Advanced Generalization Techniques

Yakir Vizel

Technion

SAT Seminar @ Technion

Symbolic Safety and Reachability

- A transition system T = (v, Init, Tr, Bad)
- T is UNSAFE if and only if there exists a number N s.t.

$$Init(v_0) \land \left(\bigwedge_{i=0}^{N-1} Tr(v_i, v_{i+1}) \right) \land Bad(v_N) \nRightarrow \bot$$

- T is SAFE if and only if there exists a safe inductive invariant Inv s.t.

$$Init \Longrightarrow Inv$$

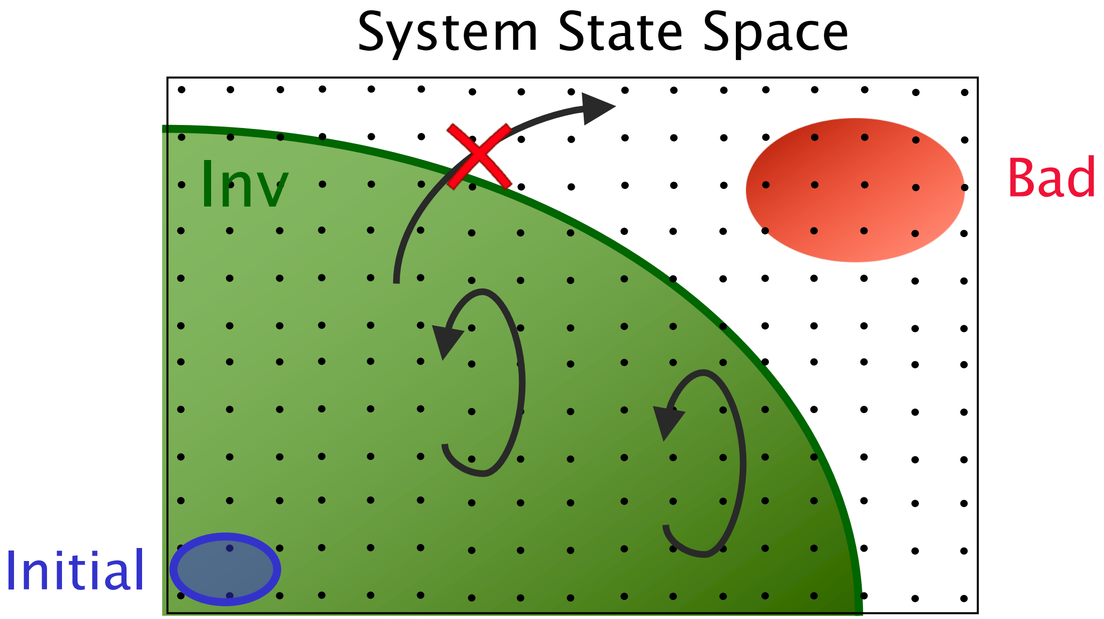
$$Inv(v) \land Tr(v, v') \Longrightarrow Inv(v')$$

Inductive

$$Inv \Longrightarrow \neg Bad$$

Safe

Inductive Invariants



System T is safe iff there exists an **inductive invariant** Inv

- Initiation Initial ⊆ Inv
- Safety Inv ∩ Bad = ∅
- Consecution TR(Inv) ⊆ Inv

i.e., if s ∈ Inv and s⇝t then t ∈ Inv

Agenda

Generalization in the context of propositional logic

- Using k-induction, interpolants and PDR

Notations

- $Tr_l^k \equiv \wedge_{i=l}^{k-1} Tr(v_i, v_{i+1})$ where $0 \leq \mathrm{l} < \mathrm{k}$
- $Tr[\varphi]^k \equiv \wedge_{i=0}^{k-1} (\varphi \wedge Tr(v_i, v_{i+1}))$

Inductive trace $\overrightarrow{F}$

A sequence of state formulas called frames

$Init$ | F_1 | .... | F_{N-1} | F_N

Properties of Traces:

- Inductive: $F_i \land Tr \rightarrow F'_{i+1}$
- Safe: $\forall i\ F_i \rightarrow \neg Bad$
- Closed: $\exists i\ F_i \rightarrow \bigvee_j^{i-1} F_j$

Extending a Trace

Add a new *safe* frame

$Init$	F_1	....	F_{N-1}	F_N	
$Init$	F_1	....	F^*_{N-1}	F^*_N	F_{N+1}

Extending a trace strengthens ***some*** but not all of the previous frames

Generating Inductive Invariants

1. N=1
2. Check if there is a counterexample of length N
   1. If it exists – return UNSAFE
   2. Otherwise, go to 3
3. Construct a safe trace $\vec{\boldsymbol{F}}$ of length N
4. If $\vec{\boldsymbol{F}}$ is closed – return SAFE
5. Increment N and go to step 2

The trace is a proof that no counterexample of length N exists

Closed means: $\exists i\ F_i \rightarrow \bigvee_j^{i-1} F_j$
In fact, $\bigvee_j^{i-1} F_j$ is the inductive invariant

Using k-Induction for Generalization

Joint work with Hari Govind V K, Vijay Ganesh, Arie Gurfinkel

(Interpolating Strong Induction @ CAV 2019)

Induction and k-induction

Induction Principle

$$Init(v_0) \rightarrow Inv(v_0)$$

$$Tr[Inv]^1 \rightarrow Inv(v_1)$$

$$Inv(v) \rightarrow \neg Bad(v)$$

k-Induction Principle (”Strong Induction”)

$$Init(v_0) \wedge Tr^{k-1} \rightarrow \bigwedge_{i=0}^{k-1} Inv(v_i)$$

$$Tr[Inv]^k \rightarrow Inv(v_k)$$

$$Inv(v) \rightarrow \neg Bad(v)$$

Relative k-induction:

$$Init(v_0) \wedge Tr^{k-1} \rightarrow \bigwedge_{i=0}^{k-1} Inv(v_i)$$

$$Tr[\phi \wedge Inv]^k \rightarrow Inv(v_k)$$

Induction and k-induction

Induction Principle

$Init(v_0) \rightarrow Inv(v_0)$

$Tr[Inv]^1 \rightarrow Inv(v_1)$

$Inv(v_0) \wedge Tr \rightarrow Inv(v_1)$

$\rightarrow \neg Bad(v)$

k-Induc

$Inv(v_0) \wedge Tr \wedge Inv(v_1) \wedge Tr \wedge \cdots \wedge Inv(v_{k-1}) \wedge Tr \rightarrow Inv(v_k)$

