Better termination proving through cooperation

Marc Brockschmidt [1] Byron Cook [2,3] Carsten Fuhs [3]

[1]RWTH Aachen University

[2]Microsoft Research Cambridge

[3]University College London

Deduktionstreffen 2013

Termination Analysis: Invariants and Rank Functions

Example

```
$y := 1;$
**while** $x > 0$ **do**
    $x := x - y;$
    $y := y + 1;$
**done**
```

- Invariant $y > 0$ and rank function x prove termination
- How do we know that we need $y > 0$? ↷ x requires it

Termination Analysis: Invariants and Rank Functions

Example

```
$y := 1$;
**while** $x > 0$ **do**
    $x := x - y$;
    $y := y + 1$;
**done**
```

- Invariant $y > 0$ and rank function x prove termination
- How do we know that we need $y > 0$? ↷ x requires it
- How do we know that x is a RF? ↷ $y > 0$ proves it

Termination by iterative strengthening: Idea

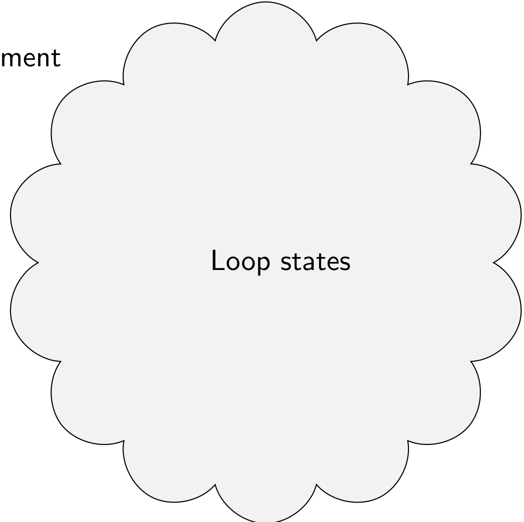
1. Safety: Provide samples (Counterexamples)
2. Rank tool: Find specific termination argument
3. Safety: Prove generality, or 1

Termination by iterative strengthening: Idea

1. Safety: Provide samples (Counterexamples)
2. Rank tool: Find specific termination argument
3. Safety: Prove generality, or 1

