案する．この型システムは，依存型を使うことで，coupling graph 上で隣接していない量子ビットペアへの CNOT ゲートの適用を禁止するものとなっている．本論文の具体的な技術的貢献は以下の通りである．

- Selinger らの量子ラムダ計算 [15] を拡張した体系 $\lambda Q^{\Pi\rightsquigarrow}$ の形式的定義を与える．
- 型付け可能なプログラムは連結性制約を満たすことを意味する型安全性定理を示す

本論文は以下のように構成される．第 2 節では，提案する体系について概要を述べる．第 3 節では，提案する体系の形式的な定義，型システムおよび操作的意味論について述べる．第 4 節では，提案する体系が備える性質について述べる．第 5 節では，型がつくプログラムの具体例を述べる．第 6 節では関連研究について述べ，第 7 節で結論と今後の課題を述べる．

2 提案体系の概要

本論文で与える $\lambda Q^{\Pi\rightsquigarrow}$ の型システムは，線形型，singleton 型，依存関数型などを使って，与えられたプログラムが連結性制約を満たすことを検査する．以下では，体系の形式的定義を与える前にそれらのアイデアを概観する．

2.1 線形型

線形型は，値が一度しか使われないことを示すための型である．第 1 節で述べたとおり，量子ビットは複製不可能であるため，量子ビットがプログラム中で複製されないことを線形型によって保証する．例えば，量子ビット q を受け取って複製し，ペアを作るような以下のプログラム M_1 は線形型システムにより型エラーとすることができる．

$$M_1 \equiv (\lambda x.\langle x, x\rangle)\ q$$

$\lambda Q^{\Pi\rightsquigarrow}$ の型システムでは，各型コンストラクタに use と呼ばれる，その型を持つ値が何回使用されるかを表す情報を付加する [18]．use κ は $0, 1, \omega$ のいずれかの値をとり，$\kappa = 0$ ならばその値は使えないこと，$\kappa = 1$ ならば高々1 回だけ用いられることを表し，$\kappa = \omega$ なら使用回数に制限がないことを表す．例えば，ペア型 $T_1 \otimes^{\kappa} T_2$ は，ペアから κ 回要素が取り出せることを示す．量子ビット型を $\mathtt{qbit}$ とすれば，高々1 回だけ用いられる量子ビット型は $\mathtt{qbit}^1$ と表記される．

型環境中の変数の型につけられた use はその変数が何回使えるかを表すので，ペアや関数適用など複数の部分式を持つ式に対する型付け規則では，各部分式のための型環境の use の合計を考慮する必要がある．例えば，ペアに対する型付け規則はだいたい以下のようになる．

$$\frac{\Gamma_1 \vdash M : T_1 \qquad \Gamma_2 \vdash N : T_2}{\Gamma_1 + \Gamma_2 \vdash \langle M, N\rangle : T_1 \otimes^{\kappa} T_2}$$

結論の $\Gamma_1 + \Gamma_2$ は，ふたつの型環境に現れる型の use を足しあわせる演算である．これによると，

$$x : \mathtt{unit}^{\omega} \vdash \langle x, x\rangle : \mathtt{unit}^{\omega} \otimes^{\kappa} \mathtt{unit}^{\omega}$$

は，ペアの両要素 x の use が ω であり $\omega + \omega = \omega$ となるので導出できるが，

$$x : \mathtt{qbit}^1 \vdash \langle x, x\rangle : \mathtt{qbit}^1 \otimes^{\kappa} \mathtt{qbit}^1$$

は，ペアの両要素 x の use 1 を足すと $1 + 1 = \omega$ になるので導出できない．

本節の以降の説明では，簡単のため use を省略することにする．

2.2 Singleton type による量子ビット型の詳細化

次に，依存型による連結性制約の検証について述べる．連結性制約で問題になるのは CNOT ゲートであるから，CNOT ゲートを適用できる量子ビットペアを型付け規則で制限できるような型システムを構築する必要がある．このために，適用できる量子ビットペアの情報，すなわち coupling graph の情報を型判断にのせ，プログラム中の量子ビット変数が coupling graph 上のどの量子ビットに対応するのかを追跡する．

このために，$\lambda Q^{\Pi\rightsquigarrow}$ では，coupling graph 上のノードに対応するラベル i, j を導入し，各量子ビットの

型を $Q(i), Q(j)$ と表現することにより，型レベルで量子ビットの追跡を行えるようにする．この i, j を `qidx` と呼ぶ．型 $Q(i)$ は `qidx` i に対応するひとつの量子ビットを表すという意味で singleton 型[6] の一種と考えられる．さらに，量子ビットペア i_1, i_2 が coupling graph 上で i_1 から i_2 への辺があるという条件を $i_1 \rightsquigarrow i_2$ と表し，型環境中に変数の型宣言とともに $i_1 \rightsquigarrow i_2$ を並べることにする．この時，CNOT ゲートの型付け規則は以下のようになる (use は省略している)．

$$\frac{i, j : \texttt{qidx} \in \Gamma \qquad i \rightsquigarrow j \in \Gamma}{\Gamma \vdash \texttt{cnot}[i, j] : Q(i) \otimes Q(j) \to Q(i) \otimes Q(j)}$$

つまり，i, j に対応する量子ビットペア (つまり，型 $Q(i), Q(j)$ をもつ式) に CNOT ゲートを適用したい場合，型環境に $i \rightsquigarrow j$ が含まれることが要求される．

このように，`qidx` を使って量子ビット型を詳細化し，連結性制約を型環境に含めることで，連結性制約を満たしているかを検証することができるようになる．

また `qidx` によって，Selinger らの量子ラムダ計算 [15] では区別できなかった型を区別できるようになる．その例が以下の f_1, f_2 である．

$$f_1 \equiv \lambda x : \texttt{qbit}.\lambda y : \texttt{qbit}.\langle x, y\rangle$$
$$f_2 \equiv \lambda x : \texttt{qbit}.\lambda y : \texttt{qbit}.\langle y, x\rangle$$

Selinger らの量子ラムダ計算では，量子ビットはすべて同じ型 `qbit` を持つため，f_1, f_2 の型はどちらも $\texttt{qbit} \to \texttt{qbit} \to \texttt{qbit} \otimes \texttt{qbit}$ となる．しかし，f_1, f_2 では戻り値の量子ビットの順番が入れ替わっており，個々の量子ビットを区別する今回の設定ではこれらを区別したい．`qidx` を用いれば，f_1, f_2 の型はそれぞれ

$$f_1 : Q(i) \to Q(j) \to Q(i) \otimes Q(j)$$
$$f_2 : Q(i) \to Q(j) \to Q(j) \otimes Q(i)$$

と表され，区別できるようになる．

2.3 依存関数型による再利用性の向上

`qidx` を使うことで，上の f_1, f_2 の型の例のように，より細かくプログラムの挙動が型で表現できるが，一方でそのままだと細かすぎる場合がある．例えば，上記の f_1 は，i, j に対応する量子ビットのみをペアにすることしかできず，他の量子ビットに適用することができない．

この問題に対処するために，$\lambda Q^{\Pi\rightsquigarrow}$ では `qidx` についての抽象化を許し量子ビットを引数に取る関数の再利用性を高める．例えば前節の関数 f_1 を，i, j について抽象化し

$$\lambda i, j : \texttt{qidx}.\lambda x : Q(i).\lambda y : Q(j).\langle x, y\rangle$$

