Accelerating Quantum Circuit Simulation with Symbolic Execution and Loop Summarization

Tian-Fu Chen[1], Yu-Fang Chen[2], Jie-Hong R. Jiang[1],
Sára Jobranová[3], Ondřej Lengál[3]

(1) National Taiwan University, Taiwan
(2) Academia Sinica, Taiwan
(3) Brno University of Technology, Czech Republic

October 28, 2024

Motivation

- Unavailability of quantum computers – HW still experimental
- In a real system, we cannot directly inspect the probability amplitudes – only qubit measurement (irreversible state collapse)
- Performance of state-of-the-art simulators is unsatisfactory for complex circuits

Overview

1. MTBDD-based simulator MEDUSA
   - MTBDD (Multi-terminal binary decision diagram) - a simple structure so operations are easy to perform, but no unnecessary overhead as with BDDs
   - Efficient gate application
2. Loop summarization technique
   - Significantly faster loop execution with a fixed number of iterations

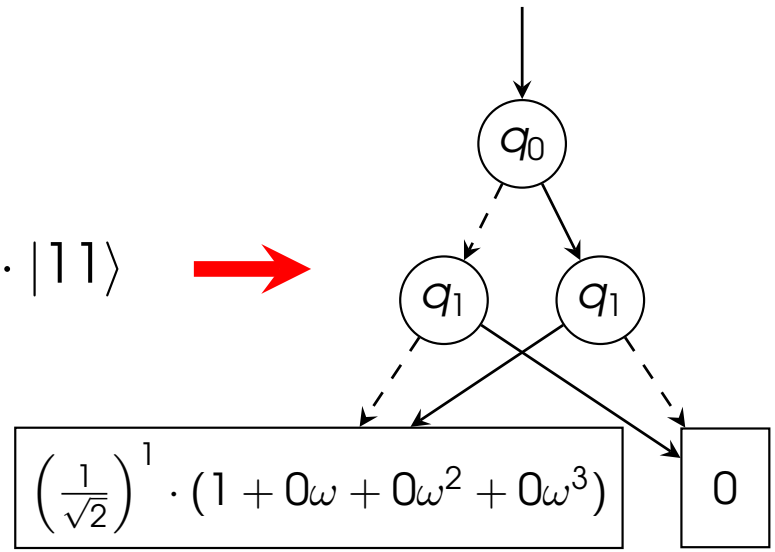
Introducing MEDUSA

- The state vector is represented by an MTBDD
- Uses the DD library Sylvan[1]
- Leaves – probability amplitudes (complex numbers)
- Exact simulation – algebraic representation for $z \in \mathbb{C}$:

$$z = \left(\frac{1}{\sqrt{2}}\right)^k \cdot (a + b\omega + c\omega^2 + d\omega^3),$$

where $a, b, c, d, k \in \mathbb{Z}$, and $\omega = e^{\frac{i\pi}{4}}$ [2]

$$|q_1 q_0\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{pmatrix} = \frac{1}{\sqrt{2}} \cdot |00\rangle + \frac{1}{\sqrt{2}} \cdot |11\rangle$$

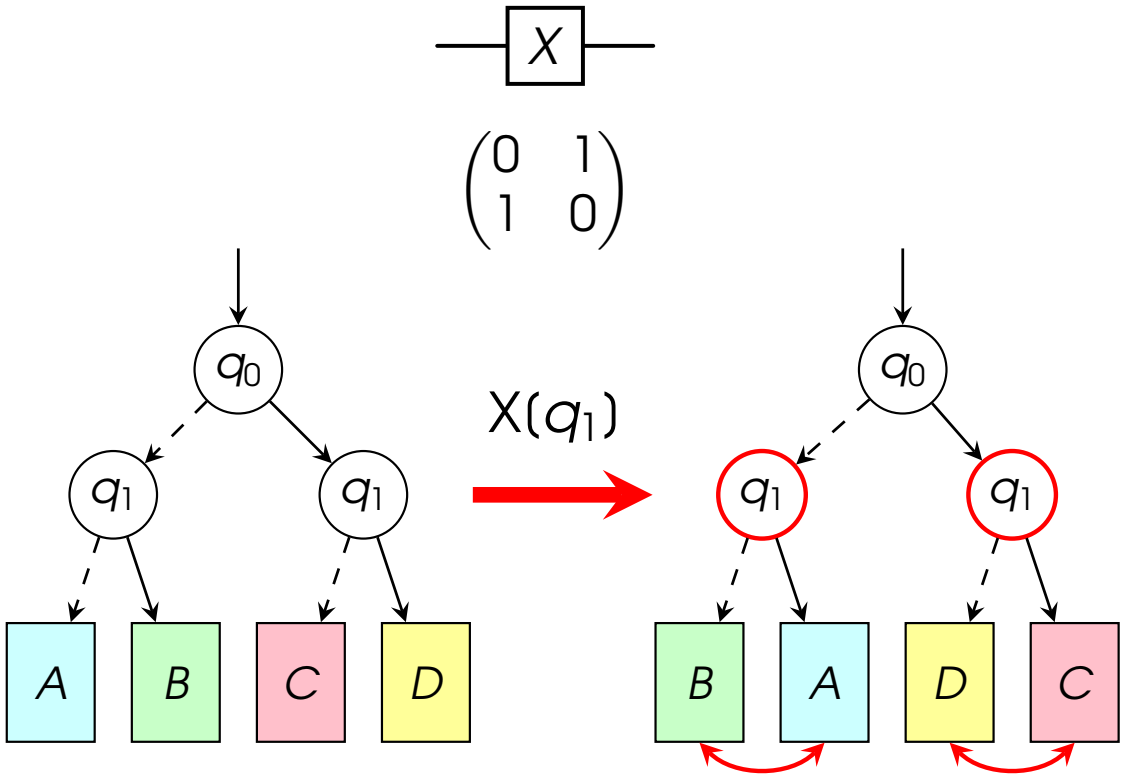


(1) Dijk, T. van and Pol, J. van de. Sylvan: Multi-Core Decision Diagrams. TACAS 2015

