Lambda Calculo Cuántico

Gabriel Eberlein
Universidad Nacional de Rosario, FCEIA

Resumen
En este informe se presenta un lenguaje de dominio específico escrito en Haskell y basado en el diseño de un cálculo lambda adaptado para computación cuántica dado en *Selinger, P and Valiron, B. (2009) Quantum Lambda Calculus.*

Índice

1. Introduccion

El informe está separado en cuatro secciones; en la primera se presenta la sintaxis del lenguaje con una extensión azucarada del mismo, en la segunda la semántica, en la tercera se explican algunas cuestiones de diseño y, finalmente, se muestra cómo utilizar el lenguaje. Al final de cada sección se dan los módulos donde se implementan los conceptos planteados con una leve explicación de cómo se hicieron.

2. Sintaxis

2.1. Sintaxis Nucleo

La sintaxis concreta del lenguaje núcleo está dada de la siguiente manera.

$program ::= decl\ program \mid \epsilon$

$decl ::= \textbf{def}\ x = term$

$term ::= \lambda x.term$
$\quad \mid atom \mid atom\ atom$
$\quad \mid \textbf{let}\ \langle x,y\rangle = term\ \textbf{in}\ term$
$\quad \mid \textbf{let rec}\ f\ x = term\ \textbf{in}\ term$
$\quad \mid \textbf{match}\ term\ \textbf{with}$
$\quad\quad (x \rightarrow term \mid y \rightarrow term)$

$atom ::= c \mid x \mid * \mid (term) \mid \langle term,term\rangle$
$\quad \mid \textbf{injl}(term) \mid \textbf{injr}(term)$
$\quad \mid \textbf{printState}\ string\ term$

$c ::= \textbf{new} \mid \textbf{meas} \mid U$

$U ::= \textbf{X} \mid \textbf{Y} \mid \textbf{Z} \mid \textbf{H} \mid \textbf{CNOT}$

2.2. Sintaxis Azucarada

El lenguaje se extiende azucarándolo con las siguientes definiciones

$\lambda x_1 \dots x_n.\, M \stackrel{\text{def}}{=} \lambda x_1.\lambda x_2.\dots.\lambda x_n.\, M$

$M_1 M_2 \dots M_n \stackrel{\text{def}}{=} (\dots((M_1 M_2) M_3) \dots M_n)$

$\langle M_1, \dots, M_n\rangle \stackrel{\text{def}}{=} \langle M_1, \langle M_2, \dots \langle M_{n-1}, M_n\rangle \dots \rangle$

$\lambda\langle x, y\rangle.M \stackrel{\text{def}}{=} \lambda z.(let\ \langle x, y\rangle = z\ in\ M)$

$\lambda *.M \stackrel{\text{def}}{=} \lambda z.M$ (z variable fresca)

$let\, x = M\, in\, N \stackrel{\text{def}}{=} (\lambda x.N)M$

$let * = M\, in\, N \stackrel{\text{def}}{=} (\lambda z.N)M$
(z variable fresca)

$let\, f\, y_1 \dots y_n = M\, in\, N \stackrel{\text{def}}{=}$
$\quad let\, f = (\lambda y_1 \dots y_n.M)\, in\, N$

$let\, rec\, f\, y_1 \dots y_n = M\, in\, N \stackrel{\text{def}}{=}$
$\quad let\, rec\, f = (\lambda y_1 \dots y_n.M)\, in\, N$

$0 \stackrel{\text{def}}{=} injr(*)$

$1 \stackrel{\text{def}}{=} injl(*)$

$if\, P\, then\, M\, else\, N \stackrel{\text{def}}{=}$
$\quad match\, P\, with\, (x \rightarrow M | y \rightarrow N)$
(x,y variables frescas)

2.3. Modulos

- Parser.hs: en este modulo se hace el parseo del texto de los archivos al árbol de sintaxis azucarada
- AST.hs: en este modulo se encuentran las definiciones de los arboles de sintaxis abstracta para términos núcleo (Term) y términos azucarados (STerm). Cabe aclarar que los STerm son fully named mientras que los Term utilizan indices de Bruijn
- Elab.hs: en este modulo se definen las funciones de elaboracion para pasar de STerm a Term.

