1 Background and Motivation

Quantum Computing is a new and emerging computational paradigm whose main idea is to use quantum mechanical phenomena, such as entanglement and superposition, in order to perform computation. A quantum computer can solve problems which are out of reach for classical computers (e.g. factorisation of large numbers [1], solving large linear systems [2]) and this has caused a major surge of interest into the development of quantum technologies. A major breakthrough was recently achieved by Google which demonstrated quantum computational advantages on existing quantum computers [3]. The development of quantum computing technologies is rapidly accelerating and has recently benefited from major investment from technological companies with dedicated quantum research teams, such as Google, Microsoft, IBM, and many other quantum startup firms. The French government also recently announced a major initiative to further develop quantum technologies [4].

Quantum technologies are also one of the main focus areas for the European Research Council [5].

The recent developments of quantum technologies point out the necessity of filling the gap between theoretical quantum algorithms and the actual (prototypes of) quantum computers. As a consequence, quantum software and in particular quantum programming languages play a key role in the future development of quantum computing. A quantum programming language is simply a programming language which can be used to write programs that may be executed on a quan-
Quantum computer. Quantum programming languages are an active research topic and new results are published every year in top-tier computer science conferences [6].

The topic of this PhD proposal is concerned with the study of quantum control within quantum programming languages. Quantum control is a very important primitive in quantum computation: it is crucial for achieving computational advantages over classical algorithms. Quantum control does not admit a classical interpretation and there is little agreement on what is the best way to handle it in a programming language.

1.1 Quantum vs Classical Control

In a classical programming language, the control flow is classical. This means that when evaluating if-statements (and more generally pattern matching), the choice function which determines the subsequent reductions would select precisely one subprogram with which to continue evaluation. This is best illustrated by an example.

Example 1. Consider the following function

\[
\text{classical} : \ (\text{Bool}, \text{Bool}) \rightarrow (\text{Bool}, \text{Bool})
\]

\[
\text{classical} \ (b_1, b_2) = \begin{cases} 
(b_1 \implies \text{False}) & \text{then } (b_1, b_2) \\
(b_1, \text{not } b_2) & \text{else } (b_1, \text{not } b_2)
\end{cases}
\]

This simple piece of pseudocode defines a function called \text{classical} which given two values of type \text{Bool} would return two values of type \text{Bool}, where the second output might be flipped, depending on what the first input is. This realises a classical controlled-not function. In particular, when evaluating \text{classical} on any input arguments, we know that the program execution would proceed in a purely deterministic manner.

With quantum control, however, the program dynamics are considerably more complicated. In order to understand what happens, let us consider the quantum controlled-not function. It is described by the unitary map whose action on the basis states is given by:

\[
\begin{align*}
\text{CX} |00\rangle &= |00\rangle \\
\text{CX} |01\rangle &= |01\rangle \\
\text{CX} |10\rangle &= |11\rangle \\
\text{CX} |11\rangle &= |10\rangle.
\end{align*}
\]

This determines a (physically realisable) quantum operation which is very important in quantum computation. The four equations above only specify the behaviour of the operation on the basis states, but the operation is defined on all possible pairs of input qubits (quantum bits) by taking its unique linear extension. This means that the function may be applied to quantum data which is in a state of superposition (the state of the system is described by a linear combination) in which case quantum entanglement may occur, which is crucial for obtaining computational advantages. For example if the first input qubit is in a superposition of \(|0\rangle\) and \(|1\rangle\), then applying the \text{CX} operation to this data results in

\[
\text{CX} \left( \frac{|0\rangle + |1\rangle \otimes |0\rangle}{\sqrt{2}} \right) = \frac{|0\rangle + |1\rangle}{\sqrt{2}}.
\]

The above operation corresponds to one application of \text{CX}, but less formally, it might seem like we have applied \text{CX} two times when we consider the intermediate linear combinations. Moreover, upon performing a quantum measurement (which is necessary for us to extract information from the system), we obtain a probability distribution of potential results (bits in this case). Because of this, quantum program dynamics are not deterministic, but are probabilistic instead. Moreover, the resulting probability distributions that describe quantum program execution cannot be explained via classical statistical mechanics and they are fundamentally different compared to the probability distributions obtained through classical probabilistic programming languages. This brings unique challenges on how to handle quantum control in quantum programming languages and related logics.
1.2 A Diagrammatic Account of Coherent Quantum Control

A special case of coherent quantum control may be achieved via the quantum switch \textsuperscript{7}. The main idea is the following: the order in which two unitary evolutions $U$ and $V$ are applied is controlled by the state of a control qubit; in particular, if the control qubit is in superposition, then both $UV$ and $VU$ are applied, in superposition (see Figure 1).

![Diagram](image1)

Figure 1: (a) Intuitive behaviour of a polarising beam splitter: vertical polarisation goes through, horizontal polarisation is reflected; (b) Quantum switch of two unitary evolutions $U$ and $V$.

The quantum switch is very interesting for multiple reasons:

- it has been experimentally realised and verified \textsuperscript{8,9};
- it leads to some computational advantages \textsuperscript{10};
- it admits an elegant logical and diagrammatic formulation (inspired by quantum optics), called the PBS-calculus, that is sound and complete \textsuperscript{11}.

Overall, the quantum switch is well-behaved and well-understood. However, this is a special case of coherent quantum control and indeed the PBS-calculus (which completely formalises it) is a low-level language. If we wish to understand how coherent quantum control behaves in general, then we have to consider more expressive languages that can form quantum control, such as quantum programming languages.

1.3 Quantum Control in Quantum Programming Languages

There are two main ways which can be used to implement quantum control, but both have advantages and disadvantages. We discuss them next.