とすることができる [†1]．この関数は，ふたつの `qidx`，それらの `qidx` に対応する量子ビット x, y を受け取り，ペアにして返す関数となり，引数として与える `qidx` を変えることで，様々な量子ビットに適用することができる．この関数の型は，依存関数型を使って

$$\Pi i, j : \texttt{qbit}.Q(i) \to Q(j) \to Q(i) \otimes Q(j)$$

と表される．

また，$\lambda Q^{\Pi\rightsquigarrow}$ では `qidx` パラメータに対する連結性制約を型で表現することができる．これは例えば以下のプログラムを考えるとわかりやすい．

$$\lambda i, j : \texttt{qidx}.\lambda x : Q(i).\lambda y : Q(j).\texttt{cnot}\,\langle x, y\rangle$$

このプログラムは 2 つの量子ビットを 1 つずつ受け取って `cnot` を適用する関数であるが，パラメータ i, j は連結性制約をみたす必要がある．このことを表現するために，$i \rightsquigarrow j \Rightarrow T$ という形の型を導入する．この型を持つ式は，型文脈に $i \rightsquigarrow j$ が存在する時に T として使うことができる．例えば，

$$\Pi i, j : \texttt{qbit}.i \rightsquigarrow j \Rightarrow Q(i) \to Q(j) \to Q(i) \otimes Q(j)$$

という型で i, j が隣接した量子ビットであることを型で表現することができる．この型は Jones の qualified type [9] の一種と考えられる．ただし，Jones の形式化とは異なり，$i \rightsquigarrow j \Rightarrow T$ 型のための明示的な項の構文を用意する．例えば，上の関数は，$\lambda Q^{\Pi\rightsquigarrow}$ では以下のように与えられる．

$$f_3 \equiv \lambda i, j : \texttt{qidx}.\lambda\_ : i \rightsquigarrow j.\ \lambda x : Q(i).\lambda y : Q(j).\texttt{cnot}\,\langle x, y\rangle$$
$$f_3 : \Pi i, j : \texttt{qbit}.i \rightsquigarrow j \Rightarrow\ Q(i) \to Q(j) \to Q(i) \otimes Q(j)$$

$\lambda\_$ の部分が型 $i \rightsquigarrow j \Rightarrow$ を導入する．制約が満たされていることの検査は，この関数を _ に適用することで行う．例えば，f_3 であれば以下のようになる．(変

†1 $\lambda i, j : \texttt{qidx}.M$ は $\lambda i : \texttt{qidx}.\lambda j : \texttt{qidx}.M$ の略記である．以降，連続するラムダ抽象は同様に略記する．

数 q_1, q_2 はそれぞれ型 $Q(i), Q(j)$ を持つとする．)

$$f_3\ i\ j\ \_\ q_1\ q_2$$

制約の解決時に，要求されている制約が型環境に存在する場合に限ってその適用に型がつくようにする．構文として _ を用いるのは，制約が満たされているかだけが重要で，制約に名前をつける必要がないからである．

2.4 連結性制約の抽象化

最後に，連結性制約の抽象化に関して述べる．動機付けとして，量子ビットを操作する関数を引数とするような以下の高階関数 f_4 を考える．(ここで $T \equiv Q(i_1) \otimes Q(i_2) \otimes Q(i_3)$ とする．)

$$f_4 \equiv \lambda g : T_g.\lambda t : T.g\ \_\ (g\ \_\ t)$$

f_4 の関数引数 g は，3 量子ビットの組を受け取ってその 3 量子ビットの組を返すとすると，この f_4 は，量子ビットの組 t に対して同じ回路 g を繰り返し適用するようなプログラムのパターンを表していると考えることができる．しかし，g の型 T_g は，g がどんな連結性制約を要求するかで異なる型になってしまう．例えば，g が i_1, i_2 が隣接していることを要求する関数であれば，T_g は

$$i_1 \rightsquigarrow i_2 \Rightarrow T \to T$$

とすべきだし，i_2, i_3 が隣接していることを要求する関数であれば，

$$i_2 \rightsquigarrow i_3 \Rightarrow T \to T$$

となる．しかし，このように関数引数が課す制約によって f_4 の型が固定されてしまうのは再利用性の観点から問題である．

この問題を解決するため，我々は $\lambda Q^{\Pi\rightsquigarrow}$ に連結性制約を表す変数 α と，α についての抽象化を導入し，連結性制約についての多相性を表現できるようにする．具体的には,「任意の n 項連結性制約述語 α について T 型」という意味の $\Pi\alpha : \mathtt{graph}^{(n)}.T$ という型を導入する．$\mathtt{graph}^{(n)}$ は α が qidx 上の n 項述語であることを表しており，$\alpha(i_1, \ldots, i_n)$ は抽象的な連結性制約として $\alpha(i_1, \ldots, i_n) \Rightarrow T$ のような形で現れる．例えば，f_4 は α についての抽象を使って，

$$f_4' \equiv \lambda\alpha : \mathtt{graph}^{(3)}.\lambda g : (\alpha(i_1, i_2, i_3) \Rightarrow T \to T).$$
$$\lambda t : T.g\ \_\ (g\ \_\ t)$$
$$: \Pi\alpha : \mathtt{graph}^{(n)}.((\alpha(i_1, i_2, i_3) \Rightarrow T \to T) \to T \to T)$$

と定義することで，後から連結性制約を具体化できるようになる．

具体的な n 項述語は，

$$(i_1, \ldots, i_n)i_{11} \rightsquigarrow j_{12}, \ldots i_{m1} \rightsquigarrow i_{m2}$$

の形で表し，$\Pi\alpha$ 型の抽象項に渡すことで α を具体化する．例えば，

$$f_4'\ ((j_1, j_2, j_3)j_1 \rightsquigarrow j_2)$$

の型は α を $((j_1, j_2, j_3)j_1 \rightsquigarrow j_2)$ で具体化した

$$(i_1 \rightsquigarrow i_2 \Rightarrow T \to T) \to T \to T$$

になり，

$$f_4'\ ((j_1, j_2, j_3)j_2 \rightsquigarrow j_3)$$

の型は

$$(i_2 \rightsquigarrow i_3 \Rightarrow T \to T) \to T \to T$$

となる．このように，連結性制約述語についての抽象化を導入することで，様々な制約を持つ関数を引数に取る高階関数を定義することができる．

3 体系 $\lambda Q^{\Pi\rightsquigarrow}$

この節では，Selinger らの量子ラムダ計算 [15] を元に，量子プログラムの coupling graph 解析のための型システムを備えた体系 $\lambda Q^{\Pi\rightsquigarrow}$ の形式的定義を与える．既に述べたように，Selinger らの体系における線形型では，複製可能な型は型コンストラクタ $!T$ を使っているが，$\lambda Q^{\Pi\rightsquigarrow}$ では Turner らの use [18] を用いている．(use 0 は Igarashi ら [8] と Mogensen [10] によっている．) 加えて，量子ビットを動的に生成するプリミティブ new を体系から取り除いている．

