A Dynamic Logic for QASM Programs

Carlos Tavares(✉)

High-Assurance Software Laboratory/INESC TEC, Braga, Portugal
ctavares@inesctec.pt

Abstract. We define a dynamic logic for QASM (Quantum Assembly) programming language, a language that requires the handling of quantum and probabilistic information. We provide a syntax and a model to this logic, providing a probabilistic semantics to the classical part. We exercise it with the *quantum coin toss* program.

Keywords: Quantum logic · Quantum programming · Dynamic logic

1 Introduction

The programming languages, calculi, and logics, developed in the course of the past 20 years, for quantum computing have been gaining relevance with the appearance of the first proof-of-concept quantum computers and quantum programming languages. One of such is the Quantum Assembly Language [CBSG17], the quantum circuit specification language in use in the commercially available quantum hardware supplied by IBM, the IBM Q platform [ibm18] (a small example of the language is depicted in Fig. 1).

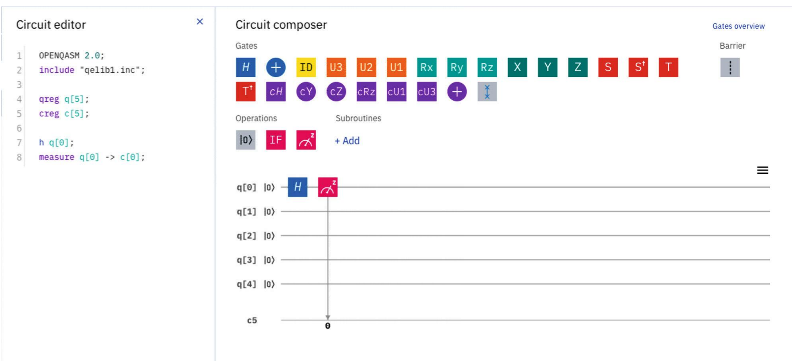

Fig. 1. Example of the definition of a circuit in the QASM language. On the right side the visual definition of the circuit and on the left side the correspondent QASM code.

Besides the description of unitary quantum circuits, the language encompasses classical control flow instructions, such as measurements, which possess a probabilistic nature, and *if statements*. We propose a dynamic logic for this language exploring two main points of interest: the direct handling of quantum and probabilistic propositions, and a possible axiomatic semantics.

L. Soares Barbosa and A. Baltag (Eds.): DaLí 2019, LNCS 12005, pp. 209–217, 2020.
https://doi.org/10.1007/978-3-030-38808-9_13

2 Quantum Computing

In this section, we introduce quantum computing from a state based perspective (i.e. by the definition of states, transitions, and acceptance states), as usually presented in the literature [Deu85]. For a more complete understanding of quantum computing, we recommend the reading of [NC02].

2.1 States

The state space of a quantum system is given by the set of unitary vectors (vectors of norm 1) definable in its respective *Hilbert space*. The qubit, the quantum version of the classical bits, consists of a *Hilbert* space of dimension 2, $\mathcal{H}^2$, with $\{|0\rangle, |1\rangle\}$ as an orthogonal basis. The correspondent state space reads as follows:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle\,;|\alpha|^2 + |\beta|^2 = 1; \lambda|\psi\rangle \cong |\psi\rangle\,, \lambda \in \mathbb{C} \tag{1}$$

Quantum systems can be combined, employing the *tensor* product $\otimes$. For a n-qubit system, the set of possible states reads as follows:

$$\bigotimes_{i=0}^{n-1} \mathcal{H}_i^2 \tag{2}$$

For systems with more than one qubit, one verifies the existence of *non-separable states*, i.e. states that cannot be written as states of individual qubits, as for instance in the following *Bell* state: $|\Phi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. The latter is the mathematical expression of the so-called physical phenomenon of *entanglement*.