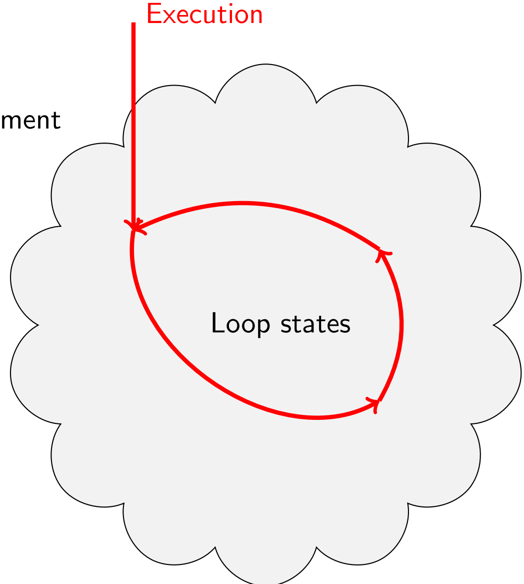
Termination by iterative strengthening

Find counterexample
then strengthen argument



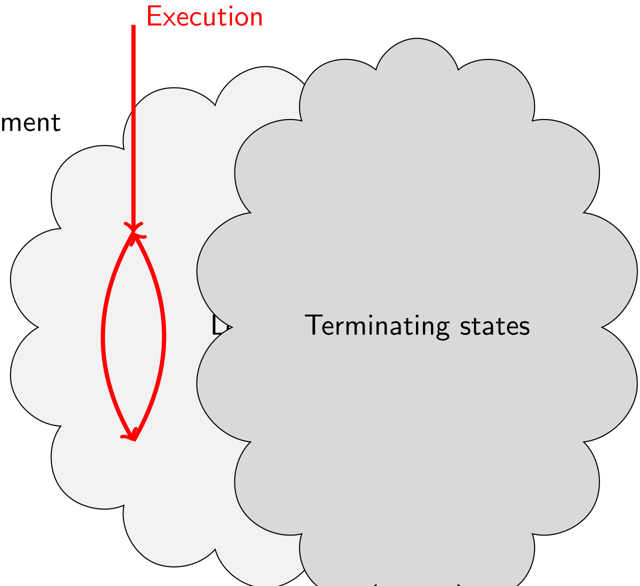
Termination by iterative strengthening

Find counterexample
then strengthen argument



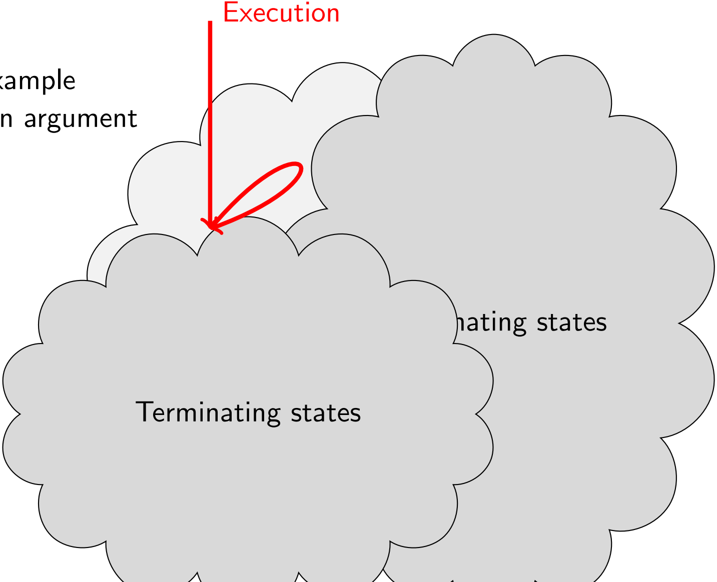
Termination by iterative strengthening

Find counterexample
then strengthen argument



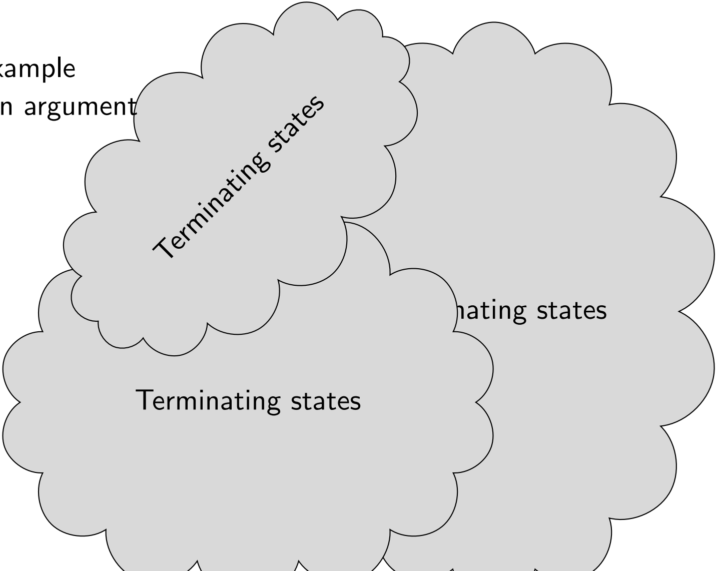
Termination by iterative strengthening

Find counterexample
then strengthen argument



Termination by iterative strengthening

Find counterexample
then strengthen argument



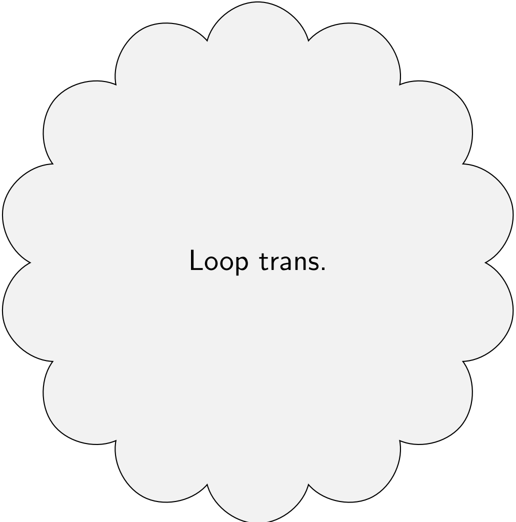
Termination by iterative strengthening: Worst case

