# Symbolic Computation

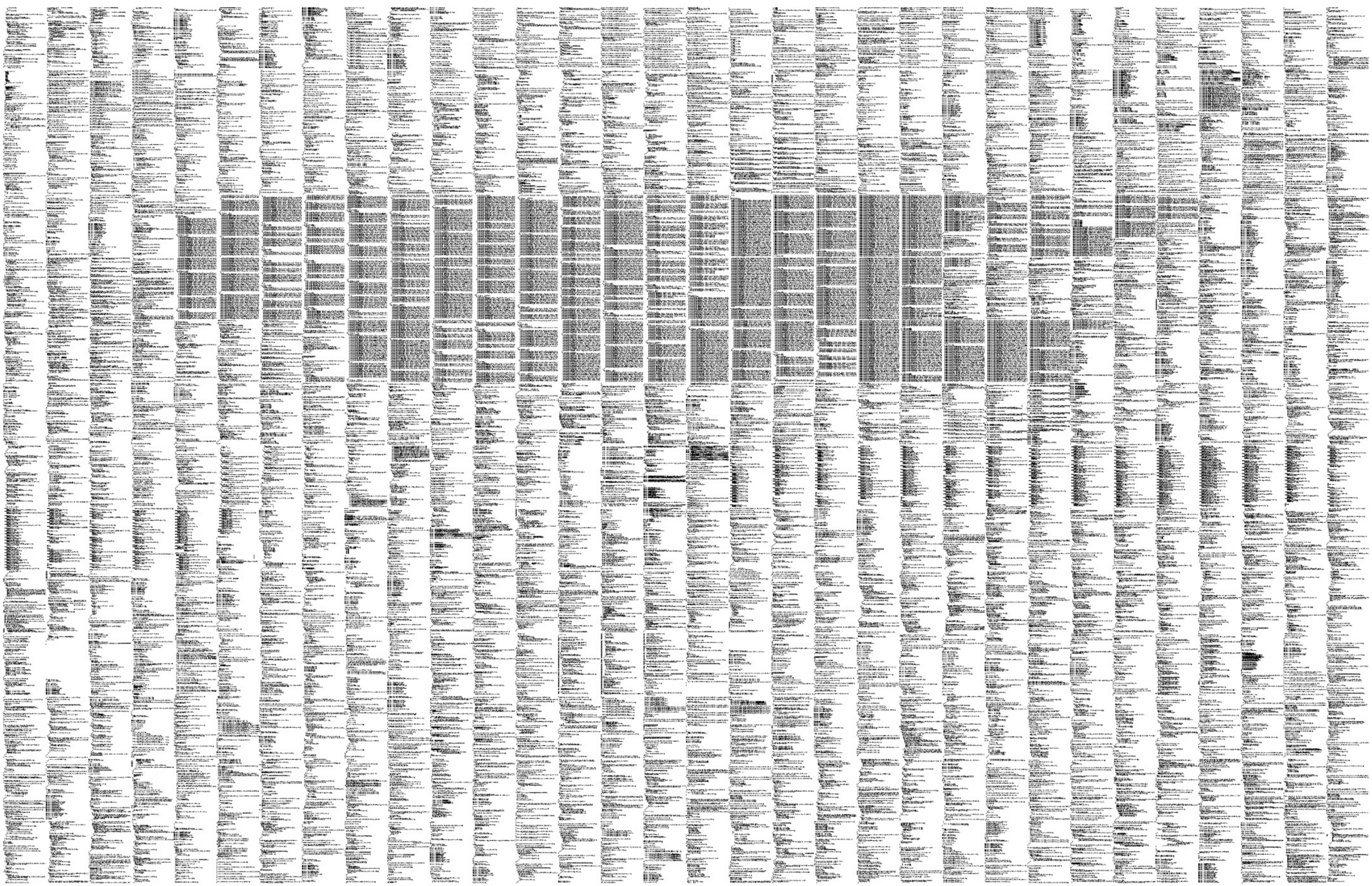
# in

# Automated Program Reasoning


**Laura Kovács**

for(syte) TU WIEN Informatics

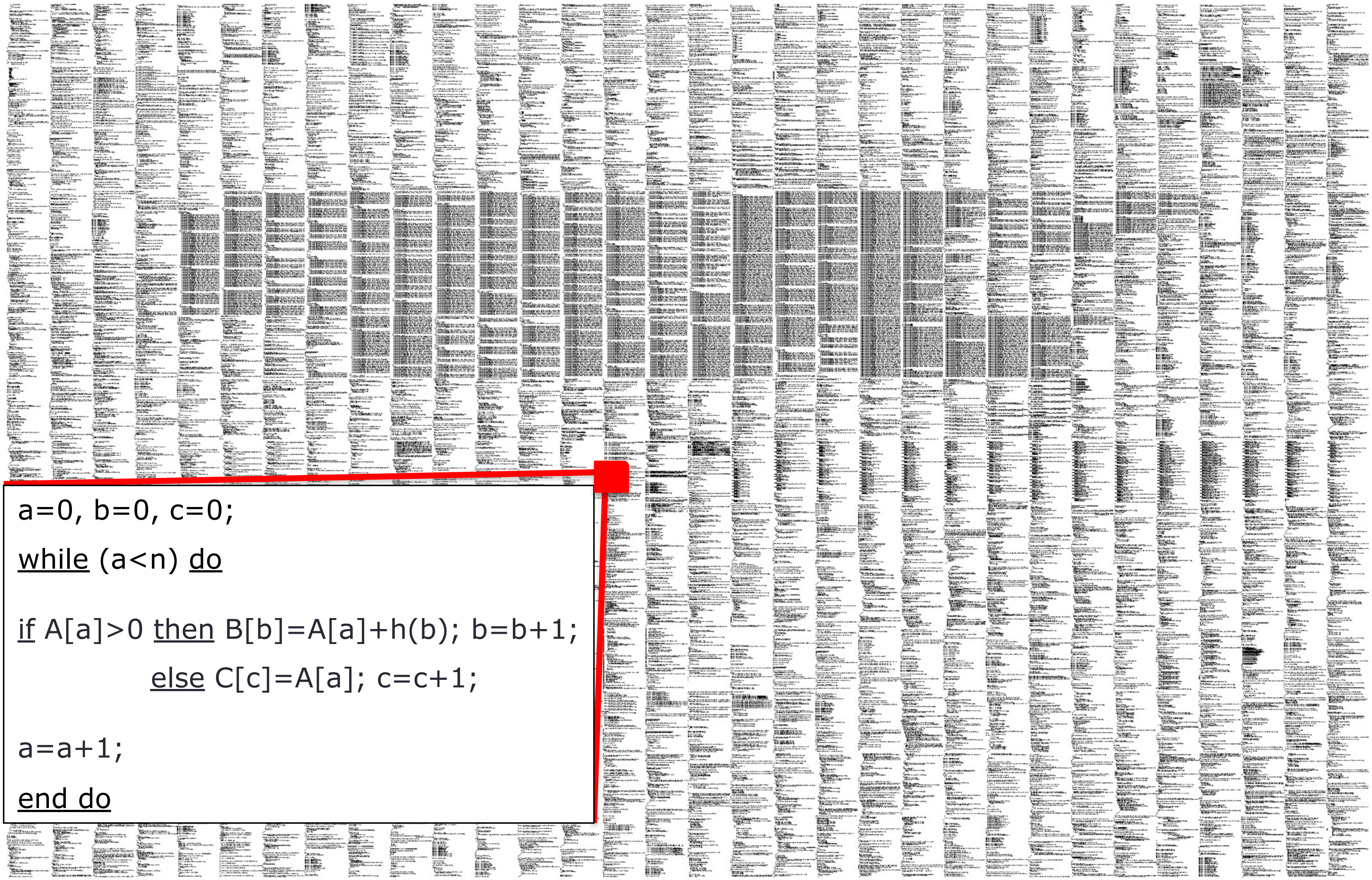
# Automating Program Reasoning



# Automating Program Reasoning

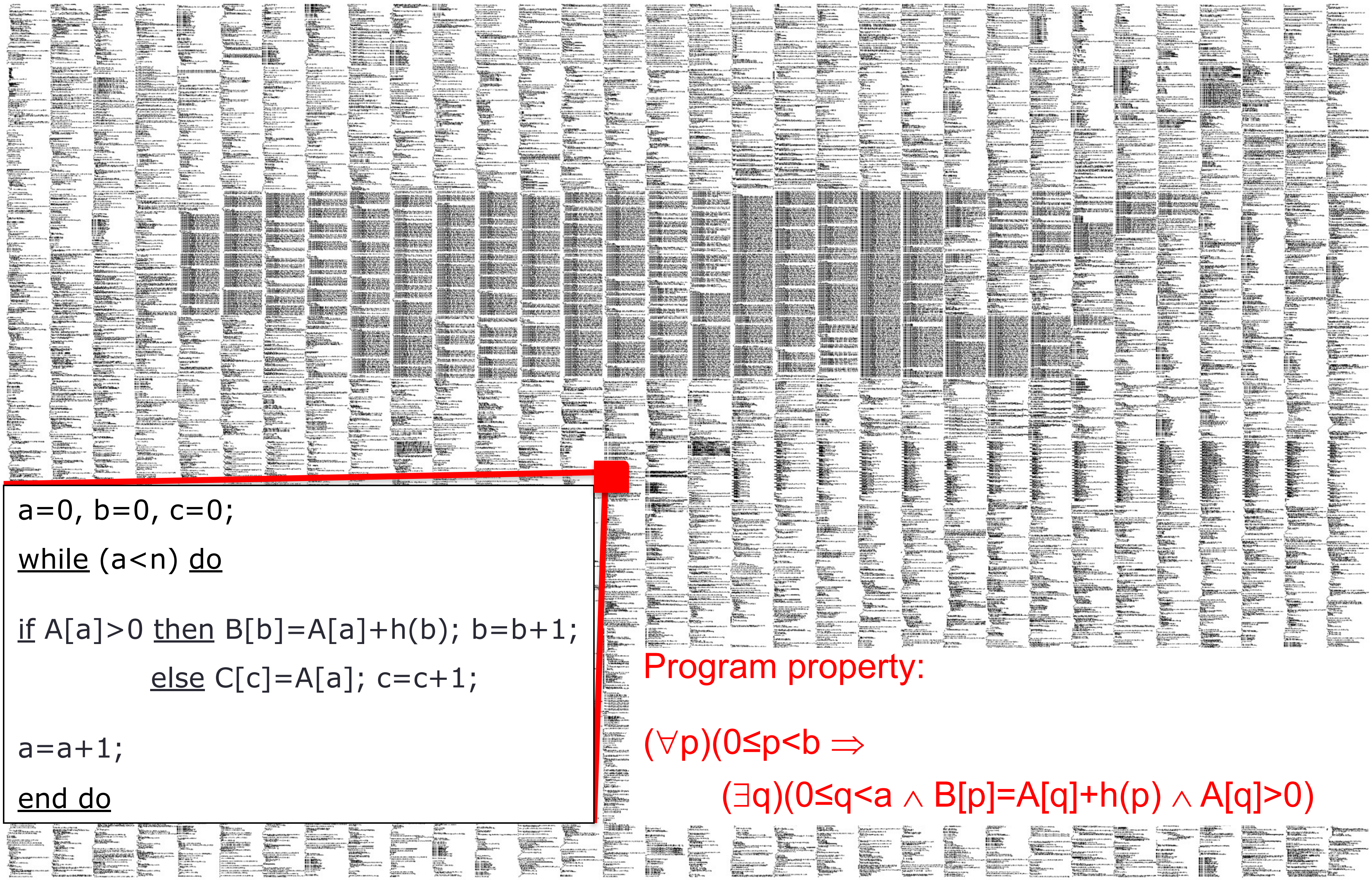
(ex. ~250kLoC, Vampire prover)

# Automating Program Reasoning



```
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
```

# Automating Program Reasoning



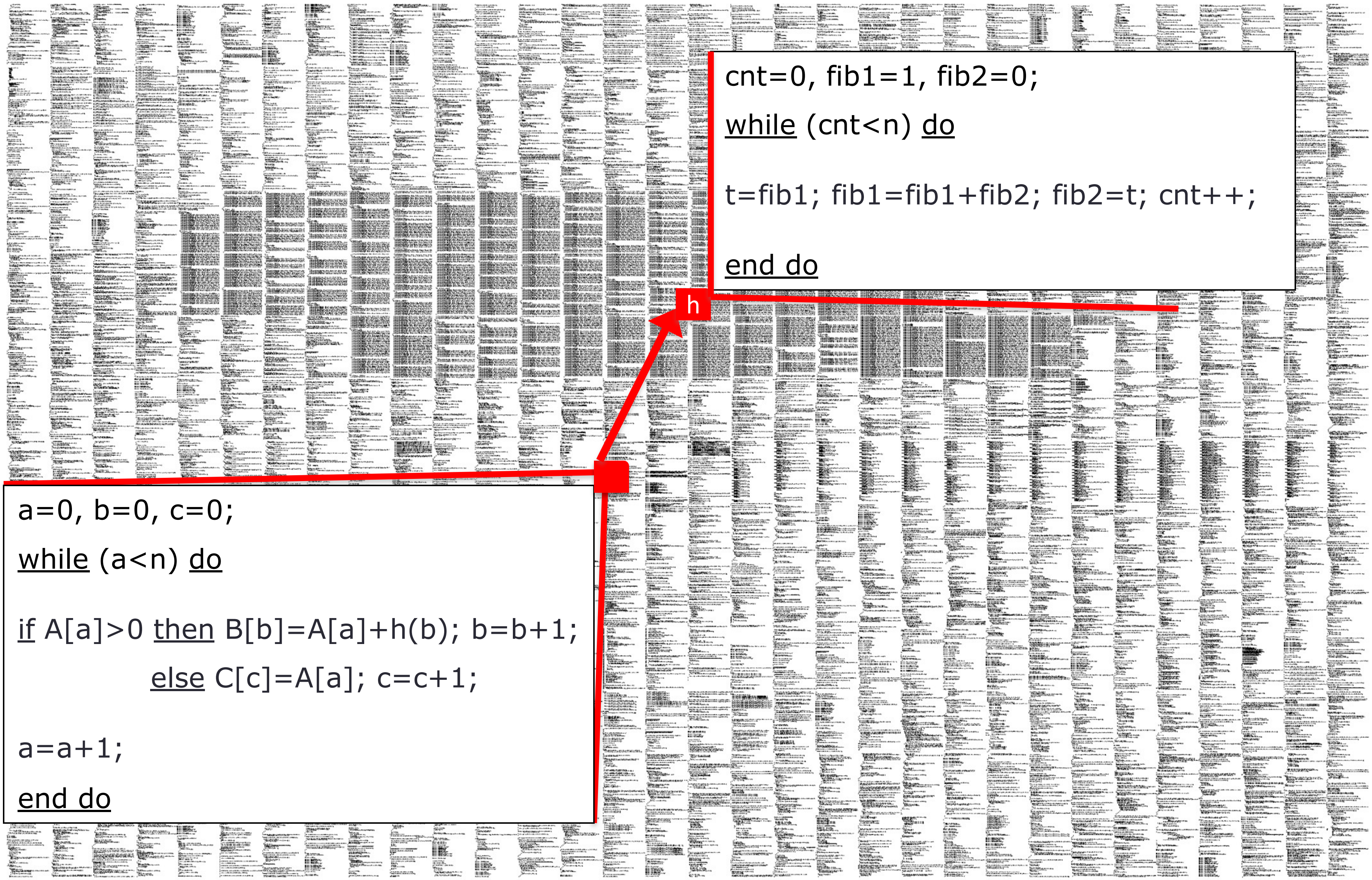
```
a=0, b=0, c=0;
while (a<n) do

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
```

Program property:

$(\forall p)(0 \leq p < b \Rightarrow$

$(\exists q)(0 \leq q < a \wedge B[p]=A[q]+h(p) \wedge A[q]>0)$

# Automating Program Reasoning




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

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

end do
```

```
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
```

# Automating Program Reasoning

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

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

end do
```

h

Program property:

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

```
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
```