2.2 Transitions (programs)

In quantum mechanics, transitions preserve *unitarity* of states. Hence, programs correspond to *unitary* operators ($U.U^\dagger = I$). For a quantum system with n qubits the *signature* of the transition operators reads as follows:

$$U^{\otimes n} : \mathcal{H}^{2\otimes n} \to \mathcal{H}^{2\otimes n}$$

In quantum computation practice, a rather less *abstract* notion is used, the so-called *quantum circuits* [Deu89], where unitary operators are approximated by compositions of *primitive unitary operators*, such as the H, X, Y, or Z gates.

2.3 Acceptance States

Measurements, (mathematically $Proj_\varphi, or\, |\varphi\rangle\langle\varphi|$), can be interpreted as a method that causes the *collapse* of *superposition* states to elements of an *orthogonal basis*, (e.g. in the qubit case $|0\rangle$ and $|1\rangle$). An acceptance state is one where the correct output is obtained upon measurement, with probability[1] greater than α.

[1] The probability of obtaining φ in a measurement is $\langle s|Proj_\varphi s\rangle$ where s is a state and $\langle.|.\rangle$ is the internal product of the *Hilbert* space. In equation (1), $|\alpha|^2$ and $|\beta|^2$, are the probabilities of obtaining $|0\rangle$ and $|1\rangle$, which is 0.5 in both cases: $(\left(\frac{1}{\sqrt{2}}\right)^2 = 0.5)$.

3 A Dynamic Logic for QASM

The QASM programming language is not a *pure* quantum programming language as it involves, *measurements*, which possess a probabilistic nature, and classical flow instructions depending on those measurements, requiring the handling of probabilistic and quantum programs. Our approach to this problem is somehow inspired in the fusion of works of Baltag and Smets [BS04,BBK+14] for the quantum part and of Kozen [Koz85,Koz81] for the probabilistic part.

3.1 Syntax

As usual in dynamic logic, the syntax is divided into two layers: one of the *programs* and one of the *formulas*. The program's layer encompasses a fragment of the QASM language, which includes the classical control instructions (*if statements*, creation of *classical* and *quantum* registers, and *measurements* of quantum registers), as well as several standard unitary operations (x, z, h and cnot gates) (Fig. 2).

$$
\begin{array}{lll}
\langle \mathit{argument} \rangle & ::= & \mathrm{id} \mid \mathrm{id}\ [\mathrm{index}] \\
\langle \mathit{test} \rangle & ::= & \langle \mathit{argument} \rangle == \langle \mathit{natural\ number} \rangle \\
\langle \pi_q \rangle & ::= & \mathrm{x\ qreg\_id}\ [\mathrm{index}] \mid \mathrm{z\ qreg\_id}\ [\mathrm{index}] \mid \mathrm{h\ qreg\_id}\ [\mathrm{index}] \\
 & \mid & \mathrm{cx\ qreg\_id}\ [\mathrm{index}],\ \mathrm{qreg\_id}\ [\mathrm{index}]\ (\textbf{unitary gates}) \\
 & \mid & \textbf{measure}\ \mathrm{qreg\_id} \rightarrow \mathrm{creg\_id}\ (\textbf{measurements}) \\
 & \mid & \pi_q ; \pi_q \\
\langle \pi \rangle & ::= & \textbf{creg}\ \mathrm{id}\ [\mathrm{size}] \mid \textbf{qreg}\ \mathrm{id}\ [\mathrm{size}]\ (\textbf{creation of registers}) \\
 & \mid & \textbf{if}\ \langle \mathit{test} \rangle\ \textbf{then}\ \pi_q\ (\textbf{if statements}) \\
 & \mid & \pi ; \pi \\
\langle p \rangle & ::= & \perp \mid \underline{0} \mid \underline{1} \mid p^{\mathit{register}}_{\mathit{index}} \\
\langle \varphi \rangle & ::= & \left(p, f_{\langle \mathit{test} \rangle} = g\right) \mid P^{\geq r} \varphi \mid \langle \pi \rangle \varphi \mid \neg \varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi
\end{array}
$$

