Wave-Style Token Machines and Quantum Lambda Calculi

Ugo Dal Lago
Università di Bologna, Italy & INRIA

Margherita Zorzi
Università di Verona, Italy

Particle-style token machines are a way to interpret proofs and programs, when the latter are written following the principles of linear logic. In this paper, we show that token machines also make sense when the programs at hand are those of a simple quantum λ-calculus with implicit qubits. This, however, requires generalising the concept of a token machine to one in which more than one particle travel around the term *at the same time*. The presence of multiple tokens is intimately related to entanglement and allows us to give a simple operational semantics to the calculus, coherently with the principles of quantum computation.

1 Introduction

One of the strongest trends in computer science is the (relatively recent) interest in exploiting new computing paradigms which go beyond the usual, classical one. Among these paradigms, quantum computing plays an important role. In particular, the quantum paradigm is having a deep impact on the notion of a computationally (in)tractable problem [18].

Even if quantum computing has catalysed the interest of a quite large scientific community, several theoretical aspects are still unexplored. As an example, the definition of a robust theoretical framework for quantum programming is nowadays still a challenge. A number of (paradigmatic) calculi for quantum computing have been introduced in the last ten years. Among them, some functional calculi, typed and untyped, have been proposed [5, 6, 7, 16, 19, 22], but we are still at a stage where it is not clear whether one calculus could be considered *canonical*. Since quantum data have to undergo restrictions such as no-cloning and no-erasing, it is not surprising that in most of the cited quantum calculi the use of resources is controlled. Linear logic therefore provides an ideal framework for quantum data treatment, since weakening and contraction (to which linear logic gives a special status) precisely correspond to erasing and copying via the Curry-Howard correspondence. But linear logic also offers another tool which has not been widely exploited in the quantum setting: its mathematical model in terms of operator algebras, i.e. the Geometry of Interaction (GoI in the following). Indeed, the latter provides a dynamical interpretation and a semantic account of the cut-elimination procedure as a flow of information circulating into a net structure. This idea can be formulated both as an algebra of bounded operators on a infinite-dimensional Hilbert space [11] or as a token-based machine [12, 14]. On the one hand, the Hilbert space on top of which the first formulation of GoI is given is precisely the canonical state space of a quantum Turing machine [2]. On the other hand, the definition of a token machine provides a mathematically simpler setting, which has already found a role in this context [4, 13].

In this paper, we show that token machines are also a model of a linear quantum λ-calculus with implicit quantum bits (qubits), called $\mathsf{Q}\Lambda$ and defined along the lines of van Tonder's λ_q [19]. This allows us to give an operational semantics to $\mathsf{Q}\Lambda$ which renders the quantum nature of $\mathsf{Q}\Lambda$ explicit: *type derivations become quantum circuits on the set of gates occurring in the underlying λ-term*. This frees us from the burden of having to define the operational semantics of quantum calculi in reduction style,

S. Alves and I. Cervesato (Eds): Third International Workshop on Linearity 2014 (LINEARITY'14)
EPTCS 176, 2015, pp. 64–78, doi:10.4204/EPTCS.176.6

which is known to be technically challenging in a similar setting [19]. On the other hand, the power of β-style axioms is retained in the form of an equational theory for which our operational semantics can be proved sound. Technically, the design of our token machine for $\mathsf{Q}\Lambda$, called $\mathsf{IAM}_{\mathsf{Q}\Lambda}$, is arguably more challenging than the one of classical token machines. Indeed, the principles of quantum computing, and the so-called *entanglement* in particular, force us to go towards *wave-style* machines, i.e., to machines where more than one particle can travel inside the program at the same time. Moreover, the possibly many tokens at hand are subject to synchronisation points, each one corresponding to unitary operators of arity greater than 1. This means that $\mathsf{IAM}_{\mathsf{Q}\Lambda}$, in principle, could suffer from deadlocks, let alone the possibility of non-termination. We here prove that these pathological situations can *never* happen. In the present paper we also establish a soundness theorem: we state and prove that the semantics induced by the token machine $\mathsf{IAM}_{\mathsf{Q}\Lambda}$ is sound with respect to $\mathsf{Q}\Lambda$'s equational theory, i.e. it is invariant with respect to term equivalence. The proof, which we only sketch and which can anyway be found in [8], is not trivial, since our notion of term has to deal with quantum superposition [15] and is thus non-standard. Finally, it is mandatory to recall that, even if the possibility of observing quantum data is a useful and expressive programming tool, considering a measurement-free calculus is a theoretically well-founded choice, since measurements can always been postponed [15]. Thus, this is not a limitation when one addresses computability issues.

The calculus $\mathsf{Q}\Lambda$ and its token machine $\mathsf{IAM}_{\mathsf{Q}\Lambda}$ are introduced in Section 2 and Section 3, respectively. Main results about $\mathsf{IAM}_{\mathsf{Q}\Lambda}$ are in Section 4. An extended version of this paper with more details, proofs and a gentle introduction to quantum computing is available [8].

2 The Calculus $\mathsf{Q}\Lambda$

An essential property of quantum programs is that quantum data, i.e. quantum bits, should always be uniquely referenced. This restriction follows from the well-known *no-cloning* and *no-erasing* properties of quantum physics, which state that a quantum bit cannot in general be duplicated nor canceled [15]. Syntactically, one captures this restriction by means of linearity: if every abstraction $\lambda x.M$ is such that there is *exactly one* free occurrence of x in M, then the substitution triggered by firing *any* redex is neither copying nor erasing and, as a consequence, coherent with the just stated principles. In this Section, we introduce a quantum linear λ-calculus in the style of van Tonder's λ_q [19] and give an equational theory for it. This is the main object of study of this paper, and is the calculus for which we will give a wave-style token machine in the coming sections.

2.1 The Language of Terms

Let us fix a finite set $\mathscr{U}$ of *unitary operators*, each on a finite-dimensional Hilbert space $\mathbb{C}^{2^n}$, where n can be arbitrary. To each such $\mathbf{U} \in \mathscr{U}$ we associate a symbol U and call n the *arity* of U. The syntactic categories of *patterns*, *bit constants*, *constants* and *terms* are defined by the following grammar:

$$P ::= x \mid \langle x, y \rangle; \qquad \textit{patterns}$$

$$B ::= |b\rangle_n; \qquad \textit{bit constants}$$

$$C ::= B \mid U; \qquad \textit{constants}$$

$$M, N ::= x \mid C \mid M \otimes N \mid MN \mid \lambda P.M. \qquad \textit{terms}$$

$$\frac{}{x:A\vdash x:A}\,(\mathsf{a_v}) \qquad \frac{}{\cdot\vdash|0\rangle:\mathbb{B}}\,(\mathsf{a_{q0}}) \qquad \frac{}{\cdot\vdash|1\rangle:\mathbb{B}}\,(\mathsf{a_{q1}}) \qquad \frac{}{\cdot\vdash U:\mathbb{B}^n\multimap\mathbb{B}^n}\,(\mathsf{a_U})$$

$$\frac{\Gamma,x:A\vdash M:B}{\Gamma\vdash\lambda x.M:A\multimap B}\,(\mathsf{I}^1_{\multimap}) \qquad \frac{\Gamma,x:A,y:B\vdash M:C}{\Gamma\vdash\lambda\langle x,y\rangle.M:(A\otimes B)\multimap C}\,(\mathsf{I}^2_{\multimap})$$

$$\frac{\Gamma\vdash M:A\multimap B \quad \Delta\vdash N:A}{\Gamma,\Delta\vdash MN:B}\,(\mathsf{E}_{\multimap}) \qquad \frac{\Gamma\vdash M:A \quad \Delta\vdash N:B}{\Gamma,\Delta\vdash M\otimes N:A\otimes B}\,(\mathsf{I}_{\otimes})$$

Figure 1: Typing Rules.

where n ranges over $\mathbb{N}$, b ranges over $\{0,1\}$, and x ranges over a denumerable, totally ordered set of variables $\mathbb{V}$. We always assume that the natural numbers occurring next to bits in any term M are pairwise distinct. This condition, by the way, is preserved by substitution when the substituted variable occurs (free) exactly once. Whenever this does not cause ambiguity, we elide labels and simply write $|b\rangle$ for a bit constant. Notice that pairs are formed via the binary operator $\otimes$.