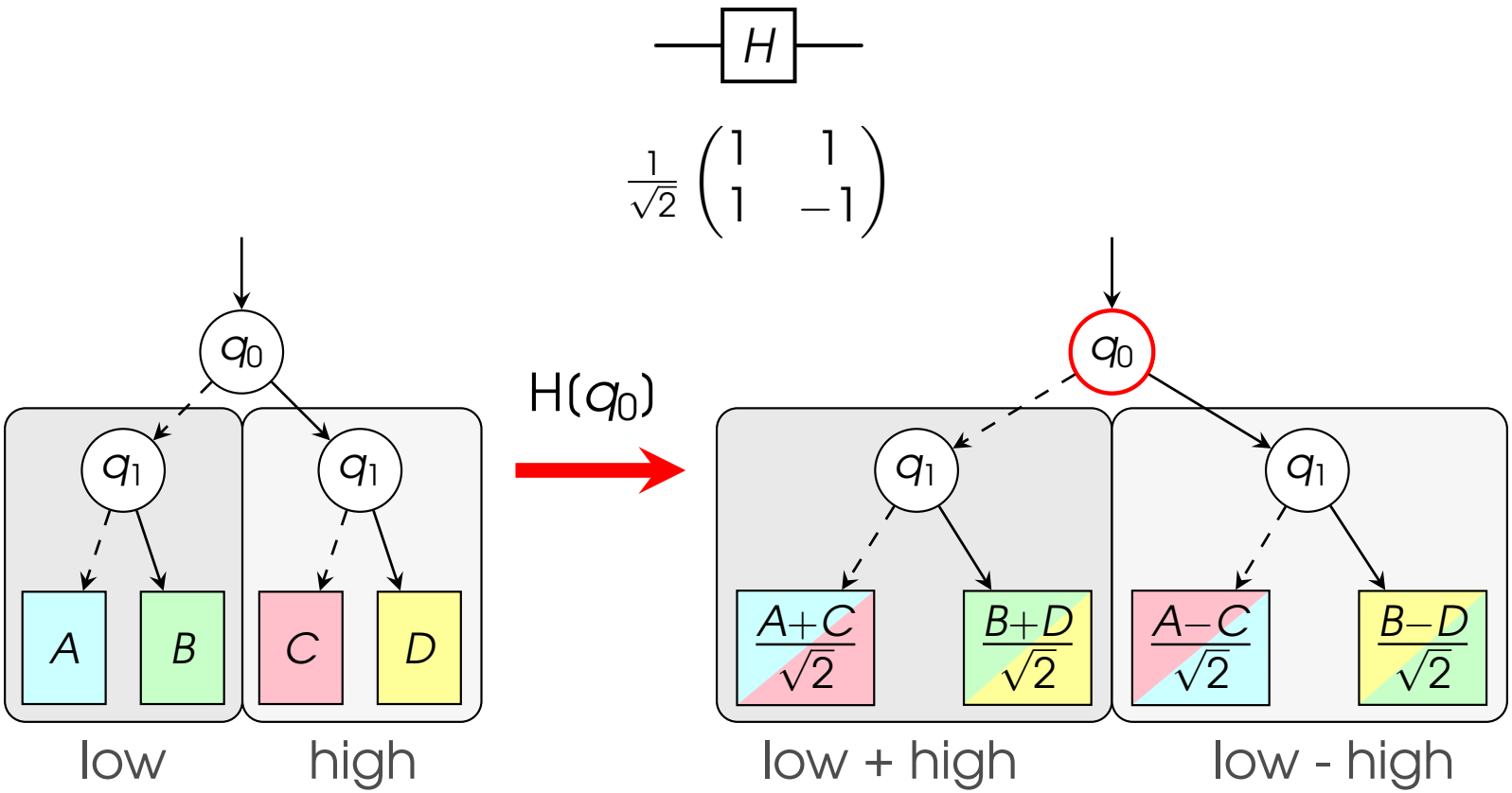
(2) Niemann, P., Zulehner, A., Drechsler, R. and Wille, R.: Overcoming the Tradeoff Between Accuracy and Compactness in Decision Diagrams for Quantum Computation. TCAD 2020

Gate Application

- Special MTBDD procedures – not only the standard MTBDD *Apply*, *Restrict* interface

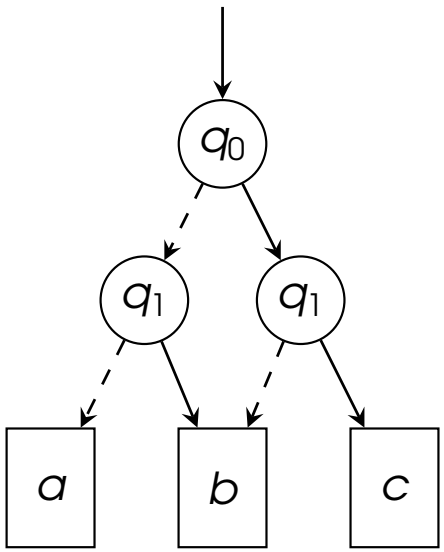


Gate Application

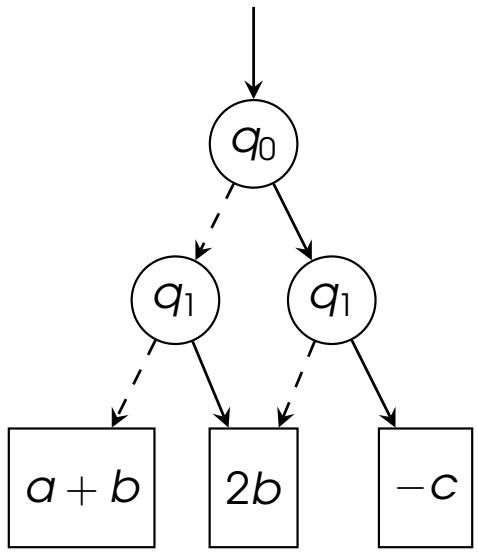


Symbolic Loop Execution

- Simulation done with a pair of symbolic MTBDDs
- Calculation of big-step semantics of loops – significant speedup (no need to re-evaluate gates)
- Loops are often a key part of quantum algorithms (e.g., Grover's search)



(a) Loop start μ_α

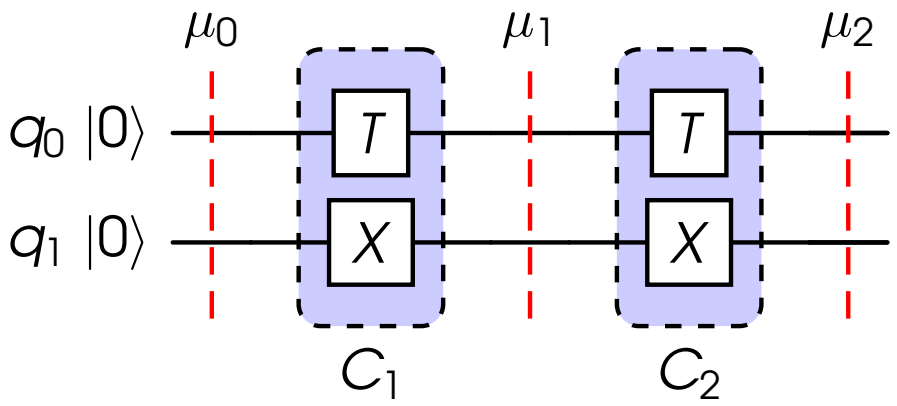


(b) Loop end μ'_α

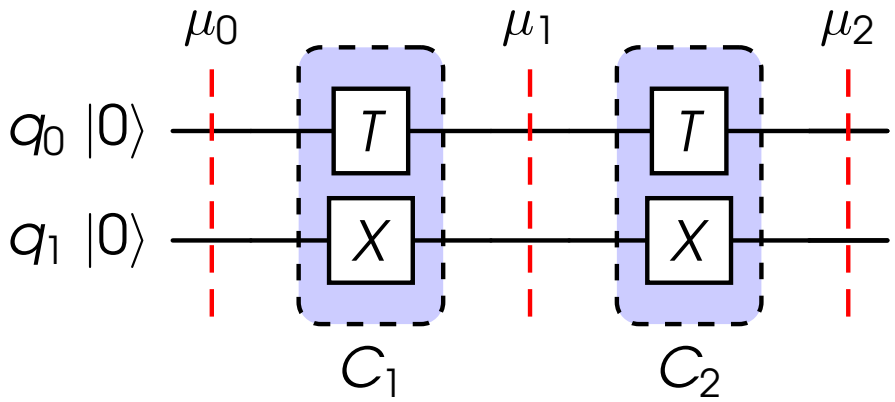
H[0]; . . . ; H[$n-1$];
for int i in [1, m] **do**
    Apply oracle;
    H[0]; . . . ; H[$n-1$];
    Apply phase shift;
    H[0]; . . . ; H[$n-1$];

