Automated Reasoning
for
System Security and Privacy

Laura Kovács

for(syte) TU WIEN Informatics

Automated Reasoning

for Rigorous Systems Engineering

In a vague sense, automated reasoning involves:

1. Representing a problem as a mathematical/logical statement

2. Computer-supported automatic check whether this statement is true

Automated Reasoning

for Rigorous Systems Engineering

My group @ TU Wien applies automated reasoning for:

- Ensuring correctness of software
- Generating program properties
- Software synthesis
- …
- **System security**

Automated Reasoning

for Rigorous Systems Engineering

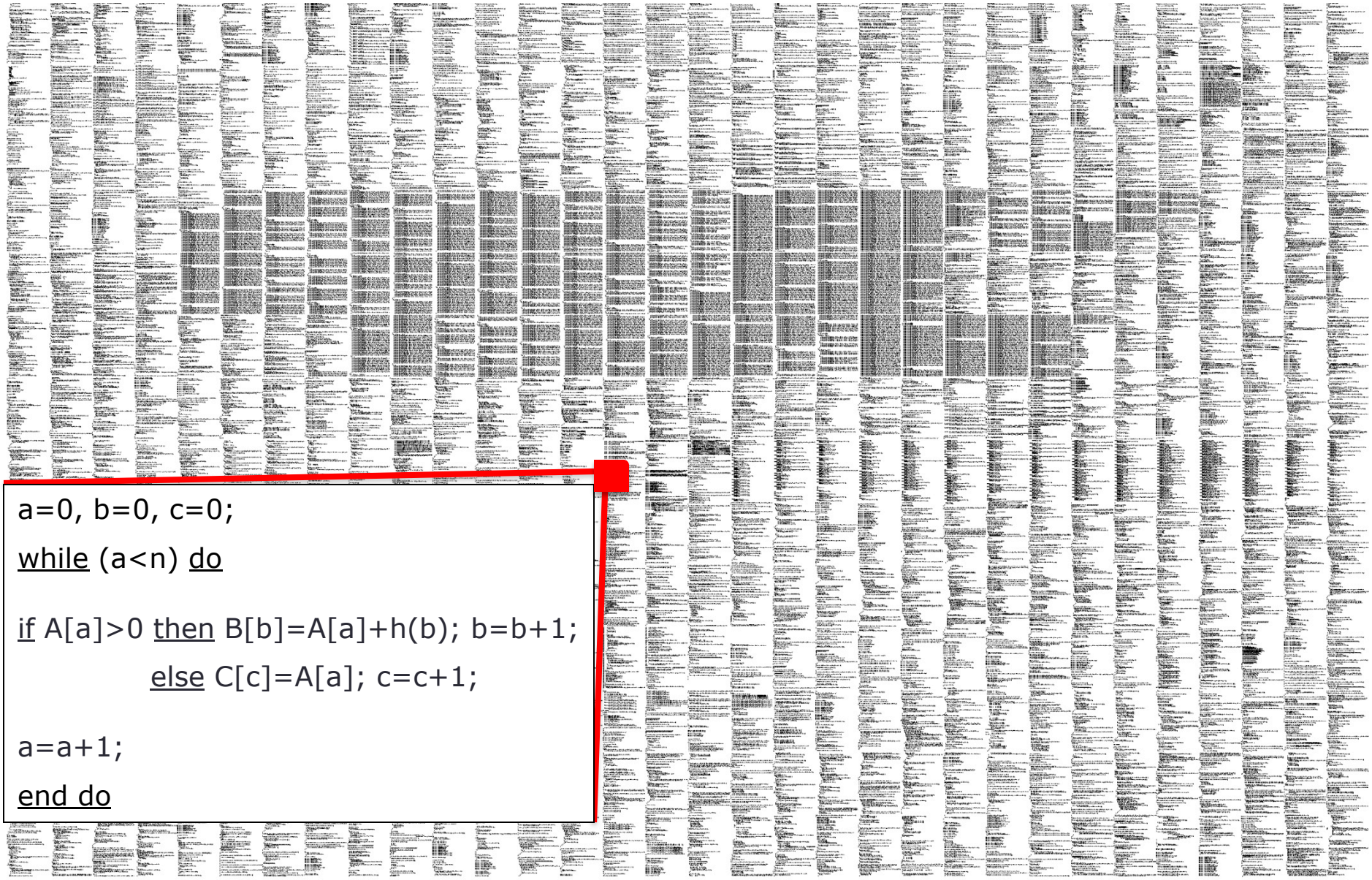
My group @ TU Wien applies automated reasoning for:

- Ensuring correctness of software
- Generating program properties
- Software synthesis
- …
- **System security**

Automated Reasoning for Software Correctness

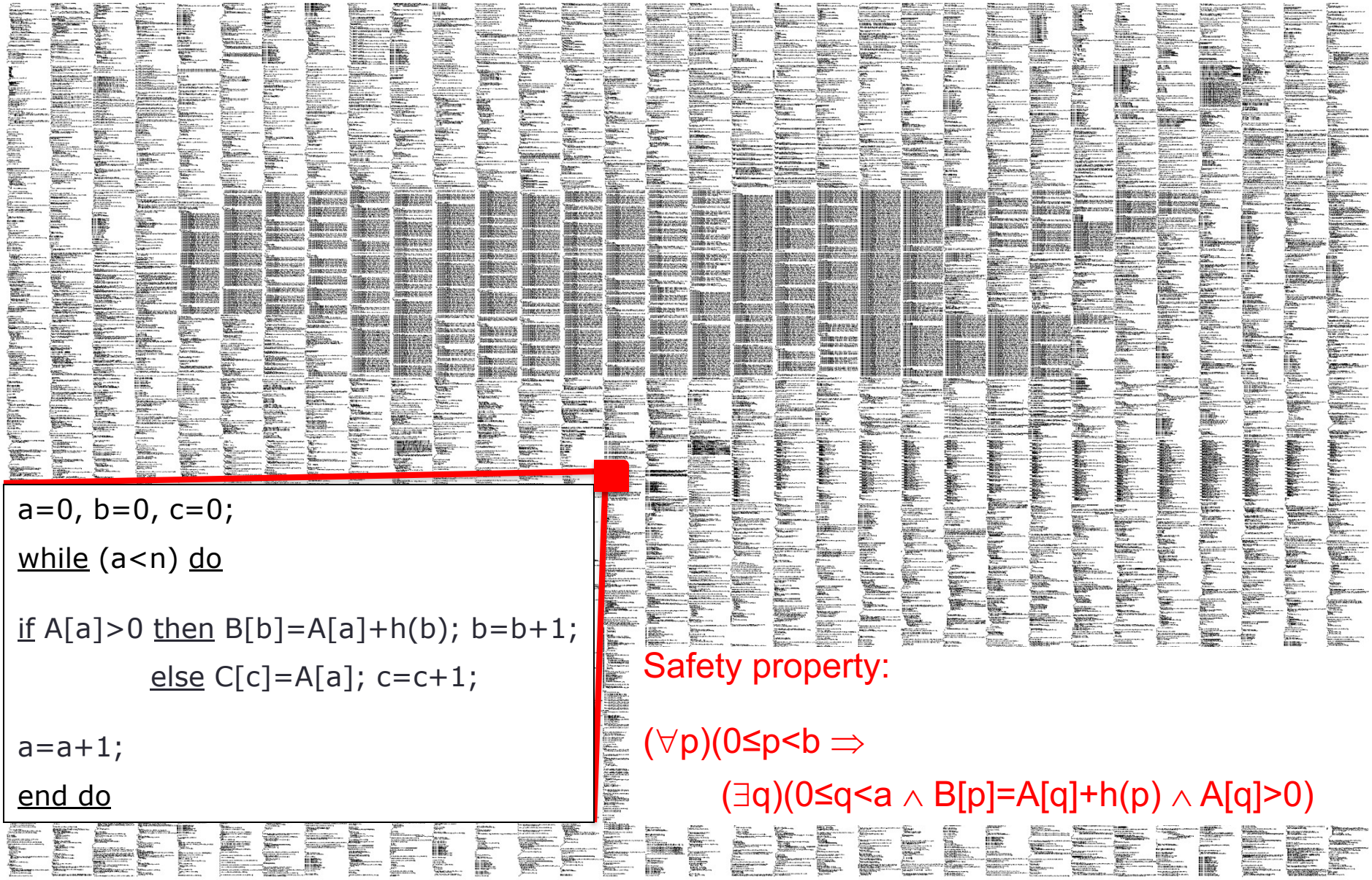
(ex. ~200kLoC, VAMPIRE prover)

Automated Reasoning for Software Correctness



```
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
```

Automated Reasoning for Software Correctness

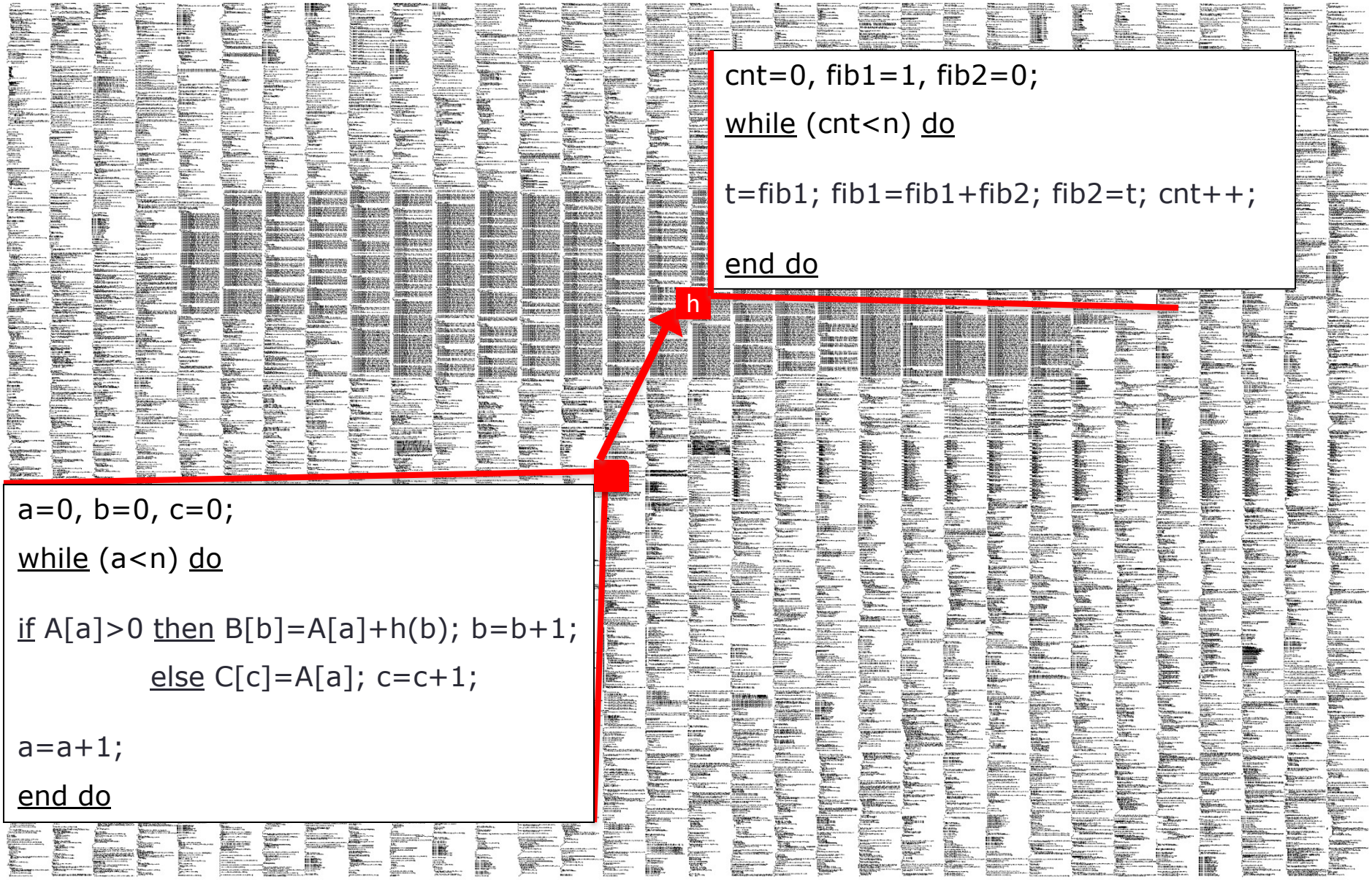


```
a=0, b=0, c=0;
while (a<n) do

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
```

