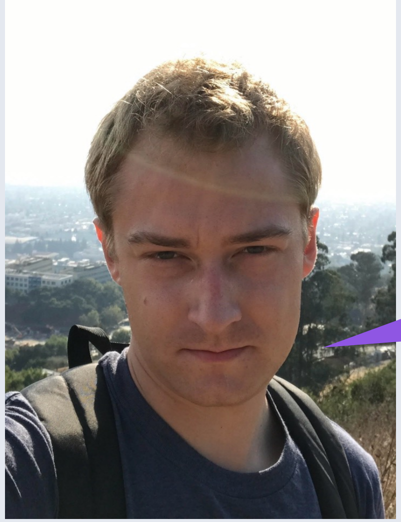
Simple Programs with NP-hard Termination

Henry Sinclair-Banks

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

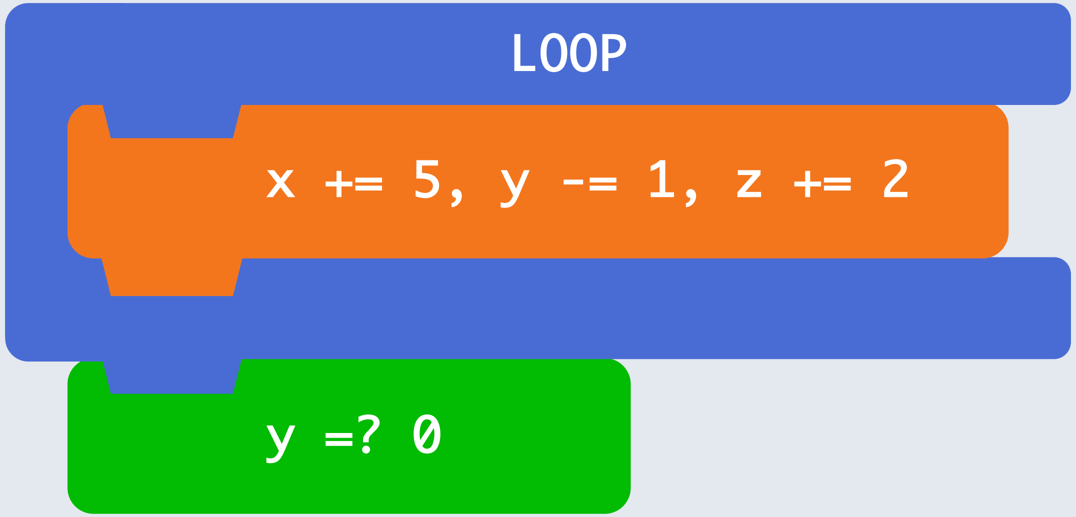
Warwick Postgraduate Colloquium in Computer Science
Theory and Foundations
11th December 2023



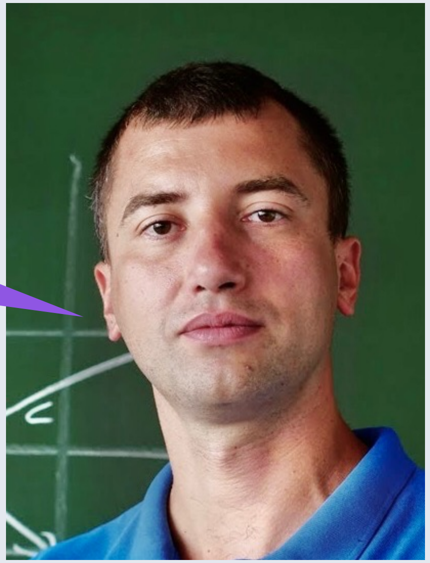
Karol

what simple programs are we considering?

- Non-negative integer counters.
- Integer additive updates.
- Non-deterministic (not nested) loops.
- Zero tests outside of loops.



Show me an example!