$Init(v_0) \wedge Tr^{k-1}$ (v_i)

$Inv(v) \rightarrow \neg Bad(v)$

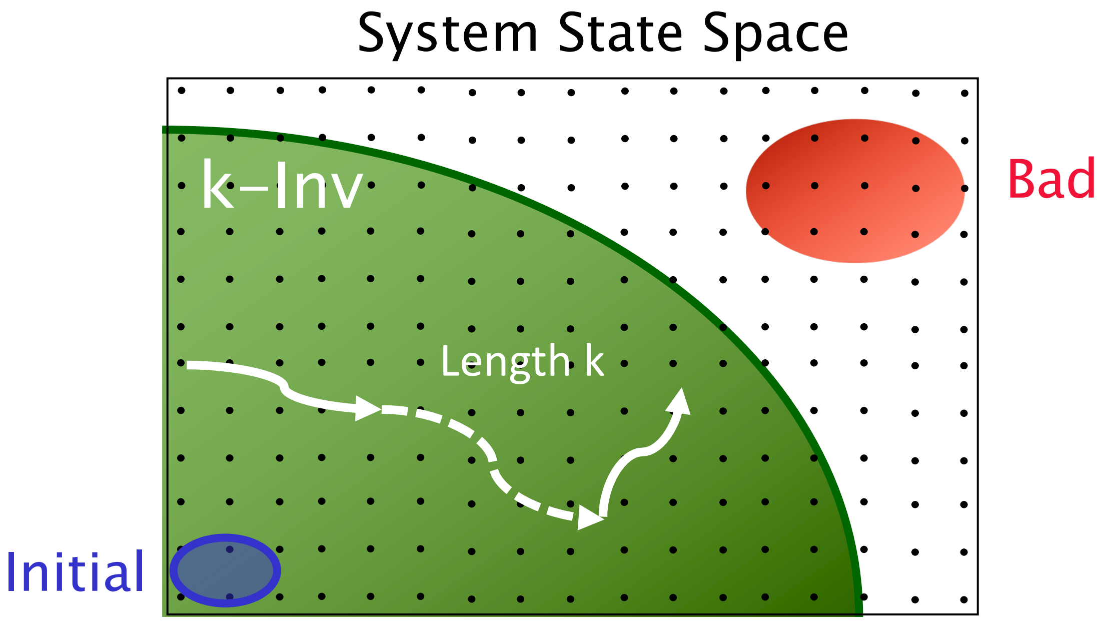
$Tr[Inv]^k \rightarrow Inv(v_k)$

Relative k-induction:

$Init(v_0) \wedge Tr^{k-1} \rightarrow \bigwedge_{i=0}^{k-1} Inv(v_i)$

$Tr[\phi \wedge Inv]^k \rightarrow Inv(v_k)$

k-Inductive Invariants



Example

Counter counts up to 64 and resets

No overflow

$(c < 66)$ is 2-inductive

$(c \neq 65 \land c < 66)$ is a 1-inductive strengthening

```
reg [7:0] c = 0;
always
if(c == 64)
      c <= 0;
else
      c <= c + 1;
end
assert property(c < 66);
```

Example: Possible Lemmas

Generate a predecessor for $c >= 66$:
$c = 65$

Block $c = 65$. Using one of:

$c = 1$
$c < 2$
…
$c < 64$

$c \neq 65$

```
reg [7:0] c = 0;
always
if(c == 64)
      c <= 0;
else
      c <= c + 1;
end
assert property(c < 66);
```

Example: Right Lemma

This work is on how strong induction can guide learning of the right inductive strengthening:

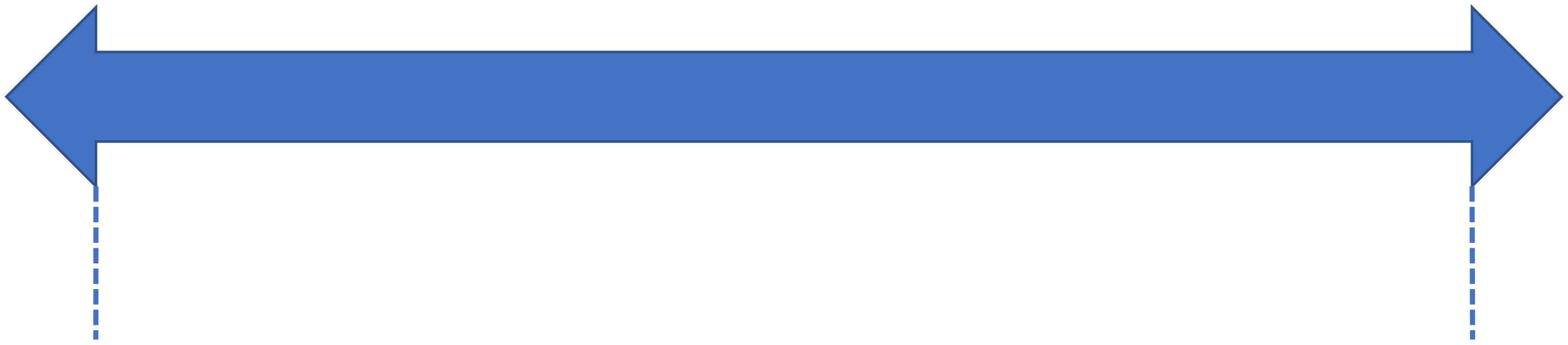
$$c \neq 65$$

```
reg [7:0] c = 0;
always
if(c == 64)
      c <= 0;
else
      c <= c + 1;
end
assert property(c < 66);
```

Why Strong Induction?