Fig. 2. Formulas layer and programs layer

On the formula side, atomic propositions are pairs $\left(p, f_{\langle test \rangle} = g_{\langle test \rangle}\right)$ where p corresponds to quantum propositions over qubit states and $f_{\langle test \rangle} = g$ corresponds to equality expressions over *the probability distributions* definable on the possible tests over classical variables. On the quantum side $\underline{0}$ and $\underline{1}$ denote that 0 or 1 are true upon measurement with 1 as probability, and the $p^{register}_{index}$ *narrows* a proposition range to a specific register and qubit, as for instance $\underline{0}^q_0$, which means that qubit 0 of register q has value 0. The $P^{\geq r}\varphi$ modality establishes restrictions to the probability of propositions for instance $P^{=0.5}p$. The $\langle \pi \rangle$ has the usual meaning of "the proposition φ *may* hold upon the execution of program π" and the usual *minimal* set of Boolean connectives is included.

3.2 Semantics

The semantics of this logic is given in terms of a *Labelled transition system* [HM80], defined by a *tuple*:

$$M = (\mathcal{G}, [\![.]\!] : \mathcal{A}_p \cup \mathcal{A}_\pi \rightarrow 2^{\mathcal{G}} \cup \mathcal{G} \times \mathcal{G}) \tag{3}$$

where $\mathcal{G}$ is a set of states and $[\![.]\!]$ a *meaning* function, from the type of the *well-formed* syntactic expressions of propositions ($\mathcal{A}_p$) and programs ($\mathcal{A}_\pi$), to the *powerset*, and *Cartesian product* of the set of states, respectively.

3.3 The State Space

A state of a program in the *QASM* language is defined by its classical and quantum components. Each of such components is divided into one or many *independent* registers, each composed of a set of quantum or classical bits, resulting in the following state space:

$$\underbrace{\underbrace{\mathcal{H}^2 \otimes \ldots \otimes \mathcal{H}^2}_{\text{quantum register}} \times \ldots}_{\mathcal{S}} \times \underbrace{\underbrace{\{0,1\} \times \ldots \times \{0,1\}}_{\text{classical register}} \times \ldots}_{\mathcal{C}} \tag{4}$$

On the classic side, we work on a probabilistic setting, due to the existence of quantum measurements, which work as *random assignments*. Thus, the set of possible states corresponds to the distributions definable on the tests[2] over the classical variables. Therefore, a distribution is given by a *measure* [Koz85] from the set of *tests* to the probability interval $[0, 1]$:

$$\mu_s : 2^{\mathcal{C}} \rightarrow [0, 1]$$

However, the actual state in this logic is defined the equality operator over two *measures*, so an actual state is characterized as a function with *signature*:

$$\mu_s : 2^{\mathcal{C}} \times 2^{\mathcal{C}} \rightarrow \{0, 1\}$$

In conclusion the state space of a QASM program is given by the Cartesian product of the possible states of the independent quantum and classical registers, denoted *Registers*, where in the former the set of states is given by the *tensor product* of quantum bits, and in the latter by the possible distributions definable over the configurations of the classical *bits*.

$$\mathcal{G} \equiv \prod_{\text{quantum register} \in Registers} \bigotimes \mathcal{H}^{2^{\otimes reg\_size}} \times \prod_{\text{classical register} \in Registers} 2^{2^{\mathcal{C}} \times 2^{\mathcal{C}}}$$

[2] Tests correspond to the σ-algebra over the valuation set $\mathcal{C}$. For valuations with a discrete domain, it corresponds to the powerset $2^{\mathcal{C}}$. Tests form a *Boolean* algebra.

