Decision Tree Learning in CEGIS-Based Termination Analysis

Satoshi Kura[1,2] Hiroshi Unno[3,4] Ichiro Hasuo[1,2]

[1]National Institute of Informatics, Tokyo, Japan

[2]The Graduate University for Advanced Studies (SOKENDAI), Kanagawa, Japan

[3]University of Tsukuba, Ibaraki, Japan

[4]RIKEN AIP, Tokyo, Japan

Termination Analysis

"Does this program terminate (or not)?"

Many methods based on ranking function synthesis

- Example-based

  [Urban et al., TACAS'16], [Fedyukovich et al., CAV'18], [Gonnord et al., PLDI'15]

- ω-automata-based [Heizmann et al., CAV'14]
- Abstract interpretation [Urban, SAS'13]

Our Contribution

We propose an **example-based** termination analyzer.

Specifically, we use ...

- CEGIS architecture [Solar-Lezama et al., ASPLOS'06]
- **transition examples** collected by SMT solvers
- **decision tree learning** to obtain piecewise affine ranking functions.

We obtained promising experimental results.

Comparison with Other Example-Based Methods

- trace examples (collected by safety verifiers)

  [Urban et al., TACAS'16], [Fedyukovich et al., CAV'18]

- transition examples (collected by SMT solvers)
  - affine ranking function

    [Gonnord et al., PLDI'15]
  - piecewise affine ranking function

    our work

Decision Tree Learning in CEGIS

- for invariant synthesis (state examples)

  [Champion et al., TACAS'18], [Ezudheen et al., OOPSLA'18], ..

- for ranking function synthesis (transition examples)

  our work

Outline

Termination Problem and CEGIS

Our Synthesizer

Experimental Results

Ranking Function [Floyd, '67]

Let S be a state space.

Let $\tau \subseteq S \times S$ be a transition relation.

$f : S \to \mathbb{Z}$ is a **ranking function** if for each $(x, x') \in \tau$

- $f(x) \geq 0$
- $f(x) > f(x')$

If such a ranking function exists, then τ terminates.

Termination Problem to Constraint Solving

Given a program, consider the following problems.

Find a (ranking) function $f(x)$ s.t.

$$(x, x') \in \tau \implies R(x, x')$$

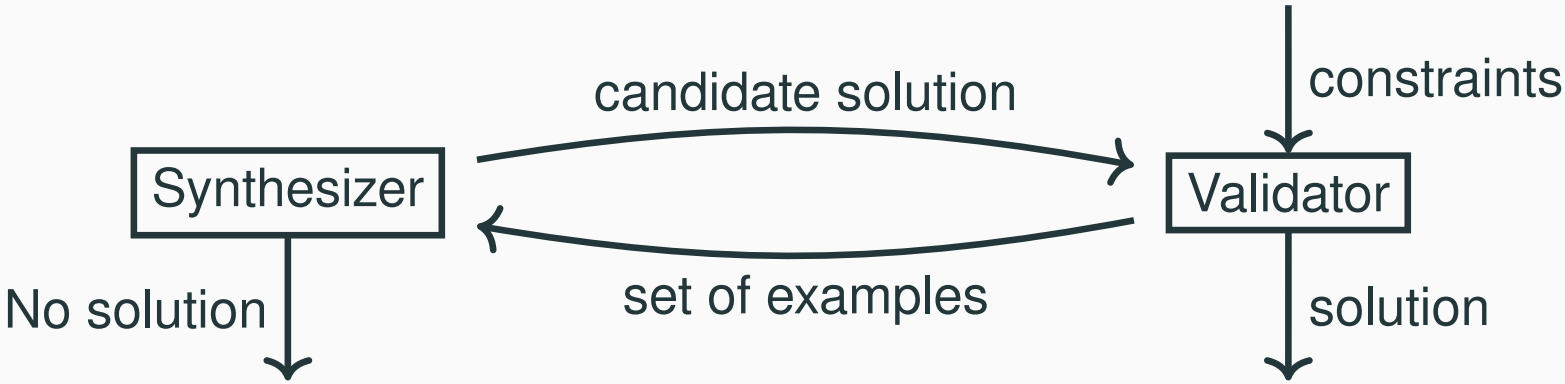
where $R(x, x') \coloneqq f(x) \geq 0 \ \wedge \ f(x) > f(x')$.

```
while (x != 0) {
if (x > 0) x--;
else x++; }
```

$$x \neq 0 \wedge x > 0 \implies R(x, x-1)$$
$$x \neq 0 \wedge x \leq 0 \implies R(x, x+1)$$

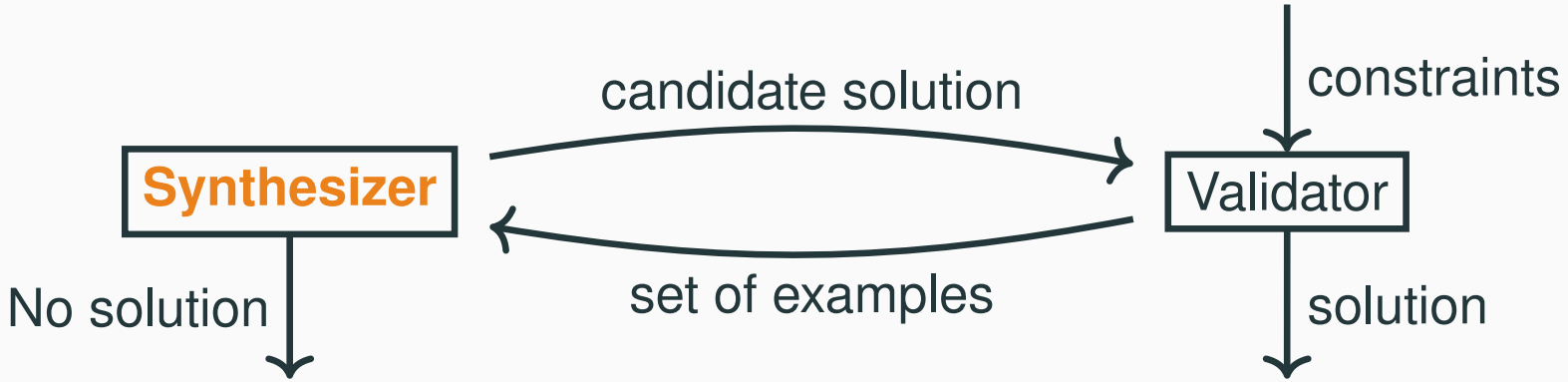
CounterExample Guided Inductive Synthesis

CEGIS [Solar-Lezama et al., ASPLOS'06]



CEGIS solves constraints via interaction between the synthesizer and the validator.