3.1 構文

i, j は qidx 変数，α は連結性制約，U は 1 量子ビットゲートを表すメタ変数であるとする．連結性制約 K，use κ，型 T，型環境 Γ，定数 c，項 M，値 V を以下の BNF で定義する．

$$
\begin{aligned}
K &::= \bullet \mid K, \alpha(i_1, \ldots, i_n) \mid K, i \rightsquigarrow j \\
\kappa &::= 0 \mid 1 \mid \omega \\
T &::= Q(i)^{\kappa} \mid \mathtt{unit}^{\kappa} \mid T \to^{\kappa} T \mid T \otimes^{\kappa} T \mid T \oplus^{\kappa} T \mid \\
&\quad K \Rightarrow T \mid \Pi i : \mathtt{qidx}.T \mid \Pi \alpha : \mathtt{graph}^{(n)}.T \\
\Gamma &::= \bullet \mid \Gamma, x : T \mid \Gamma, i : \mathtt{qidx} \mid \Gamma, \alpha : \mathtt{graph}^{(n)} \mid \\
&\quad \Gamma, K \\
c &::= () \mid \mathtt{meas}[i] \mid \mathtt{cnot}[i, j] \mid U[i] \\
M &::= c \mid x \mid \lambda x : T.M \mid MM \mid \langle M, M \rangle \mid \\
&\quad \mathtt{let}\ \langle x, y \rangle = M\ \mathtt{in}\ M \mid \mathtt{inl}\ M \mid \mathtt{inr}\ M \mid \\
&\quad \mathtt{match}\ M\ \mathtt{with}\ (x \mapsto M \mid y \mapsto M) \mid \\
&\quad \mathtt{let\ rec}\ f = \lambda x.M\ \mathtt{in}\ M \mid \\
&\quad \lambda i : \mathtt{qidx}.M \mid M\ i \mid \lambda \alpha : \mathtt{graph}^{(n)}.M \mid \\
&\quad M\ (i_1, \ldots, i_n)K \mid \lambda\_ : K.M \mid M\_ \\
V &::= c \mid x \mid \lambda x : T.M \mid \lambda i : \mathtt{qidx}.M \mid \lambda\_ : K.M \mid \\
&\quad \lambda \alpha : \mathtt{graph}^{(n)}.M \mid \langle V, V \rangle \mid \mathtt{inl}\ V \mid \mathtt{inr}\ V
\end{aligned}
$$

まず，連結性制約 K について説明する．連結性制約 K は直感的には $\mathtt{qidx}$ 上の関係 $i \rightsquigarrow j$ の列であるが，連結性制約述語 α を用いた $\alpha(i_1, \ldots, i_n)$ という制約が現れてもよい．また，$\bullet$ は空列を表し，しばしば省略する．

既に述べたように，use は 0, 1, ω のいずれかである．後述するように，use が 0 の型を持つ変数は参照できないため 0 を除いて定義を与えることも可能と思われるが，主に型環境の足し算などの定義が簡単になるため採用している．型のうち，$Q(i)$ は量子ビット型である．量子ビット型は $\mathtt{qidx}\ i$ に依存する依存型であり，i によって異なる量子ビットを型レベルで区別することができる．さらに，(トップレベルに現れる) 量子ビット変数の型の $\mathtt{qidx}$ は互いに相異なるようにすることで (定義 4.1)，異なる量子ビットに同じ型が割り当てられないようにする．また，量子ビット型の use は 0 または 1 でなければならない．これは構文ではなく，規則を使って型の well-formedness として強制する．$K \Rightarrow T$ 型を持つ項は，連結性制約 K を満たすときにのみ T として用いることができることを表す型である．$\Pi i : \mathtt{qidx}.T$ 型は，T 型が i に関して多相的に振る舞う依存関数型である．i は束縛変数である．$\Pi \alpha : \mathtt{graph}^{(n)}.T$ 型は，n 項連結性制約述語 α に関して多相的に振る舞う依存関数型である．α は束縛変数である．

型環境は，変数の型宣言，$\mathtt{qidx}$ 変数や連結性制約述語変数の宣言，連結性制約の仮定の列である．連結性制約のときと同様，$\bullet$ は空列を表す記号であり，しばしば省略する．

最後に，項について説明を与える．項のほとんどは Selinger らの量子ラムダ計算と同じであるため，拡張した点のみを述べる．まず $\lambda i : \mathtt{qidx}.M$ は M が i について多相的に振る舞うような関数を表し (i は束縛変数である)，$M\ i$ によって M を $\mathtt{qidx}\ i$ で具体化して用いる．また，$\lambda \alpha : \mathtt{graph}^{(n)}.M$ は M が n 項連結性制約述語を引数に取る関数であり (α は束縛変数である)，$M\ (i_1, \ldots, i_n)K$ によって α を具体化する．ここで，$(i_1, \ldots, i_n)K$ は制約 K に現れる $\mathtt{qidx}\ i_1, \ldots, i_n$ を束縛した n 項連結性制約述語である．$\lambda\_ : K.M$ は連結性制約 K について明示的に抽象したものであり，$M\_$ とすることでその制約を満たしていれば解消することができる．c は定数を表す項であり，$\mathtt{meas}[i]$, $\mathtt{cnot}[i, j]$ および量子ゲート U である．$\mathtt{meas}[i]$ は $\mathtt{qidx}\ i$ に対する測定を行い古典ビットを返す関数であり，$\mathtt{cnot}[i, j]$ は $\mathtt{qidx}\ i, j$ に対応する 2 量子ビットに適用される CNOT ゲートである．本体系では [15] にならい，古典ビットの型は直和型を使って，$\mathtt{bit} = \mathtt{unit}^{\omega} \oplus^{\omega} \mathtt{unit}^{\omega}$ と定義し，0 と 1 をそれぞれ $\mathtt{inr}\ ()$ と $\mathtt{inl}\ ()$ に対応させる．U は 1 量子ビットゲートを表すメタ変数であり，具体的には X ゲートや H ゲートなどがある．$\mathtt{meas}$ と同様，どの量子ビットに適用するかを $[i]$ で指定する．

また，$\mathtt{qidx}$ の代入，連結性制約述語の代入，項変数への項の代入をそれぞれ $[j/i]$, $[(i_1, \ldots, i_n)K/\alpha]$, $[M/x]$ と書く，これらは全て束縛変数捕捉を避ける代入である．定義は簡単なので省略するが，図 2 に示す連結性制約述語の制約 K への代入 $[(i_1, \ldots, i_n)K/\alpha]K'$ の定義のみ注意が必要である．連結性制約述語 $(i_1, \ldots, i_n)K$ が α へ代入される過程で，α が具体的な引数を伴う場合には K への代入が発生する．例えば，以下のようになる．

$$[(j_1, j_2, j_3)j_1 \rightsquigarrow j_2/\alpha]\alpha(i_1, i_2, i_3) = i_1 \rightsquigarrow i_2$$

$$
\begin{aligned}
[(i_1,\ldots,i_n)K/\alpha]\bullet &= \bullet \\
[(i_1,\ldots,i_n)K/\alpha](K', i \rightsquigarrow j) &= \left([(i_1,\ldots,i_n)K/\alpha]K'\right), i \rightsquigarrow j \\
[(i_1,\ldots,i_n)K/\alpha](K', \beta(j_1,\ldots,j_m)) &= \begin{cases} \left([(i_1,\ldots,i_n)K/\alpha]K'\right), [j_1/i_1,\ldots,j_n/i_n]K & \text{if } \beta=\alpha \text{ and } n=m \\ \left([(i_1,\ldots,i_n)K/\alpha]K'\right), \beta(j_1,\ldots,j_n) & \text{if } \beta \neq \alpha \end{cases}
\end{aligned}
$$

図 2　連結性制約の代入

$+$	0	1	ω
0	0	1	ω
1	1	ω	ω
ω	ω	ω	ω

$\times$	0	1	ω
0	0	0	0
1	0	1	ω
ω	0	ω	ω

表 1　use 上の和積演算

3.2　型システム

まず，use に関する操作，関係，および演算をいくつか定義する．

定義 3.1 型 T に対し，$|T|$ で型 T の最も外側にある use を表す．

$$
\begin{aligned}
|Q(i)^\kappa| &= \kappa \qquad |\mathtt{unit}^\kappa| = \kappa \\
|T \to^\kappa T'| &= \kappa \qquad |T \otimes^\kappa T'| = \kappa \\
|T \oplus^\kappa T'| &= \kappa \qquad |K \Rightarrow T| = |T| \\
|\Pi i : \mathtt{qidx}.T| &= |T| \qquad |\Pi\alpha : \mathtt{graph}^{(n)}.T| = |T|
\end{aligned}
$$