3.4 Propositions

As seen in Sect. 3.1, propositions correspond to a pair of quantum and classical propositions, where quantum propositions are of type $2^{\mathcal{S}}$, the *powerset* of the quantum state space, and the probabilistic propositions of the type $2^{\mathcal{C}\times\mathcal{C}}$, the pairs of *fuzzy predicates*[3] definable on the state space $2^{\mathcal{C}\times\mathcal{C}}$. Therefore, the type of the global propositions reads as follows:

$$p : 2^{\mathcal{S}} \times \left(2^{\mathcal{C}\times\mathcal{C}}\right)$$

Definition 1. *Semantics for proposition constructors.*

We define $proj_q$ *as the quantum part of a proposition, and* $proj_p$ *as the probabilistic part of the proposition.*

i. $[[\underline{1}]] = \{s|\ \langle s|Proj_{\underline{1}}s\rangle = 1\}$. *Similarly for* $[[\underline{0}]]$.
$[[\bot]] - \emptyset$.
$[[p^{register}_{index}]$ - *The set where the proposition* p*, restricted to a register and a specific qubit index, holds.*
ii. $[\![(p, f = g)]\!] = \{s|s \in [\![p]\!] \wedge f(proj_p(s)) = g(proj_p(s))\}$ *and* $proj_p(s) \in \mathcal{C}$.
iii. $[\![P^{\geq r}\varphi]\!] = \{s|\ \langle s \mid Proj_{proj_q \varphi} s\rangle \geq r\}$.
The set of states where quantum proposition component φ *holds with probability greater than* r.
iv. $[\![\varphi_1 \wedge \varphi_2]\!] = \{s|s \in [\![proj_q(\varphi_1) \cap proj_q(\varphi_2)]\!] \wedge s \in [\![proj_p(\varphi_1) \cap proj_p(\varphi_2)]\!]\}$
v. $[\![\varphi_1 \vee \varphi_2]\!] = \{s|s \in [\![proj_q(\varphi_1) \cup proj_q(\varphi_2)]\!] \wedge s \in [\![proj_p(\varphi_1) \cup proj_p(\varphi_2)]\!]\}$
vi. $[\![\neg\varphi]\!] = \{s|s \notin [\![proj_q \varphi]\!] \wedge s \notin [\![proj_p \varphi]\!]\}$
vii. $[\![\langle\pi\rangle\, \varphi]\!] = \{s|\exists u : (s, u) \in [\![\pi]\!] \wedge u \in [\![\varphi]\!]\}$
The set of states where the proposition φ *holds upon the execution of program* π.

3.5 Program Semantics

Programs in this logic correspond to deterministic relations between states:

$$[\![.]\!] : \mathcal{A}_\pi \rightarrow \mathcal{G} \times \mathcal{G} \tag{5}$$

This function denotes an *accessibility relation*, i.e. *directed* valid transitions between pairs of states (source to output), under the action of a given program.

[3] A *fuzzy predicate* corresponds to a *measurable function* [Koz85] from the set of states to the probability interval $[0, 1]$, in this case, $\mathcal{C} \rightarrow [0, 1]$. The *fuzzy predicate* is characteristic of a test.

Definition 2. *Semantics for programs (accessibility relation)*

$p \in 2^{\mathcal{S}}$ - *any quantum proposition*
$\alpha \in 2^{\mathcal{C}\times\mathcal{C}}$ - *any probabilistic proposition* ($f_{\langle test\rangle} = g$).

(n) Creation of registers (upon a register is created its value is necessarily 0, both for quantum and the probabilistic parts):
$[\![creg\ reg\_id\ [size]]\!] = \{(s,u)|s \in [\![(p, \perp_{reg\_id})]\!] \wedge u \in [\![(p, f_{reg\_id=0}(u) = 1)]\!]\}$
$[\![qreg\ reg\_id\ [size]]\!] = \{(s,u)|s \in [\![(\perp_{reg\_id}, \alpha)]\!] \wedge u \in [\![\left(\underline{0}^{reg\_id}_{0..size-1}, \alpha\right)]\!]\}$.
Pairs of states where $\perp$ *holds in the source state and 0 in the output state.*
(h) Hadamard operator:

$$\begin{aligned}
&[\![h\ reg_id\ [index]]\!] = \\
&\quad \{(s,u)|s \in [\![\left((Pr^{=p_i}p) \wedge \underline{0}^{reg_id}_{index}, \alpha\right)]\!] \vee s \in [\![\left((Pr^{=p_i}p) \wedge \underline{1}^{reg_id}_{index}, \alpha\right)]\!] \\
&\quad \wedge u \in [\![\left(Pr^{=p_i*0.5}(p \wedge \underline{0}^{reg_id}_{index}) \wedge Pr^{=p_i*0.5}(p \wedge \underline{1}^{reg_id}_{index}), \alpha\right)]\!]\} \\
&\quad \cup\{(s,u)|s \in [\![\left(Pr^{=p_i*0.5}(p \wedge \underline{0}^{reg_id}_{index}) \wedge Pr^{=p_i*0.5}(p \wedge \underline{1}^{reg_id}_{index}), \alpha\right)]\!] \wedge \\
&\quad (u \in [\![\left((Pr^{=p_i}p) \wedge \underline{0}^{reg_id}_{index}, \alpha\right)]\!] \vee u \in [\![\left((Pr^{=p_i}p) \wedge \underline{1}^{reg_id}_{index}, \alpha\right)]\!])\}
\end{aligned}$$

Pairs of states defined by either 0 *or* 1 *on the source state and a superposition of* 0 *and* 1 *in the output state, or* vice-versa.
(x) X operator:

$$\begin{aligned}
&[\![x\ reg_id\ [index]]\!] = \{(s,u)|s \in [\![\left(p \wedge \underline{1}^{reg_id}_{index}, \alpha\right)]\!] \wedge u \in [\![\left(p \wedge \underline{0}^{reg_id}_{index}, \alpha\right)]\!] \\
&\quad \vee s \in [\![\left(p \wedge \underline{0}^{reg_id}_{index}, \alpha\right)]\!] \wedge u \in [\![\left(p \wedge \underline{1}^{reg_id}_{index}, \alpha\right)]\!]\}
\end{aligned}$$

Pairs of states where 0 *holds in the source state and* 1 *in the output state, or* vice-versa *(same effect as a classical not gate).*
(m) Measure:

$$\begin{aligned}
&[\![measure\ qreg_id \rightarrow\ creg_id]\!] \\
&\quad = \{(s,u)|s \in [\![\left(\bigwedge_i^{2^{size}} P^{=p_i}\underline{i}, \mathcal{D}_{creg_id}(\bigwedge_i f_{creg_id==i})\right)]\!] \\
&\quad\quad \wedge u \in [\![\left(\bigvee_i i, \bigwedge_i f_{creg_id==i}(u) == p_i\right)]\!]\}
\end{aligned}$$

Pairs of states where the probability distribution of the valuations of a set of qubits in the source state, is the same as the verified in a set of classical bits in the output state, where $\mathcal{D}_{creg\_id}$ *denotes a distribution* compatible upon measurement *with* $\bigwedge_i f_{creg\_id==i}$ ($\{d|meas\circ d = f\}$ *where* $\circ$ *is the* Lebesgue integral*)*

(;) Sequence
$[\![\pi_1;\pi_2]\!] = \{(s,u)|\exists t(s,t) \in [\![\pi_1]\!] \wedge (t,u) \in [\![\pi_2]\!]\}$

4 An Example: A Quantum Coin Tossing Program

This section, illustrates the logic through the proof of correctness of a *simple* quantum program for *quantum coin tossing* (prepare a qubit in a superposition state and measure it, obtaining 0 or 1 with equal probability), which translates into the following QASM program:

```
OPENQASM 2.0;
include "qelib1.inc";
qreg q[1];
creg c[1];
h q[0];
measure q[0] -> c[0];
```