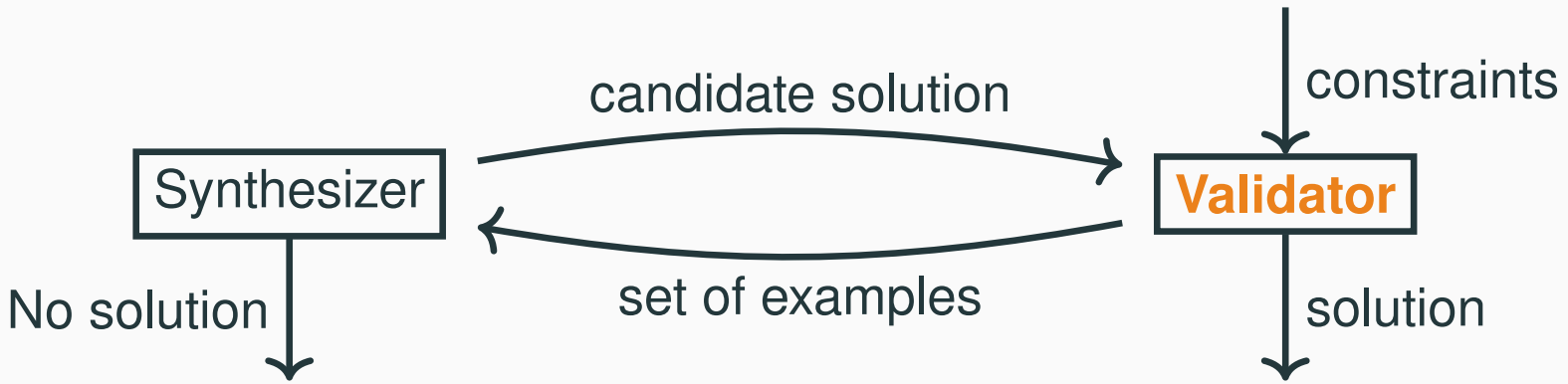
Synthesizer



Given a set of (transition) examples,

- return a candidate solution if exists
  - candidate solution = ranking function for examples
- answer “no solution” otherwise

Validator



Given a candidate solution for the examples,

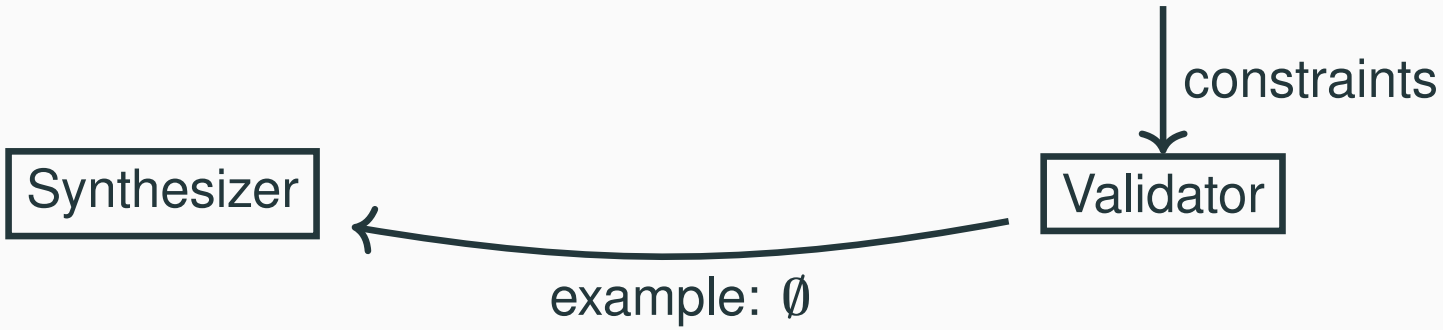
- if the candidate solution is a genuine solution, return the solution
- otherwise, add counterexamples to the set of examples using SMT solvers.

CEGIS Example

Constraints:

$$x \neq 0 \land x > 0 \implies R(x, x-1)$$
$$x \neq 0 \land x \leq 0 \implies R(x, x+1)$$

where $R(x, x') \coloneqq f(x) \geq 0 \land f(x) > f(x')$.



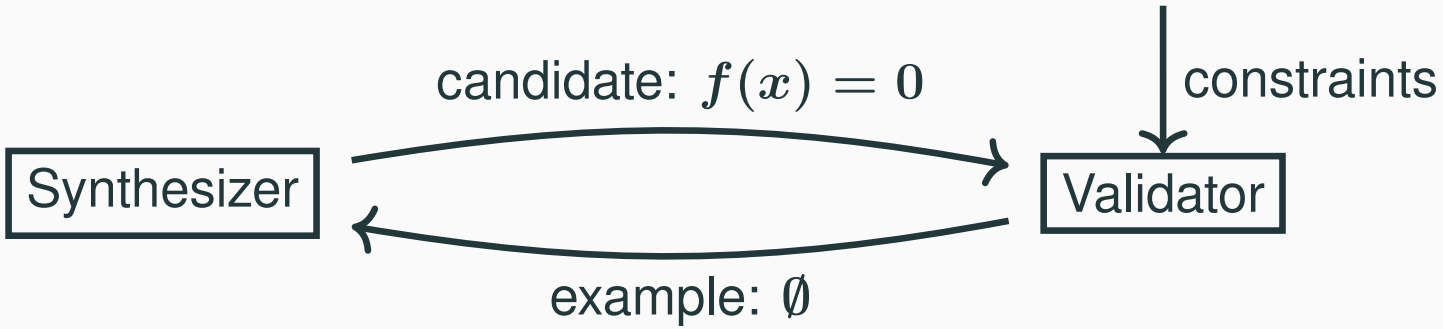
A constraint satisfaction problem is given. The set of examples is $\emptyset$.

CEGIS Example

Constraints:

$$x \neq 0 \wedge x > 0 \implies R(x, x-1)$$
$$x \neq 0 \wedge x \leq 0 \implies R(x, x+1)$$

where $R(x, x') \coloneqq f(x) \geq 0 \wedge f(x) > f(x')$.



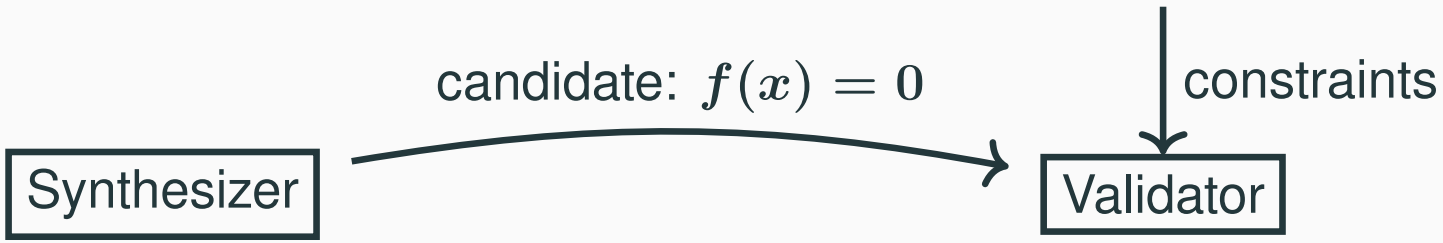
The synthesizer returns a candidate solution for $\emptyset$.