定義 3.2 Use 上の全順序関係を $0 \leq 1 \leq \omega$ と定める．また，型 T, T' が，最も外側の use を除いて等しく，かつ $|T| \leq |T'|$ であるとき，またそのときに限り $T \leq T'$ と書く．また，型環境 Γ, Γ' についての関係 $\Gamma \leq \Gamma'$ を以下で定義する．

$\bullet \leq \bullet$
$(\Gamma'_1, i : \mathtt{qidx}) \leq (\Gamma'_2, i : \mathtt{qidx})$ if $\Gamma'_1 \leq \Gamma'_2$
$(\Gamma'_1, \alpha : \mathtt{graph}^{(n)}) \leq (\Gamma'_2, \alpha : \mathtt{graph}^{(n)})$ if $\Gamma'_1 \leq \Gamma'_2$
$(\Gamma'_1, K) \leq (\Gamma'_2, K)$ if $\Gamma'_1 \leq \Gamma'_2$
$(\Gamma'_1, x : T) \leq (\Gamma'_2, x : T')$ if $\Gamma'_1 \leq \Gamma'_2$ and $T \leq T'$

定義 3.3 Use 上の和および積を表 1 で定める． 型 T, T' が最も外側の use を除いて型の構造および use が等しいとき，型 $T + T'$ を T の最も外側の use を $|T| + |T'|$ としたものと定義する．また，use κ および型 T に対し κT は T の最も外側の use に κ を掛けた型と定義する．さらに，型環境 Γ に対し，$\kappa\Gamma$ は Γ 中の型に κ を掛けて得られる型環境と定義する．

定義 3.4 型環境 Γ_1, Γ_2 に対して，型環境 $\Gamma_1 + \Gamma_2$ を以下で定義する．

$$
\begin{aligned}
&\bullet + \bullet = \bullet \\
&(\Gamma'_1, i : \mathtt{qidx}) + (\Gamma'_2, i : \mathtt{qidx}) \\
&\qquad = (\Gamma'_1 + \Gamma'_2), i : \mathtt{qidx} \\
&(\Gamma'_1, \alpha : \mathtt{graph}^{(n)}) + (\Gamma'_2, \alpha : \mathtt{graph}^{(n)}) \\
&\qquad = (\Gamma'_1 + \Gamma'_2), \alpha : \mathtt{graph}^{(n)} \\
&(\Gamma'_1, K) + (\Gamma'_2, K) = (\Gamma'_1 + \Gamma'_2), K \\
&(\Gamma'_1, x : T) + (\Gamma'_2, x : T') \\
&\qquad = (\Gamma'_1 + \Gamma'_2), x : (T + T')
\end{aligned}
$$

$i : \mathtt{qidx}, x : Q(i)^1 + i : \mathtt{qidx}, x : Q(i)^1 = i : \mathtt{qidx}, x : Q(i)^\omega$ となるが，型 $Q(i)^\omega$ は well-formed ではないので，実際には型導出には出現しない．

定義 3.5 型環境 Γ に含まれるすべての連結性制約の和集合 K' が $K \subseteq K'$ を満たすとき，またそのときに限り $K \subseteq \Gamma$ と書く．

$\lambda Q^{\Pi\rightsquigarrow}$ の型システムの判断は以下の 4 つである．

1. $\vdash \Gamma$ は型環境 Γ の well-formedness で，直感的には，Γ で宣言される各種変数に重複がなく，各型や制約が先行する型環境のもとで well-formed であることを意味している．
2. $\Gamma \vdash K$ は型環境 Γ のもとでの連結性制約 K の well-formedness についての判断である．K に現れる $\mathtt{qidx}$ および α が Γ に存在していれば K は Γ のもとで well-formed である．
3. $\Gamma \vdash T$ は型 T が型環境 Γ のもとで well-formed であることを表す．
4. $\Gamma \vdash M : T$ は型付け判断であり，型環境 Γ の

もとで項 M に型 T が付くことを意味する．

型環境，連結性制約および型の well-formedness の規則を図 3, 4, 5 に示す．型環境と連結性制約についての各規則については先に述べたことを反映したものとなっている．量子ビット型 $Q(i)^{\kappa}$ は i が Γ で定義されており κ が ω ではない時に well formed である．ペア型と直和型の use の条件は，全体が複数回使える場合には，部分も複数回使えるものでなければならないことを要求している．

型付け規則を図 6 に示す．基本的に [15] をふまえた規則となっているので，新たに導入した型に関係する規則を中心に説明する．$\texttt{cnot}[i,j]$ の型付け規則 T-CNot では，連結性制約 $i \rightsquigarrow j$ を要求するようにすることで，CNOT ゲートを適用する量子ビットペアがこの向きの辺によって直接隣接していることを要求する．T-Conn および T-ConnApp はそれぞれ連結性制約の抽象および解決のための規則である．T-QAbs および T-QApp は $\texttt{qidx}$ 変数に関する抽象および適用である．T-GAbs および T-GApp は 連結性制約述語の抽象および具体化を表している．

3.3 操作的意味論

操作的意味論を定義する前に，まず，coupling graph を定義する．

定義 3.6 Coupling graph C を，順序付きの列 $L = q_1 \dots q_n$ および接続関係 E によって $C = (L, E)$ と定義する．各 q_k $(k \in \{1, \dots, n\})$ は 1 つの量子ビットを表している．また，E の要素は $(q_{k_1}, q_{k_2}) \in E$ の形をしており，これは q_{k_1}, q_{k_2} が辺で直接つながっていることを表す．

Coupling graph のもとで，1 ステップの評価関係を

$$C \vdash [\psi, M] \to_p [\psi', M']$$

と定義する．ここで $[\psi, M]$ が quantum closure [15] であり，ψ は量子状態を，M は項を表し，p はこのステップが確率 p で起こることを表す．さらに，ψ, M は $C = (L, E)$ の $L = q_1 \dots q_n$ と対応している．ψ は空間 $\mathbb{C}^{2 \otimes n}$ 上の正規化されたベクトルであり，$\psi = |q_1 \dots q_n\rangle = \sum_{x \in \{0,1\}^n} c_x |x\rangle$ と書く．また M に現れる自由変数はすべて量子ビットを表す変数であり，その全てが $q_1, \dots, q_n$ のいずれかであるとする．

Selinger らの体系における quantum closure との違いは，Selinger らの体系では coupling graph 中の L も quantum closure に含まれていたことである．これは，Selinger らの体系では $\texttt{new}$ の存在により評価の過程で新たな量子ビットが生成され，L が更新されるためである．提案体系 $\lambda Q^{\Pi \rightsquigarrow}$ では， $\texttt{new}$ は存在せず，すべての量子ビットが静的に確保された状態を考えているため，quantum closure から L を移し，予め定められた coupling graph C のもとでの評価であることを意識した形式化を行っている．

これらの定義のもとで，評価関係の導出規則 (の一部) を図 7 に示す．注目すべきは $\texttt{cnot}$ の適用に関する規則である．$\texttt{cnot}$ が量子ビットペアに適用される際には，それらのペアが coupling graph C で直接隣接していることを要求する．もし隣接していない量子ビットペアに対して $\texttt{cnot}$ が適用されれば行き詰まり状態となる．その他の規則については Selinger らの規則と同様である．

4 体系の性質

この節では，$\lambda Q^{\Pi \rightsquigarrow}$ が満たす性質として型安全性，すなわち，型付けされるプログラムは連結性制約違反を犯すことがないことが満たさせることを述べる．型安全性は，Progress と Preservation 定理によって示される [21] [13].