3. Semántica

Se presentan las reglas de la semántica en paso chico salvo por las de congruencia que se describen en paso grande para que sean mas acorde a lo expresado en el código.

3.1. Estrategia de Reduccion

La evaluación del lenguaje sigue una estrategia de reducción Call By Value; esto es importante aclararlo, ya que si se eligiese una estrategia Call By Name la computación sería distinta a lo deseado. Por ejemplo:

```
def q = new 0
def main = let * = H q in H q
```

Si se utilizase la estrategia Call By Name, cada vez que se invoque a **q** se estaría creando un nuevo cubit por lo que el estado resultado sería $\frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$ (dos cubits balanceados).

En cambio con la estrategia Call By Value que utilizamos, **q** siempre hará referencia a un único cubit y el código generará un estado resultado $|0\rangle$. (Al aplicarse dos veces la compuerta H, esta se cancela; más adelante se ve por qué).

3.2. Reglas de evaluación

$\rightarrow_p$ indica un paso chico con probabilidad p.

El estado de evaluación está dado por una 3-upla $(\Gamma, |\psi\rangle, t)$; donde Γ es el ambiente de las variables definidas, $|\psi\rangle$ es el estado de los cubits y t es el término a evaluar.

Reglas de control clásico

$$(\Gamma, |\psi\rangle, x) \rightarrow (\Gamma, |\psi\rangle, \Gamma(x))$$

$$(\Gamma, |\psi\rangle, (\lambda x.\text{N})\text{V}) \rightarrow (\Gamma, |\psi\rangle, \text{N[V/x]})$$

$$(\Gamma, |\psi\rangle, \text{let } \langle \text{x,y} \rangle = \langle \text{V,W} \rangle \text{ in N}) \rightarrow (\Gamma, |\psi\rangle, \text{N[V/x,W/y]})$$

$$(\Gamma, |\psi\rangle, \text{match injl(V) with } (\text{x} \rightarrow M \mid y \rightarrow N) \rightarrow (\Gamma, |\psi\rangle, \text{M[V/x]})$$

$$(\Gamma, |\psi\rangle, \text{match injr(W) with } (\text{x} \rightarrow M \mid y \rightarrow N) \rightarrow (\Gamma, |\psi\rangle, \text{N[W/y]})$$

$$(\Gamma, |\psi\rangle, \text{let rec f x = M in N}) \rightarrow (\Gamma, |\psi\rangle, \text{N}[\lambda x.(\text{let rec f x = M in M})/\text{f}])$$

$$(\Gamma, |\psi\rangle, \text{printState "str" V}) \rightarrow (\Gamma, |\psi\rangle, \text{V})$$

Reglas de operaciones cuánticas

$$(\Gamma, |\psi\rangle, \text{U } \langle q_0, q_1, ..., q_n \rangle) \rightarrow (\Gamma, |\psi'\rangle, \langle q_0 q_1 \dots q_n \rangle)$$
$(|\psi'\rangle$ es $|\psi\rangle$ luego de aplicar la compuerta U a los qbits $q_0 \dots q_n$)

$$(\Gamma, \alpha|\psi_0\rangle + \beta|\psi_1\rangle, \text{meas } x_i) \rightarrow_{|\alpha|^2} (\Gamma, |\psi_0\rangle, 0)$$

$$(\Gamma, \alpha|\psi_0\rangle + \beta|\psi_1\rangle, \text{meas } x_i) \rightarrow_{|\beta|^2} (\Gamma, |\psi_1\rangle, 1)$$

$$(\Gamma, |q_0 \dots q_n\rangle, \text{new 0}) \rightarrow (\Gamma, |q_0 \dots q_n\rangle \otimes |0\rangle, q_{n+1})$$

$$(\Gamma, |q_0 \dots q_n\rangle, \text{new 1}) \rightarrow (\Gamma, |q_0 \dots q_n\rangle \otimes |1\rangle, q_{n+1})$$