1. Safety: Look at everything, then return old sample
2. Rank tool: Find **too** specific termination argument
3. Safety: Can't prove generality, repeat 1

Termination by iterative strengthening: Worst case

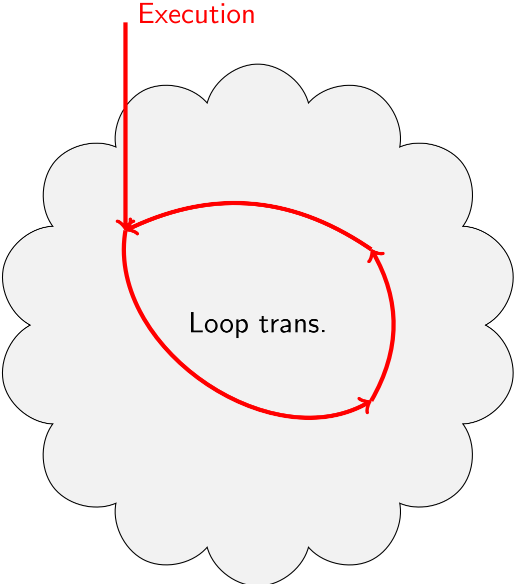
1. Safety: Look at everything, then return old sample
2. Rank tool: Find **too** specific termination argument
3. Safety: Can't prove generality, repeat 1

Termination by iterative simplification



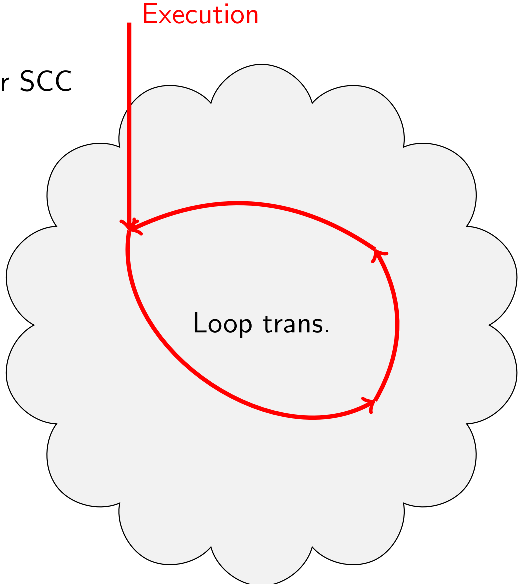
Loop trans.

