The Definition of λ_Q

May 14, 2021

Contents

1 Syntax

t	$::=$		**terms**:
		x	variable
		`unit`	constant unit
		`true`	constant true
		`false`	constant false
		$\lambda\ x : T.t$	function abstraction
		$t\ t$	function application
		(t, t)	pair
		$t.1$	first projection
		$t.2$	second projection
		`if` t `then` t `else` t	conditional
		`run` C	static lifting
		$\kappa\ p : W.C$	circuit abstraction
v	$::=$		**values**:
		$\lambda\ x : T.t$	abstraction value
		(v, v)	pair value
		`unit`	unit value
		`true`	true value
		`false`	false value

		$\kappa\ p : W.C$	circuit value
T	$::=$		**types**:
		`Unit`	unit type
		`Bool`	boolean type
		$T \times T$	product type
		$T \to T$	function type
		$T \rightsquigarrow T$	circuit type
Γ	$::=$		**contexts**:
		$\varnothing$	empty context
		$\Gamma, x : T$	term variable binding
W	$::=$		**wire types**:
		`1`	wire unit type
		`Bit`	bit type
		`Qubit`	qubit type
		$W \otimes W$	wire product type
p	$::=$		**wire patterns**:
		$()$	empty
		w	wire variable
		(p, p)	wire pair
C	$::=$		**circuits**:
		$\texttt{output}\ p$	output a pattern
		$p_2 \leftarrow \texttt{gate}\ g\ p_1; C$	gate application
		$p \leftarrow C; C$	circuit composition
		$x \hookleftarrow \texttt{lift}\ p; C$	dynamic lifting
		$\texttt{capp}\ t\ \texttt{to}\ p$	circuit application
g	$::=$		**gates**:
		$\texttt{new}_0$	generate a bit 0
		$\texttt{new}_1$	generate a bit 1
		$\texttt{init}_0$	generate a qubit 0
		$\texttt{init}_1$	generate a qubit 1
		`meas`	measurement gate
		`discard`	disgard gate
G	$::=$		**gate types**:
		$\mathcal{G}(W, W)$	simple gate type
Ω	$::=$		**wire contexts**:
		$\varnothing$	empty context
		$\Omega, w : W$	wire variable binding

2 Typing Rules

First, we need to define what is a **well-formed wire context**.

Definition 1 (Well-formed Wire Contexts)**.** *A wire context Ω is well-formed, if there are no duplicate wire variables in it. For simplicity, we always assume the wire contexts are well-formed in the following contexts. And when we write Ω_1, Ω_2, we require Ω_1 and Ω_2 to be disjoint to preserve the well-formedness.*

Since there are some different kinds of terms: (λ-)terms, wire patterns, gates and circuits, we have defined some different typing relations for each of them.

- $\Omega \vdash p : W$ is the typing relation for patterns.
- $\Gamma; \Omega \vdash C : W$ is the typing relation for circuits;
- $\Gamma \vdash t : T$ is the typing relation for (λ-)terms;
- $g : G$ is the typing relation for gates.

2.1 Typing rules for gates

Note that since we only support built-in gates, the typing rules for gates are extremely simple, just assigning a type to each built-in gate.

Typing rules for gates: $\boxed{g : G}$

$$\frac{}{\texttt{new}_0 : \mathcal{G}(\texttt{1}, \texttt{Bit})}$$

$$\frac{}{\texttt{new}_1 : \mathcal{G}(\texttt{1}, \texttt{Bit})}$$

$$\frac{}{\texttt{init}_0 : \mathcal{G}(\texttt{1}, \texttt{Qubit})}$$

$$\frac{}{\texttt{init}_1 : \mathcal{G}(\texttt{1}, \texttt{Qubit})}$$

$$\frac{}{\texttt{meas} : \mathcal{G}(\texttt{Qubit}, \texttt{Bit})}$$

$$\frac{}{\texttt{discard} : \mathcal{G}(\texttt{Bit}, \texttt{1})}$$

2.2 Typing rules for patterns

Typing rules for patterns: $\boxed{\Omega \vdash p : W}$

$$\frac{}{\varnothing \vdash () : One}$$

$$\frac{}{w : W \vdash w : W}$$

$$\frac{\Omega_1 \vdash p_1 : W_1 \quad \Omega_2 \vdash p_2 : W_2}{\Omega_1, \Omega_2 \vdash (p_1, p_2) : W_1 \otimes W_2}$$