We will sometime write $|b_1b_2\ldots b_k\rangle$ for $|b_1\rangle\otimes|b_2\rangle\otimes\ldots\otimes|b_k\rangle$ (where $b_1,\ldots,b_n\in\{0,1\}$). We work modulo variable renaming; in other words, terms are equivalence classes modulo α-conversion. Substitution up to α-equivalence is defined in the usual way. Observe that the terms of $\mathsf{Q}\Lambda$ are the ones of a λ-calculus with pairs (which are accessed by pattern-matching) endowed with constants for bits and unitary operators. We don't consider measurements here, and discuss the possibility of extending the language of terms in sections 5 and 6.

2.2 Judgements and Typing Rules.

We want terms to be non-duplicable and non-erasable by construction and, as a consequence, we adopt a linear type discipline. Formally, the set of types is defined as follows

$$A ::= \mathbb{B} \;\big|\; A\multimap B \;\big|\; A\otimes B,$$

where $\mathbb{B}$ is the ground type of bits. We write $\mathbb{B}^n$ for the n-fold tensor product $\overbrace{\mathbb{B}\otimes\ldots\otimes\mathbb{B}}^{n\text{ times}}$. Judgements and environments are defined as follows:

- A *linear environment* Γ is a (possibly empty) finite set of assignments in the form $x:A$. We impose that in a linear environment, each variable x occurs *at most* once;
- If Γ and Δ are two linear environments assigning types to distinct sets of variables, Γ,Δ denotes their union;
- A *judgement* is an expression $\Gamma\vdash M:A$, where Γ is a linear environment, M is a term, and A is a type in $\mathsf{Q}\Lambda$.

Typing rules are in Figure 1. Observe that contexts are treated multiplicatively and, as a consequence, variables always appear exactly once in terms. In other words, a *strictly linear type discipline* is enforced.

Example 1 (EPR States) *Consider the term $M_{EPR}=\lambda\langle x,y\rangle.CNOT(Hx\otimes y)$. M_{EPR} encodes the quantum circuit on two input qubits which has the ability to produce an* entangled *state from any element of*

the underlying computational basis[1]*. It can be given the type* $\mathbb{B}\otimes\mathbb{B}\multimap\mathbb{B}\otimes\mathbb{B}$ *in the empty context. The following is a type derivation* π_{EPR} *for it:*

$$\dfrac{\dfrac{\cdot\vdash CNOT:\mathbb{B}\otimes\mathbb{B}\multimap\mathbb{B}\otimes\mathbb{B} \qquad \dfrac{\dfrac{\cdot\vdash H:\mathbb{B}\multimap\mathbb{B}\quad x:\mathbb{B}\vdash x:\mathbb{B}}{x:\mathbb{B}\vdash Hx:\mathbb{B}}(\mathsf{E}_{\multimap}) \qquad y:\mathbb{B}\vdash y:\mathbb{B}}{x:\mathbb{B},y:\mathbb{B}\vdash Hx\otimes y:\mathbb{B}\otimes\mathbb{B}}(\mathsf{I}_{\otimes})}{x:\mathbb{B},y:\mathbb{B}\vdash CNOT(Hx\otimes y):\mathbb{B}\otimes\mathbb{B}}(\mathsf{E}_{\multimap})}{\cdot\vdash M_{EPR}:\mathbb{B}\otimes\mathbb{B}\multimap\mathbb{B}\otimes\mathbb{B}}(\mathsf{I}^2_{\multimap})$$

M_{EPR} and π_{EPR} will be used as running examples in the rest of this paper, together with the following type derivation ρ_{EPR}:

$$\dfrac{\pi_{EPR}\triangleright\cdot\vdash M_{EPR}:\mathbb{B}\otimes\mathbb{B}\multimap\mathbb{B}\otimes\mathbb{B} \qquad \dfrac{\cdot\vdash|0\rangle_1:\mathbb{B}\quad\cdot\vdash|1\rangle_2:\mathbb{B}}{\cdot\vdash|0\rangle_1\otimes|1\rangle_2:\mathbb{B}\otimes\mathbb{B}}(\mathsf{I}_{\otimes})}{\cdot\vdash M_{EPR}(|0\rangle_1\otimes|1\rangle_2):\mathbb{B}\otimes\mathbb{B}}(\mathsf{E}_{\multimap})$$

If $\pi\triangleright\Gamma\vdash(\lambda x.M)N:A$, one can build a type derivation ρ with conclusion $\Gamma\vdash M\{x/N\}:A$ in a canonical way, and similarly when $\pi\triangleright\Gamma\vdash(\lambda\langle x,y\rangle.M)(N\otimes L):A$. This, as expected, is a consequence of the following:

Lemma 1 (Substitution Lemma) *If $\pi\triangleright\Gamma,x_1:A_1,\ldots,x_n:A_n\vdash M:B$ and for every $1\leq i\leq n$ there is $\rho_i\triangleright\Delta_i\vdash N_i:A_i$, then there is a canonically defined derivation $\pi\{x_1,\ldots,x_n/\rho_1,\ldots,\rho_n\}$ of $\Gamma,\Delta_1,\ldots,\Delta_n\vdash M\{x_1,\ldots,x_n/N_1,\ldots,N_n\}:B$.*

Proof. Just proceed by the usual, simple induction on π. □

2.3 An Equational Theory.

The λ-calculus is usually endowed with notions of reduction or equality, both centred around the β-rule, according to which a function $\lambda x.M$ applied to an argument N *reduces to* (or *can be considered equal to*) the term $M\{N/x\}$ obtained by replacing all free occurrences of x with N. A reduction relation implicitly provides the underlying calculus with a notion of computation, while an equational theory is more akin to a reasoning technique. Giving a reduction relation on $\mathsf{Q}\Lambda$ terms directly, however, is problematic. What happens when a n-ary unitary operator U is faced with an n-tuple of qubits $|b_1\ldots b_n\rangle$? Superposition should somehow arise, but how can we capture it?

In this section, an equational theory for $\mathsf{Q}\Lambda$ will be introduced. In the next sections, we will show that the semantics induced by token machines is *sound* with respect to it. The equational theory we are going to introduce will be a binary relation on formal, weighted sums of $\mathsf{Q}\Lambda$ terms:

Definition 1 (Superposed Term) *A* superposed term *of type* (Γ,A) *is a formal sum*

$$\mathscr{T}=\sum_{i=1}^{n}\kappa_i M_i$$

[1]The quantum circuit EPR is built out from the unitary gates H (the so-called Hadamard gate) and *CNOT*. The unary gate H is able to create a superposition of elements of the computational basis $|0\rangle$ and $|1\rangle$, i.e. a linear combination in the form $\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)$ or $\frac{1}{\sqrt{2}}(|0\rangle-|1\rangle)$. The binary gate *CNOT* negates its second argument, according to the value of the first one. We provide two simple examples of entangled and non-entangled quantum states. The state $|\psi\rangle=\frac{1}{\sqrt{2}}|00\rangle+\frac{1}{\sqrt{2}}|11\rangle$ is entangled whereas any state $\phi=\alpha|00\rangle+\beta|01\rangle$ is not. In fact, it is possible to express the latter in the mathematically equivalent form $\phi=|0\rangle\otimes(\alpha|0\rangle+\beta|1\rangle)$. See [8] for a gentle introduction to quantum computing essential notions.

where for every $1 \leq i \leq n$, $\kappa_i \in \mathbb{C}$ *and there is* π_i *such that that* $\pi_i \triangleright \Gamma \vdash M_i : A$. *In this case, we write* $\Gamma \vdash \mathscr{T} : A$.

Superposed terms will be denoted by metavariables like $\mathscr{T}$ or $\mathscr{S}$. Please observe that terms in a superposed term are *uniformly* typed, i.e., they can be given the same type in the same context. Please, notice that:

- If $\pi \triangleright \cdot \vdash U|b_1 \ldots b_k\rangle : \mathbb{B}^k$, then there is a naturally defined superposed term of type $(\cdot, \mathbb{B}^k)$ which is nothing more than the element of $\mathbb{C}^{2^k}$ obtained by applying $\mathbf{U}$ to the vector $|b_1 \ldots b_k\rangle$ With a slight abuse of notation, this superposed term will be indicated with $\mathbf{U}|b_1 \ldots b_k\rangle$.
- All the term constructs can be generalised to operators on superposed terms, with the proviso that the types match. As an example if $\mathscr{T} = \sum_i \alpha_i M_i$ where $\pi_i \triangleright \Gamma \vdash M_i : A \multimap B$ and $\rho \triangleright \Delta \vdash N : A$, then $\mathscr{T}N$ denotes the superposed term $\mathscr{S} = \sum_i \alpha_i (M_i N)$. Indeed, there exist type derivations $\sigma_i \triangleright \Gamma, \Delta \vdash M_i N : B$ each obtained applying the rule $(\mathsf{E}_{\multimap})$ to π_i and ρ.