Safety property:

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

Automated Reasoning for Software Correctness



Automated Reasoning for Software Correctness

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

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

end do
```

h

Safety property:

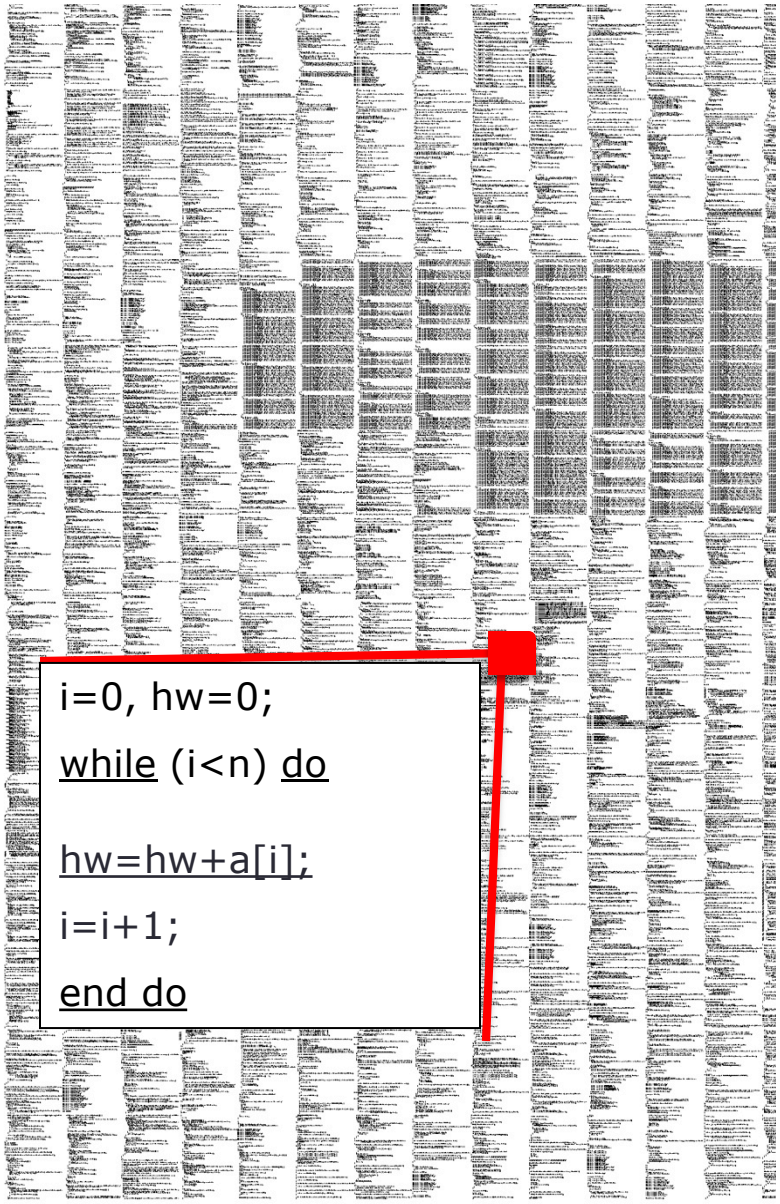
$fib1^4 + fib2^4 + 2*fib1*fib2^3 - 2\ fib1^3*fib2 - fib1^2*fib2^2 - 1 = 0$

```
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
```

Automated Reasoning for Software Correctness

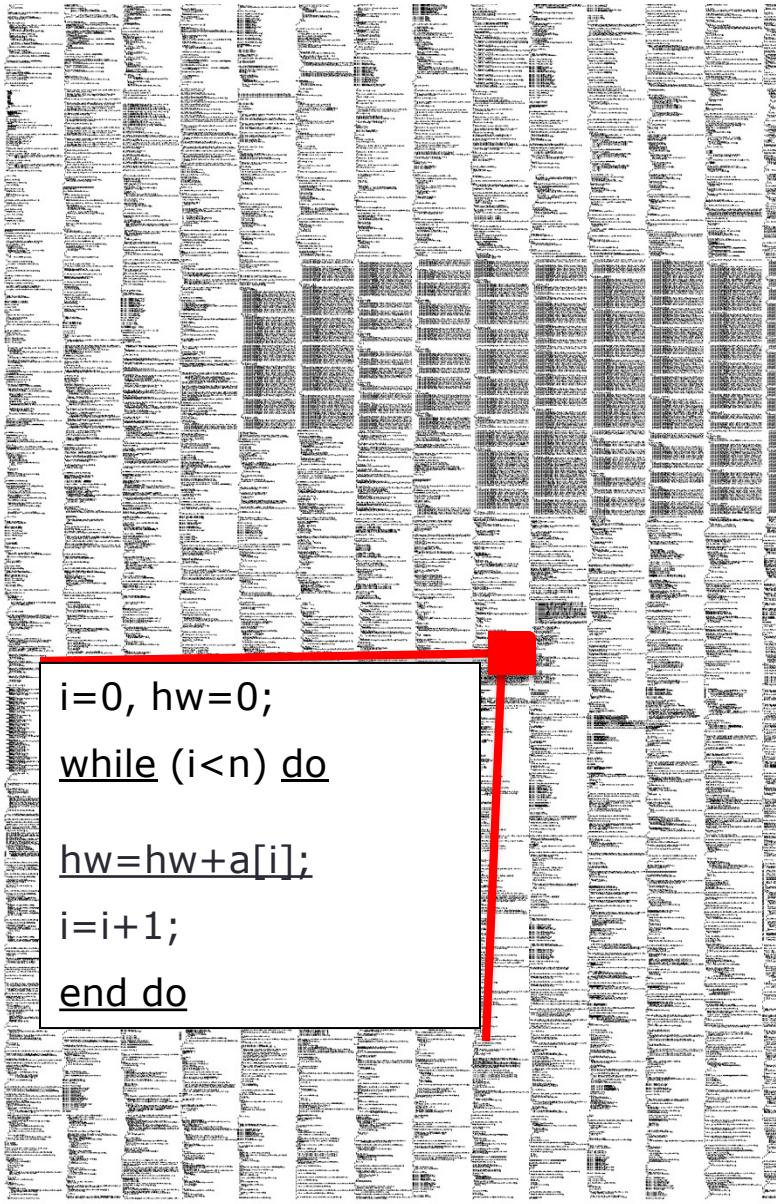
Automated Reasoning for System Security and Privacy

Automated Reasoning for Security and Privacy



- Array **a**: bit-wise representation of a secret key
- Hamming weight **hw**: number of 1s in the key

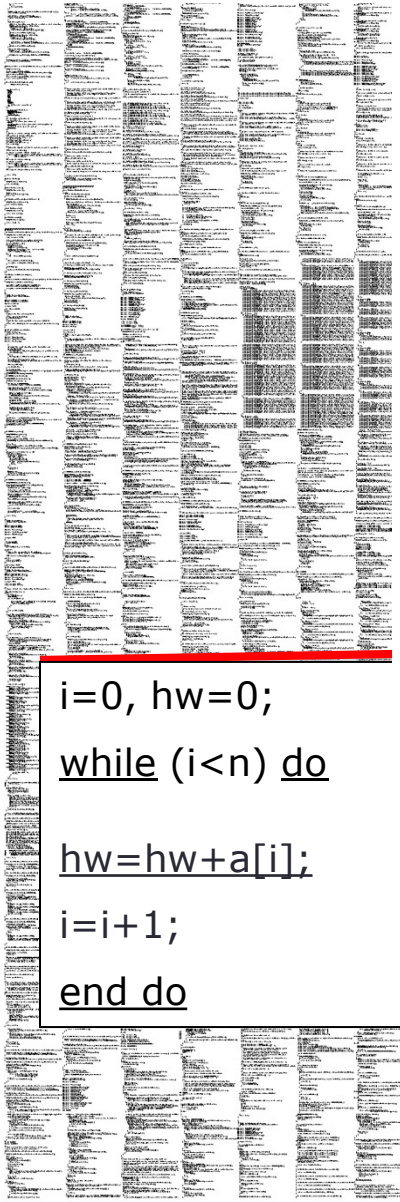
Automated Reasoning for Security and Privacy



- Array **a**: bit-wise representation of a secret key
- Hamming weight **hw**: number of 1s in the key

- Leaking **hw ?**
  (e.g. measure of side-channel leakage)

Automated Reasoning for Security and Privacy



i=0, hw=0;

while (i<n) do

hw=hw+a[i];

i=i+1;

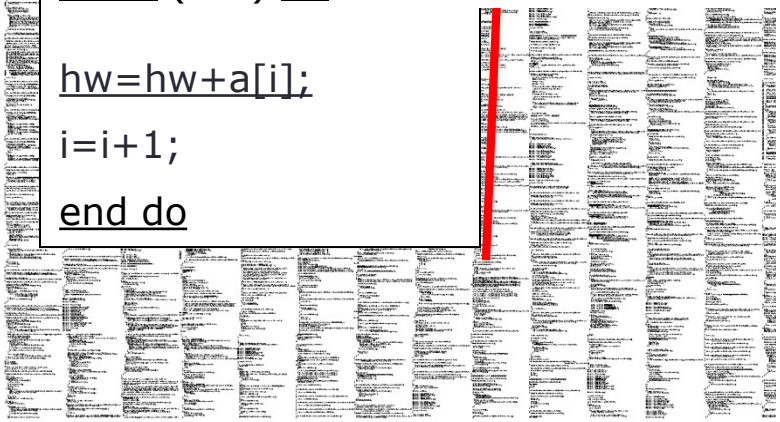
end do

Verifying Relational Properties using Trace Logic

Gilles Barthe*†, Renate Eilers‡, Pamina Georgiou‡, Bernhard Gleiss‡, Laura Kovács‡§, Matteo Maffei‡
*Max Planck Institute for Security and Privacy, Germany
†IMDEA Software Institute, Spain
‡TU Wien, Austria
§Chalmers University of Technology, Sweden

***Abstract*—We present a logical framework for the verification of relational properties in imperative programs. Our framework reduces verification of relational properties of imperative programs to a validity problem in trace logic, an expressive instance of first-order predicate logic. Trace logic draws its expressiveness from its syntax, which allows expressing properties over computation traces. Its axiomatization supports fine-grained reasoning about intermediate steps in program execution, notably loop iterations. We present an algorithm to encode the semantics of programs as well as their relational properties in trace logic, and then show how first-order theorem proving can be used to reason about the resulting trace logic formulas. Our work is implemented in the tool RAPID and evaluated with examples coming from the security field.**

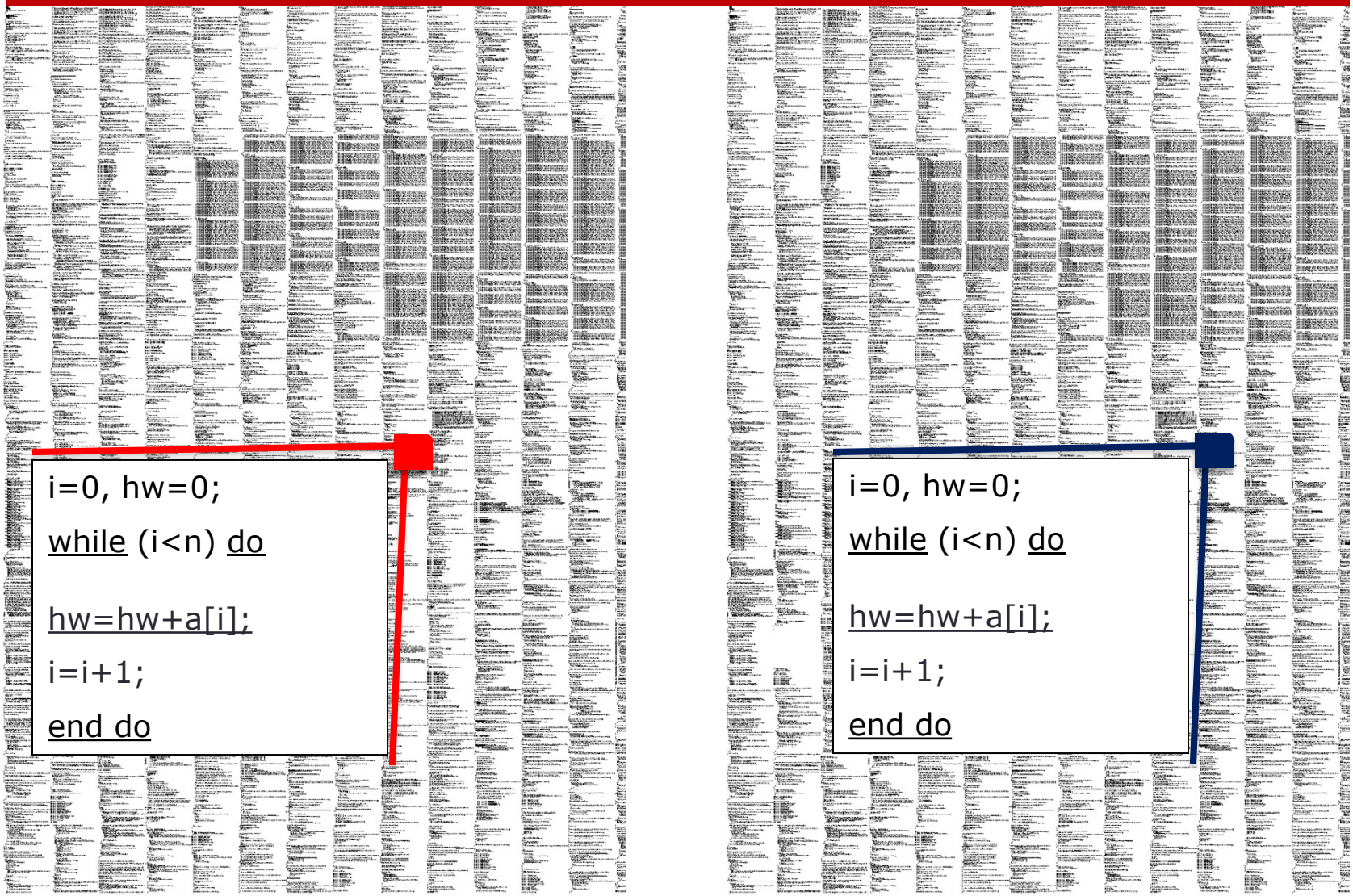
```
 1   func main()
 2  {
 3       const Int[] a;
 4       const Int alength;
 5
 6       Int i = 0;
 7       Int hw = 0;
 8
 9       while (i < alength)
10       {
11          hw = hw + a[i];
12          i = i + 1;
13       }
14    }
```



- No matter what permutation of **a**, the **hw** is the same

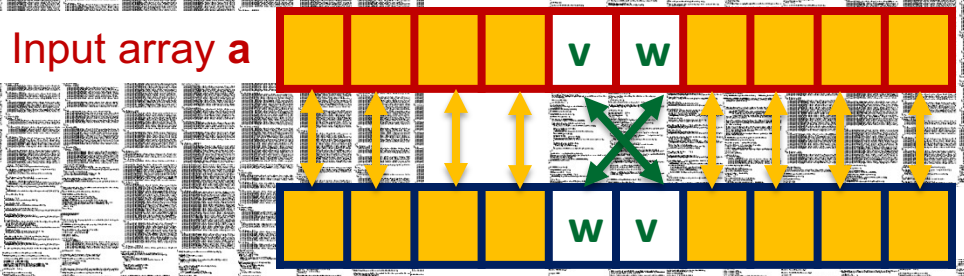
Automated Reasoning for Security and Privacy

Relational Verification



Automated Reasoning for Security and Privacy

Relational Verification

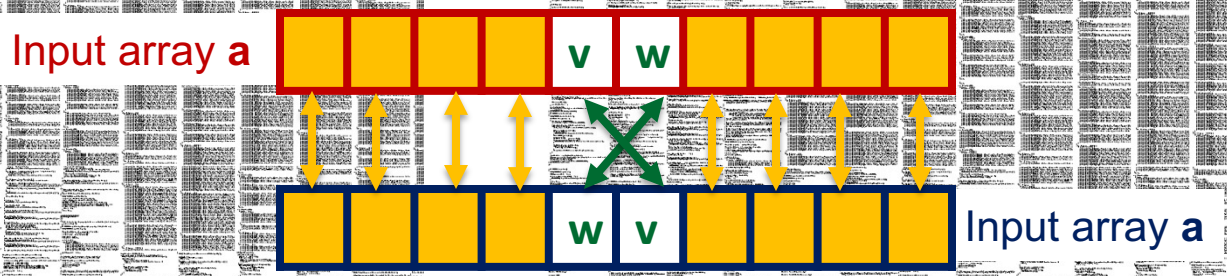