The correctness of such program implies the following post-condition:

$$\big(\underline{0}_0^q \vee \underline{1}_0^q, f_{\langle c[0]==1\rangle}(x) = 0.5 \wedge f_{\langle c[0]==0\rangle}(x) = 0.5\big) \text{ with } x \in \mathcal{C} \tag{6}$$

where $\underline{0} \vee \underline{1}$ denotes the quantum qubit q has either, mutually exclusively, the values 0 or 1, and $\mathcal{C} = \{0,1\}$. The fact that post-condition (6) holds upon the execution of the program $\mathrm{qreg\ q[1]; creg\ c[1]; h\ q[0]; measure\ q[0]} \rightarrow \mathrm{c[0]}$ is expressed through the following formula:

$$\begin{aligned}&\langle \mathrm{qreg\ q[1]; creg\ c[1]; h\ q[0]; measure\ q[0]} \rightarrow \mathrm{c[0]}\rangle \\ &\quad \big(\underline{0}_0^q \vee \underline{1}_0^q, f_{\langle c[0]==1\rangle}(x) = 0.5\big) \wedge \big(\underline{0}_0^q \vee \underline{1}_0^q, f_{\langle c[0]==0\rangle}(x) = 0.5\big) \text{ with } x \in \mathcal{C}\end{aligned}$$

This is proved by the rules of Definition 2:

Proof.

$$\begin{aligned}&[\![\langle \mathrm{qreg\ q[1]; creg\ c[1]; h\ q[0]; measure\ q[0]} \rightarrow \mathrm{c[0]}\rangle \\ &\quad \big(\underline{0}_0^q \vee \underline{1}_0^q, f_{\langle c[0]==1\rangle}(x) = 0.5 \wedge f_{\langle c[0]==0\rangle}(x) = 0.5\big)]\!] \\ &= \\ &\{s | \exists u : (s,u) \in [\![\mathrm{qreg\ q[1]; creg\ c[1]; h\ q[0]; measure\ q[0]} \rightarrow \mathrm{c[0]}]\!] \\ &\wedge u \in [\![\big(\underline{0}_0^q \vee \underline{1}_0^q, f_{\langle c[0]==1\rangle}(proj_p(u)) = 0.5 \wedge f_{\langle c[0]==0\rangle}(proj_p(u)) = 0.5\big)]\!]\} \\ &\text{ with } proj_p(u) \in \mathcal{C} \\ &= \text{ (use of the (;) rule)} \\ &\{s | \exists u : \exists t : (s,t) \in [\![\mathrm{qreg\ q[1]; creg\ c[1]; h\ q[0]}]\!] \wedge (t,u) \in [\![\mathrm{measure\ q[0]} \rightarrow \mathrm{c[0]}]\!] \\ &\quad \wedge u \in [\![\big(\underline{0}_0^q \vee \underline{1}_0^q, f_{\langle c[0]==1\rangle}(proj_p(u)) == 0.5 \wedge f_{\langle c[0]==0\rangle}(proj_p(u)) = 0.5\big)]\!]\} \\ &= \text{ (use of the (m) rule)} \\ &\{s | \exists u : \exists t : (s,t) \in [\![\mathrm{qreg\ q[1]; creg\ c[1]; h\ q[0]}]\!] \\ &\quad \wedge t \in [\![\big(P^{=0.5}\underline{0}_0^q, P^{=0.5}\underline{1}_0^q, \mathcal{D}_c(f_{\langle c[0]==0\rangle} \wedge f_{\langle c[0]==1\rangle})\big)]\!]\end{aligned}$$

$\wedge\, u \in [\![(\underline{0}_0^q \vee \underline{1}_0^q, f_{\langle c[0]==1\rangle}(proj_p(u)) == 0.5 \wedge f_{\langle c[0]==0\rangle}(proj_p(u)) = 0.5)]\!]\}$