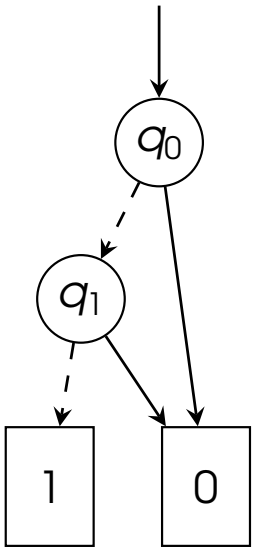
Alg: Grover's search

Symbolic Loop Execution – Example



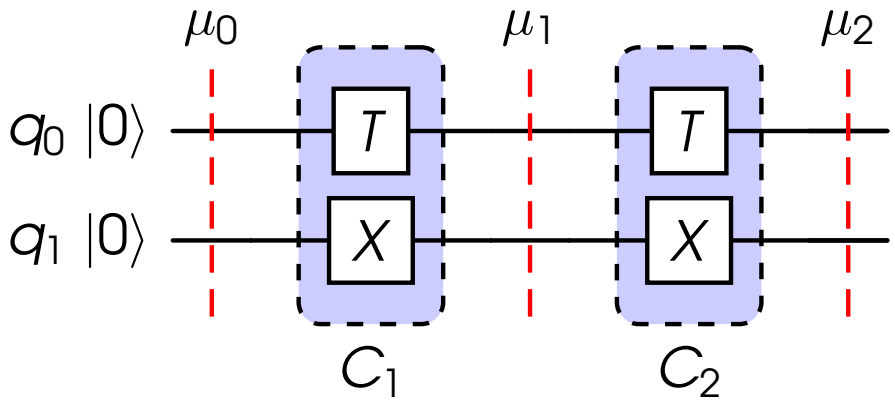
Symbolic Loop Execution – Example

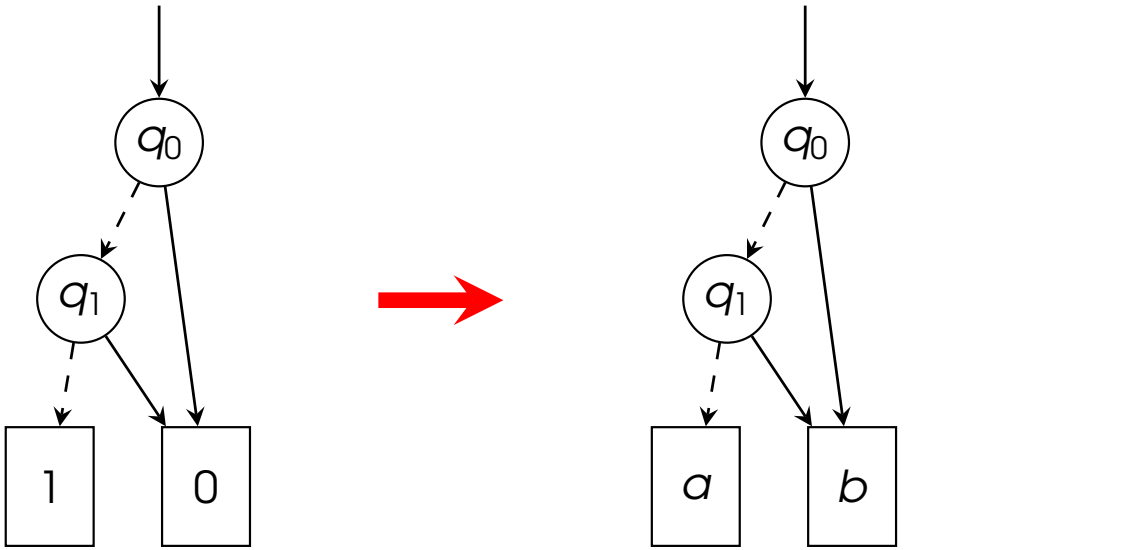




(a) Initial state μ_0

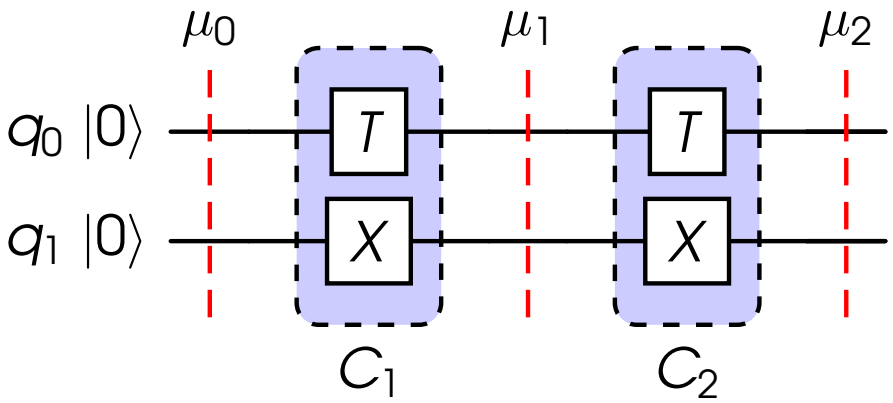
Symbolic Loop Execution – Example

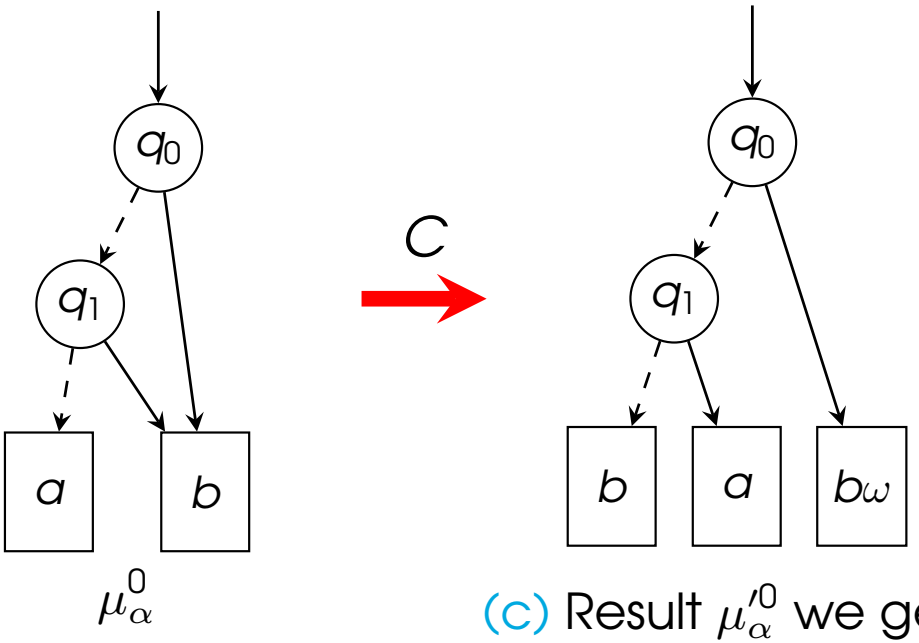




(a) Initial state μ_0 (b) Initial abstraction μ_α^0 of μ_0