CEGIS Example

Constraints:

$$x \neq 0 \wedge x > 0 \implies R(x, x-1)$$
$$x \neq 0 \wedge x \leq 0 \implies R(x, x+1)$$

where $R(x, x') \coloneqq f(x) \geq 0 \wedge f(x) > f(x')$.



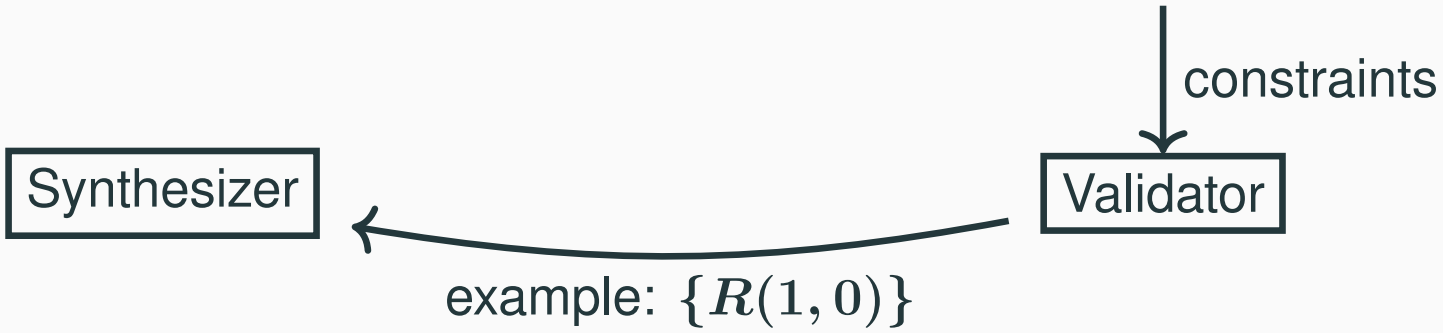
The validator finds a new counterexample $R(1, 0)$.

CEGIS Example

Constraints:

$$x \neq 0 \wedge x > 0 \implies R(x, x-1)$$
$$x \neq 0 \wedge x \leq 0 \implies R(x, x+1)$$

where $R(x, x') \coloneqq f(x) \geq 0 \wedge f(x) > f(x')$.



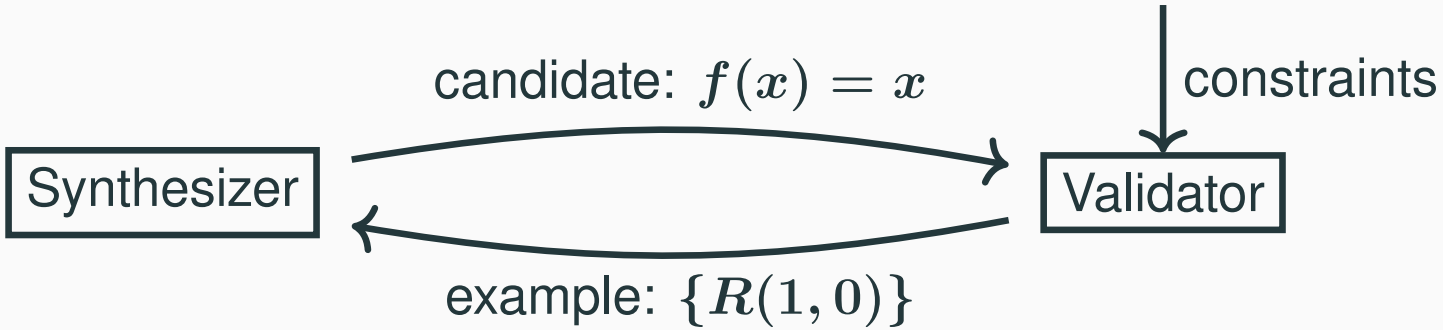
The validator finds a new counterexample $R(1, 0)$.

CEGIS Example

Constraints:

$$x \neq 0 \wedge x > 0 \implies R(x, x-1)$$
$$x \neq 0 \wedge x \leq 0 \implies R(x, x+1)$$

where $R(x, x') \coloneqq f(x) \geq 0 \wedge f(x) > f(x')$.



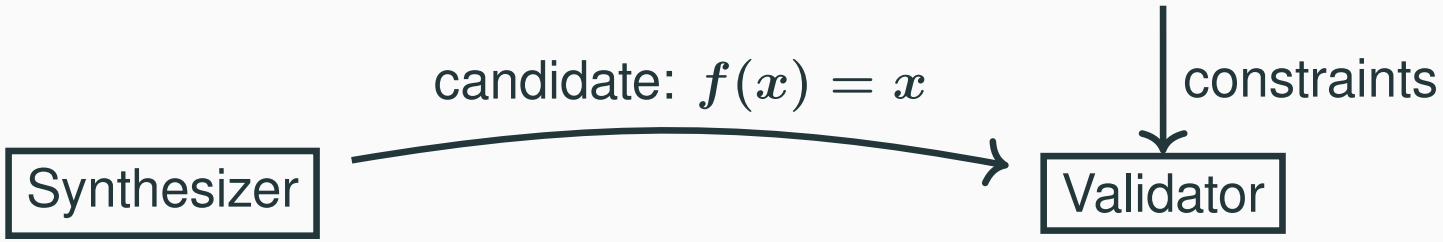
The synthesizer returns a candidate solution for $\{R(1, 0)\}$.

CEGIS Example

Constraints:

$$x \neq 0 \wedge x > 0 \implies R(x, x-1)$$
$$x \neq 0 \wedge x \leq 0 \implies R(x, x+1)$$

where $R(x, x') \coloneqq f(x) \geq 0 \wedge f(x) > f(x')$.



