Programming and Simulation of Quantum Search Agents

Matthias Klusch
German Research Center for
Artificial Intelligence
Multiagent System Group
Saarbrücken, Germany
klusch@dfki.de

René Schubotz
Saarland University
Computer Science Department
Saarbrücken, Germany
schubotz@gmx.de

ABSTRACT

Key idea of this work is to appropriately extend one prominent generic agent architecture, namely InteRRap [8], to the case of a quantum pattern matching (QPM) based type-I quantum search agent (QSA) that is supposed to run on a hybrid quantum computer, and to show it's feasibility by instantiating the respective QuantumInteRRap architecture. For a comprehensive and in-depth introduction to quantum computation (QC) we refer the interested reader to [10]. An extended version of this work can be found at [6].

Keywords

Multiagent Systems: Brokering and matchmaking, Agents: Architectures: reactive and deliberative

1. QC DESIGN FLOW

In [5] a master-slave architecture of a *hybrid quantum computer* is proposed. The CPU of a *classical machine* (CM) performs high-level dynamic control and scheduling of quantum operations carried out in a *quantum machine* (QM). The QM consists of quantum memory [5], quantum processing unit (QPU) with error correction, quantum bus, and quantum device controller (QDC) with interface to the CM. Using a high-level *quantum programming language*[11, 2, 12] (QPL) quantum algorithms are represented by QPL code containing high-level primitives for logical quantum operations, interleaved with classical work-flow statements. A layered software architecture[13] involving a four-phase computer-aided design flow assists by mapping such QPL sources into efficient and robust physical implementations. During the flow's first phase, the QPL source is transformed into a *quantum intermediate representation* (QIR) by the CM. Then, the CPU synthesizes and optimizes a *Quantum Assembly Language* (QASM) representation of a suitable quantum circuit. In the third phase, the QASM instructions are compiled into a device-specific *Quantum Physical Operations Language* (QPOL) representation. Finally, QPOL code is translated by the QDC into quantum machine instructions that are passed to and executed by the QPU. The QPU performs scheduled sequences of measurement and basic qubit operations with error correction. The QM returns the results of finalising quantum measurements to the CM.

AAMAS'07, May 14–18, 2007, Honolulu, Hawai'i, USA.

2. GENERIC QC AGENT ARCHITECTURE QUANTUMINTERRAP

In [8], an architecture for multi-agent systems is presented. The proposed *InteRRap* combines both the reactive and the deliberate paradigm, and explicitly represents knowledge, plans and strategies. In the following, the InteRRap architecture is embedded in the context of QC multi-agent systems and the high-level components of the *QuantumInteRRap* architecture and their basic interplay are explained. In detail, the *WIF* provides the agent's means for perception, actuation and communication. According to the classification of QC agents in [5], the WIF of *type-I QC agents* solely provides classical communicative facilities, whereas a *type-IIa QC agent*'s WIF is solely equipped with facilities for direct quantum particle transmission, and a *type-IIb QC agent*'s WIF bears both classical and quantum communicative facilities. In consequence of the deployed communicative facilities, *type-I QC agents* do not suffer from incomplete knowledge due to not yet measured qubits. *Type-IIb QC agents* may obtain quantum knowledge by classical communication assuming a non-antagonistic agent scenario, whilst *type-IIa QC agents* are inherently uninformed about received but not yet measured qubits. The *BBL* implements the agent's basic behaviour and is linked to the WIF for communication and actuation. The BBL has access to a set of executable *patterns of behaviour* (PoB) which can be activated by external stimulus or by the LPL. PoB divide into basic reactive short term actions and pieces of procedural knowledge that are appropriate to stimulate *quantum patterns of behaviour* (qPoB). Such a qPoB may be the preparation of quantum states, or the synthesis and optimization of quantum primitives resulting in QASM code fragments (section 1). The *LPL* has access to a standard from-scratch planning algorithm and to a plan library consisting of classical hierachical single-agent skeletal plans and QPL codes of *generic quantum algorithms.* A choosen plan may include PoB and *quantum deliberation.* In case of quantum deliberation, a quantum algorithm is instantiated by transformation of the appropriate QPL code into valid QPOL including synthesized QASM fragments and executed on a QM under

978-81-904262-7-5 (RPS)

control of the LPL. In consequence of the deployed communication facilities, *type-II QC agents* may perform quantum operations on received qubits, whereas *type-I QC agents* are restricted to self-prepared qubits. The *CPL* implements a mechanism for devising joint plans, has a notion of protocols and communication strategies and access to a joint plan library. In contrast to *type-I QC agents*, the CPL of *type-II QC agents* is able to devise joint plans involving quantum communication and quantum cooperation strategies. To this end, QPL representations of respective quantum concepts are transformed to valid QPOL and executed on the QM which is at the disposal of the type-II QC agent considered.

