Moments in Time

Algebraic Analysis of Solvable Loops

Laura Kovács

for(syte) TU WIEN Informatics

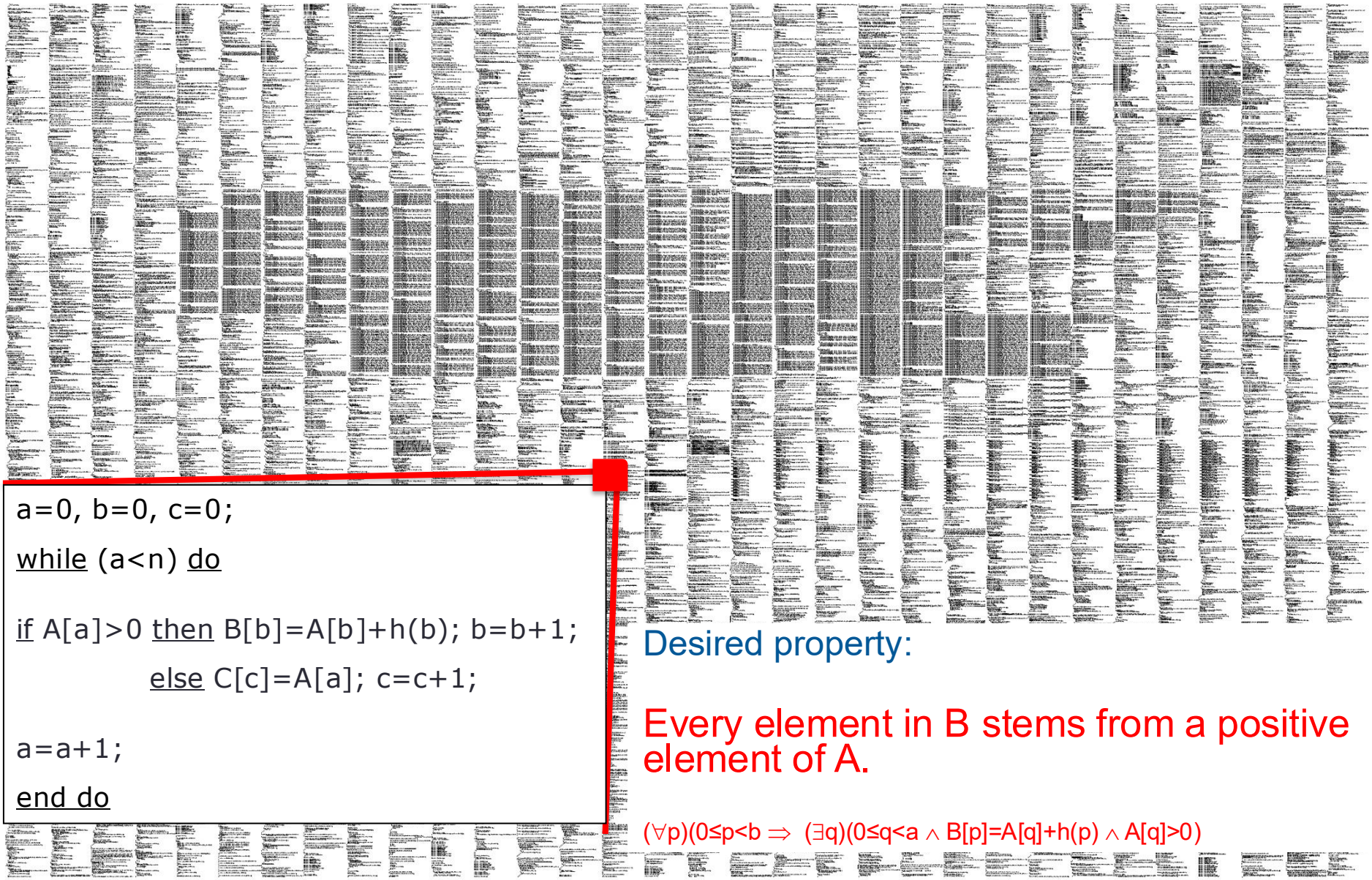
Solvable Program Analysis

Solvable Program Analysis

```
a=0, b=0, c=0;
while (a<n) do

if A[a]>0 then B[b]=A[b]+h(b); b=b+1;
          else C[c]=A[a]; c=c+1;

a=a+1;
end do
```

Solvable Program Analysis

Does the property hold?



```
a=0, b=0, c=0;
while (a<n) do

if A[a]>0 then B[b]=A[b]+h(b); b=b+1;
          else C[c]=A[a]; c=c+1;

a=a+1;
end do
```

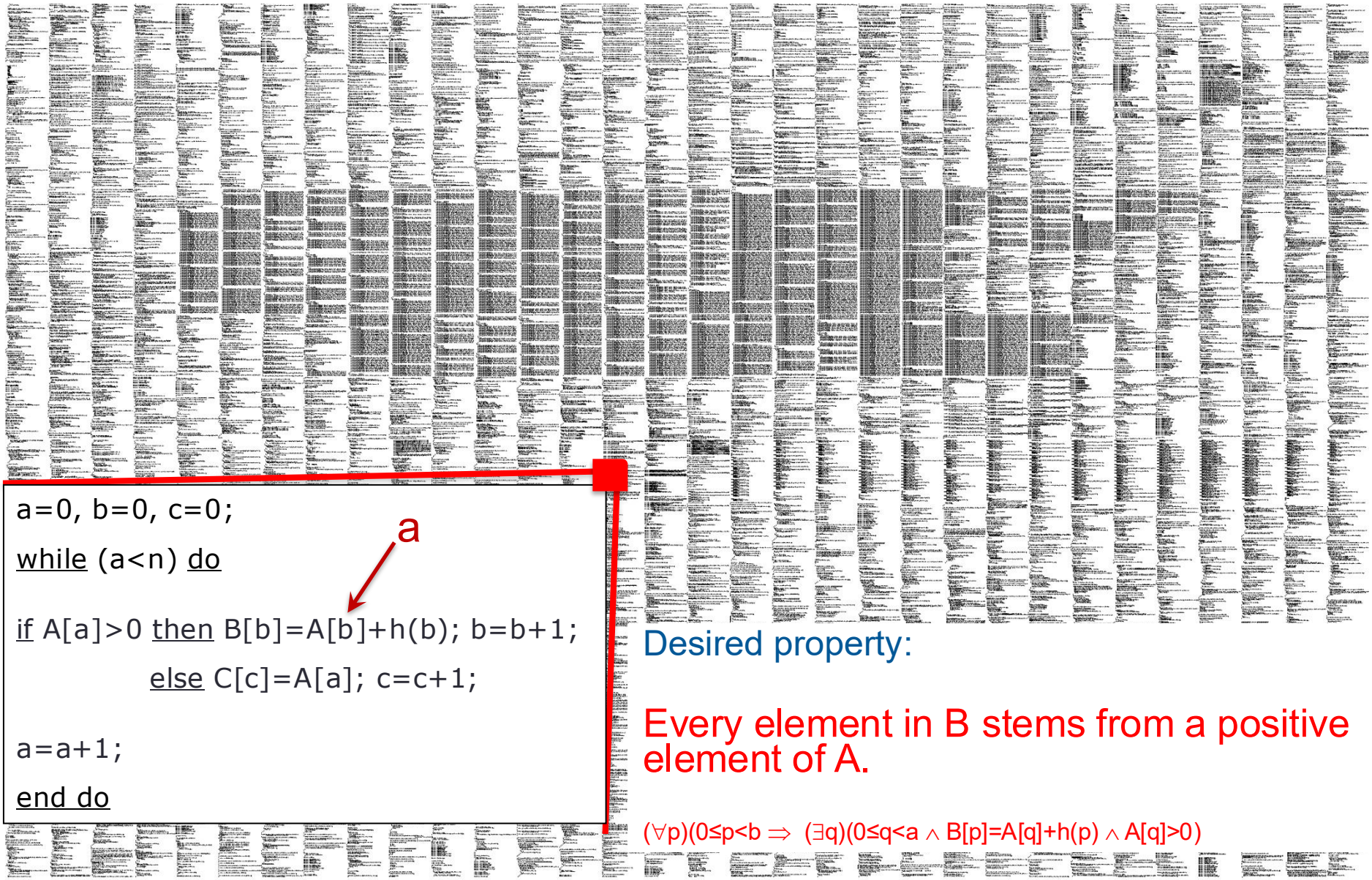
Desired property:

Every element in B stems from a positive element of A.

$(\forall p)(0 \le p < b \Rightarrow (\exists q)(0 \le q < a \wedge B[p] = A[q] + h(p) \wedge A[q] > 0)$

Solvable Program Analysis

Does the property hold?



```
a=0, b=0, c=0;
while (a<n) do

if A[a]>0 then B[b]=A[b]+h(b); b=b+1;
          else C[c]=A[a]; c=c+1;

a=a+1;
end do
```

a

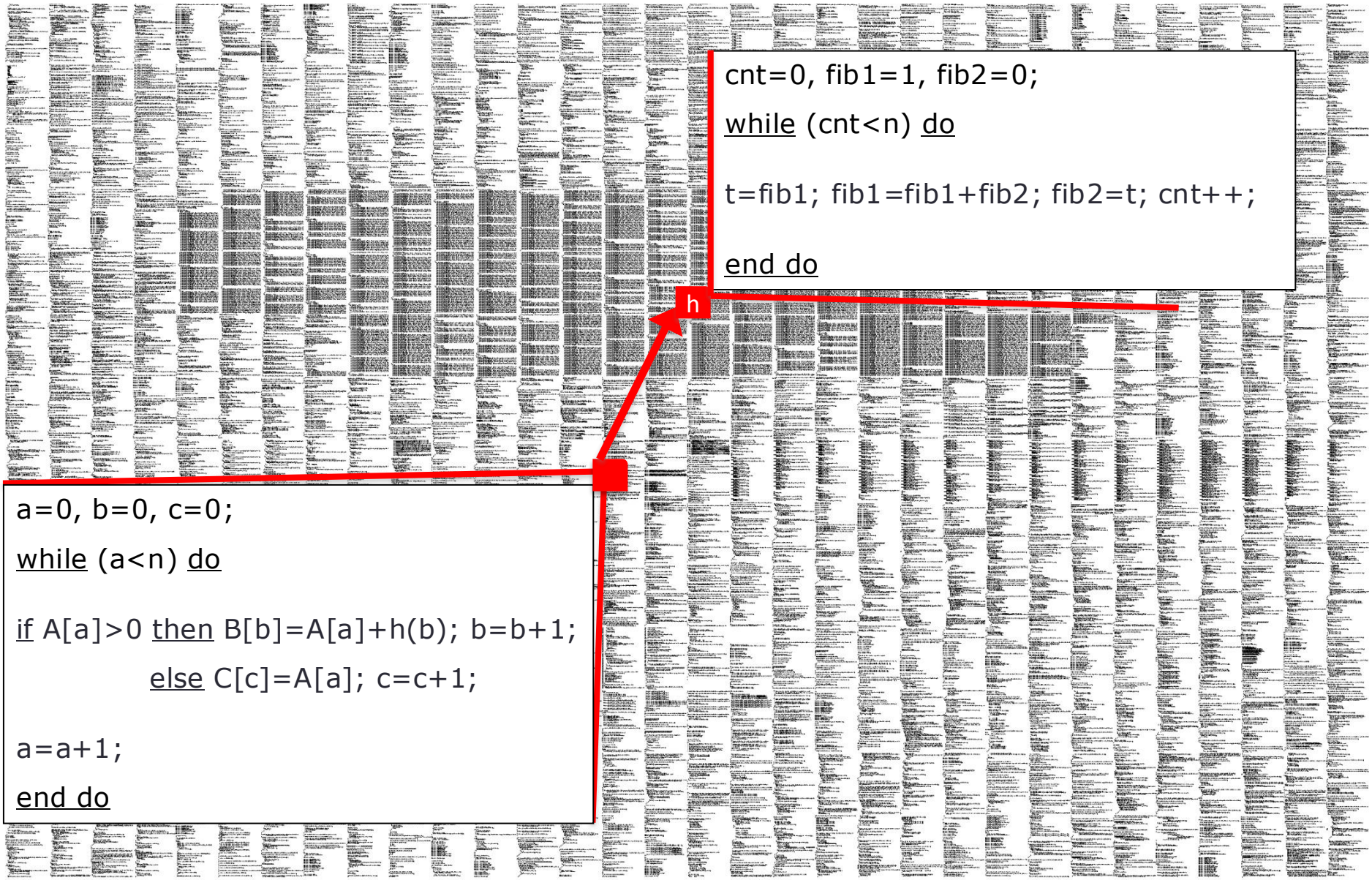
Desired property:

Every element in B stems from a positive element of A.

$(\forall p)(0 \le p < b \Rightarrow (\exists q)(0 \le q < a \wedge B[p]=A[q]+h(p) \wedge A[q]>0)$

Solvable Program Analysis

What "constant" property does the code have?