Termination by iterative simplification



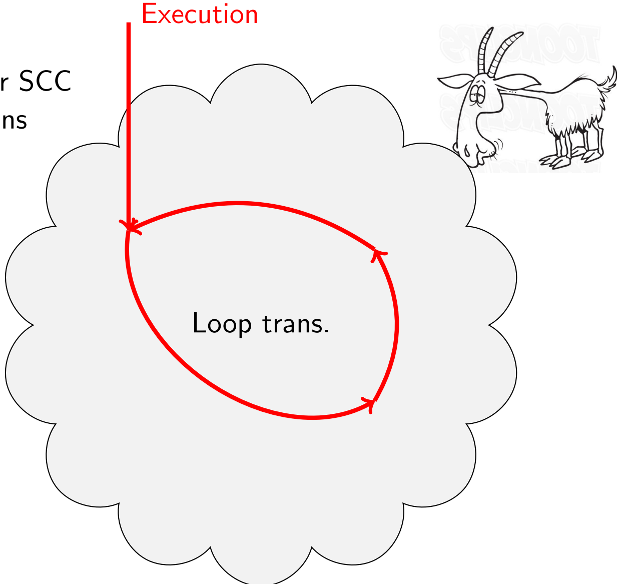
Termination by iterative simplification

Find rank function for SCC



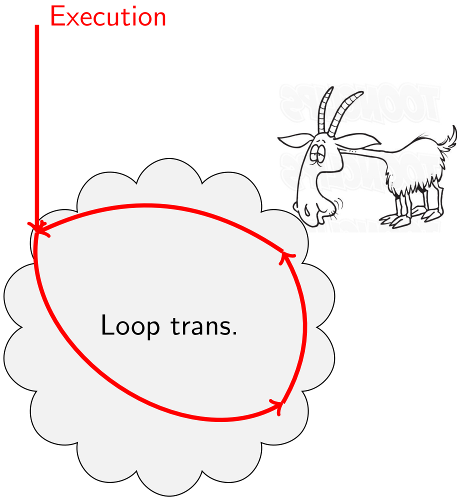
Termination by iterative simplification

Find rank function for SCC
then remove transitions



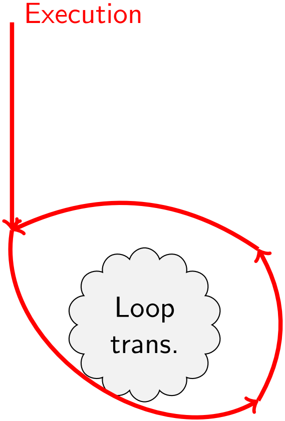
Termination by iterative simplification

Find rank function for SCC
then remove transitions



Termination by iterative simplification

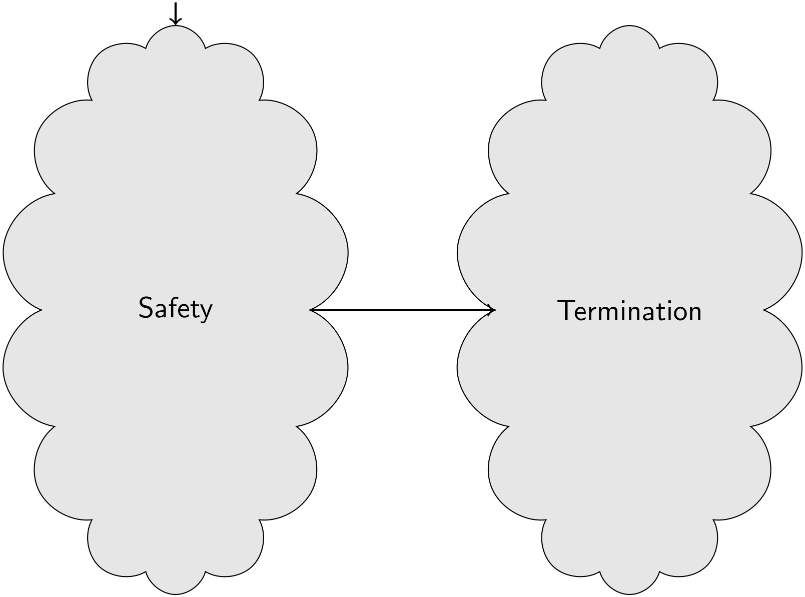
Find rank function for SCC
then remove transitions



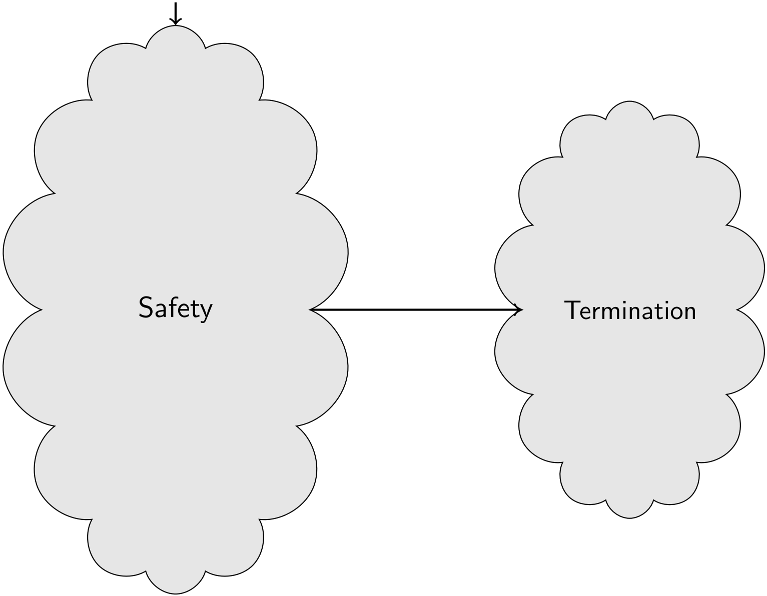
Termination by cooperation

1. Safety: Provide samples (Counterexamples)
2. Rank tool: Find termination argument **in context**
3. Rank tool: Mark definitely terminating parts
4. Safety: Prove generality for rest, or 1