まず，準備として quantum closure についての型付けを定義する．

定義 4.1 Coupling graph $C = (L, E)$ に対して，ある型環境 Γ があって以下をすべて満たすとき，またそのときに限り $C \vdash [\psi, M] : T$ と書く．

- $L = q_1 \cdots q_n$ の時，$\psi \in \mathbb{C}^{2 \otimes n}$
- $q_k \in L$ iff $q_k : Q(i_k)^1 \in \Gamma$
- $\forall k_1, k_2 \in \{1, \dots, n\}$ に対して $k_1 \neq k_2$ ならば $i_{k_1} \neq i_{k_2}$
- 任意の $q_{k_1} : Q(i_{k_1})^1, q_{k_2} : Q(i_{k_2})^1 \in \Gamma$ に対して $i_{k_1} \rightsquigarrow i_{k_2} \in \Gamma$ iff $(q_{k_1}, q_{k_2}) \in C$
- $\Gamma \vdash M : T$

$\boxed{\vdash \Gamma}$ **Well-formed environments**

$$\frac{}{\vdash \bullet}\ \text{WF-Ctx-Empty} \qquad \frac{\vdash \Gamma \qquad i \notin dom(\Gamma)}{\vdash \Gamma, i : \texttt{qidx}}\ \text{WF-Ctx-Qidx} \qquad \frac{\vdash \Gamma \qquad \Gamma \vdash K}{\vdash \Gamma, K}\ \text{WF-Ctx-Conn}$$

$$\frac{\vdash \Gamma \qquad \alpha \notin dom(G)}{\vdash \Gamma, \alpha : \texttt{graph}^{(n)}}\ \text{WF-Ctx-GVar} \qquad \frac{\vdash \Gamma \qquad \Gamma \vdash T \qquad x \notin dom(\Gamma)}{\vdash \Gamma, x : T}\ \text{WF-Ctx-Var}$$

図 3　型環境の well-formedness

$\boxed{\Gamma \vdash K}$ **Well-formed connectivity**

$$\frac{}{\Gamma \vdash \bullet}\ \text{WF-Conn-Empty} \qquad \frac{i, j : \texttt{qidx} \in \Gamma}{\Gamma \vdash i \rightsquigarrow j}\ \text{WF-Conn-Edge}$$

$$\frac{i_1, \ldots, i_n : \texttt{qidx} \in \Gamma \qquad \alpha : \texttt{graph}^{(n)} \in \Gamma}{\Gamma \vdash \alpha(i_1, \ldots, i_n)}\ \text{WF-Conn-Graph}$$

図 4　連結性制約の well-formedness

$\boxed{\Gamma \vdash T}$ **Well-formed types**

$$\frac{\Gamma, i : \texttt{qidx} \vdash T \qquad \vdash \Gamma}{\Gamma \vdash \Pi i : \texttt{qidx}.T}\ \text{WF-Type-Pi-Qidx} \qquad \frac{\vdash \Gamma \qquad i : \texttt{qidx} \in \Gamma \qquad \kappa \leq 1}{\Gamma \vdash Q(i)^{\kappa}}\ \text{WF-Type-Qbit}$$

$$\frac{\vdash \Gamma}{\Gamma \vdash \texttt{unit}^{\kappa}}\ \text{WF-Type-Unit} \qquad \frac{\Gamma \vdash T_1 \qquad \Gamma \vdash T_2 \qquad \kappa \leq \min\{|T_1|, |T_2|\}}{\Gamma \vdash T_1 \oplus^{\kappa} T_2}\ \text{WF-Type-Sum}$$

$$\frac{\Gamma, K \vdash T}{\Gamma \vdash K \Rightarrow T}\ \text{WF-Type-Conn} \qquad \frac{\Gamma \vdash T_1 \qquad \Gamma \vdash T_2 \qquad \kappa \leq \min\{|T_1|, |T_2|\}}{\Gamma \vdash T_1 \otimes^{\kappa} T_2}\ \text{WF-Type-Pair}$$

図 5　型の well-formedness

定義 4.1 は，量子状態のベクトル長が L の長さと対応していること，`qidx` および連結性制約によって coupling graph C と対応した文脈 Γ のもとで項に型が付くことを表している．すなわち，$C \vdash M : T$ は M の型付け判断において連結性制約を満たすことを意味する．

これを使って，Progress は以下のように述べられる．この文言は M が CNOT ゲートを適用しようとしている時に行き詰まらないことを含んでいる．

定理 4.2 (Progress) Coupling graph C に対し，量子状態が ψ であるような項 M が $C \vdash [\psi, M] : T$ を満たすとする．このとき M は値であるか，あるいはある ψ', M', p が存在して $C \vdash [\psi, M] \to_p [\psi', M']$ となる．

Preseravtion を述べる前に各種代入補題を述べる．

補題 4.3 $\Gamma_1, x : S, \Gamma_2 \vdash M : T$ かつ $\Gamma_3 \vdash V : S$ かつ $\vdash \Gamma_1 + \Gamma_3$ ならば $(\Gamma_1 + \Gamma_3), \Gamma_2 \vdash [V/x]M : T$.

補題 4.4 $\Gamma, i : \texttt{qidx}, \Gamma' \vdash M : T$ かつ $j : \texttt{qidx} \in \Gamma$ ならば $\Gamma, [j/i]\Gamma' \vdash [j/i]M : [j/i]T$.

補題 4.5 $\Gamma, \alpha : \texttt{graph}^{(n)}, \Gamma' \vdash M : T$ かつ $\Gamma, i_1 : \texttt{qidx}, \ldots, i_n : \texttt{qidx} \vdash K$ ならば $\Gamma, [(i_1, \ldots, i_n)K/\alpha]\Gamma' \vdash [(i_1, \ldots, i_n)K/\alpha]M : [(i_1, \ldots, i_n)K/\alpha]T$.

定理 4.6 (Preservation) $C \vdash [\psi, M] : T$ かつ $C \vdash [\psi, M] \to_p [\psi', M']$ ならば $C \vdash [\psi', M'] : T$.

