Solving the Frobenius Problem in Z3: Exploring Quantifier Elimination

Paul Anton
Email: panton@tudelft.nl

TU Delft

Research question

Does quantifier elimination improve Z3's performance on two-coin Frobenius Coin Problem instances over LIA, compared to its default quantifier handling strategy?

Subqestions:

1. Correctness: Do both configurations return satisfiable results with the correct Frobenius number as the model?
2. Runtime Performance: How does runtime compare between the default and QE-based approaches as the instance difficulty increases?
3. Memory Usage: How does memory consumption vary between configurations under increasing problem size?
4. Scalability: At what instance size does the default strategy begin to degrade in comparison to the QE-based approach?

Z3 and SMT Solving

- SMT = SAT + Theories (Arithmetic, Strings etc.)
- Z3 - an efficient SMT Solver [1]
- Amazon ~ a billion SMT queries per day in 2022 [2]

Listing 1: A satisfiable query

```
(set-logic QF_LIA)

(declare-const x Int)
(declare-const y Int)

(assert (> x 0))
(assert (> y 0))
(assert (= (+ (* 2 x) (* 3 y)) 5))

(check-sat)
```

Listing 2: An unsatisfiable query

```
(set-logic QF_LIA)

(declare-const x Int)
(declare-const y Int)

(assert (> x 0))
(assert (> y 0))
(assert (= (+ (* 2 x) (* 3 y)) 1))

(check-sat)
```

Frobenius Coin problem and LIA

- LIA = Linear Integer Arithmetic (integers + x*y not allowed)
- Frobenius : given 2 coin denominations, what is the largest unrepresentable amount with these 2 cons?
- Can be encoded into an SMT query over LIA with 3 universal quantifiers
- Z3: out of 54 satisfiable instances, Z3 times out on 51 [3]
- Z3's quantifier handling strategy, QSAT, handles 69 to 768 quantifiers in under 0.08 sec [4], thus, heuristics not adapted to Frobenius
- Modify Z3 itslef? No: modify Z3's approach throught tactics, strategies that transform or reduce logical goals, and combine them with tacticals, higher-order operators that combine tactics in various ways
- LIA admits quantifier elimination (qelim) : quantified F <=> quantifier-free F, however checks satisfiability directly through QSAT. Why not eliminate quantifiers first? Expensive in general, however, in our case, bounded number of quantifier alternation and number of variables per block indicate feasibility of a qelim strategy

Methodology: Default vs Customized Tactic Pipeline

Z3's approach

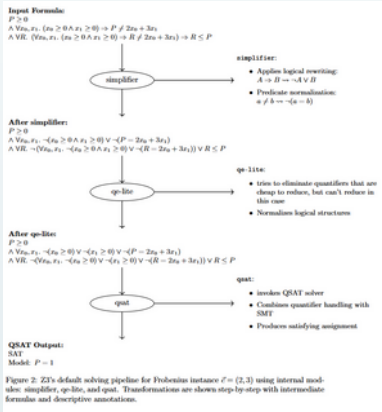

Figure 2: Z3's default solving pipeline for Frobenius instance $\vec{c} = (2, 3)$ using internal modules: simplifier, qe-lite, and qsat. Transformations are shown step-by-step with intermediate formulas and descriptive annotations.

Our approach

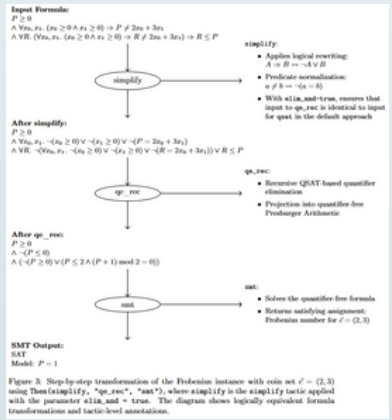

Figure 3: Step-by-step transformation of the Frobenius instance with coin set $\vec{c} = (2, 3)$ using `Then(simplify, "qe_rec", "sat")`, where `simplify` is the `simplify` tactic applied with the parameter `elim_and = true`. The diagram shows logically equivalent formula transformations and tactic-level annotations.