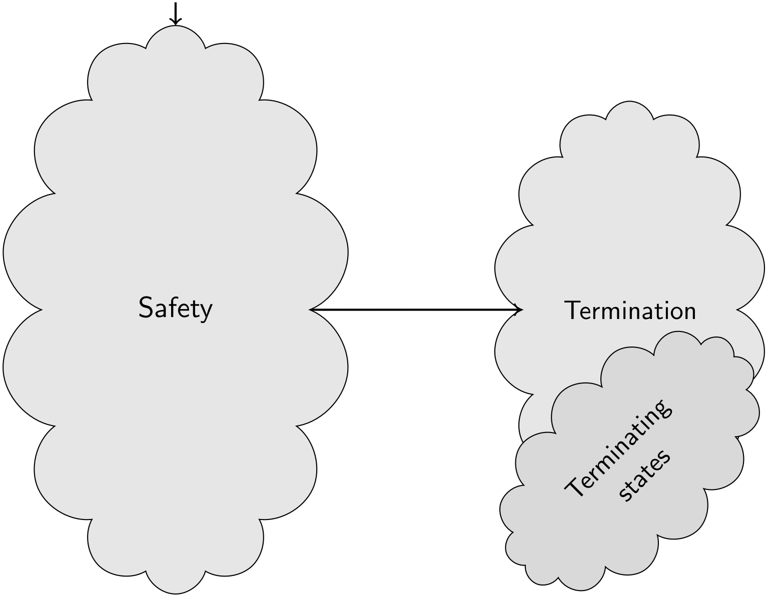
Cooperation: High-level view



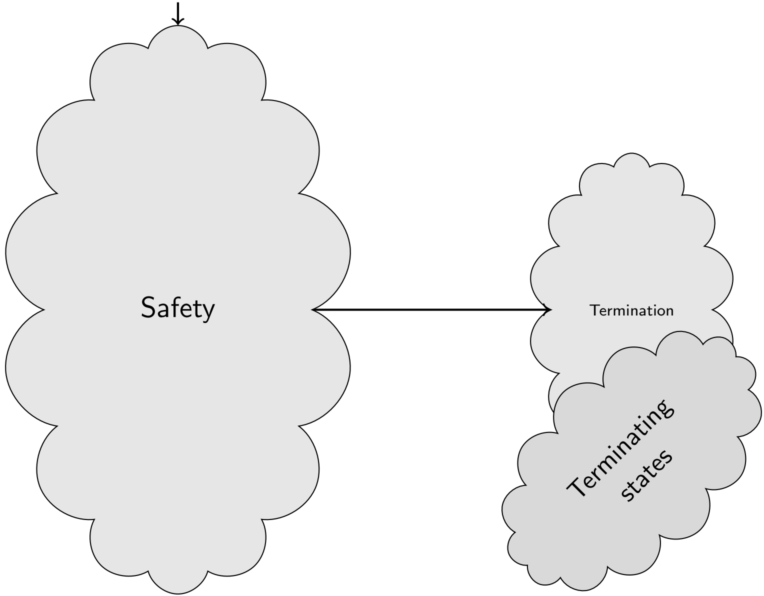
Cooperation: High-level view



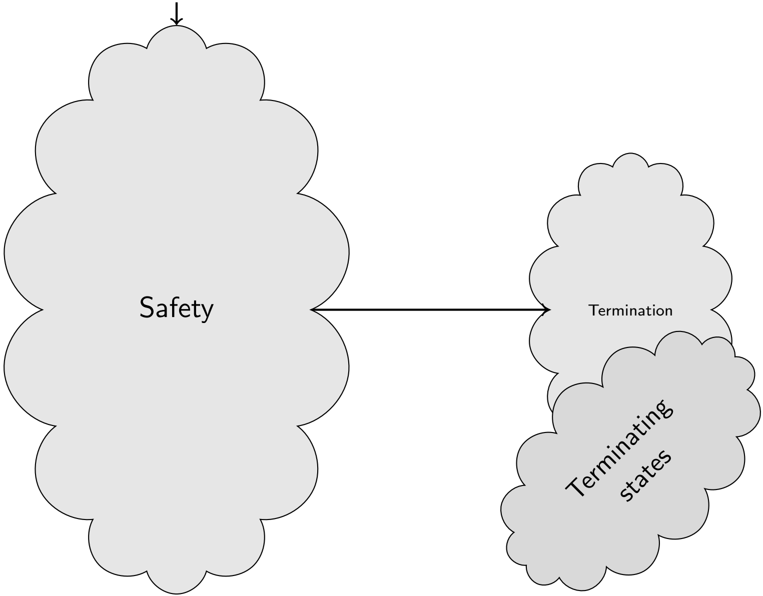
Cooperation: High-level view



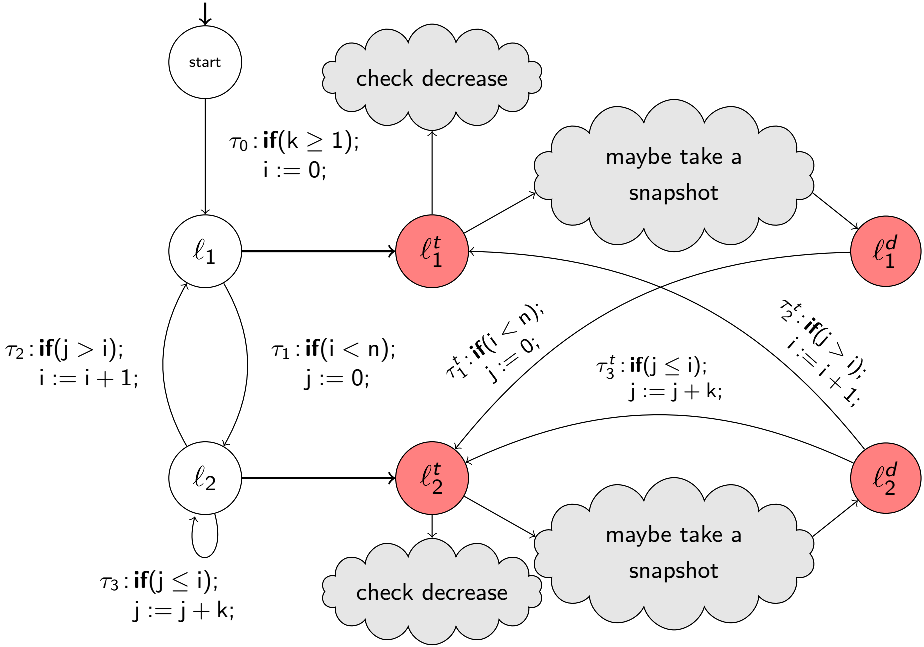
Cooperation: High-level view



Cooperation: High-level view



Cooperation: High-level view


start
ℓ_1
ℓ_2
ℓ_1^t
ℓ_2^t
ℓ_1^d
ℓ_2^d
check decrease
maybe take a snapshot
maybe take a snapshot
check decrease
τ_0 : **if**(k $\geq$ 1); i := 0;
τ_1 : **if**(i < n); j := 0;
τ_2 : **if**(j > i); i := i + 1;
τ_3 : **if**(j $\leq$ i); j := j + k;
τ_1^t : **if**(i < n); j := 0;
τ_2^t : **if**(j > i); i := i + 1;
τ_3^t : **if**(j $\leq$ i); j := j + k;

Cooperation: High-level view

Intuition:

- **Safety subgraph**: original program
- **Termination subgraph**: instrumented copy

Cooperation: High-level view

Intuition:

- **Safety subgraph**: original program
- **Termination subgraph**: instrumented copy

- **Ranking**: Simplify problem, “point out hard bits”

Cooperation: High-level view

Intuition:

- **Safety subgraph**: original program
- **Termination subgraph**: instrumented copy

- **Ranking**: Simplify problem, “point out hard bits”
- **Safety**: Analyze whole program, “point out invariants”

Cooperation: High-level view

Intuition:

- **Safety subgraph**: original program
- **Termination subgraph**: instrumented copy

- **Ranking**: Simplify problem, “point out hard bits”
- **Safety**: Analyze whole program, “point out invariants”

Approach:

- Analyze whole SCC, not counterexample slice

Cooperation: High-level view

Intuition:

- **Safety subgraph**: original program
- **Termination subgraph**: instrumented copy