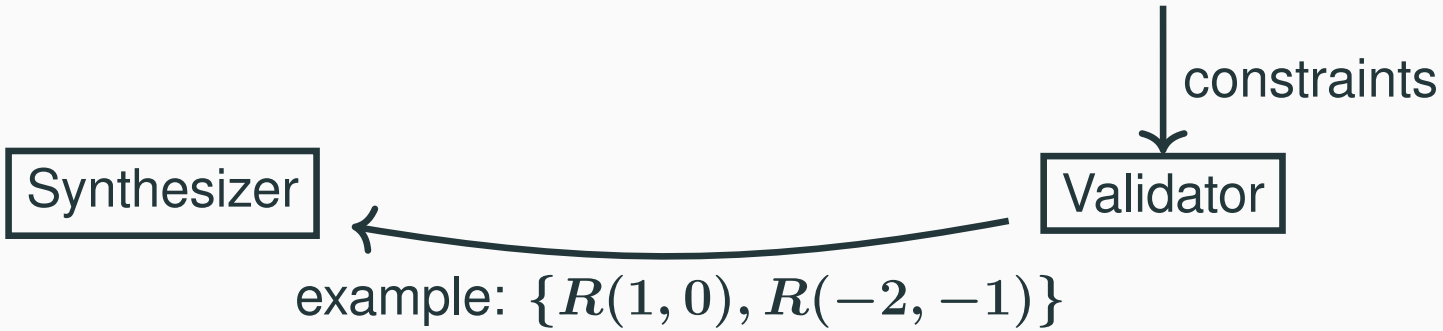
The validator finds a new counterexample $R(-2, -1)$.

CEGIS Example

Constraints:

$$x \neq 0 \land x > 0 \implies R(x, x-1)$$
$$x \neq 0 \land x \leq 0 \implies R(x, x+1)$$

where $R(x, x') \coloneqq f(x) \geq 0 \land f(x) > f(x')$.



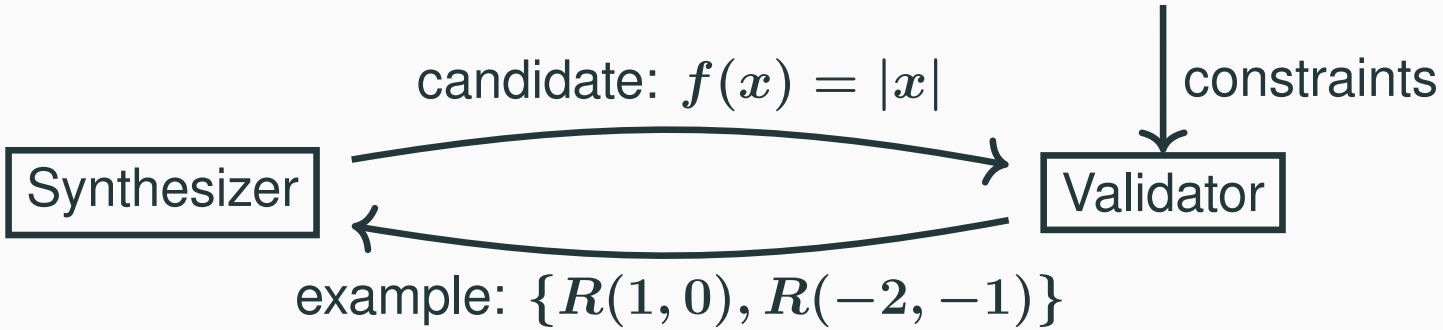
The validator finds a new counterexample $R(-2, -1)$.

CEGIS Example

Constraints:

$$x \neq 0 \wedge x > 0 \implies R(x, x-1)$$
$$x \neq 0 \wedge x \leq 0 \implies R(x, x+1)$$

where $R(x, x') \coloneqq f(x) \geq 0 \wedge f(x) > f(x')$.



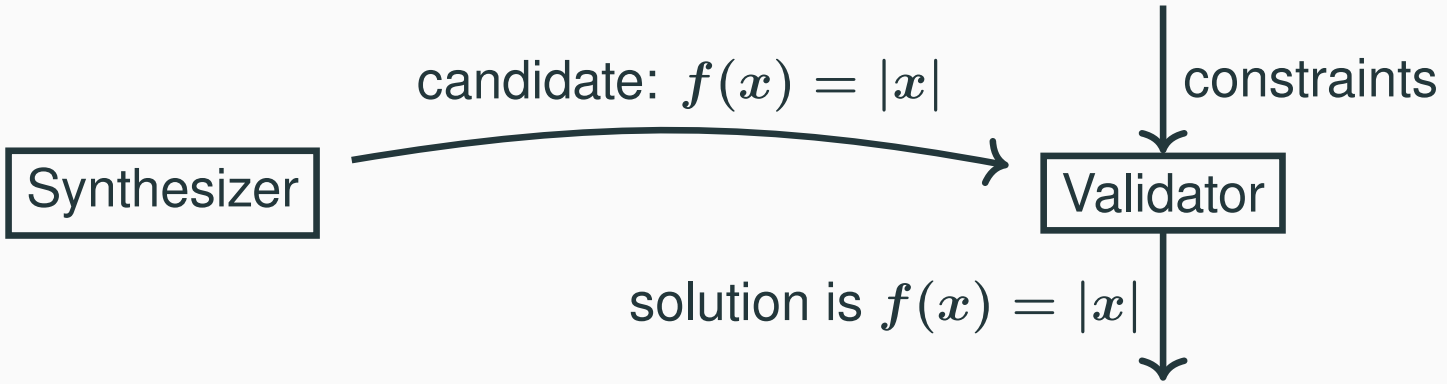
The synthesizer returns a candidate solution for $\{R(1,0), R(-2,-1)\}$.

CEGIS Example

Constraints:

$$x \neq 0 \wedge x > 0 \implies R(x, x-1)$$
$$x \neq 0 \wedge x \leq 0 \implies R(x, x+1)$$

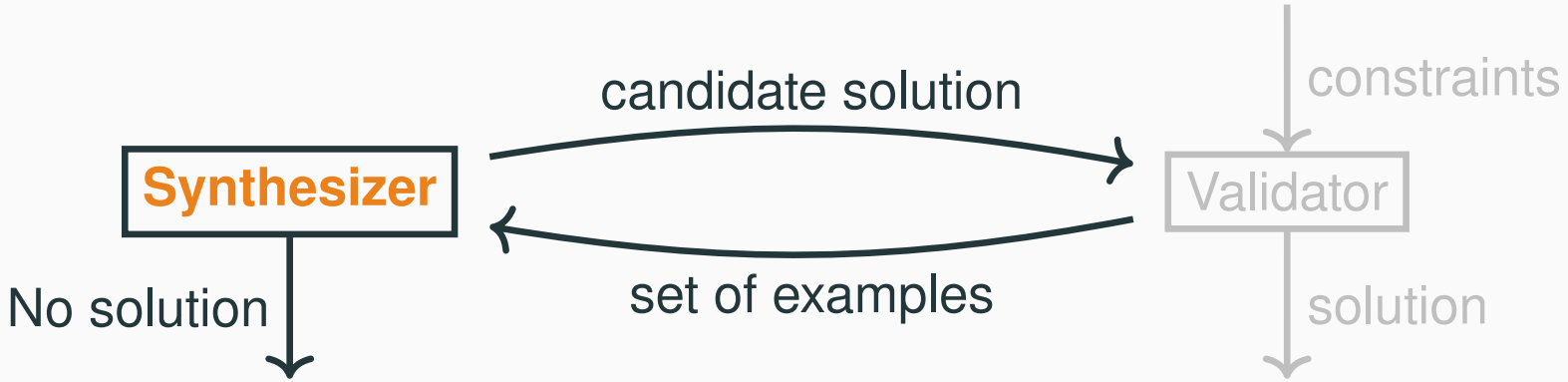
where $R(x, x') \coloneqq f(x) \geq 0 \wedge f(x) > f(x')$.



