Lecture 15: Quantum λ-calculus

Summary.
(1) Syntax and operational semantics. Types.

(2) Examples: representation of quantum programs.

(3) Towards a Curry-Howard-Lambek corespondence for quantum computation.

Luís Soares Barbosa,
Univ. Minho *(Informatics Dep.)* & Inl *(Quantum Software Engineering Group)*

Motivation.

The development of a quantum λ-calculus is primarily associated to the quest for a (high-order) functional programming language for quantum computation. Actually there are several proposals for such a calculus, corresponding to different ways in which quantum computation is conceptualized. In this lecture our focus is on the *quantum data, classic control* paradigm as in, for example, the well-known QRAM model, where a classical machine controls a quantum device through a program specifying a number of unitary transformations and measurements over quantum data. Such a program is represented by a λ-term in which quantum data is somehow embedded.

Note, however, that there are a number of different proposals for a λ-calculus for quantum computation — see, for examples references [5, 2] for two approaches to expressing pure quantum computations without measurement, or [1] for a calculus in which quantum data is represented by density matrices, or M. Ying proposal for the incorporation of quantum control [6].

Typically, programs in a quantum λ-calculus operate over both classical and quantum data, and this fundamental distinction has to be lifted to the calculus. The basic issue to keep in mind is that quantum data cannot be reused (duplicated), as stated by the famous *non-cloning* principle. In a quantum program qubits must be uniquely referenced, i.e. no two variable occurrences may refer to the same qubit. This is expressed syntactically by a *linearity*: a term $\lambda \mathtt{x}.\mathtt{t}$ is linear if the variable $\mathtt{x}$ is used at most once along the evaluation of $\mathtt{t}$. This is a main concern in the calculus below and, in particular, in the construction of its typing system.

Syntax.

$$M, N, P \ni x \mid c \mid M\,N \mid \lambda x.M \mid \langle M, N\rangle \mid \mathsf{let}\,\langle x, y\rangle = M\,\mathsf{in}\,N \mid \mathsf{if}\,M\,\mathsf{then}\,N\,\mathsf{else}\,P$$

where $x \in X$, for X an infinite set of variables, and c ranges over the following constants,

$$c \ni * \mid 0 \mid 1 \mid \mathit{new} \mid \mathit{ms} \mid U$$

where new stands for a function for state preparation (accepts a classical bit b, returns qubit $|b\rangle$), ms for a function performing a measurement (in the canonical basis), and U for the application of an unitary transformation. Common abbreviations include

$$\begin{aligned} \mathsf{let}\,x = M\,\mathsf{in}\,P &\stackrel{\text{abv}}{=} (\lambda x.P)\,M \\ \lambda\langle x, y\rangle.P &\stackrel{\text{abv}}{=} \lambda z.(\mathsf{let}\,\langle x, y\rangle = z\,\mathsf{in}\,P) \end{aligned}$$

The notions of α-equivalence, free variable and substitution are defined as usual. Terms encode quantum algorithms, e.g.

Example [fair coin].

$$\mathit{coin} \;=\; \lambda *.\mathit{ms}(H(\mathit{new}\,0))$$

At first sight, it seemed reasonable to include a term to directly represent a qubit, e.g. $|\phi\rangle$, as in a function $\lambda x.|\phi\rangle$ which constantly outputs $|\phi\rangle$. The problem comes from entanglement: given two qubits entangled (and therefore not representable in the form $|\phi\rangle \otimes |\phi'\rangle$) there are no ways to represent in a term the variables corresponding to the first and second qubits in the entangled pair.

Operational semantics.

The operational semantics is given in terms of a reduction machine, which somehow represents a quantum processor acting over a quantum memory. The problem mentioned above requires some form of indirect representation of the quantum state of the underlying a program. This entails the notion of a quantum closure:

$$[Q, L, M]$$

where Q is a normalized vector in $\otimes^n \mathbb{C}^2$, M is a λ-term, and L is an ordered list $|x_1 \cdots x_n\rangle$ of term variables meaning that variable x_i is bound in term M to the qubit i.

Example.

$$[\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle), |p, q\rangle, \lambda x.x\,p\,q]$$

where p and q represent, respectively, the two qubits in the entangled state $|p,q\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$.

Given the probabilistic nature of measurement, the reduction machine is probabilistic:

$$(S, V, R, pr)$$

where S is a set of states, $V \subseteq S$ is the subset of value states (in which reduction terminates), $R \subseteq S - V \times S$ is a set of reductions, and $pr : R \longrightarrow [0,1]$ is a probability function, such that the number of states related by R with each state is finite and

$$\sum_{y \in \{y \mid (x,y) \in R\}} pr(x,y) \leq 1$$

Notation $x \rightarrow_\rho y$ stands for $pr(x,y) = \rho$, which extends, as expected, to n-step reductions: $x \rightarrow_\rho^n y \stackrel{abv}{=} (pr^n(x,y) = \rho)$, where

$$pr^n(x,y) \;=\; \sum_{z \in \{z \mid (x,z) \in R\}} pr(x,z)\, pr^{n-1}(z,y)$$

The basic relation is *reachability with non-zero probability* ($x \rightarrow_{>0}^n y$ for some $n \geq 0$).