It is now time to define our equational theory, which will be defined on superposed terms of the same type. Formally, $\approx$ is a binary relation on superposed terms, indexed by contexts and types. The fact that $\mathscr{T} \approx_{\Gamma,A} \mathscr{S}$ is indicated with $\Gamma \vdash \mathscr{T} \approx \mathscr{S} : A$. The relation $\approx$ is defined inductively, by the rules in Figure 2. Notice that for each Γ, A, $\approx_{\Gamma,A}$ is by construction an equivalence relation.

Example 2 *As an example, consider the term* $M_{EPR}(|0\rangle_1 \otimes |1\rangle_2)$ *from Example 1. The equations in the following chain can all be derived through axioms and context closure rules:*

$$M_{EPR}(|0\rangle \otimes |1\rangle) \approx CNOT(H|0\rangle \otimes |1\rangle) \approx \frac{1}{\sqrt{2}} CNOT(|0\rangle \otimes |1\rangle) + \frac{1}{\sqrt{2}} CNOT(|1\rangle \otimes |1\rangle)$$
$$\approx \frac{1}{\sqrt{2}}|0\rangle \otimes |1\rangle + \frac{1}{\sqrt{2}} CNOT(|1\rangle \otimes |1\rangle) \approx \frac{1}{\sqrt{2}}|0\rangle \otimes |1\rangle + \frac{1}{\sqrt{2}}|1\rangle \otimes |0\rangle.$$

The context (which is $\cdot$*) and the type (which is* $\mathbb{B}^2$*) have been elided for the sake of readability. By rule* trans*, we can derive that*

$$\cdot \vdash EPR(|0\rangle \otimes |1\rangle) \approx \frac{1}{\sqrt{2}}|0\rangle \otimes |1\rangle + \frac{1}{\sqrt{2}}|1\rangle \otimes |0\rangle : \mathbb{B}^2.$$

In other words, EPR, when fed with $|0\rangle \otimes |1\rangle$*, produces an entangled pair of qubits. The fourth superposed term in the chain above has the remarkable property of not being* homogeneous*, i.e., of being the sum of two terms which are not identical up to the value of bit constants.*

Please observe that the equational theory we have just defined can *hardly* be seen as an operational semantics for $\mathsf{Q}\Lambda$. Although equations can of course be oriented, it is the very nature of a superposed term which is in principle problematic from the point of view of quantum computation: what is the mathematical nature of a superposed term? Is it an element of an Hilbert Space? And if so, of *which one*? If we consider a simple language such as $\mathsf{Q}\Lambda$, the questions above may appear overly rhetorical, but we claim they are not. For example, what would be the quantum meaning of linear beta-reduction? If we want to design beta-reduction according to the principles of quantum computation, it has to be, at least, easily reversible (unless measurement is implicit in it). Moving towards more expressive languages, this non-trivial issue becomes more difficult and a number of constraints have to be imposed (for example, superposition of terms can be allowed, but only between homogenous terms [19]). This is the reason for which promising calculi [19] fail to be canonical models for quantum programming languages. This issue has been faced in literature without satisfactory answers, yielding a number of convincing arguments in favour of the (implicit or explicit) classical control of quantum data [5, 16]. As observed above, our equational theory permits non-homogeneous superposed terms in a very natural way.

Axioms

$$\frac{\Gamma \vdash (\lambda\langle x,y\rangle.M)(N\otimes L) : A}{\Gamma \vdash (\lambda\langle x,y\rangle.M)(N\otimes L) \approx M\{x,y/N,L\} : A}\ \mathsf{beta.pair}$$

$$\frac{\Gamma \vdash (\lambda x.M)N : A}{\Gamma \vdash (\lambda x.M)N \approx M\{x/N\} : A}\ \mathsf{beta} \qquad \frac{\cdot \vdash U|b_1\ldots b_k\rangle : \mathbb{B}^k}{\cdot \vdash U|b_1\ldots b_k\rangle \approx \mathbf{U}|b_1\ldots b_k\rangle : \mathbb{B}^k}\ \mathsf{quant}$$

Context Closure

$$\frac{\Gamma \vdash \mathscr{T} \approx \mathscr{S} : A \multimap B \quad \Delta \vdash M : A}{\Gamma,\Delta \vdash \mathscr{T}M \approx \mathscr{S}M : B}\ \mathsf{l.a} \qquad \frac{\Gamma \vdash M : A \multimap B \quad \Delta \vdash \mathscr{T} \approx \mathscr{S} : A}{\Gamma,\Delta \vdash M\mathscr{T} \approx M\mathscr{S} : B}\ \mathsf{r.a} \qquad \frac{\Gamma,x:A \vdash \mathscr{T} \approx \mathscr{S} : B}{\Gamma \vdash \lambda x.\mathscr{T} \approx \lambda x.\mathscr{S} : A \multimap B}\ \mathsf{in.}\lambda$$

$$\frac{\Gamma,x:A,y:B \vdash \mathscr{T} \approx \mathscr{S} : C}{\Gamma \vdash \lambda\langle x,y\rangle.\mathscr{T} \approx \lambda\langle x,y\rangle.\mathscr{S} : A\otimes B \multimap C}\ \mathsf{in.}\lambda\mathsf{.pair} \qquad \frac{\Gamma \vdash \mathscr{T} \approx \mathscr{S} : A \quad \Delta \vdash M : B}{\Gamma,\Delta \vdash \mathscr{T}\otimes M \approx \mathscr{S}\otimes M : A\otimes B}\ \mathsf{l.in.tens}$$

$$\frac{\Gamma \vdash M : A \quad \Delta \vdash \mathscr{T} \approx \mathscr{S} : B}{\Gamma,\Delta \vdash M\otimes\mathscr{T} \approx M\otimes\mathscr{S} : A\otimes B}\ \mathsf{r.in.tens} \qquad \frac{\Gamma \vdash \mathscr{T} \approx \mathscr{S} : A \quad \Gamma \vdash \mathscr{V} : A}{\Gamma \vdash \alpha\mathscr{T}+\mathscr{V} \approx \alpha\mathscr{S}+\mathscr{V} : A}\ \mathsf{sum}$$

Reflexive, Symmetric and Transitive Closure

$$\frac{\Gamma \vdash \mathscr{T} : A}{\Gamma \vdash \mathscr{T} \approx \mathscr{T} : A}\ \mathsf{refl} \qquad \frac{\Gamma \vdash \mathscr{T} \approx \mathscr{S} : A}{\Gamma \vdash \mathscr{S} \approx \mathscr{T} : A}\ \mathsf{sym} \qquad \frac{\Gamma \vdash \mathscr{T} \approx \mathscr{S} : A \quad \Gamma \vdash \mathscr{S} \approx \mathscr{V} : A}{\Gamma \vdash \mathscr{T} \approx \mathscr{V} : A}\ \mathsf{trans}$$

Figure 2: Equational Theory

3 A Token Machine for $\mathsf{Q}\Lambda$

In this section we describe an interpretation of $\mathsf{Q}\Lambda$ type derivations in terms of a specific token machine called $\mathsf{IAM}_{\mathsf{Q}\Lambda}$. Before formally defining $\mathsf{IAM}_{\mathsf{Q}\Lambda}$, it is necessary to give some preliminary concepts.

With a slight abuse of notation, a permutation $\sigma : \{1,\ldots,n\} \to \{1,\ldots,n\}$ will be often applied to sequences of length n with the obvious meaning: $\sigma(a_1,\ldots,a_n) = a_{\sigma(1)},\ldots,a_{\sigma(n)}$. Similarly, such a permutation can be seen as the *unique* unitary operator on $\mathbb{C}^{2^n}$ which sends $|b_1\cdots b_n\rangle$ to $|b_{\sigma(1)}\cdots b_{\sigma(n)}\rangle$.