定理 4.2 および定理 4.6 より，与えられた coupling graph C に対応する型環境 Γ を用いて型がつくようなプログラムは，`cnot` の適用に際して連結性制約違反とならないことが保証される．

5　例

ここでは，これまで定義してきた体系で型がつく項をいくつか与える．説明を簡単にするため，適宜ラム

$\boxed{\Gamma \vdash M : T}$ **Typing**

$$\frac{}{\Gamma \vdash () : \mathtt{unit}^{\kappa}}\ \text{T-Unit} \qquad \frac{\vdash \Gamma \qquad i : \mathtt{qidx} \in \Gamma}{\Gamma \vdash \mathtt{meas}[i] : Q(i)^1 \to^{\omega} \mathtt{bit}}\ \text{T-Meas}$$

$$\frac{\vdash \Gamma \qquad i, j : \mathtt{qidx} \in \Gamma \qquad i \rightsquigarrow j \in \Gamma}{\Gamma \vdash \mathtt{cnot}[i, j] : Q(i)^1 \otimes^1 Q(j)^1 \to^{\omega} Q(i)^1 \otimes^1 Q(j)^1}\ \text{T-CNot}$$

$$\frac{\vdash \Gamma \qquad x : T' \in \Gamma \qquad 1 \leq |T| \qquad T \leq T'}{\Gamma \vdash x : T}\ \text{T-Var} \qquad \frac{\Gamma, x : T_1 \vdash M : T_2 \qquad \kappa\Gamma = \Gamma}{\Gamma \vdash \lambda x : T_1.M : T_1 \to^{\kappa} T_2}\ \text{T-Abs}$$

$$\frac{\Gamma_1 \vdash M : T_1 \to^{\kappa} T_2 \qquad \Gamma_2 \vdash N : T_1 \qquad \vdash \Gamma_1 + \Gamma_2 \qquad \kappa \geq 1}{\Gamma_1 + \Gamma_2 \vdash MN : T_2}\ \text{T-App}$$

$$\frac{\Gamma \vdash M : T_1 \qquad \Gamma \vdash T_1 \oplus^{\kappa} T_2}{\Gamma \vdash \mathtt{inl}\ M : T_1 \oplus^{\kappa} T_2}\ \text{T-SumL} \qquad \frac{\Gamma \vdash M : T_2 \qquad \Gamma \vdash T_1 \oplus^{\kappa} T_2}{\Gamma \vdash \mathtt{inr}\ M : T_1 \oplus^{\kappa} T_2}\ \text{T-SumR}$$

$$\frac{\Gamma_1 \vdash P : T_1 \oplus^{\kappa} T_2 \qquad \Gamma_2, x : T_1 \vdash M : T_3 \qquad \Gamma_2, y : T_2 \vdash N : T_3 \qquad \kappa \geq 1 \qquad \vdash \Gamma_1 + \Gamma_2}{\Gamma_1 + \Gamma_2 \vdash \mathtt{match}\ P\ \mathtt{with}\ (x \mapsto M \mid y \mapsto N) : T_3}\ \text{T-Match}$$

$$\frac{\Gamma_1 \vdash M : T_1 \qquad \Gamma_2 \vdash N : T_2 \qquad \kappa \leq \min\{|T_1|, |T_2|\} \qquad \vdash \Gamma_1 + \Gamma_2}{\Gamma_1 + \Gamma_2 \vdash \langle M, N \rangle : T_1 \otimes^{\kappa} T_2}\ \text{T-PairCons}$$

$$\frac{\Gamma_1 \vdash M : T_1 \otimes^{\kappa} T_2 \qquad \Gamma_2, x : T_1, y : T_2 \vdash N : T_3 \qquad \kappa \geq 1 \qquad \vdash \Gamma_1 + \Gamma_2}{\Gamma_1 + \Gamma_2 \vdash \mathtt{let}\ \langle x, y \rangle = M\ \mathtt{in}\ N : T_3}\ \text{T-PairDest}$$

$$\frac{\omega \cdot \Gamma_1, f : T_1 \to^{\omega} T_2 \vdash \lambda x : T_1.M : T_1 \to^{\omega} T_2 \qquad \Gamma_2, f : T_1 \to^{\omega} T_2 \vdash N : T_3 \qquad \vdash \omega \cdot \Gamma_1 + \Gamma_2}{\omega \cdot \Gamma_1 + \Gamma_2 \vdash \mathtt{let\ rec}\ f = \lambda x : T_1.M\ \mathtt{in}\ N : T_3}\ \text{T-Rec}$$

$$\frac{\Gamma, K \vdash M : T}{\Gamma \vdash \lambda\_ : K.M : K \Rightarrow T}\ \text{T-Conn} \qquad \frac{\Gamma \vdash M : K \Rightarrow T \qquad K \subseteq \Gamma}{\Gamma \vdash M\_ : T}\ \text{T-ConnApp}$$

$$\frac{\Gamma, i : \mathtt{qidx} \vdash M : T}{\Gamma \vdash \lambda i : \mathtt{qidx}.M : \Pi i : \mathtt{qidx}.T}\ \text{T-QAbs} \qquad \frac{\Gamma \vdash M : \Pi i : \mathtt{qidx}.T \qquad j : \mathtt{qidx} \in \Gamma}{\Gamma \vdash M\ j : [j/i]T}\ \text{T-QApp}$$

$$\frac{\Gamma, \alpha : \mathtt{graph}^{(n)} \vdash M : T}{\Gamma \vdash \lambda\alpha : \mathtt{graph}^{(n)}.M : \Pi\alpha : \mathtt{graph}^{(n)}.T}\ \text{T-GAbs}$$

$$\frac{\Gamma \vdash M : \Pi\alpha : \mathtt{graph}^{(n)}.T \qquad \Gamma, i_1 : \mathtt{qidx}, \ldots, i_n : \mathtt{qidx} \vdash K}{\Gamma \vdash M\ (i_1, \ldots, i_n)K : [(i_1, \ldots, i_n)K/\alpha]T}\ \text{T-GApp}$$

図 6　型付け規則

ダ抽象の引数の型を省略し，`let` 式を用いる．

例 5.1

$\mathtt{let}\ f = \lambda j_1, j_2.\lambda\alpha.\lambda\_.\lambda g.\lambda x_1, x_2.\ g\ j_1\ j_2\ \_\ \langle x_1, x_2 \rangle\ \mathtt{in}$
$\mathtt{let}\ \langle q_1, q_2 \rangle = f\ i_1\ i_2\ ((j_1, j_2)\bullet)\ \_\ (\lambda x.x)\ q_1\ q_2\ \mathtt{in}$
$f\ i_1\ i_2\ ((j_1, j_2)j_1 \rightsquigarrow j_2)\ \_\ (\lambda\_.\mathtt{cnot})\ q_1\ q_2$

関数 f は，直感的には量子ビットペアを受け取って量子ビットペアを返す関数 g と，2 つの量子ビット x, y を受け取り，g を $\langle x, y \rangle$ に適用する関数である．この例では，f の型は次のようになる．

$$\begin{aligned}&\Pi j_1, j_2 : \mathtt{qidx}.\Pi\alpha : \mathtt{graph}^{(2)}.\alpha(j_1, j_2) \Rightarrow \\ &\quad (\Pi j_3, j_4 : \mathtt{qidx}.\alpha(j_3, j_4) \Rightarrow \\ &\qquad Q(j_3)^1 \otimes^1 Q(j_4)^1 \to^{\omega} Q(j_3)^1 \otimes^1 Q(j_4)^1) \\ &\quad \to^{\omega} Q(j_1)^1 \otimes^1 Q(j_2)^1 \to^{\omega} Q(j_1)^1 \otimes^1 Q(j_2)^1\end{aligned}$$

g の制約が α によって抽象化されていることが重要である．なぜならば，例にもある通り，g には様々な制約を持つ関数が与えられうるからである．今回の例では，f の 1 回目の適用時には g に恒等関数が与えられているため，$\alpha(i_1, i_2)$ は $\bullet$ として振る舞う．それに対し，2 回目の適用時には `cnot` が与えられているため，$\alpha(i_1, i_2)$ は $i_1 \rightsquigarrow i_2$ という制約として振る舞う．