- total V-probability: $pr_V(x) = \sum_{n=0}^{\infty} \sum_{v \in V} pr^n(x,v)$
- divergence-probability: $pr_\infty(x) = lim_{n \to \infty} \sum_{x \in S} pr^n(x,y)$
- error-probability: $pr_{err}(x) = 1 - pr_V(x) - pr_\infty(x)$

In some situations it is useful to relax reachability to include null probability ($x \rightsquigarrow y$) because a null probability of getting to a certain state is not an absolute warranty of its impossibility, due to decoherence and imprecision of physical operations. Thus, a state $x \in S$ is *consistent* if there is no error state e such that $x \rightsquigarrow e$, where e is an *error state* if $e \notin V$ and $\sum_{y \in S} pr(e,y) < 1$.

Exercise 1

Show that $pr_{err}(x) = 0$ if x is consistent. Does the converse hold?

Operational semantics of the quantum λ-calculus

The reduction machine for the quantum λ-calculus is probabilistic and adopts a *call-by-value* reduction strategy. Its purpose is to evaluate a quantum closure until a value state is reached. A value state is a quantum closure whose term is a value, defined by

$$V, V' \;\ni\; x \mid \lambda x.M \mid \langle V, V' \rangle \mid * \mid 0 \mid 1 \mid new \mid ms \mid U$$

Classical control:

$$[Q, L, (\lambda x.M)P] \longrightarrow_1 [Q, L, M[x := P]]$$

$$[Q, L, \mathtt{let}\ \langle x, y\rangle = \langle V, V'\rangle\ \mathtt{in}\ N] \longrightarrow_1 [Q, L, N[x := V, y := V']]$$

$$[Q, L, \mathtt{if}\ 0\ \mathtt{then}\ N\ \mathtt{else}\ P] \longrightarrow_1 [Q, L, P]$$

$$[Q, L, \mathtt{if}\ 1\ \mathtt{then}\ N\ \mathtt{else}\ P] \longrightarrow_1 [Q, L, N]$$

Quantum data:

$$[Q, |x_1, \cdots, x_n\rangle, \mathit{new}\ 0] \longrightarrow_1 [Q \otimes |0\rangle, |x_1, \cdots, x_n, x_{n+1}\rangle, x_{n+1}]$$

$$[Q, |x_1, \cdots, x_n\rangle, \mathit{new}\ 1] \longrightarrow_1 [Q \otimes |1\rangle, |x_1, \cdots, x_n, x_{n+1}\rangle, x_{n+1}]$$

$$[Q, L, U\langle x_1, \cdots, x_n\rangle] \longrightarrow_1 [Q', L, \langle x_1, \cdots, x_n\rangle]$$

$$[\alpha|Q_0\rangle + \beta|Q_1\rangle, L, \mathit{ms}\ x_i] \longrightarrow_{|\alpha|^2} [|Q_0\rangle, L, 0]$$

$$[\alpha|Q_0\rangle + \beta|Q_1\rangle, L, \mathit{ms}\ x_i] \longrightarrow_{|\beta|^2} [|Q_1\rangle, L, 1]$$

In the rule dealing with $U\langle x_1, \cdots, x_n\rangle$, Q' is the state produced by applying U to qubits indexed by variables x_1 to x_n. In the rule for measurements, $|Q_0\rangle = \sum_j \alpha_j|\phi_j\rangle \otimes |0\rangle \otimes |\psi_j\rangle$ where $|\phi_j\rangle$ is a i-qubit state, so that the measured qubit is the one pointed to by x_i, and similarly for $|Q_1\rangle$.

Congruence rules:

$$\frac{[Q, L, N] \longrightarrow_\rho [Q', L', N']}{[Q, L, MN] \longrightarrow_\rho [Q', L, MN']} \qquad \frac{[Q, L, M] \longrightarrow_\rho [Q', L', M']}{[Q, L, MV] \longrightarrow_\rho [Q', L', M'V]}$$

$$\frac{[Q, L, N] \longrightarrow_\rho [Q', L', N']}{[Q, L, \langle M, N\rangle] \longrightarrow_\rho [Q', L', \langle M, N'\rangle]} \qquad \frac{[Q, L, M] \longrightarrow_\rho [Q', L', M']}{[Q, L, \langle M, V\rangle] \longrightarrow_\rho [Q', L', \langle M', V\rangle]}$$

$$\frac{[Q, L, M] \longrightarrow_\rho [Q', L', M']}{[Q, L, \mathtt{if}\ M\ \mathtt{then}\ N\ \mathtt{else}\ P] \longrightarrow_\rho [Q', L', \mathtt{if}\ M'\ \mathtt{then}\ N\ \mathtt{else}\ P]}$$

$$\frac{[Q, L, M] \longrightarrow_\rho [Q', L', M']}{[Q, L, \mathtt{let}\ \langle x, y\rangle = M\ \mathtt{in}\ N] \longrightarrow_\rho [Q', L', \mathtt{let}\ \langle x, y\rangle = M'\ \mathtt{in}\ N]}$$

Call-by-value or call-by-name?