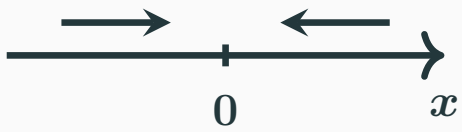
The validator confirms that $f(x) = |x|$ is a genuine solution.

Outline

Overview of Our Synthesizer



- Input: a set of transition examples
- Output: a candidate solution for the input
- Search space: piecewise affine functions

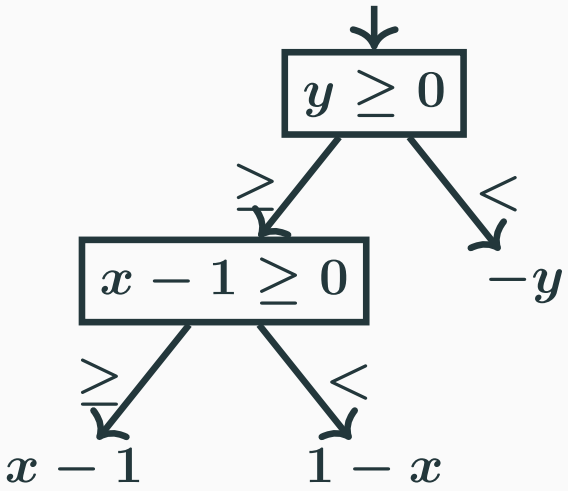


$$\mapsto \qquad f(x) = \begin{cases} x & x \geq 0 \\ -x & x < 0 \end{cases}$$

Decision Tree Representation

To represent piecewise affine ranking functions, our synthesizer uses **decision trees**.

- Node = halfspace (affine inequality)
- Leaf = affine function



$$f(x, y) = \begin{cases} x - 1 & y \geq 0 \ \wedge \ x - 1 \geq 0 \\ 1 - x & y \geq 0 \ \wedge \ x - 1 < 0 \\ -y & y < 0 \end{cases}$$

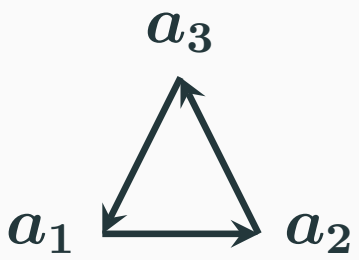
Outline of Our Algorithm

Key idea: handling cycles in transition examples

1. **Detect explicit cycles**
   - Proceed if no explicit cycle exists
2. Refine the current segmentation
   - We get a decision tree that ranks “non-crossing examples”.
3. Cut implicit cycles
   - Eventually, there is no implicit cycle.
   - We get a decision tree that ranks all examples.

Phase 1: Detect Explicit Cycles

If $\{R(a_1, a_2), R(a_2, a_3), \ldots, R(a_n, a_1), \ldots\}$ is given as examples,



this witnesses non-termination.

We call this an **explicit cycle**.

Outline of Our Algorithm

1. Detect explicit cycles
    - Proceed if no explicit cycle exists
2. **Refine the current segmentation**
    - We get a decision tree that ranks “non-crossing examples”.
3. Cut implicit cycles
    - Eventually, there is no implicit cycle.
    - We get a decision tree that ranks all examples.

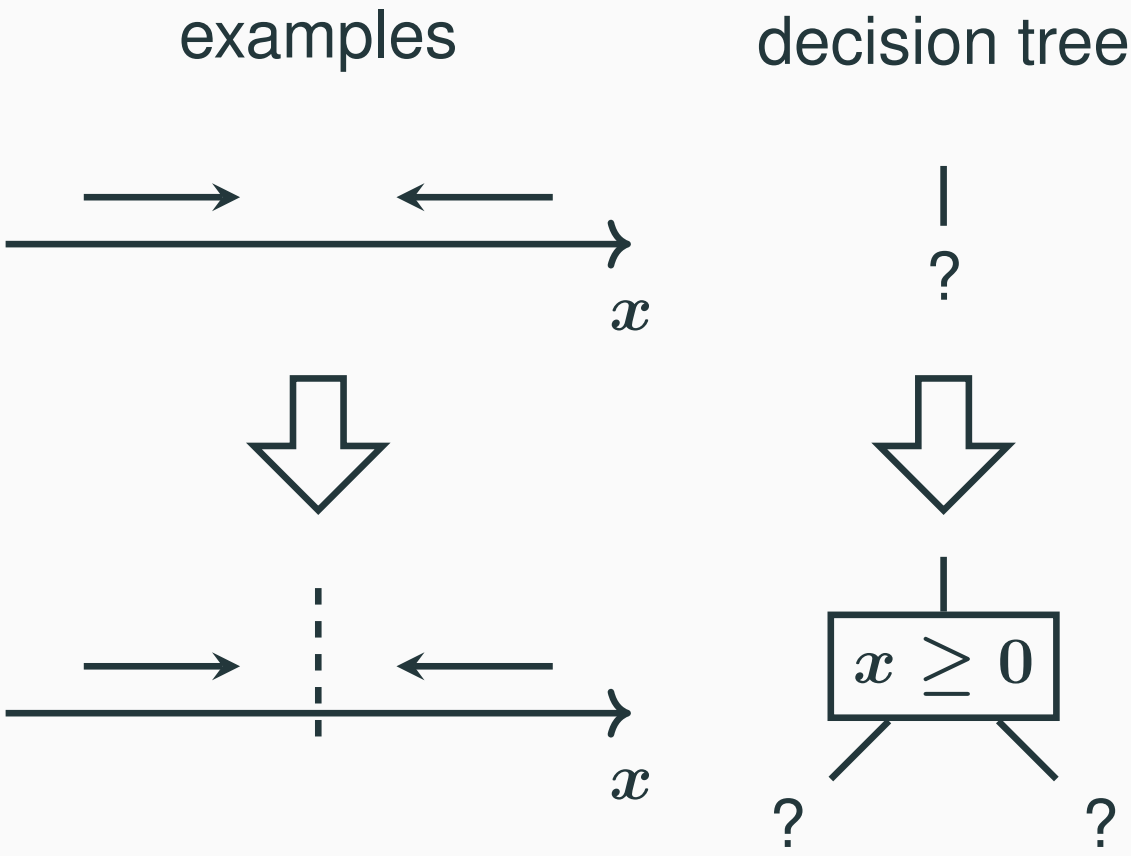
Phase 2: Refine the Current Segmentation