- **Ranking**: Simplify problem, “point out hard bits”
- **Safety**: Analyze whole program, “point out invariants”

Approach:

- Analyze whole SCC, not counterexample slice
- Remove transitions after proof

Cooperation: Evaluation

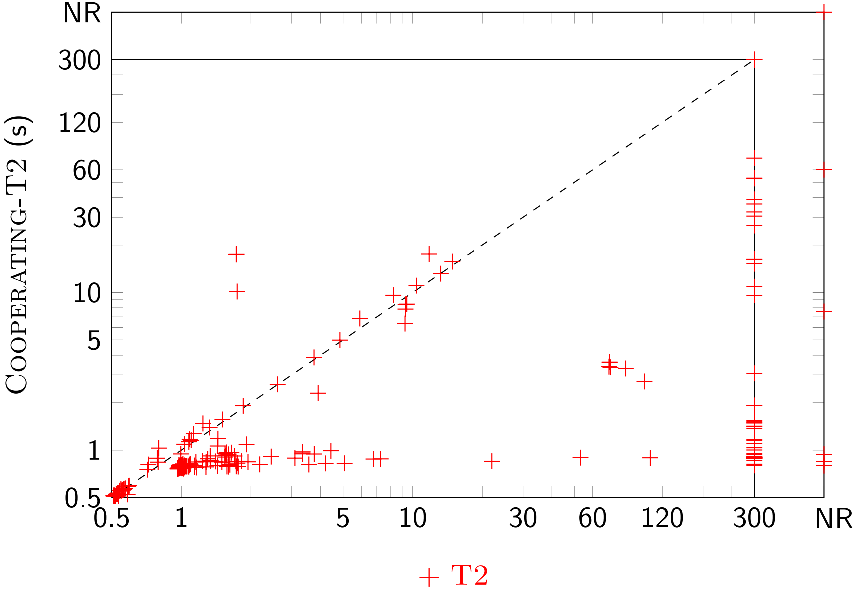
Evaluated on 449 termination proving benchmarks
260 known terminating, 181 known non-terminating, 8 unknown
Sources: Windows drivers, Apache, PostgreSQL, . . .

Cooperation: Evaluation

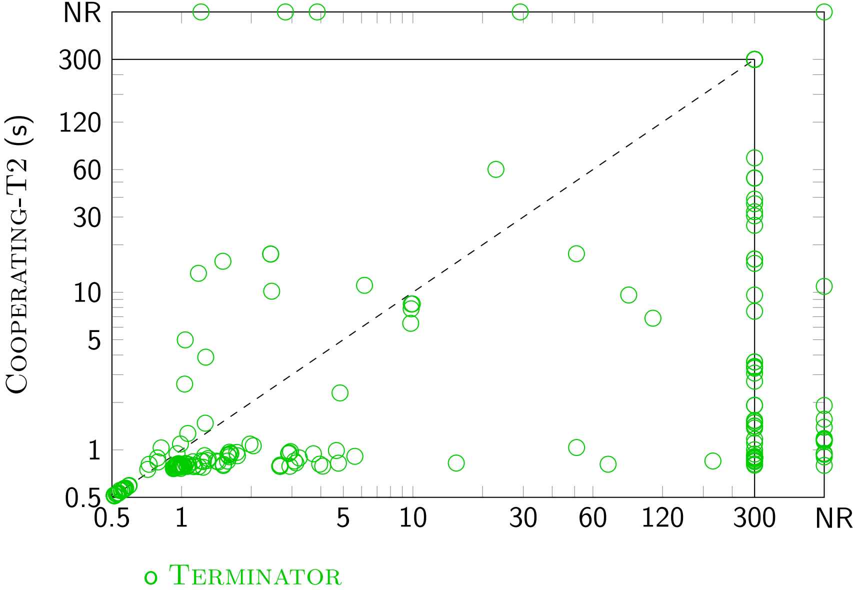
Evaluated on 449 termination proving benchmarks
260 known terminating, 181 known non-terminating, 8 unknown
Sources: Windows drivers, Apache, PostgreSQL, ...