The rules above define a λ-calculus with a call-by-values evaluation strategy. As measurement is probabilistic, the evaluation strategy adopted affects not only the efficiency of execution, but also the results themselves.

Consider $M = (\lambda x.\mathtt{badd}\, xx)(ms(H(new\, 0)))$, whrer $\mathtt{badd}$ is Boolean addition.

$$[\,|\,\rangle, (\lambda x.\mathtt{badd}\, xx)(ms(H(new\, 0)))]$$

$$\longrightarrow_1 \quad [|0\rangle, (\lambda x.\mathtt{badd}\, xx)(ms\,(H\, p0))]$$

$$\longrightarrow_1 \quad [\tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle), (\lambda x.\mathtt{badd}\, xx)(ms\, p0))]$$

$$\begin{cases} \longrightarrow_{0.5} \quad [|0\rangle, (\lambda x.\mathtt{badd}\, xx)(0)] \\ \longrightarrow_{0.5} \quad [|1\rangle, (\lambda x.\mathtt{badd}\, xx)(1)] \\ \longrightarrow_1 \quad [|0\rangle, \mathtt{badd}\, 00] \\ \longrightarrow_1 \quad [|1\rangle, \mathtt{badd}\, 11] \\ \longrightarrow_1 \quad [|0\rangle, 0] \\ \longrightarrow_1 \quad [|1\rangle, 1] \end{cases}$$

thus, returning 0 with probability 1. If a call-by-name evaluation strategy was used the result would be

$$[\,|\,\rangle, (\lambda x.\mathtt{badd}\, xx)(ms(H(new\, 0)))]$$

$$\longrightarrow_1 \quad [\,|\,\rangle, \mathtt{badd}\,(ms(H(new\, 0)))(ms(H(new\, 0)))]$$

$$\begin{cases} \longrightarrow_{0.25} \quad [|01\rangle, 1] \\ \longrightarrow_{0.25} \quad [|10\rangle, 1] \\ \longrightarrow_{0.25} \quad [|00\rangle, 0] \\ \longrightarrow_{0.25} \quad [|11\rangle, 0] \end{cases}$$

yielding 0 or 1 with the same probability.

Types.

The reduction machine can produce *error-states* — e.g. $[Q, L, H(\lambda x.x)]$ or $[Q, |x, y, z\rangle, U\langle x, x\rangle]$ — which correspond to run-time errors. The purpose of a type system is precisely to get rid of such states, capturing correctly the notion of *duplication*. A similar concern will be crucial in defining linear logic below.

$$A, B \ni \mathtt{bit} \mid \mathtt{qubit} \mid !A \mid A \otimes B \mid A \multimap B \mid \top$$

where $A \otimes B$ types pairs of elements of type A and B, $A \multimap B$ is the type of functions from A to B, $\top$ is the type of constant $*$, and $!A$ is the type of *duplicable* elements of type A. Any value of type $!A$ can be used in a context in which a value of type A is expected (i.e. used only once, even if it is a duplicable value), leading to the following *subtyping* relation $\precsim$, defined under the overall condition $n = 0 \Rightarrow m = 0$:

$$\frac{}{!^n\mathbf{bit} \precsim !^m\mathbf{bit}}\,(\mathbf{bit}) \qquad \frac{}{!^n\mathbf{qubit} \precsim !^m\mathbf{qubit}}\,(\mathbf{qubit}) \qquad \frac{}{!^n\top; \precsim !^m\top}\,(\top)$$

$$\frac{A_1 \precsim B_1 \quad A_2 \precsim B_2}{!^n(A_1 \otimes A_2) \precsim !^m(B_1 \otimes B_2)}\,(\otimes) \qquad \frac{A \precsim A' \quad B \precsim B'}{!^n(A' \multimap B) \precsim !^m(A \otimes B')}\,(\multimap)$$

Exercise 2

Let QT denote the set of types for quantum λ-calculus. Show that $(QT, \precsim)$ is a preorder and that the quotient of QT by $\precsim$-symmetric closure forms a poset under $\precsim$.