例 5.2 以下は，連結性制約 α が有用である 2 つ目の

$$C \vdash [\psi, (\lambda x : T.M)V] \to_1 [\psi, [V/x]M]$$
$$C \vdash [\psi, (\lambda i : \texttt{qidx}.M)\ j] \to_1 [\psi, [j/i]M]$$
$$C \vdash [\psi, (\lambda \alpha : \texttt{graph}^{(n)}.M)\ (i_1, \ldots, i_n)K] \to_1 [\psi, [(i_1, \ldots, i_n)K/\alpha]M]$$
$$C \vdash [\psi, (\lambda \_ : K.M)\_] \to_1 [\psi, M]$$

$$\frac{(q_i, q_j) \in C}{C \vdash [\psi, \texttt{cnot}[\_, \_]\,\langle q_i, q_j\rangle] \to_1 [\psi', \langle q_i, q_j\rangle]}$$

$$\frac{C \vdash [\psi, M] \to_p [\psi', M']}{C \vdash [\psi, M\ \_] \to_p [\psi', M'\ \_]} \qquad \frac{C \vdash [\psi, M] \to_p [\psi', M']}{C \vdash [\psi, M\ i] \to_p [\psi', M'\ i]}$$

$$\frac{C \vdash [\psi, M] \to_p [\psi', M']}{C \vdash [\psi, M\ (i_1, \ldots, i_n)K] \to_p [\psi', M'\ (i_1, \ldots, i_n)K]}$$

図 7　評価規則 (一部抜粋)

例である．

$$\begin{aligned}
&\texttt{let}\ f = \lambda j_1, j_2, j_3, j_4.\lambda\alpha.\lambda_.\lambda g.\lambda x_1, x_2, x_3, x_4.\\
&\qquad \texttt{let}\,\langle x_1, x_2\rangle = g\ j_1\ j_2\ _\,\langle x_1, x_2\rangle\ \texttt{in}\\
&\qquad \texttt{let}\,\langle x_3, x_4\rangle = g\ j_3\ j_4\ _\,\langle x_3, x_4\rangle\ \texttt{in}\\
&\qquad \langle x_1, x_2, x_3, x_4\rangle\\
&\texttt{in}\\
&f\ i_1\ i_2\ i_3\ i_4\ ((j_1, j_2)j_1 \rightsquigarrow j_2)\ _\ (\lambda_.\texttt{cnot})\ q_1\ q_2\ q_3\ q_4
\end{aligned}$$

関数 f は量子ビットペアを受け取って量子ビットペアを返す関数 g と，4 つの量子ビット x_1, x_2, x_3, x_4 をうけとり，g を $\langle x_1, x_2\rangle, \langle x_3, x_4\rangle$ に適用する関数である．この例では，f の型は次のようになる．

$$\begin{aligned}
&\Pi j_1, j_2, j_3, j_4 : \texttt{qidx}.\Pi\alpha : \texttt{graph}^{(2)}.\alpha(j_1, j_2), \alpha(j_3, j_4)\\
&\quad \Rightarrow (\Pi j_5, j_6 : \texttt{qidx}.\alpha(j_5, j_6) \Rightarrow\\
&\qquad Q(j_5)^1 \otimes^1 Q(j_6)^1 \to^\omega Q(j_5)^1 \otimes^1 Q(j_6)^1)\\
&\quad \to^\omega Q(j_1)^1 \to^1 Q(j_2)^1 \to^1 Q(j_3) \to^1 Q(j_4)\\
&\quad \to^1 Q(j_1)^1 \otimes^1 Q(j_2)^1 \otimes^1 Q(j_3)^1 \otimes^1 Q(j_4)^1
\end{aligned}$$

引数の g は f の内部で異なる量子ビットペアに対して適用されるため，g の制約はそのどちらについても必要になるが，α によって g の制約を抽象化しているため，型に現れているように $\alpha(j_1, j_2), \alpha(j_3, j_4)$ といった記述が可能である．

6　関連研究

Hietala らは，量子ビットを誤り度によって区別する型システム [7] を提案し，QWIRE [12] という量子回路記述言語を用いて実装している．誤り度は依存型によって表現され，誤り度 n を持つ量子ビット型は $\texttt{Qubit}\ n$ と表現される．このような量子ビット型を用いることで，量子ゲートの適用によって増加していく誤り度を追跡することができる．例えば量子ゲート U が 1 量子ビットを受け取って誤り度を k 増加させるような量子ゲートであるとすると，U の型は $\Pi n : \texttt{Nat}.\texttt{Qubit}\ n \to \texttt{Qubit}\ (n + k)$ のように表現することができる．また，誤り度が t 以下である量子ビットに対して誤り訂正を行うゲート EC の型は，$\Pi n : \texttt{Nat}.n \leq t \Rightarrow \texttt{Qubit}\ n \to \texttt{Qubit}\ 0$ のように表現することができる．型付け導出を得る過程で誤り度の最大値を求めることができるため，量子誤り訂正によって訂正できる許容範囲内であるかどうかを型システムで検証することが可能である．このように，誤り度によって量子ビットを類別するという考え方は，我々の提案体系 $\lambda Q^{\Pi\rightsquigarrow}$ と類似したものである．

Coupling graph 上の物理量子ビットを量子プログラム上の論理量子ビットへどう割り当てるかという問題は量子ビット割り当て問題とよばれ，連結性制約を満たすようにプログラム変換が行われる．Siraichi ら [17][16] は量子ビット割り当て問題を部分グラフ同型性判定問題と Token Swapping という問題をもって定式化した．また，これらの問題に対する厳密な解法とヒューリスティック解法のそれぞれを示し，速度計測などの評価を行っている．我々の提案体系 $\lambda Q^{\Pi\rightsquigarrow}$ では，量子ビット割り当てがすでに行われた状態で型が付くかの判断を行っているが，今後型システムと割

り付けアルゴリズムの協調を考えるさいにこれらの文献が参考になる．

Ohori [11] は，ある種の部分構造論理に対応する型付計算体系を考え，(直観主義論理に対応する型システムによる) 型導出をこの体系の導出への変換する問題が，古典コンピュータにおけるレジスタ割り当て問題に対応することを示した．低レベルなアーキテクチャ制約を型システムで表現する点は本研究と近い．我々の体系も上述の量子ビット割り当て問題のターゲット言語として使えると考えられる．

Fu らは Quipper [5] を発展させ，依存型と線形型を両方備えた Proto-Quipper-D [4][3] を提案している．また線形型と依存型の統合について操作的意味論，表示的意味論を与えるとともに，その圏論的構造についても考察がなされている．ただし，Proto-Quipper は量子ビット自体を依存型で区別する体系ではなく，例えば QFT 回路といった入力の量子ビット数が可変である回路に与える量子ビットの数の不整合を防いだり，あるいは存在型によって可逆計算の文脈における garbage bit をより簡潔な記法で扱えるようにし，uncomputation の機構を簡潔に実現するといった目的で依存型が用いられている．

7 結論

7.1 本研究のまとめ

本研究では，Selinger らの量子ラムダ計算に依存型を導入することによって，量子プログラムが連結性制約を満たしていることを型システムで保証するための体系 $\lambda Q^{\Pi\rightsquigarrow}$ を提案した．そして，型が付いたプログラムが実際に連結性制約を違反せずに実行されるという型安全性を証明した．

7.2 今後の課題

Selinger らの量子ラムダ計算に存在した `new` についての扱いは今後の課題の 1 つである．`new` が関数内部で呼ばれうることを考えると，連結性制約を考慮したまま動的に生成される量子ビットを追跡することは難しい．さらに，量子ビットが生成されるタイミングで coupling graph 上のノードと対応付ける必要があるが，これは量子ビット割り当て問題をはらんでいるため，`new` を取り入れる際の課題の一つである．よって，そもそも `new` を取り入れるのか，取り入れる場合にどのような形式化を行い，型システムによってどこまで保証するのかを考える必要がある．