```
i=0, hw=0;
while (i<n) do

hw=hw+a[i];
i=i+1;
end do
```

```
i=0, hw=0;
while (i<n) do

hw=hw+a[i];
i=i+1;
end do
```

Automated Reasoning for Security and Privacy

Relational Verification



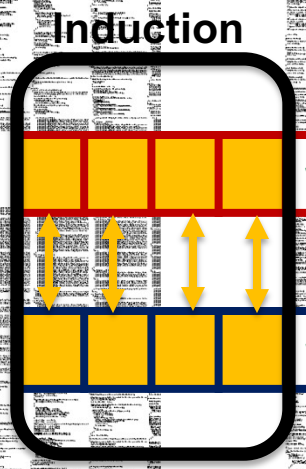

```
i=0, hw=0;
while (i<n) do

hw=hw+a[i];
i=i+1;
end do
```

```
i=0, hw=0;
while (i<n) do

hw=hw+a[i];
i=i+1;
end do
```

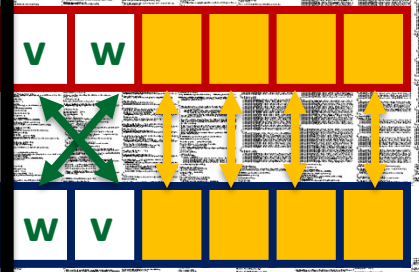
hw = hw

Automated Reasoning for Security and Privacy

Relational Verification



Input array **a**



Input array **a**

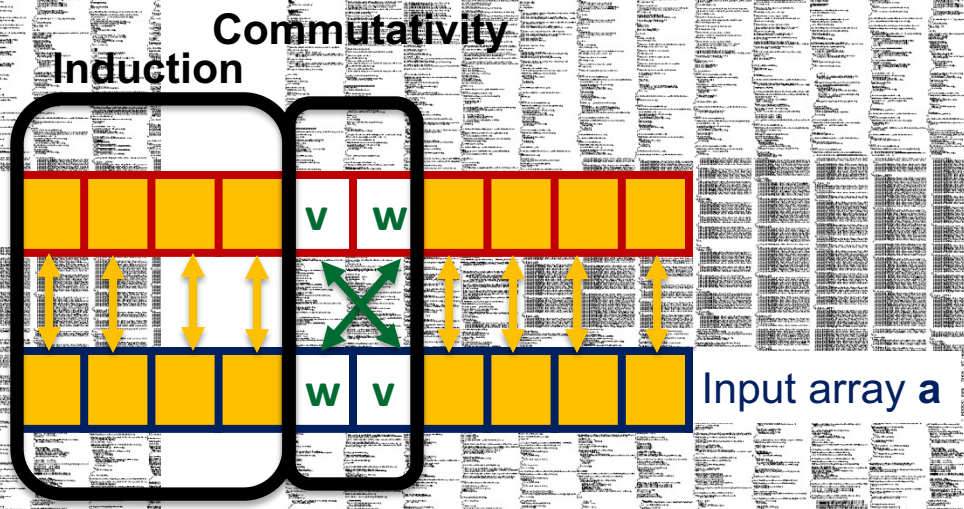
```
i=0, hw=0;
while (i<n) do

hw=hw+a[i];
i=i+1;
end do
```

```
i=0, hw=0;
while (i<n) do

hw=hw+a[i];
i=i+1;
end do
```

hw = hw

Automated Reasoning for Security and Privacy

Relational Verification



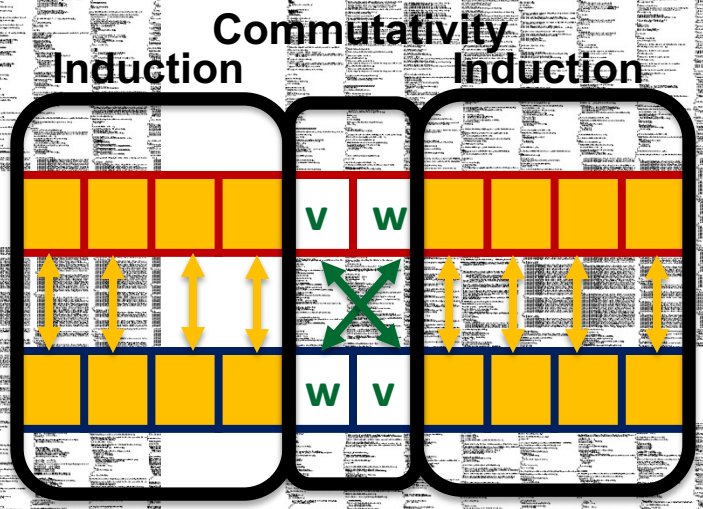

```
i=0, hw=0;
while (i<n) do

hw=hw+a[i];
i=i+1;
end do
```

```
i=0, hw=0;
while (i<n) do

hw=hw+a[i];
i=i+1;
end do
```

hw = hw

Automated Reasoning for Security and Privacy

Relational Verification



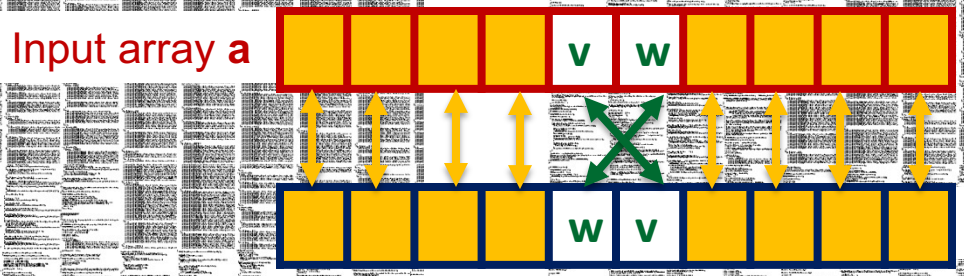

