

# CRTSolver : Solving Non-Linear Integer Equations using the Chinese Remainder Theorem

An Extended Abstract

Martin Brain<sup>1</sup>, Maheen Matin<sup>2</sup>

<sup>1</sup>City St. George's, University of London, Northampton Square, London, EC1V 0HB, United Kingdom

## Abstract

The theory of integers allows for non-linear terms and is thus undecidable in the general case. This is not only a practical problem but also discourages work on the decidable and easy fragments of the theory. In this extended abstract we present a simple semi-decision procedure for the polynomial integer equations which shows significant promise in preliminary experiments with synthetic benchmarks. We hope that this will illuminate the considerable improvements that are possible with well known techniques.

## Keywords

Satisfiability modulo theories, Non-linear integer equations, Diophantine equations

## 1. Introduction

A reasonable performance floor for SMT solvers is that they should be faster than a human solving the same problem with a pen and paper. In the case of non-linear expressions over theory of integers this floor is not always met. For example consider the following problem:

In our experiments we found that but observing will allow many humans to solve it faster.

This has an immediate practical consequence of their being no algorithms for the general case but we also believe that it has had chilling effect on the theory in general. We believe there are many interesting and useful fragments of the theory of integers for which there are practical decision procedures, semi-decision procedures or good heuristics. In this paper we will consider one such fragment, existential polynomial equations. In Section 2 we propose an abstraction-refinement style algorithm that uses the Chinese Remainder Theorem. Section ?? we describe some preliminary experiments using our implementation of the algorithm. Finally Section 4 discusses the future potential of this approach.

## 2. Algorithm

This section gives a semi-decision procedure for polynomial integer equations. It is based on the following two elementary observations:

1. If a set of equations is satisfiable over the integers then it will be satisfiable modulo any number. We use the converse – if a set of equations is unsatisfiable over a specific modulus then it must be unsatisfiable over the integers.
2. If a set of equations is solvable modulo  $p$  and  $q$  with  $p$  and  $q$  are coprime then they will be solvable modulo  $p * q$ . This is a consequence of the Chinese Remainder Theorem.

These are used to create an abstraction-refinement loop using a pair of solvers. One computes the solutions modulo a product of primes (giving an over-approximation of the true satisfying assignments) and the other checks these candidates. The algorithm consists of a loop of four phases, illustrated in Figure 1

SMT 2025: 23rd International Workshop on Satisfiability Modulo Theories, August 10–11, 2025, Glasgow, UK

✉ martin.brain@citystgeorges.ac.uk (M. Brain)

id 0000-0003-4216-7151 (M. Brain)



© 2025 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

MB: I have set up some TODO note macros. Remove them as they are done. Use mmsays to mark things you want to note.

MB: Feel free to change the title.

MB: If being first author is valuable to you, please put yourself first. Otherwise I suggest we use alphabetic by surname.

MB: Register for an ORCID if you want and fill in any other contact in-

Figure 1: A semi-decision procedure for polynomial integer equations.

Table 1  
Performance Comparison between CRTSolver and ...

Variables	Equations	Degree	SAT	Iterations	CRT Time w/ Integer	CRT Time w/ BV	cvc5	z3
?	?	?	?	?	?	?	?	?
?	?	?	?	?	?	?	?	?

Figure 2: A cactus plot of the results.

Pick Prime

Solve Modulo Subproblems

Extract Candidates

Check Candidates

We should probably say something about correctness, complexity or completeness. That might be “We don’t know”.

I think there are theorems which says something like:

- This is a semi-decision procedure (if it terminates then it is correct)
- If all of the variables are bounded so they have a finite range then the algorithm will terminate

but I am unsure if we will have time to prove these.

3. Results

We have implemented the algorithm in Section 2 in a tool called CRTSo1ver which is available at <https://whatever> under LICENSE-GOES-HERE. It uses two instances of cvc5, one to solve the modulo sub-problems and a second to check the candidate results. The following options are available

We have created a set of benchmarks that

In Table 1 we give a comparison with . These results are also shown in a cactus plot in Figure 2.

These result suggest that

4. Conclusion

It works (on the limited set of benchmarks we have tried).

It shows enough promise to be investigated further. We want to try to implement it in an actual solver.

There are many open questions and possible future directions:

- Are there better encodings for solving the modulo sub problems in the BV solver?
- We select primes sequentially, is that the best thing to do or are there better strategies given the encoding, the constants in the problem, how previous candidates have failed, etc.
- How many candidates should be checked, how should they be picked?
- Can the failed candidates be used to improve the modulo sub problems?
- Can this technique be extended to non-equalities ( $\neq$ ), inequalities ( $\leq$ ) and non-polynomial expressions?

MB: Fill in the description

MB: Create a flow-chart of the phases. I can do this if you don't feel like doing it

MB: If have a worked example and want to include it, here would be good.

MB: Think about this.

MB: Please help fill in the gaps and finish the text. There doesn't need to be much test.

MB: Describe the options that will

## Declaration on Generative AI

*Either:*

The author(s) have not employed any Generative AI tools.

*Or (by using the activity taxonomy in [ceur-ws.org/genai-tax.html](http://ceur-ws.org/genai-tax.html)):*

During the preparation of this work, the author(s) used X-GPT-4 and Gramby in order to: Grammar and spelling check. Further, the author(s) used X-AI-IMG for figures 3 and 4 in order to: Generate images. After using these tool(s)/service(s), the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

,

MB: We need to fill this out. I haven't used any generative AI tools.