$=$ (use of (;) and (h). u can be eliminated because $u \in [\![...]\!]$ is true)

$\{s|\exists t : \exists t' : (s,t') \in [\![\text{qreg q[1]; creg c[1]}]\!]$

$\wedge \big(t' \in [\![(\underline{0}_0^q, \mathcal{D}_c(f_{\langle c[0]==0\rangle} \wedge f_{\langle c[0]==1\rangle}))]\!] \vee t' \in [\![(\underline{1}_0^q, \mathcal{D}_c(f_{\langle c[0]==0\rangle} \wedge f_{\langle c[0]==1\rangle}))]\!]\big)$

$\wedge\, t \in [\![(P^{=0.5}\underline{0}_0^q, P^{=0.5}\underline{1}_0^q, \mathcal{D}_c(f_{\langle c[0]==0\rangle} \wedge f_{\langle c[0]==1\rangle}))]\!]\}$

$=$ (use of (;) and (nreg) rules. t can be eliminated because $t \in [\![...]\!]$ is true)

$\{s|\exists t' : \exists t'' : (s,t'') \in [\![\text{qreg q[1]}]\!] \wedge t'' \in [\![(\underline{0}_0^q, \perp_c)]\!]$

$\wedge \big(t' \in [\![(\underline{0}_0^q, \mathcal{D}_c(f_{\langle c[0]==0\rangle} \wedge f_{\langle c[0]==1\rangle}))]\!] \vee t' \in [\![(\underline{1}_0^q, \mathcal{D}_c(f_{\langle c[0]==0\rangle} \wedge f_{\langle c[0]==1\rangle}))]\!]\big)\}$

$=$ (use of (;) and (nreg). t' can be eliminated because $t' \in [\![...]\!]$ is true)

$\{s| : \exists t'' : s \in [\![(\perp^q, \perp^c)]\!] \wedge t'' \in [\![(0_0^q, \perp^c)]\!]\}$

$=$ (t" can be eliminated because $t'' \in [\![...]\!]$ is true)

$\{s|s \in [\![(\perp^q, \perp^c)]\!]\}$ where s is valid state, finishing the proof. □

5 Conclusions

The paper defined a dynamic logic for a fragment of QASM, combining existent works on dynamic logics for quantum and probabilistic programs and we proved the correctness of a quantum coin toss. However, the logic is still work in progress, being necessary the extension to other examples.

Acknowledgements. The author wishes to thank Luís Barbosa and Leandro Gomes, for the useful discussions during the course of this work. The author was funded by an individual grant of reference SFRH/BD/116367/2016, conceded by the FCT - Fundação para a Ciência e Tecnologia under the POCH programme and MCTES national funds. This work was also supported by the KLEE project(POCI-01-0145-FEDER-030947-PTDC/CCI-COM/30947/2017), funded by ERDF by the Operational Programme for Competitiveness and Internationalisation, COMPETE2020 Programme and by National Funds through the Portuguese funding agency, FCT.

References

[BBK+14] Baltag, A., Bergfeld, J., Kishida, K., Sack, J., Smets, S., Zhong, S.: PLQP & company: decidable logics for quantum algorithms. Int. J. Theor. Phys. **53**(10), 3628–3647 (2014)

[BS04] Baltag, A., Smets, S.: The logic of quantum programs. In: Proceedings of the 2nd International Workshop on Quantum Programming Languages, pp. 39–56 (2004). https://www.mathstat.dal.ca/~selinger/qpl2004/proceedings.html

[CBSG17] Cross, A.W., Bishop, L.S., Smolin, J.A., Gambetta, J.M.: Open quantum assembly language. arXiv preprint arXiv:1707.03429 (2017)

[Deu85] Deutsch, D.: Quantum theory, the Church-Turing principle and the universal quantum computer. Proc. Roy. Soc. Lond. A. Math. Phys. Sci. **400**(1818), 97–117 (1985)