2.3 Typing rules for circuits

Typing rules for circuits: $\boxed{\Gamma; \Omega \vdash C : W}$

$$\frac{\Omega \vdash p : W}{\Gamma; \Omega \vdash \texttt{output}\ p : W} \qquad \text{(C-OUTPUT)}$$

$$\frac{g : \mathcal{G}(W_1, W_2) \quad \Omega_1 \vdash p_1 : W_1 \quad \Omega_2 \vdash p_2 : W_2 \quad \Gamma; \Omega_2, \Omega \vdash C : W}{\Gamma; \Omega_1, \Omega \vdash p_2 \leftarrow \texttt{gate}\ g\ p_1; C : W} \qquad \text{(C-GATE)}$$

$$\frac{\Gamma; \Omega_1 \vdash C : W \quad \Omega \vdash p : W \quad \Gamma; \Omega, \Omega_2 \vdash C' : W'}{\Gamma; \Omega_1, \Omega_2 \vdash p \leftarrow C; C' : W'} \qquad \text{(C-COMPOSE)}$$

$$\frac{\Omega \vdash p : W \quad \Gamma, x : |W|; \Omega' \vdash C : W'}{\Gamma; \Omega, \Omega' \vdash x \hookleftarrow \texttt{lift}\ p; C : W'} \qquad \text{(C-LIFT)}$$

$$\frac{\Gamma \vdash t : W_1 \rightsquigarrow W_2 \quad \Omega \vdash p : W_1}{\Gamma; \Omega \vdash \texttt{capp}\ t\ \texttt{to}\ p : W_2} \qquad \text{(C-CAPP)}$$

2.4 Typing rules for terms

Typing rules for (λ-)terms: $\boxed{\Gamma \vdash t : T}$

$$\frac{\Gamma; \varnothing \vdash C : W}{\Gamma \vdash \texttt{run}\ C : |W|} \qquad \text{(T-RUN)}$$

$$\frac{\Omega \vdash p : W_1 \quad \Gamma; \Omega \vdash C : W_2}{\Gamma \vdash \kappa\ p : W_1.C : W_1 \rightsquigarrow W_2} \qquad \text{(T-CABS)}$$

$$\frac{x : T \in \Gamma}{\Gamma \vdash x : T} \qquad \text{(T-VAR)}$$

$$\frac{\Gamma, x : T_1 \vdash t_2 : T_2}{\Gamma \vdash \lambda\ x : T_1.t_2 : T_1 \to T_2} \qquad \text{(T-ABS)}$$

$$\frac{\Gamma \vdash t_1 : T_{11} \to T_{12} \quad \Gamma \vdash t_2 : T_{11}}{\Gamma \vdash t_1\ t_2 : T_{12}} \qquad \text{(T-APP)}$$

$$\frac{}{\Gamma \vdash \texttt{true} : \mathit{Bool}} \qquad \text{(T-TRUE)}$$

$$\frac{}{\Gamma \vdash \texttt{false} : \mathit{Bool}} \qquad \text{(T-FALSE)}$$

$$\frac{\Gamma \vdash t_1 : \texttt{Bool} \quad \Gamma \vdash t_2 : T \quad \Gamma \vdash t_3 : T}{\texttt{if } t_1 \texttt{ then } t_2 \texttt{ else } t_3 : T} \tag{T-IF}$$

$$\frac{}{\Gamma \vdash \texttt{unit} : Unit} \tag{T-UNIT}$$

$$\frac{\Gamma \vdash t_1 : T_1 \quad \Gamma \vdash t_2 : T_2}{\Gamma \vdash (t_1, t_2) : T_1 \times T_2} \tag{T-PAIR}$$

$$\frac{\Gamma \vdash t : T_1 \times T_2}{\Gamma \vdash t.1 : T_1} \tag{T-FST}$$

$$\frac{\Gamma \vdash t : T_1 \times T_2}{\Gamma \vdash t.2 : T_2} \tag{T-SEC}$$

3 Operational Semantics

TODO: add semantics.