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)?"

Ranking function [Floyd, '67]

$f : S \to \mathbb{Z}$ is a **ranking function**

if for each $(x, x') \in \tau$

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

If a ranking function exists, then τ terminates.

Our Contribution

We propose an example-based termination analyzer.

Specifically, we use ...

- CEGIS architecture [Solar-Lezama et al., ASPLOS'06]
- **transition examples**
  - not trace examples [Urban et al., TACAS'16] [Fedyukovich et al., CAV'18]
- **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)

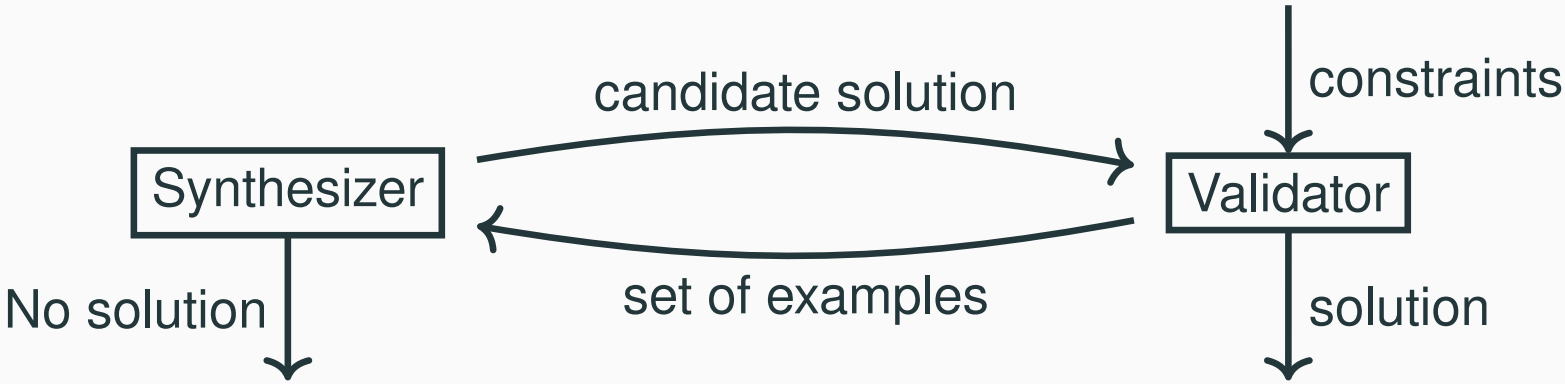
  [Champion et al., TACAS'18], [Ezudheen et al., OOPSLA'18], ..

- for ranking function synthesis (transition examples)

  our work

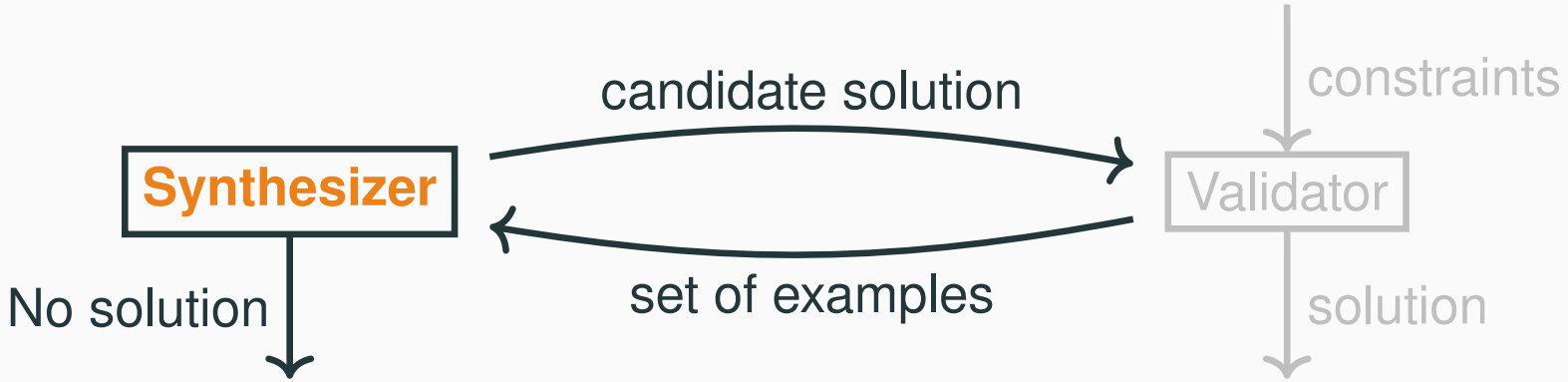
CounterExample Guided Inductive Synthesis

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



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

Overview of Our Synthesizer

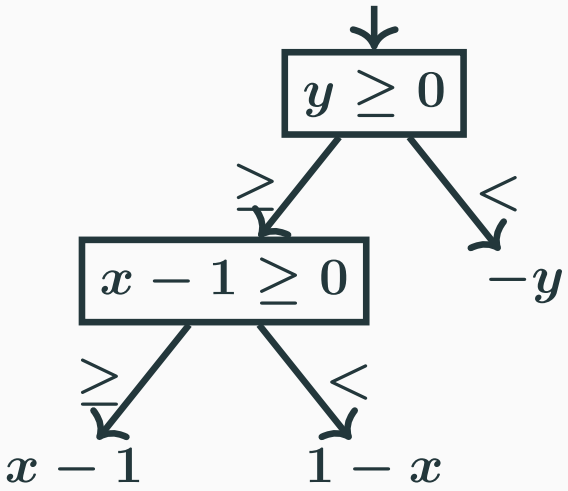


- Input: a set of transition examples
- Output: a candidate solution for the input
- Search space: piecewise affine functions

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 Synthesizer 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.

Experimental Results

Experiment

- Our tool (MuVal)
- Compared with AProVE, iRankFinder, and Ultimate Automizer
- Benchmarks: Termination Competition 2020 (C Integer)

Result

- MuVal solved more than Ultimate Automizer.
- MuVal solved faster than iRankFinder.
- Some benchmarks were solved only by MuVal.