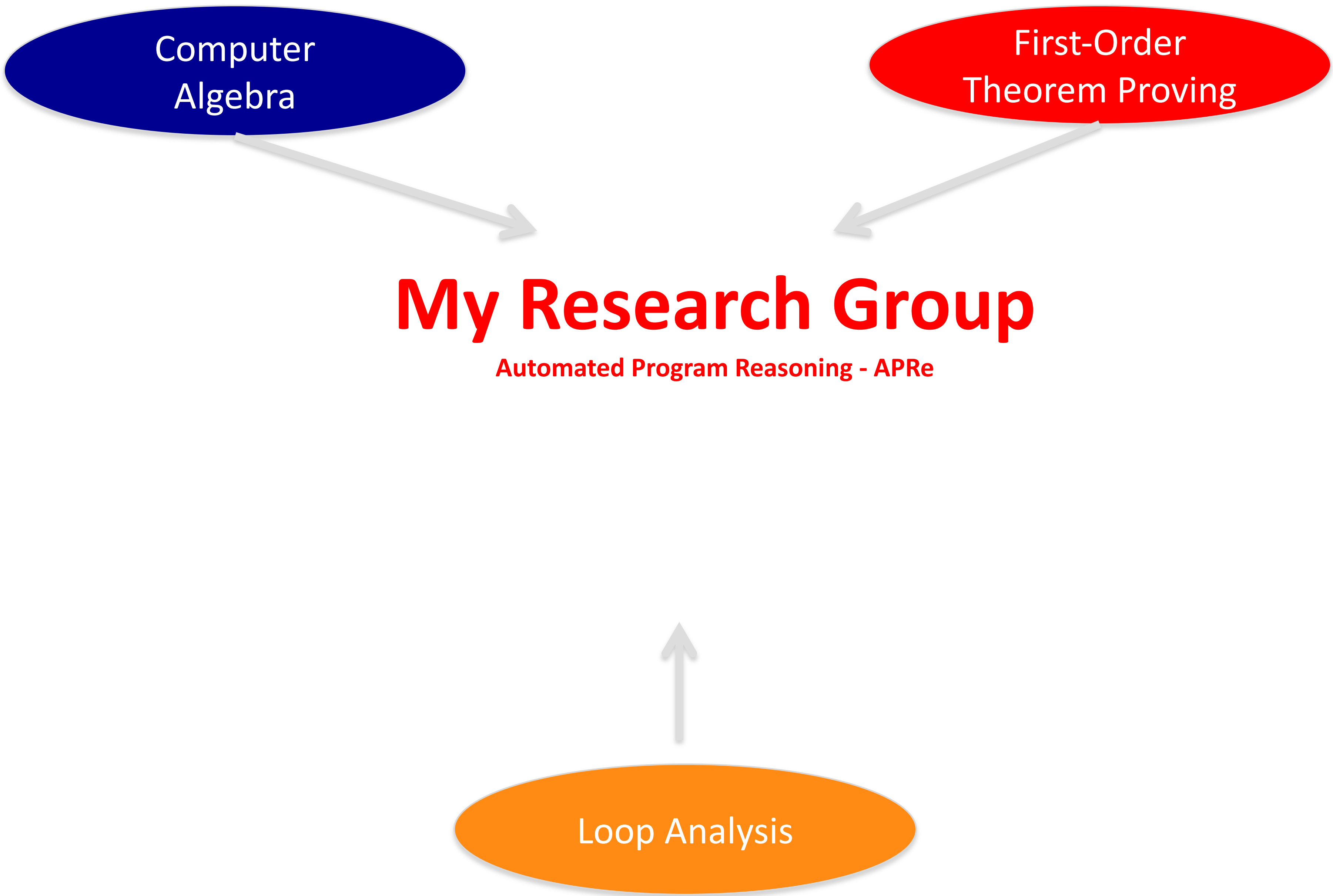
# Automating Program Reasoning

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

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

end do
```

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

```
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
```

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


Computer Algebra
First-Order Theorem Proving
My Research Group
Automated Program Reasoning - APRe
Loop 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 webservices™

Program Analysis

# Our Recipe in Automated Program Reasoning

# Our Recipe in Automated Program Reasoning

Loop

Assertions

# Our Recipe in Automated Program Reasoning

Loop

Extend language with extra symbols:

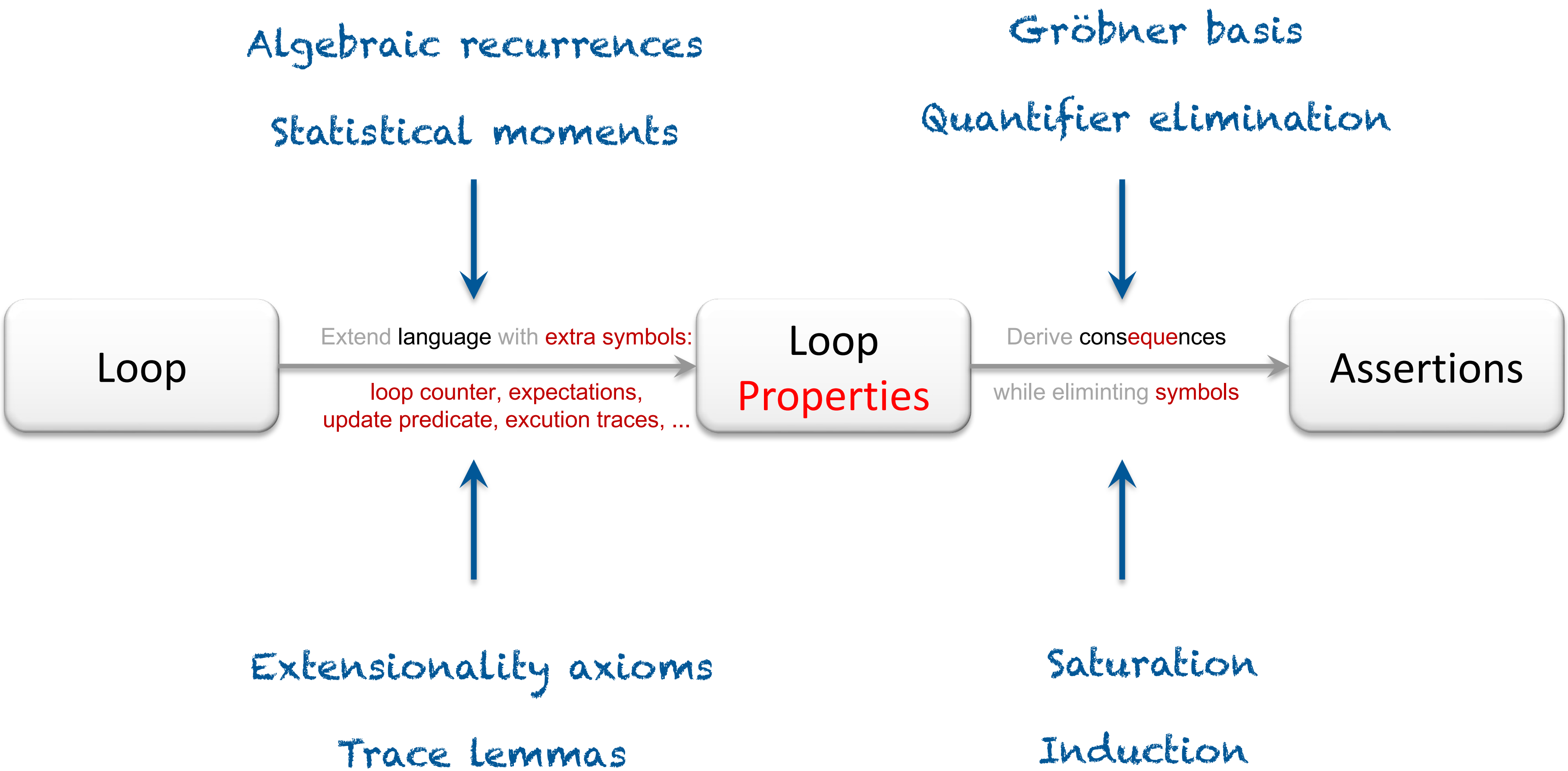
loop counter, expectations,
update predicate, excution traces, ...

Loop
Properties

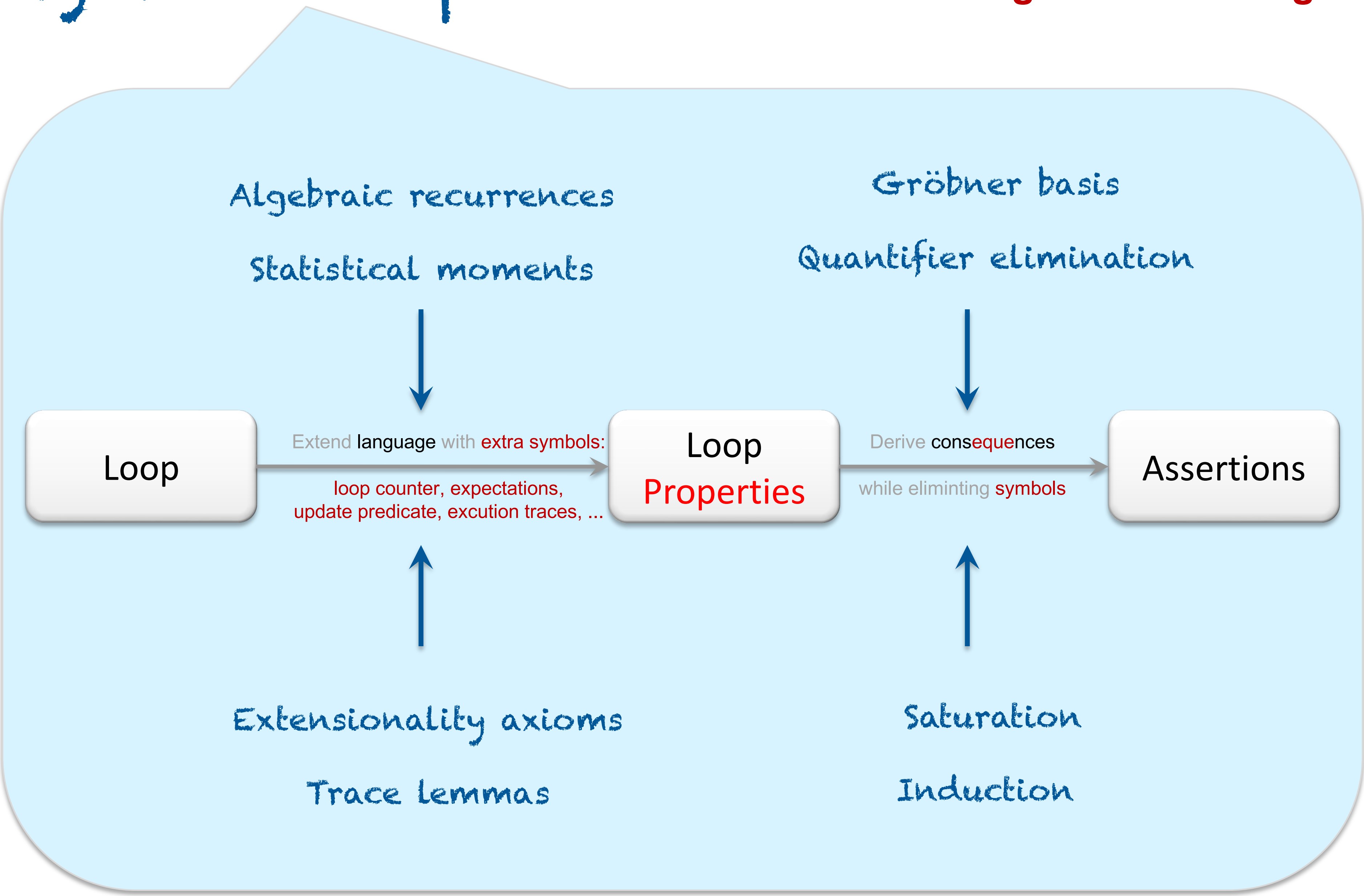
Assertions

# Our Recipe in Automated Program Reasoning

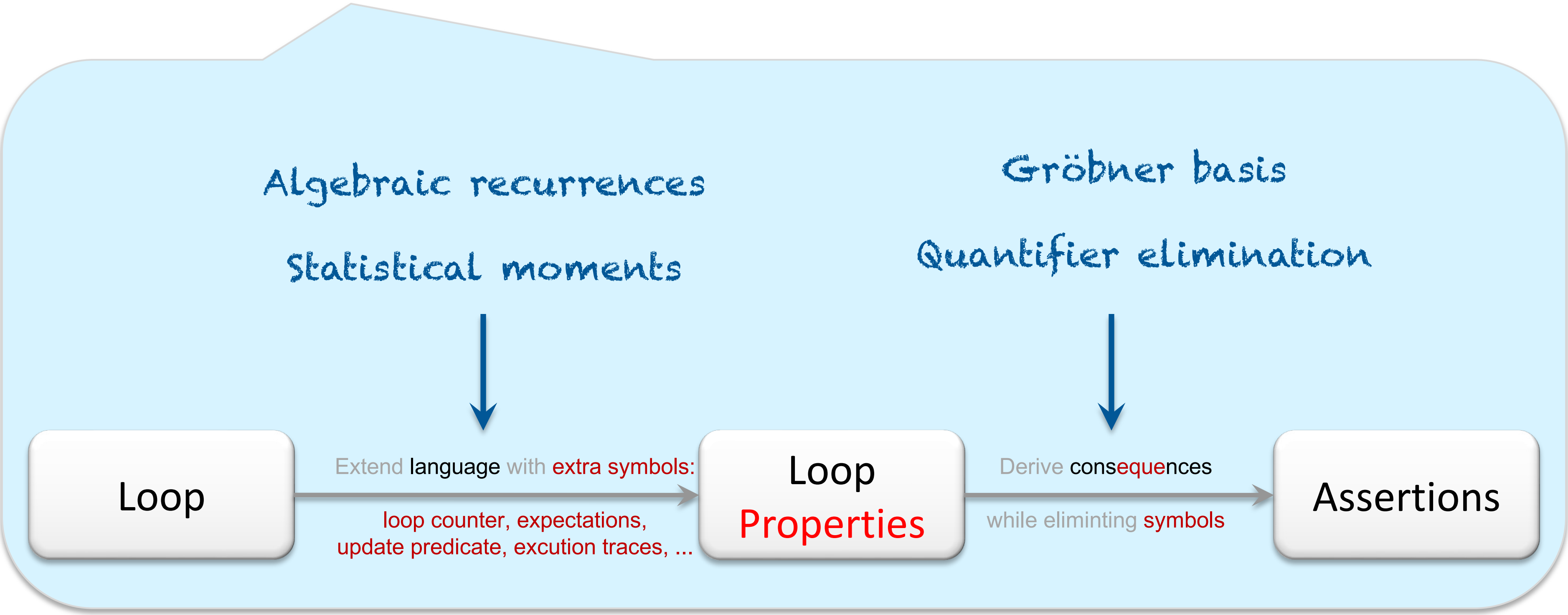
# Our Recipe in Automated Program Reasoning



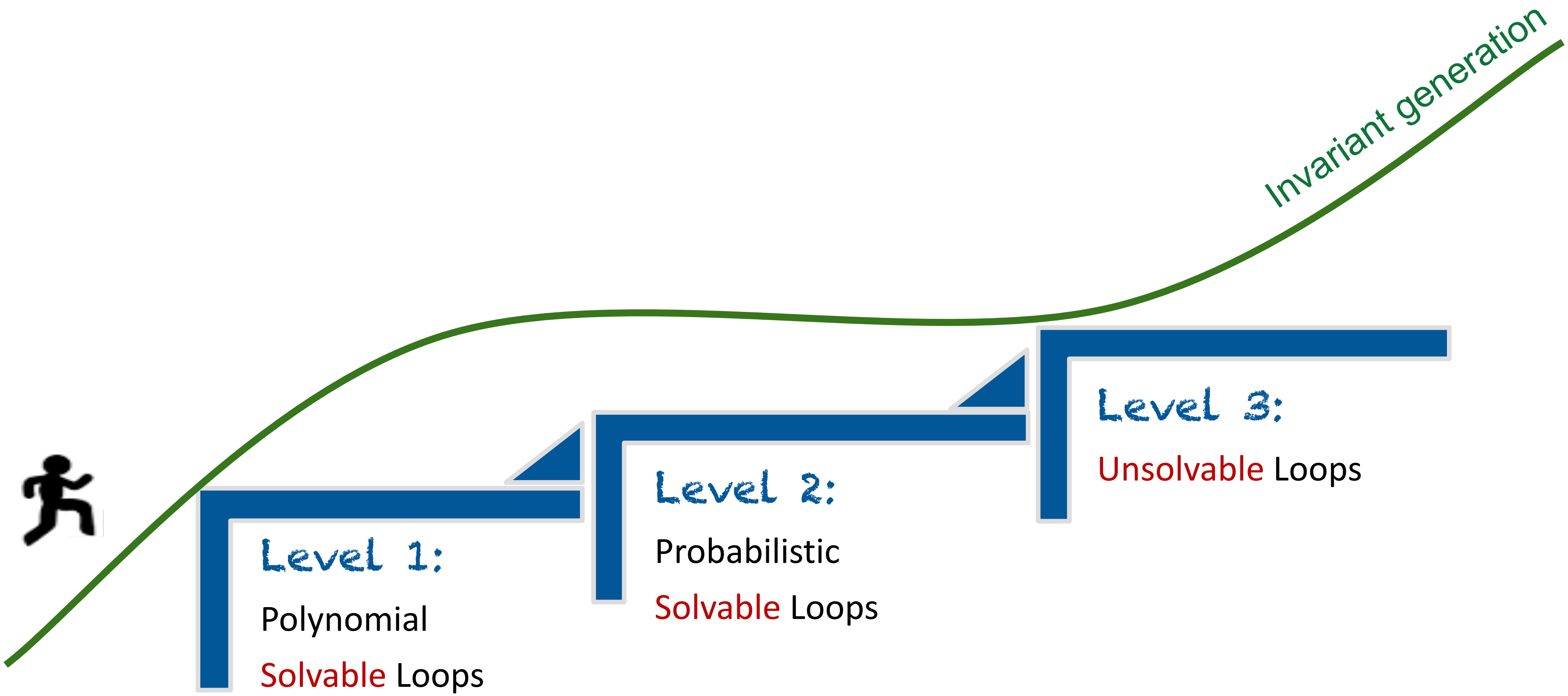
# Symbolic Computation in Automated Program Reasoning



# Symbolic Computation in Automated Program Reasoning



# Symbolic Computation in Automated Program Reasoning



# Symbolic Computation in Automated Program Reasoning

## Level 1: Polynomial Solvable Loops

x:=1; y:=0;

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

# Symbolic Computation in Automated Program Reasoning

## Level 1: Polynomial Solvable Loops

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

$n \geq 0$

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

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

# Symbolic Computation in Automated Program Reasoning

## Level 1: Polynomial Solvable Loops

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