One way to introduce quantum control is to simply assume a built-in constant into the type system which represents the CX operation (e.g. \textsuperscript{12–17}). This approach interacts very well with quantum measurements and classical programming primitives such as classical control, recursion, recursive types, pattern matching, higher-order functions, etc. However, the downside of this approach is that there is only one way to induce quantum control and that is through the built-in constant CX which takes two qubits as inputs. Any other instance of quantum control occurring at higher types has to be implemented by the programmer via the CX constant. Therefore, despite the proliferation of higher-order programming constructs, in order to achieve quantum control, the programmer is forced to work on the level of qubits which enforces a very low-level programming workflow. Simply put, if programmers wish to achieve quantum computational advantages, then they are forced into qubit-level programming. An analogue to this in classical programming is to force programmers to implement their functions via boolean logic gates, which is clearly undesirable.

The other main way of introducing quantum control is via linear-algebraic lambda calculi (e.g. \textsuperscript{18–22}). Here, the main idea is to extend the simply-typed lambda calculus with programming
primitives that allow the programmer to form linear combinations of terms (in the linear-algebraic sense). Such calculi may be specialised to quantum programming languages by introducing a rigorous typing discipline. Compared to the previous approach we mentioned, the advantage of this approach is that one can easily represent quantum control at a higher-level and there is no need to introduce specific constants for dealing with this. However, this approach suffers from many disadvantages: it is difficult to support quantum measurements and many classical programming features such as recursion and recursive types. Simply put, this approach does not interact well with classical computation, which is crucial for the implementation of variational quantum algorithms \cite{23,24}.

2 Programme de la thèse

The main problem which underlies the above issues is that we currently do not have a good understanding of how to achieve a clean separation between quantum and classical control flow. The main objectives of this PhD proposal are:

- to perform a detailed analysis of quantum control in quantum programming languages;
- to formulate a formal type system with both quantum and classical control primitives;
- to describe a type-safe operational semantics for the above type system which is consistent with the laws of quantum mechanics;
- to compare the proposed type system with the well-behaved PBS-calculus in terms of expressivity, structural properties, decidable properties, etc.;
- to establish a sound and computationally adequate mathematical interpretation of this system.

Work-in-progress by Jia, Lindenhovius, Mislove and Zamdzhiev \cite{25} demonstrates how to separate quantum and classical probabilistic effects by enforcing a typing discipline that distinguishes between quantum and classical programs. However, the programming language described in \cite{25} handles quantum control by assuming a built-in constant, which we explained is unsatisfactory. This paper will serve as a good starting point for the PhD student.

The type system. The first step in this program is to design a suitable type system with a precise syntax for the formation of quantum control primitives. As mentioned above, one option is to use an approach similar to the linear-algebraic lambda calculi, where one is able to form linear combinations of terms. However, the student should also consider formulating novel approaches. For instance, by adapting techniques from Reversible Computation \cite{26,27} he can then introduce a notion of unitary pattern matching that could be used for the introduction of quantum control. It is also likely the student will have to introduce additional modalities (adapting approaches from Modal Logic \cite{28,29}) in the system which enforce consistency with the laws of quantum mechanics and which ensure that quantum control is coherent in the presence of quantum measurements and probabilistic computational effects.

The proposed type system is intended to be higher-level and more expressive compared to the PBS-calculus. However, with greater expressivity, it is likely that the system will be more complicated from a structural perspective. For instance, PBS diagrams are valid by construction, but a more expressive language may not have this property by default – the type system has to identify more general (and complicated) formation conditions that ensure the terms indeed represent coherent quantum control.
Operational Semantics. The next step is to formulate a type-safe operational semantics. The overall structure of the semantics is clear—it should be described by a discrete probabilistic reduction relation in a similar way to \cite{25}. However, the operational semantics will also need to account for the construction of controlled quantum unitary functions. This is going to be the main difficulty here. It seems likely that a two-stage operational semantics could be useful towards this goal.

Mathematical Semantics. After the type-safety of the proposed system is established, the student should formulate a sound and adequate mathematical semantics of the language. The overall structure of the semantics would require identifying a category in which to interpret classical computation and another category for interpreting quantum computation. This can follow \cite{25}, where two such categories are identified:

- classical category: the Kleisli category of a commutative probabilistic monad over dcpo’s \cite{30};
- quantum category: hereditarily atomic von Neumann algebras and completely-positive sub-unital maps over them.

However, to interpret the additional modalities and quantum control primitives, the student has to identify a suitable subcategory of the quantum one where coherent quantum control may be formed. Identifying the largest such subcategory is one of the main objectives of this thesis. A likely candidate is the category of Hilbert spaces with unitary maps. The relationship between this subcategory and the ambient quantum one has to be studied extensively and precisely characterised.

2.1 Approche scientifique.

Bibliographie. The PhD student should begin his research by conducting an extensive literature review. Here we mentioned several papers that can be used as a starting point, but there are many more which we have not mentioned. In particular, the student has to review relevant literature on type systems, logics, (quantum) programming language design, categorical semantics, operator algebras, diagrammatic languages and quantum foundations. Articles on these topics are published in a wide-range of computer science conferences and journals, but also in mathematical and physical journals.

Méthodologie. The PhD student is going to undertake research in Theoretical Computer Science. He is expected to formulate a consistent theory that describes quantum control in quantum programming languages. The vast majority of the work will be theoretical and the student is expected to provide precise definitions to describe his ideas. He also has to state lemmas, propositions and theorems which describe the main results of his research and to provide detailed proofs for them. There are many expected properties that the proposed type system needs to have and they all admit clear and concise descriptions. Proving that these properties hold is expected and will be evidence that the PhD student has succeeded in his research.

References