3.3. Reglas de congruencia

$\overset{\text{str}}{\Downarrow_p}$ **indica un paso grande imprimiendo str a consola con probabilidad p.**

$$\frac{(\Gamma, |\psi\rangle, N) \overset{\text{str}}{\Downarrow_p} (\Gamma', |\psi'\rangle, V)}{(\Gamma, |\psi\rangle, MN) \overset{\text{str}}{\Downarrow_p} (\Gamma', |\psi'\rangle, MV)}$$

$$\frac{(\Gamma, |\psi\rangle, M) \overset{\text{str}}{\Downarrow_p} (\Gamma', |\psi'\rangle, W)}{(\Gamma, |\psi\rangle, MV) \overset{\text{str}}{\Downarrow_p} (\Gamma', |\psi'\rangle, WV)}$$

$$\frac{(\Gamma, |\psi\rangle, N) \overset{\text{str}}{\Downarrow_p} (\Gamma', |\psi'\rangle, V)}{(\Gamma, |\psi\rangle, \langle M, N\rangle) \overset{\text{str}}{\Downarrow_p} (\Gamma', |\psi'\rangle, \langle M, V\rangle)}$$

$$\frac{(\Gamma, |\psi\rangle, M) \overset{\text{str}}{\Downarrow_p} (\Gamma', |\psi'\rangle, W)}{(\Gamma, |\psi\rangle, \langle M, V\rangle) \overset{\text{str}}{\Downarrow_p} (\Gamma', |\psi'\rangle, \langle W, V\rangle)}$$

$$\frac{(\Gamma, |\psi\rangle, N) \overset{\text{str}}{\Downarrow_p} (\Gamma', |\psi'\rangle, V)}{(\Gamma, |\psi\rangle, injl(N)) \overset{\text{str}}{\Downarrow_p} (\Gamma', |\psi'\rangle, injl(V))}$$

$$\frac{(\Gamma, |\psi\rangle, N) \overset{\text{str}}{\Downarrow_p} (\Gamma', |\psi'\rangle, V)}{(\Gamma, |\psi\rangle, injr(N)) \overset{\text{str}}{\Downarrow_p} (\Gamma', |\psi'\rangle, injr(V))}$$

$$\frac{(\Gamma, |\psi\rangle, P) \overset{\text{str}}{\Downarrow_p} (\Gamma', |\psi'\rangle, V)}{(\Gamma, |\psi\rangle, match\ P\ with \dots) \overset{\text{str}}{\Downarrow_p} (\Gamma', |\psi'\rangle, match\ V\ with \dots)}$$

$$\frac{(\Gamma, |\psi\rangle, M) \overset{\text{str}}{\Downarrow_p} (\Gamma', |\psi'\rangle, W)}{(\Gamma, |\psi\rangle, let\ \langle x, y\rangle = M\ in\ N) \overset{\text{str}}{\Downarrow_p} (\Gamma', |\psi'\rangle, let\ \langle x, y\rangle = W\ in\ N)}$$

$$\frac{(\Gamma, |\psi\rangle, N) \overset{\text{str}}{\Downarrow_p} (\Gamma', |\psi'\rangle, V)}{(\Gamma, |\psi\rangle, printState\ "str'"\ N) \overset{str\ +\!\!+\ str'\ +\!\!+\ |\psi'\rangle}{\Downarrow_p} (\Gamma', |\psi'\rangle, printState\ "str'"\ V)}$$

3.4. Valores

Los valores como resultado de una evaluación se definen de la siguiente manera,

$val ::= c$ | **qbit** n
| $\lambda x.term$
| **injl** val | **injr** val
| * | $\langle val, val\rangle$

3.5. Modulos

- **AST.hs:** Se define el tipo de dato Value para representar los valores posibles de la evaluacion.
- **Eval.hs:** En este modulo se aplican las regla de la semántica a Term devolviendo un Value.