$n \geq 0$

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

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

2. Solve recurrences → closed forms of loop variables

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

# Symbolic Computation in Automated Program Reasoning

## Level 1: Polynomial Solvable Loops

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

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

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

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

# Symbolic Computation in Automated Program Reasoning

## 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

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

4. Eliminate expressions in n ← Gröbner basis computation

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

# Symbolic Computation in Automated Program Reasoning

## Level 1: Polynomial Solvable Loops

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

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

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

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

4. Eliminate expressions in n ← Gröbner basis computation

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

→ Finite basis of polynomial invariant ideal

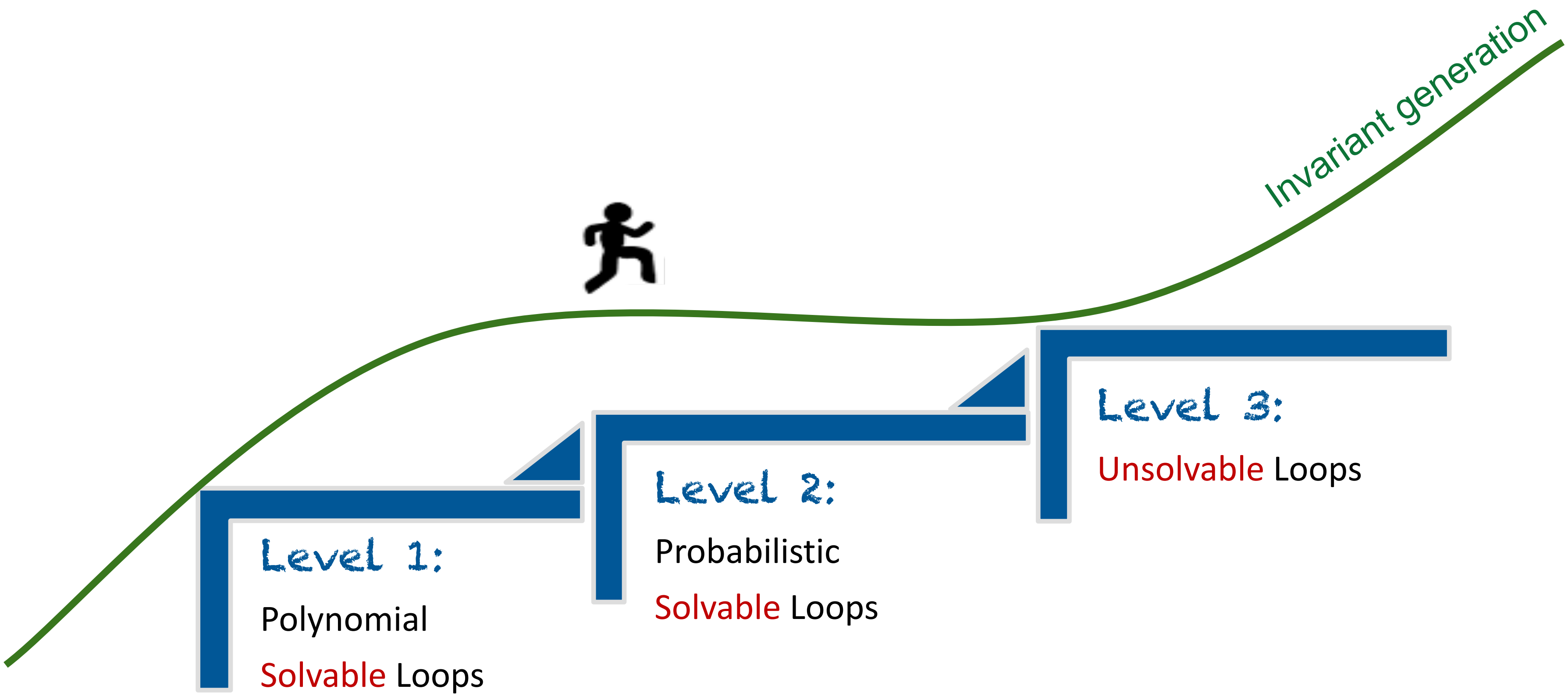
# Symbolic Computation in Automated Program Reasoning

## 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
    $\leftarrow$ C-finite recurrences of loop variables
  - Tests are ignored $\rightarrow$ non-deterministic programs
- Automation via symbolic summation and Gröbner basis computation
  - ALIGATOR tool https://ahumenberger.github.io/aligator/
- Further applications: loop termination, synthesis, deductive verification