Suppose given a unitary operator $\mathbf{U} \in \mathscr{U}$ of arity $n \in \mathbb{N}$. Now, take a natural number $m \geq n$ and n distinct natural numbers $j_1,\ldots,j_n$, all of them smaller or equal to m. With $\mathbf{U}_m^{j_1,\ldots,j_n}$ (or simply with $\mathbf{U}^{j_1,\ldots,j_n}$) we indicate the unitary operator of arity m which acts like $\mathbf{U}$ on the quantum bits indexed with $j_1,\ldots,j_n$ and leave all the other qubits unchanged. In the following, with a slight abuse of notation, occurrences of types in type derivations are confused with types themselves. On the other hand, occurrences of types *inside other types* will be defined quite precisely, as follows.

Contexts (types with a hole) are denoted by metavariables like C or D. A context C is said to be *a context for a type A* if $C[\mathbb{B}] = A$. *Negative contexts* (i.e., contexts where the hole is in negative position) are denoted by metavariables like N,M. *Positive* ones are denoted by metavariables like P,Q. An *occurrence* of $\mathbb{B}$ in the type derivation π is a pair $O = (A,C)$, where A is an occurrence of a type in π and C is a context for A. Sequences of occurrences are indicated with metavariables like φ,ψ (possibly indexed). All sequences of occurrences we will deal with do not contain duplicates. Type constructors $\multimap$ and $\otimes$ can be generalised to operators on occurrences and sequences of occurrences, e.g. $(A,C) \multimap B$ is just $(A \multimap B, C \multimap B)$. If a sequence of occurrences φ contains the occurrences $O_1,\ldots,O_n$, we emphasise it by indicating it with $\varphi(O_1,\ldots,O_n)$.

Given (an occurrence of) a type A, all positive and negative occurrences of $\mathbb{B}$ inside A form sequences called $\mathscr{P}(A)$ and $\mathscr{N}(A)$, respectively. These are defined as follows (where $\cdot$ is sequence concatenation):

$$\begin{aligned}
\mathscr{P}(\mathbb{B}) &= (\mathbb{B},[\cdot]);\\
\mathscr{N}(\mathbb{B}) &= \varepsilon;\\
\mathscr{P}(A\otimes B) &= (\mathscr{P}(A)\otimes B)\cdot(A\otimes\mathscr{P}(B));\\
\mathscr{N}(A\otimes B) &= (\mathscr{N}(A)\otimes B)\cdot(A\otimes\mathscr{N}(B));\\
\mathscr{P}(A\multimap B) &= (\mathscr{N}(A)\multimap B)\cdot(A\multimap\mathscr{P}(B));\\
\mathscr{N}(A\multimap B) &= (\mathscr{P}(A)\multimap B)\cdot(A\multimap\mathscr{N}(B)).
\end{aligned}$$

Example 3 *As an example, the positive occurrences in the type $\mathbb{B}\multimap\mathbb{B}\otimes\mathbb{B}$ should be the two rightmost ones. And, indeed,*

$$\begin{aligned}
\mathscr{P}(\mathbb{B}\multimap\mathbb{B}\otimes\mathbb{B}) &= (\mathscr{N}(\mathbb{B})\multimap\mathbb{B}\otimes\mathbb{B})\cdot(\mathbb{B}\multimap\mathscr{P}(\mathbb{B}\otimes\mathbb{B}))\\
&= \varepsilon\cdot(\mathbb{B}\multimap\mathscr{P}(\mathbb{B}\otimes\mathbb{B}))\\
&= \mathbb{B}\multimap\mathscr{P}(\mathbb{B}\otimes\mathbb{B})\\
&= (\mathbb{B}\multimap(\mathscr{P}(\mathbb{B})\otimes\mathbb{B}))\cdot(\mathbb{B}\multimap(\mathbb{B}\otimes\mathscr{P}(\mathbb{B})))\\
&= (\mathbb{B},\mathbb{B}\multimap([\cdot]\otimes\mathbb{B})),(\mathbb{B},\mathbb{B}\multimap(\mathbb{B}\otimes[\cdot])).
\end{aligned}$$

Similarly, one can prove that $\mathscr{N}(\mathbb{B}\multimap\mathbb{B}\otimes\mathbb{B}) = (\mathbb{B},[\cdot]\multimap(\mathbb{B}\otimes\mathbb{B}))$.

For every type derivation π, $\mathscr{B}(\pi)$ is the sequence of all these occurrences of $\mathbb{B}$ in π which are introduced by the rules $(\mathsf{a_{q0}})$ and $(\mathsf{a_{q1}})$ (recall Figure 1). Similarly, $\mathscr{V}(\pi)$ is the corresponding sequence of binary

digits, seen as a vector in $\mathbb{C}^{2^{|\mathscr{B}(\pi)|}}$. Both in $\mathscr{B}(\pi)$ and in $\mathscr{V}(\pi)$, the order is the one induced by the natural number labeling the underlying bit in π.

Example 4 *Consider the following type derivation, and call it τ:*

$$\frac{\cdot \vdash |0\rangle_2 : \mathbb{B}_1 \quad \cdot \vdash |1\rangle_1 : \mathbb{B}_2}{\cdot \vdash |0\rangle_2 \otimes |1\rangle_1 : \mathbb{B}_3 \otimes \mathbb{B}_4}\,(\mathsf{I}_\otimes)$$

There are four occurrences of $\mathbb{B}$ in it, and we have indexed it with the first four positive natural numbers, just to be able to point at them without being forced to use the formal, context machinery. Only two of them, namely the upper ones, are introduced by instances of the rules $(\mathsf{a}_{\mathsf{q}0})$ and $(\mathsf{a}_{\mathsf{q}1})$. Moreover, the rightmost one serves to type a bit having an index (namely 1) greater than the one in the other instance (namely 2). As a consequence, $\mathscr{B}(\tau)$ is the sequence $\mathbb{B}_2,\mathbb{B}_1$. The two instances introduces bits 0 and 1; then $\mathscr{V}(\pi) = |1\rangle \otimes |0\rangle$. As another example, one can easily compute $\mathscr{B}(\pi_{EPR})$ and $\mathscr{V}(\pi_{EPR})$ (where π_{EPR} is from Example 1), finding out that both are the empty sequence.

We are finally able to define, for every type derivation π, the abstract machine $\mathscr{A}_\pi$ interpreting it:

- The *states* of $\mathscr{A}_\pi$ form a set $\mathscr{S}_\pi$ and are in the form $\mathsf{S} = (O_1,\ldots,O_n,\mathbf{Q})$ where:
  - $O_1,\ldots,O_n$ are occurrences of the type $\mathbb{B}$ in π;
  - $\mathbf{Q}$ is a *quantum register* on n quantum bits, i.e. a normalised vector in $\mathbb{C}^{2^n}$.
- The *transition relation* $\rightarrow_\pi \subseteq \mathscr{S}_\pi \times \mathscr{S}_\pi$ is defined based on π, following the rules from Figure 3. In the last rule, $\mathbb{B}$ in the type of U is simply denoted through its index, and for every $1 \leq k \leq m$, i_k is the position of $\mathbb{B}_k$ in the sequence φ. The transition rules induced by $(\mathsf{I}^2_{\multimap})$ have been elided for the sake of simplicity (see [8]).

The number of positive (negative, respectively) occurrences of $\mathbb{B}$ in the conclusion of π is said to be the *output arity* (the *input arity*, respectively) of π. Suppose, for the sake of simplicity, that π is a type derivation of $\cdot \vdash M : A$. An *initial state* for a quantum register $\mathbf{Q}$ is a state in the form $(\mathscr{N}(A)\cdot\mathscr{B}(\pi),\mathbf{Q}\otimes\mathscr{V}(\pi))$. Given a permutation σ on n elements, a *final state* for a quantum register $\mathbf{Q}$ *and* σ is a state in the form $(\varphi,\mathbf{Q})$, where $\varphi = \sigma(\mathscr{P}(A))$. A *run* of $\mathscr{A}_\pi$ is simply a finite or infinite sequence $\mathsf{S}_1,\mathsf{S}_2,\ldots$ of states from $\mathscr{S}_\pi$ such that $\mathsf{S}_i \rightarrow_\pi \mathsf{S}_{i+1}$ for every i.