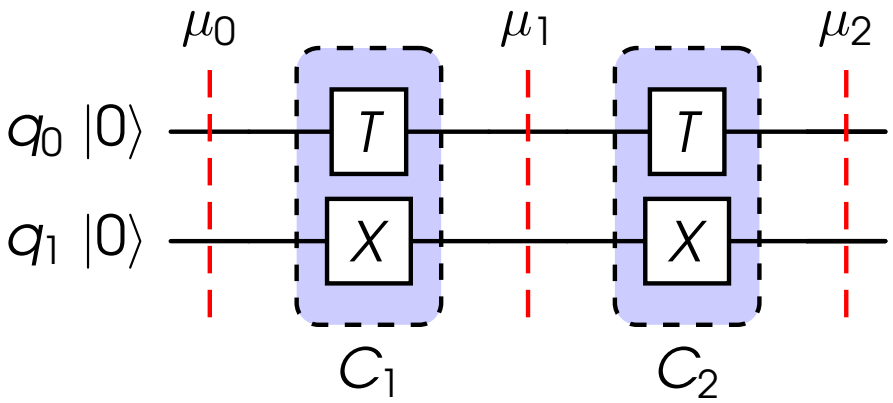
Symbolic Loop Execution – Example

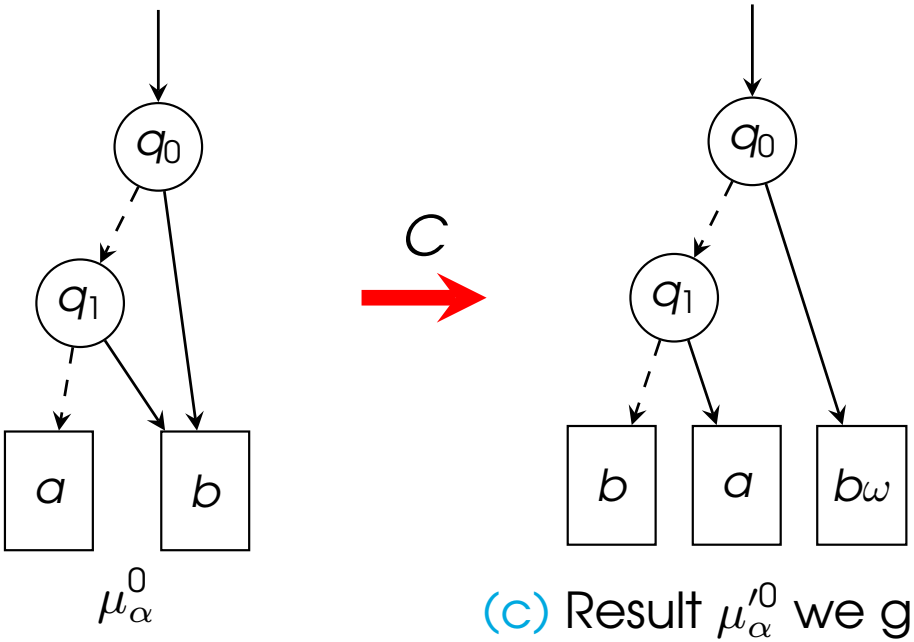




(c) Result $\mu_\alpha'^0$ we get after applying C to μ_α^0

Symbolic Loop Execution – Example



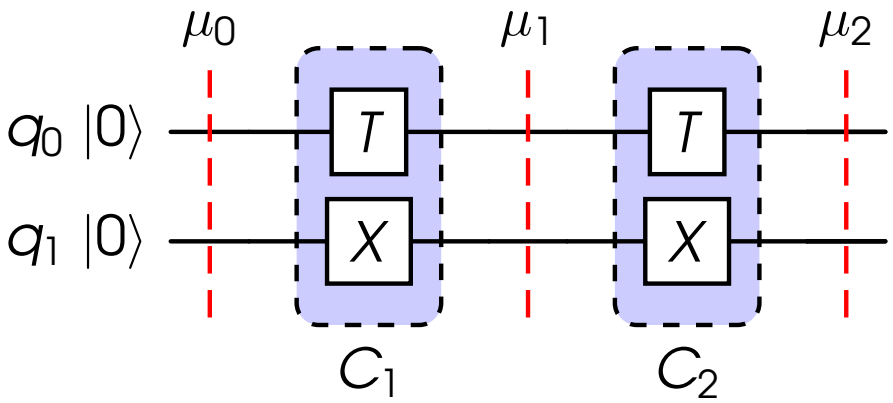


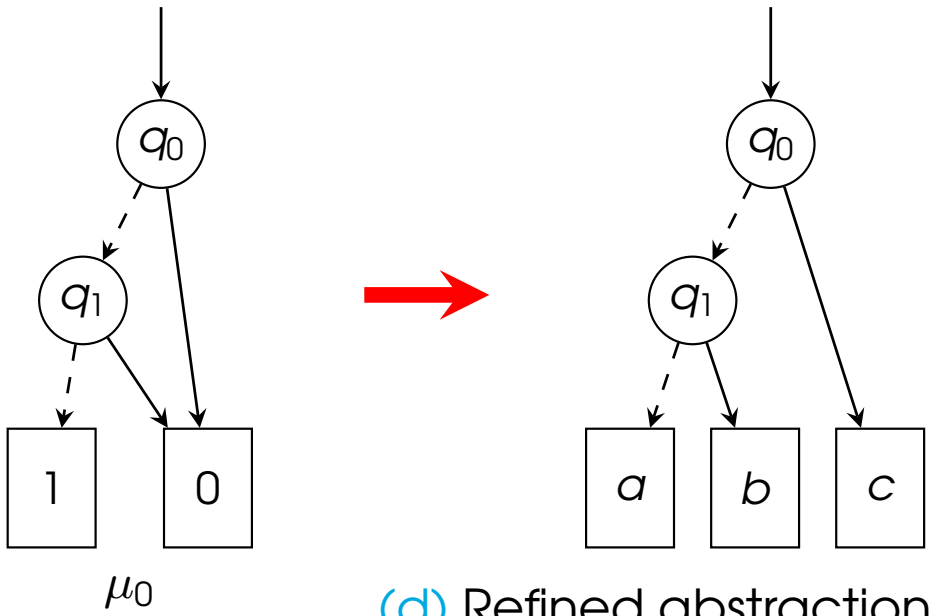
$$a_{i+1} \leftarrow b_i$$
$$b_{i+1} \leftarrow a_i$$
$$b_{i+1} \leftarrow b_i \cdot \omega$$