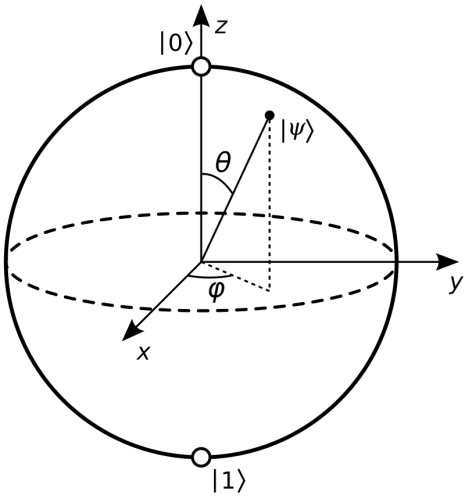
4. Diseño

En esta sección se explica como funcionan las operaciones cuanticas, como se implemento el estado que lleva el software y la monada utilizada

4.1. Operaciones Cuanticas

Para aclarar lo planteado al inicio del informe, el lenguaje es un cálculo lambda relativamente clásico extendido para realizar operaciones del modelo de circuitos cuánticos. Esto quiere decir que la computación cuántica se realiza a través de aplicar compuertas a cubits que transforman sus estados.

El estado de un cubit puede representarse mediante un punto en la esfera de Bloch, esto se puede implementar como un vector de dos componentes complejos. Extendiendo a n cubits seria un vector de 2^n componentes complejos.



Representacion de un cubit en la esfera de Bloch

El efecto de las compuertas en la representación de la esfera de Bloch es rotar al vector π radianes sobre algún eje. Como las compuertas aplican rotaciones de un vector en el espacio, estas se pueden dar como matrices de transformación y así son implementadas en el programa.

Paso entonces a explicar las compuertas cuanticas y las dos operaciones basicas que componen el modelo de circuitos.

Operaciones Basicas

- `new:` introduce un qbit al estado siendo este el de menor significancia, el qbit inicialmente colapsara al argumento dado (0 o 1) siempre que sea medido.
- `meas:` mide un qbit colapsando su estado a 0 o 1, luego de ser colapsado el qbit se mantendrá en el mismo estado siempre que sea medido.

Compuertas Cuanticas
Como el efecto de la mayoría de las compuertas es aplicar una rotación del vector, no necesariamente representan un comportamiento lógico directo. De todas formas, en los casos donde si ocurra esto lo aclaro.

- `X:` Aplica una rotación sobre el eje X, es equivalente a la compuerta clasica NOT. Sea $|\psi\rangle = a|0\rangle + b|1\rangle$ entonces $X|\psi\rangle = b|0\rangle + a|1\rangle$
- `Y:` Aplica una rotación sobre el eje Y. Sea $|\psi\rangle = a|0\rangle + b|1\rangle$ entonces $Y|\psi\rangle = bi|0\rangle - ai|1\rangle$
- `Z:` Aplica una rotación sobre el eje Z. Sea $|\psi\rangle = a|0\rangle + b|1\rangle$ entonces $Z|\psi\rangle = a|0\rangle - b|1\rangle$
- `H:` Aplica una rotacion sobre el eje $\frac{(X+Z)}{\sqrt{2}}$. Sea $|\psi\rangle = a|0\rangle + b|1\rangle$ entonces $H|\psi\rangle = \frac{1}{\sqrt{2}}((a+b)|0\rangle + (a-b)|1\rangle)$.
  Cabe aclarar que si se aplica H a un estado base $|0\rangle$ o $|1\rangle$ se obtiene un cubit balanceado (con las mismas chances de ser 0 o 1).
- `CNOT:` El nombre de esta compuerta es

Controlled Not, se aplica a dos cubits y efectúa el siguiente mapeo, $|a, b\rangle \mapsto |a, a \oplus b\rangle$ ($\oplus$ es el operador xor). La idea es que si el primer cubit colapsa a 1 se aplique una operación NOT al segundo.

