Simple Programs with NP-hard Termination

Henry Sinclair-Banks

About a part of ongoing work with Dmitry Chistikov, Wojciech Czerwiński, Łukasz Orlikowski, and Karol Węgrzycki.

Warwick Postgraduate Colloquium in Computer Science
Theory and Foundations
11th December 2023
what simple programs are we considering?

Karol

• Non-negative integer counters.

• Integer additive updates.

• Non-deterministic (not nested) loops.

• Zero tests outside of loops.
Precondition: $x = v$, $y = 0$.

Postcondition: $x = 5v$, $y = 0$.

We can Multiply!
**Show me an example!**

**Precondition:** \( x = v, \ y = 0. \)

**Postcondition:** \( x = 5v, \ y = 0. \)

We can Multiply!

We can also Divide!
INPUT: A program and initial values ($v_1, v_2, ...$).

QUESTION: Can the program terminate?

In this example, suppose $v_2 = 0$, then the program terminates $\iff v_1$ is divisible by 5.

Importantly, everything is encoded in unary!

Instance size: sum of absolute values of all updates.
The program terminates \iff v is not divisible by 5.

Initialisation:
\[ x = v, y = 0 \]

Sequence:
- \[ x += 1, y += 3 \]
- \[ x += 1, y -= 1 \]
- \[ x -= 5, y += 5 \]
- \[ x =? 0 \]
- \[ y =? 0 \]
- \[ x -= 4 \]

We can test for non-divisibility!
The program terminates $\iff v$ is not divisible by 5.

**Example:** suppose $v = 23$.

We can test for non-divisibility!
Suppose $\phi$ contains $n$ variables.

Associate the first $n$ primes to the variables:
\[
x_1 \leftarrow p_1, \, x_2 \leftarrow p_2, \, \ldots, \, x_n \leftarrow p_n.
\]

An assignment will be represented by $v \in [0, p_1 \cdot p_2 \cdot \ldots \cdot p_n)$, where
\[
v \equiv 0 \mod p_i \iff x_i = \text{false}, \text{ and } \quad v \equiv 1 \mod p_i \iff x_i = \text{true}.
\]

Idea: 1/ Guess $v$ (Chinese Remainder Theorem $\Rightarrow$ $v$ exists for every assignment)
2/ Check $v$ corresponds to an assignment.
3/ Check the evaluation under $v$'s assignment.
2/ Check $\nu$ corresponds to an assignment.

Want to check: for each $i$, either $\nu \equiv 0 \mod p_i$ or $\nu \equiv 1 \mod p_i$.

$\Rightarrow \nu \not\equiv 2 \mod p_i, \quad \nu \not\equiv 3 \mod p_i, \quad \ldots$, and $\nu \not\equiv (p_i - 1) \mod p_i$.

$\Rightarrow \nu - 2 \not\equiv 0 \mod p_i, \quad \nu - 3 \not\equiv 0 \mod p_i, \quad \ldots$, and $\nu - (p_i - 1) \not\equiv 0 \mod p_i$.

$\Rightarrow p_i$ does not divide $\nu - 2$, and

$p_i$ does not divide $\nu - 3$, and

$\ldots$, and

$p_i$ does not divide $\nu - (p_i - 1)$.
3/ Check the evaluation under \( v \)'s assignment:

The only way to fail a clause \((x_1 \lor \overline{x}_2 \lor \overline{x}_3)\) is to set: 
\(x_1 = \text{false}, x_2 = \text{true}, \text{and } x_3 = \text{true}.\)

That means \( v \equiv 0 \mod p_1, \ v \equiv 1 \mod p_2, \text{ and } v \equiv 1 \mod p_3.\)

Precisely, \( v \equiv 0 \mod 2, \ v \equiv 1 \mod 3, \text{ and } v \equiv 1 \mod 5. \)

\[ \Rightarrow v \equiv 16 \mod 30. \]

So \((x_1 \lor \overline{x}_2 \lor \overline{x}_3)\) is not satisfied if \( v - 16 \equiv 0 \mod 30.\)

Therefore, this clause is satisfied if \(30\) does not divide \(v - 16.\)
we can use our simple programs to implement 3-SAT using non-divisibility.

1/ Guess \( v \in \{0, p_1 \cdot p_2 \cdot \ldots \cdot p_n\} \).

We will store \( v \) on counter \( x \).
we can use our simple programs to implement 3-SAT using non-divisibility.

2/ Check \( v \) represents an assignment:

- \( \text{NON-DIV}[x-2, p_1] \)
- \( \text{NON-DIV}[x-3, p_1] \)
- \( \ldots \)
- \( \text{NON-DIV}[x-(p_1-1), p_1] \)
- \( \ldots \)
- \( \text{NON-DIV}[x-2, p_n] \)
- \( \text{NON-DIV}[x-3, p_n] \)
- \( \ldots \)
- \( \text{NON-DIV}[x-(p_n-1), p_n] \)

Initialisation:
\[ x = 0, y = 0 \]

Loop:
\[ x += 1 \]

Check \( v \) represents an assignment
we can use our simple programs to implement 3-SAT using non-divisibility.

Henry

3/ Check the evaluation under \( v \)'s assignment:

Suppose clause 1 = \((x_1 \lor \overline{x}_2 \lor \overline{x}_3)\), then try

\[
\text{NON-DIV}[x-16, 30]
\]

to test satisfaction.

Initialisation:
\[
x = 0, y = 0
\]

LOOP
\[
x += 1
\]

Check \( v \) represents an assignment

Check the evaluation under \( v \)'s assignment

Clause 1:
\[
\text{NON-DIV}[x-16, 30]
\]

Clause 2:
\[
\text{NON-DIV}[x-r_2, p_i \cdot p_j \cdot p_k]
\]

Clause \( m \):
\[
\text{NON-DIV}[x-r_m, p_i' \cdot p_j' \cdot p_k']
\]
we can use our simple programs to implement 3-SAT using non-divisibility.

The program terminates

\[ \iff \exists \nu \text{ that } \nu \text{ represents an assignment that satisfies } \phi \iff \phi \text{ is satisfiable.} \]

**Theorem:** Termination for simple programs with two counters and zero-tests between loops is NP-hard.
We can replace the zero-tests between loops with just one additional counter!

**Theorem:** Termination for simple programs with two counters and zero-tests between loops is NP-hard.

**Corollary:** Termination for simple programs with three counters is NP-hard. (with no zero tests at all!)
Corollary: Termination for simple programs with three counters is NP-hard.

starring...

Dmitry as Dmitry
Henry as Henry
Karol as Karol
Łukasz as Łukasz
Wojciech as Wojtek

Thank you!

Presented by Henry Sinclair-Banks
Warwick Postgraduate Colloquium in Computer Science Winter 2023

http://henry.sinclair-banks.com