# Symbolic Computation in Automated Program Reasoning



# Symbolic Computation in Automated Program Reasoning

## 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?

# Symbolic Computation in Automated Program Reasoning

## 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?

# Symbolic Computation in Automated Program Reasoning

## 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?

# Symbolic Computation in Automated Program Reasoning

## 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?

# Symbolic Computation in Automated Program Reasoning

## 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?

# Symbolic Computation in Automated Program Reasoning

## Level 2: Probabilistic Solvable Loops

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

- E[x]=0

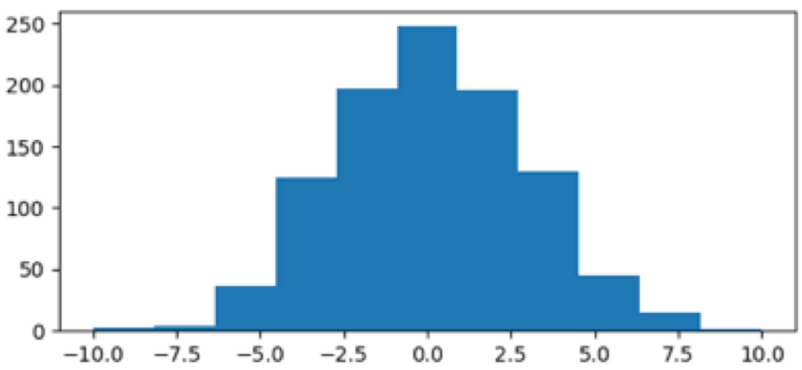
```
x:=0;
while … do
   x:=2*x-1 [1/2] 2*x+1;
end do
```

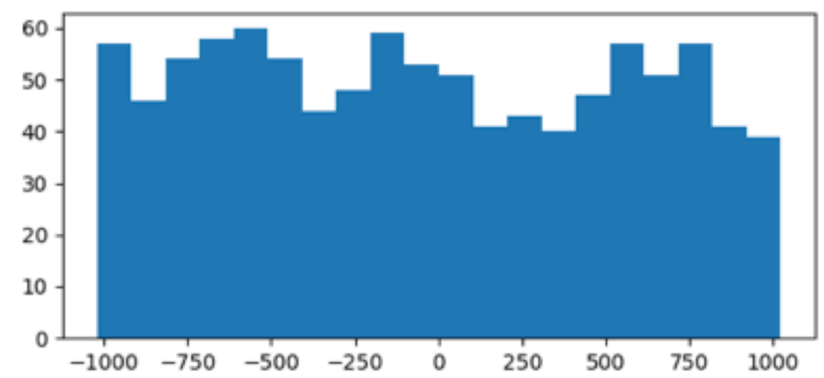
- E[x]=0

- 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!

# Symbolic Computation in Automated Program Reasoning

## 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!

# Symbolic Computation in Automated Program Reasoning

## 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?

# Symbolic Computation in Automated Program Reasoning

## 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!

# Symbolic Computation in Automated Program Reasoning

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

  Reason about higher-order statistical moments of variables!

# Symbolic Computation in Automated Program Reasoning

## 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* $k \in \mathbb{N}$.

# Symbolic Computation in Automated Program Reasoning

## 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 $k \in \mathbb{N}$.*

**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.

# Symbolic Computation in Automated Program Reasoning

## 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;
}
```