Solvable Program Analysis

What "constant" property does the code have?

```
cnt=0, fib1=1, fib2=0;
while (cnt<n) do

t=fib1; fib1=fib1+fib2; fib2=t; cnt++;

end do
```

h

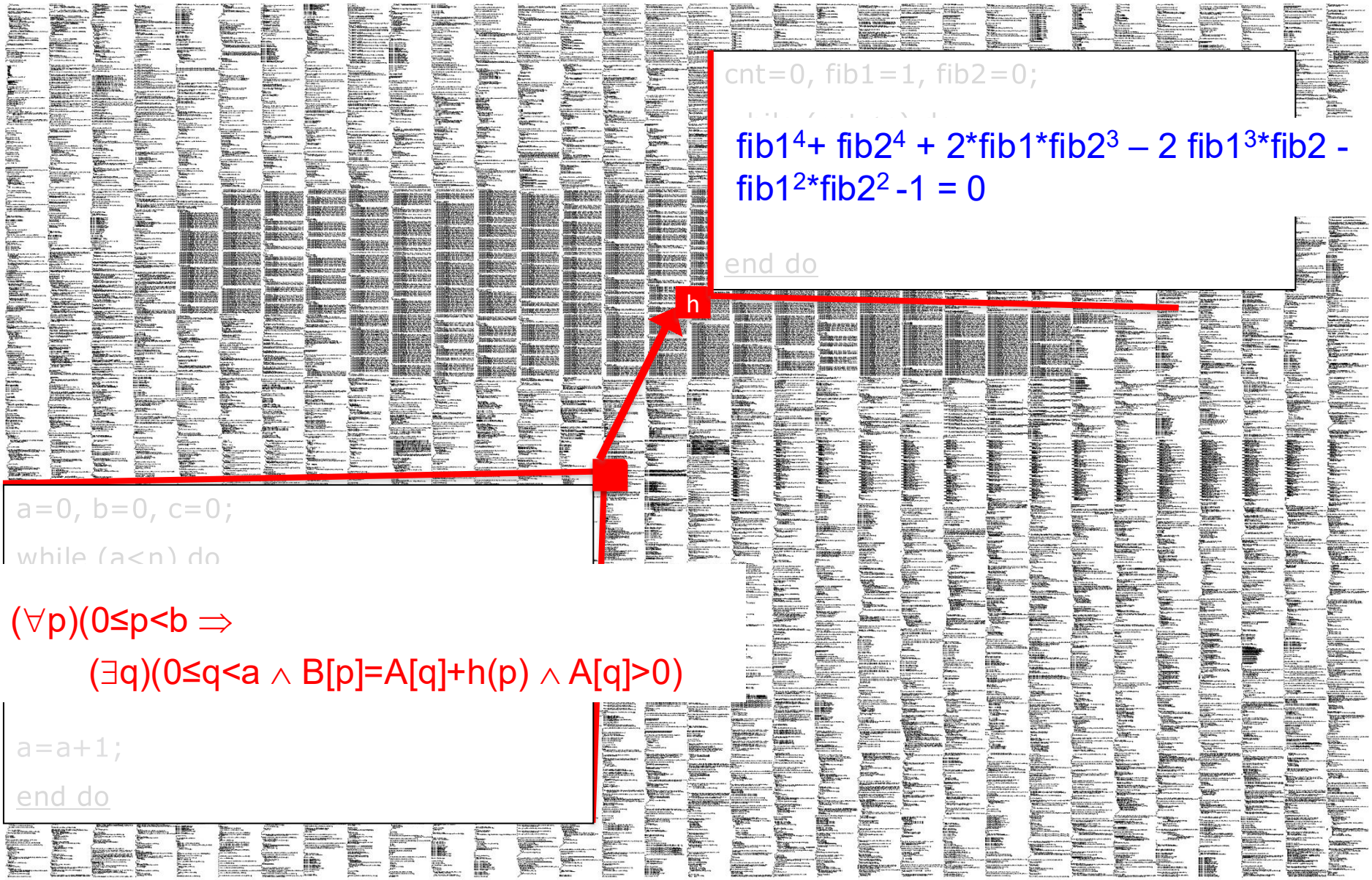
Program property/Invariant:

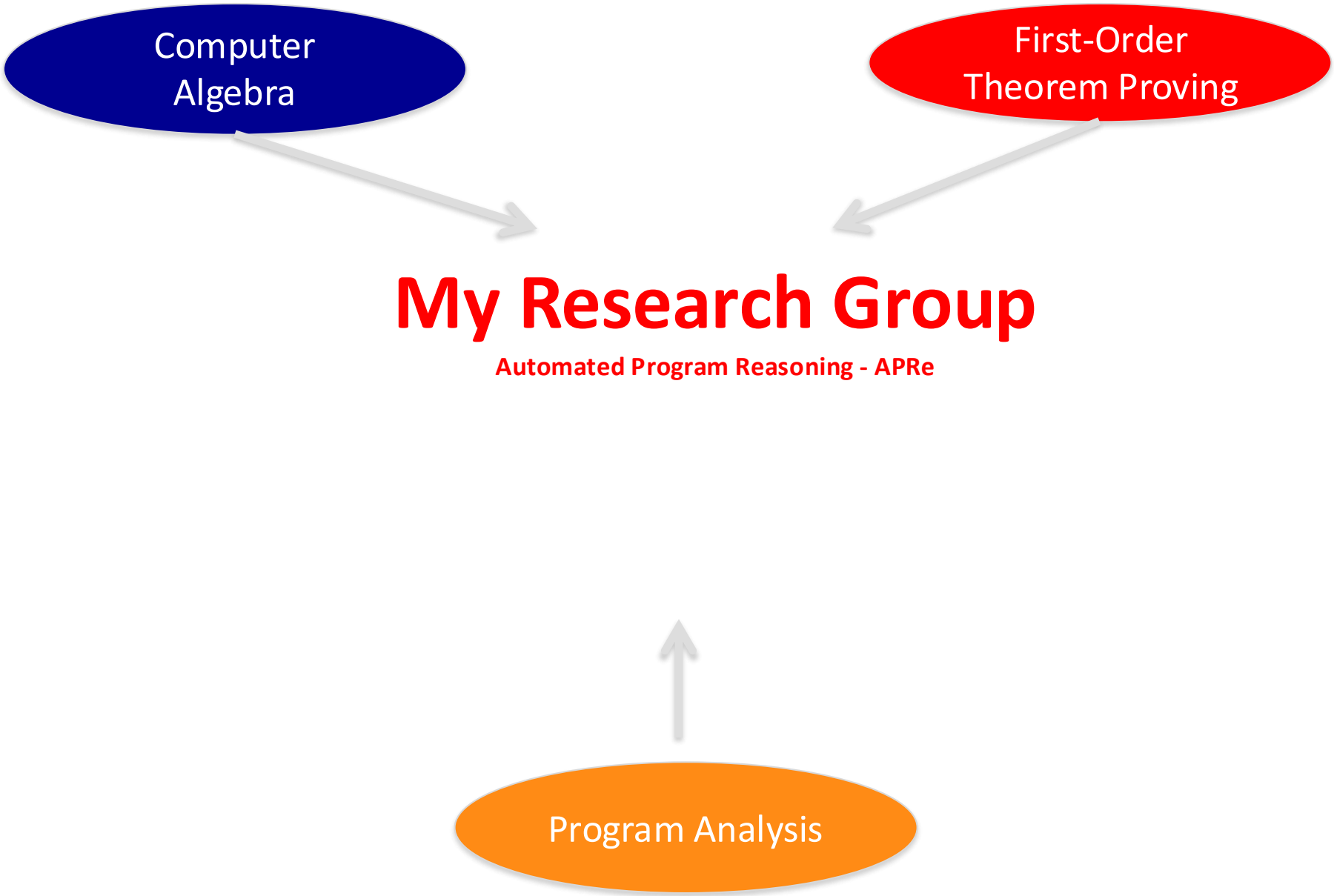
$fib1^4 + fib2^4 + 2*fib1*fib2^3 - 2\ fib1^3*fib2 - fib1^2*fib2^2 - 1 = 0$

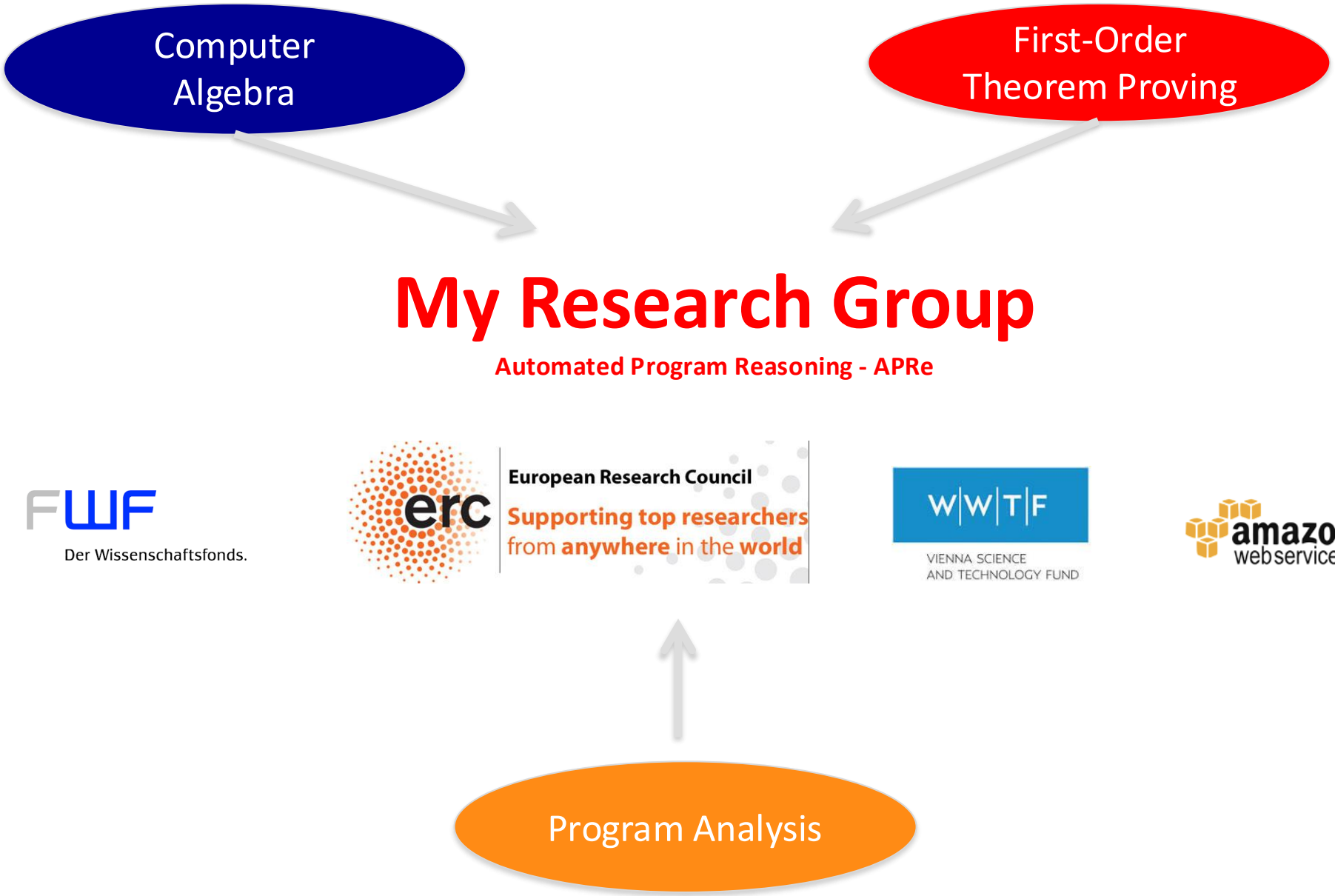
```
a=0, b=0, c=0;
while (a<n) do

if A[a]>0 then B[b]=A[a]+h(b); b=b+1;
          else C[c]=A[a]; c=c+1;

a=a+1;
end do
```

Solvable Program Analysis

replace code by symbolic formulas




Computer Algebra
First-Order Theorem Proving
My Research Group
Automated Program Reasoning - APRe
Program Analysis


Computer Algebra
First-Order Theorem Proving
My Research Group
Automated Program Reasoning - APRe
FWF
Der Wissenschaftsfonds.
erc
European Research Council
Supporting top researchers from anywhere in the world
WWTF
VIENNA SCIENCE AND TECHNOLOGY FUND
amazon web services™
Program Analysis

Algebraic recurrences

Gröbner basis

Statistical moments

Variable elimination

Computer Algebra

First-Order Theorem Proving

My Research Group

Automated Program Reasoning - APRe

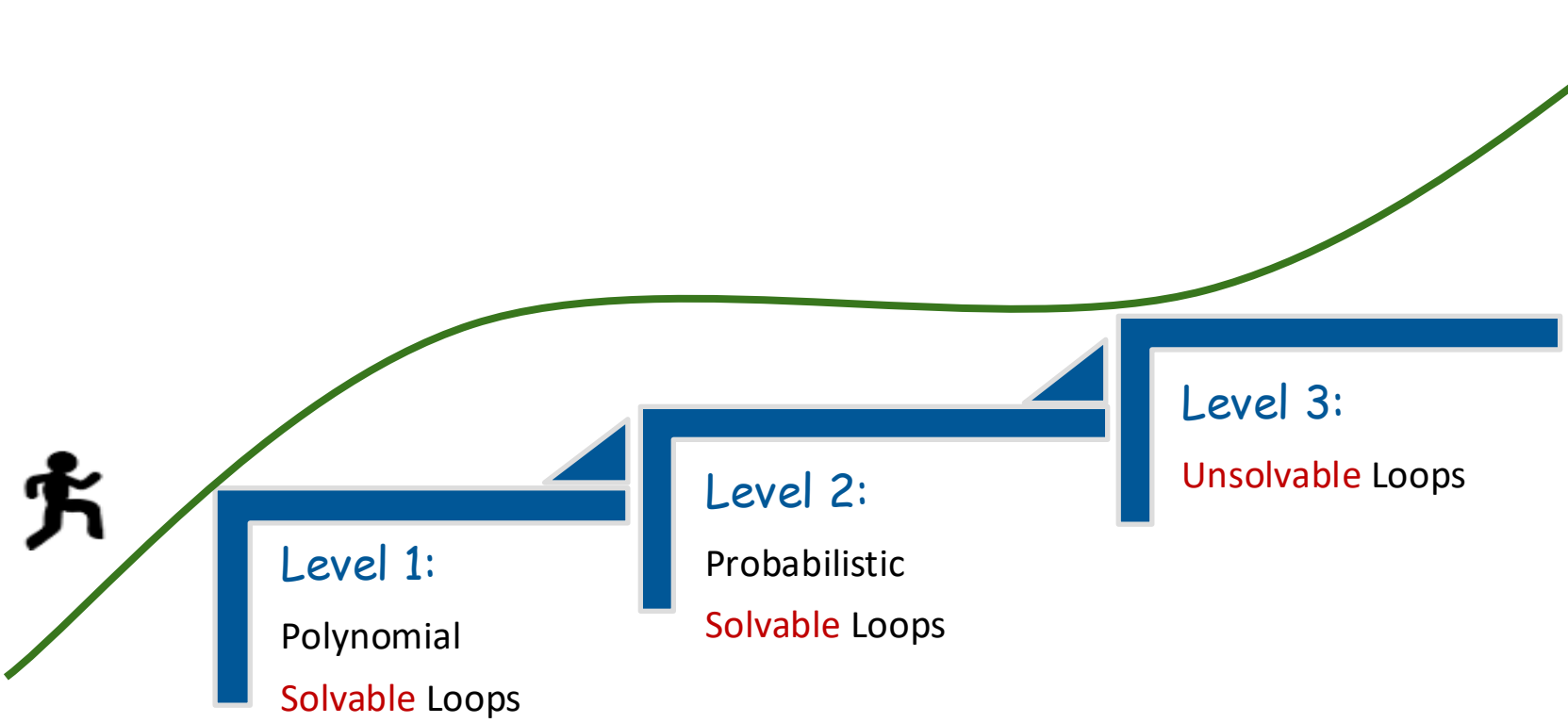
FWF Der Wissenschaftsfonds.