3. QUANTUM PATTERN MATCHING

The problem of determining the *closest match* with a given pattern $p \in \Sigma^M$ of size $M \ll N$ in an unstructured database string $w \in \Sigma^N$ of size N has been solved in [7] by extending Grover's search algorithm [4]. The query complexity of QPM has been proven to be $O(\sqrt{N-M})$ allowing for a significant speedup by an order of magnitude compared to classical matching approaches such as *approximate swapped matching*[1] in $O(N \log(M) \log(\min(M, |\Sigma|)))$. Applying a *compile once, run many* approach, the QPM algorithm enables to search for an arbitrary large number of distinct patterns in a given database. To this end, the i-th position of w is encoded by $|i\rangle \in H_2^{\otimes \lceil log(N) \rceil}$. Hence, the quantum state

$$|\chi\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle$$

superposes all positions of database string w. For each symbol $\sigma \in \Sigma$, a symbol oracle Q_σ is given by

$$Q_\sigma |\chi\rangle = (\mathbf{1} - 2 \sum_{1 \le j \le N} |j\rangle \langle j|) |\chi\rangle$$

for positions j of w satisfying $w_j = \sigma$. The algorithm is constituted by iterating the phase shifts induced by a symbol oracle Q_{σ_l} for a *random symbol* σ_l of the pattern followed by Grover's operator

$$P_\psi |\chi\rangle = (2 |\psi\rangle \langle\psi| - 1) |\chi\rangle, \quad |\psi\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle$$

In order to apply the effects of Q_{σ_l} to the correct starting positions $|k\rangle$ of database substrings $\langle w_k ... w_l ... w_{k+M-1} \rangle$, states $|k\rangle$ corresponding to positions of w need to be permuted in each iteration. This can be done reversibly using

$$P_\pi^l |\chi\rangle = P_\pi^l \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} |k\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} |k + l \mod N\rangle$$

If $M \ll N$, sampling randomly over M elements will lead to searching with high probability over all symbols of the pattern. On average, a position with a partial match of $M' \le M$ individual symbols will experience $\frac{M'}{M}$ phase shifts.

4. TYPE-I QUANTUM SEARCH AGENT

How to model a type-I quantum search agent (QSA) by use of the QuantumInteRRap architecture presented in section 2? The QSA is supposed to run on a hybrid quantum computer which is, for example, provided with a unstructured classical *local database* (LDB). For the sake of simplicity, we consider this LDB as a long static string $w \in \Sigma^N$ of size N. Figure 1 illustrates how a QPM based *Type-I QSA* can be modeled as a *QuantumInteRRap* agent.

Algorithm 1 Quantum pattern matching

Input: $w \in \Sigma^N, p \in \Sigma^M, n = \lceil \log(N) \rceil$
Output: $m \in \mathbb{N}$
Quantum Variables: $|\chi\rangle \in H_2^{\otimes n}$
Classic Variables: $r, i, j \in \mathbb{N}$

1: Choose $r \in \left[0, \lfloor \sqrt{N - M + 1} \rfloor\right]$ uniformly
2: Set $|\chi\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} |k\rangle$
3: **for** $i = 1$ to r **do**
4: Choose $j \in [0, M-1]$ uniformly
5: Set $|\chi\rangle = P_\psi (P_\pi^j)^{-1} Q_{\sigma_j} P_\pi^j |\chi\rangle$
6: **end for**
7: Set m to the result of the measurement of $|\chi\rangle$

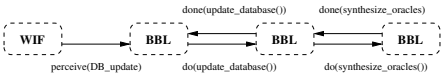

Figure 2: Processing a database update

Once a setup or update of LDB is perceived (see figure 2), the symbol oracles Q_σ for $\sigma \in \Sigma$ need to be synthesized by the BBL using a QASM compiler on the basis of section 3. The resulting quantum circuits are stored in QASM source for further usage.

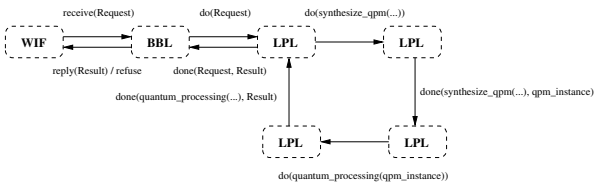

Figure 3: Processing a pattern request

Upon receipt of request s containing some pattern $p \in \Sigma^M$ from another agent A_1 (see figure 3), the QSA locally computes the index of the closest match m to p in LDB and returns m to A_1. Both requests and replies are transmitted via classical channel, and requests are directly quantum coded (see section 1) prior to quantum pattern matching. In more detail, the BBL asks the LPL to process the transmitted pattern $p \in \Sigma^M$. To this end, the LPL devises and executes a local plan involving a proper instance of the QPM algorithm. The instantiation includes necessary symbol oracles computed by the BBL into the skeletal QPM algorithm scheme. The resulting QASM code makes use of qPoB to prepare initial quantum states, for example. It is transformed into valid QPOL and transmitted to the QDC. After instructing the QPU to perform a quantum pattern matching, the QDC returns the measurement result back to the LPL. By passing the result to lower level of the world interface, the requesting agent A_1 receives the computed closest