✘

(c) Result $\mu_\alpha'^0$ we get after applying C to μ_α^0

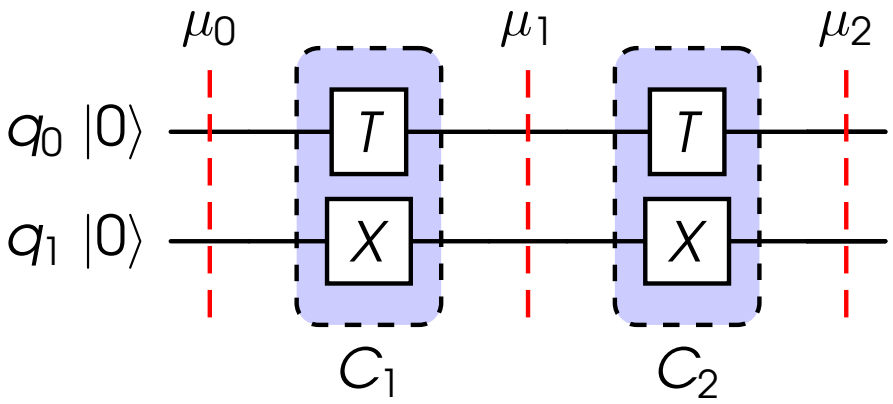
Symbolic Loop Execution – Example

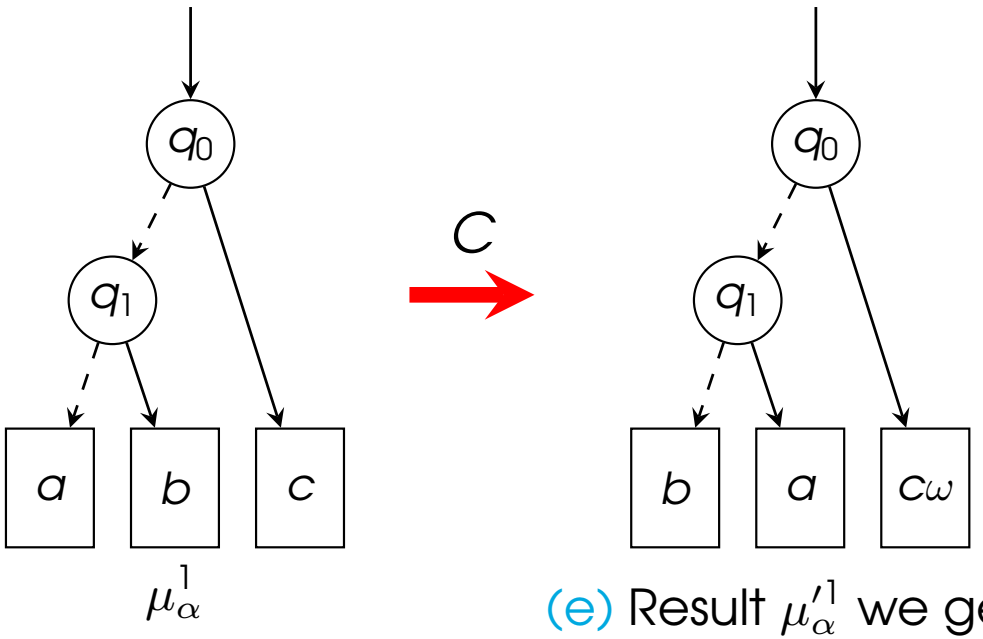




(d) Refined abstraction μ_α^1 of μ_0

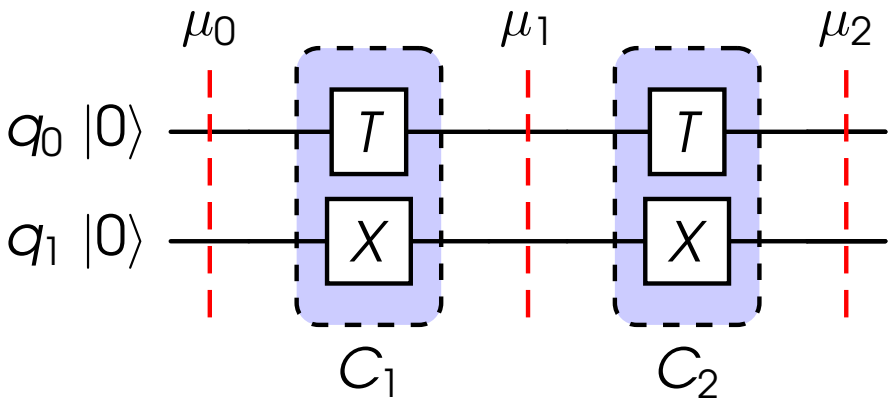
Symbolic Loop Execution – Example

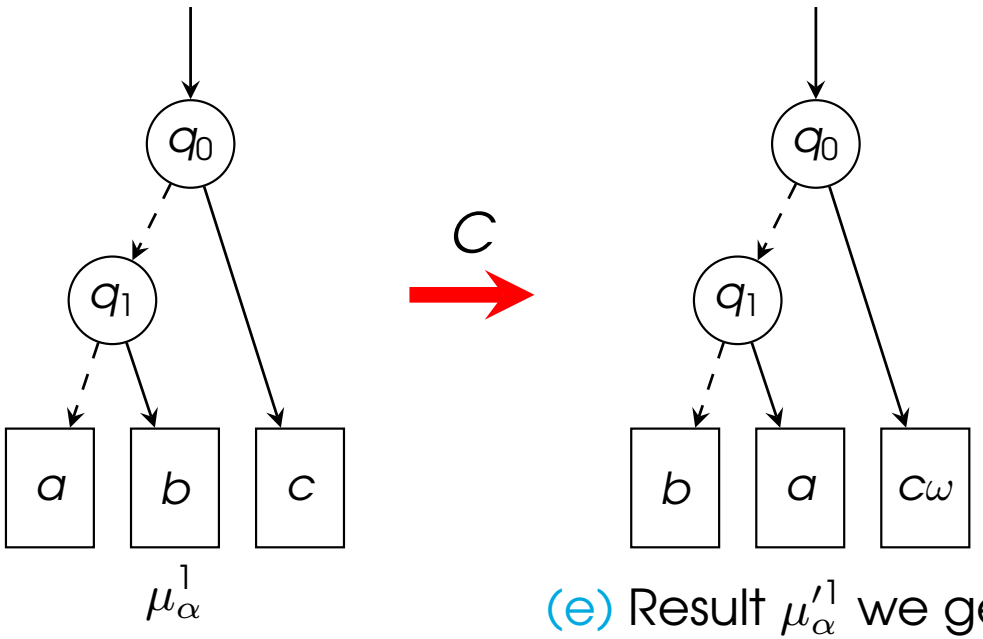




(e) Result $\mu_\alpha'^1$ we get after applying C to μ_α^1

Symbolic Loop Execution – Example



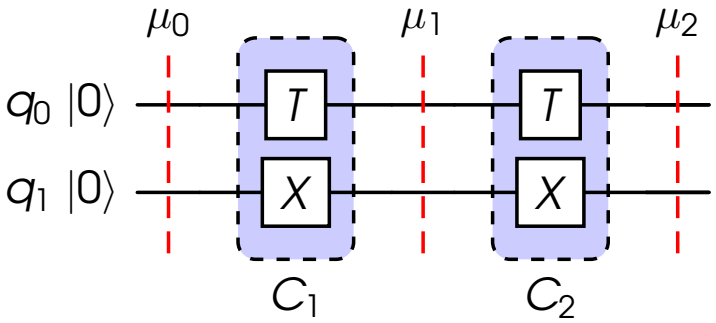


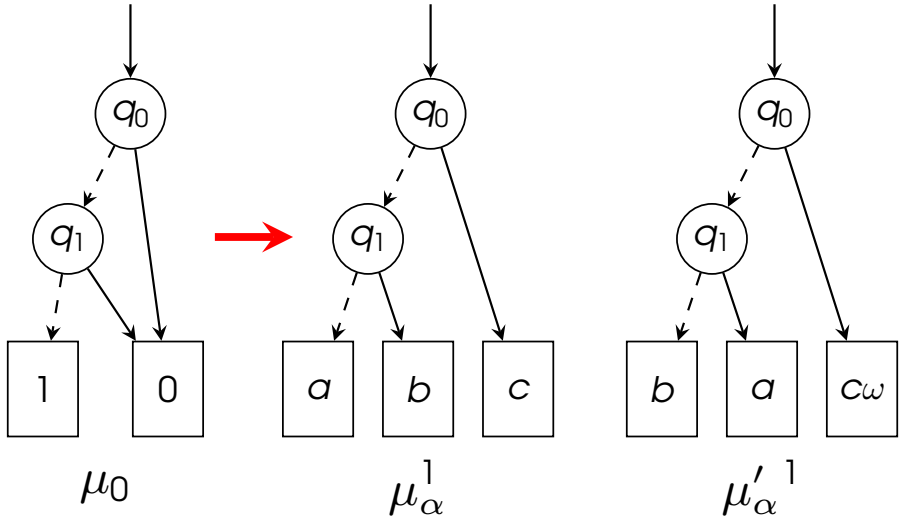
$$
\begin{aligned}
a_{i+1} &\leftarrow b_i \\
b_{i+1} &\leftarrow a_i \\