WWTF VIENNA SCIENCE AND TECHNOLOGY FUND

amazon web services™

Program Analysis

Solvable Algebraic Program Analysis



Solvable Algebraic Program Analysis

Level 1: Polynomial Solvable Loops

x:=1; y:=0;

while … do x:=2*x; y:=$\frac{1}{2}$ ∗y+1 end do

Solvable Algebraic Program Analysis

Level 1: Polynomial Solvable Loops

```
x:=1; y:=0;
while … do  x:=2*x;  y:=1/2 *y+1  end do
```

n≥0

1. Express state from (n+1)th iteration in terms of the nth iteration → algebraic recurrences of loop variables

$$\begin{cases} x(n+1) = 2 * x(n) \\ y(n+1) = \frac{1}{2} * y(n) + 1 \end{cases}$$

Solvable Algebraic Program Analysis

Level 1: Polynomial Solvable Loops

```
x:=1; y:=0;
while … do  x:=2*x;  y:=1/2 *y+1  end do
```

n≥0

1. Express state from (n+1)th iteration in terms of the nth iteration → algebraic recurrences of loop variables

$$\begin{cases} x(n+1) = 2 * x(n) \\ y(n+1) = \frac{1}{2} * y(n) + 1 \end{cases}$$

2. Solve recurrences → closed forms of loop variables

$$\begin{cases} x(n) = 2^n * x(0) \\ y(n) = \frac{1}{2^n} * y(0) - \frac{2}{2^n} + 2 \end{cases}$$

Solvable Algebraic Program Analysis

Level 1: Polynomial Solvable Loops

```
x:=1; y:=0;
while … do  x:=2*x;  y:=1/2 *y+1  end do
```

$n \geq 0$, $a=2^n$, $b=2^{-n}$

1. Express state from $(n+1)^{th}$ iteration in terms of the n^{th} iteration → algebraic recurrences of loop variables

$$\begin{cases} x(n+1) = 2 * x(n) \\ y(n+1) = \frac{1}{2} * y(n) + 1 \end{cases}$$

2. Solve recurrences → closed forms of loop variables

$$\begin{cases} x(n) = 2^n * x(0) \\ y(n) = \frac{1}{2^n} * y(0) - \frac{2}{2^n} + 2 \end{cases}$$

3. Derive algebraic dependencies among exponentials in n

$$\begin{cases} x(n) = a * x(0). \\ y(n) = b * y(0) - 2 * b + 2 \\ 0 = a * b - 1 = 2^n * \frac{1}{2^n} - 1 \end{cases}$$

Solvable Algebraic Program Analysis

Level 1: Polynomial Solvable Loops

x:=1; y:=0;

while … do x:=2*x; y:=$\frac{1}{2}$ *y+1 end do

n≥0, a=2^n, b=2^{-n}

1. Express state from (n+1)th iteration in terms of the nth iteration → algebraic recurrences of loop variables

$$\begin{cases} x(n+1) = 2 * x(n) \\ y(n+1) = \frac{1}{2} * y(n) + 1 \end{cases}$$

2. Solve recurrences → closed forms of loop variables

$$\begin{cases} x(n) = 2^n * x(0) \\ y(n) = \frac{1}{2^n} * y(0) - \frac{2}{2^n} + 2 \end{cases}$$

3. Derive algebraic dependencies among exponentials in n

$$\begin{cases} x(n) = a * x(0). \\ y(n) = b * y(0) - 2 * b + 2 \\ 0 = a * b - 1 = 2^n * \frac{1}{2^n} - 1 \end{cases}$$

4. Eliminate expressions in n ← Gröbner basis computation

$$x * y - 2 * x + 2 = 0$$

Solvable Algebraic Program Analysis

Level 1: Polynomial Solvable Loops

x:=1; y:=0;
while … do x:=2*x; y:=$\frac{1}{2}$ *y+1 end do

n≥0, a=2^n, b=2^{-n}

1. Express state from (n+1)th iteration in terms of the nth iteration → algebraic recurrences of loop variables

$$\begin{cases} x(n+1) = 2 * x(n) \\ y(n+1) = \frac{1}{2} * y(n) + 1 \end{cases}$$

2. Solve recurrences → closed forms of loop variables

$$\begin{cases} x(n) = 2^n * x(0) \\ y(n) = \frac{1}{2^n} * y(0) - \frac{2}{2^n} + 2 \end{cases}$$

3. Derive algebraic dependencies among exponentials in n

$$\begin{cases} x(n) = a * x(0). \\ y(n) = b * y(0) - 2 * b + 2 \\ 0 = a * b - 1 = 2^n * \frac{1}{2^n} - 1 \end{cases}$$

4. Eliminate expressions in n ← Gröbner basis computation

$$x * y - 2 * x + 2 = 0$$

→ Finite basis of polynomial invariant ideal

Solvable Algebraic Program Analysis

Level 1: Polynomial Solvable Loops

joint work w A. Humenberger, M. Jaroschek, A. Varonka (ISSAC17,VMCAI18,RAMICS23)

- Loops with polynomial assignments and nested conditionals
  - Structural constraints on assignments with polynomial rhs
    ← C-finite recurrences of loop variables
  - Tests are ignored → non-deterministic programs
- Automation via symbolic summation and variable elimination
  - POLAR tool https://github.com/probing-lab/polar

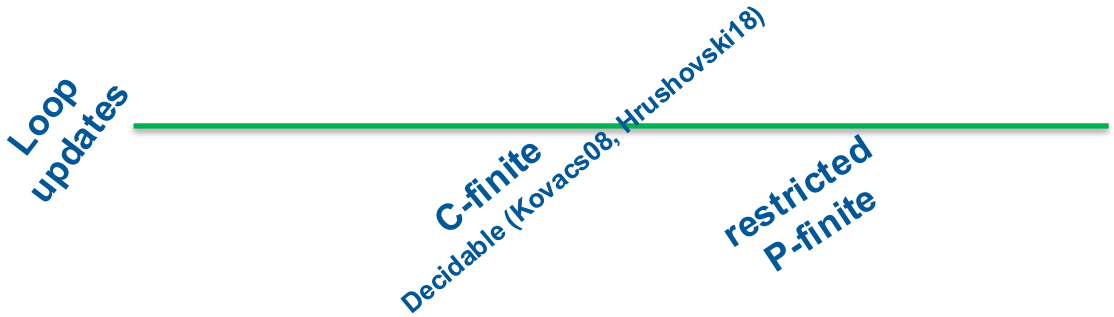
Solvable Algebraic Program Analysis

Level 1: Polynomial Solvable Loops

joint work w A. Humenberger, M. Jaroschek, A. Varonka (ISSAC17,VMCAI18,RAMICS23)

- Loops with polynomial assignments and nested conditionals
  - Structural constraints on assignments with polynomial rhs
    ← C-finite recurrences of loop variables
  - Tests are ignored → non-deterministic programs

- Automation via symbolic summation and variable elimination
  - POLAR tool https://github.com/probing-lab/polar



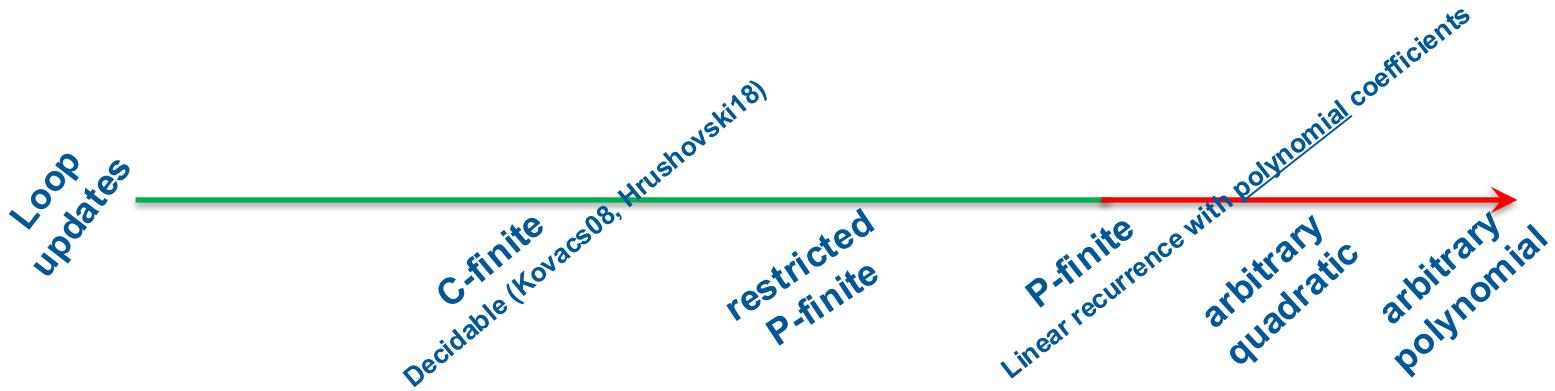
Solvable Algebraic Program Analysis

Level 1: Polynomial Solvable Loops

joint work w A. Humenberger, M. Jaroschek, A. Varonka (ISSAC17,VMCAI18,RAMICS23)

- Loops with polynomial assignments and nested conditionals
  - Structural constraints on assignments with polynomial rhs
    ← C-finite recurrences of loop variables
  - Tests are ignored → non-deterministic programs

- Automation via symbolic summation and variable elimination
  - POLAR tool https://github.com/probing-lab/polar



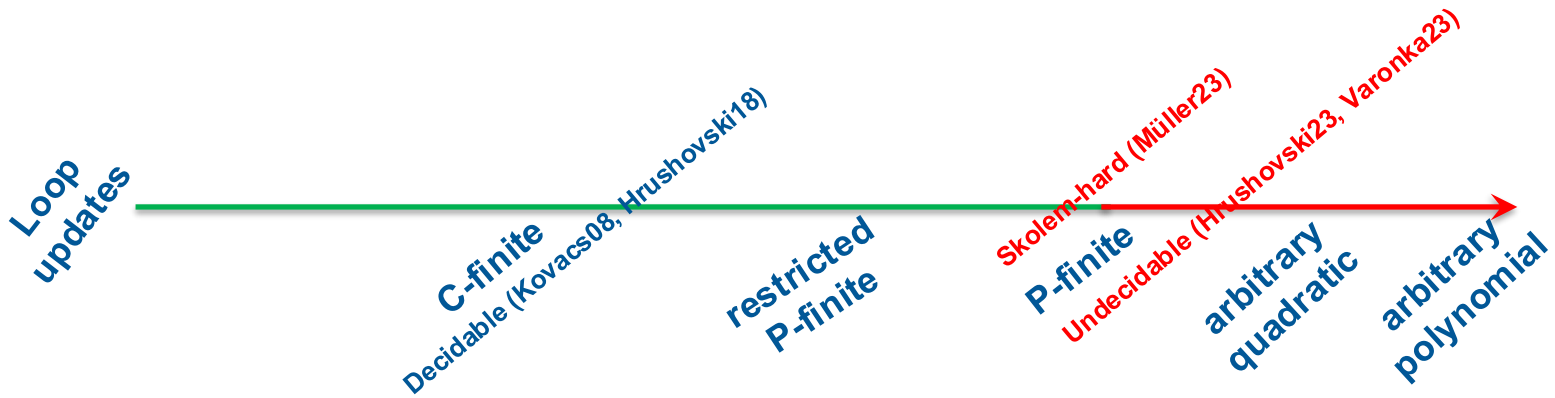
Solvable Algebraic Program Analysis

Level 1: Polynomial Solvable Loops

joint work w A. Humenberger, M. Jaroschek, A. Varonka (ISSAC17,VMCAI18,RAMICS23)

- Loops with polynomial assignments and nested conditionals
  - Structural constraints on assignments with polynomial rhs
    ← C-finite recurrences of loop variables
  - Tests are ignored → non-deterministic programs

- Automation via symbolic summation and variable elimination
  - POLAR tool https://github.com/probing-lab/polar



Solvable Algebraic Program Analysis

Level 1: Polynomial Solvable Loops

joint work w A. Humenberger, M. Jaroschek, A. Varonka (ISSAC17,VMCAI18,RAMICS23)