$$\boxed{E[s_n] = \frac{9}{4}n}$$

$$\boxed{\mathrm{Var}[s_n] = \frac{20d^2 + 27}{16}n}$$

**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;
}
```

$$\boxed{E[s_n] = \frac{9}{4}n}$$

$$\boxed{\mathrm{Var}[s_n] = \frac{347}{16}n + \frac{128}{3}}$$

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

→ **Higher-order moments of loop variables are computable.**

# Symbolic Computation in Automated Program Reasoning

## 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`

# Symbolic Computation in Automated Program Reasoning

## 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
```

- Expected values of monomials
  - sufficient to understand E-variables

  $E[X], E[X^2], E[XY], \ldots$

Probabilistic updates:

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

# Symbolic Computation in Automated Program Reasoning

## 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
```

- Expected values of monomials
  - sufficient to understand E-variables

  $E[X], E[X^2], E[XY], \ldots$

Probabilistic updates:

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

Stochastic recurrences over E-variables:

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

# Symbolic Computation in Automated Program Reasoning

## 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
```

- Expected values of monomials
  - sufficient to understand E-variables

  $E[X], E[X^2], E[XY], \ldots$

- Computing with E-variables

  $E[c] = c$

  $E[X + cY] = E[X] + cE[Y]$

  $E[X \cdot Y] \neq E[X] \cdot E[Y]$ unless X, Y are independent

Stochastic recurrences over E-variables:

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

# Symbolic Computation in Automated Program Reasoning

## 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
```

- Expected values of monomials
  - necessary to understand E-variables

  $E[X], E[X^2], E[XY], \dots$

- Computing with E-variables

  $E[c] = c$

  $E[X + cY] = E[X] + cE[Y]$

  $E[X \cdot Y] \neq E[X] \cdot E[Y]$ unless X, Y are independent

  Expected values of monomials cannot be simplified

Stochastic recurrences over E-variables:

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

# Symbolic Computation in Automated Program Reasoning

## Level 2: Probabilistic Solvable Loops

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

- x(n+1) = 2*x(n) Level 1

- x(n+1) = x(n) + unif(0,1)

# Symbolic Computation in Automated Program Reasoning

## Level 2: Probabilistic Solvable Loops

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

- x(n+1) = 2*x(n) Level 1
- x(n+1) = x(n) + unif(0,1)

# Symbolic Computation in Automated Program Reasoning

## Level 2: Probabilistic Solvable Loops

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

- x(n+1) = 2*x(n)  Level 1
- x(n+1) = x(n) + unif(0,1) 

— use **moments**: treat each moment as a separate **E-variable**

- $E[x(n+1)] = E[x(n)] + 1/2$
  $E[x^2(n+1)] = E[x^2(n)] + \ldots$
  $E[x^3(n+1)] = \ldots$
  …

# Symbolic Computation in Automated Program Reasoning

## Level 2: Probabilistic Solvable Loops

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

- $x(n+1) = 2*x(n)$  Level 1

- $x(n+1) = x(n) + unif(0,1)$ 

— use **moments**: treat each moment as a separate **E-variable**

- $E[x(n+1)] = E[x(n)] + 1/2$
  $E[x^2(n+1)] = E[x^2(n)] + \ldots$
  $E[x^3(n+1)] = \ldots$
  $\ldots$

— solve **stochastic recurrences**: closed forms over **E-variables**

# Symbolic Computation in Automated Program Reasoning

## 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}
```

