Quantum-Based SMT Solving for String Theory

Beatrice Casey, Joanna C. S. Santos, Andrew Hennessee
{bcasey6, joannacss, ahennes3}@nd.edu
University of Notre Dame, Department of Computer Science

PRESENTER:
Beatrice Casey
bcasey6@nd.edu

Overview

Satisfiability Modulo Theory (SMT) solvers are a useful tool that can be applied to a variety of problems, such as configuring relationships in distributed systems, detecting race conditions, and program analysis. String constraints are particularly difficult for SMT solvers to navigate, as the search space is generally large.

Often times, classical SMT solvers will have to quit generating a solution for string constraints because it takes too long to find the solution. In this work, we explore creating **a quantum-enabled SMT solver for string theory by using quantum annealing and Quadratic Unconstrained Binary Optimization (QUBO)**.

Our preliminary results demonstrate that it is feasible to transform these string constraints to QUBO, and generate solutions for given constraints

Background

SMT Solvers

- They determine whether logical formulas are ***satisfiable*** (i.e., have a valid assignment) under certain theories such as: *Arithmetic* (integers, reals), *Arrays*, *Bit-vectors*, *Uninterpreted functions*, etc.
- They extend SAT solvers by supporting richer expressions beyond Boolean logic.
- Example: for the logical formula below, an SMT solver (e.g., Z3) could be used to determine that this formula is unsatisfiable (**UNSAT**) in integer arithmetic.

$$x > 0 \wedge y = x + 2 \wedge y < 0$$

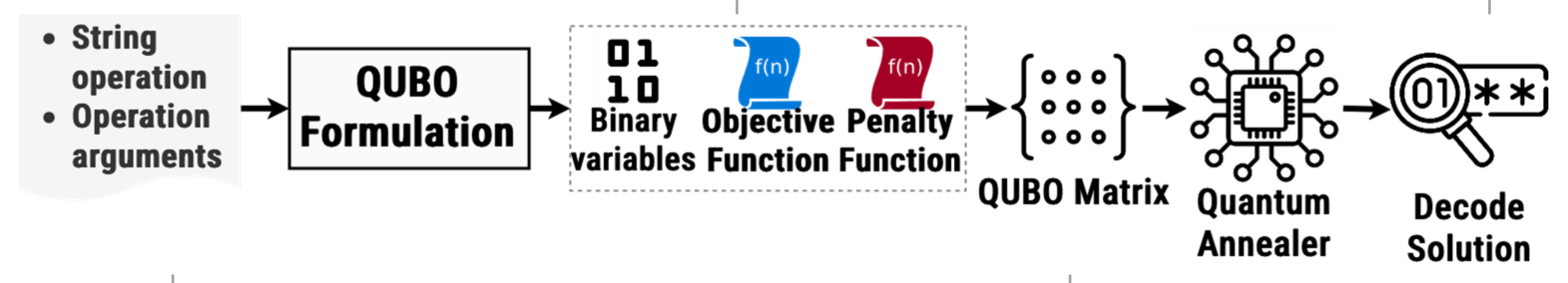
Quadratic Unconstrained Binary Optimization (QUBO)

- It is a framework for optimization problems defined by binary variables, an objective function, optional penalty functions, and a QUBO matrix.
- It is particularly useful for mapping problems onto quantum hardware (particularly quantum annealers), which are designed for combinatorial problems.

Research and Results

We define the necessary binary variables, objective, and penalty functions for each supported string constraint

After running the QUBO matrix through a quantum (or simulated) annealer, we decode the solution and get the resultant string



Our approach supports the following constraints: string equality, substring matching, substring indexOf, string length, string replace and replace all, string reversal, string concatenation, string includes, palindrome generation, and regex matching.

We encode the extracted features/transformed constraints into our QUBO Matrix

Table 1: Results from our approach to sample string constraints. The matrices are abbreviated due to space limitations.