- Loops with polynomial assignments and nested conditionals
  - Structural constraints on assignments with polynomial rhs
    ← C-finite recurrences of loop variables
  - Tests are ignored → non-deterministic programs
- Automation via symbolic summation and variable elimination
  - POLAR tool https://github.com/probing-lab/polar
- Further challanges: code termination, reactive synthesis, safety verification

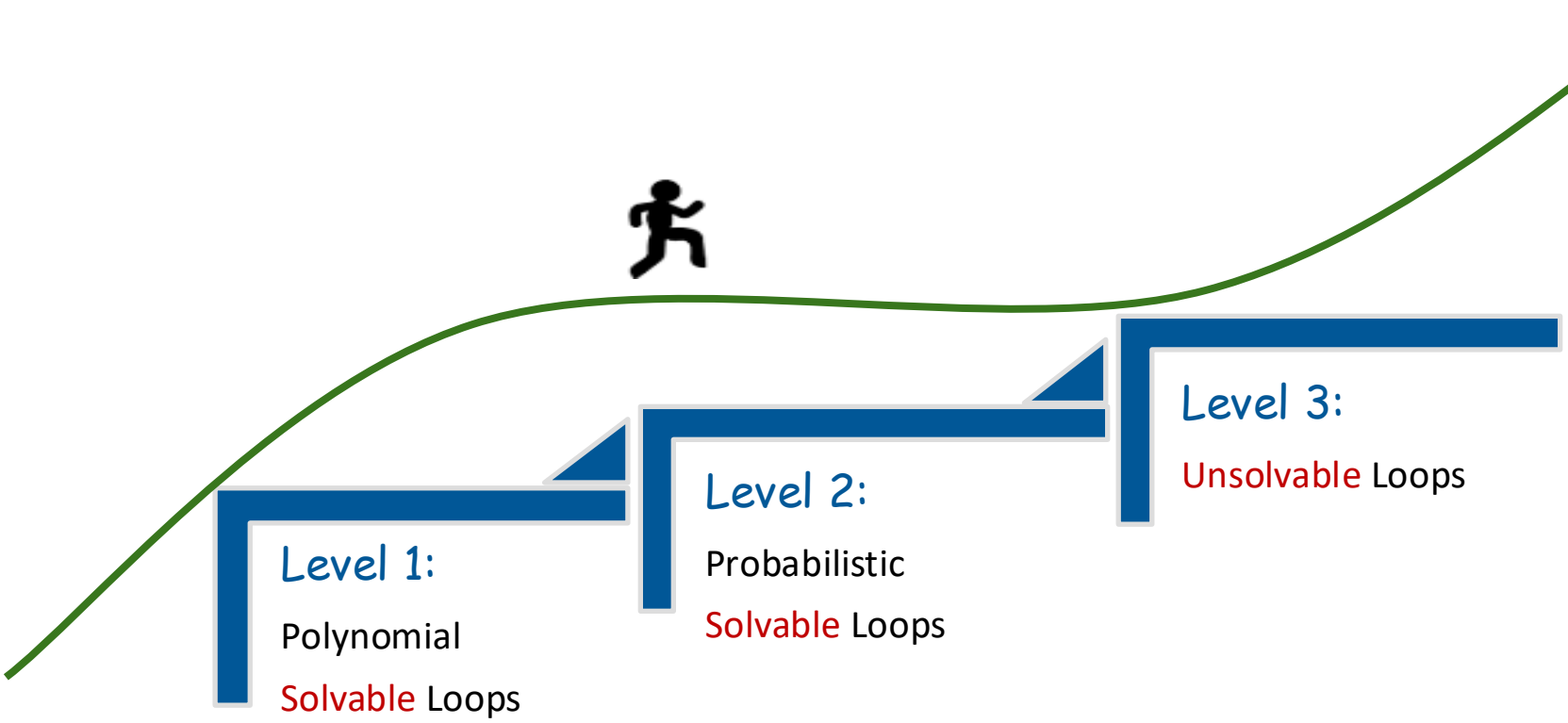
Solvable Algebraic Program Analysis

Level 1: Polynomial Solvable Loops

joint work w A. Humenberger, M. Jaroschek, A. Varonka (ISSAC17,VMCAI18,RAMICS23)

- Loops with polynomial assignments and nested conditionals
  - Structural constraints on assignments with polynomial rhs
    ← C-finite recurrences of loop variables
  - Tests are ignored → non-deterministic programs
- Automation via symbolic summation and variable elimination
  - POLAR tool https://github.com/probing-lab/polar
- Further challanges: code termination, reactive synthesis, safety verification
- Immediate challenge: solvability beyond finite-valued loop tests

Solvable Algebraic Program Analysis



Solvable Algebraic Program Analysis

Level 2: Probabilistic Solvable Loops

```
x:=0;
while … do
   x:=x-1 [1/2] x+1;
end do
```

- What is the behaviour of a probabilistic loop?

Solvable Algebraic Program Analysis

Level 2: Probabilistic Solvable Loops

```
x:=0;
while … do
   x:=x-1 [1/2] x+1;
end do
```

- What is the behaviour of a probabilistic loop?
- What is the expected value of a loop variable, e.g. x?

Solvable Algebraic Program Analysis

Level 2: Probabilistic Solvable Loops

```
x:=0;
while … do
    x:=x-1 [1/2] x+1;
end do
```

- E[x]=0

- What is the behaviour of a probabilistic loop?

- What is the expected value of a loop variable, e.g. x?

Solvable Algebraic Program Analysis

Level 2: Probabilistic Solvable Loops

```
x:=0;
while … do
    x:=x-1 [1/2] x+1;
end do
```

- E[x]=0

```
x:=0;
while … do
    x:=2*x-1 [1/2] 2*x+1;
end do
```

- What is the behaviour of a probabilistic loop?
- What is the expected value of a loop variable, e.g. x?

Solvable Algebraic Program Analysis

Level 2: Probabilistic Solvable Loops

```
x:=0;
while … do
    x:=x-1 [1/2] x+1;
end do
```

- E[x]=0

```
x:=0;
while … do
    x:=2*x-1 [1/2] 2*x+1;
end do
```

- E[x]=0

- What is the behaviour of a probabilistic loop?

- What is the expected value of a loop variable, e.g. x?

Solvable Algebraic Program Analysis

Level 2: Probabilistic Solvable Loops

```
x:=0;
while … do
    x:=x-1 [1/2] x+1;
end do
```

- E[x]=0

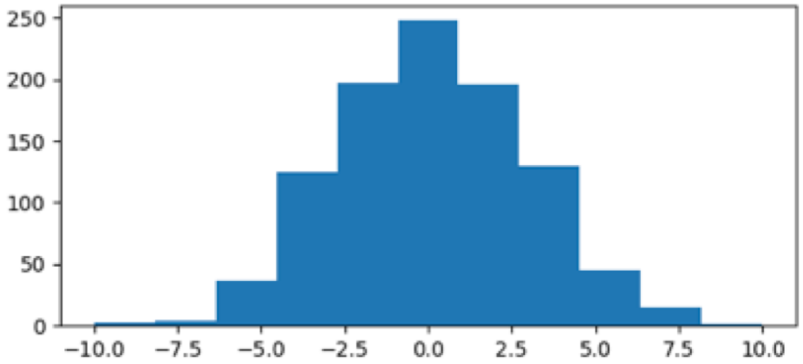
```
x:=0;
while … do
    x:=2*x-1 [1/2] 2*x+1;
end do
```

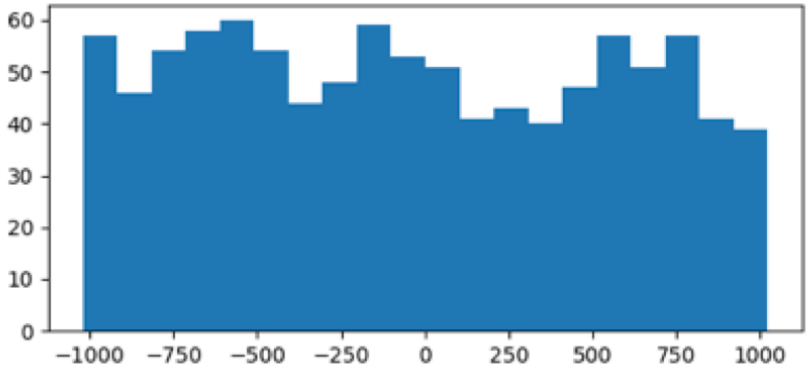
- E[x]=0

- What is the behaviour of a probabilistic loop?
- What is the expected value of a loop variable, e.g. x?
- In both programs above, the expected value of x is the same.
  Yet, the programs are not the same!

Solvable Algebraic Program Analysis

Level 2: Probabilistic Solvable Loops





- What is the behaviour of a probabilistic loop?
- What is the expected value of a loop variable, e.g. x?
- In both programs above, the expected value of x is the same.
  Yet, the programs are not the same!

Solvable Algebraic Program Analysis

Level 2: Probabilistic Solvable Loops

```
x:=0;
while … do
    x:=x-1 [1/2] x+1;
end do
```

- E[x]=0

```
x:=0;
while … do
    x:=2*x-1 [1/2] 2*x+1;
end do
```

- E[x]=0

- What is the behaviour of a probabilistic loop?

- Can we characterize/recover the value distribution of loop variables?

Solvable Algebraic Program Analysis

Level 2: Probabilistic Solvable Loops

```
x:=0;
while … do
    x:=x-1 [1/2] x+1;
end do
```

- E[x]=0

```
x:=0;
while … do
    x:=2*x-1 [1/2] 2*x+1;
end do
```

- E[x]=0

- What is the behaviour of a probabilistic loop?

- Can we characterize/recover the value distribution of loop variables?

  Reason about higher-order statistical moments of variables!

Solvable Algebraic Program Analysis

Level 2: Probabilistic Solvable Loops

```
x:=0;
while … do
   x:=x-1 [1/2] x+1;
end do
```

- E[x(n)]=0, Var[x(n)]= $\frac{4^n}{3}$

```
x:=0;
while … do
   x:=2*x-1 [1/2] 2*x+1;
end do
```

- E[x]=0, Var[x(n)]= n

- What is the behaviour of a probabilistic loop?

using E-variables
(expectations over monomials)

- Can we characterize/recover the value distribution of loop variables?

  Reason about higher-order statistical moments of variables!

Solvable Algebraic Program Analysis

Level 2: Probabilistic Solvable Loops

```
x:=0;
while … do
   x:=x-1 [1/2] x+1;
end do
```

- E[x(n)]=0, Var[x(n)]= $\frac{4^n}{3}$

```
x:=0;
while … do
   x:=2*x-1 [1/2] 2*x+1;
end do
```

- E[x]=0, Var[x(n)]= n

- What is the behaviour of a probabilistic loop?

using E-variables
(expectations over monomials)

- Can we characterize/recover the value distribution of loop variables?

  Reason about higher-order statistical moments of variables!

$$E[x(n+1)] = \tfrac{1}{2}*E[x(n)-1] + \tfrac{1}{2}* E[x(n)+1]$$

Solvable Algebraic Program Analysis

Level 2: Probabilistic Solvable Loops

```
x:=0;
while … do
   x:=x-1 [1/2] x+1;
end do
```

- E[x(n)]=0, Var[x(n)]= $\frac{4^n}{3}$

```
x:=0;
while … do
   x:=2*x-1 [1/2] 2*x+1;
end do
```

- E[x]=0, Var[x(n)]= n

- What is the behaviour of a probabilistic loop?

using E-variables
(expectations over monomials)

- Can we characterize/recover the value distribution of loop variables?

Reason about higher-order statistical moments of variables!

$$E[x(n+1)] = \tfrac{1}{2}*E[x(n)-1] + \tfrac{1}{2}* E[x(n)+1] = \ldots = E[x(n)]$$
$$E[x^2(n+1)] = E[x^2(n)] + \ldots$$

Solvable Algebraic Program Analysis

Level 2: Probabilistic Solvable Loops

joint work w E. Bartocci, M. Stankovic, M. Moosbrugger (ATVA19, TACAS20)

```
x:=0;
while … do
   x:=x-1 [1/2] x+1;
end do
```

- E[x(n)]=0, Var[x(n)]= $\frac{4^n}{3}$

```
x:=0;
while … do
   x:=2*x-1 [1/2] 2*x+1;
end do
```

- E[x]=0, Var[x(n)]= n

DEFINITION 14 (MOMENT-COMPUTABILITY). *A probabilistic loop $\mathcal{P}$ is* moment-computable *if a closed-form of $\mathbb{E}(x_n^k)$ exists and is computable for all $x \in Vars(\mathcal{P})$ and $k \in \mathbb{N}$.*

Solvable Algebraic Program Analysis

Level 2: Probabilistic Solvable Loops

joint work w E. Bartocci, M. Stankovic, M. Moosbrugger (ATVA19, TACAS20)

```
x:=0;
while … do
   x:=x-1 [1/2] x+1;
end do
```

- E[x(n)]=0, Var[x(n)]= $\frac{4^n}{3}$

```
x:=0;
while … do
   x:=2*x-1 [1/2] 2*x+1;
end do
```

- E[x]=0, Var[x(n)]= n

DEFINITION 14 (MOMENT-COMPUTABILITY). *A probabilistic loop $\mathcal{P}$ is* moment-computable *if a closed-form of $\mathbb{E}(x_n^k)$ exists and is computable for all $x \in Vars(\mathcal{P})$ and $k \in \mathbb{N}$.*

Moment-Computability $\stackrel{?}{=}$ Solvability

Solvable Algebraic Program Analysis

Level 2: Probabilistic Solvable Loops

joint work w E. Bartocci, M. Stankovic, M. Moosbrugger (ATVA19, TACAS20, OOPSLA22)

```
x:=0;
while … do
   x:=x-1 [1/2] x+1;
end do
```

- E[x(n)]=0, Var[x(n)]= $\frac{4^n}{3}$

```
x:=0;
while … do
   x:=2*x-1 [1/2] 2*x+1;
end do
```

- E[x]=0, Var[x(n)]= n

DEFINITION 14 (MOMENT-COMPUTABILITY). *A probabilistic loop $\mathcal{P}$ is* moment-computable *if a closed-form of $\mathbb{E}(x_n^k)$ exists and is computable for all $x \in Vars(\mathcal{P})$ and $k \in \mathbb{N}$.*

Moment-Computability = Solvability

Theorem 6 (Moment-Computability). *A probabilistic loop $\mathcal{P}$ is moment-computable if* (1) none of its non-finite variables depends on itself polynomially, and (2) if the variables in all if-conditions are finite.