Example 5 (A run of $\mathsf{IAM}_{\mathsf{Q}\Lambda}$) *Consider the term M_{EPR} and its type derivation π_{EPR} (see Example 1). Forgetting about terms and marking different occurrences of $\mathbb{B}$ with distinct indices, we obtain:*

$$\dfrac{\dfrac{\dfrac{\cdot \vdash \mathbb{B}_9 \otimes \mathbb{B}_{10} \multimap \mathbb{B}_{11} \otimes \mathbb{B}_{12} \qquad \dfrac{\dfrac{\cdot \vdash : \mathbb{B}_{21} \multimap \mathbb{B}_{22} \quad \mathbb{B}_{23} \vdash \mathbb{B}_{24}}{\mathbb{B}_{17} \vdash \mathbb{B}_{18}}\,(\mathsf{E}_{\multimap}) \qquad \mathbb{B}_{19} \vdash \mathbb{B}_{20}}{\mathbb{B}_{13},\mathbb{B}_{14} \vdash \mathbb{B}_{15} \otimes \mathbb{B}_{16}}\,(\mathsf{I}_\otimes)}{\mathbb{B}_5,\mathbb{B}_6 \vdash \mathbb{B}_7 \otimes \mathbb{B}_8}\,(\mathsf{E}_{\multimap})}{\cdot \vdash \mathbb{B}_1 \otimes \mathbb{B}_2 \multimap \mathbb{B}_3 \otimes \mathbb{B}_4}\,(\mathsf{I}^2_{\multimap})$$

Let us consider the following computation of $\mathscr{A}_{\pi_{EPR}}$:

$$\begin{aligned}
&(\mathbb{B}_1,\mathbb{B}_2,\mathbf{Q}) \rightarrow^*_\pi (\mathbb{B}_5,\mathbb{B}_6,\mathbf{Q}) \rightarrow^*_\pi (\mathbb{B}_{13},\mathbb{B}_{14},\mathbf{Q}) \rightarrow_\pi (\mathbb{B}_{17},\mathbb{B}_{19},\mathbf{Q}) \rightarrow^*_\pi (\mathbb{B}_{23},\mathbb{B}_{20},\mathbf{Q})\\
&\quad \rightarrow^*_\pi (\mathbb{B}_{24},\mathbb{B}_{16}) \rightarrow_\pi (\mathbb{B}_{24},\mathbb{B}_{10},\mathbf{Q}) \rightarrow_\pi (\mathbb{B}_{21},\mathbb{B}_{10},\mathbf{Q}) \rightarrow_\pi (\mathbb{B}_{22},\mathbb{B}_{10},\mathbf{H}^1(\mathbf{Q}))\\
&\quad \rightarrow_\pi (\mathbb{B}_{18},\mathbb{B}_{10},\mathbf{H}^1(\mathbf{Q})) \rightarrow_\pi (\mathbb{B}_{15},\mathbb{B}_{10},\mathbf{H}^1(\mathbf{Q})) \rightarrow_\pi (\mathbb{B}_9,\mathbb{B}_{10},\mathbf{H}^1(\mathbf{Q}))\\
&\quad \rightarrow_\pi (\mathbb{B}_{11},\mathbb{B}_{12},\mathbf{CNOT}^{1,2}(\mathbf{H}^1(\mathbf{Q}))) \rightarrow^*_\pi (\mathbb{B}_7,\mathbb{B}_8,\mathbf{CNOT}^{1,2}(\mathbf{H}^1(\mathbf{Q})))\\
&\quad \rightarrow_\pi (\mathbb{B}_3,\mathbb{B}_4,\mathbf{CNOT}^{1,2}(\mathbf{H}^1(\mathbf{Q}))).
\end{aligned}$$

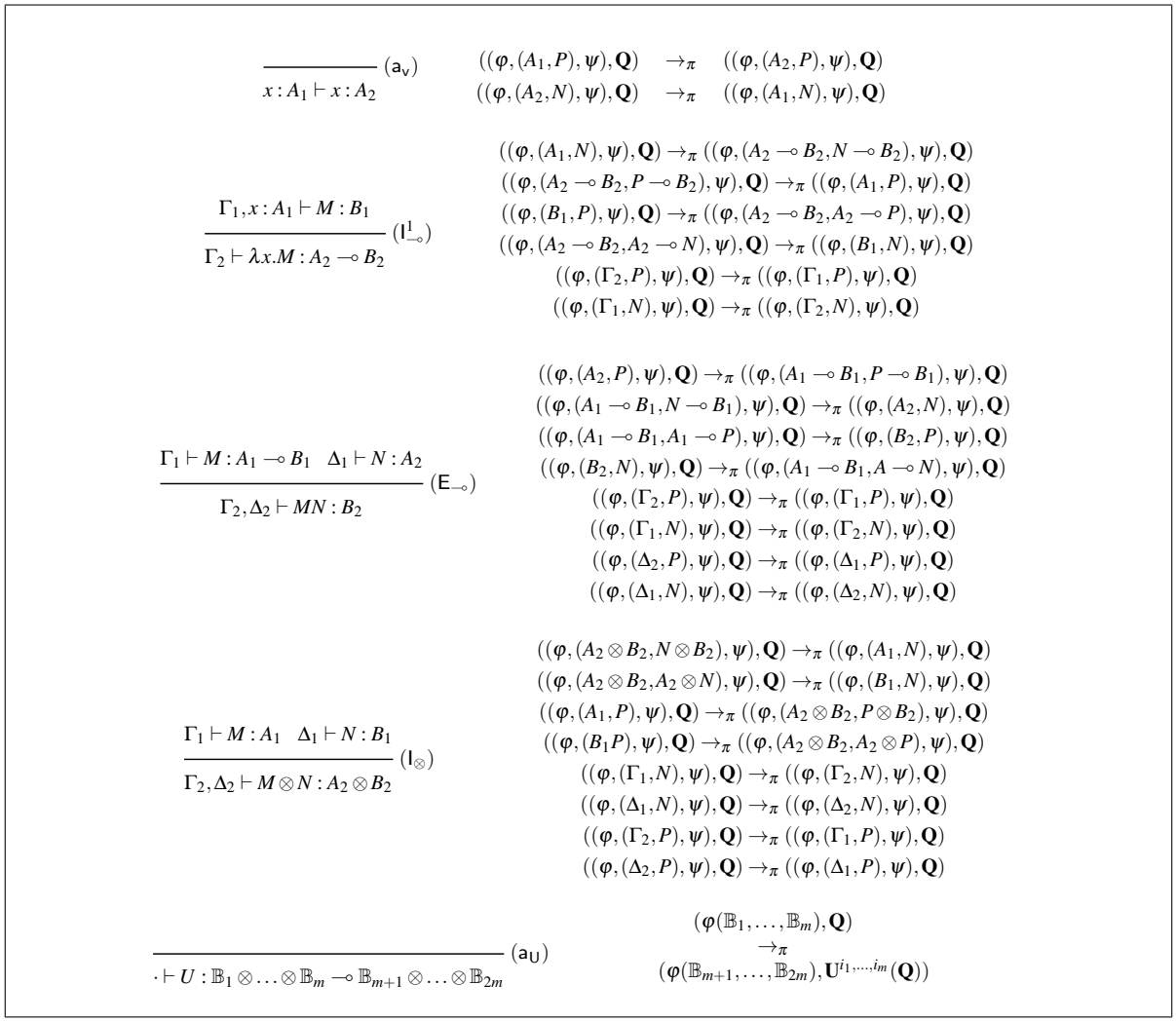

Figure 3: $\mathsf{IAM}_{\mathsf{Q}\Lambda}$ Transition Rules

Notice that the occurrence of CNOT acts as a synchronisation operator: the second token is stuck at the occurrence $\mathbb{B}_{10}$ until the first token arrives (from the occurrence $\mathbb{B}_{15}$) as a control input of the CNOT and the corresponding reduction step actually occurs.

What the example above shows, indeed, is that the presence of a potential *entanglement* in π is intimately related to the necessity of *synchronisation* in the underlying machine $\mathscr{A}_\pi$: if all unitary operators in π can be expressed as the tensor product of unitary operators of arity one (and, thus, entanglement is not possible), then synchronisation is simply not necessary.

