Distributed Quantum Programming

Ellie D’Hondt and Yves Vandriessche

Vrije Universiteit Brussel, Belgium
Ellie.DHondt@vub.ac.be
Yves.Vandriessche@vub.ac.be

Abstract. In this paper we explore the structure and applicability of the Distributed Measurement Calculus (DMC), a formal assembly language for distributed measurement-based quantum computations. We describe its syntax and semantics, both operational and denotational, and state several properties that are crucial to the practical usability of our language, such as equivalence of our semantics, as well as compositionality and context-freeness of DMC programs. We show how to put these properties to use by constructing a composite program that implements distributed controlled operations, and demonstrate that the semantics of this program does not change under the various composition operations. Our formal model is meant to be the basis of a virtual machine for distributed quantum computations, where programming execution no longer needs to be analysed by hand, while at the same time formal properties may be relied upon. Several insights on how to move towards this virtual level are given.

1 Introduction

During the last decennia, quantum information has managed to become a significant field of research in the exact and applied sciences. Although it is a relatively new discipline one can currently discern several sub-disciplines such as quantum cryptography, information theory, computability, error correction, fault tolerance, computations and of course there is also the experimental research on the construction of quantum computers [NC00]. Nevertheless, the development of quantum information as a proper computational domain of computer science is lagging behind. Indeed there is no such thing as a quantum computational paradigm. By this we mean a framework in which quantum problems can be expressed and solved in terms of data structures, algorithms, techniques such as abstraction, all of these wrapped up in an associated programming language. Paradigm building has proved to be an extremely useful approach in computer science. It has led to theoretically equal but practically very different programming frameworks, such as functional, imperative, logic and object-oriented programming. For this reason we expect this approach to be crucial also in developing quantum programming paradigms.

The first step in paradigm building is to construct a low-level quantum programming language and determine its properties. By low-level we mean that we need to define syntactical expressions for each physically allowed quantum
Distributed Quantum Programming

operation: preparation, unitary transformation, measurement, combined with classical control expressions if so desired. The syntax in its own is not the goal, but rather a means by which to facilitate investigations with computer science techniques. First, we need to determine the functionality of a quantum program, its semantics. The most obvious way to do this is the operational semantics, the most practical is the denotational semantics. While in the former a program’s meaning is given as a sequence of state-changing operations, in the latter it is instead a mathematical object. Paramount is linking both, so that one can use whichever in future analyses. Through a programming language’s semantics one can investigate notions such as composition and context-freeness. These properties are crucial when one wants to build more complex programs. Indeed, they allow the semantics of these larger programs to be built up from that of smaller components using rules for composition, rather than having to be determined from scratch. While these properties may seem obvious, computer science is littered with examples where they were mistakenly taken to be true, leading to problems in programming language development (see for example [BA81]).

Recent advances in quantum communication and cryptographic computations motivate the need for a programming paradigm centred on distributed quantum computations. In a distributed system one has concurrently acting agents in separated locations, operating locally on a quantum state, which may be entangled over agents, and coordinating their actions via communication. Formal frameworks for distributed quantum computation have only very recently begun to appear. Typically, these are a combination of classical process theory, which formalises notions of concurrency and communication, the quantum circuit model, i.e. local operations are unitary transformations of an agent’s qubits, and given initial shared entanglement. First, there is the work in [LJ04, JL05], which is directly built upon classical process calculi. While this model profits from being closely related to existing classical models, the disadvantage is that it is hard to focus on quantum behaviour. A different approach is advanced in the model known as communicating quantum processes or CQP [GN04, GNP05], where the typical communication primitives of process calculi are combined with computational primitives from QPL [Sel03]. The basic model of CQP is more transparent. This model serves as a basis for the development of formal verification techniques, in particular for proving the security of larger scale quantum communication systems. In related work, a probabilistic model-checking tool built upon an existing automated verification component is developed [GNP05]. There is also the work in [AM05], which is specifically tailored to security issues in cryptographic protocols. In our work, however, we are much more interested in the expressiveness of quantum distributed computations and the behavioural properties of distributed primitives. In a way, we take the inverse approach, assuming that computations are well-defined and investigating what programming concepts are at work, instead of the other way around.

While the fact that formal verification tools for distributed quantum computation are currently under development may suggest that a mature distributed paradigm already exists, this is actually not the case. Distributed protocols are