Solvable Algebraic Program Analysis

Level 2: Probabilistic Solvable Loops

joint work w E. Bartocci, M. Stankovic, M. Moosbrugger (ATVA19, TACAS20, OOPSLA22)

A

```
real x := -1, y :=  1;
real s := 0, f := 0, d;

while (true){
    f    := 1 [3/4] 0;
    x    := x + f * rand(1-d,  1+d);
    y    := y + f * rand(2-2d,2+2d);
    s    := x + y;
}
```

$[\quad] = -$

$\mathrm{Var}[\quad] = \quad +$

B

```
real x := rand (-9, 7), y := rand (-7, 9);
real s := 0, f := 0;

while (true){
    f    := 1 [3/4] 0;
    x    := x + f * rand(-3,5);
    y    := y + f * rand(-6,10);
    s    := x + y;
}
```

$[\quad] = -$

$\mathrm{Var}[\quad] = \quad + \quad$

Higher-order moments of loop variables are computable/solvable:

- Parametrized distributions
- Random polynomial assignments → C-finite recurrences over moments
- Finite-valued multi-path conditions

Solvable Algebraic Program Analysis

Level 2: Probabilistic Solvable Loops

joint work w E. Bartocci, M. Stankovic, M. Moosbrugger (ATVA19, TACAS20, OOPSLA22)

```
f, x, y, s = 0, -1, 1, 0
while (true):
  f = 1 [3/4] 0
  x = x + f*rand(1-d, 1+d)
  y = y + f*rand(2-2d, 2+2d)
  s = x + y
```

Goal: Higher-order Moments, e.g. `s^2`

Solvable Algebraic Program Analysis

Level 2: Probabilistic Solvable Loops

joint work w E. Bartocci, M. Stankovic, M. Moosbrugger (ATVA19, TACAS20, OOPSLA22)

- Expected values of monomials
  → E-variables: moments in time

```
f, x, y, s = 0, -1, 1, 0
while (true):
  f = 1 [3/4] 0
  x = x + f*rand(1-d, 1+d)
  y = y + f*rand(2-2d, 2+2d)
  s = x + y
```

Goal: Higher-order Moments, e.g. `s^2`

Solvable Algebraic Program Analysis

Level 2: Probabilistic Solvable Loops

joint work w E. Bartocci, M. Stankovic, M. Moosbrugger (ATVA19, TACAS20, OOPSLA22)

- Expected values of monomials
  → E-variables: moments in time

```
f, x, y, s = 0, -1, 1, 0
while (true):
  f = 1 [3/4] 0
  x = x + f*rand(1-d, 1+d)
  y = y + f*rand(2-2d, 2+2d)
  s = x + y
```

Probabilisic updates

$x_i = a_i x_i + P_i(x_1, \ldots, x_{i-1})\ [p_i]\ b_i x_i + Q_i(x_1, \ldots, x_{i-1})$

Goal: Higher-order Moments, e.g. `s^2`

Solvable Algebraic Program Analysis

Level 2: Probabilistic Solvable Loops

joint work w E. Bartocci, M. Stankovic, M. Moosbrugger (ATVA19, TACAS20, OOPSLA22)

- Expected values of monomials
  → E-variables: moments in time

```
f, x, y, s = 0, -1, 1, 0
while (true):
  f = 1 [3/4] 0
  x = x + f*rand(1-d, 1+d)
  y = y + f*rand(2-2d, 2+2d)
  s = x + y
```

Probabilisic updates

$x_i = a_i x_i + P_i(x_1, \ldots, x_{i-1})\ [p_i]\ b_i x_i + Q_i(x_1, \ldots, x_{i-1})$

Goal: Higher-order Moments, e.g. `s^2`

- Stochastic recurrences over E-variables

$$E[x_i(n+1)] = p_i \cdot E\big[a_i x_i(n) + P_i(x_1(n), \ldots, x_{i-1}(n))\big] + (1-p_i) \cdot E\big[b_i x_i(n) + Q_i(x_1(n), \ldots, x_{i-1}(n))\big].$$

Solvable Algebraic Program Analysis

Level 2: Probabilistic Solvable Loops

joint work w E. Bartocci, M. Stankovic, M. Moosbrugger (ATVA19, TACAS20, OOPSLA22)

- Expected values of monomials
  → E-variables: moments in time

- Computing with E-variables

E[c] = c
E[X+cY] = E[X] + cE[Y]
E[X*Y] ≠ E[X] * E[Y], unless X,Y are independent

```
f, x, y, s = 0, -1, 1, 0
while (true):
  f = 1 [3/4] 0
  x = x + f*rand(1-d, 1+d)
  y = y + f*rand(2-2d, 2+2d)
  s = x + y
```

Goal: Higher-order Moments, e.g. `s^2`

- Stochastic recurrences over E-variables

$$E[x_i(n+1)] = p_i \cdot E\big[a_i x_i(n) + P_i(x_1(n), \ldots, x_{i-1}(n))\big] + (1-p_i) \cdot E\big[b_i x_i(n) + Q_i(x_1(n), \ldots, x_{i-1}(n))\big].$$

Solvable Algebraic Program Analysis

Level 2: Probabilistic Solvable Loops

joint work w E. Bartocci, M. Stankovic, M. Moosbrugger (ATVA19, TACAS20, OOPSLA22)

- Expected values of monomials
  → E-variables: moments in time

- Computing with E-variables

E[c] = c
E[X+cY] = E[X] + cE[Y]
E[X*Y] ≠ E[X] * E[Y], unless X,Y are independent

```
f, x, y, s = 0, -1, 1, 0
while (true):
  f = 1 [3/4] 0
  x = x + f*rand(1-d, 1+d)
  y = y + f*rand(2-2d, 2+2d)
  s = x + y
```

Goal: Higher-order Moments, e.g. `s^2`

- Stochastic recurrences over E-variables

$$E[x_i(n+1)] = p_i \cdot E\big[a_i x_i(n) + P_i(x_1(n), \ldots, x_{i-1}(n))\big] + (1-p_i) \cdot E\big[b_i x_i(n) + Q_i(x_1(n), \ldots, x_{i-1}(n))\big].$$

- Solve stochastic recurrences: closed forms over E-variables Level 1 ✓

Solvable Algebraic Program Analysis

Level 2: Probabilistic Solvable Loops

joint work w E. Bartocci, M. Stankovic, M. Moosbrugger (ATVA19, TACAS20, OOPSLA22)

```
f, x, y, s = 0, -1, 1, 0
while (true):
  f = 1 [3/4] 0
  x = x + f*rand(1-d, 1+d)
  y = y + f*rand(2-2d, 2+2d)
  s = x + y
```

goal: {s^2}

Solvable Algebraic Program Analysis

Level 2: Probabilistic Solvable Loops

joint work w E. Bartocci, M. Stankovic, M. Moosbrugger (ATVA19, TACAS20, OOPSLA22)

1. Set S := goal
2. While S is not empty:
3. Pick an E-variable from S
4. Get recurrence over E-variables
5. Add new E-variables to S
6. Solve the system of recurrences

```
f, x, y, s = 0, -1, 1, 0
while (true):
  f = 1 [3/4] 0
  x = x + f*rand(1-d, 1+d)
  y = y + f*rand(2-2d, 2+2d)
  s = x + y
```

goal: {s^2}

$S = \{\mathbf{s^2}\} \rightarrow \mathbf{s^2} \rightarrow E[s^2(n+1)] = E[(x(n+1)+y(n+1))^2]$

Solvable Algebraic Program Analysis

Level 2: Probabilistic Solvable Loops

joint work w E. Bartocci, M. Stankovic, M. Moosbrugger (ATVA19, TACAS20, OOPSLA22)

1. Set S := goal
2. While S is not empty:
3. Pick an E-variable from S
4. Get recurrence over E-variables
5. Add new E-variables to S
6. Solve the system of recurrences

```
f, x, y, s = 0, -1, 1, 0
while (true):
  f = 1 [3/4] 0
  x = x + f*rand(1-d, 1+d)
  y = y + f*rand(2-2d, 2+2d)
  s = x + y
```

goal: {s^2}

$S = \{\mathbf{s^2}\} \rightarrow \mathbf{s^2} \rightarrow E[s^2(n+1)] = E[(x(n+1)+y(n+1))^2]$

$\rightarrow E[s^2(n+1)] = E[x^2(n+1)] + 2\,E[xy(n+1)] + E[y^2(n+1)]$

$\rightarrow S = \{\mathbf{s^2}, \mathbf{x^2}, \mathbf{xy}, \mathbf{y^2}\}$

Solvable Algebraic Program Analysis

Level 2: Probabilistic Solvable Loops

joint work w E. Bartocci, M. Stankovic, M. Moosbrugger (ATVA19, TACAS20, OOPSLA22)

1. Set S := goal
2. While S is not empty:
3. Pick an E-variable from S
4. Get recurrence over E-variables
5. Add new E-variables to S
6. Solve the system of recurrences

```
f, x, y, s = 0, -1, 1, 0
while (true):
  f = 1 [3/4] 0
  x = x + f*rand(1-d, 1+d)
  y = y + f*rand(2-2d, 2+2d)
  s = x + y
```

goal: {s^2}

$S = \{s^2, x^2, xy, y^2\} \rightarrow x^2$

Solvable Algebraic Program Analysis

Level 2: Probabilistic Solvable Loops

joint work w E. Bartocci, M. Stankovic, M. Moosbrugger (ATVA19, TACAS20, OOPSLA22)

1. Set S := goal
2. While S is not empty:
3. Pick an E-variable from S
4. Get recurrence over E-variables
5. Add new E-variables to S
6. Solve the system of recurrences

```
f, x, y, s = 0, -1, 1, 0
while (true):
  f = 1 [3/4] 0
  x = x + f*rand(1-d, 1+d)
  y = y + f*rand(2-2d, 2+2d)
  s = x + y
```

goal: {s^2}

$$S= \{\mathbf{s^2}, \mathbf{x^2}, \mathbf{xy}, \mathbf{y^2}\} \rightarrow \mathbf{x^2} \rightarrow E[x^2(n+1)] = E[\big(x(n)+f(n+1)*rand(1-d,1+d)\big)^2]$$

Solvable Algebraic Program Analysis

Level 2: Probabilistic Solvable Loops

joint work w E. Bartocci, M. Stankovic, M. Moosbrugger (ATVA19, TACAS20, OOPSLA22)

1. Set S := goal
2. While S is not empty:
3. Pick an E-variable from S
4. Get recurrence over E-variables
5. Add new E-variables to S
6. Solve the system of recurrences

```
f, x, y, s = 0, -1, 1, 0
while (true):
  f = 1 [3/4] 0
  x = x + f*rand(1-d, 1+d)
  y = y + f*rand(2-2d, 2+2d)
  s = x + y
```

goal: {s^2}

$S = \{\mathbf{s^2}, \mathbf{x^2}, \mathbf{xy}, \mathbf{y^2}\} \rightarrow \mathbf{x^2} \rightarrow E[x^2(n+1)] = E[\big(x(n)+f(n+1)*rand(1-d,1+d)\big)^2]$

$\rightarrow E[x^2(n+1)] = E[x^2(n)] + 2\,E[x(n)]*E[f(n+1)] + E[f^2(n+1)]\left(1 + \frac{d^2}{3}\right)$

Solvable Algebraic Program Analysis

Level 2: Probabilistic Solvable Loops

joint work w E. Bartocci, M. Stankovic, M. Moosbrugger (ATVA19, TACAS20, OOPSLA22)

1. Set S := goal
2. While S is not empty:
3. Pick an E-variable from S
4. Get recurrence over E-variables
5. Add new E-variables to S
6. Solve the system of recurrences

```
f, x, y, s = 0, -1, 1, 0
while (true):
  f = 1 [3/4] 0
  x = x + f*rand(1-d, 1+d)
  y = y + f*rand(2-2d, 2+2d)
  s = x + y
```

goal: {s^2}

$S = \{\mathbf{s^2}, \mathbf{x^2}, \mathbf{xy}, \mathbf{y^2}\} \rightarrow \mathbf{x^2} \rightarrow E[x^2(n+1)] = E[\big(x(n)+f(n+1)*rand(1-d,1+d)\big)^2]$

$\rightarrow E[x^2(n+1)] = E[x^2(n)] + 2\,E[x(n)]*E[f(n+1)] + E[f^2(n+1)]\left(1 + \frac{d^2}{3}\right)$

$\rightarrow S = \{\mathbf{s^2}, \mathbf{x^2}, \mathbf{xy}, \mathbf{y^2}, \mathbf{x}, \mathbf{f}, \mathbf{f^2}\}$

and so on …

Solvable Algebraic Program Analysis

Level 2: Probabilistic Solvable Loops

joint work w E. Bartocci, M. Stankovic, M. Moosbrugger (ATVA19, TACAS20, OOPSLA22)