Given a type derivation π, the relation $\rightarrow_\pi$ enjoys a strong form of confluence:

Proposition 1 (One-step Confluence of $\rightarrow_\pi$) *Let $\mathsf{S},\mathsf{R},\mathsf{T} \in \mathscr{S}_\pi$ be such that $\mathsf{S} \rightarrow_\pi \mathsf{R}$ and $\mathsf{S} \rightarrow_\pi \mathsf{T}$. Then either $\mathsf{R} = \mathsf{T}$ or there exists a state U such that $\mathsf{R} \rightarrow_\pi \mathsf{U}$ and $\mathsf{T} \rightarrow_\pi \mathsf{U}$.*

Proof. By simply inspecting the various rules. Notice that there are no critical pairs in $\rightarrow_\pi$. □

The way $\mathscr{A}_\pi$ is built by following a type derivation π induces the following notion:

Definition 2 *Given a type derivation π, the partial function* computed by *π is denoted as $[\pi]$, has domain $\mathbb{C}^{2^n}$ and codomain $\mathbb{C}^{2^m}$ (where n and m are the input and output arity of π) and is defined by stipulating that $[\pi](\mathbf{Q}) = \mathbf{R}$ iff any initial state for $\mathbf{Q}$ rewrites into a final state for $\mathbf{S}$ and σ, where $\mathbf{S} = \sigma^{-1}(\mathbf{R})$.*

Given a type derivation π, $[\pi]$ is either always undefined or always defined. Indeed, the fact any initial configuration (for, say, $\mathbf{Q}$) rewrites to a final configuration or not does *not* depend on $\mathbf{Q}$ but only on π:

Lemma 2 (Uniformity) *For every type derivation π and for every occurrences $O_1,\ldots,O_n$, $P_1,\ldots,P_n$, there is a unitary operator $\mathbf{U}$ such that whenever $(O_1,\ldots,O_n,\mathbf{Q}) \rightarrow_\pi (P_1,\ldots,P_n,\mathbf{R})$ it holds that $\mathbf{R} = \mathbf{U}(\mathbf{Q})$.*

Proof. Observe that for every $O_1,\ldots,O_n$, $P_1,\ldots,P_n$ there is *at most* one of the rules defining $\rightarrow_\pi$ which can be applied. Moreover, notice that each rule acts uniformly on the underlying quantum register. □

In the following section, we will prove that $[\pi]$ is always a *total* function, and that it makes perfect sense from a quantum point of view.

4 Main Properties of $\mathsf{IAM}_{\mathsf{Q}\Lambda}$

In this section, we will give some crucial results about $\mathsf{IAM}_{\mathsf{Q}\Lambda}$. More specifically, we prove that runs of this token machine are indeed finite and end in final states. We proceed by putting $\mathsf{Q}\Lambda$ in correspondence to MLL, inheriting its very elegant proof theory and token machines.

4.1 A Correspondence Between MLL and $\mathsf{Q}\Lambda$

Let $\mathbb{A} = \{\alpha,\beta,\ldots\}$ be a countable set of *propositional atoms*. A *formula A* of Multiplicative Linear Logic (MLL) is given by the following grammar:

$$A,B ::= \alpha \;\mid\; \alpha^\perp \;\mid\; A \otimes B \;\mid\; A \parr B.$$

Linear negation can be extended to all formulas in the usual way:

$$(\alpha^\perp)^\perp = \alpha; \qquad A \otimes B^\perp = A^\perp \parr B^\perp; \qquad A \parr B^\perp = A^\perp \otimes B^\perp.$$

This way, $A^{\perp\perp}$ is just A. The one-sided sequent calculus for MLL is very simple:

$$\frac{}{\vdash A, A^{\perp}}\,\mathsf{ax} \qquad \frac{\vdash \Gamma, A \quad \vdash \Delta, A^{\perp}}{\vdash \Gamma, \Delta}\,\mathsf{cut} \qquad \frac{\vdash \Gamma, A \quad \vdash \Delta, B}{\vdash \Gamma, \Delta, A \otimes B}\otimes \qquad \frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \parr B}\parr$$

The logic MLL enjoys cut-elimination: there is a terminating algorithm turning any MLL proof into a cut-free proof of the same conclusion. A notion of *structural equivalence* between two MLL proofs ξ, μ having the same conclusion $\vdash \Gamma$ can be easily defined and holds only if ξ and μ are essentially *the same* proof modulo renaming of the formulas occurring in ξ and μ. Remarkably, two MLL proofs which are structurally equivalent are actually the *same* proof, a result which does not hold in more expressive logics like MELL. More details on that can be found in [8].

Any $\mathsf{Q}\Lambda$ type derivation π can be put in correspondence with *some* MLL proofs. We inductively define the map $(\cdot)^{\bullet}$ from $\mathsf{Q}\Lambda$ types to MLL formulas as follows:

$$(\mathbb{B})^{\bullet} = \alpha; \qquad (A \multimap B)^{\bullet} = (A)^{\bullet\perp} \parr (B)^{\bullet}; \qquad (A \otimes B)^{\bullet} = (A)^{\bullet} \otimes (B)^{\bullet}$$

Given a judgment $J = \Gamma \vdash M : A$ and a natural number $n \in \mathbb{N}$, the MLL sequent *corresponding* to J and n is the following one:

$$\vdash \underbrace{\alpha^{\perp}, \ldots, \alpha^{\perp}}_{n \text{ times}}, ((B_1)^{\bullet})^{\perp}, \ldots, ((B_m)^{\bullet})^{\perp}, (A)^{\bullet},$$

where $\Gamma = x_1 : B_1, \ldots, x_m : B_m$. For every π, we define now a set of MLL proofs $\mathscr{I}(\pi)$. This way, every type derivation π for $J = \Gamma \vdash M : A$ such that n bits occur in M, is put in relation to possibly many MLL proofs of the sequent corresponding to J and n. One among them is called the *canonical proof* for π. The set $\mathscr{I}(\pi)$ and canonical proofs are defined by induction on the structure of the underlying type derivation π. The type constructions of $\mathsf{Q}\Lambda$ are mapped to the corresponding MLL logical operators, rules $(\mathsf{a_{q0}})$ and $(\mathsf{a_{q1}})$ are mapped to axioms, and rule $(\mathsf{a_U})$ is mapped to a proof encoding a permutation of the involved atoms. When the latter is the identity, we get the canonical proof for π. For more details, please refer to [8].

Given an MLL proof ξ, let us denote as T_{ξ} the class of all finite sequences of atom occurrences in ξ. The relation $\mapsto_{\xi}$ can be extended to a relation on T_{ξ} by stipulating that

$$(O_1, \ldots, O_{n-1}, P, O_{n+1}, \ldots, O_m) \mapsto_{\xi} (O_1, \ldots, O_{n-1}, R, O_{n+1}, \ldots, O_m)$$

whenever $P \mapsto_{\xi} R$. As usual, $\mapsto_{\xi}^{+}$ is the transitive closure of $\mapsto_{\xi}$.

Let us now consider a type derivation π in $\mathsf{Q}\Lambda$, its quantum token machine $\mathscr{A}_{\pi}$, and any $\xi \in \mathscr{I}(\pi)$. States of $\mathscr{A}_{\pi}$ can be mapped to T_{ξ} by simply forgetting the underlying quantum register and mapping any occurrence of π to the corresponding atom occurrence in ξ. This way one gets a map

$$\mathscr{R}_{\pi,\xi} : \mathscr{S}_{\pi} \to \mathsf{T}_{\xi}$$

such that, given a state $\mathsf{S} = (O_1, \ldots, O_n, \mathbf{Q})$ in $\mathscr{S}_{\pi}$, $|\mathscr{R}_{\pi,\xi}(\mathsf{S})| = n$, i.e., the number of occurrences in S is the same as the length of $\mathscr{R}_{\pi,\xi}(\mathsf{S})$. Each reduction step on the token machine $\mathscr{A}_{\pi}$ corresponds to *at least one* reduction step in the MLL machine $\mathscr{M}_{\xi}$, where $\xi \in \mathscr{I}(\pi)$ is the canonical proof:

Lemma 3 *Let us consider a token machine $\mathscr{A}_{\pi}$ and two states $\mathsf{S}, \mathsf{R} \in \mathscr{S}_{\pi}$. If $\mathsf{S} \to_{\pi} \mathsf{R}$ and $\xi \in \mathscr{I}(\pi)$ is canonical, then $\mathscr{R}_{\pi,\xi}(\mathsf{S}) \mapsto_{\xi}^{+} \mathscr{R}_{\pi,\xi}(\mathsf{R})$.*

Proof. This goes by induction on the structure of π. □

Any (possible) pathological situation on the quantum token machine, then, can be brought back to a corresponding (absurd) pathological situation in the MLL token machine. This is the principle that will guide us in the rest of this section.