Terms in the calculus are typed through *typing judgements* — $\Delta \rhd M : A$, where Δ is a set of typed variables $\{x_1 : A_1, \cdots, x_n : A_n\}$[1]. Each constant c has an associated type A_c as follows:

$$A_0, A_1 = \mathbf{bit} \quad A_{new} = \mathbf{bit} \multimap \mathbf{qubit} \quad A_U = \mathbf{qubit}^{\otimes n} \multimap \mathbf{qubit}^{\otimes n} \quad A_{ms} = \mathbf{qubit} \multimap !\mathbf{bit}$$

Exercise 3

Rule (ax_2) establishes type $!A_c$ as the *most generic* type for c. Use this fact to show that no qubit created through new can have the type $!\mathbf{qubit}$.

---

[1]Whenever several contexts Δ_1, Δ_2, ... Δ_n, appear in a typing judgement they are assumed to be disjoint.

Typing rules

$$\frac{A \precsim B}{\Delta, x : A \ \rhd\ x : B}\,(ax_1) \qquad \frac{!A_c \precsim B}{\Delta \ \rhd\ c : B}\,(ax_2) \qquad \frac{}{\Delta \ \rhd\ * : !^n\top}\,(\top)$$

$$\frac{x : A, \Delta \ \rhd\ M : B}{\Delta \ \rhd\ \lambda x.M : A \multimap B}\,(\lambda_1) \qquad \frac{\Gamma, !\Delta, x : A \ \rhd\ M : B}{\Gamma, !\Delta \ \rhd\ \lambda x.M : !^{n+1}(A \multimap B)}\,(\lambda_2),\ \text{if}\, \mathcal{FV}(M) \cap |\Gamma| = \emptyset$$

$$\frac{\Gamma_1, !\Delta \ \rhd\ M : A \multimap B \qquad \Gamma_2, !\Delta \ \rhd\ N : A}{\Gamma_1, \Gamma_2, !\Delta \ \rhd\ MN : B}\,(app)$$

$$\frac{\Gamma_1, !\Delta \ \rhd\ M : bit \multimap B \qquad \Gamma_2, !\Delta \ \rhd\ N : A \qquad \Gamma_2, !\Delta \ \rhd\ P : A}{\Gamma_1, \Gamma_2, !\Delta \ \rhd\ \text{if}\, M\, \text{then}\, N\, \text{else}\, P : A}\,(cond)$$

$$\frac{!\Delta, \Gamma_1 \ \rhd\ M_1 : !^n A_1 \qquad !\Delta, \Gamma_2 \ \rhd\ M_2 : !^n A_2}{!\Delta, \Gamma_1, \Gamma_2 \ \rhd\ \langle M_1, M_2\rangle : !^n(A_1 \otimes A_2)}\,(\otimes_{in})$$

$$\frac{!\Delta, \Gamma_1 \ \rhd\ M : !^n(A_1 \otimes A_2) \qquad !\Delta, \Gamma_2, x : !^n A_1, y : !^n A_2 \ \rhd\ N : A}{!\Delta, \Gamma_1, \Gamma_2 \ \rhd\ \text{let}\, \langle x, y\rangle = M\, \text{in}\, N : A}\,(\otimes_{out})$$

Well-typed quantum closure: $\Gamma \models [Q, L, M] : A$

A quantum closure $[Q, L, M]$ is well-typed of type A in a context Γ if $|L| \cap |\Gamma| = \emptyset$, $\mathcal{FV}(M) - |\Gamma| \subseteq |L|$, and

$$\Gamma, x_1 : qubit, \cdots, x_n : qubit \ \rhd\ M : A$$

is a valid typing judgement, where $\mathcal{FV}(M) - |\Gamma| = \{x_1, \cdots, x_n\}$.

A quantum closure is a *program* if $|\Gamma| = \emptyset$.

The properties of this typing system are similar to those of the one used in Lecture 7 for the simply-typed λ-calculus. In particular (see [4] for proof hints),

- Given a program $[Q, L, M]$ of type A and a derivation

  $$[Q, L, M] \ \rightsquigarrow^* \ [Q', L', M']$$

  $[Q', L', M']$ is still a program of type A. This property is known as *subject reduction* means that well-typedness is preserved by the reduction rules (i.e. by program execution), even in presence of decoherence and imprecision of the physical operations (cf, the use of $\rightsquigarrow$ in the statement).

- A well-typed program does not reach an error state. I.e. any probabilistic computation path of such a program is either infinite, or reaches a value state in a finite number of steps.This property is known as *type safety*.

- There exists a *type-inference algorithm* for the quantum λ-calculus.

Exercise 4

The type-inference algorithm mentioned above is described in detail in [3]. Provide a full implementation in Haskell of this algorithm.

Examples.

Example [fair coin]

$$\triangleright\ \mathsf{coin} : \top \multimap \mathsf{bit}$$

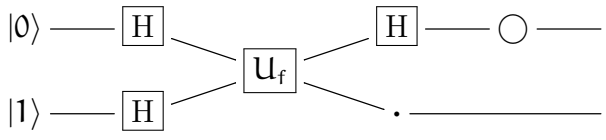
where $\mathsf{coin} = \lambda * .\mathsf{ms}(\mathsf{H}(\mathit{new}\, 0))$, as above.