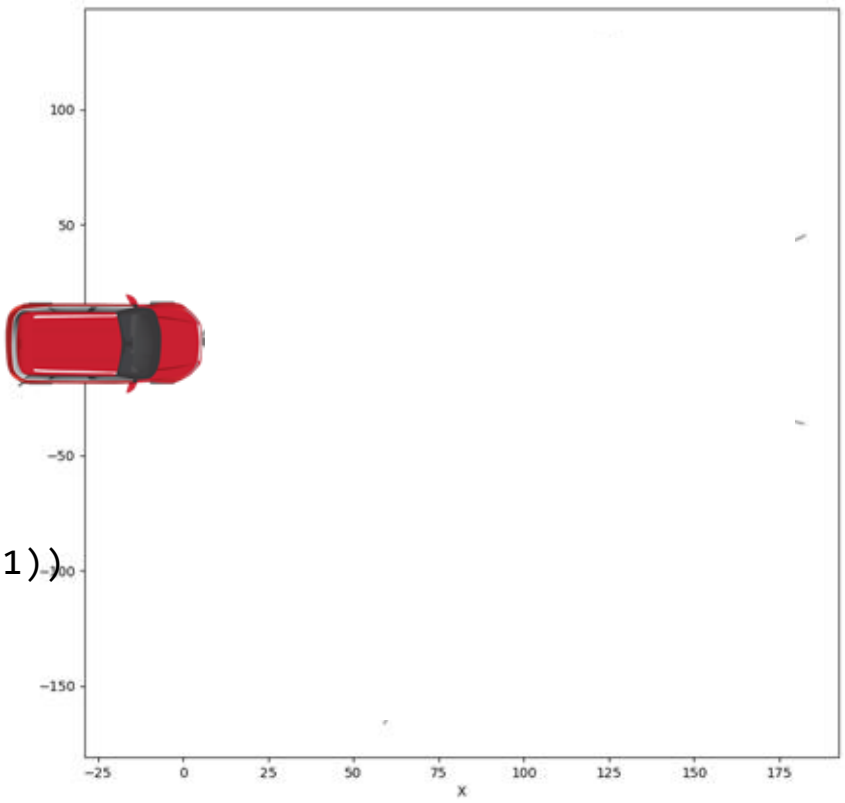
- Level 1 + Random polynomial assignments + finite-valued conditions
  - C-finite recurrences of E-variables
- Automation via symbolic summation and moment-based computation
  - POLAR tool https://github.com/probing-lab/polar
- Further challenges: dynamic control, sensitivity, probabilistic inferences

Solvable Algebraic Program Analysis

Level 2: Probabilistic Solvable Loops – Autonomous Car

joint work w M. Moosbrugger, J. Müllner (iFM23, POPL24)

```
v0 = 10, t = 0.1, K = -0.5
psi = Normal(0, 0.01)
v = Uniform(6.5, 8.0)
x = Uniform(-0.1, 0.1)
y = Uniform(-0.5, -0.3)
while *:
   x = x + t*v*cos(psi)
   y = y + t*v*sin(psi)
   v = v + t*(K*(v - v0) + Uniform(-0.1, 0.1))
   psi = psi + Normal(0, 0.01)
```

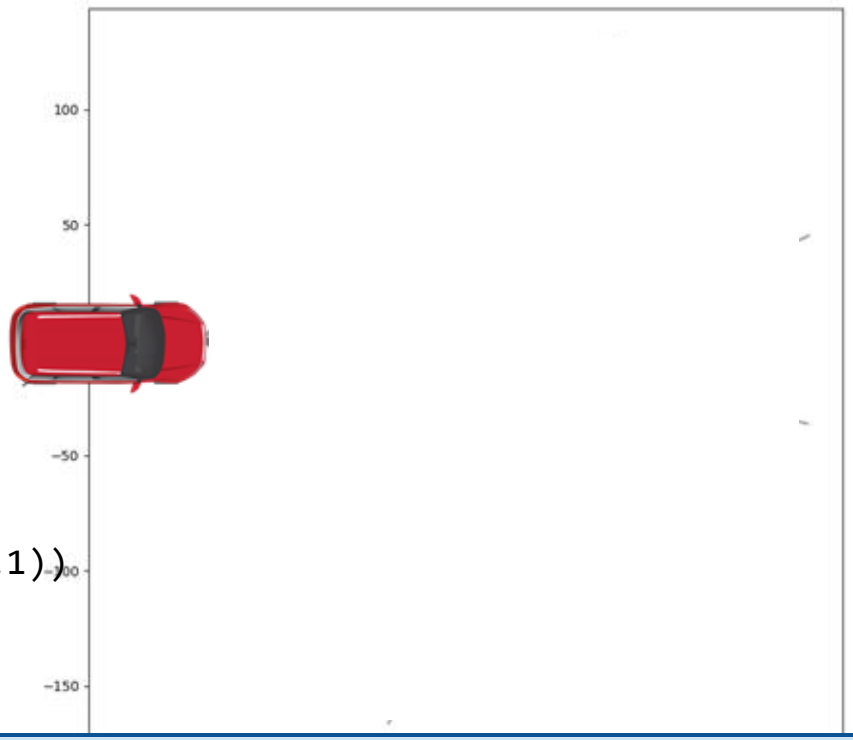


Solvable Algebraic Program Analysis

Level 2: Probabilistic Solvable Loops – Autonomous Car

joint work w M. Moosbrugger, J. Müllner (iFM23, POPL24)

```
v0 = 10, t = 0.1, K = -0.5
psi = Normal(0, 0.01)
v = Uniform(6.5, 8.0)
x = Uniform(-0.1, 0.1)
y = Uniform(-0.5, -0.3)
while *:
  x = x + t*v*cos(psi)
  y = y + t*v*sin(psi)
  v = v + t*(K*(v - v0) + Uniform(-0.1, 0.1))
  psi = psi + Normal(0, 0.01)
```



What is happening with the vehicle's position?

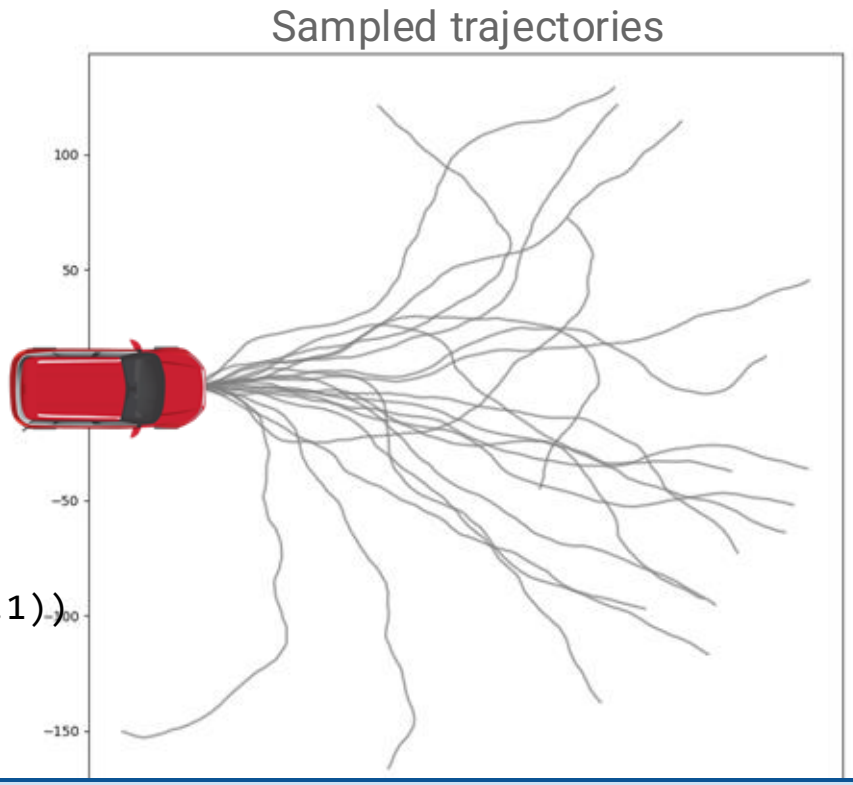
Is there a tendency towards a direction? Variance?

Solvable Algebraic Program Analysis

Level 2: Probabilistic Solvable Loops – Autonomous Car

joint work w M. Moosbrugger, J. Müllner (iFM23, POPL24)

```
v0 = 10, t = 0.1, K = -0.5
psi = Normal(0, 0.01)
v = Uniform(6.5, 8.0)
x = Uniform(-0.1, 0.1)
y = Uniform(-0.5, -0.3)
while *:
  x = x + t*v*cos(psi)
  y = y + t*v*sin(psi)
  v = v + t*(K*(v - v0) + Uniform(-0.1, 0.1))
  psi = psi + Normal(0, 0.01)
```



What is happening with the vehicle's position?

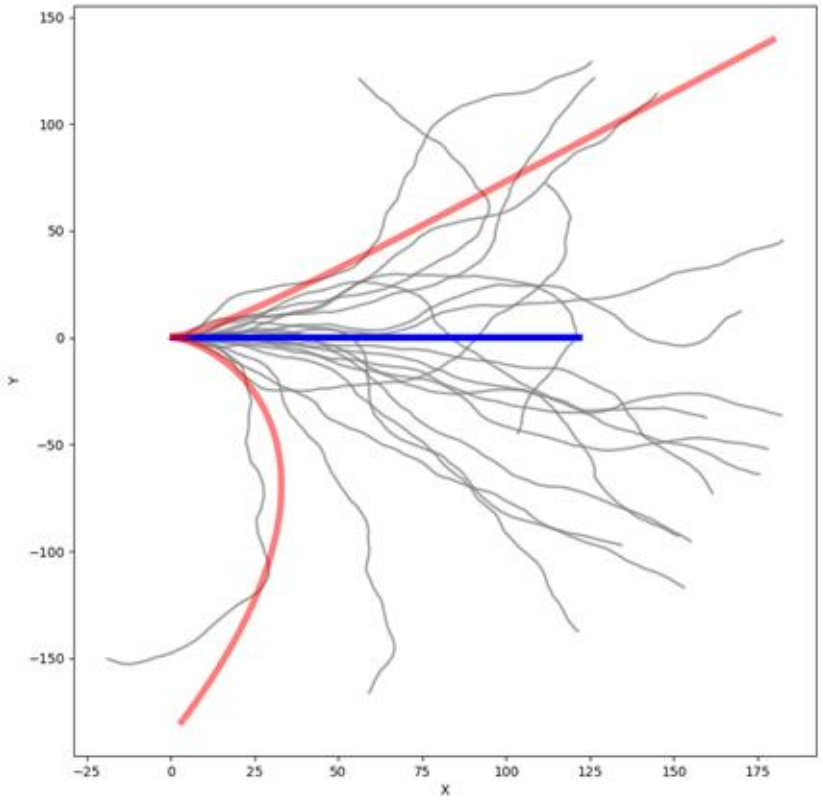
Is there a tendency towards a direction? Variance?

Solvable Algebraic Program Analysis

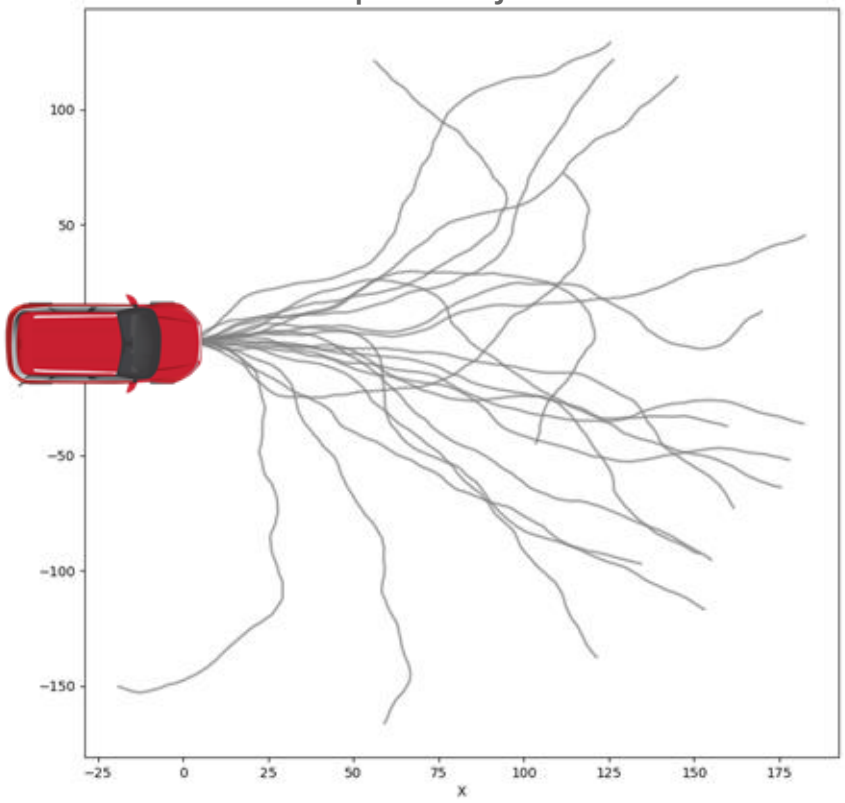
Level 2: Probabilistic Solvable Loops – **Autonomous Car**

joint work w M. Moosbrugger, J. Müllner (iFM23, POPL24)

POLAR



Sampled trajectories



We compute exact expected value, standard deviation, without sampling.