4.2 Termination, Progress and Soundness

The first property we want to be sure about is that every computation of any token machine $\mathscr{A}_\pi$ always terminates. The second one is progress (i.e. deadlock-freedom). In both cases, we use in an essential way the correspondence between $\mathsf{Q}\Lambda$ and MLL.

Proposition 2 (Termination) *For any quantum token machine $\mathscr{A}_\pi$, any sequence $\mathsf{S} \to_\pi \mathsf{R} \to_\pi \ldots$ is finite.*

Proof. Suppose, for the sake of contradiction, than there exists an infinite computation in $\mathscr{A}_\pi$. This implies by Lemma 3 that there exists an infinite path in the token machine $\mathscr{M}_\xi$ where ξ is the canonical MLL proof for π. This is a contradiction, because paths in MLL proofs are well-known to be always finite. □

Progress (i.e. deadlock-freedom) is more difficult to prove than termination. Given a type derivation π, an *argument occurrence* is any negative occurrence (A,N) of $\mathbb{B}$ in a (a_U) axiom. We extend this definition to the corresponding atom occurrence when $\xi \in \mathscr{I}(\pi)$. A *result occurrence* is defined similarly, but the occurrence has to be positive.

Proposition 3 (Progress) *Suppose π is a type derivation in $\mathsf{Q}\Lambda$ and $\mathsf{S} \in \mathscr{S}_\pi$ is initial. Moreover, suppose that $\mathsf{S} \to_\pi^* \mathsf{R}$. Then either R is final or $\mathsf{R} \to_\pi \mathsf{T}$ for some $\mathsf{T} \in \mathscr{S}_\pi$.*

Proof. Let us consider a computation $\mathsf{S}_1 \to_\pi \ldots \to_\pi \mathsf{S}_k$ on a quantum token machine $\mathscr{A}_\pi$. Suppose that the state S_k is a deadlocked state, i.e. S_k is not a final state, and that there exists no S_m such that $\mathsf{S}_k \to_\pi \mathsf{S}_m$. The fact S_k is a deadlocked state means that $l \geq 1$ occurrences in S_k are argument occurrences, since the latter are the only points of synchronisation of the machine. Let us consider any *maximal* sequence

$$\mathscr{R}_{\pi,\xi}(\mathsf{S}_1) \mapsto_\xi \ldots \mapsto_\xi \mathscr{R}_{\pi,\xi}(\mathsf{S}_k) \mapsto_\xi \mathsf{Q}_1 \mapsto_\xi \ldots \mapsto_\xi \mathsf{Q}_n, \tag{1}$$

where $\xi \in \mathscr{I}(\pi)$ is the canonical proof corresponding to π. Observe that in (1), all occurrences of atoms in ξ are visited exactly once, including those corresponding to argument and result occurrences from π. Notice, however, that the argument and result occurrences of the unitary operators affected by S_k cannot have been visited along the subsequence $\mathscr{R}_{\pi,\xi}(\mathsf{S}_1) \mapsto_\xi \ldots \mapsto_\xi \mathscr{R}_{\pi,\xi}(\mathsf{S}_k)$ (otherwise we would visit the occurrences in S_k at least twice, which is not possible). Now, form a directed graph whose nodes are the unitary constants $U_1,\ldots,U_h$ which block S_k, plus a node F (representing the conclusion of π), and whose edges are defined as follows:

- there is an edge from U_i to U_j iff along $\mathsf{Q}_1 \mapsto_\xi \ldots \mapsto_\xi \mathsf{Q}_n$ one of the l independent computations corresponding to a blocked occurrence in S_k is such that a result occurrence of U_i is followed by an argument occurrence of U_j and the occurrences between them are neither argument nor result occurrences.
- there is an edge from U_i to F iff along $\mathsf{Q}_1 \mapsto_\xi \ldots \mapsto_\xi \mathsf{Q}_n$ one of the l traces is such that a result occurrence of U_i is followed by a final occurrence of an atom and the occurrences between them are neither argument nor result occurrences.

The thus obtained graph has the following properties:

- Every node U_i has at least one incoming edge, because otherwise the configuration S_k would not be deadlocked.
- As a consequence, the graph must be cyclic, because otherwise we could topologically sort it and get a node with no incoming edges (meaning that some of the U_i would not be blocked!). Moreover, the cycle does not include F, because the latter only has incoming nodes.

From any cycle involving the U_j, one can induce the presence of a cycle in the token machine $\mathscr{M}_\mu$ for some $\mu \in \mathscr{I}(\pi)$. Indeed, such a μ can be formed by simply choosing, for each U_j, the "good" permutation, namely the one linking the incoming edge and the outgoing edge which are part of the cycle. This way, we have reached the absurd starting from the existence of a deadlocked computation. □

The immediate consequence of the termination and progress results is that $[\pi]$ is always a *total* function. The way $\mathscr{A}_\pi$ is defined ensures that $[\pi]$ is obtained by feeding some of the inputs of a unitary operator $\mathbf{U}$ with some bits (namely those occurring in π). $\mathbf{U}$ is itself obtained by composing the unitary operators occurring in π, which can thus be seen as a program computing a quantum circuit. In a way, then, token machines both show that $\mathsf{Q}\Lambda$ *is a truly quantum calculus* and can be seen as the right operational semantics for it.

The last step consists in understanding the relation between token machines and the equational theory on superposed terms introduced in Section 2.3. First of all, observe that $\mathscr{T} = \sum_{i=1}^{n} \kappa_i M_i$ has type A in the context Γ, then $M_1, \ldots, M_n$ all have type A in the context Γ. But there is more to that: for every $1 \leq i \leq n$, there is *exactly one* type derivation $\pi_i \triangleright \Gamma \vdash M_i : A$. This holds because two such type derivations π_i and ρ_i are such that the canonical proofs in $\mathscr{I}(\pi_i)$ and $\mathscr{I}(\rho_i)$ are structurally equivalent, thus identical. It is then possible to extend the definition of $[\cdot]$ to superposed terms: if $\mathscr{T} = \sum_{i=1}^{n} \kappa_i M_i$ has type A in Γ, then $[\mathscr{T}]$, when fed with a vector x, returns $\sum_{i=1}^{n} \kappa_i [\pi_i](x)$, where π_i is the *unique* derivation giving type A to M_i in the context Γ. Remarkably, token machines behave in accordance to the equational theory: this is our Soundness Theorem.

Theorem 1 (Soundness) *Given $\mathscr{T}$ and $\mathscr{S}$ superposed terms, if $\Gamma \vdash \mathscr{T} \approx \mathscr{S} : A$, then $[\mathscr{T}] = [\mathscr{S}]$.*

Proof. We only give a sketch of the proof. More details can been found in [8]. The first step consists in proving that any derivation of $\Gamma \vdash \mathscr{T} \approx \mathscr{S} : A$ can be put in *normal form*, a concept defined by giving an order on the rules in Figure 2. More specifically, define the following two sets of rules:

$$\mathsf{AX} = \{\mathsf{beta}, \mathsf{beta.pair}, \mathsf{quant}\};$$
$$\mathsf{CC} = \{\mathsf{l.a}, \mathsf{r.a}, \mathsf{in}.\lambda, \mathsf{in}.\lambda.\mathsf{pair}, \mathsf{l.in.tens}, \mathsf{r.in.tens}\}.$$

A derivation of $\Gamma \vdash \mathscr{T} \approx \mathscr{S} : A$ is said to be *in normal form* (and we write $\Gamma \vdash \mathscr{T} \sim \mathscr{S} : A$) iff

- either the derivation is obtained by applying rule refl;
- or any branch in the derivation consists in instances of rules from AX, possibly followed by instances of rules in CC, possibly followed by instances of sum, possibly followed by instances of sym possibly followed by instances of trans.