c_{i+1} &\leftarrow c_i \cdot \omega
\end{aligned}
$$

✓

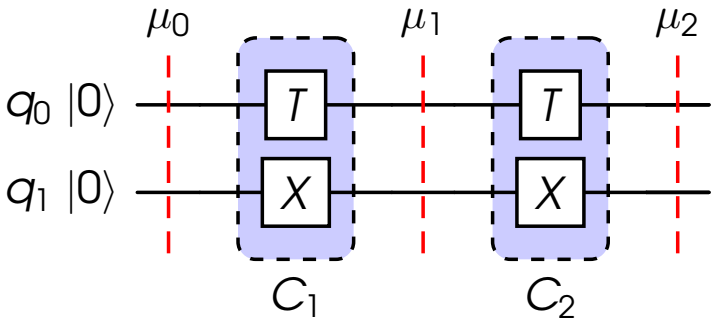
(e) Result $\mu_\alpha'^1$ we get after applying C to μ_α^1

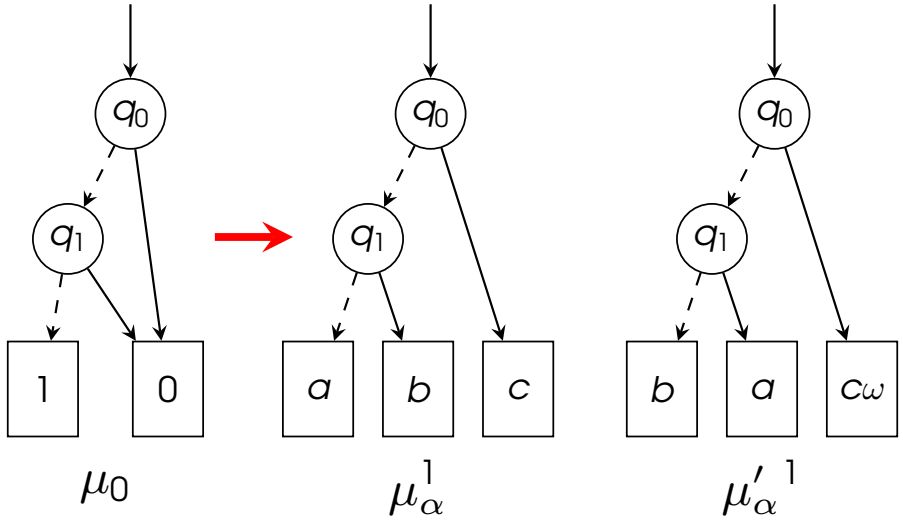
Symbolic Loop Execution – Example

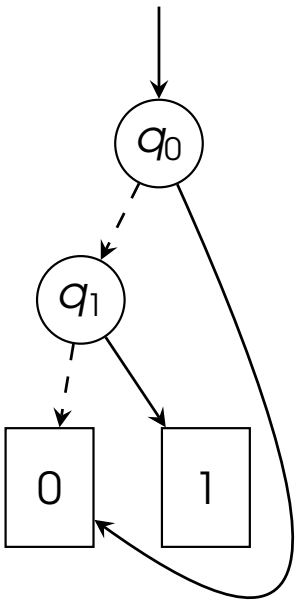




Symbolic Loop Execution – Example



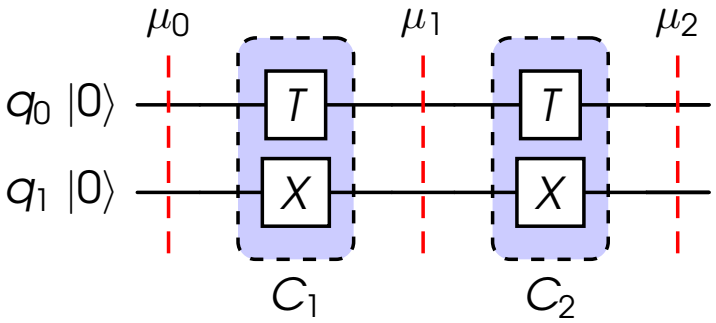


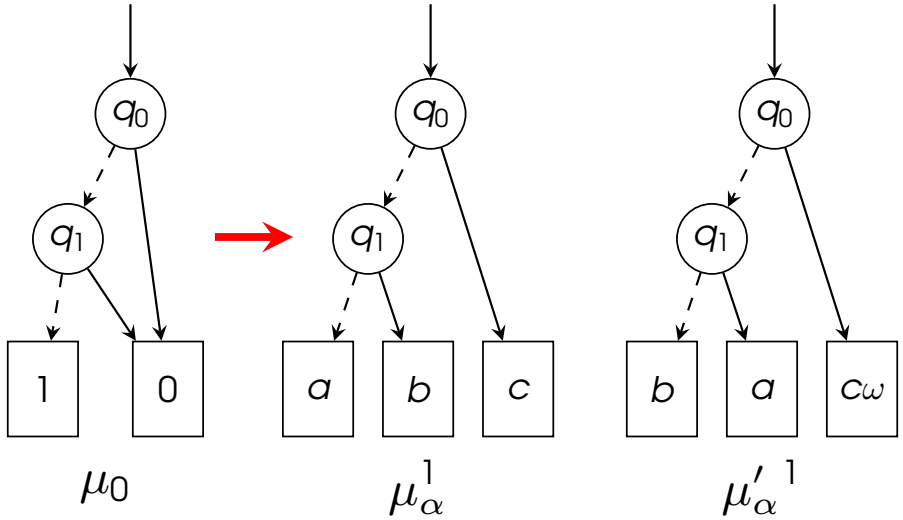


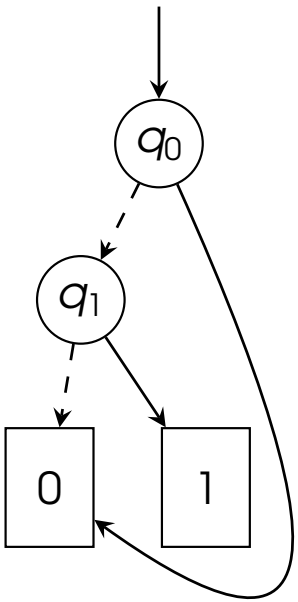
$$
\begin{aligned}
a_1 &\leftarrow b_0 && (=0)\\
b_1 &\leftarrow a_0 && (=1)\\
c_1 &\leftarrow c_0 \cdot \omega && (=0)
\end{aligned}
$$