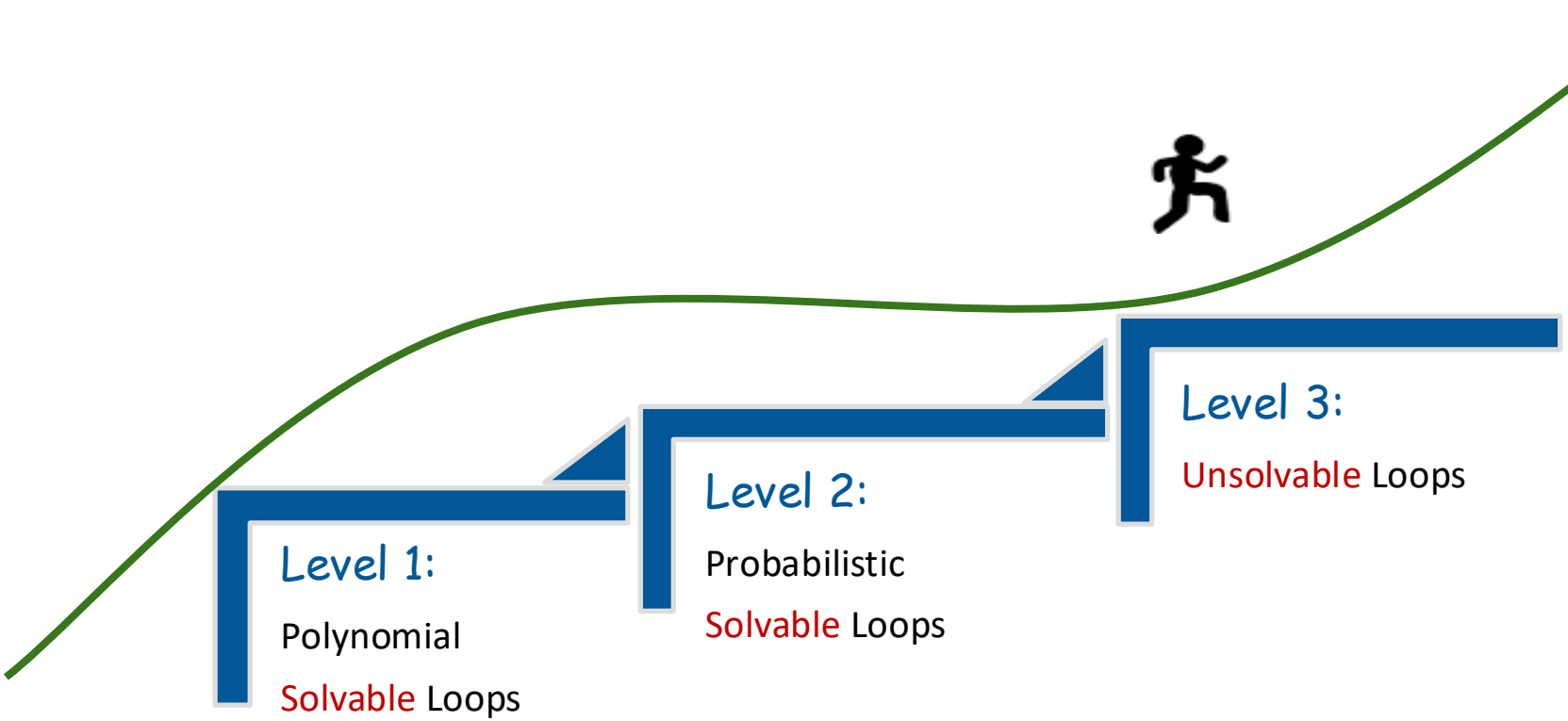
Solvable Algebraic Program Analysis

Level 2: Probabilistic Solvable Loops

joint work w E. Bartocci, M. Stankovic, M. Moosbrugger (ATVA19, TACAS20, OOPSLA22)

- Level 1 + Random polynomial assignments + finite-valued conditions
  - C-finite recurrences of E-variables
- Automation via symbolic summation and moment-based computation
  - POLAR tool https://github.com/probing-lab/polar
- Further challenges: dynamic control, sensitivity, probabilistic inferences
- Immediate challenge: solvability of (positive) almost sure termination

Solvable Algebraic Program Analysis



Solvable Algebraic Program Analysis

Level 3: Unsolvable Loops

```
a:=-2; b:=3; y:=0;
while … do
   a:=2*a+b^2;
   b:=2*b-b^2;
   y:=1/2 *y+1
end do
```

Solvable Algebraic Program Analysis

Level 3: Unsolvable Loops

```
a:=-2; b:=3; y:=0;
while … do
   a:=2*a+b^2;
   b:=2*b-b^2;
   y:=1/2 *y+1
end do
```

- Polynomial updates → non-C-finite recurrences

Level 1 and Level 2 ⊗

Solvable Algebraic Program Analysis

Level 3: Unsolvable Loops

```
a:=-2; b:=3; y:=0;
while … do
   a:=2*a+b^2;
   b:=2*b-b^2;
   y:=1/2 *y+1
end do
```

n≥0

- Polynomial updates → non-C-finite recurrences Level 1 and Level 2 ⊗
- Yet, x(n)=a(n)+b(n) satisfy a C-finite recurrence: a(n+1)+b(n+1)=2*(a(n)+b(n))

Solvable Algebraic Program Analysis

Level 3: Unsolvable Loops → Level 1: Solvable Loops

```
a:=-2; b:=3; y:=0;
while … do
   a:=2*a+b^2;
   b:=2*b-b^2;
   y:=1/2 *y+1
end do
```

n≥0

```
x:=1; y:=0;
while … do
   x:=2*x;
   y:=1/2 *y+1
end do
```

- Polynomial updates → non-C-finite recurrences — Level 1 and Level 2 ⊗
- Yet, x(n)=a(n)+b(n) satisfy a C-finite recurrence: a(n+1)+b(n+1)=2*(a(n)+b(n))
- Unsolvable loop over a, b, y → Solvable loop over x, y — Level 1 ✓

Solvable Algebraic Program Analysis

Level 3: Unsolvable Loops → Level 1: Solvable Loops

```
a:=-2; b:=3; y:=0;
while … do
   a:=2*a+b^2;
   b:=2*b-b^2;
   y:=1/2 *y+1
end do
```

n≥0

```
x:=1; y:=0;
while … do
   x:=2*x;
   y:=1/2 *y+1
end do
```

- Polynomial updates → non-C-finite recurrences Level 1 and Level 2 ⊗
- Yet, x(n)=a(n)+b(n) satisfy a C-finite recurrence: a(n+1)+b(n+1)=2*(a(n)+b(n))
- Unsolvable loop over a, b, y → Solvable loop over x, y Level 1 ✓

  Invariant: (a+b)*y-2*(a+b)+2=0 ← x*y-2*x+2=0

Solvable Algebraic Program Analysis

Level 3: Unsolvable Loops → Level 1: Solvable Loops

```
a:=-2; b:=3; y:=0;
while … do
   a:=2*a+b^2;
   b:=2*b-b^2;
   y:=1/2 *y+1
end do
```

n≥0

```
x:=1; y:=0;
while … do
   x:=2*x;
   y:=1/2 *y+1
end do
```

- Polynomial updates → non-C-finite recurrences Level 1 and Level 2 ⊗

Constant propagation
(compiler optimization)

C-finite recurrence: a(n+1)+b(n+1)=2*(a(n)+b(n))

- Unsolvable loop over a, b, y → Solvable loop over x, y Level 1 ✓

Invariant: (a+b)*y-2*(a+b)+2=0 ← x*y-2*x+2=0

Solvable Algebraic Program Analysis

Level 3: Unsolvable Loops → Level 1 or Level 2: Solvable Loops

joint work w D. Amrollahi, E. Bartocci, G. Kenison, M. Stankovic, M. Moosbrugger (SAS22)

- Level 1 or Level 2 + non-C-finite recurrences
- Compute polynomial relations P over variables with non C-finite recurrences
- If P is C-finite expression, use P to solve unsolvable loops
- Automation via variable dependency analysis and polynomial constraint solving
  - POLAR tool

https://github.com/probing-lab/polar

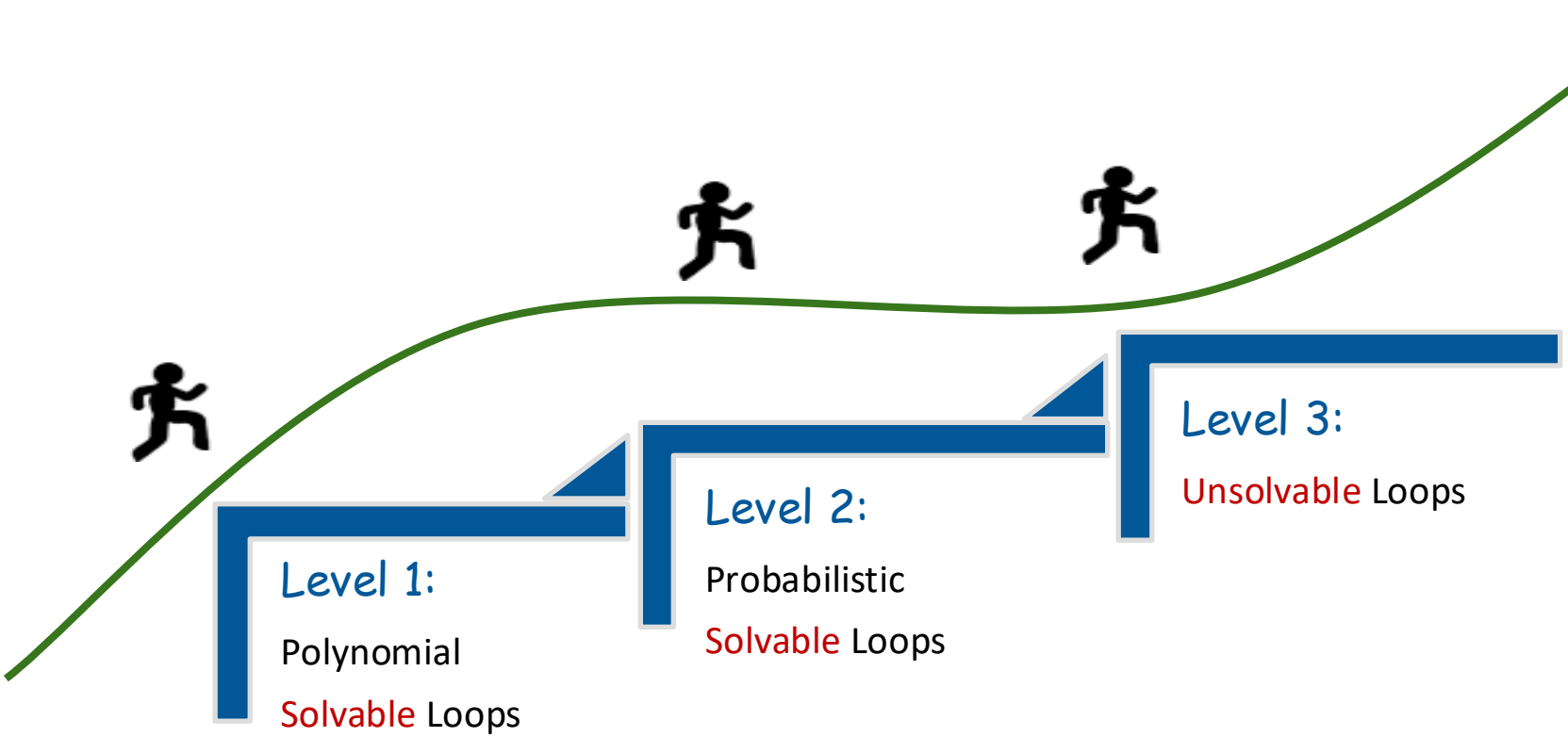
Solvable Algebraic Program Analysis

Level 3: Unsolvable Loops → Level 1 or Level 2: Solvable Loops

joint work w D. Amrollahi, E. Bartocci, G. Kenison, M. Stankovic, M. Moosbrugger (SAS22)
join work w A. Varonka, G. Kenison, S. Hitarth (STACS 2026)

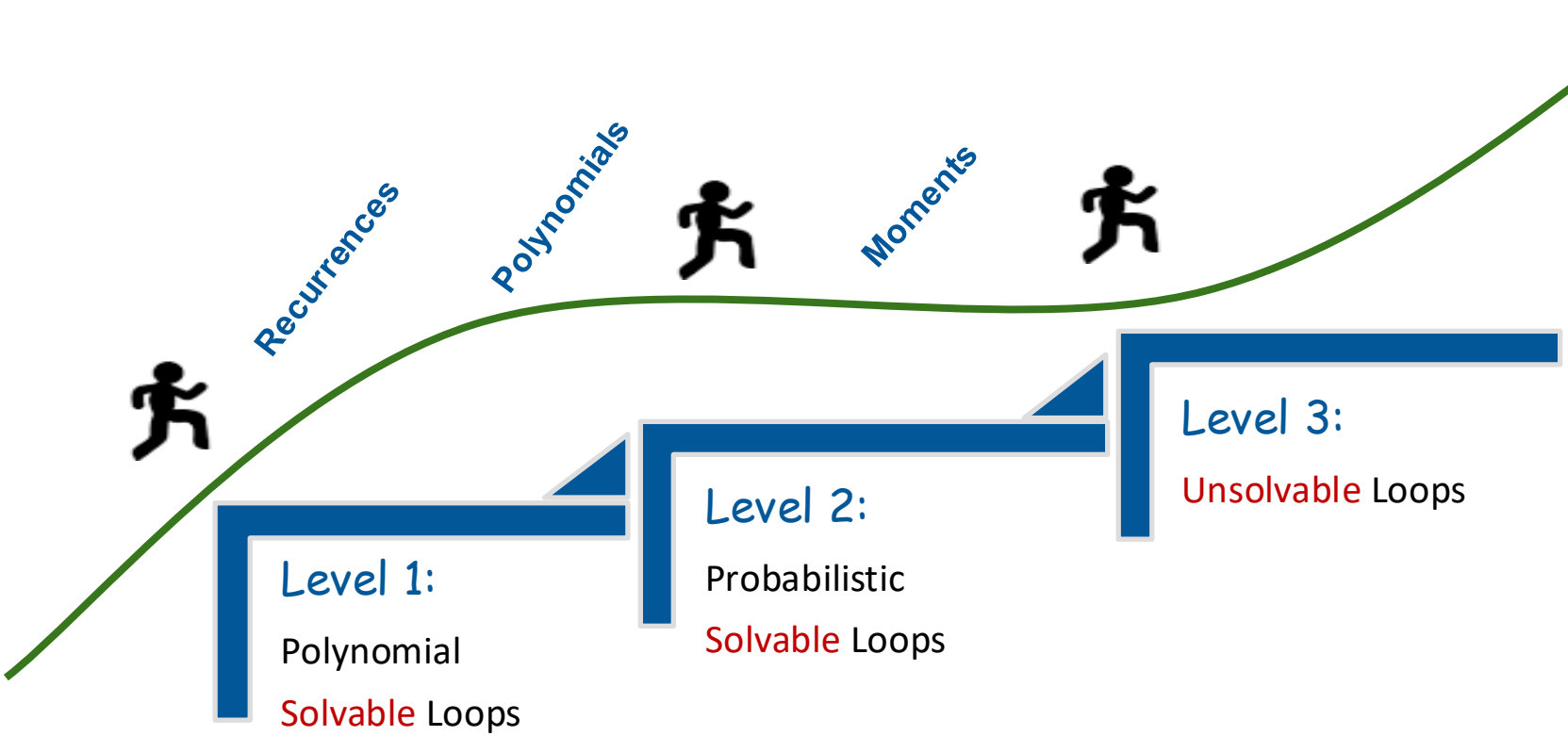
- Level 1 or Level 2 + non-C-finite recurrences
- Compute polynomial relations P over variables with non C-finite recurrences
- If P is C-finite expression, use P to solve unsolvable loops
- Automation via variable dependency analysis and polynomial constraint solving
  - POLAR tool https://github.com/probing-lab/polar
- Immediate challenge: synthesize solvabe loops from linear/polynomial invariants