- More concise than induction
  - There are systems where k-inductive invariants are exponentially smaller than inductive invariants
- Complete for loop free paths
  - The property itself is k-inductive over loop free paths
  - No need of searching for strengthening

Intuition



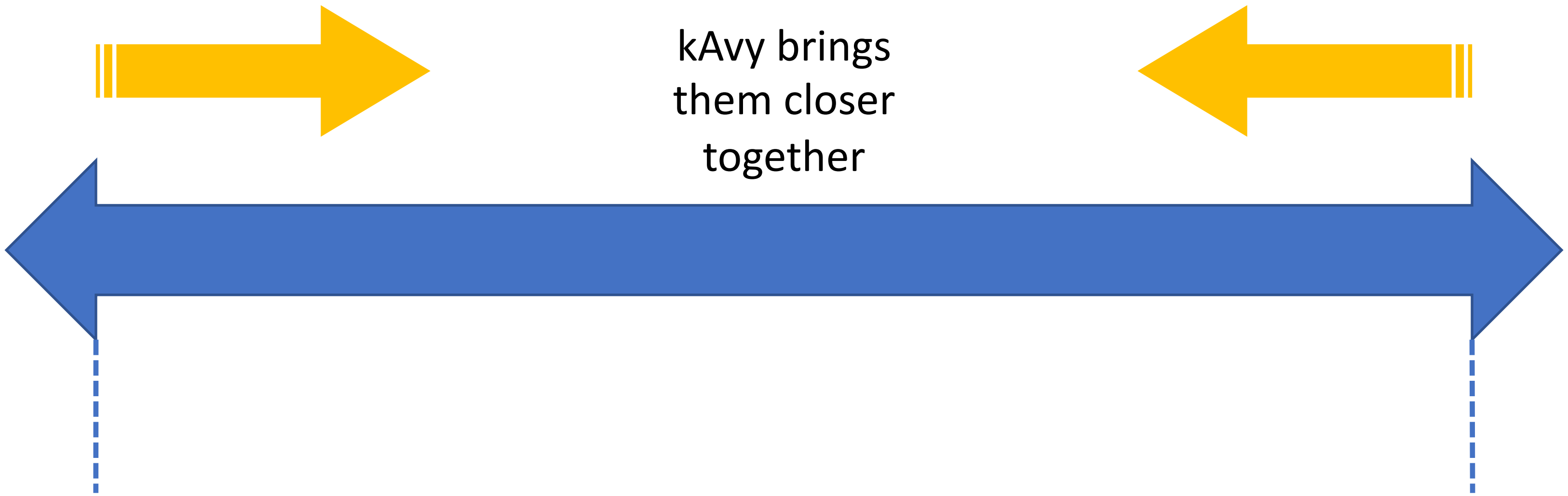
Inductive Invariants:
Given a property P, look for a **strengthening** of P, s.t. the strengthening is an **inductive invariant**:

- Start from P and iteratively strengthen

k-Induction:
Given a property P, look for a **k** s.t. P is **k-inductive**:

- Start from k=1 and iteratively increase k till P becomes k-inductive

Intuition



Inductive Invariants:
Given a property P, look for a **strengthening** of P, s.t. the strengthening is an **inductive invariant**:

- Start from P and iteratively strengthen

k-Induction:
Given a property P, look for a **k** s.t. P is **k-inductive**:

- Start from k=1 and iteratively increase k till P becomes k-inductive

Why not just strong induction?

- Properties are rarely k-inductive for small k
  - E.g., Require extra supporting invariants
- Loop free constraints prevent scalability

$$\forall i,j \quad i \neq j \rightarrow \neg \bigwedge_{var \in v} var_i = var_j$$

- No known effective way to generalize k-inductive invariants in hardware
- Verifying k-inductive invariants as hard as generating 1-inductive invariants

Avy

$Init$	F_1	....	F_{N-1}	F_N

Find a maximum i such that the following formula is UnSAT:

$$(F_i(v_i) \wedge Tr) \wedge (F_{i+1}(v_{i+1}) \wedge Tr) \wedge \cdots \wedge (F_N(v_N) \wedge Tr) \wedge \neg Bad(v_{N+1})$$

We call such an i the extension level of $\vec{\boldsymbol{F}}$

Avy

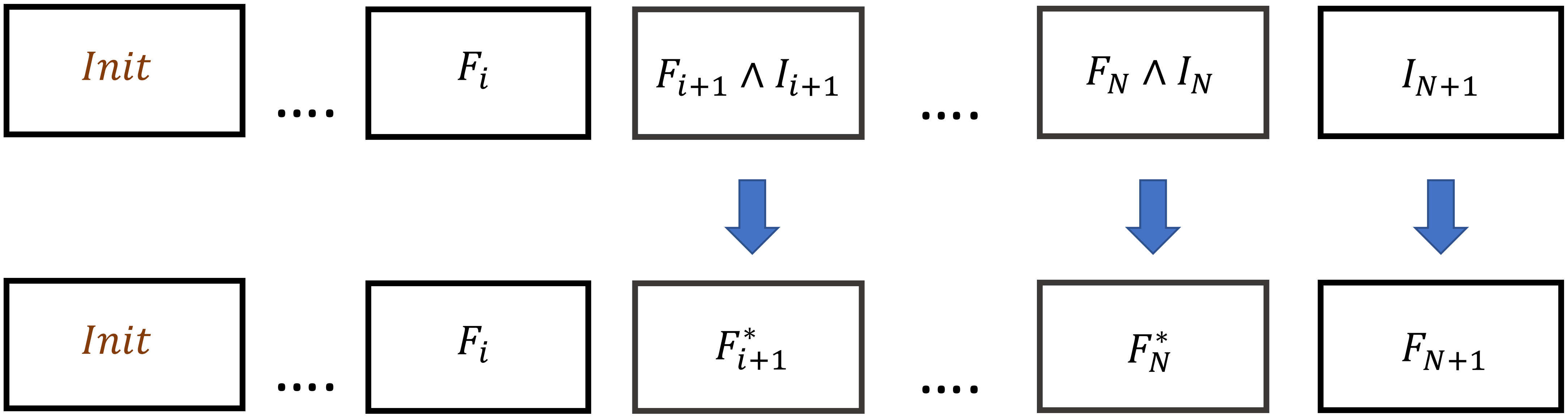
$Init$ F_1 .... F_{N-1} F_N

Extract a sequence interpolant:

$$(F_i(v_i) \land Tr) \land (F_{i+1}(v_{i+1}) \land Tr) \land \quad \ldots \quad \land (F_N(v_N) \land Tr) \land Bad(v_{N+1})$$

I_{i+1} I_{i+2} ... I_N I_{N+1}

Avy



The Limitation of the Extension Level

- Find a maximum i such that the following formula is UnSAT:

$$(F_i(v_i) \wedge Tr) \wedge (F_{i+1}(v_{i+1}) \wedge Tr) \wedge \cdots \wedge (F_N(v_N) \wedge Tr) \wedge Bad(v_{N+1})$$

- This is a one dimensional search
  - k is fixed to be 1

- The search for an inductive invariant is “limited” by k=1

A Strong Extension Level

- Assume the following is UnSAT:

$$(F_i(v_i) \wedge Tr) \wedge (F_{i+1}(v_{i+1}) \wedge Tr) \wedge \cdots \wedge (F_N(v_N) \wedge Tr) \wedge Bad(v_{N+1})$$

- Try to weaken the *initial* assumption (here F_i):

$$(F_{i+1}(v_i) \wedge Tr) \wedge (F_{i+1}(v_{i+1}) \wedge Tr) \wedge \cdots \wedge (F_N(v_N) \wedge Tr) \wedge Bad(v_{N+1})$$

A Strong Extension Level

- We can continue this process
- However, assume the formula becomes satisfiable

$$(F_{i+2}(v_i) \wedge Tr) \wedge (F_{i+2}(v_{i+1}) \wedge Tr) \wedge (F_{i+2}(v_{i+2}) \wedge Tr) \wedge \cdots \wedge (F_N(v_N) \wedge Tr) \wedge Bad(v_{N+1})$$

- Can try and increase k

$$(F_{i+2}(v_{i-1}) \wedge Tr) \wedge (F_{i+2}(v_i) \wedge Tr) \wedge (F_{i+2}(v_{i+1}) \wedge Tr) \wedge (F_{i+2}(v_{i+2}) \wedge Tr) \wedge \cdots \wedge (F_N(v_N) \wedge Tr) \wedge Bad(v_{N+1})$$

k=4

A Strong Extension Level

- A *strong extension level* (SEL) is a pair (i, k) s.t. the following formula is UnSAT:

$$Tr[F_i]_{i-k+1}^{i+1} \wedge (F_{i+1}(v_{i+1}) \wedge Tr) \wedge (F_{i+2}(v_{i+2}) \wedge Tr) \wedge \cdots \wedge (F_N(v_N) \wedge Tr) \wedge Bad(v_{N+1})$$

- Try to find an optimal SEL
- Optimal = maximum possible i, with minimal k

kAvy

$Init$ | F_1 | .... | F_{N-1} | F_N

Find a maximum (i, k) such that the following formula is UnSAT:

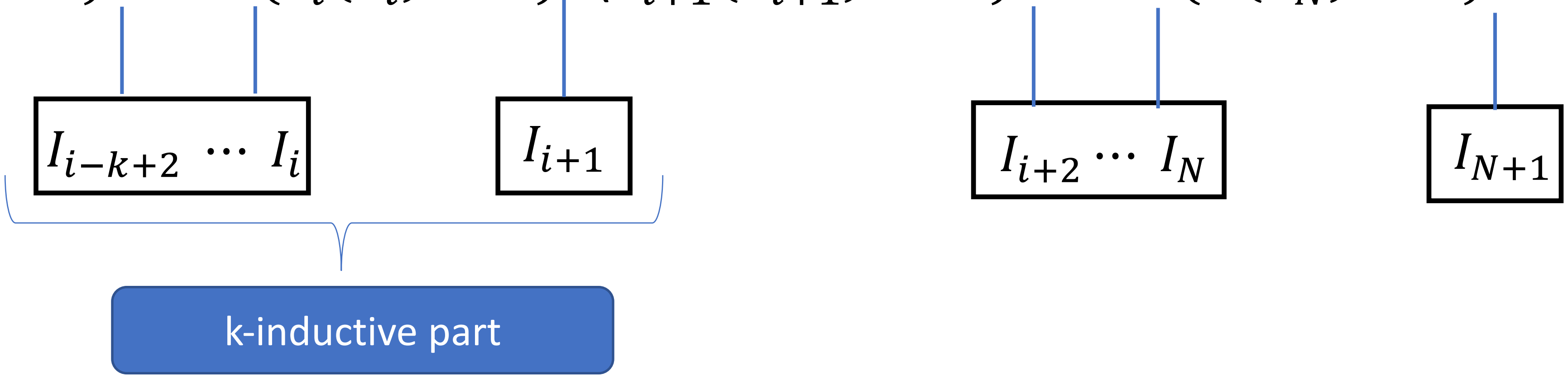
$$(F_i(v_{i-k+1}) \wedge Tr) \wedge \cdots \wedge (F_i(v_i) \wedge Tr) \wedge (F_{i+1}(v_{i+1}) \wedge Tr) \wedge \cdots \wedge (F(v_N) \wedge Tr) \wedge Bad(v_{N+1})$$

kAvy

$Init$ | F_1 | .... | F_{N-1} | F_N

Extract a sequence interpolant:

$$(F_i(v_{i-k+1}) \wedge Tr) \wedge \cdots \wedge (F_i(v_i) \wedge Tr) \wedge (F_{i+1}(v_{i+1}) \wedge Tr) \wedge \cdots \wedge (F(v_N) \wedge Tr) \wedge Bad(v_{N+1})$$



kAvy

$Init$ .... F_{i-k+1}

$F_{i-k+2} \wedge I_{i-k+2}$ ... $F_i \wedge \bigwedge_{j=i-k+2}^{i} I_j$ $F_{i+1} \wedge \bigwedge_{j=i-k+2}^{i+1} I_j$