```
i=0, hw=0;
while (i<n) do
hw=hw+a[i];
i=i+1;
end do
```

```
i=0, hw=0;
while (i<n) do
hw=hw+a[i];
i=i+1;
end do
```

hw = hw

Automated Reasoning for Security and Privacy

Relational Verification



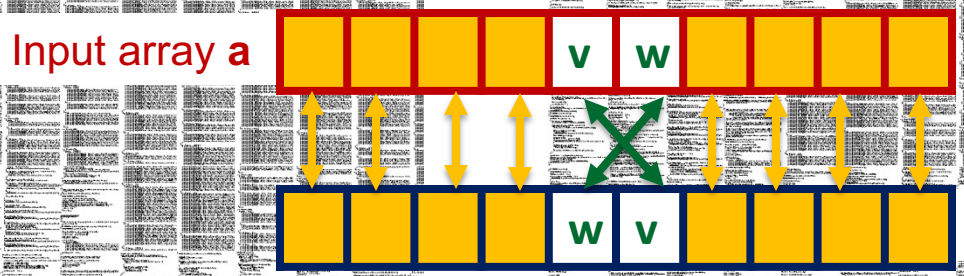

```
i=0, hw=0;
while (i<n) do

hw=hw+a[i];
i=i+1;
end do
```

```
i=0, hw=0;
while (i<n) do

hw=hw+a[i];
i=i+1;
end do
```

hw = hw

Automated Reasoning for Security and Privacy

Relational Verification (non-interference, sensitivity)