型推論アルゴリズムを設計することも今後の課題である．型推論アルゴリズムによって，与えられたプログラムを実行する際に必要な制約を抽出することができ，量子ビット割当問題やプログラム変換のヒントとして用いることができるようになる．また，実装面では，本体系の `qidx` の抽象や連結性制約の抽象，およびそれらの解決を暗黙的にできるような実装を考えている．例えば，第 5 節の例 5.2 で触れた以下のプログラムを考える．

$$
\begin{array}{l}
\texttt{let } f = \lambda j_1, j_2, j_3, j_4.\lambda\alpha.\lambda\_.\lambda g.\lambda x_1, x_2, x_3, x_4. \\
\qquad \texttt{let } \langle x_1, x_2\rangle = g\ j_1\ j_2\ \_\ \langle x_1, x_2\rangle \texttt{ in} \\
\qquad \texttt{let } \langle x_3, x_4\rangle = g\ j_3\ j_4\ \_\ \langle x_3, x_4\rangle \texttt{ in} \\
\qquad \langle x_1, x_2, x_3, x_4\rangle \\
\texttt{in} \\
f\ i_1\ i_2\ i_3\ i_4\ ((j_1, j_2)j_1 \rightsquigarrow j_2)\ \_\ (\lambda\_.\texttt{cnot})\ q_1\ q_2\ q_3\ q_4
\end{array}
$$

もし `qidx` や連結性制約の抽象と適用が省略できれば，以下のように書け，可読性が向上する．

$$
\begin{array}{l}
\texttt{let } f = \lambda g.\lambda x_1, x_2, x_3, x_4. \\
\qquad \texttt{let } \langle x_1, x_2\rangle = g\ \langle x_1, x_2\rangle \texttt{ in} \\
\qquad \texttt{let } \langle x_3, x_4\rangle = g\ \langle x_3, x_4\rangle \texttt{ in} \\
\qquad \langle x_1, x_2, x_3, x_4\rangle \\
\texttt{in} \\
f\ \texttt{cnot}\ q_1\ q_2\ q_3\ q_4
\end{array}
$$

連結性制約の抽象と適用は，それぞれ関数の抽象と適用のタイミングでまとめて行うことにすれば容易に実現できる．しかし，連結性制約述語 α に関しては，束縛変数の数や順番などを考慮する必要があり，連結性制約ほど単純ではないため，今後の課題としたい．

謝辞

多くの助言を頂いた末永幸平氏に深く感謝いたします．

参 考 文 献

[1] Altenkirch, T. and Grattage, J.: A functional quantum programming language, *Proceedings of the 20th Annual IEEE Symposium on Logic in Computer Science (LICS' 05)*, June 2005, pp. 249–258.

[2] Devitt, S. J.: Performing Quantum Computing Experiments in the Cloud, (2016).

[3] Fu, P., Kishida, K., Ross, N. J., and Selinger, P.: A Tutorial Introduction to Quantum Circuit Programming in Dependently Typed Proto-Quipper, *Proceedings of Reversible Computation - 12th International Conference, RC 2020, Oslo, Norway, July 9-10, 2020, Proceedings*, Lanese, I. and Rawski, M.(eds.), Lecture Notes in Computer Science, Vol. 12227, Springer, 2020, pp. 153–168.

[4] Fu, P., Kishida, K., and Selinger, P.: Linear Dependent Type Theory for Quantum Programming Languages: Extended Abstract, *Proceedings of LICS '20: 35th Annual ACM/IEEE Symposium on Logic in Computer Science, Saarbrücken, Germany, July 8-11, 2020*, Hermanns, H., Zhang, L., Kobayashi, N., and Miller, D.(eds.), ACM, 2020, pp. 440–453.

[5] Green, A. S., Lumsdaine, P. L., Ross, N. J., Selinger, P., and Valiron, B.: Quipper: a scalable quantum programming language, *Proceedings of ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI '13, Seattle, WA, USA, June 16-19, 2013*, Boehm, H. and Flanagan, C.(eds.), ACM, 2013, pp. 333–342.

[6] Hayashi, S.: Singleton, Union, and Intersection Types for Program Extraction, *Information and Computation*, Vol. 109, No. 1(1994), pp. 174 – 210.

[7] Hietala, K., Rand, R. S., and Hicks, M.: Tracking Errors through Types in Quantum Programs, Presented at Workshop on Programming Languages for Quantum Computing, January 2020.

[8] Igarashi, A. and Kobayashi, N.: Type-based analysis of communication for concurrent programming languages, *Proceedings of Static Analysis Symposium (SAS'97)*, Van Hentenryck, P.(ed.), Berlin, Heidelberg, Springer Berlin Heidelberg, 1997, pp. 187–201.

[9] Jones, M. P.: A Theory of Qualified Types, *Sci. Comput. Program.*, Vol. 22, No. 3(1994), pp. 231–256.

[10] Mogensen, T. Æ.: Types for 0, 1 or many uses, *Proceedings of Implementation of Functional Languages (IFL'98)*, Clack, C., Hammond, K., and Davie, T.(eds.), Berlin, Heidelberg, Springer Berlin Heidelberg, 1998, pp. 112–122.

[11] Ohori, A.: Register Allocation by Proof Transformation, *Proceedings of European Symposium on Programming (ESOP2003)*, Degano, P.(ed.), Berlin, Heidelberg, Springer Berlin Heidelberg, 2003, pp. 399–413.

[12] Paykin, J., Rand, R., and Zdancewic, S.: QWIRE: a core language for quantum circuits, *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017, Paris, France, January 18-20, 2017*, Castagna, G. and Gordon, A. D.(eds.), ACM, 2017, pp. 846–858.

[13] Pierce, B. C.: *Types and Programming Languages*, MIT Press, 2002.

[14] Selinger, P. and Valiron, B.: A lambda calculus for quantum computation with classical control, *Mathematical Structures in Computer Science*, Vol. 16, No. 3(2006), pp. 527–552.

[15] Selinger, P. and Valiron, B.: Quantum Lambda Calculus, *Semantic Techniques in Quantum Computation*, Gay, S. and Mackie, I.(eds.), Cambridge University Press, 2009, pp. 135–172.

[16] Siraichi, M. Y., Santos, V. F. d., Collange, C., and Pereira, F. M. Q. a.: Qubit Allocation As a Combination of Subgraph Isomorphism and Token Swapping, *Proc. ACM Program. Lang.*, Vol. 3, No. OOPSLA(2019), pp. 120:1–120:29.

[17] Siraichi, M. Y., Santos, V. F. d., Collange, S., and Pereira, F. M. Q.: Qubit Allocation, *Proceedings of the 2018 International Symposium on Code Generation and Optimization*, CGO 2018, New York, NY, USA, Association for Computing Machinery, 2018, pp. 113–125.

[18] Turner, D. N., Wadler, P., and Mossin, C.: Once Upon a Type, *Proceedings of the Seventh International Conference on Functional Programming Languages and Computer Architecture*, FPCA '95, New York, NY, USA, ACM, 1995, pp. 1–11.

[19] van Tonder, A.: A lambda calculus for quantum computation, *SIAM Journal on Computing*, Vol. 33, No. 5(2004), pp. 1109–1135.

[20] Wootters, W. and Zurek, W.: A single quantum cannot be cloned, *Nature*, Vol. 299(1982), pp. 802–803.

[21] Wright, A. K. and Felleisen, M.: A Syntactic Approach to Type Soundness, *Information and Computation*, Vol. 115, No. 1(1994), pp. 38–94.