examples

decision tree

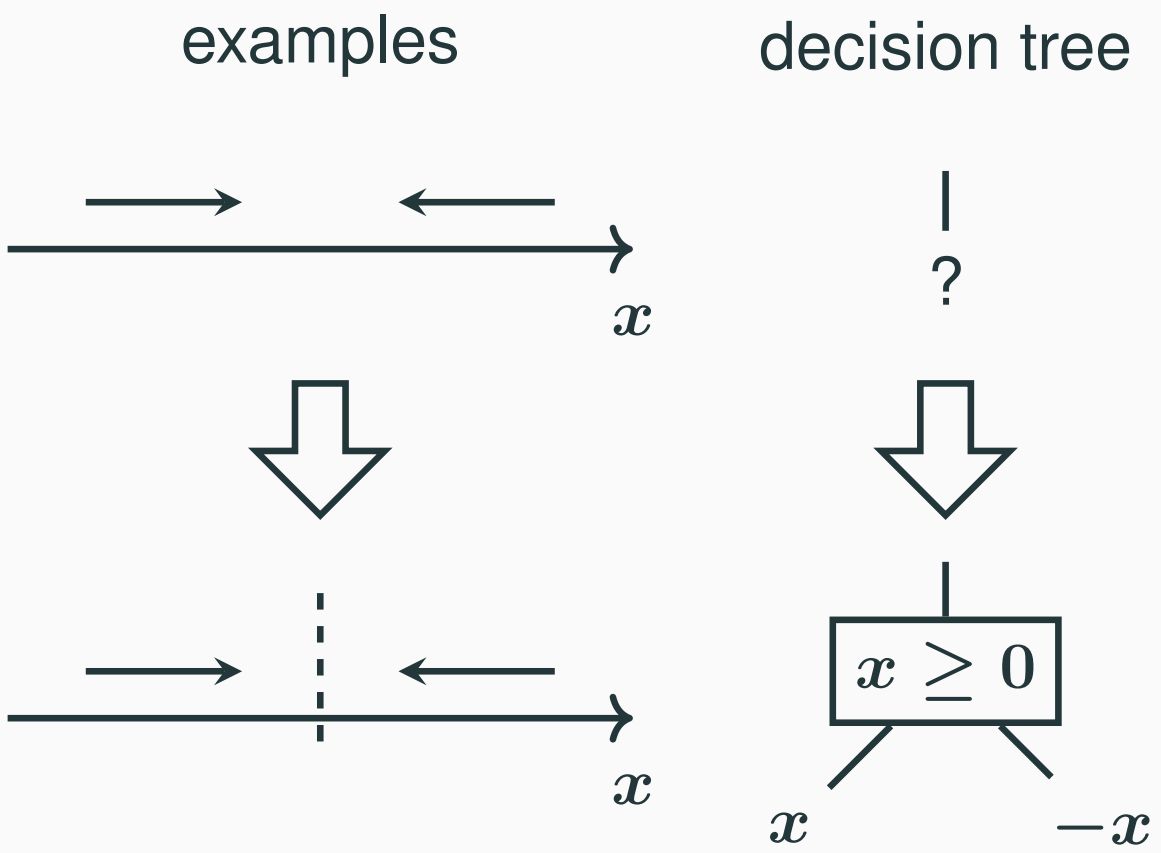
No affine ranking function

Phase 2: Refine the Current Segmentation



... until each leaf has an affine ranking function.

Phase 2: Refine the Current Segmentation



No affine ranking function

Divide state space by good halfspace

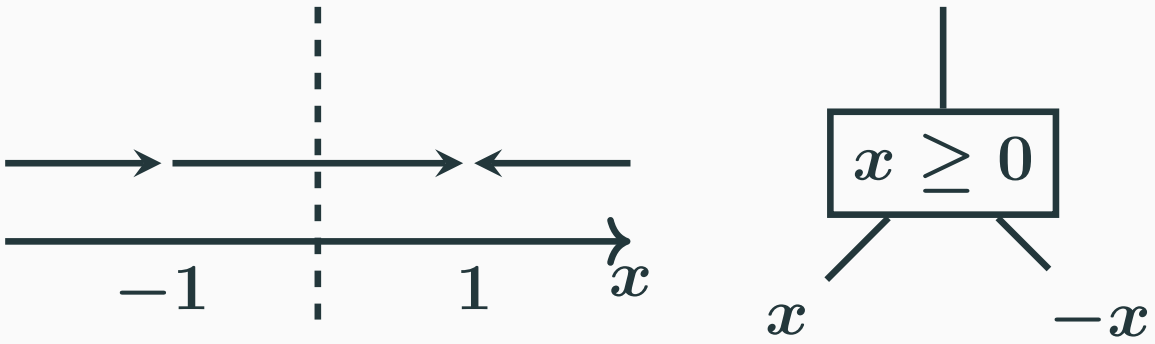
... until each leaf has an affine ranking function.

Outline of Our Algorithm

1. Detect explicit cycles
    - Proceed if no explicit cycle exists
2. Refine the current segmentation
    - We get a decision tree that ranks “non-crossing examples”.
3. **Cut implicit cycles**
    - Eventually, there is no implicit cycle.
    - We get a decision tree that ranks all examples.

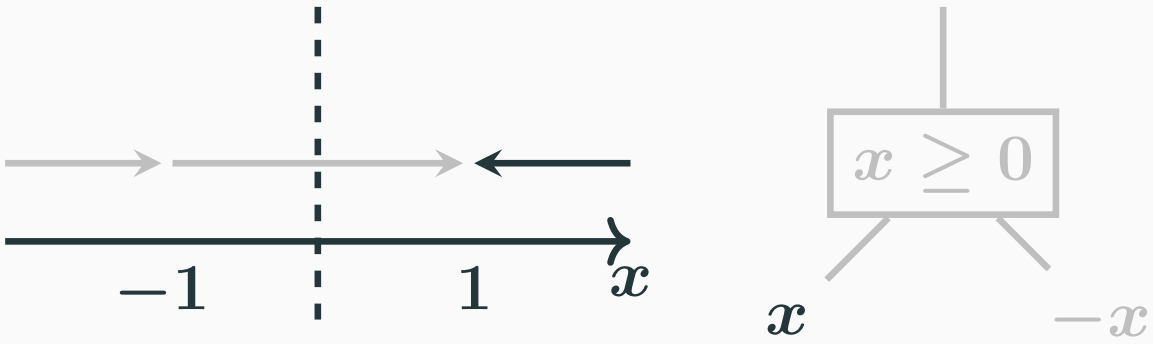
Crossing Examples

If a “crossing example” exists after Phase 2, functions at each leaf cannot be determined independently.