Example [Deutsch algorithm]

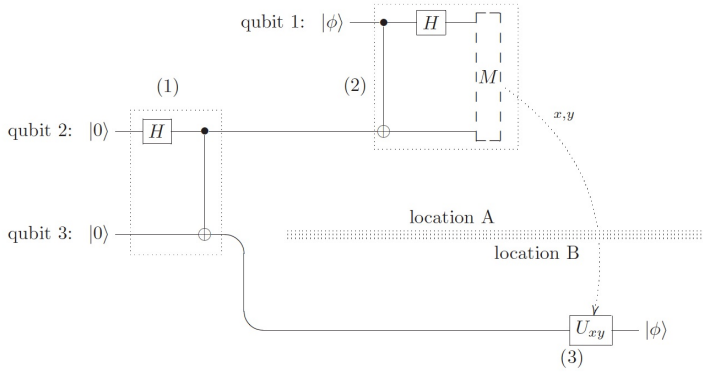
$$\triangleright\ \mathsf{Deutsch} : !((\mathsf{qubit} \otimes \mathsf{qubit} \multimap \mathsf{qubit} \otimes \mathsf{qubit}) \multimap \mathsf{bit})$$

where

$$\begin{aligned}
&\mathsf{Deutsch}\ U_f\ = \\
&\qquad \mathsf{let\,comb\,f\,g}\ =\ \lambda\langle x, y\rangle.\langle fx, gy\rangle \\
&\qquad \mathsf{in\ let}\langle x, y\rangle\ =\ (\mathsf{comb\,H}\,(\lambda x.x))\,(U_f\langle \mathsf{H}(\mathit{new}\, 0), \mathsf{H}(\mathit{new}\, 1)\rangle) \\
&\qquad \mathsf{in\ ms\ x}
\end{aligned}$$



Example [the teleportation protocol]



- Component (1): generates an EPR pair of entangled qubits:

$$\triangleright\ C_1 : !(\top \multimap \mathsf{qubit} \otimes \mathsf{qubit})$$

  where $C_1 = \lambda x.\mathsf{CNOT}\,\langle \mathsf{H}(new\,0), new\,0\rangle$

- Component (2): performs a Bell measurement and outputs two classical bits::

$$\triangleright\ C_2 : !(\mathsf{qubit} \multimap (\mathsf{qubit} \multimap \mathsf{bit} \otimes \mathsf{bit}))$$

  where $C_2 = \lambda q_1.\lambda q_2.(\mathsf{let}\,\langle x, y\rangle = \mathsf{CNOT}\,\langle q_1, q_2\rangle\,\mathsf{in}\,\langle ms(Hx), msy\rangle$

- Component (3): performs a correction::

$$\triangleright\ U : !(\mathsf{qubit} \multimap (\mathsf{bit} \otimes \mathsf{bit} \multimap \mathsf{qubit}))$$

  where

$$U = \lambda q.\lambda\langle x, y\rangle.\mathsf{if}\,x\,\mathsf{then}\,(\mathsf{if}\,y\,\mathsf{then}\,U_{11}\,q\,\mathsf{else}\,,U_{10}\,q)\ \mathsf{else}\,(\mathsf{if}\,y\,\mathsf{then}\,U_{01}\,q\,\mathsf{else}\,,U_{00}\,q)$$

  where

$$U_{00} \mathrel{\widehat{=}} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad U_{01} \mathrel{\widehat{=}} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad U_{10} \mathrel{\widehat{=}} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad U_{11} \mathrel{\widehat{=}} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Thus yielding

$$\triangleright\ \mathsf{Teleportation} : (\mathsf{qubit} \multimap \mathsf{bit} \otimes \mathsf{bit}) \otimes (\mathsf{bit} \otimes \mathsf{bit} \multimap \mathsf{qubit})$$

where

$$\begin{aligned} \mathsf{Teleportation} \ = {} & \mathsf{let}\,\langle x, y\rangle = C_1 * \,\mathsf{in} \\ & \mathsf{let}\,f = C_2\,x\,\mathsf{in} \\ & \mathsf{let}\,g = U\,y\,\mathsf{in}\,\langle f, g\rangle \end{aligned}$$

Thus, the teleportation protocol creates two functions f and g, non duplicable because they depend on the state of the pair of entangled qubits x and y, and such that $(g \cdot f)(z) = z$ for an arbitrary qubit z, and $(f \cdot g)(x, y) = (x, y)$ for bits x and y. This pair of mutually inverse functions can only be used once because each of them contains an embedded qubit. Actually, they witness a *single-use isomorphim* between the (otherwise non isomorphic) types qubit and $\mathsf{bit} \otimes \mathsf{bit}$.