Wojtek

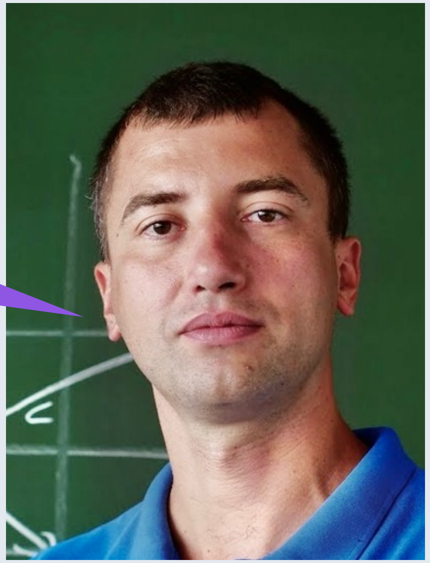
```
LOOP
    x -= 1, y += 1
x =? 0
LOOP
    x += 5, y -= 1
y =? 0
```

Precondition: x = v, y = 0.

Postcondition: x = 5v, y = 0.

We can Multiply!

Show me an example!



Wojtek

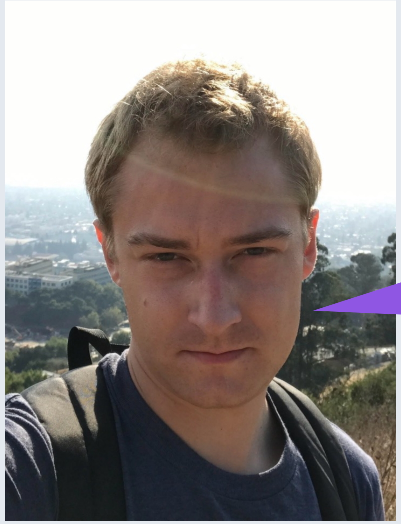
```
LOOP
    x -= 5, y += 1
x =? 0
LOOP
    x += 1, y -= 1
y =? 0
```

Precondition: x = v, y = 0.

Postcondition: x = 5v, y = 0.

We can Multiply!

We can also Divide!



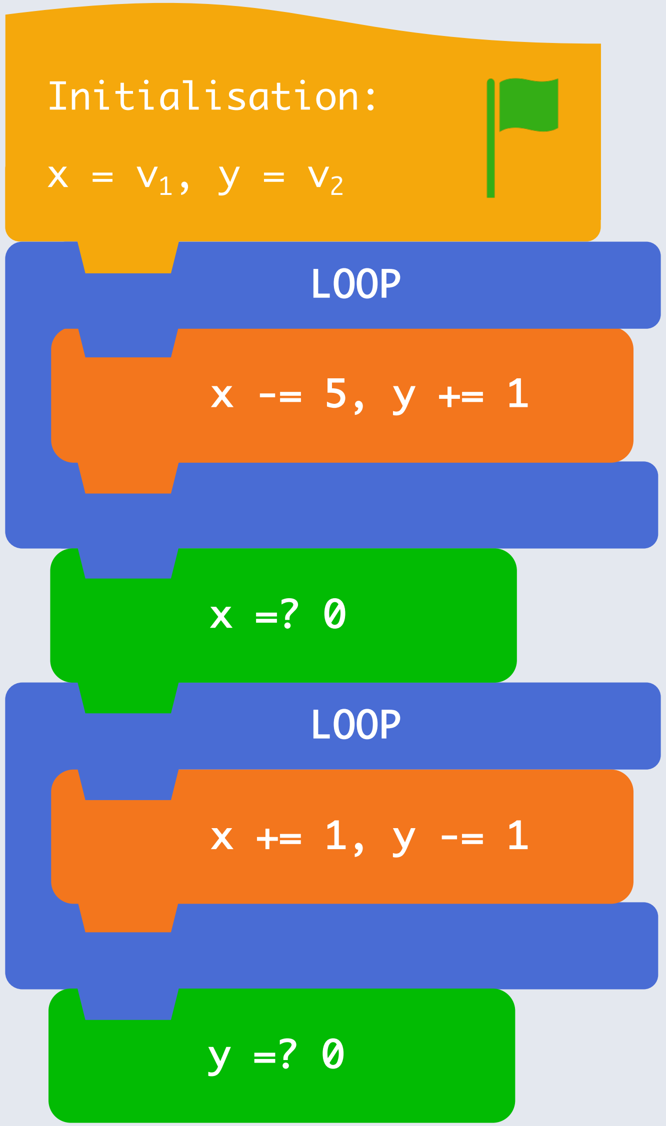
Karol

what is the termination decision problem?

