Synthesizing Ranking Functions from Bits and Pieces

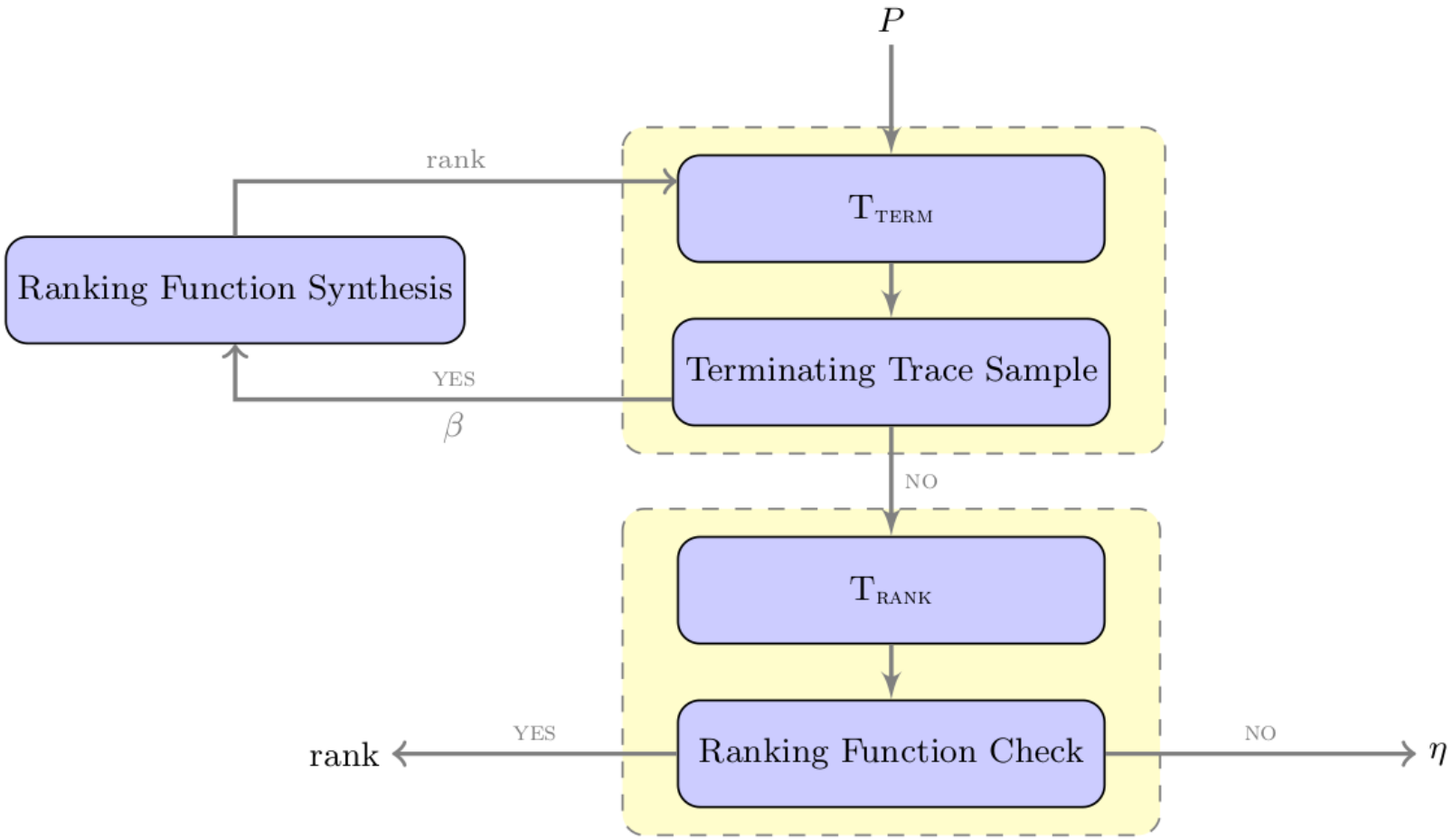
Caterina Urban et al.

February 2, 2021

Overview

- Synthesizing ranking function to prove termination based on safety checking.
- Bits: **bits** of information from terminating executions.
  Pieces: Extrapolate from these bits to obtain ranking functions **pieces** and combine them into ranking functions.
- Algorithm implemented in SeaHorn targeting on C code.