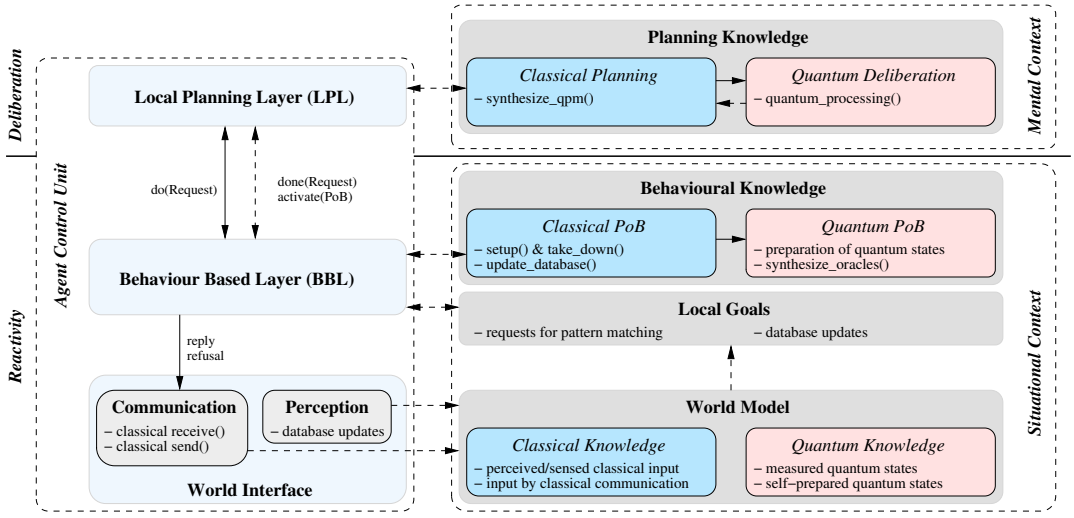

Figure 1: QuantumInteRRap architecture of a QPM based type-I quantum search agent

match to its query string, or a failure message in case of the search failed.

5. SIMULATION AND BENCHMARKING

To demonstrate the operability of our type-I QSA, we have programmed and simulated it by use of open source quantum simulators [14, 11, 3]. Memory and runtime performances for simulations on CM for various query patterns of size $|p| = 4$ and database strings are given in the extended version of this work [6].

6. CONCLUSION

We presented a hybrid generic architecture for QC agents as an extension of the InteRRap architecture, discussed basic engineering aspects of how to realize QC agents on hybrid quantum computers, instantiated the QuantumInteRRap architecture by concrete means of a QPM based type-I quantum search agent, showed results of comparative performance testing of its simulation with different quantum simulators, and thus demonstrated its functionality by example. The theoretical performance of the QPM based QSA over classical edit based string matching provides strong evidence for the expected significant speed up of a service matchmaking process of type-I quantum matchmaker agents compared to the classical case. To the best of our knowledge, the presented architecture, programming and simulation of a type-I QSA is unique as there are no alternative QC agent implementations available yet. Our work combines related work from quantum computing and agent based computing, and builds upon existing work on QC agents in [5]. However, the quantum realization of semantic service matchmaking remains one open problem. Our ongoing research focuses on solving this problem.

7. REFERENCES

[1] Amir. Approximate swapped matching. *Inf. Process. Lett.*, 83(1):33–39, 2002.

[2] Bettelli. Toward an architecture for quantum programming, eur. phys. j., 25:181–200, 2003., 2003.

[3] B. Butscher. Non-technical description of libquantum, enyo.de/libquantum/.

[4] L. K. Grover. A fast quantum mechanical algorithm for database search, arxiv.org/quant-ph/9605043, 1996.

[5] M. Klusch. Toward quantum computational agents. In Nickles et al. [9], pages 170–186.

[6] M. Klusch and R. Schubotz. Programming and simulation of quantum search agents, www.dfki.de/ klusch/publications/qsa.pdf, 2007.

[7] P. Mateus and Y. Omar. Quantum pattern matching, arxiv.org/quant-ph/0508237. 2005.

[8] J. Müller and M. Pischel. The agent architecture interrap: Concept and application, technical report rr-93-26, dfki saarbrucken, 1993, 1993.

[9] M. Nickles, M. Rovatsos, and G. Weiß, editors. *Agents and Computational Autonomy (AAMAS 2003).*

[10] M. A. Nielsen and I. L. Chuang. *Quantum computation and quantum information.* Cambridge Univ. Press, Cambridge, 2000.

[11] B. Oemer. Quantum programming in qcl, master thesis, technical university of vienna, computer science department, 2000.

[12] P. Selinger. Towards a quantum programming language. *Mathematical. Structures in Comp. Sci.*, 14(4):527–586, 2004.

[13] K. M. Svore, A. V. Aho, A. W. Cross, I. Chuang, and I. L. Markov. A layered software architecture for quantum computing design tools. *Computer*, 39(1):74–83, 2006.

[14] Viamontes. Gate-level simulation of quantum circuits, arxiv.org/quant-ph/0208003, 2002.