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 $\Leftrightarrow$ v_1 is divisible by 5.

Importantly, everything is encoded in unary!
Instance size: sum of absolute values of all updates.



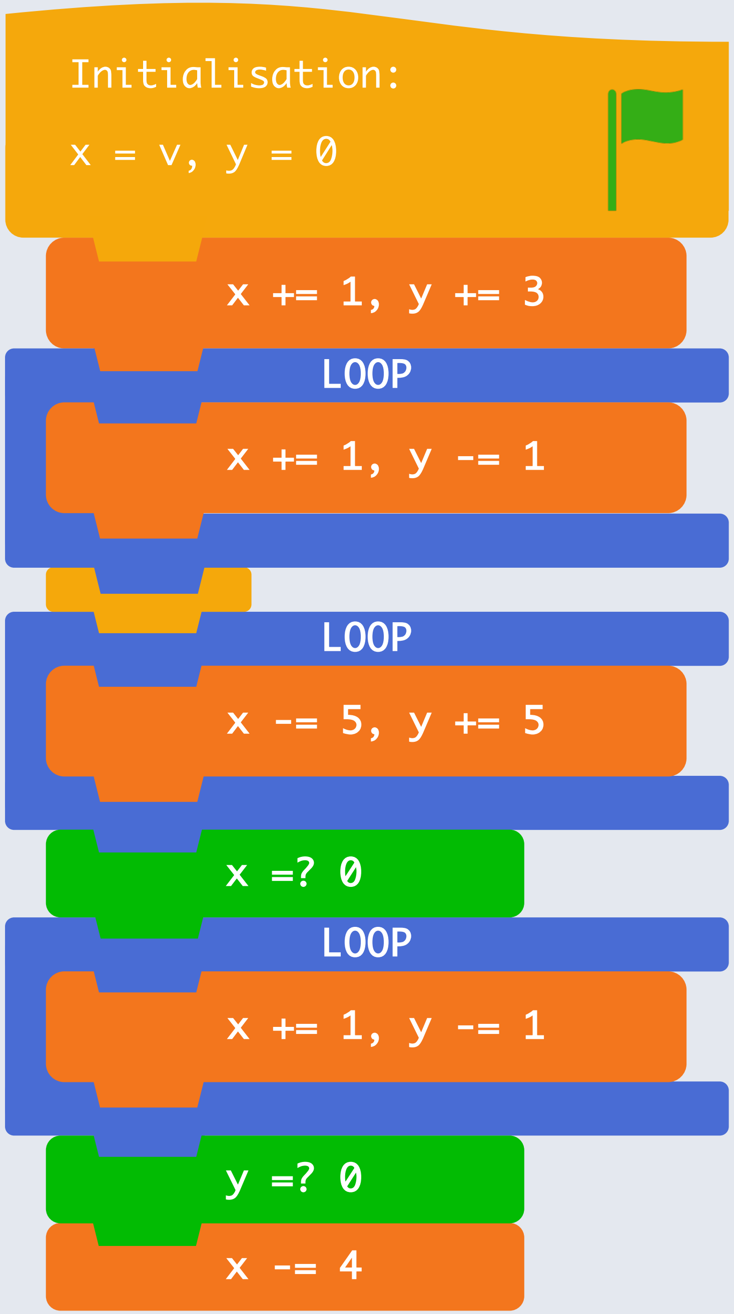
Henry

We can test for non-divisibility!

The program terminates

$\Leftrightarrow$

v is not divisible by 5.



Henry

We can test for non-divisibility!

The program terminates

$\Longleftrightarrow$

v is not divisible by 5.

NON-DIV[x, 5]

Example: suppose v = 23.

x = 24, y = 3

x = 25, y = 2

x = 0, y = 27

x = 27, y = 0

x = 23, y = 0

Initialisation:

x = 23, y = 0

x += 1, y += 3

LOOP

x += 1, y -= 1

LOOP

x -= 5, y += 5

x =? 0

LOOP

x += 1, y -= 1

y =? 0

x -= 4

We can test 3-SAT using conjunctions of non-divisibility tests!

Dmitry

Suppose ϕ contains n variables.