The 1-inductive proof of k-induction

$F_{i+2} \wedge I_{i+2}$ ... $F_{N-1} \wedge I_{N-1}$ I_{N+1}

A Top-Down Approach for Finding Optimal SEL

- Start from the maximal i and k
- When UnSAT, decrease k
- When SAT decrease i

$$(F_N(v_0) \wedge Tr) \wedge (F_N(v_1) \wedge Tr) \wedge \cdots \wedge (F_N(v_N) \wedge Tr) \wedge Bad(v_{N+1})$$

$$(F_{N-1}(v_0) \wedge Tr) \wedge (F_{N-1}(v_1) \wedge Tr) \wedge \cdots \wedge (F_{N-1}(v_{N-1}) \wedge Tr) \wedge (F_N(v_N) \wedge Tr) \wedge Bad(v_{N+1})$$

$$Tr[F_{N-2}]^{N-1} \wedge (F_{N-1}(v_{N-1}) \wedge Tr) \wedge \cdots \wedge (F_N(v_N) \wedge Tr) \wedge Bad(v_{N+1})$$

A Top-Down Approach for Finding Optimal SEL

- Start from the maximal i and k
- When UnSAT, decrease k
- When SAT decrease i

$$(F_{N-1}(v_0) \land Tr) \land (F_{N-1}(v_1) \land Tr) \land \cdots \land (F_{N-1}(v_{N-1}) \land Tr) \land (F_N(v_N) \land Tr) \land Bad(v_{N+1})$$

$$(F_{N-1}(v_1) \land Tr) \land \cdots \land (F_{N-1}(v_{N-1}) \land Tr) \land (F_N(v_N) \land Tr) \land Bad(v_{N+1})$$

$$(F_{N-1}(v_2) \land Tr) \land \cdots \land (F_{N-1}(v_{N-1}) \land Tr) \land (F_N(v_N) \land Tr) \land Bad(v_{N+1})$$

A Top-Down Approach for Finding Optimal SEL

- More formally, the optimal SEL (i, k):

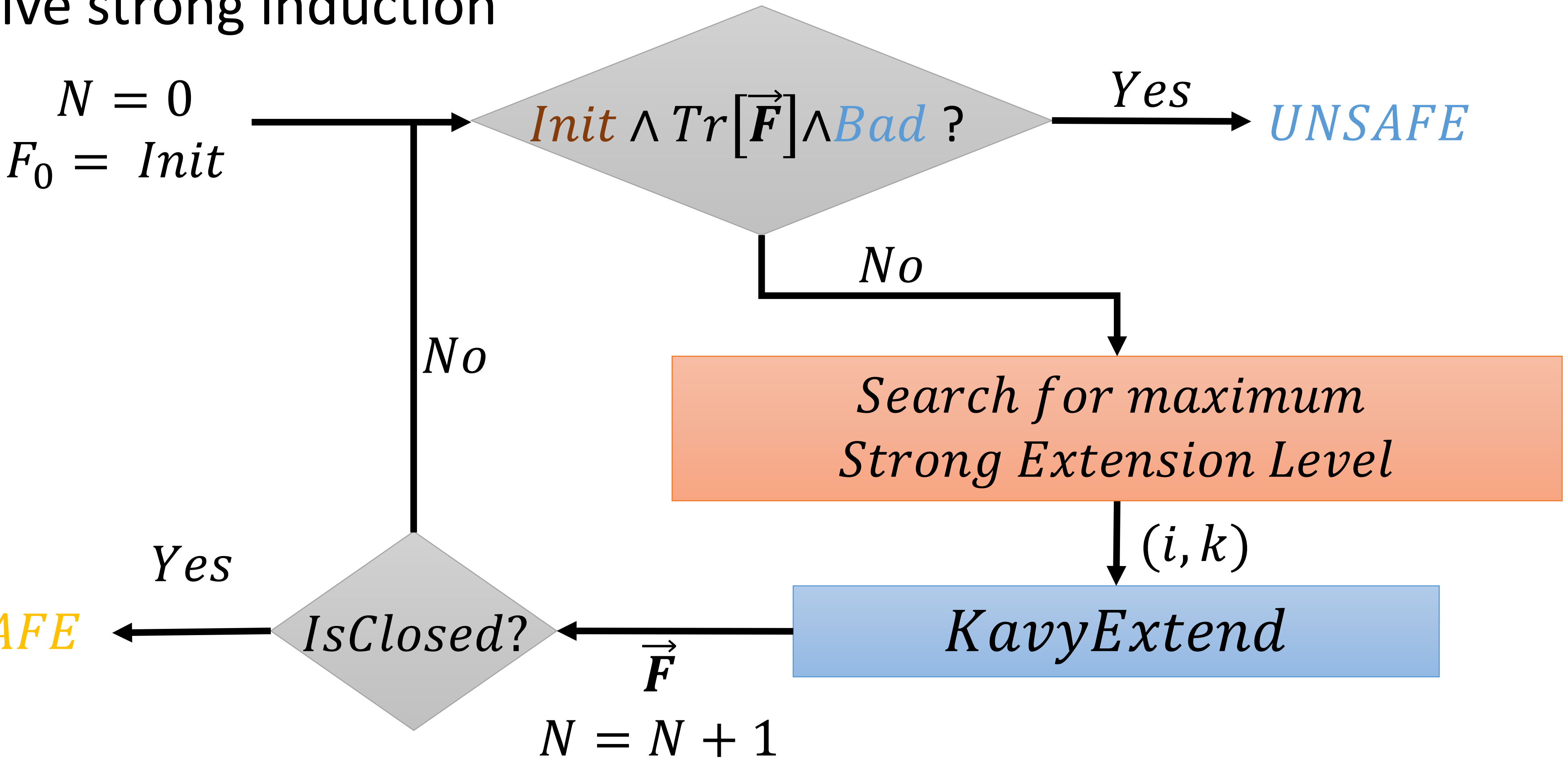
$$i \coloneqq max\left\{j \mid 0 \leq j \leq N \cdot Tr\left[F_j\right]^{j+1} \wedge (F_{j+1}\left(v_{j+1}\right) \wedge Tr) \dots \wedge (F_N(v_N) \wedge Tr) \wedge Bad(v_{N+1})\right\}$$