Example [execution of the teleportation protocol]

In the sequel, consider the following abbreviations:

$$\begin{aligned} M_{p,p'} &\mathrel{\widehat{=}} \mathsf{let}\, f = C_2\, p\, \mathsf{in}\, \mathsf{let}\, g = U p'\, \mathsf{in}\, g(f p_0) \\ B_{p_1} &\mathrel{\widehat{=}} \lambda q_1.\mathsf{let}\, \langle p, p' \rangle = \mathsf{CNOT}\langle q_1, p_1 \rangle \in \langle \mathsf{ms}(\mathsf{Hp}), \mathsf{msp}' \rangle \\ U_{p_2} &\mathrel{\widehat{=}} \lambda\langle x, y \rangle.\mathsf{if}\, x\, \mathsf{then}\, (\mathsf{if}\, y\, \mathsf{then}\, U_{11} p_2\, \mathsf{else}\, , U_{10} p_2) \\ &\qquad\qquad \mathsf{else}\, (\mathsf{if}\, y\, \mathsf{then}\, U_{01} p_2\, \mathsf{else}\, , U_{00} p_2) \end{aligned}$$

$$[\alpha|0\rangle + \beta|1\rangle,\ \mathrm{let}\langle p, p'\rangle = C_1 * \ \mathrm{in}\ \mathrm{let}\, f = C_2 p\ \mathrm{in}\ \mathrm{let}\, g = U p'\ \mathrm{in}\ g(f p_0)]$$

$$\longrightarrow_1 \quad [\alpha|0\rangle + \beta|1\rangle,\ \mathrm{let}\langle p, p'\rangle = \mathrm{CNOT}\,\langle H(new\,0), new\,0\rangle\ \mathrm{in}\ M_{p,p'}]$$

$$\longrightarrow_1 \quad [(\alpha|0\rangle + \beta|1\rangle) \otimes |0\rangle,\ \mathrm{let}\,\langle p, p'\rangle = \mathrm{CNOT}\,\langle Hp1, new\,0\rangle\ \mathrm{in}\ M_{p,p'}]$$

$$\longrightarrow_1 \quad [(\alpha|0\rangle + \beta|1\rangle) \otimes \tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle),\ \mathrm{let}\,\langle p, p'\rangle = \mathrm{CNOT}\,\langle p1, new\,0\rangle\ \mathrm{in}\ M_{p,p'}]$$

$$\longrightarrow_1 \quad [(\alpha|0\rangle + \beta|1\rangle) \otimes \tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |0\rangle,\ \mathrm{let}\,\langle p, p'\rangle = \mathrm{CNOT}\,\langle p1, p2\rangle\ \mathrm{in}\ M_{p,p'}]$$

$$\longrightarrow_1 \quad [(\alpha|0\rangle + \beta|1\rangle) \otimes \tfrac{1}{\sqrt{2}}(|00\rangle + |11\rangle),\ \mathrm{let}\,\langle p, p'\rangle = \langle p1, p2\rangle\ \mathrm{in}\ M_{p,p'}]$$

$$\longrightarrow_1 \quad [(\alpha|0\rangle + \beta|1\rangle) \otimes \tfrac{1}{\sqrt{2}}(|00\rangle + |11\rangle),\ \mathrm{let}\, f = C_2\, p_1\ \mathrm{in}\ \mathrm{let}\, g = U\, p2\ \mathrm{in}\ g(f p_0)]$$

$$\longrightarrow_1^* \quad [(\alpha|0\rangle + \beta|1\rangle) \otimes \tfrac{1}{\sqrt{2}}(|00\rangle + |11\rangle),\ U_{p_2}(B p_1, p_0)]$$

$$\longrightarrow_1 \quad [(\alpha|0\rangle + \beta|1\rangle) \otimes \tfrac{1}{\sqrt{2}}(|00\rangle + |11\rangle),\ U_{p_2}(\mathrm{let}\,\langle p, p'\rangle = \mathrm{CNOT}\langle p_0, p_1\rangle\ \mathrm{in}\ \langle ms(Hp), ms\, p'\rangle)]$$

$$\longrightarrow_1 \quad [\tfrac{1}{\sqrt{2}}(\alpha|000\rangle + \alpha|011\rangle + \beta|110\rangle + \beta|101\rangle),\ U_{p_2}(\mathrm{let}\,\langle p, p'\rangle = \langle p_0, p_1\rangle\ \mathrm{in}\ \langle ms(Hp), ms\, p'\rangle)]$$

$$\longrightarrow_1 \quad [\tfrac{1}{\sqrt{2}}(\alpha|000\rangle + \alpha|011\rangle + \beta|110\rangle + \beta|101\rangle),\ U_{p_2}\langle ms(Hp_0), ms\, p_1\rangle]$$