Associate the first n primes to the variables:

$$x_1 \leftarrow p_1, \quad x_2 \leftarrow p_2, \quad \ldots, \quad x_n \leftarrow p_n.$$

An assignment will be represented by $v \in [0, p_1 \cdot p_2 \cdot \ldots \cdot p_n)$, where

$$v \equiv 0 \bmod p_i \iff x_i = \text{false}, \text{ and}$$

$$v \equiv 1 \bmod p_i \iff x_i = \text{true}.$$

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.

We can test 3-SAT using conjunctions of non-divisibility tests!

Dmitry

2/ Check v corresponds to an assignment.

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

$\Rightarrow$ $v \not\equiv 2 \mod p_i$, $v \not\equiv 3 \mod p_i$, …, and $v \not\equiv (p_i - 1) \mod p_i$.

$\Rightarrow$ $v - 2 \not\equiv 0 \mod p_i$, $v - 3 \not\equiv 0 \mod p_i$, …, and $v - (p_i - 1) \not\equiv 0 \mod p_i$.

$\Rightarrow$ p_i does not divide $v - 2$, and
p_i does not divide $v - 3$, and
…, and
p_i does not divide $v - (p_i - 1)$.

We can test 3-SAT using conjunctions of non-divisibility tests!

Dmitry

3/ Check the evaluation under v's assignment:

The only way to fail a clause $(x_1 \vee \overline{x_2} \vee \overline{x_3})$ is to set:
$x_1 =$ false, $x_2 =$ true, and $x_3 =$ true.

That means $v \equiv 0 \bmod p_1$, $v \equiv 1 \bmod p_2$, and $v \equiv 1 \bmod p_3$.
Precisely, $v \equiv 0 \bmod 2$, $v \equiv 1 \bmod 3$, and $v \equiv 1 \bmod 5$.

$\Rightarrow v \equiv 16 \bmod 30$.

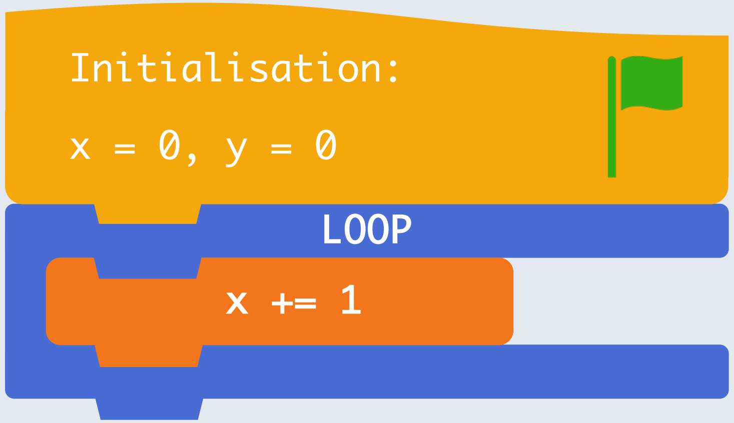
So $(x_1 \vee \overline{x_2} \vee \overline{x_3})$ is not satisfied if $v - 16 \equiv 0 \bmod 30$.
Therefore, this clause is satisfied if 30 does not divide $v - 16$.

Henry

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.



Henry

We can use our simple programs to implement 3-SAT using non-divisibility.

2/ Check v represents an assignment:

NON-DIV[x-2, p_1]

NON-DIV[x-3, p_1]

...

NON-DIV[x-(p_1-1), p_1]

...

NON-DIV[x-2, p_n]

NON-DIV[x-3, p_n]

...

NON-DIV[x-(p_n-1), p_n]

Initialisation:

x = 0, y = 0

LOOP

x += 1

Check v represents an assignment

Henry

We can use our simple programs to implement 3-SAT using non-divisibility.