Overview


P
rank
Ranking Function Synthesis
T_TERM
Terminating Trace Sample
YES
β
NO
T_RANK
rank
YES
Ranking Function Check
NO
η

Preliminaries

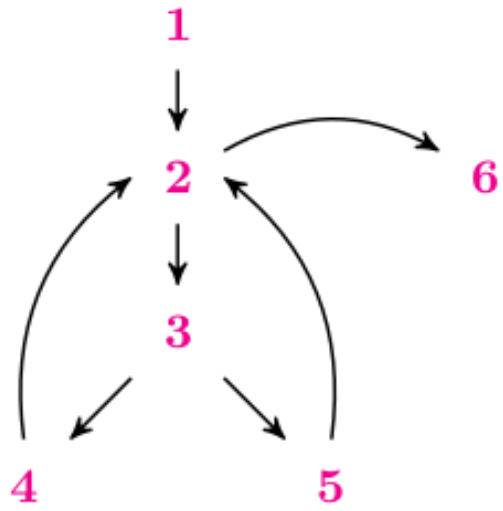
- Transition System: $\langle \Sigma, \tau \rangle$: states and transition relation.
- $s \in \Sigma$ is a pair $\langle l, \bar{x} \rangle$ where l is program control point and $\bar{x}$ is the vector of values of variables.
- A state is *terminating*, *non-terminating*, *potentially terminating* and *potentially nonterminating*.
- Ranking function: $\forall s, s'.\tau(s, s') \Rightarrow rank(s') < rank(s)$, $\forall s.rank(s) \geq 0$.
- Control flow graph induced by the transition system.

Loops in Control Flow Graph

- Loop header, entry edge, loop edge and exit edge.

```
int [1]x := ?
while [2](x ≠ 0) do
    if [3](x < 10) then
        [4]x := x + 1
    else
        [5]x := −x
    fi
od[6]
```

(a)



(b)

Verifying Safety Properties

Verifying Nontermination:

```
int [1] x := 0, y := 9
while [2] (x ≠ y) do
    [3] x := x + 1
    [4] y := y + 1
od[5]
```

(a)

```
int [1] x := 0, y := 9
while [2] (x ≠ y) do
    [3] x := x + 1
    [4] y := y + 1
od
assert (false)[5]
```

(b)

Verifying a ranking function:

```
int [1] x := ?, r := max{−x, 21 − x, x + 1}
while [2] (x ≠ 0) do
    r := r − 1
    assert (r ≥ 0)
    if [3] (x < 10) then [4] x := x + 1 else [5] x := −x fi
od[6]
```