# Symbolic Computation in Automated Program Reasoning

## 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
7. Compute moment-based invariants

```
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}
```

# Symbolic Computation in Automated Program Reasoning

## 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
7. Compute moment-based invariants

```
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}
```

$$\{s^2\}$$

# Symbolic Computation in Automated Program Reasoning

## 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
7. Compute moment-based invariants

```
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}
```

$$\{s^2\} \to s^2$$

# Symbolic Computation in Automated Program Reasoning

## 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
7. Compute moment-based invariants

```
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}
```

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

# Symbolic Computation in Automated Program Reasoning

## 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
7. Compute moment-based invariants

```
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}
```

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

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

# Symbolic Computation in Automated Program Reasoning

## 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
7. Compute moment-based invariants

```
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}
```

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

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

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

# Symbolic Computation in Automated Program Reasoning

## 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
7. Compute moment-based invariants

```
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}
```

$\rightarrow x^2$

# Symbolic Computation in Automated Program Reasoning

## 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
7. Compute moment-based invariants

```
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}
```

$$\to x^2 \to E[x^2(n+1)] = E[(x(n) + f(n+1) \cdot rand(1-d, 1+d))^2]$$

# Symbolic Computation in Automated Program Reasoning

## 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
7. Compute moment-based invariants

```
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}
```

$$\rightarrow \; x^2 \; \rightarrow \; E[x^2(n+1)] = E[(x(n) + f(n+1) \cdot rand(1-d, 1+d))^2]$$

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

# Symbolic Computation in Automated Program Reasoning

## 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
7. Compute moment-based invariants

```
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}
```

$$\rightarrow x^2 \rightarrow E[x^2(n+1)] = E[(x(n) + f(n+1) \cdot rand(1-d, 1+d))^2]$$

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

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

# Symbolic Computation in Automated Program Reasoning

## 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
7. Compute moment-based invariants

```
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}
```

$$\rightarrow x^2 \rightarrow E[x^2(n+1)] = E[(x(n) + f(n+1) \cdot rand(1-d, 1+d))^2]$$

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

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

**...and so on ...**

# Symbolic Computation in Automated Program Reasoning

## Level 2: Probabilistic Solvable Loops

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

$$E[f(n+1)] = \frac{3}{4}$$

$$E[x(n+1)] = E[x(n)] + \frac{3}{4}E[f(n+1)]$$

$$E[y(n+1)] = E[y(n)] + \frac{3}{2}E[f(n+1)]$$

$$E[s(n+1)] = E[x(n+1)] + E[y(n+1)]$$

$$\ldots$$

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

# Symbolic Computation in Automated Program Reasoning

## Level 2: Probabilistic Solvable Loops

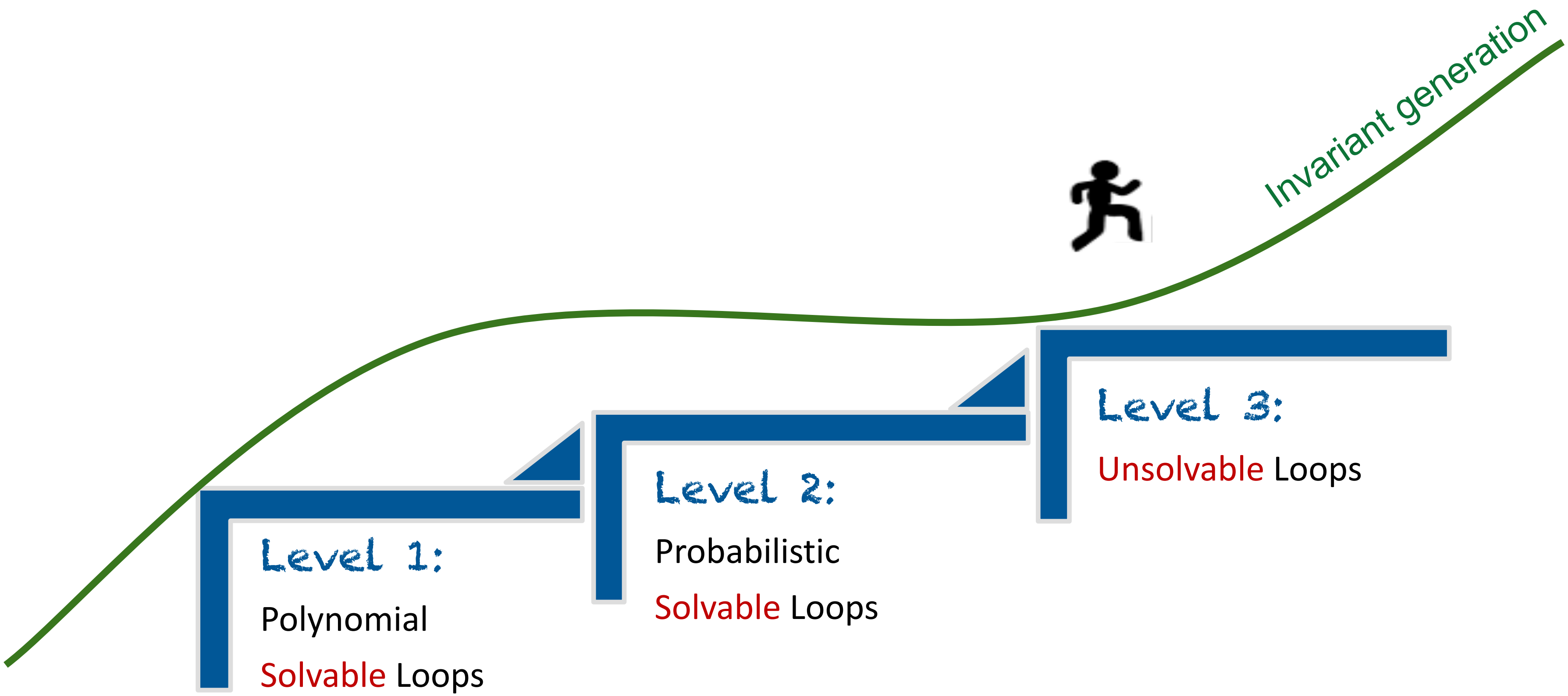
joint work w A. Humenberger, M. Jaroschek, A. Varonka (ISSAC17,VMCAI18,RAMICS18)

- Level 1 + Random polynomial assignments + finite-valued conditions
  - C-finite recurrences of E-variables
  - Tests are finite-valued

- Automation via symbolic summation and moment-based computation
  - POLAR tool https://github.com/probing-lab/polar

- Further applications: probabilistic termination, sensitivity, probabilistic inferences

# Symbolic Computation in Automated Program Reasoning



# Symbolic Computation in Automated Program Reasoning

## 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
```

$y := \frac{1}{2} * y + 1$

# Symbolic Computation in Automated Program Reasoning

## 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 $\rightarrow$ non-C-finite recurrences

Level 1 and Level 2 ⊗

# Symbolic Computation in Automated Program Reasoning

## Level 3: Unsolvable Loops

```
a:=-2; b:=3; y:=0;
while ... do
   a:=2*a+b²;
   b:=2*b-b²;
   y:=1/2 *y+1
end do
```

$n \geq 0$

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

- Yet, $x(n)=a(n)+b(n)$ satisfy a C-finite recurrence: $a(n+1)+b(n+1)=2*(a(n)+b(n))$

# Symbolic Computation in Automated Program Reasoning

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
```

$n \geq 0$

```
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 ✓

# Symbolic Computation in Automated Program Reasoning

## Level 3: Unsolvable Loops → Level 1: Solvable Loops