```
i=0, hw=0;
while (i<n) do

hw=hw+a[i];
i=i+1;
end do
```

```
i=0, hw=0;
while (i<n) do

hw=hw+a[i];
i=i+1;
end do
```

hw = hw

Automated Reasoning for Security and Privacy

New Reasoning Challenges:

- **First-order theories** of data structures

  quantifiers + int/real, naturals/term algebras, arrays, …

Automated Reasoning for Security and Privacy

New Reasoning Challenges:

- **First-order theories** of data structures

  quantifiers + int/real, naturals/term algebras, arrays, …

- **Induction**

  not a first-order property

Automated Reasoning for Security and Privacy

New Reasoning Challenges:

- **First-order theories** of data structures

  quantifiers + int/real, naturals/term algebras, arrays, …

- **Induction**
  not a first-order property

- **Aggregates** (sum, min, max, …)
  not first-order quantifiers

What Kind of Automated Reasoners can be Used?

	Input	Examples	Impact
SAT Solver	Propositional formulae	MiniSat, Lingeling	Intel
SMT Solvers	(First-order) formulae + theories	CVC4, Z3	Microsoft, Amazon
Theorem Provers	First-order formulae (+ theories)	Vampire, E	Intel, Amazon
Proof Assistants (interactive)	Higher-order formulae	Isabelle, Coq	Intel, Apple

Our Automated Reasoner: VAMPIRE

	Input	Examples	Impact
SAT Solver	Propositional formulae	MiniSat, Lingeling	Intel
SMT Solvers	(First-order) formulae + theories	CVC4, Z3	Microsoft, Amazon
Theorem Provers	First-order formulae (+ theories)	**Vampire**, E	Intel, Amazon
Proof Assistants (interactive)	Higher-order formulae	Isabelle, Coq	Intel, Apple

What is VAMPIRE?

- An automated theorem prover for first-order logic and theories.

  https://vprover.github.io/download.html

- Completely automatic: once you started a proof attempt, it can only be interrupted by terminating the process.

- Champion of the CASC world-cup in first-order theorem proving: won CASC > 50 times.

VAMPIRE

- It produces detailed proofs
- It competes with SMT solvers on their problems
- In normal operation, it is saturation-based
- It is portfolio-based - works best when uses lots of strategies