Verifying Termination via Safety: the Algorithm

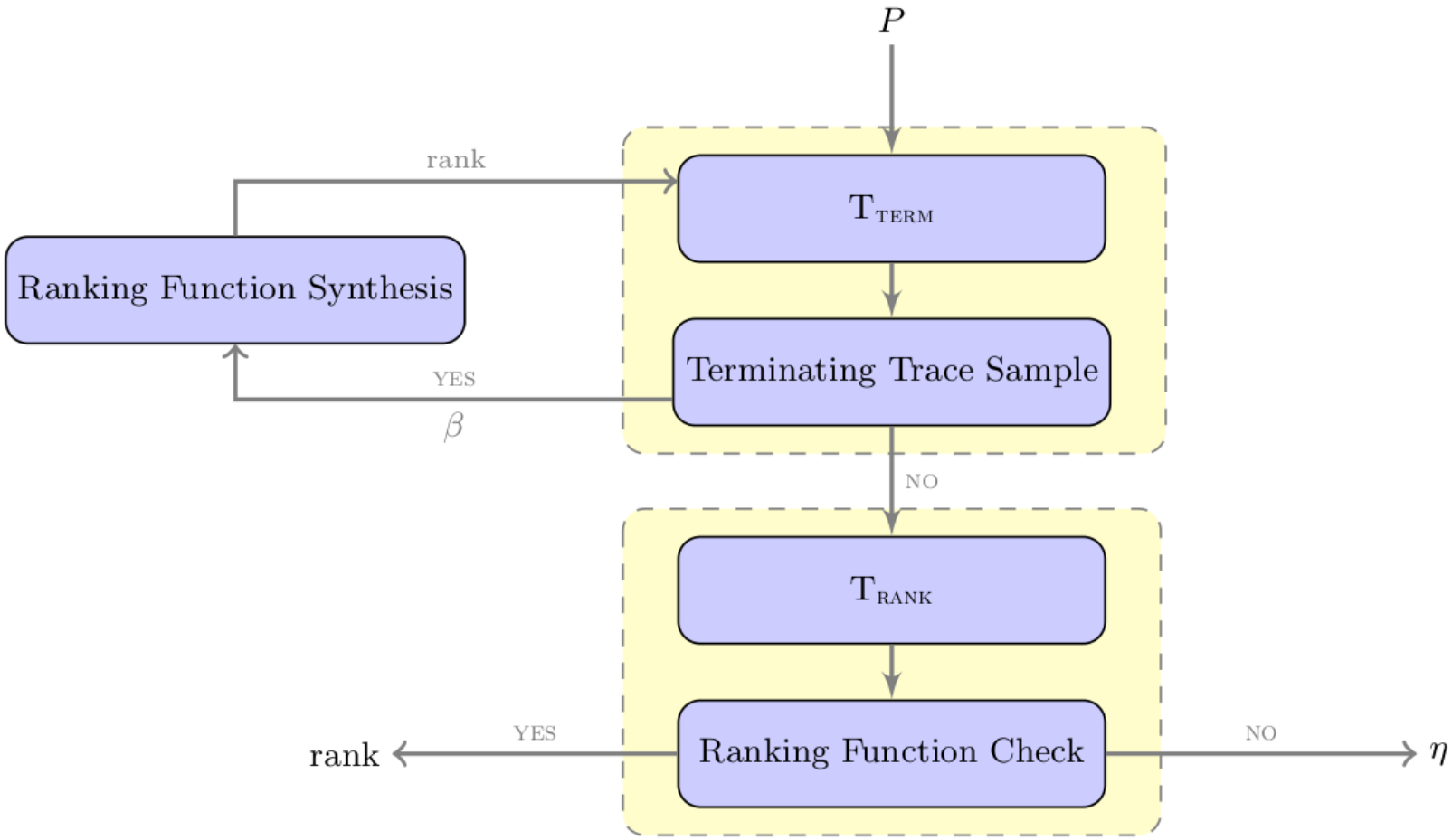
Algorithm 1 : Program Termination

```
1: function IsTerminating(⟨Σ, τ⟩)
2:     R ← ∅
3:     for h ∈ getLoops(⟨Σ, τ⟩) do                ▷ h is a loop header in the program
4:         r : ρ ← isLoopTerminating(h, ⟨Σ, τ⟩)
5:         if r then                                          ▷ the loop is terminating
6:             R ← R [h ↦ ρ]
7:         else return FALSE : ρ                  ▷ ρ is a potentially non-terminating state
8:     return TRUE : R                            ▷ R is a ranking function for the program
```

Algorithm 2 : Loop Termination

```
1: function IsLoopTerminating(h, ⟨Σ, τ⟩)                          ▷ h is the loop header
2:     rank ← 0                                  ▷ candidate ranking function initialization
3:     B ← ∅
4:     while TRUE do
5:         β ← getTerminatingTrace(h, ⟨Σ, τ⟩, rank)
6:         if β then                          ▷ there are terminating traces violating rank
7:             B ← B ∪ β
8:             rank ← getCandidateRankingFunction(rank, B)
9:         else                              ▷ there are no terminating traces violating rank
10:            η ← isRankingFunction(rank)
11:            if η then                          ▷ η is a potentially non-terminating state
12:                return FALSE : η
13:            else                                ▷ rank is a ranking function for the loop
14:                return TRUE : rank
```

Verifying Termination via Safety



Search for Ranking Function Counterexamples

$\mathbf{T_{TERM}}$ **Transformation**: For a program $\langle \Sigma, \tau \rangle$ and candidate ranking function $rank$. Construct new set of state $\Sigma' = (\Sigma \times \mathbb{Z}) \cup \{\omega\}$, and modified transition relation τ':

- For each loop entry transition $\tau(s, \langle h, \bar{x} \rangle)$, there exists an τ^{rank} s.t.

$$\tau^{rank}(\langle s, r \rangle, \langle \langle h, \bar{x} \rangle, r' \rangle) \Leftrightarrow \tau(s, \langle h, \bar{x} \rangle) \wedge r' = rank(\bar{x})$$

- For each loop transition $tau(\langle h, \bar{x} \rangle, s)$, there exists a loop transition $r^{\ominus}$ s.t.