```
a:=-2; b:=3; y:=0;
while ... do
   a:=2*a+b²;
   b:=2*b-b²;
   y:=1/2 *y+1
end do
```

$n \geq 0$

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

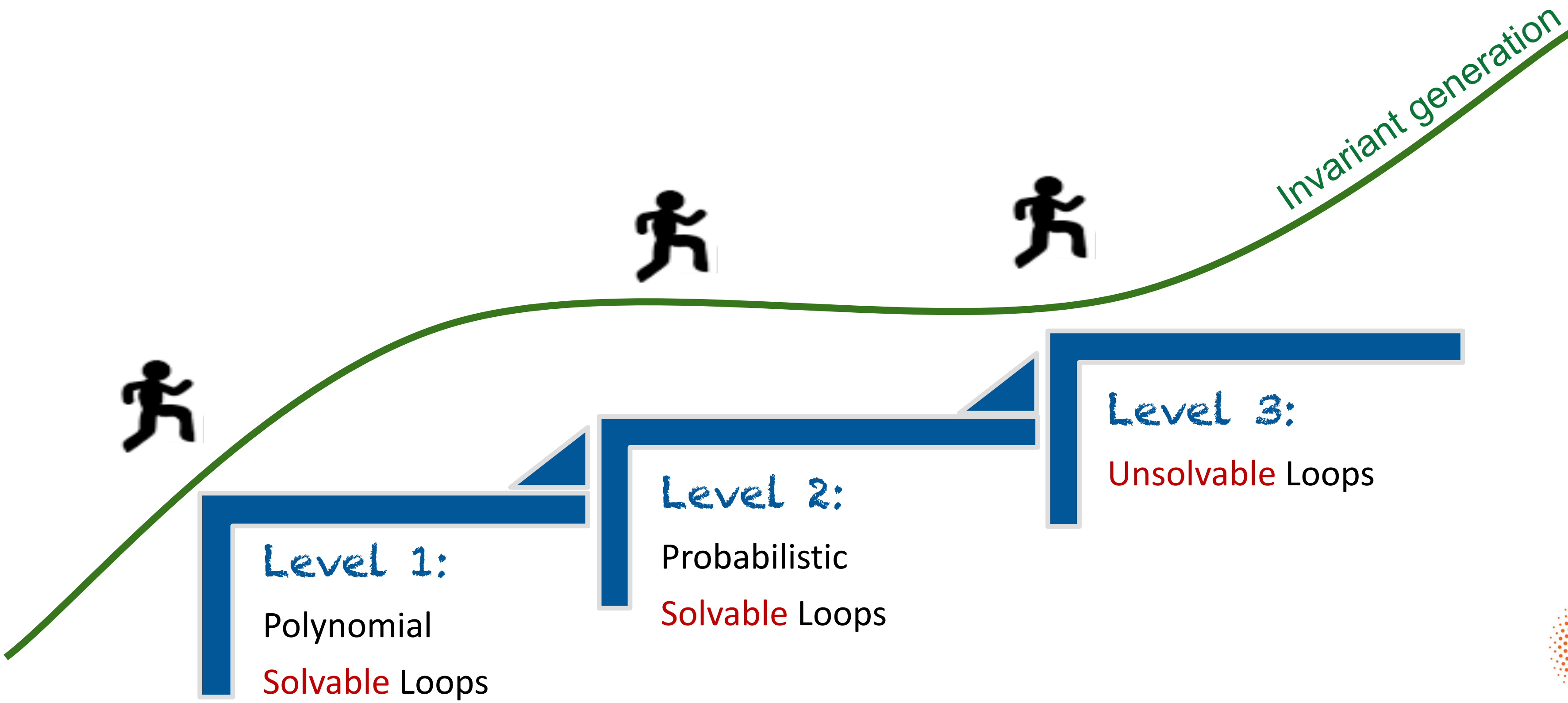
# Symbolic Computation in Automated Program Reasoning

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

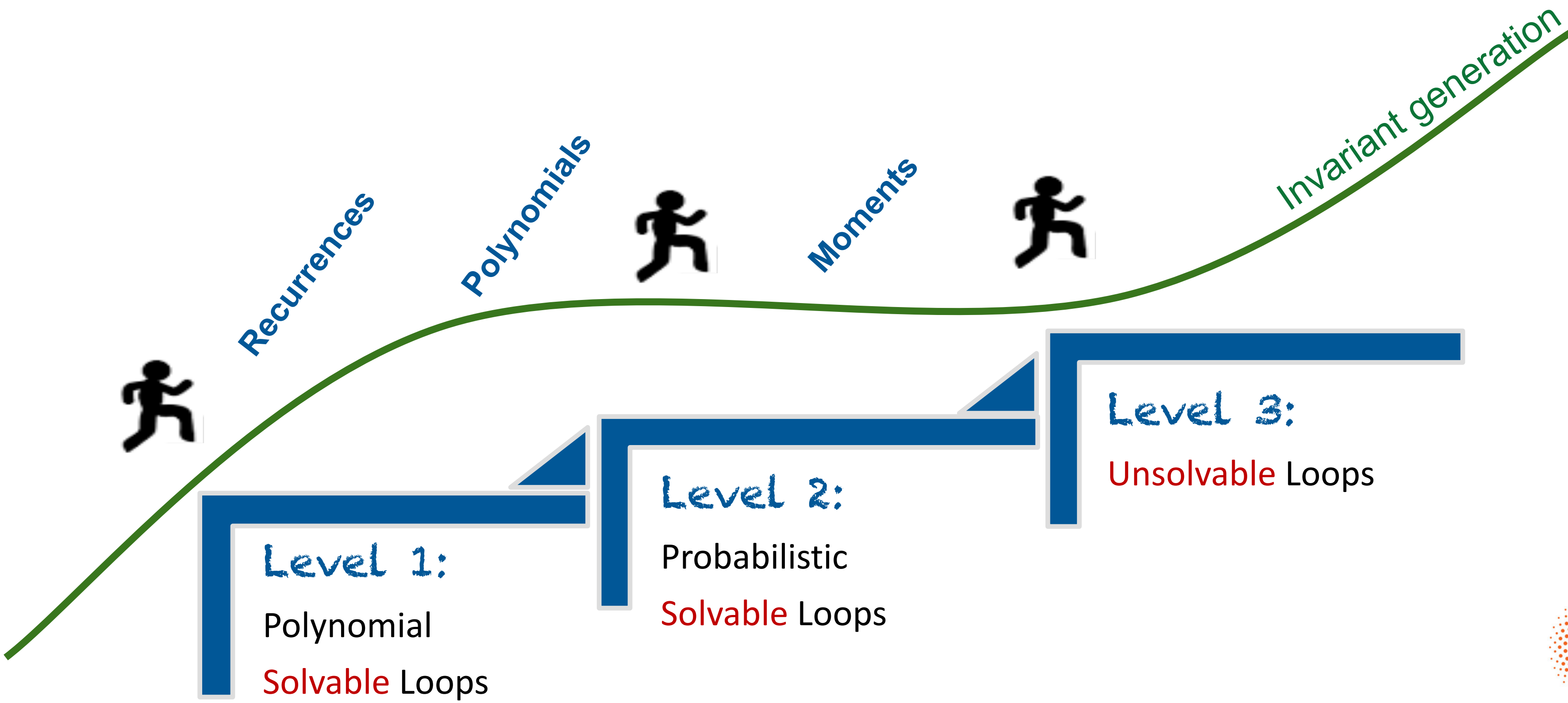
joint work w D. Amrollahi, E. Bartocci, G. Kenison, M. Stankovic, M. Moosbrugger (SAS22)

- 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

# Symbolic Computation in Automated Program Reasoning



# Symbolic Computation in Automated Program Reasoning



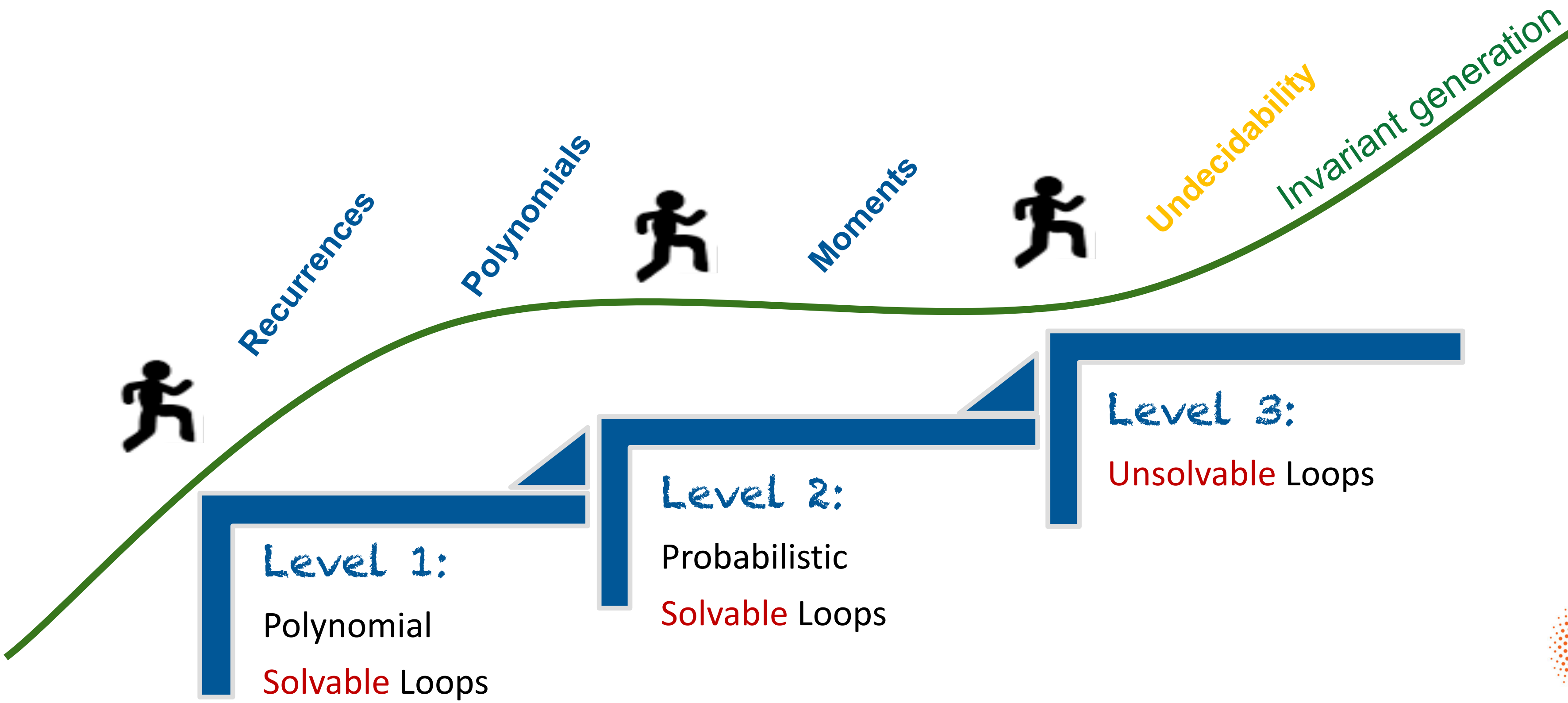