Crossing Examples

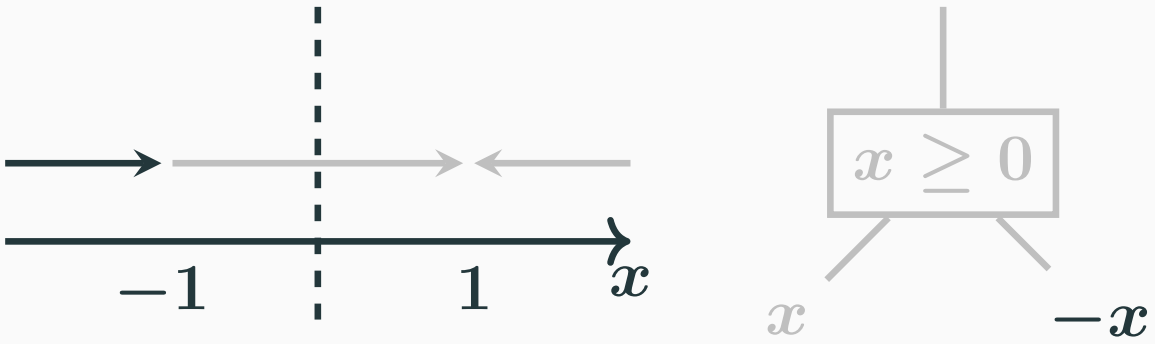
If a “crossing example” exists after Phase 2, functions at each leaf cannot be determined independently.



x ranks all transition examples in $x \geq 0$.

Crossing Examples

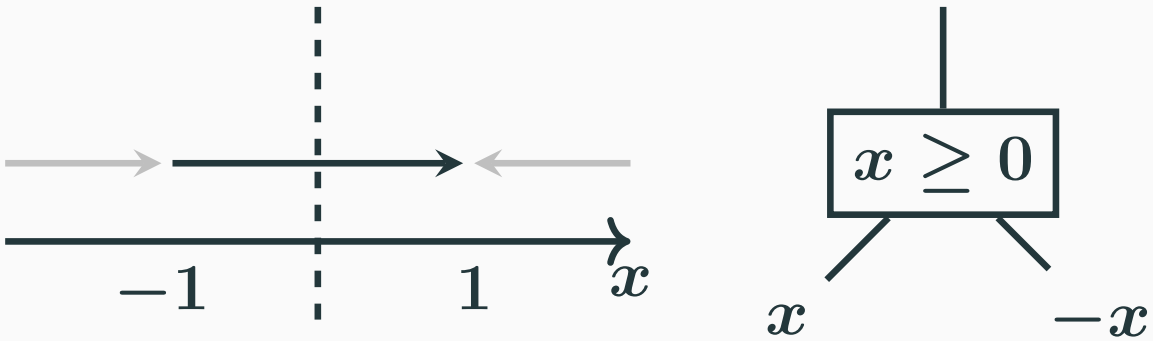
If a “crossing example” exists after Phase 2, functions at each leaf cannot be determined independently.



$-x$ ranks all transition examples in $x < 0$.

Crossing Examples

If a “crossing example” exists after Phase 2, functions at each leaf cannot be determined independently.



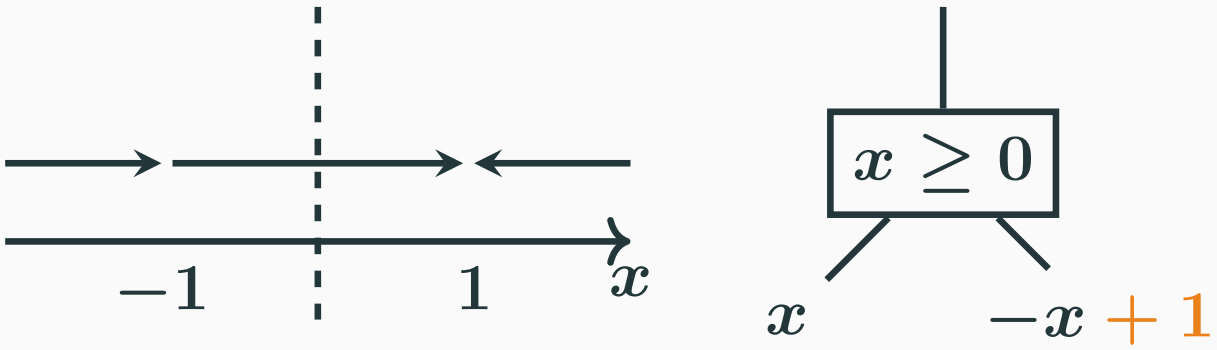
$f(x) = \text{if } x \geq 0 \text{ then } x \text{ else } -x$

does **NOT** rank the crossing example.

$$f(-1) \not> f(1)$$

Crossing Examples

If a “crossing example” exists after Phase 2, functions at each leaf cannot be determined independently.

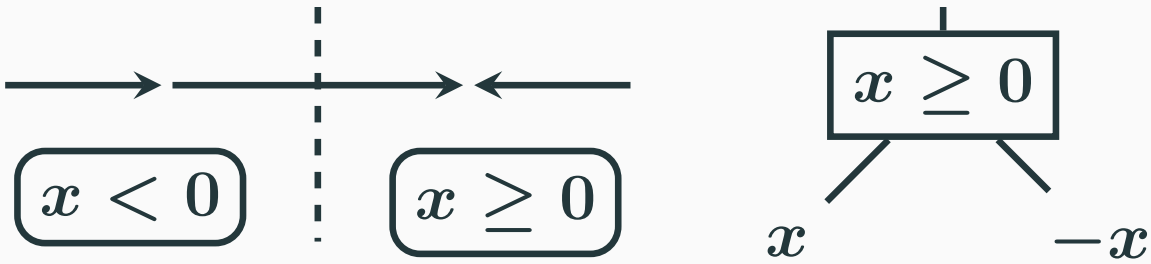


We need to add some constant.

$$f(x) = \text{if } x \geq 0 \text{ then } x \text{ else } -x+1$$

Cycle Detection Theorem

Given a decision tree that ranks non-crossing examples,



let G be the graph induced by crossing examples.