$$r^{\ominus}(\langle \langle h, \bar{x} \rangle, r \rangle, \langle s, r' \rangle) \Leftrightarrow \tau(\langle h, \bar{x} \rangle, s) \wedge r' = r \ominus 1$$

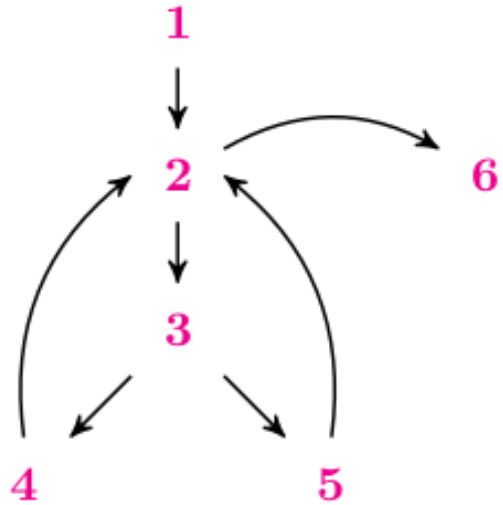
- For each loop exit transition $\tau(s, s')$, there exists a transition $\tau^{<}$ to the error state ω s.t.

$$\tau^{<}(\langle s, r \rangle, \omega) \stackrel{def}{=} r < 0$$

Example: $\mathbf{T_{TERM}}$ Transformation

```
int ¹x := ?
while ²(x ≠ 0) do
    if ³(x < 10) then
        ⁴x := x + 1
    else
        ⁵x := −x
    fi
od⁶
```

(a)



(b)

```
int ¹x := ?, r := rank
while ²(x ≠ 0) do
    r := r − 1
    if ³(x < 10) then ⁴x := x + 1 else ⁵x := −x fi
od
assert (r ≥ 0)⁶
```

Search for Ranking Function Counterexamples

Theorem

The error state ω is reachable
iff.
A finite trace jump out of loop with header h and visit h more than $rank(\bar{x})$ times.

Validating Ranking Function

$\mathbf{T_{RANK}}$ Transformation:

$$\tau^{rank}(\langle s, r\rangle, \langle\langle h, \bar{x}\rangle, r'\rangle) \Leftrightarrow \tau(s, \langle h, \bar{x}\rangle) \wedge r' = rank(\bar{x})$$

$$r^{\ominus}(\langle\langle h, \bar{x}\rangle, r\rangle, \langle s, r'\rangle) \Leftrightarrow \tau(\langle h, \bar{x}\rangle, s) \wedge r' = r \ominus 1$$

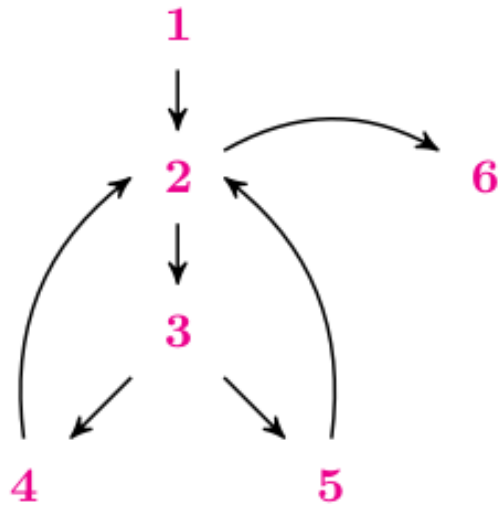
For **loop transitions** $\tau(s, s')$.

$$\tau^{<}(\langle s, r\rangle, \omega) \stackrel{def}{=} r < 0$$

Example: $\mathbf{T_{RANK}}$ Transformation

```
int [1]x := ?
while [2](x ≠ 0) do
    if [3](x < 10) then
        [4]x := x + 1
    else
        [5]x := −x
    fi
od[6]
```

(a)



(b)

```
int [1]x := ?, r := max{−x, 21 − x, x + 1}
while [2](x ≠ 0) do
    r := r − 1
    assert (r ≥ 0)
    if [3](x < 10) then [4]x := x + 1 else [5]x := −x fi
od[6]
```

Validating Ranking Functions

Theorem

The error state ω is reachable
iff.
The corresponding finite trace is the prefix of a an infinite trace which visits the loop header h more than $rank(\bar{x})$ times.