$$\longrightarrow_1 \quad [\tfrac{1}{2}(\alpha|000\rangle + \alpha|011\rangle + \alpha|100\rangle + \alpha|111\rangle + \beta|010\rangle + \beta|001\rangle + \beta|110\rangle + \beta|101\rangle),\ U_{p_2}\langle msp_0, ms\, p_1\rangle]$$

$$\begin{cases} \longrightarrow_{\frac{1}{2}} & [\tfrac{1}{\sqrt{2}}(\alpha|000\rangle + \alpha|011\rangle + \beta|010\rangle + \beta|001\rangle, U_{p_2}\langle 0, msp_1\rangle] \\ \longrightarrow_{\frac{1}{2}} & [\tfrac{1}{\sqrt{2}}(\alpha|100\rangle + \alpha|111\rangle + \beta|110\rangle + \beta|101\rangle, U_{p_2}\langle 1, msp_1\rangle] \end{cases}$$

$$\begin{cases} \longrightarrow_{\frac{1}{2}} & [(\alpha|000\rangle + \beta|001\rangle, U_{p_2}\langle 0, 0\rangle] & \longrightarrow_1^* & [(\alpha|000\rangle + \beta|001\rangle, U_{00} p_2] \\ \longrightarrow_{\frac{1}{2}} & [(\alpha|011\rangle + \beta|010\rangle, U_{p_2}\langle 0, 1\rangle] & \longrightarrow_1^* & [(\alpha|011\rangle + \beta|010\rangle, U_{01} p_2] \\ \longrightarrow_{\frac{1}{2}} & [(\alpha|100\rangle + \beta|101\rangle, U_{p_2}\langle 1, 0\rangle] & \longrightarrow_1^* & [(\alpha|100\rangle + \beta|101\rangle, U_{10} p_2] \\ \longrightarrow_{\frac{1}{2}} & [(\alpha|111\rangle + \beta|110\rangle, U_{p_2}\langle 1, 1\rangle] & \longrightarrow_1^* & [(\alpha|111\rangle + \beta|110\rangle, U_{11} p_2] \end{cases}$$

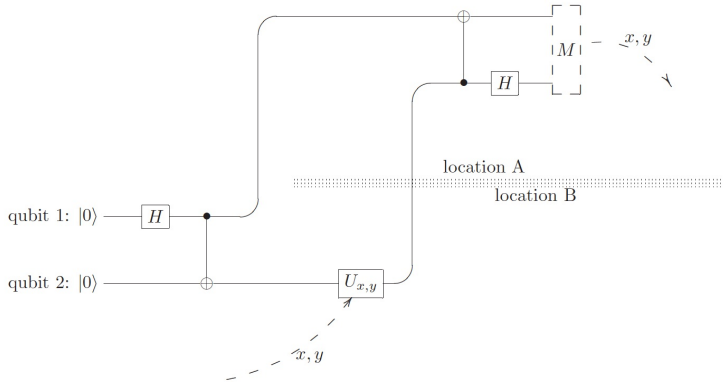
$$\begin{cases} \longrightarrow_1 & [(\alpha|000\rangle + \beta|001\rangle, p_2] & = & [|00\rangle \otimes (\alpha|0\rangle + \beta|1\rangle, p_2] \\ \longrightarrow_1 & [(\alpha|010\rangle + \beta|011\rangle, p_2] & = & [|01\rangle \otimes (\alpha|0\rangle + \beta|1\rangle, p_2] \\ \longrightarrow_1 & [(\alpha|100\rangle + \beta|101\rangle, p_2] & = & [|10\rangle \otimes (\alpha|0\rangle + \beta|1\rangle, p_2] \\ \longrightarrow_1 & [(\alpha|110\rangle + \beta|111\rangle, p_2] & = & [|11\rangle \otimes (\alpha|0\rangle + \beta|1\rangle, p_2] \end{cases}$$

Exercise 5

Justify each step of the reduction above.

Exercise 6

Consider now the dense coding protocol depicted below:



Reduce the following quantum closure

$$[|\,\rangle,\ \mathsf{let}\langle \mathsf{p}, \mathsf{p}'\rangle = \mathsf{C}_1 * \ \mathsf{in}\, \mathsf{let}\, \mathsf{f} = \mathsf{C}_2\mathsf{p}\, \mathsf{in}\, \mathsf{let}\, \mathsf{g} = \mathsf{U}\mathsf{p}'\, \mathsf{in}\, \mathsf{f}(\mathsf{g}\langle \mathsf{0}, \mathsf{1}\rangle)]$$

Exercise 7

Reference [4] extends the calculus with

- a term for recursive function definition;
- the possibility to accommodate infinite data types in the language.

Read the paper and discuss typing and reduction for these new terms. Give examples.

Towards a Curry-Howard-Lambek Correspondence for quantum computation.