	Term (#)	Term (avg. s)
Cooperating-T2	245	3.42
AProVE	197	2.21
KITTeL	196	4.65
T2	189	5.15
AProVE+Interproc	185	1.53
Terminator	177	4.99
Size-Change/MCNP	156	17.50
ARMC	138	16.16

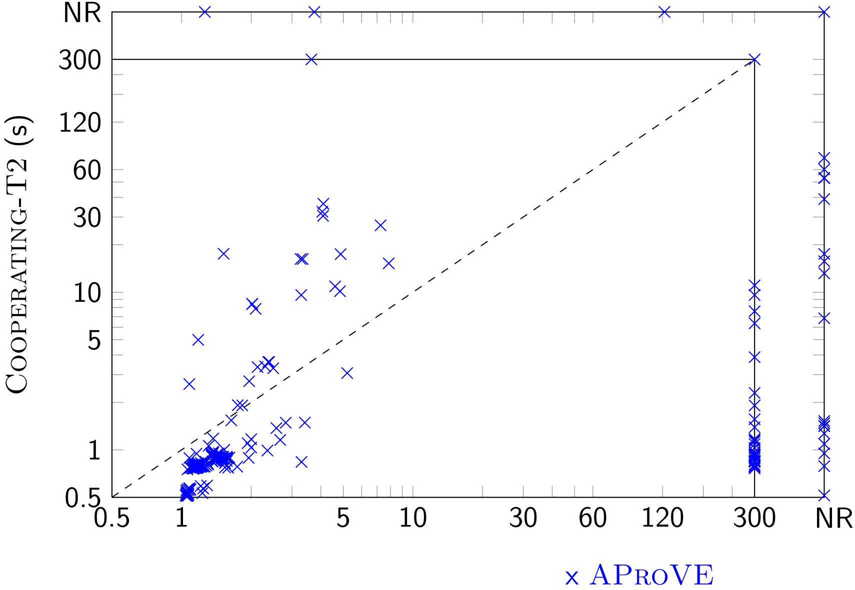
Cooperation: Evaluation



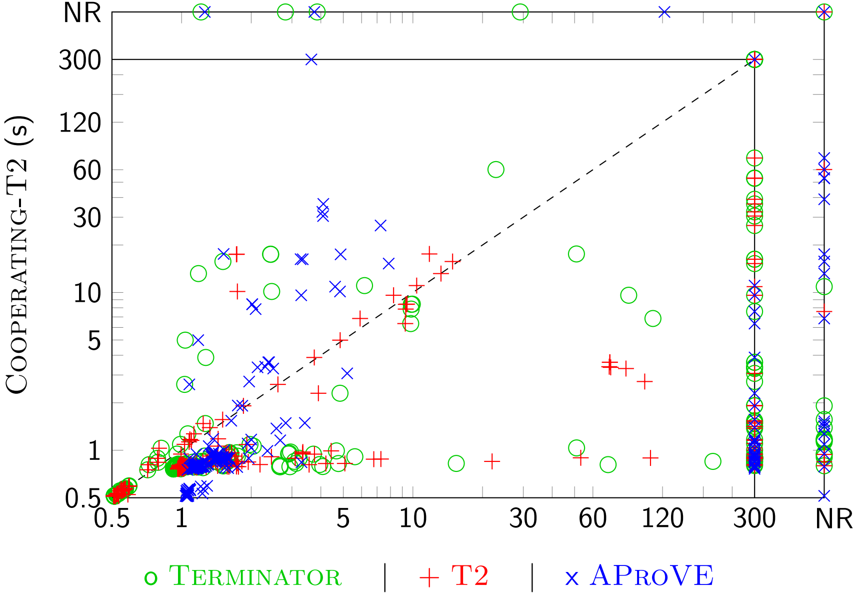
Cooperation: Evaluation



Cooperation: Evaluation



Cooperation: Evaluation



Cooperation: Evaluation

Evaluated on 449 termination proving benchmarks
260 known terminating, 181 known non-terminating, 8 unknown
Sources: Windows drivers, Apache, PostgreSQL, ...

	Term (#)	Term (avg. s)
Cooperating-T2	245	3.42
AProVE	197	2.21
KITTeL	196	4.65
T2	189	5.15
AProVE+Interproc	185	1.53
Terminator	177	4.99
Size-Change/MCNP	156	17.50
ARMC	138	16.16

Sources available: `http://research.microsoft.com/en-us/projects/t2/`