(f) Evaluated result μ_1

Symbolic Loop Execution – Example

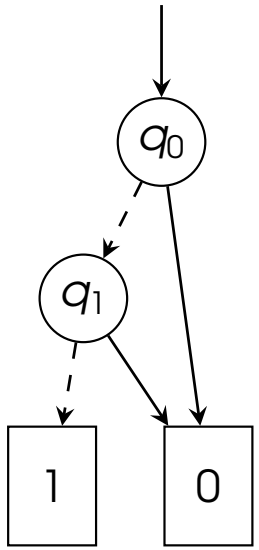






$$
\begin{aligned}
a_1 &\leftarrow b_0 && (=0)\\
b_1 &\leftarrow a_0 && (=1)\\
c_1 &\leftarrow c_0 \cdot \omega && (=0)
\end{aligned}
$$

(f) Evaluated result μ_1



$$
\begin{aligned}
a_2 &\leftarrow b_1 && (=1)\\
b_2 &\leftarrow a_1 && (=0)\\
c_2 &\leftarrow c_1 \cdot \omega && (=0)
\end{aligned}
$$

(g) Final result μ_2

Results of MEDUSA – base

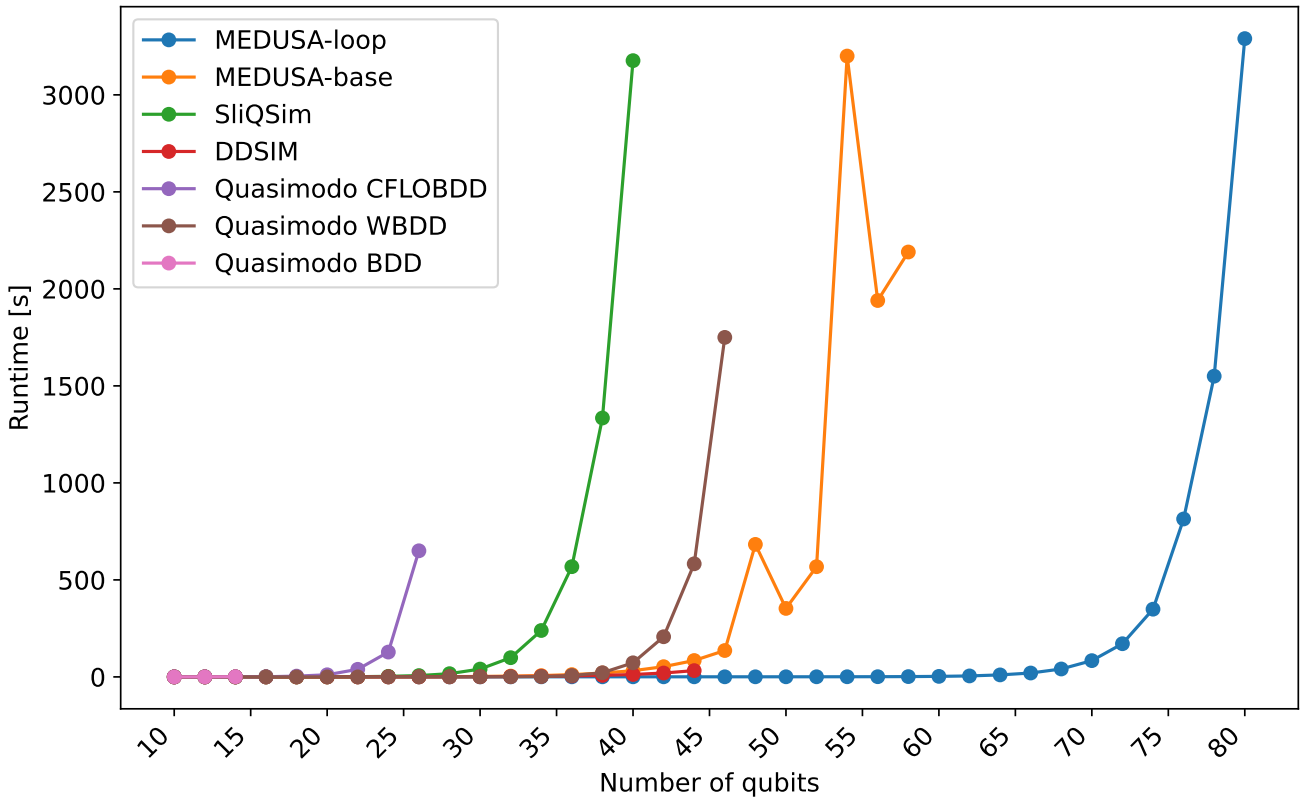
	circuit	#q	#G	MEDUSA$_{base}$ time	MEDUSA$_{base}$ mem	SliQSim time	SliQSim mem	DDSIM time	DDSIM mem	Quas[CFLOBDD] time	Quas[CFLOBDD] mem	Quas[WBDD] time	Quas[WBDD] mem	Quas[BDD] time	Quas[BDD] mem	Quokka# time	Quokka# mem
FEYNMAN	gf2^{32}_mult	96	3,322	0.26	40	1.35	13	0.10	71	0.66	460	0.11	502	0.82	450	0.87	45
	gf2^{128}_mult	384	50,043	20.40	231	264.81	38	5.28	235	10.50	478	4.76	1,159	27.60	498	15.39	570
	gf2^{256}_mult	768	198,395	163.00	1,635	TO	TO	41.21	538	43.30	531	38.50	4,989	238.00	633	71.28	2,324
	hwb10	16	31,764	0.80	51	84.20	15	0.21	38	4.72	466	0.22	447	1.56	445	TO	TO
	hwb11	15	87,789	2.64	103	660.93	22	0.49	70	12.80	475	0.52	449	1.51	448	TO	TO
	hwb12	20	171,482	5.80	205	2,568.02	35	1.13	133	27.20	510	1.35	456	6.43	457	3,193.79	1,070
MOG	10	30	2,433	0.20	42	1.26	12	0.08	34	9.08	595	0.05	456	TO	TO	62.68	40
	11	33	3,746	0.36	45	3.12	13	0.13	42	48.80	906	0.08	462	TO	TO	167.01	56
RANDOM	85	85	255	1.00	52	0.47	15	2.12	64	ERR	ERR	0.11	485	ERR	ERR	0.03	12
	94	94	282	79.60	337	0.78	18	4.45	76	ERR	ERR	74.30	521	ERR	ERR	0.08	13
	99	99	297	9.58	173	0.38	12	2.61	79	ERR	ERR	0.67	526	ERR	ERR	0.08	13
REVLIB	apex5_290	1,025	2,909	1.75	62	0.37	44	1.03	536	0.26	467	1.33	1,214	3.95	516	2.11	73
	cps_292	923	2,763	1.19	58	0.21	31	1.25	485	0.22	465	1.09	1,035	2.82	528	1.39	60
	seq_314	1,617	5,990	4.96	98	1.35	109	4.11	835	0.54	477	3.71	1,776	14.00	537	3.65	124
REVLIB-H	cpu_register_32_405	328	1,978	0.46	214	0.09	15	0.42	195	0.57	469	0.70	668	0.33	457	ERR	ERR
	e64-bdd_295	195	516	1.98	239	2.49	14	2.00	127	0.62	477	0.54	614	1.91	496	ERR	ERR
	ex5p_296	206	736	7.61	283	12.03	21	3.57	132	0.99	490	1.15	691	6.23	549	ERR	ERR

num fastest time TO timeout (1h) time: [s], mem: [MiB]

num fastest accurate simulator (`MEDUSA` or `SliQSim`)