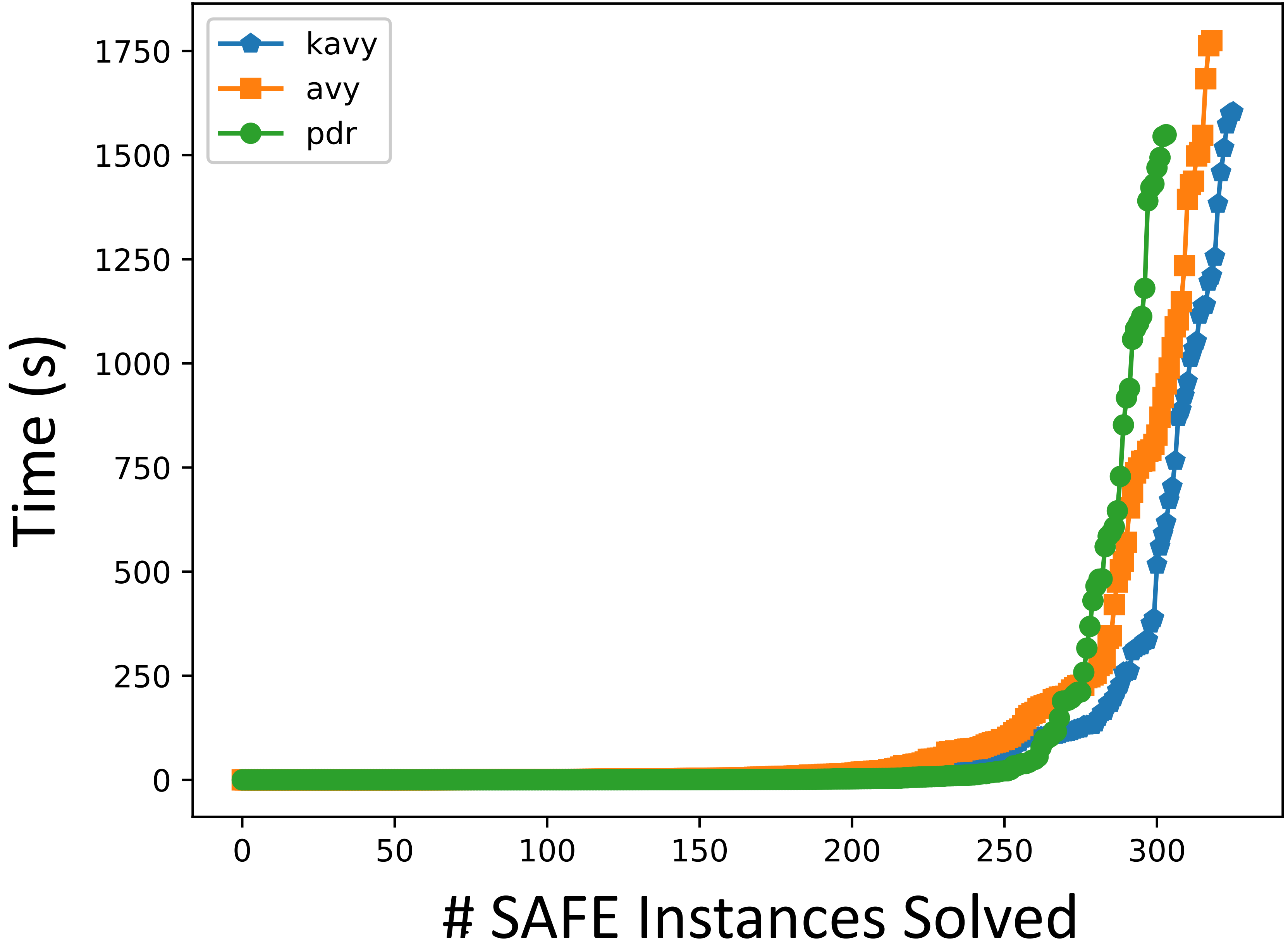
$$k \coloneqq min\{l \mid 0 \leq l \leq (i+1) \cdot Tr[F_i]_{i-l+1}^{i+1} \wedge (F_{i+1}(v_{i+1}) \wedge Tr) \dots \wedge (F_N(v_N) \wedge Tr) \wedge Bad(v_{N+1})\}$$