- It supports lots of extra features and options helpful, for example, system security, including induction and theory reasoning.

Automated Reasoning with VAMPIRE

for Security and Privacy

Benchmarks	Vampire				CVC4	Z3
	S	S+A	F	F+A		
1-hw-equal-arrays	✓	✓	-	✓	✓	✓
2-hw-last-position-swapped	-	✓	-	-	✓	✓
3-hw-swap-and-two-arrays	-	✓	-	-	-	-
4-hw-swap-in-array-lemma	-	✓	-	-	-	-
4-hw-swap-in-array-full	-	✓	-	-	-	-
1-ni-assign-to-high	✓	✓	✓	✓	✓	✓
2-ni-branch-on-high-twice	✓	✓	✓	✓	✓	✓
3-ni-high-guard-equal-branches	✓	✓	✓	✓	✓	✓
4-ni-branch-on-high-twice-prop2	✓	✓	-	-	✓	✓
5-ni-temp-impl-flow	-	-	✓	✓	✓	✓
6-ni-branch-assign-equal-val	-	-	✓	✓	✓	✓
7-ni-explicit-flow	✓	✓	✓	✓	✓	✓
8-ni-explicit-flow-while	✓	✓	-	✓	✓	✓
9-ni-equal-output	✓	-	-	-	-	✓
10-ni-rsa-exponentiation	✓	✓	✓	✓	✓	-
1-sens-equal-sums	✓	✓	✓	✓	✓	✓
2-sens-equal-sums-two-arrays	✓	✓	✓	✓	-	-
3-sens-abs-diff-up-to-k	-	-	-	-	✓	✓
4-sens-abs-diff-up-to-k-two-arrays	-	-	-	-	-	-
5-sens-two-arrays-equal-k	✓	✓	✓	✓	-	-
6-sens-diff-up-to-explicit-k	✓	✓	✓	✓	-	-
7-sens-diff-up-to-explicit-k-sum	-	-	✓	✓	-	-
8-sens-explicit-swap	-	-	✓	✓	-	-
9-sens-explicit-swap-prop2	-		✓	✓	-	-
10-sens-equal-k	✓	✓	✓	✓	-	-
11-sens-equal-k-twice	✓	✓	✓	✓	-	-
12-sens-diff-up-to-forall-k	-	-	✓	✓	✓	-
Total VAMPIRE	15	18	17	19		
Unique VAMPIRE	1	4	0	0		
Total	25				14	13

Automated Reasoning with VAMPIRE

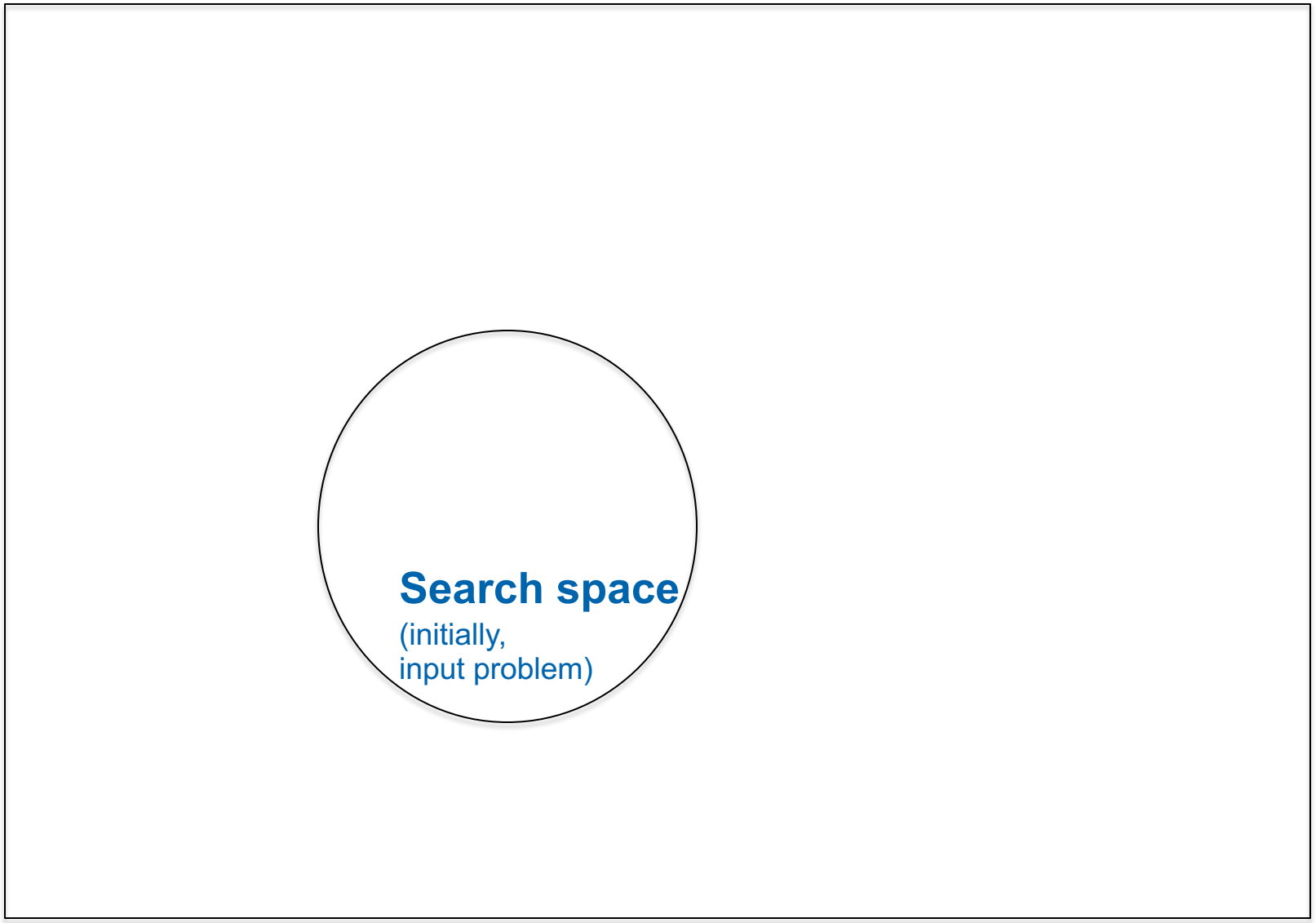
Proof by Refutation

Given an input problem with assumptions F_1, …, F_n and goal G:

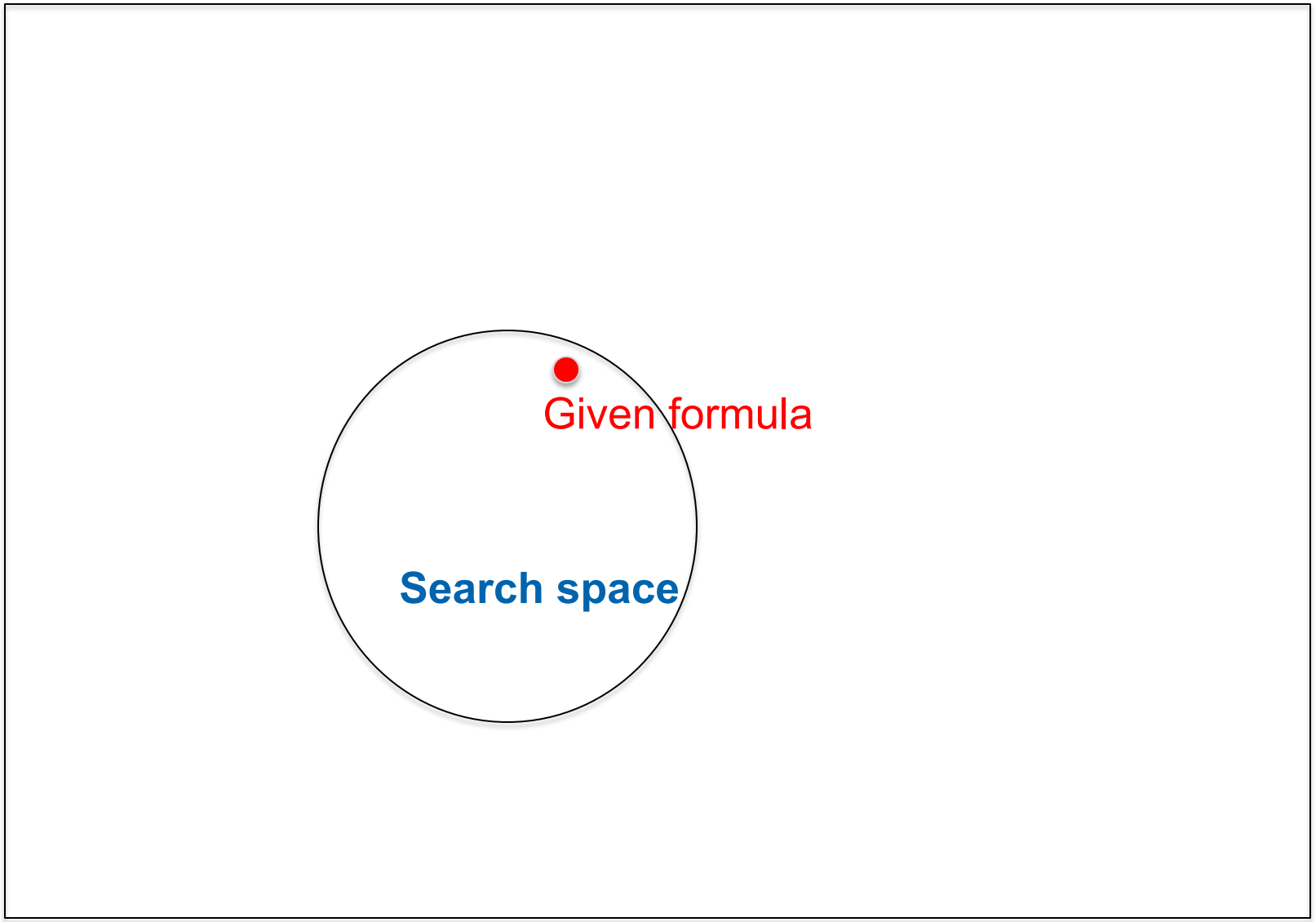
1. Negate the conjecture (¬G);

2. Establish unsatisfiability of the set of formulas F_1, …, F_n, ¬G.

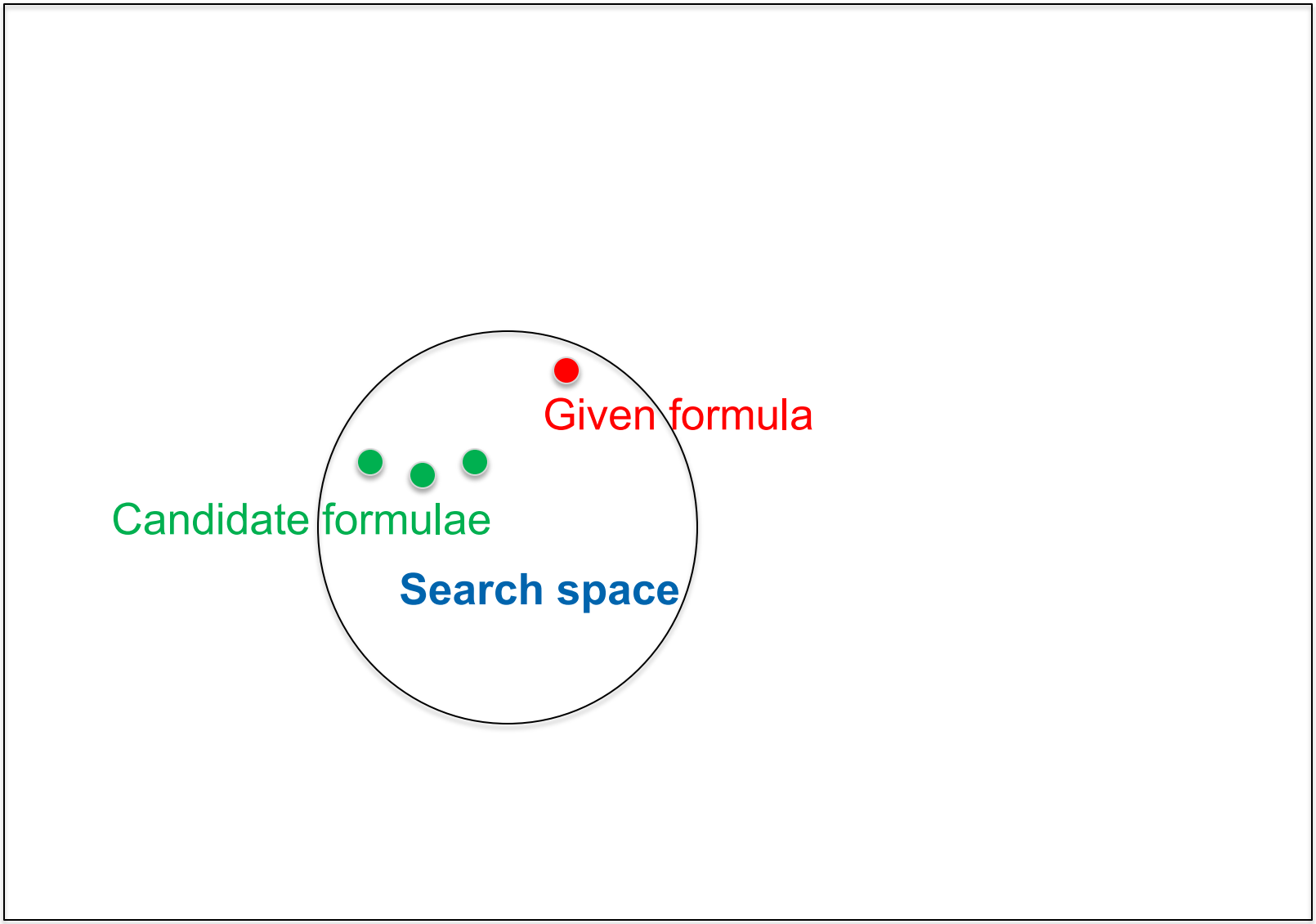
Automated Reasoning with VAMPIRE – Saturation



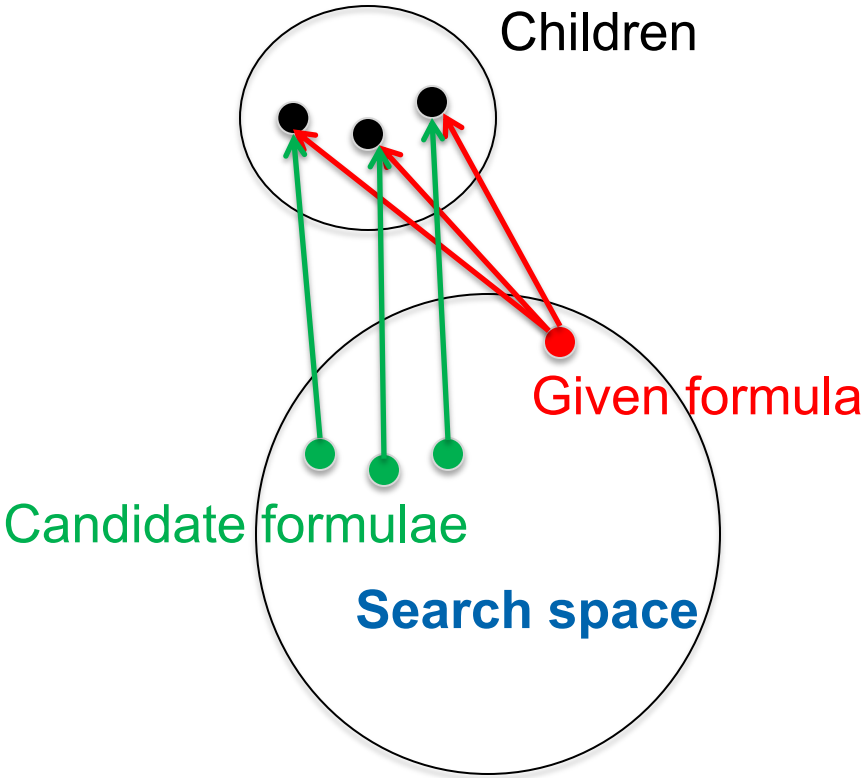
Automated Reasoning with VAMPIRE – Saturation



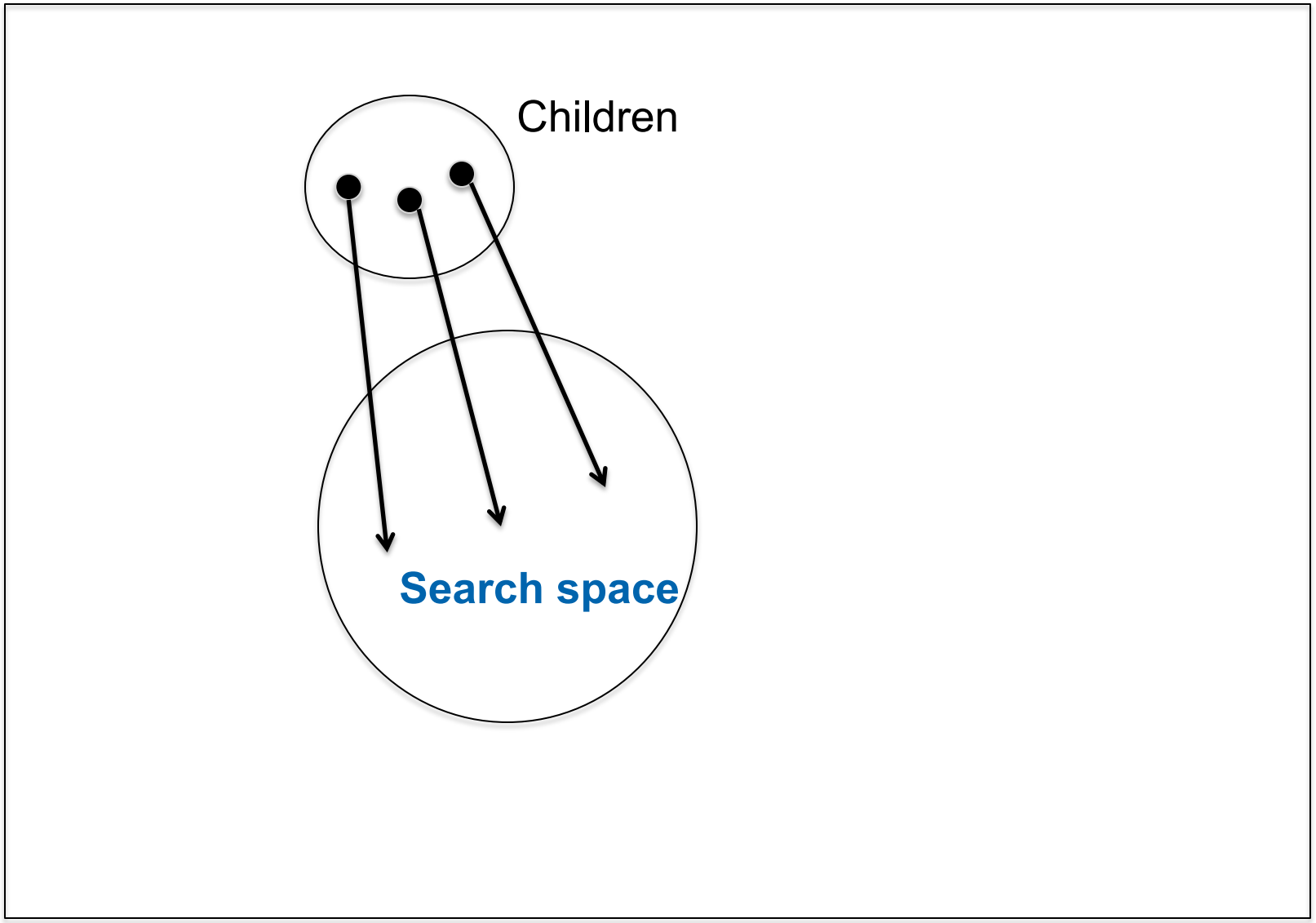
Automated Reasoning with VAMPIRE – Saturation



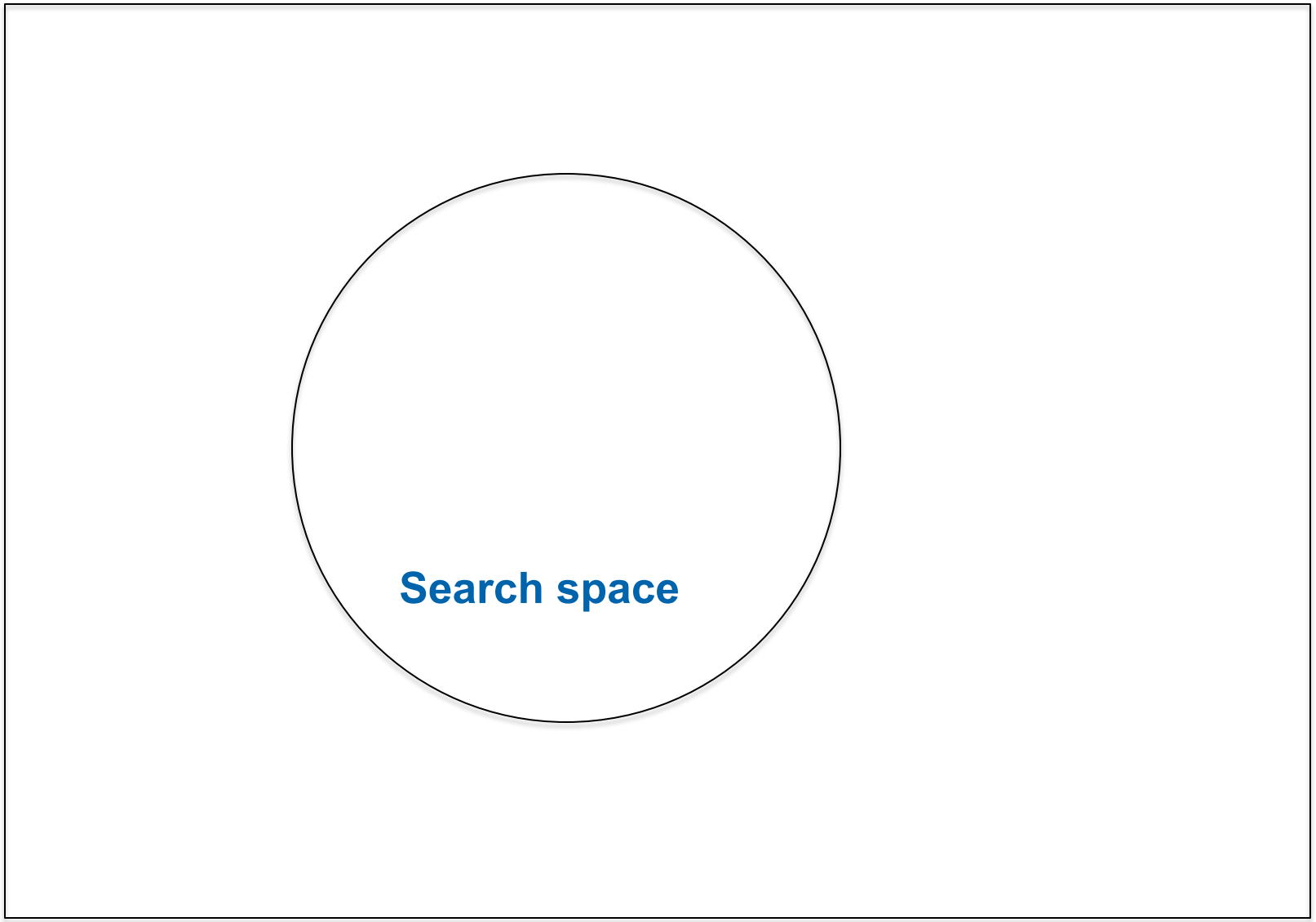
Automated Reasoning with VAMPIRE – Saturation



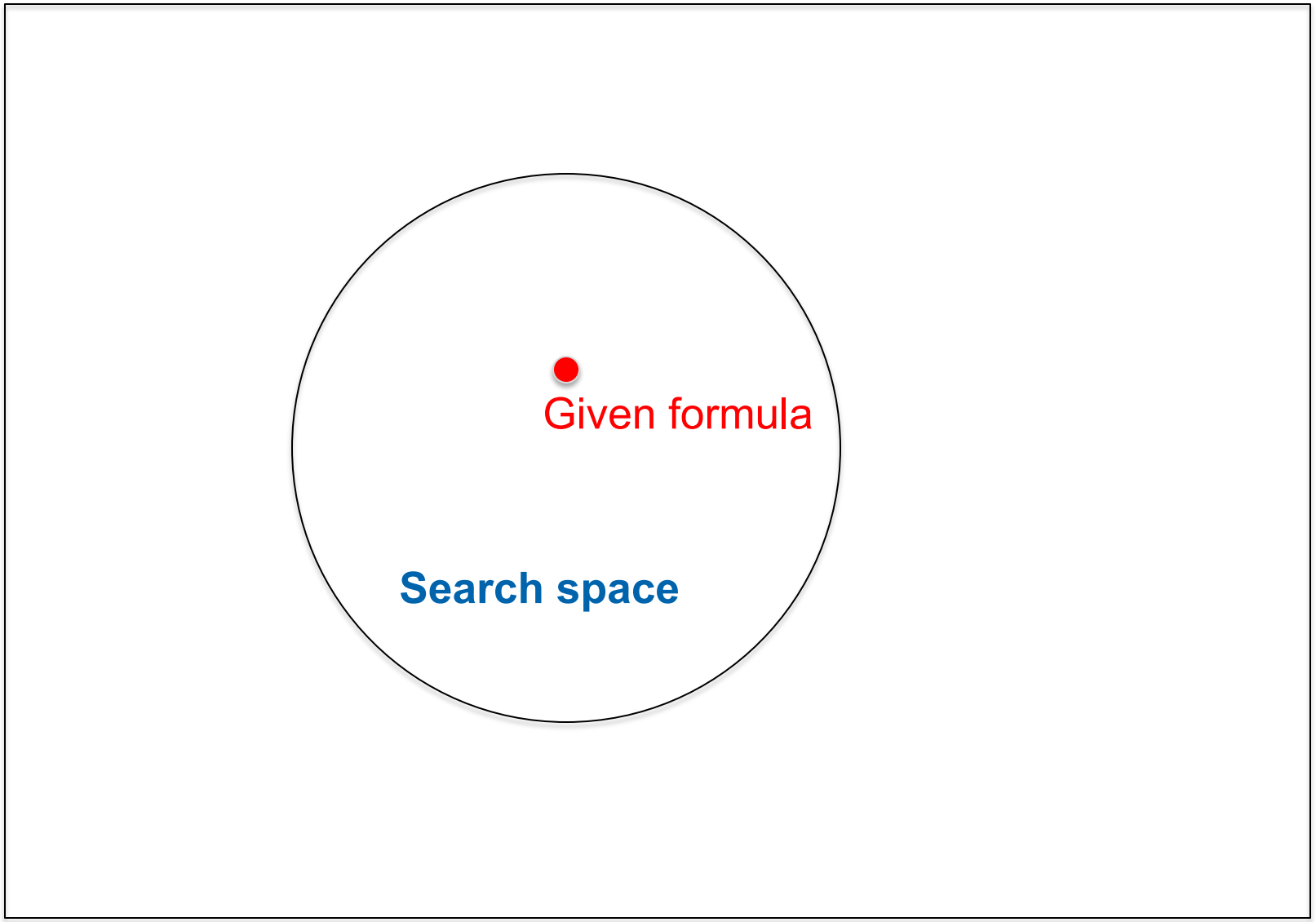
Automated Reasoning with VAMPIRE – Saturation



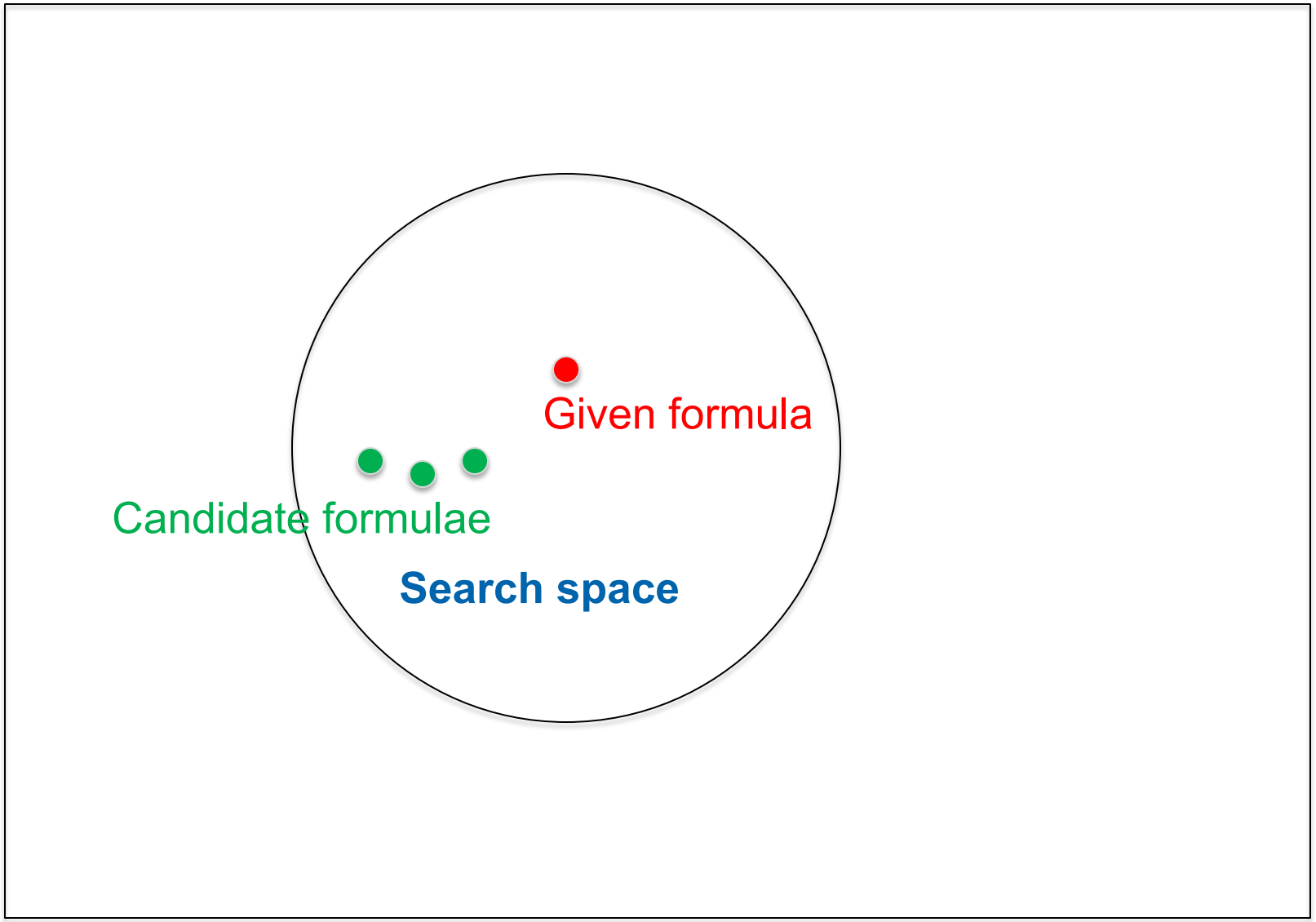
Automated Reasoning with VAMPIRE – Saturation