Results

Coin 1	#solved QSAT	#solved QE_rec	Mean QSAT (s)	RSE QSAT (%)	Mean QE_rec (s)	RSE QE_rec (%)
2	30	30	0.02	0.0%	0.02	0.0%
3	30	30	0.04	0.0%	0.03	0.0%
5	30	30	0.12	8.3%	0.06	0.0%
7	30	30	0.68	10.3%	0.13	0.0%
11	30	30	1.98	5.1%	0.68	4.4%
13	30	30	15.37	8.5%	1.44	4.9%
17	**20**	30	28.28	**14.8%**	6.37	4.9%
19	**26**	30	36.87	7.1%	10.13	3.7%
23	**0**	**30**	-	-	20.85	4.1%
> 23	0	0	-	-	-	-

Table 1: Comparison of QSAT and QE_rec on the first 9 Frobenius instances. RSE is the relative standard error, computed as $\frac{\text{SE}}{\text{mean}} \times 100$.

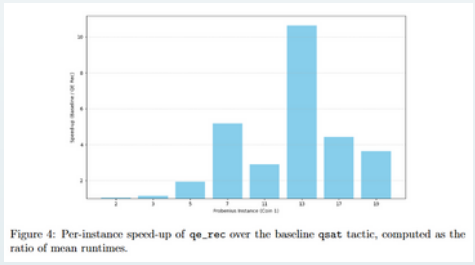

Figure 4: Per-instance speed-up of `qe_rec` over the baseline `qsat` tactic, computed as the ratio of mean runtimes.

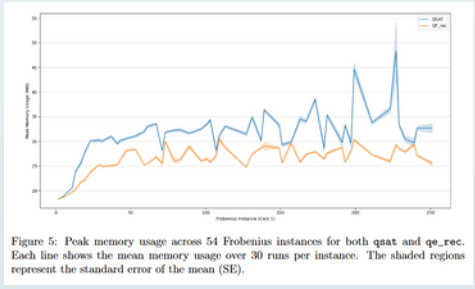

Figure 5: Peak memory usage across 54 Frobenius instances for both `qsat` and `qe_rec`. Each line shows the mean memory usage over 30 runs per instance. The shaded regions represent the standard error of the mean (SE).

- Both approaches returned the correct Frobenius number on solved instances
- qe_rec solved all 9 benchmark cases; qsat timed out on 3 harder instances (see Table 1)
- qe_rec achieved up to 10× speed-up over qsat as instance complexity increased (Figure 4)
- Runtime variance was lower with qe_rec; relative standard error (RSE) remained under 5% in most cases
- Peak memory usage was consistently lower and more stable for qe_rec across all instances (Figure 5)
- Results are based on 30 runs per instance, each instance with a timeout of 60 seconds, to ensure statistical robustness

Conclusion

1. Correctness: Both approaches returned sat and valid models on all solved instances.
2. Runtime: qe_rec was up to 10× faster than qsat, with speed-ups increasing on harder instances.
3. Memory: qe_rec used less memory and showed more consistent usage with lower variability.
4. Scalability: qsat failed on harder instances like (17, 19) and (23, 29); qe_rec solved them all.

These results confirm that quantifier elimination using qe_rec improves both runtime and memory efficiency over Z3's default QSAT-based strategy on the Frobenius Coin Problem instances tested and also highlight how customizing Z3's approach, which is based on hand-crafted heuristics, can lead to overall better performance. Nevertheless, this does not imply that our approach is superior in general, as performance may vary on larger instances, an aspect which was not covered due to the limitations imposed by the 60 second timeout and number of instances being included in the experiments.

Limitations

- Focused only on satisfiable 2-coin Frobenius instances with simple quantifier structures
- Did not evaluate performance on unsatisfiable or higher-dimensional (≥3 coins) instances
- Internal behavior of Z3 not deeply analyzed; improvements may stem from solver heuristics
- Only tested the qe_rec tactic; other quantifier elimination tactics were excluded due to returning unsat on some of the instances

References

[1] Leonardo de Moura and Nikolaj Bjørner. Z3: An efficient smt solver. In C. R. Ramakrishnan and Jakob Rehof, editors, Tools and Algorithms fo Construction and Analysis of Systems, pages 337–340, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-78800-3.

[2] Neha Rungta. A billion smt queries a day. 2022. URL https://www.amazon.science/publications/a-billion-smt-queries-a-day

[3] Jautomata Michal Hečko. Rozhodování logiky pomocí automatů [online]. Master's thesis, Brno University of TechnologyBrno, 2024 [cit. 2025-05-
URL https://theses.cz/id/y8bZ66/. SUPERVISOR:

[4] Nikolaj S Bjørner and Mikoláš Janota. Playing with quantified satisfaction. LPAR (short papers), 35:15–27, 2015.