3/ Check the evaluation under v's assignment:

Suppose clause 1 = $(x_1 \vee \overline{x_2} \vee \overline{x_3})$, then try

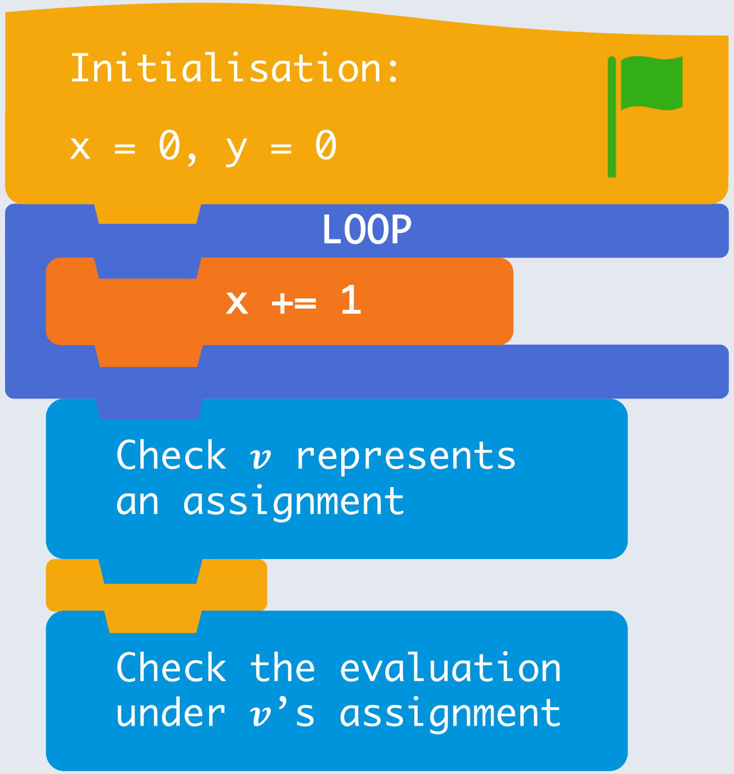
NON-DIV[x-16, 30] to test satisfaction.

Clause 1: NON-DIV[x-16, 30]

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

...

Clause m: NON-DIV[x- r_m, $p'_i \cdot p'_j \cdot p'_k$]



Henry

We can use our simple programs to implement 3-SAT using non-divisibility.

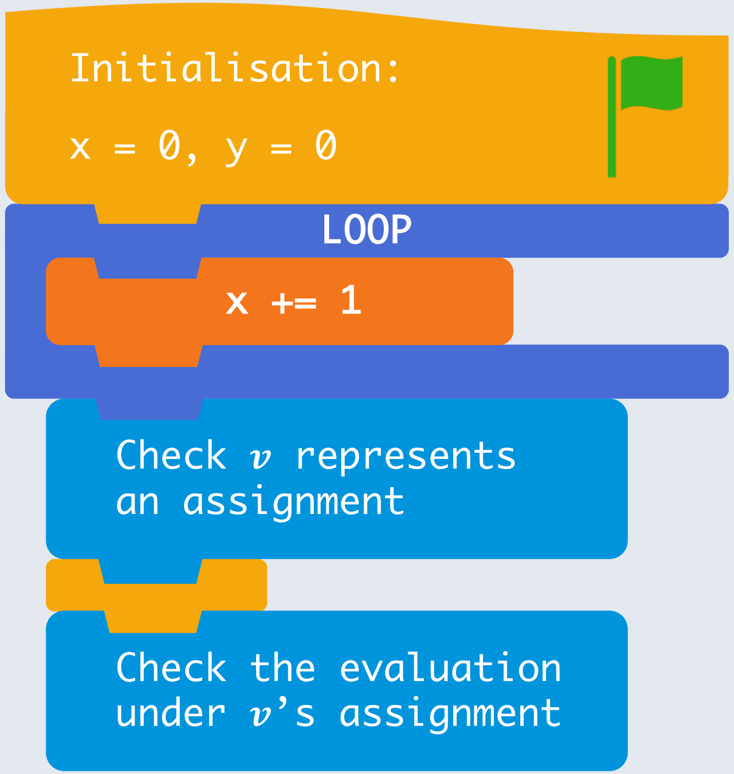
The program terminates

$\Leftrightarrow$

there exists v that v represents an assignment that satisfies ϕ

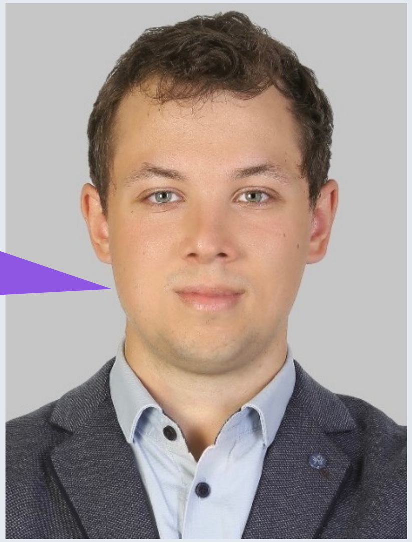
$\Leftrightarrow$

ϕ 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!