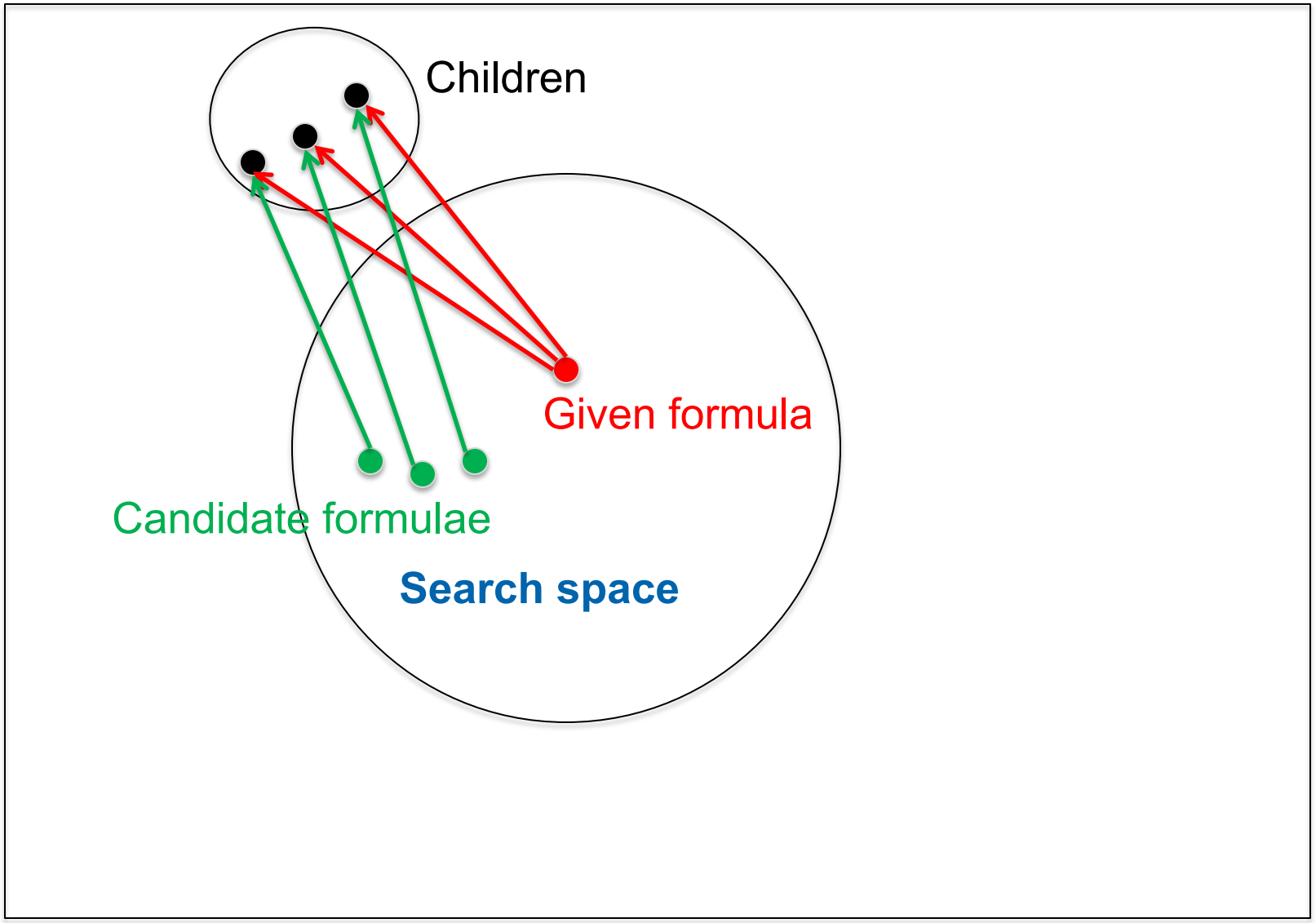
Automated Reasoning with VAMPIRE – Saturation



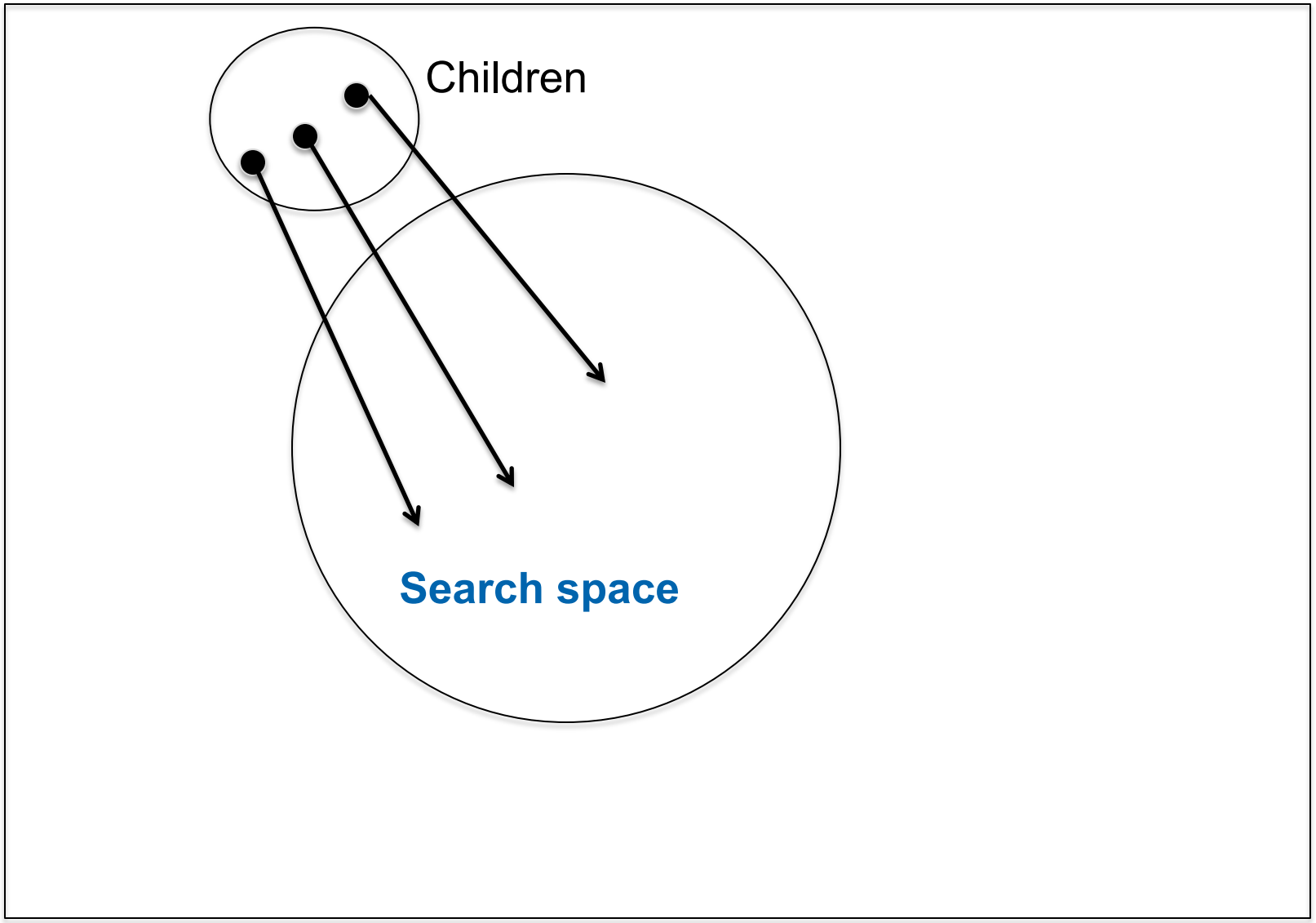
Automated Reasoning with VAMPIRE – Saturation



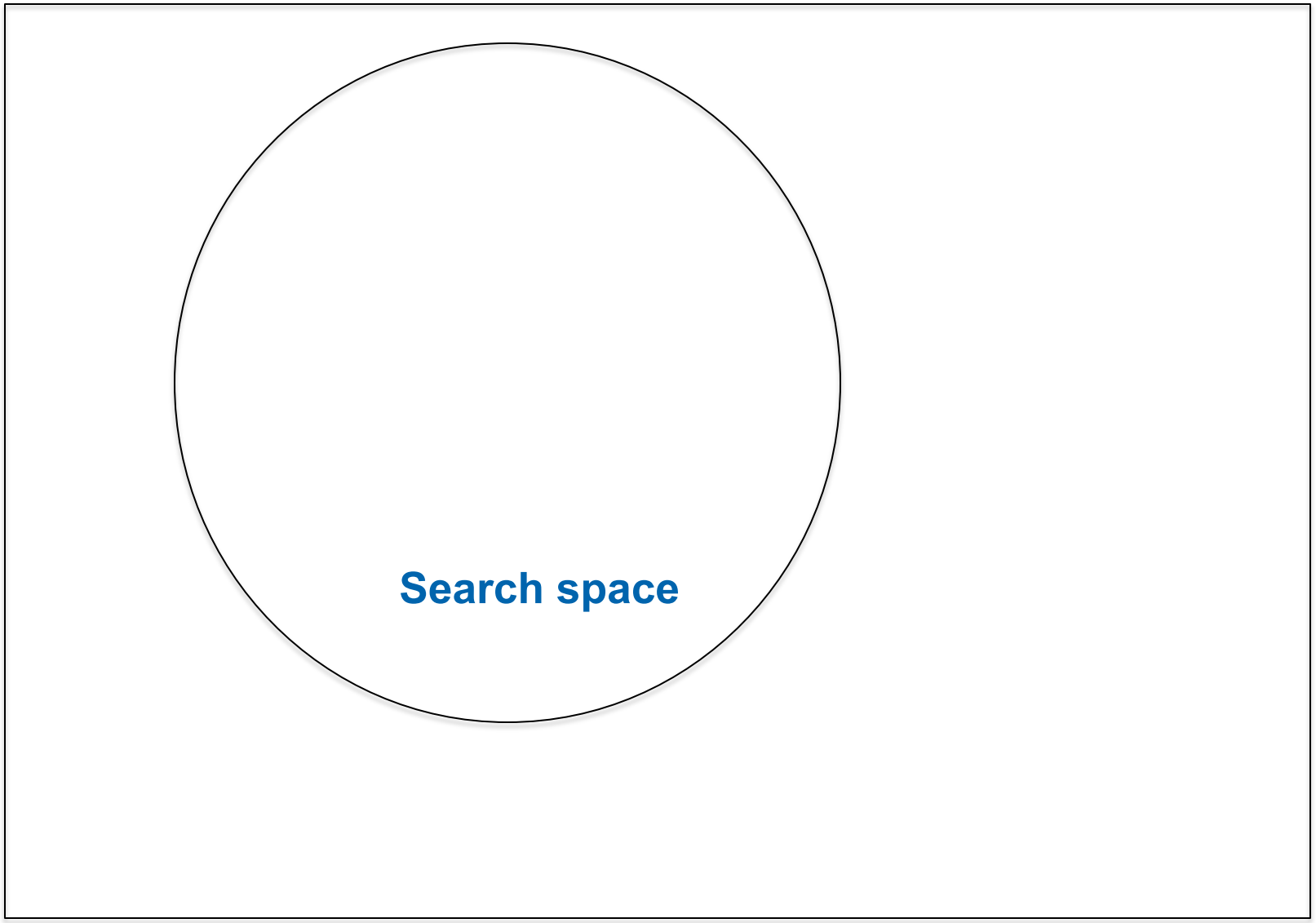
Automated Reasoning with VAMPIRE – Saturation



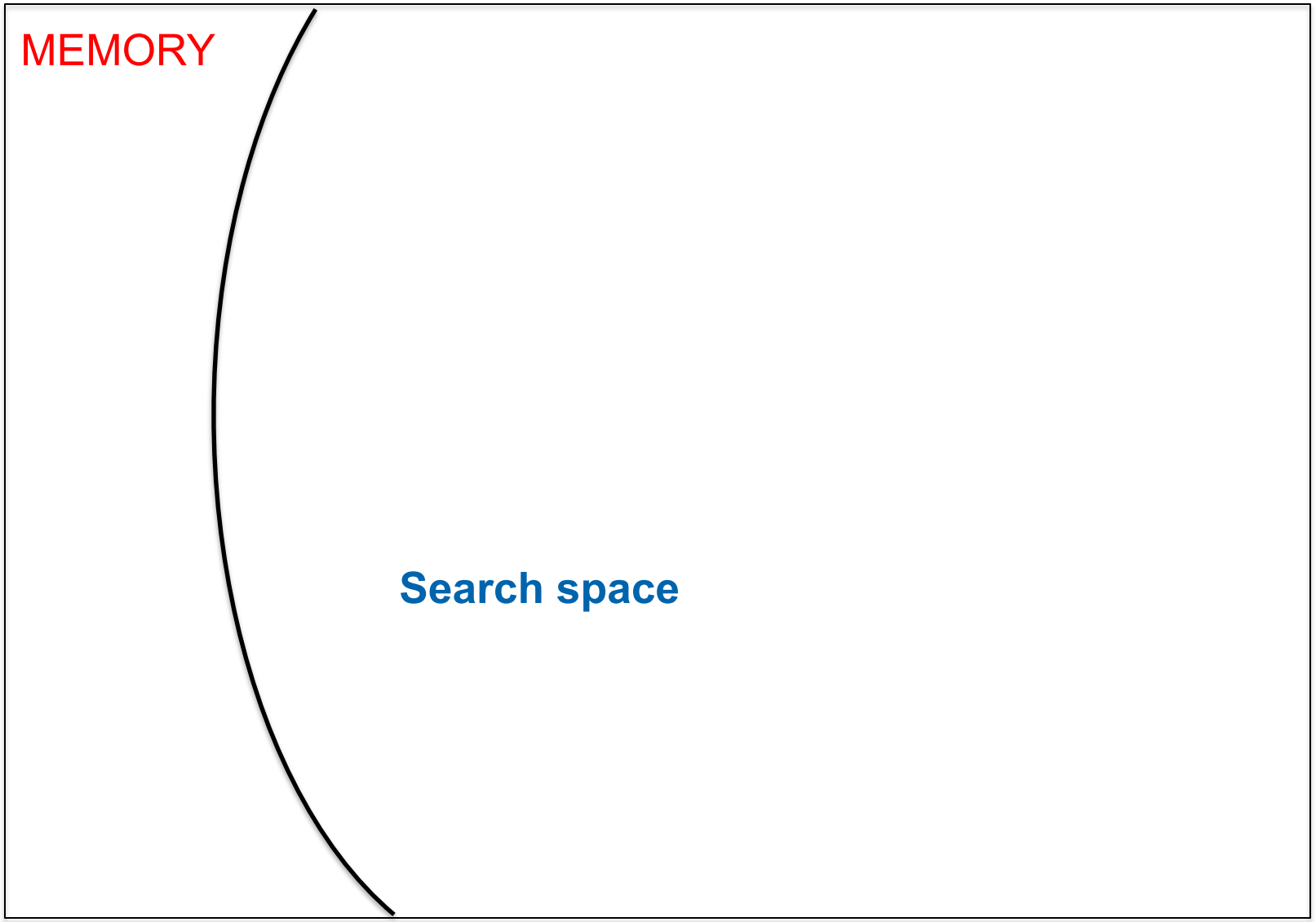
Automated Reasoning with VAMPIRE – Saturation



Automated Reasoning with VAMPIRE – Saturation



Automated Reasoning with VAMPIRE – Saturation



Automated Reasoning with VAMPIRE – In practice

In practice there are three possible scenarios:

1. At some moment proof is found; in this case, the input is valid/true.

2. Saturation will terminate without ever finding a proof, in this case the input is satisfiable.

3. Saturation will run until we run out of resources, but without ever finding a proof. In this case it is unknown whether the input is valid.

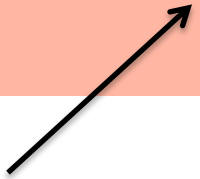
Automated Reasoning with VAMPIRE – In practice

In practice there are three possible scenarios:

1. At some moment proof is found; in this case, the input is valid/true.

2. Saturation will terminate without ever finding a proof, in this case the input is satisfiable.

3. Saturation will run until we run out of resources, but without ever finding a proof. In this case it is unknown whether the input is valid.



CHALLENGE: How to solve unknown?

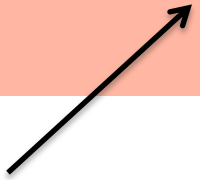
Automated Reasoning with VAMPIRE – In practice

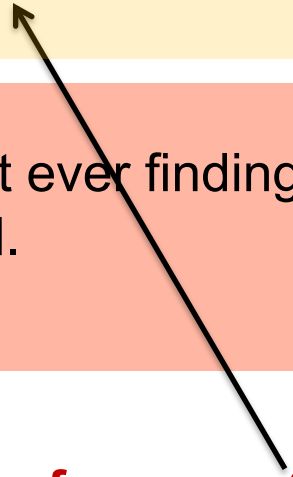
In practice there are three possible scenarios:

1. At some moment proof is found; in this case, the input is valid/true.

2. Saturation will terminate without ever finding a proof, in this case the input is satisfiable.

3. Saturation will run until we run out of resources, but without ever finding a proof. In this case it is unknown whether the input is valid.





CHALLENGE: How to solve unknown? How to improve performance?

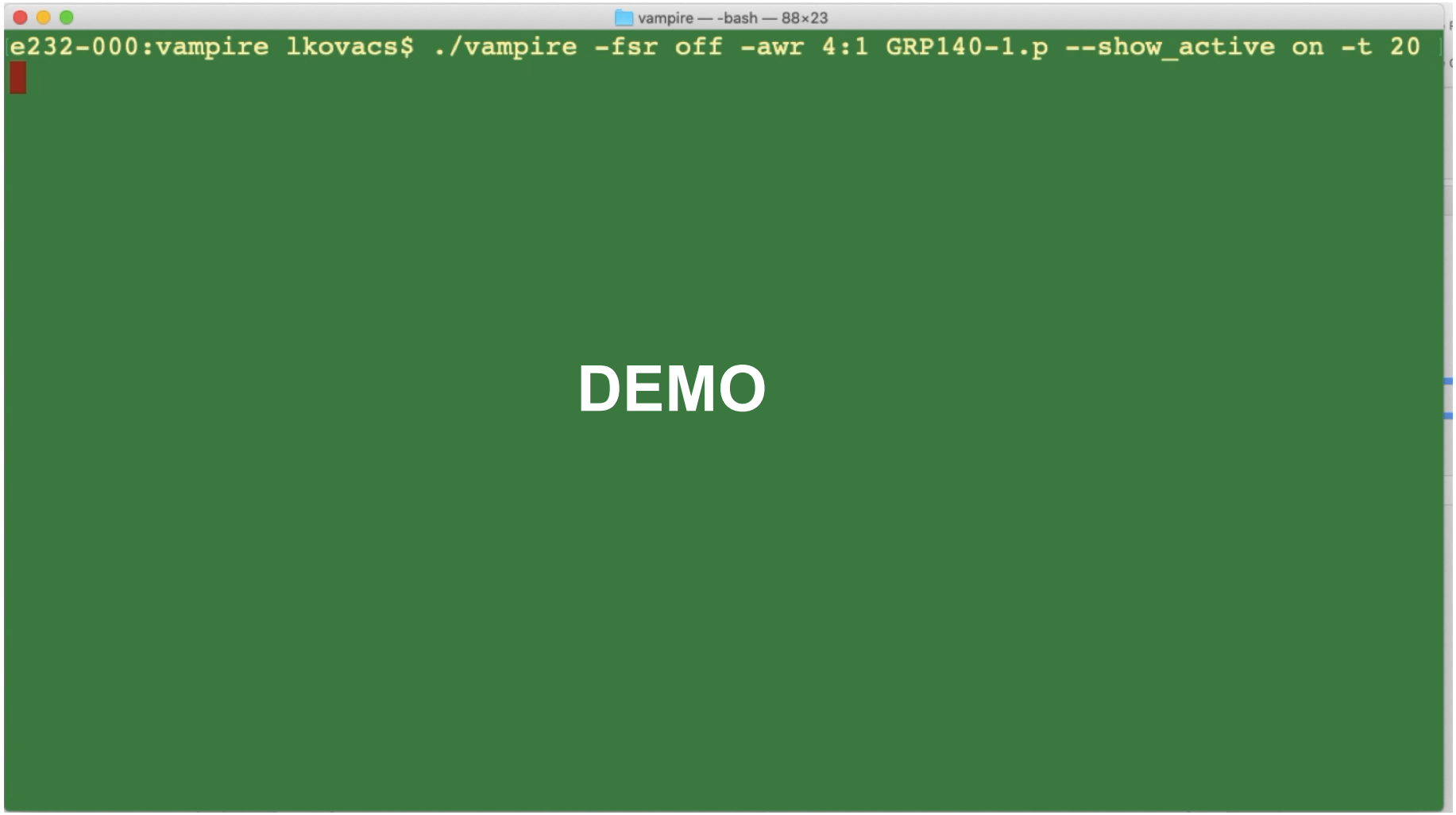
Automated Reasoning with VAMPIRE – Our recipe

I can't get no satisfaction:

-and I try …
-and I try …
-and I try …
-and I try