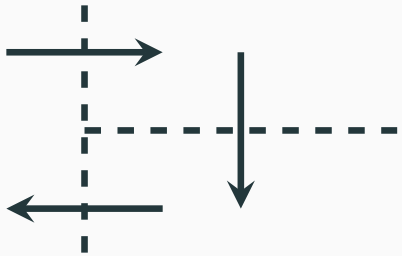
Theorem.

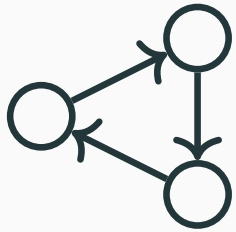
If G is acyclic, then there is a decision tree for all transition examples without changing the segmentation.

Phase 3: Cut Implicit Cycles

By the cycle detection theorem,

it suffices to handle the following case.



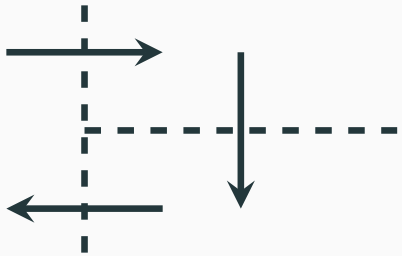


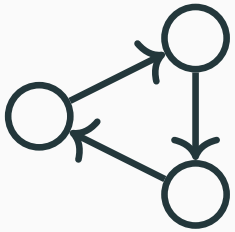
If there is such an “implicit cycle”, then cut the cycle.

Phase 3: Cut Implicit Cycles

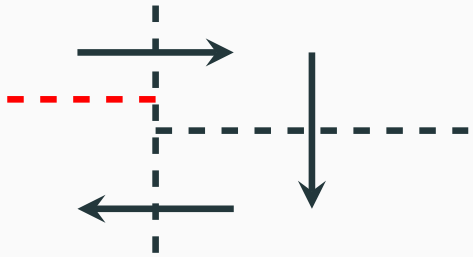
By the cycle detection theorem,

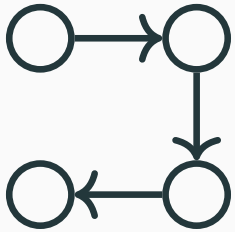
it suffices to handle the following case.





If there is such an “implicit cycle”, then cut the cycle.





Repeat this until no implicit cycle.

Outline of Our Algorithm

1. Detect explicit cycles
    - Proceed if no explicit cycle exists
2. Refine the current segmentation
    - We get a decision tree that ranks “non-crossing examples”.
3. Cut implicit cycles
    - Eventually, there is no implicit cycle.
    - We get a decision tree that ranks all examples.

Outline

Setting

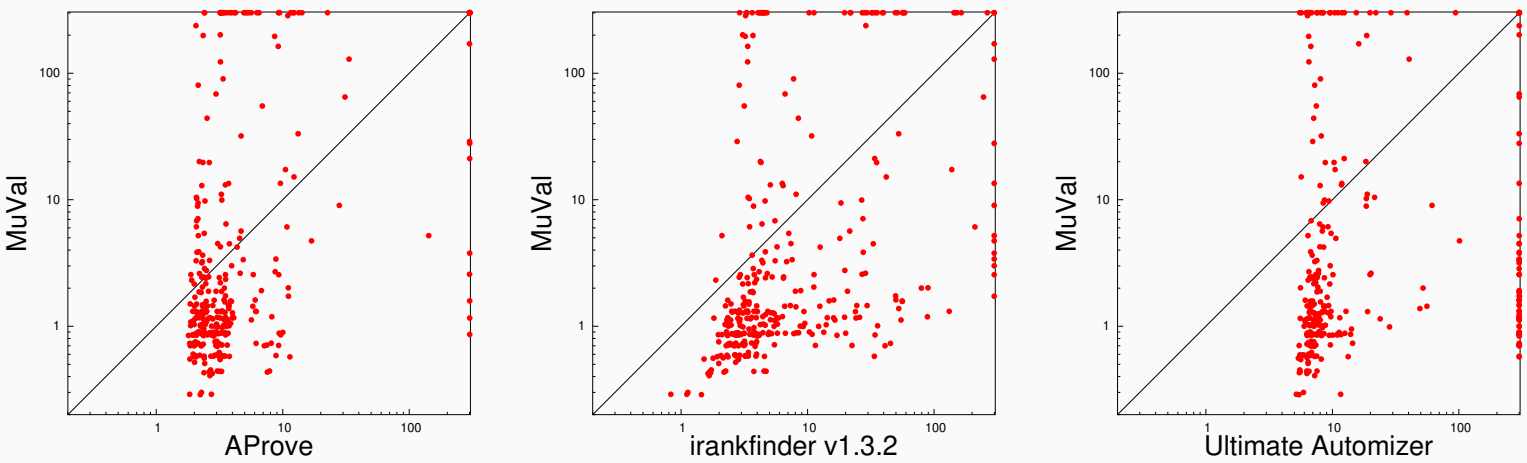
- Our tool (MuVal)
  - Synthesizer: Phase 1-3 + two heuristics (eager/lazy) to obtain simple candidates
- Benchmarks: Termination Competition 2020 (C Integer)
- Compared with AProVE, iRankFinder, and Ultimate Automizer
- Environment: StarExec (https://www.starexec.org/)

Number of Solved Benchmarks

	Yes	No	Timeout (300 s)	Unknown (give up)
MuVal (eager)	204	89	42	0
MuVal (lazy)	200	84	51	0
AProVE	216	100	16	3
iRankFinder	208	92	0	34
Ultimate Automizer	180	83	2	70

- Ultimate Automizer < MuVal < iRankFinder < AProVE

Runtime



Lower triangle = MuVal is faster

- MuVal solved faster than iRankFinder in 265 / 335 benchmarks.

Selected Benchmarks

Some of the benchmarks are solved by MuVal but not by AProVE.

- TelAviv-Amir-Minimum_true-termination.c
  - Requires piecewise affine ranking functions

$$f(x, y) = \mathbf{min}\{x, y\}$$

  - Solved by MuVal but not by AProVE.
- LogMult.c
  - Contains nonlinear transition (`y = y * y;`)
  - Has affine ranking function
  - Solved only by MuVal

Conclusions and Future Work

Conclusions

- Decision tree learning for ranking function synthesis
- Promising experimental results
  - MuVal solved more than Ultimate Automizer.
  - MuVal solved faster than iRankFinder.
  - Some benchmarks were solved only by MuVal.

Future work

- Improving performance by combining ML
- Extension to ranking supermartingale synthesis