erc

FWF
Der Wissenschaftsfonds.

W|W|T|F
VIENNA SCIENCE
AND TECHNOLOGY FUND

amazon
webservices™

# Symbolic Computation in Automated Program Reasoning



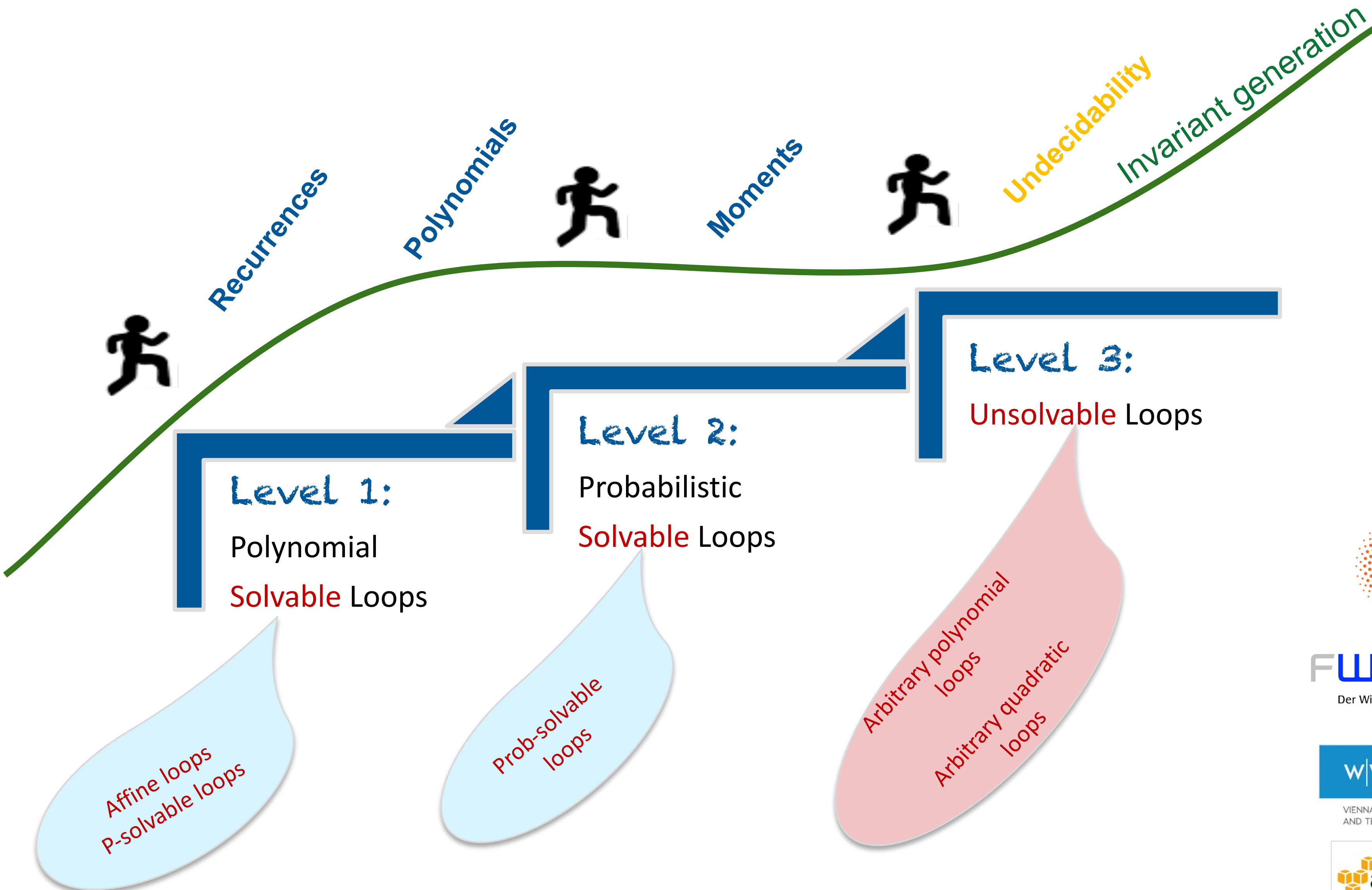

erc

FWF
Der Wissenschaftsfonds.

W|W|T|F
VIENNA SCIENCE
AND TECHNOLOGY FUND

amazon
web services™

# Symbolic Computation in Automated Program Reasoning



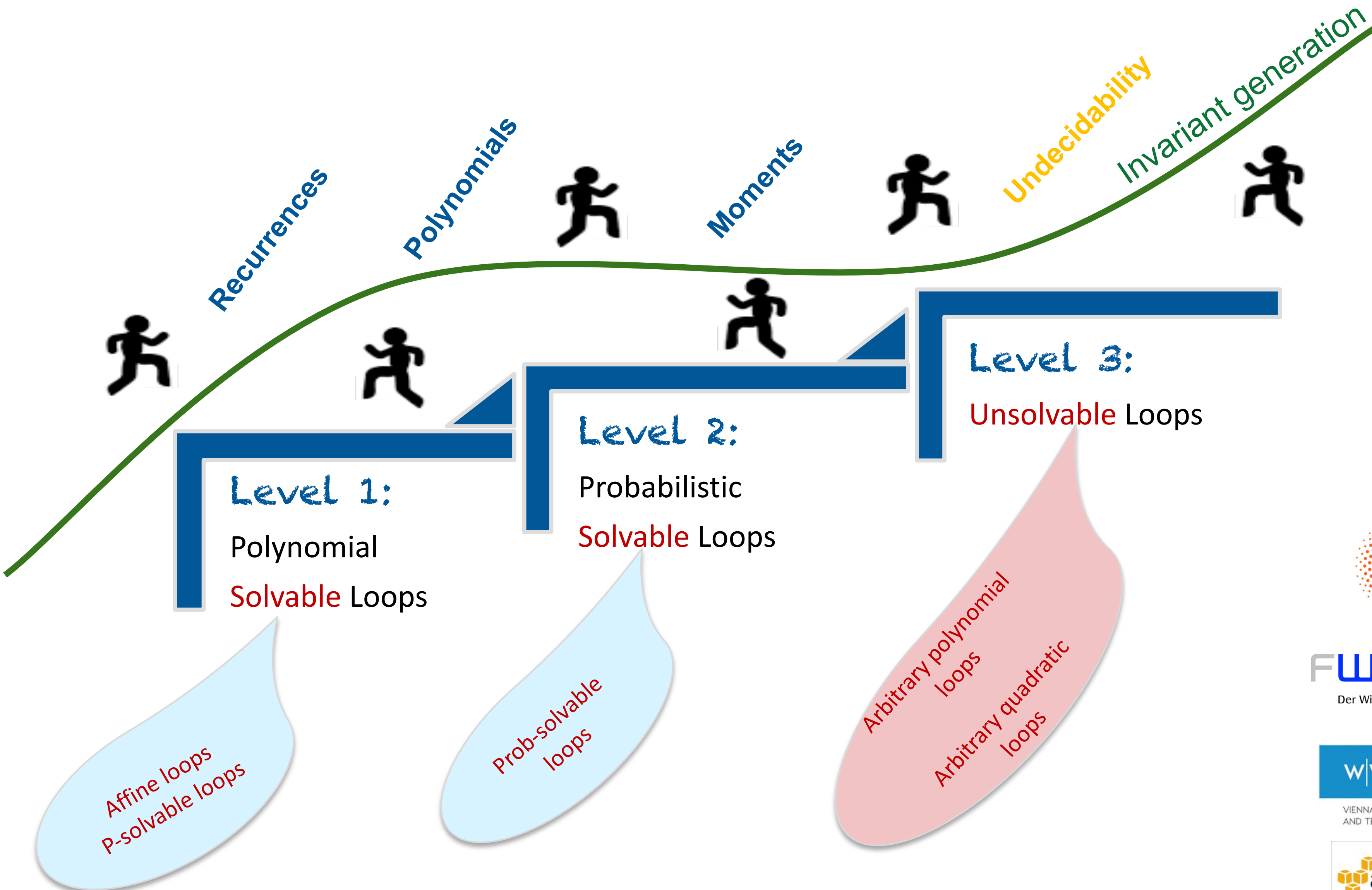

erc

FWF Der Wissenschaftsfonds.

W|W|T|F VIENNA SCIENCE AND TECHNOLOGY FUND

amazon webservices™

# Symbolic Computation in Automated Program Reasoning




erc

FWF Der Wissenschaftsfonds.

WWTF VIENNA SCIENCE AND TECHNOLOGY FUND

amazon web services™