In other words, a derivation of $\Gamma \vdash \mathscr{T} \approx \mathscr{S} : A$ is in normal form iff rules are applied in a certain order. As an example, we cannot apply transitivity or symmetry closure rules too early, i.e., before context closure rules. One may wonder whether this restricts the class of provable equivalences. Infact it does not: $\Gamma\Gamma \mathscr{T} \approx \mathscr{S} : A$ iff $\Gamma\Gamma \mathscr{T} \sim \mathscr{S} : A$, a result which is not particularly deep although a bit tedious to prove [8]. Once we have this result in our hands, however, proving Soundness becomes much easier, since the difficult and problematic rules, namely those in CC, are applied to superposed terms of a very specific shape, namely those obtained through AX. □

5 Related Work

In [13], a geometry of interaction model for Selinger and Valiron's quantum λ-calculus [16] is defined. The model is formulated in particle-style. In [4] QMLL, an extension of MLL with a new kind of

modality, is studied. QMLL is sound and complete with respect to quantum circuits, and an interactive (particle-style) abstract machine is defined. In both cases, adopting a particle-style approach has a bad consequence: the "quantum" tensor product does *not* coincide with the tensor product in the sense of linear logic. Here we show that adopting the wave-style approach solves the problem. Quantum extensions of game semantics are partially connected to this work. See, for example [10, 9]. Purely linear quantum lambda-calculi (*with* measurements) can be given a fully abstract denotational semantics, like the one proposed by Selinger and Valiron [17]. In their work, closure (necessary to interpret higher-order functions) is not obtained via traces and is not directly related in any way to the geometry of interaction. Moreover, morphisms are just linear maps, and so the model is far from being a quantum operational semantics. A language of terms similar to $\mathsf{Q}\Lambda$ has been also studied in [21], where the calculus of proof-nets $\mathsf{MLL}_{\mathsf{qm}}$ is introduced. $\mathsf{MLL}_{\mathsf{qm}}$'s syntax also includes a measurement box-like operator (which models the possibility of "observe" the value of a quantum bit [15]). A multi-token machine semantics for $\mathsf{MLL}_{\mathsf{qm}}$ proof-nets is defined and proved to be *sound*, i.e. invariant along reduction of proof nets. Moreover, although a λ-calculus is given, together with a compilation scheme to $\mathsf{MLL}_{\mathsf{qm}}$ proof-nets, the considered λ-calculus is one with *explicit* qubits, contrary to $\mathsf{Q}\Lambda$. Finally, Arrighi and Dowek's work shows that turning a sum-based algebraic λ-calculus into a quantum computational model can be highly non-trivial [1].

6 Conclusions

We have introduced $\mathsf{IAM}_{\mathsf{Q}\Lambda}$, an interactive abstract machine which provides a sound operational characterisation of any type derivation in a linear quantum λ-calculus $\mathsf{Q}\Lambda$. This is an example of a concrete wave-style token machine whose runs cannot be seen simply as the asynchronous parallel composition of particle-style runs. Interestingly, synchronisation is intimately related to entanglement: if, for example, only unary operators occur in a term (i.e. entanglement is *not* possible), synchronisation is not needed and everything collapses to the particle-style. Our investigation is open to some possible future directions. A natural step will be to extend the syntax of terms and types with an exponential modality. The generalisation of the token machine to this more expressive language would be an interesting and technically challenging subject. Moreover, giving a formal status to the connection between wave-style and the presence of entanglement is a fascinating subject which we definitely aim to investigate further. Finally, an interesting proof-theoretical investigation would consist in analysing the possible connections the with the deep inference-oriented graph formalism developed in [3].

References

[1] Pablo Arrighi & Gilles Dowek (2008): *Linear-algebraic lambda-calculus: higher-order, encodings, and confluence.* In: *RTA*, pp. 17–31, doi:10.1007/978-3-540-70590-1_2.

[2] E. Bernstein & U. Vazirani (1997): *Quantum Complexity Theory*. *SIAM J. Comput.* 26(5), pp. 1411–1473, doi:10.1137/S0097539796300921.

[3] R. Blute, A. Guglielmi, I. Ivanov, P. Panangaden & L. Straßburger (2014): *A Logical Basis for Quantum Evolution and Entanglement*. In: *Categories and Types in Logic, Language, and Physics*, *LNCS* 8222, pp. 90–107, doi:10.1007/978-3-642-54789-8_6.

[4] U. Dal Lago & C. Faggian (2011): *On Multiplicative Linear Logic, Modality and Quantum Circuits*. In: *QPL*, *Electron. Proc. Theor. Comput. Sci.* 95, pp. 55–66, doi:10.4204/EPTCS.95.6.

[5] U. Dal Lago, A. Masini & M. Zorzi (2009): *On a Measurement-Free Quantum Lambda Calculus with Classical Control*. *Math. Structures Comput. Sci.* 19(2), pp. 297–335, doi:10.1017/S096012950800741X.

[6] U. Dal Lago, A. Masini & M. Zorzi (2010): *Quantum Implicit Computational Complexity*. Theoret. Comput. Sci. 411(2), pp. 377–409, doi:10.1016/j.tcs.2009.07.045.

[7] U. Dal Lago, A. Masini & M. Zorzi (2011): *Confluence Results for A Quantum Lambda Calculus with Measurements*. Electron. Notes Theor. Comput. Sci. 270(2), pp. 251–261, doi:10.1016/j.entcs.2011.01.035.

[8] U. Dal Lago & M. Zorzi (2013): *Wave-Style Token Machines and Quantum Lambda Calculi (Long Version)*. Available at `http://arxiv.org/abs/1307.0550`.

[9] Y. Delbecque (2011): *Game Semantics for Quantum Data*. Electron. Notes Theor. Comput. Sci. 270(1), pp. 41–57, doi:10.1016/j.entcs.2011.01.005.

[10] Y. Delbecque & P. Panangaden (2008): *Game Semantics for Quantum Stores*. Electron. Notes Theor. Comput. Sci. 218, pp. 153–170, doi:10.1016/j.entcs.2008.10.010.

[11] J.-Y. Girard (1989): *Geometry of Interaction I: Interpretation of System F*. In: Proc. of the Logic Colloquium '88, pp. 221–260, doi:10.1016/s0049-237x(08)70271-4.

[12] G. Gonthier, M. Abadi & J.-J. Lévy (1992): *The Geometry of Optimal Lambda Reduction*. In: *POPL*, pp. 15–26, doi:10.1145/143165.143172.

[13] I. Hasuo & N. Hoshino (2011): *Semantics of higher-order quantum computation via geometry of interaction*. In: *LICS*, pp. 237–246, doi:10.1109/LICS.2011.26.

[14] Ian Mackie (1995): *The Geometry of Interaction Machine*. In: *POPL*, pp. 198–208, doi:10.1145/199448.199483.

[15] M. Nielsen & I. Chuang (2000): *Quantum computation and quantum information*. Cambridge University Press.

[16] P. Selinger & B. Valiron (2006): *A lambda calculus for quantum computation with classical control*. Math. Structures Comput. Sci. 16(3), pp. 527–552, doi:10.1017/S0960129506005238.

[17] Peter Selinger & Benoît Valiron (2008): *On a Fully Abstract Model for a Quantum Linear Functional Language*. Electron. Notes Theor. Comput. Sci. 210, pp. 123–137, doi:10.1016/j.entcs.2008.04.022.

[18] Peter W. Shor (1997): *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*. SIAM J. Comput. 26(5), pp. 1484–1509, doi:10.1137/S0097539795293172.

[19] A. van Tonder (2004): *A lambda calculus for quantum computation*. SIAM J. Comput. 33(5), pp. 1109–1135, doi:10.1137/S0097539703432165.

[20] M. Volpe, L. Viganò & M Zorzi (2014): *Quantum States Transformation and Branching Distributed Temporal Logic*. In: *WOLLIC*, *LNCS* 8652, pp. 1–19, doi:10.1007/978-3-662-44145-9_1.

[21] Akira Yoshimizu, Ichiro Hasuo, Claudia Faggian & Ugo Dal Lago (2014): *Measurements in Proof Nets as Higher-Order Quantum Circuits*. In: *ESOP*, *LNCS* 8410, pp. 371–391, doi:10.1007/978-3-642-54833-8_20.

[22] M. Zorzi (2013): *On Quantum Lambda Calculi: a Foundational Perspective*. Math. Structures Comput. Sci., pp. 1–94. Accepted for Publication.