4.2. El Estado

El estado de la evaluación está dado por un nuevo tipo de dato llamado PState compuesto por dos componentes:

- **variables definidas:** esto simplemente es el entorno donde se llevan los valores de las variables libres declaradas
- **estado cuántico de los cubits:** es un vector de complejos que lleva el estado del sistema cuántico, el valor de cada elemento representa la amplitud de probabilidad de que el sistema colapse a su indice en el vector.

4.3. La Monada

Se define una nueva clase de monada llamada MonadQuantum que deriva de MonadIO, MonadState PState y MonadError. También se le agregan métodos para recuperar y actualizar los componentes de PState, imprimir un string, imprimir el estado cuántico y manipular el estado cuántico.

Se define un tipo QState usando transformadores de monadas sobre la monada IO:
`StateT PState (ExceptT Error IO)`

Esta monada puede instanciarse directamente en la clase MonadQuantum y es la que se utiliza finalmente para programar el software

4.4. Modulos

- **Monad.hs:** En este modulo se puede ver la definición de la monada utilizada junto con la clase MonadQuantum y el estado del programa.
- **Qbit.hs:** En este modulo se encuentran las funciones necesarias para realizar las distintas operaciones sobre el vector de los cubits como por ejemplo; agregar un cubit, medir un cubit, aplicar una compuerta e intercambiar un par de estos.
- **Tensor.hs:** En este modulo solamente se define una clase Tensor con un metodo para aplicar el producto tensorial y se instancian las matrices y los vectores.
- **PrettyPrinter.hs:** En este modulo se define la función prettyQbits utilizada para formatear el estado antes de imprimirlo a consola.

5. Uso

El proyecto está subido a github para ser clonado. Para poder utilizarlo se debe tener las herramientas **stack** y **cabal** instaladas.

Una vez instalado el proyecto y las herramientas, primero debe inicializarse usando:
`stack init`

Luego para buildear el proyecto:
`stack build`

Y finalmente puede ejecutarse usando:
`stack run -- <opciones>`

Las opciones de ejecución válidas son las siguientes:

- `(-h/--help)`: imprime un menú con las opciones de ejecución.
- `(-i/--interactive)`: ejecuta el modo interactivo.
- `(-f/--file) <archivo>`: evalúastack run el archivo pasado por linea de comando.

5.1. Modo Interactivo

El modo interactivo tiene las siguientes opciones de comandos:

- `:q` - Sale del modo interactivo y termina la ejecución.
- `:h` - Imprime un menú de ayuda con los comandos utilizables.
- `:c` - Limpia el estado de evaluación por completo
- `:st` - Muestra el estado de los cubits como si se hubiese usado el comando printState
- `:l <archivo>` - Carga y evalúa un archivo, sus efectos quedan guardados en el estado del modo interactivo.
- `:p <decl/term>` - Parsea la declaración o el termino dado y devuelve su AST azucarado.
- `<decl/term>` - Evalúa la declaración o el termino dado.

5.2. Programas de Ejemplo

Hay escritos dos programas de ejemplo para ver cómo se puede utilizar el lenguaje.

`stack run -- -f ejemplos/coin.ql`
Este programa define una función recursiva que en cada recursion tira una moneda (mide un cubit balanceado), si el resultado es 1 aplica la compuerta de Hadamard a un cubit, si el resultado es 0 termina y lo retorna.

`stack run -- -f ejemplos/tp.ql`
Este programa es el algoritmo de tele-transportación cuántica; utilizando un par de cubits entrelazados, puede copiar el estado cuántico de un tercer cubit.

Referencias

[1] Wikipedia. Quantum logic gate, 2025. Último acceso: 2 de febrero de 2025.

[2] P. Selinger y B. Valiron. *Semantic Techniques in Quantum Computation*, chapter 4. Cambridge University Press, 2009.