- Can be done by minimizing a proof of unsatisfiability

kAvy

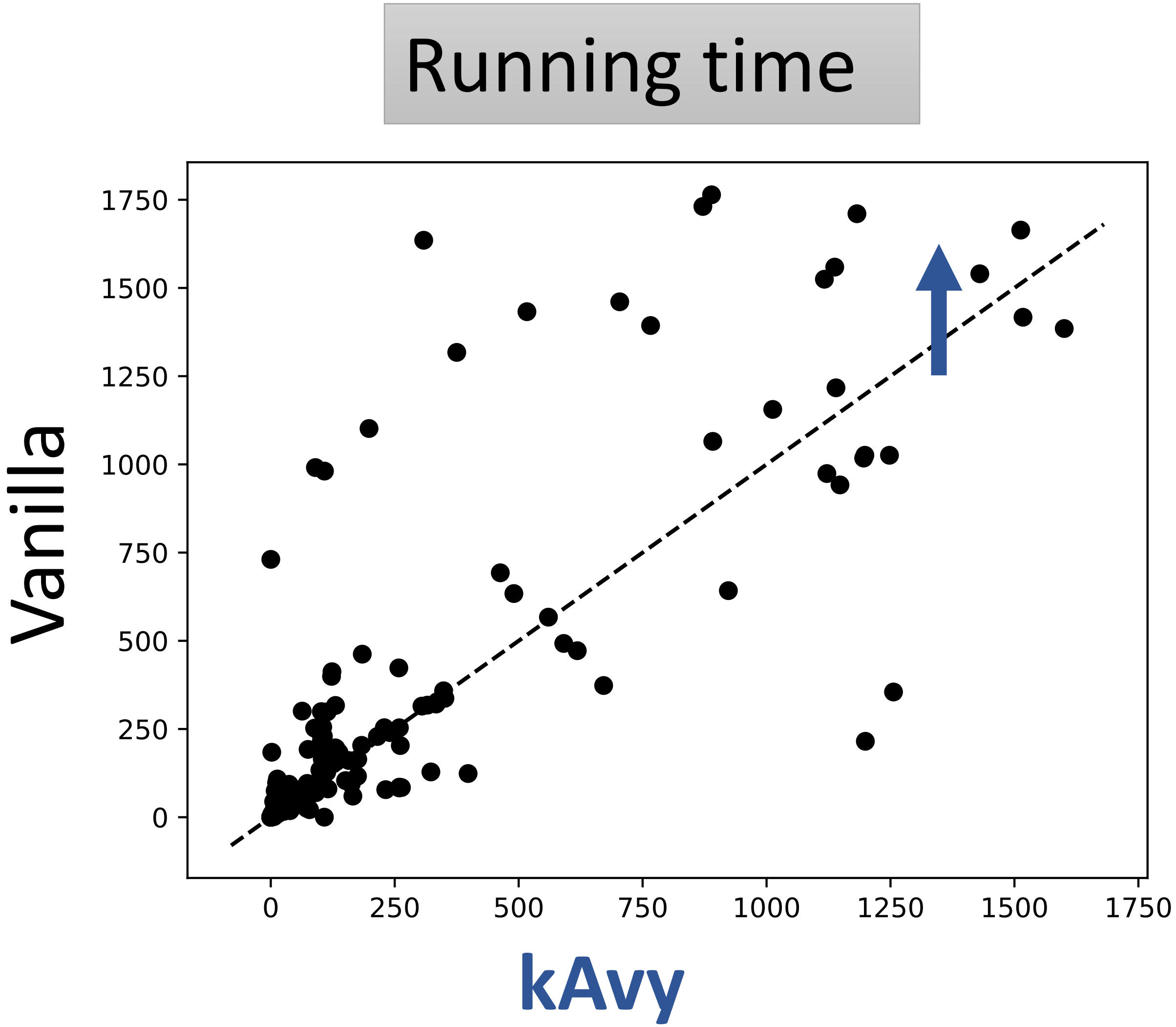
Avy style interpolation-guided extension of inductive trace using relative strong induction