Solvable Algebraic Program Analysis

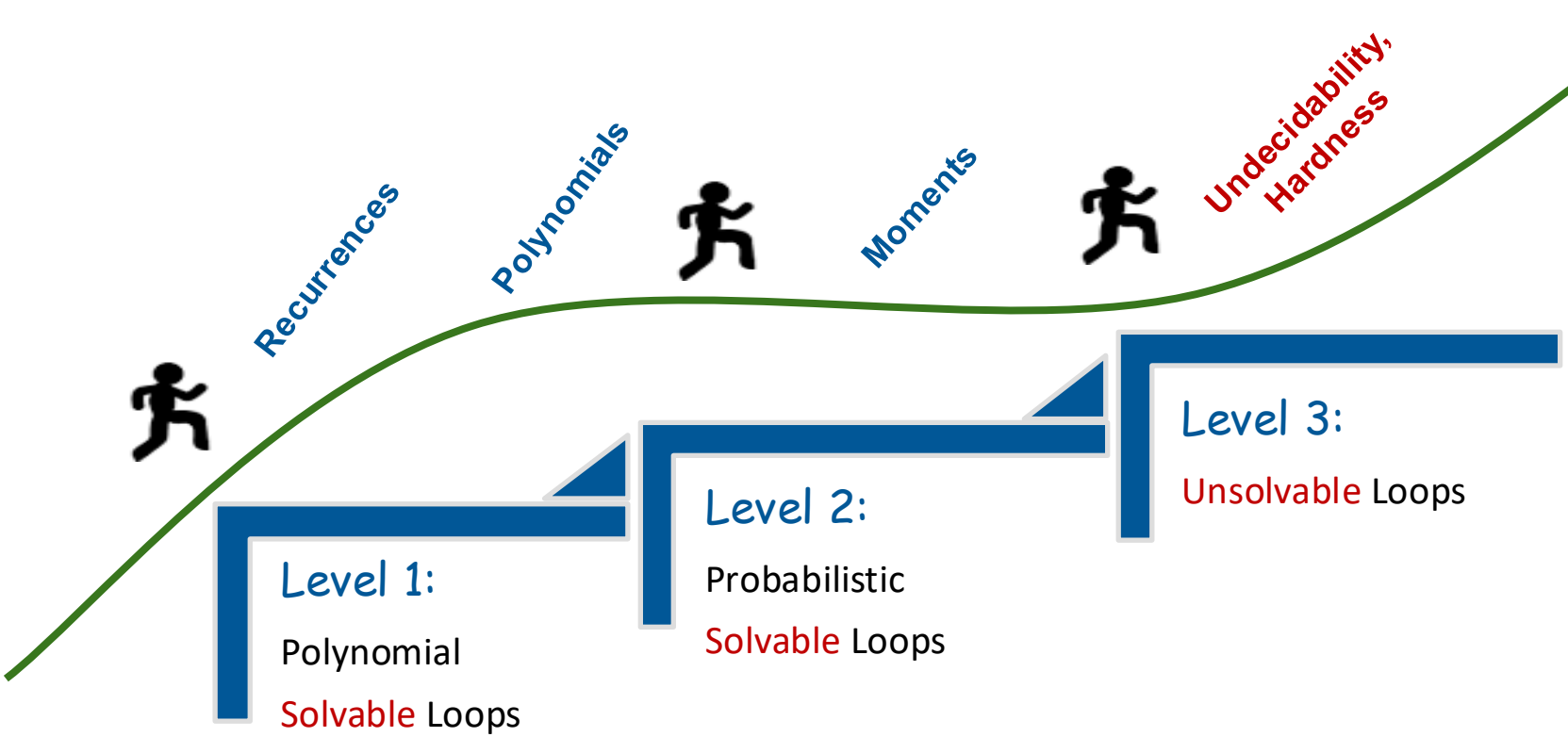


erc
FWF
Der Wissenschaftsfonds.
WWTF
VIENNA SCIENCE AND TECHNOLOGY FUND
amazon web services™

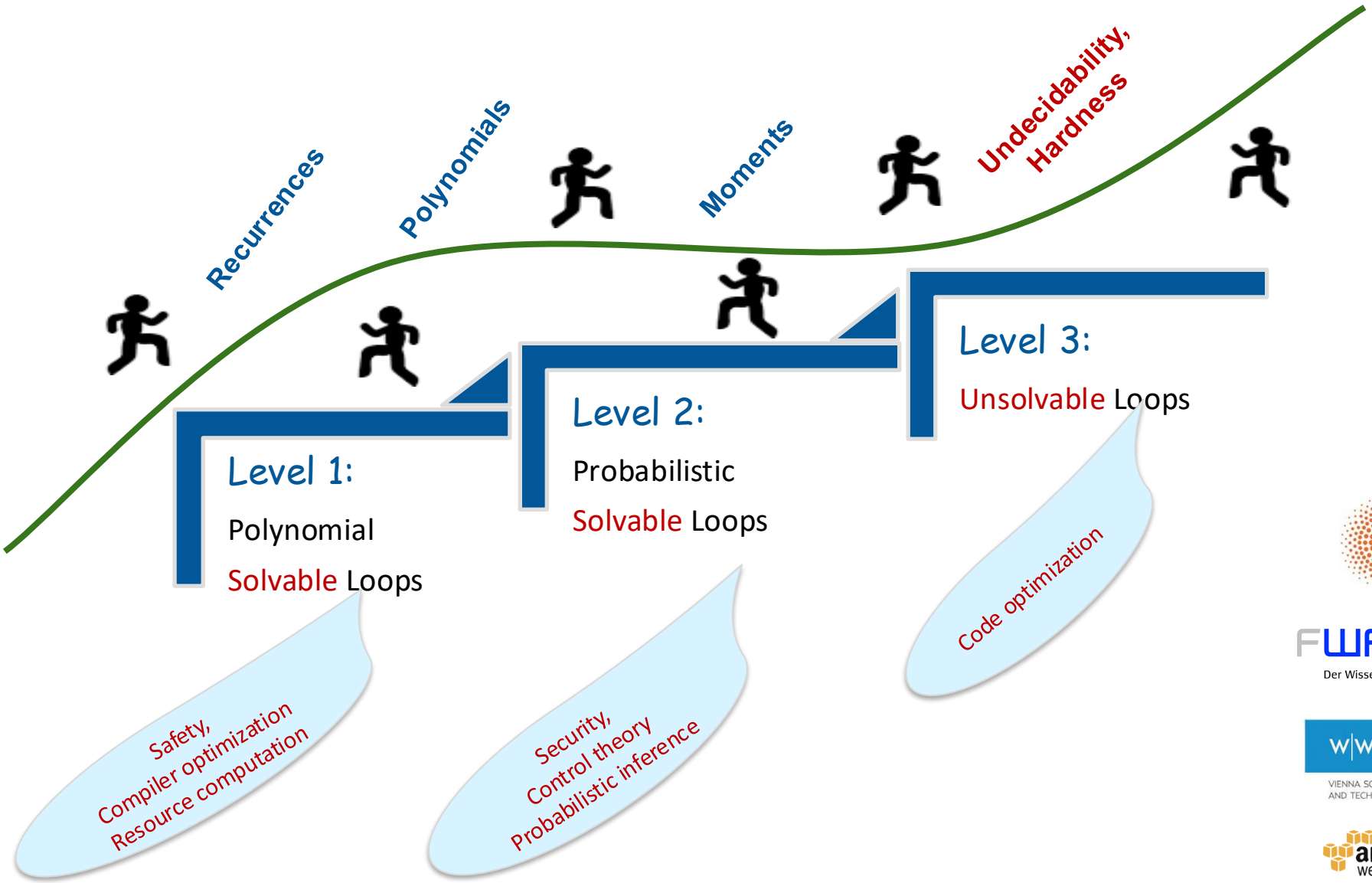
Solvable Algebraic Program Analysis



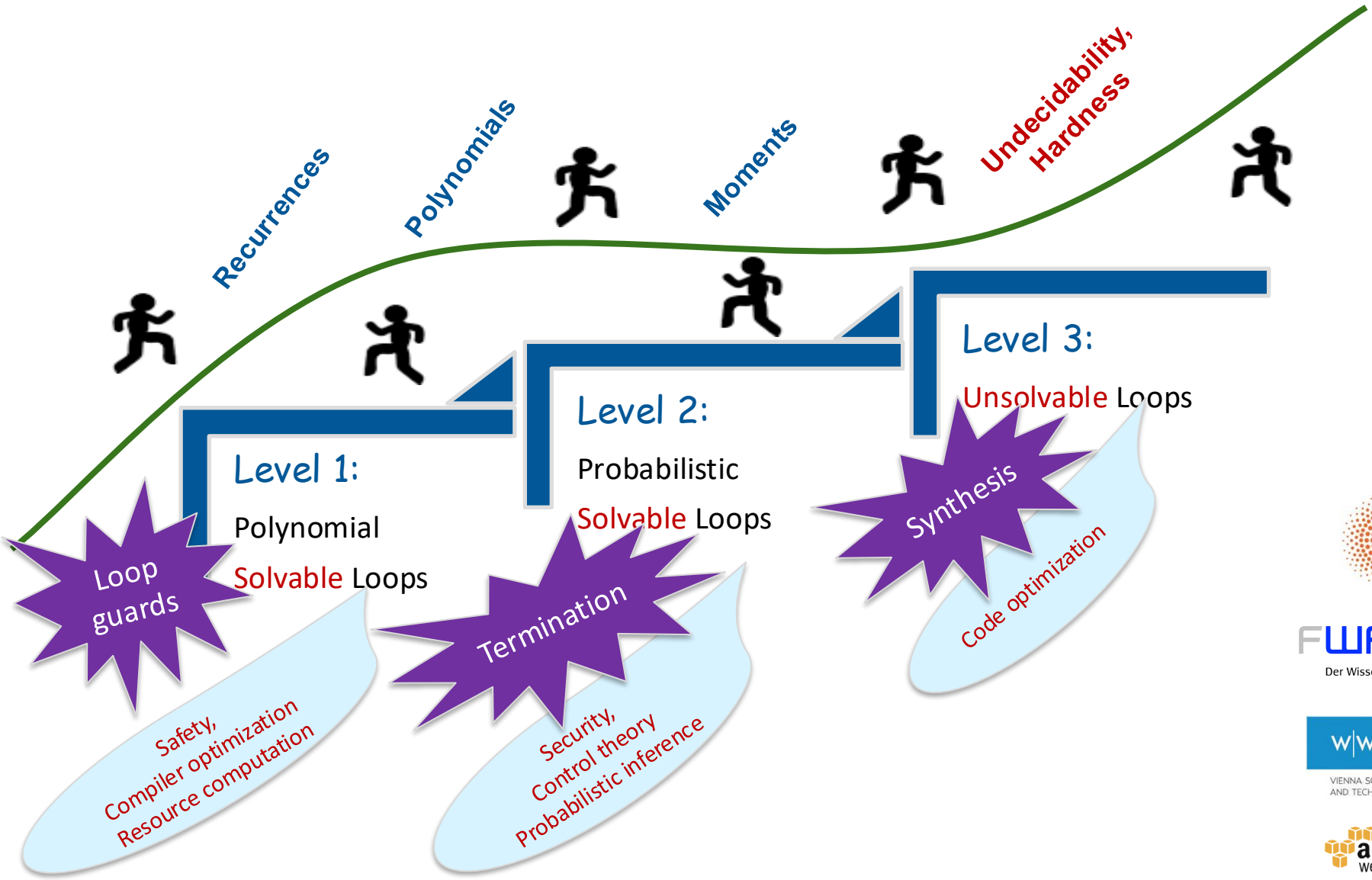
Solvable Algebraic Program Analysis



Solvable Algebraic Program Analysis



Solvable Algebraic Program Analysis



erc
FWF
Der Wissenschaftsfonds.
WWTF
VIENNA SCIENCE AND TECHNOLOGY FUND
amazon web services™