ERR error

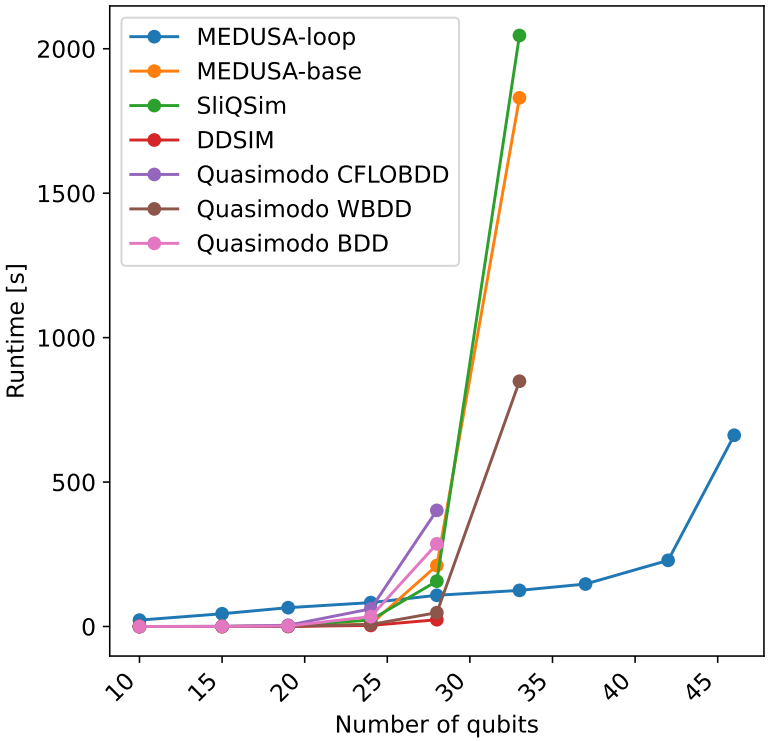
Results of MEDUSA – loop



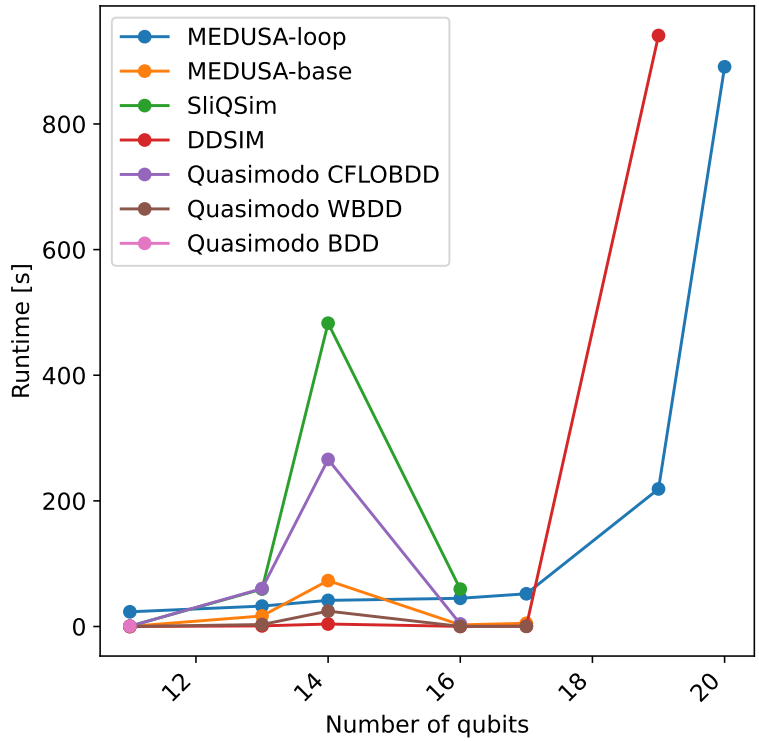
(a) Grover's search

(MEDUSA-base → without loop summarization
MEDUSA-loop → with loop summarization)

Results of MEDUSA – loop



(b) Period Finding (without QFT)



(c) Quantum Counting (without QFT)

(MEDUSA-base → without loop summarization
MEDUSA-loop → with loop summarization)

Results of MEDUSA – loop

	circuit	#q	#G	MEDUSA$_{loop}$		MEDUSA$_{base}$		SliQSim		DDSIM		Quas [CFLOBDD]		Quas [WBDD]		Quas [BDD]	
				time	mem	time	mem	time	mem	time	mem	time	mem	time	mem	time	mem
Grover	7	14	480	0	99	0	37	0	12	0	30	0	463	0	444	1	445
	13	26	8,037	0	137	1	47	7	13	0	49	650	3,405	0	459	TO	TO
	20	40	140,721	0	187	32	387	3,176	25	12	118	TO	TO	73	769	TO	TO
	22	44	310,367	0	196	84	1,088	TO	TO	32	254	TO	TO	583	1,083	TO	TO
	23	46	461,646	1	200	136	1,735	TO	TO	TO	TO	TO	TO	1,750	1,708	TO	TO
	29	58	4,676,916	2	215	2,190	10,032	TO	TO	TO	TO	TO	TO	TO	TO	TO	TO
	40	80	292,359,936	3,290	251	TO	TO	TO	TO	TO	TO	TO	TO	TO	TO	TO	TO
PF	07_03_10	10	2,294	23	1,891	0	27	0	12	0	30	0	458	0	442	0	440
	19_09_15	28	39,321,545	109	2,154	247	32	587	3,002	178	31,144	1,580	459	198	452	2,160	455
	22_11_05	33	146,800,628	125	922	1,830	38	2,046	10,293	TO	TO	TO	TO	849	454	TO	TO
	22_11_15	33	448,790,444	128	1,662	3,020	27	TO	TO	TO	TO	TO	TO	2,650	454	TO	TO
	31_15_15	46	277,025,390,495	673	1,973	TO	TO	TO	TO	TO	TO	TO	TO	TO	TO	TO	TO
QC	07_03_15	11	6,108	24	2,092	0	42	1	12	0	33	0	459	0	443	1	446
	10_05_05	16	40,937	45	2,115	3	83	60	15	0	42	4	459	0	446	TO	TO
	11_05_05	17	81,898	52	2,116	5	109	TO	TO	0	65	TO	TO	0	447	TO	TO
	12_06_15	19	376,760	250	7,691	TO	TO	TO	TO	1,280	294	TO	TO	TO	TO	TO	TO
	13_06_15	20	753,593	919	9,502	TO	TO	TO	TO	TO	TO	TO	TO	TO	TO	TO	TO

num fastest time TO timeout (1h) time: [s], mem: [MiB]

num fastest accurate simulator (`MEDUSA` or `SliQSim`)

Conclusion

- Accurate MTBDD-based simulator MEDUSA with efficient gate application procedures
- Technique for accelerating simulation of loops with fixed number of iterations using symbolic execution

Future work

- Support of **while** $M_{q_i} = 0$ **do** ($\cdots$) loops
- Closed form update formulae
- Verification