Constraint	Matrix	Output
Reverse ‘hello’ and replace ‘e’ with ‘a’	$\begin{bmatrix} -1 & 0 & 0 & \cdots & 0 \\ 0 & -1 & 0 & \cdots & 0 \\ 0 & 0 & -1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & -1 \end{bmatrix}$	ollah
Generate a palindrome with length 6	$\begin{bmatrix} 1.00 & 0.00 & 0.00 & \cdots & -2.00 & 0.00 & 0.00 \\ 0.00 & 1.00 & 0.00 & \cdots & 0.00 & -2.00 & 0.00 \\ 0.00 & 0.00 & 1.00 & \cdots & 0.00 & 0.00 & -2.00 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0.00 & 0.00 & 0.00 & \cdots & 1.00 & 0.00 & 0.00 \\ -2.00 & 0.00 & 0.00 & \cdots & 0.00 & 1.00 & 0.00 \\ 0.00 & -2.00 & 0.00 & \cdots & 0.00 & 0.00 & 1.00 \end{bmatrix}$	OnFFnO
Generate the regex a[bc]+ with length 5	$\begin{bmatrix} -2.00 & 0.00 & 0.00 & \cdots & 0.00 \\ 0.00 & 2.00 & 0.00 & \cdots & 0.00 \\ 0.00 & 0.00 & 2.00 & \cdots & 0.00 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0.00 & 0.00 & 0.00 & \cdots & -1.00 \end{bmatrix}$	abcbb
Concatenate ‘hello’ and ‘ world’, and replace all ‘l’ with ‘x’	$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & -1 \end{bmatrix}$	hexxo worxd
Generate a string of length 6 that contains the substring ‘hi’ at index 2	$\begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 0 & 2 & 0 & \cdots & 0 \\ 0 & 0 & 6 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & -42 \end{bmatrix}$	qphiqp

- Our results demonstrate that our approach successfully encodes each constraint problem into the QUBO matrix, and generates the expected output.
- We can transform string accurately and generate strings with structural constraints, in the instances of palindromes and regular expressions.
- We can also generate ’flexible’ examples, even when enforcing substring positions. This means that our method can generate a unique string, while still enforcing the necessary constraints.

State of the Art

While recent works [1,2,3] focused on quantum software engineering and the way it will change from classical software engineering, there is no work that focuses on applying quantum computing to SMT solving for string theory. One work investigates the use of quantum computing, specifically Grover’s Algorithm, for SMT solving for bit-vector theory[4]. Our work differs by focusing on string theory and using quantum annealing and QUBO.

Future Work

This work lays the foundation for quantum-enabled SMT solving. While we tested our work on a simulated annealer, future works involve testing our existing constraints with a quantum annealer, extending to cover more string constraints, and implementing this framework as part of a system that requires SMT solving (such as a symbolic execution framework).

Conclusion

This work demonstrates the feasibility of using QUBO formulations and quantum annealing to solve complex string constraints in SMT problems. By supporting a variety of string operations, some of which are difficult for classical solvers, we show that quantum-based approaches can expand the capabilities of SMT solving. This opens the door to more efficient techniques for problems such as program analysis, and demonstrates a first step towards using quantum computing to address software engineering problems.

References

[1] Zhao, J. “Quantum Software Engineering: Landscapes and horizons,” arXiv 2021.
[2] Arias, S. *et al.*, “Let’s do it right the first time: Survey on security concerns in the way to Quantum Software Engineering,” *Neurocomputing*, vol. 538, p. 126199, Jun. 2023. doi:10.1016/j.neucom.2023.03.060
[3] Dwivedi, K. *et al.* “Quantum Software Engineering and Quantum Software Development Lifecycle: A survey,” *Cluster Computing*, Mar. 2024. doi:10.1007/s10586-024-04362-1
[4] Shang-Wei L., *et al.* “A Quantum SMT Solver for Bit-Vector Theory,” arXiv 2023

Acknowledgement

This material is based upon work supported by the Center for Quantum Technologies under the Industry-University Cooperative Research Center Program at the US National Science Foundation under Grant No. 2224985.