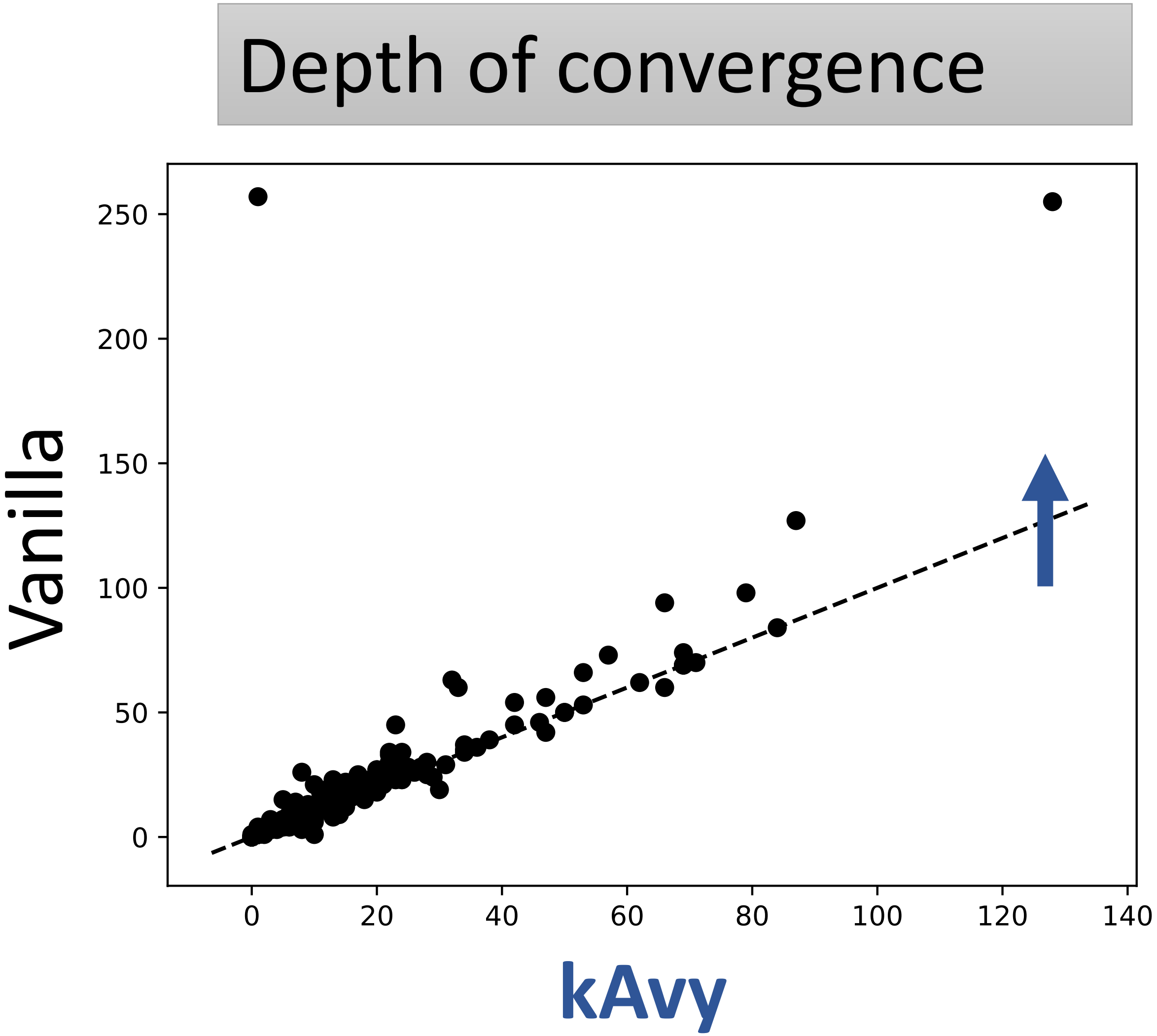




kavy
avy
pdr
1750
1500
1250
1000
750
500
250
0
Time (s)
0
50
100
150
200
250
300
SAFE Instances Solved

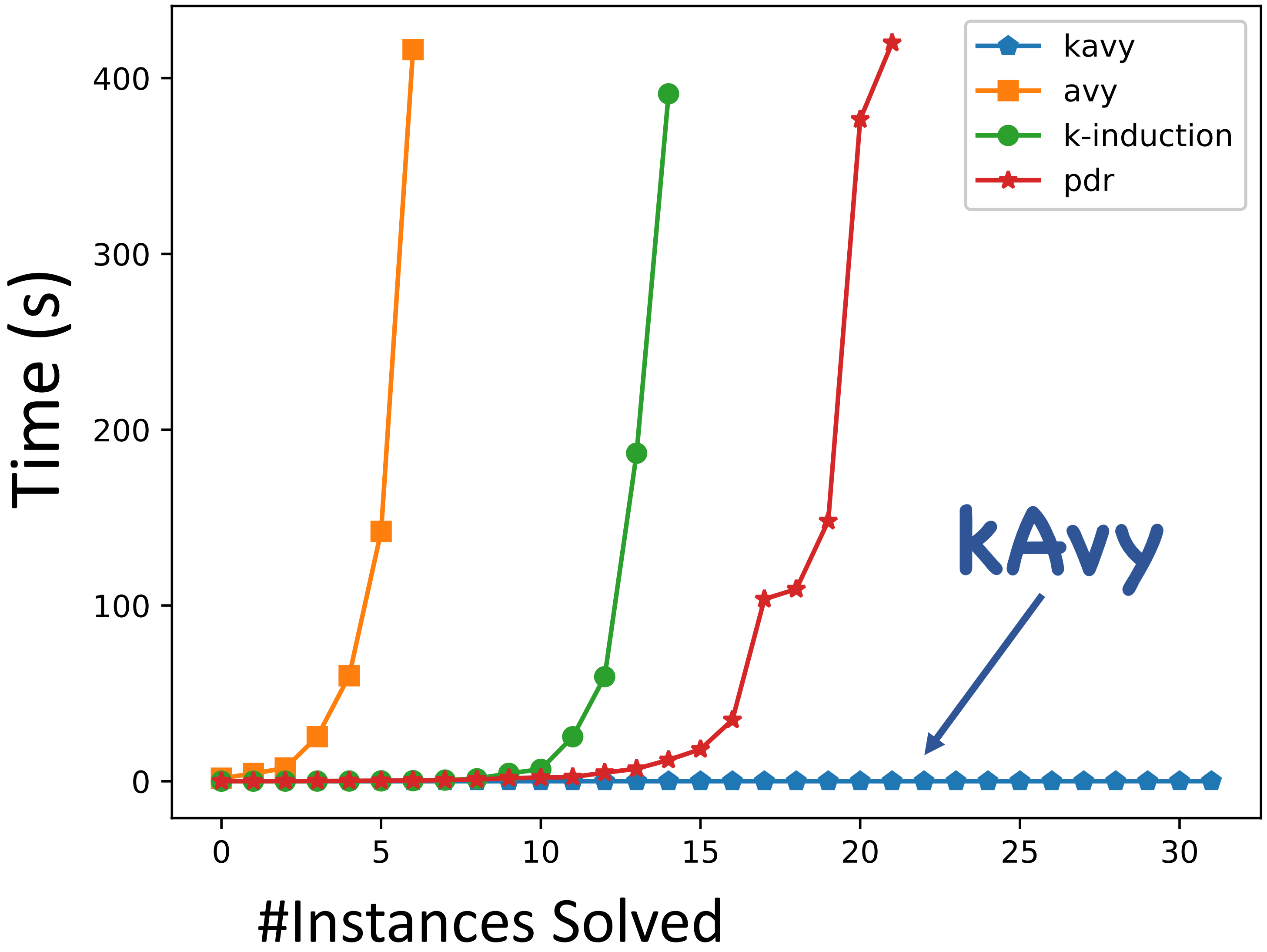
kAvy vs kAvy-Vanilla (1-induction)





Points above the diagonal are better for **kAvy**

On *shift* instances



Instances from HWMCC:

- HWMCC’ 15
  - shift1add2048.aig
  - shift1add256.aig
  - shift1add262144.aig
  - shift1add512.aig
  - shift1add524288.aig
- HWMCC’ 17
  - shift1 add262144.aig