[The Rolling Stones]

Automated Reasoning with VAMPIRE – Our recipe



Conclusion and Outlook

1. Automated reasoning will remain central in rigorous systems engineering.

   The role of automated reasoning in these areas is and will be growing.

Conclusion and Outlook

1. Automated reasoning will remain central in rigorous systems engineering.

   The role of automated reasoning in these areas is and will be growing.

2. Automated reasoners will be used by a large number of users who do not understand automated reasoning and by users with very elementary knowledge of logic.

Conclusion and Outlook

1. Automated reasoning will remain central in rigorous systems engineering.

   The role of automated reasoning in these areas is and will be growing.

2. Automated reasoners will be used by a large number of users who do not understand automated reasoning and by users with very elementary knowledge of logic.

   Security cannot be handled by engineers alone

Conclusion and Outlook

1. Automated reasoning will remain central in rigorous systems engineering.

   The role of automated reasoning in these areas is and will be growing.

2. Automated reasoners will be used by a large number of users who do not understand automated reasoning and by users with very elementary knowledge of logic.

Security cannot be handled by engineers alone.

Security cannot be handled by experts alone.

Conclusion and Outlook

1. Automated reasoning will remain central in rigorous systems engineering.

   The role of automated reasoning in these areas is and will be growing.

2. Automated reasoners will be used by a large number of users who do not understand automated reasoning and by users with very elementary knowledge of logic.

3. Automated reasoning with quantifiers and theories will remain the main challenge in ensuring system security/safety (at least) for the next decade.

erc European Research Council Supporting top researchers from anywhere in the world
Knut and Alice Wallenberg Foundation
WALLENBERG ACADEMY FELLOWS
FWF Der Wissenschaftsfonds.
WWTF VIENNA SCIENCE AND TECHNOLOGY FUND