Łukasz

Lower Bounds for the Reachability Problem in Fixed Dimensional VASSes

Wojciech Czerwiński
wczerwin@mimuw.edu.pl
University of Warsaw
Poland

Łukasz Orlikowski
lo418363@students.mimuw.edu.pl
University of Warsaw
Poland

ABSTRACT

We study the complexity of the reachability problem for Vector Addition Systems with States (VASSes) in fixed dimensions. We provide

and the most clear evidence for that is the existence of big complexity gaps for the problem in small fixed dimensions. The prominent example here is the dimension three with complexity gap

LEMMA 2.5. *Let* $\mathrm{src} \xrightarrow{\rho} \mathrm{trg}$ *be a run of a* $(d+1)$*-VASS* V *and let* $\mathrm{src} = c_0, c_1, \ldots, c_{n-1}, c_n = \mathrm{trg}$ *be some of the configurations on* ρ. *Let* ρ_j *for* $j \in [1,n]$ *be the parts of the run* ρ *starting in* c_{j-1} *and finishing in* c_j, *namely*

$$c_0 \xrightarrow{\rho_1} c_1 \xrightarrow{\rho_2} \ldots \xrightarrow{\rho_{n-1}} c_{n-1} \xrightarrow{\rho_n} c_n.$$

Let $S_1, \ldots, S_d \subseteq [0,n]$ *be the sets of indices of* c_j, *in which we want to zero-test counters numbered* $1, \ldots, d$, *respectively and let* $N_{j,i} = |\{k \geq j \mid k \in S_i\}|$ *for* $i \in [1,d], j \in [0,n]$ *be the number of zero-tests, which we want to perform on the* i*-th counter starting from configuration* c_j *(in other words after the run* ρ_j *for* $j > 0$*). Then if:*

1. $\mathrm{src}[d+1] = \sum_{i=1}^{d} N_{0,i} \cdot \mathrm{src}[i]$;
2. *for each* $j \in [1,n]$ *we have* $\mathrm{eff}(\rho_j, d+1) = \sum_{i=1}^{d} N_{j,i} \cdot \mathrm{eff}(\rho_j, i)$; *and*
3. $\mathrm{trg}[d+1] - 0$

then for each $i \in [1,d]$ *and for each* $j \in S_i$ *we have* $c_j[i] = 0$.

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

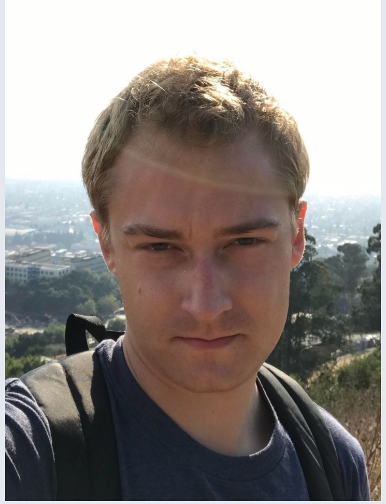
Corollary: Termination for simple programs with three counters is NP-hard. (with no zero tests at all!)

Corollary: Termination for simple programs with three counters is NP-hard.

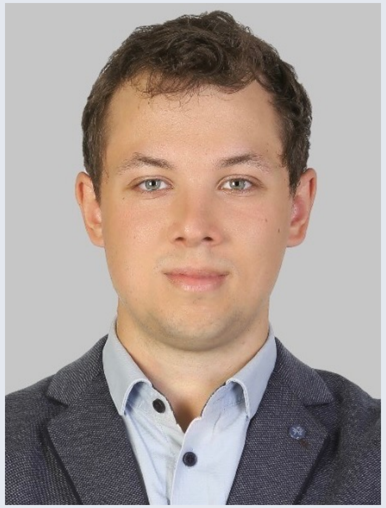
starring...

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