Once fixed a specific quantum λ-calculus, the Curry-Howard-Lambek correspondence will have (tunned) families of *symmetrical monoidal* categories, and *linear logic* in the remaining vertexes of the triangle. The former was already discussed in the previous module. Let us give an idea of what linear logic is about.

Linear logic

In a previous lecture we resorted to Natural Deduction to introduce a proof system for a propositional logic. Let us consider here the equivalent formulation in term of the Getzen sequent calculus in which handling assumptions is more explicit. In a sequent

$$H_1, H_2, \cdots H_n \vdash A$$

assumptions are supposed to form a sequence rather than a set. The *structural* rules are directly concerned with the manipulation of assumptions:

$$\frac{}{A \vdash A}\,(\mathsf{Id}) \qquad \frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, B, A, \Delta \vdash C}\,(\mathsf{Exchange})$$

$$\frac{\Gamma, A, A \vdash B}{\Gamma, A \vdash B}\,(\mathsf{Contract}) \qquad \frac{\Gamma \vdash B}{\Gamma, A, \Delta \vdash B}\,(\mathsf{Weak})$$

Then, the sequent calculus adds the *logic* rules in the form of a *rigth* and a *left* rule, depending on which side of a sequent the connective appears (this can be shown equivalent to the pair of introduction / elimination rules used before). Thus, for the $\wedge, \Rightarrow$ fragment, one gets:

$$\frac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma, \Delta \vdash A \wedge B}\,(\wedge\,\mathrm{R}) \qquad \frac{\Gamma, A, B \vdash C}{\Gamma, A \wedge B \vdash C}\,(\wedge\,\mathrm{L})$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}\,(\Rightarrow\,\mathrm{R}) \qquad \frac{\Gamma \vdash A \quad B, \Delta \vdash C}{\Gamma, A \Rightarrow B, \Delta \vdash C}\,(\Rightarrow\,\mathrm{L})$$

$$\frac{\Gamma \vdash A \quad A, \Delta \vdash B}{\Gamma, \Delta \vdash B}\,(\mathsf{cut})$$

Note that the (cut) allows the use of lemmas in formal proofs. An important notion in proof theory is that of *cut elimination*, a proof transformation in order to eliminate all occurrences of this rule in the proof. The latter, with all lemmas removed, becomes completely explicit, i.e. *cut-free*. Such a transformation is always possible as proved by Gentzen himself.

What is the *linear* version of this logic?

Basically, the same logic (usually linear conjunctiona and linear implication are written as $\otimes$ and $\multimap$, respectively) without both the ($\mathsf{Contract}$) and the (Weak) rules. Note those rules are responsible for *duplicating* and *discarding* assumptions. In a category with

products, for example a CCC, they correspond to the diagonal function and projections, i.e.

$$\frac{f : \Gamma \times A \times A \longrightarrow B}{f \cdot (id \times \Delta_A) : \Gamma \times A \longrightarrow B} \qquad \frac{f : \Gamma \longrightarrow B}{f \cdot \pi_1 : \Gamma \times A \longrightarrow B}$$

Exercise 8

Discuss if the following sequents can be proved in linear logic:

- $A \vdash A \otimes A$
- $(A \otimes A) \multimap B \vdash A \multimap B$

The structural rules can be re-introduced in linear logic through the exponential ! operator defined by

$$\frac{\Gamma, A \vdash B}{\Gamma, !A \vdash B}\,(!L) \qquad \frac{!\Gamma \vdash A}{!\Gamma \vdash !A}\,(!R)$$

yielding the new structural rules:

$$\frac{\Gamma \vdash B}{\Gamma, !A \vdash B}\,(Weak) \qquad \frac{\Gamma, !A, !A \vdash B}{\Gamma, !A \vdash B}\,(Contract)$$

Classical implication is recovered as $A \Rightarrow B \mathrel{\widehat{=}} !A \multimap B$.

References

[1] Alejandro Díaz-Caro. A lambda calculus for density matrices. In B.Y. Chang, editor, *Programming Languages and Systems. APLAS 2017.* Springer Lecture Notes in Computer Science (10695), 2017.

[2] Ugo Dal Lago, Andrea Masini, and Margherita Zorzi. On a measurement-free quantum lambda calculus with classical control. *Math. Struct. Comput. Sci.*, 19(2):297–335, 2009.

[3] Peter Selinger and Benoît Valiron. A lambda calculus for quantum computation with classical control. *Mathematical Structures in Computer Science*, 16(3):527–552, 2006.

[4] Peter Selinger and Benoît Valiron. Quantum lambda calculus. In Simon Gay and Ian Mackie, editors, *Semantic Techniques in Quantum Computation*, pages 135–172. Cambridge University Press, 2009.

[5] André van Tonder. A lambda calculus for quantum computation. *SIAM J. Comput.*, 33(5):1109–1135, 2004.

[6] M. Ying. *Foundations of Quantum Programming*. Elsevier, 2016.