Synthesis of Candidate Ranking Function

- Target: affine ranking function pieces.
- $\{\langle \bar{x}_1, r_1 \rangle, \langle \bar{x}_2, r_2 \rangle, \ldots\}$ is the set of pairs mapping the initial states of terminating traces to number of iterations need for termination.
- Template: $\bar{m} \cdot \bar{x} + q$

$$\begin{aligned} \bar{m} \cdot \bar{x}_1 + q &= r_1 \\ \bar{m} \cdot \bar{x}_2 + q &= r_2 \\ &\vdots \end{aligned}$$

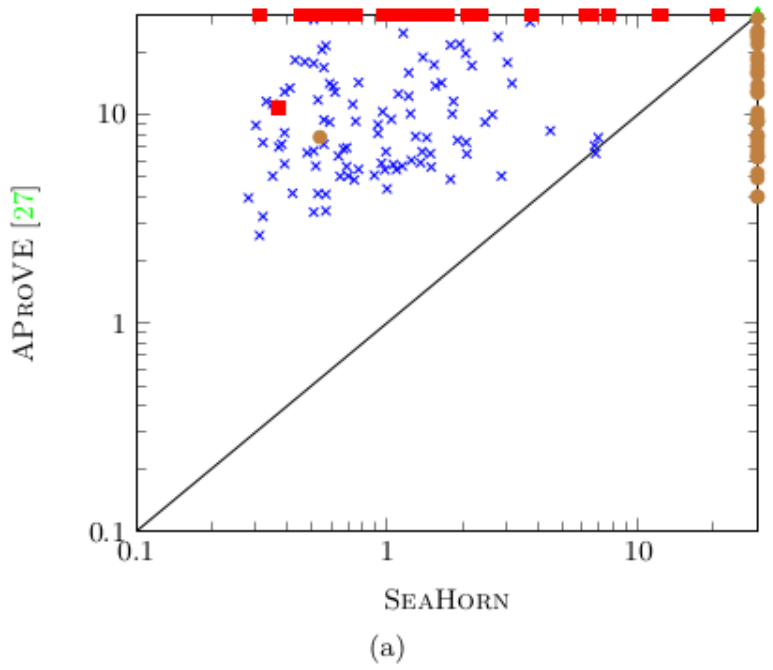
Then utilize these ranking pieces to form piecewise, multi-phase and lexicographic ranking functions.

Implementation and Experiments

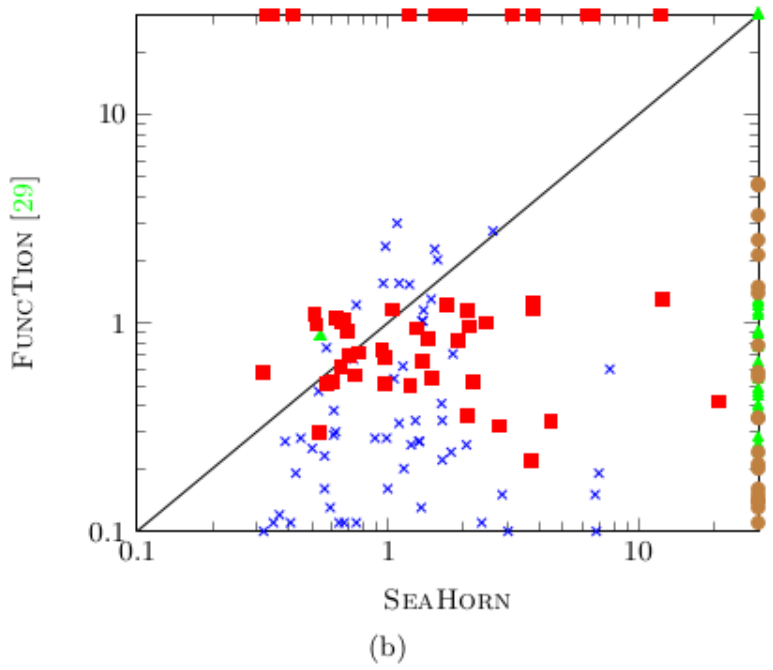
Implemented in SeaHorn.
Experimental evaluation is conducted on SV-COMP 2015.

	Tot	Time
SeaHorn	135	1.71s
AProVE [27]	129	10.77s
FuncTion [29]	111	0.55s
HIPTnT+ [21]	152	0.62s
Ultimate [15]	109	8.45s

Experimental Evaluation



(a)



(b)