# Embracing Overapproximation for Proving Nontermination

Byron Cook, Carsten Fuhs, Kaustubh Nimkar, Peter O'Hearn
University College London

## ABSTRACT

One of the most fundamental program properties is termination: a program's execution will end regardless of its input. Bugs due to nontermination of a subroutine can be a nuisance, particularly since they cannot reliably be detected by testing. Although termination is undecidable, in practice we can use incomplete but effective methods to prove termination. However, failure to find a termination proof for a program does not imply nontermination.

A program is terminating iff its transition relation (restricted to reachable states) is well-founded. Consider (overapproximating) abstractions: their transition relation is a superrelation of the original program's transition relation. Every subrelation of a well-founded relation is itself well-founded, so proving an abstraction's termination also proves the original program's termination. The reverse, unfortunately, is not true: an abstraction's nontermination does not imply nontermination for the original program.

Thus, when proving nontermination with standard techniques [3], so far we cannot use overapproximating program analysis techniques. However, overapproximation is the workhorse of program analysis and would allow to analyze programs on complex domains like dynamically allocated data structures or nonlinear arithmetic. Here, currently only little support for nontermination proving is available.

Now *closed recurrence* [1]—existence of program states $C$ where *every* execution must stay in $C$ forever—comes to the rescue, allowing us to have our cake and eat it: we can now use most (well-behaved) overapproximations, formalized as *live abstractions*—the original program's final states must be final also in the abstraction—to prove that a nonterminating execution exists in the original program.

## BODY

*We embrace overapproximation to prove nontermination: a live abstraction's closed recurrence implies the input program's nontermination [2].*

## REFERENCES

[1] Hong-Yi Chen, Byron Cook, Carsten Fuhs, Kaustubh Nimkar, and Peter O'Hearn. Proving nontermination via safety. In *Proc. TACAS 2014*.

[2] Byron Cook, Carsten Fuhs, Kaustubh Nimkar, and Peter O'Hearn. Disproving termination with overapproximation. In *Proc. FMCAD 2014*.

[3] Ashutosh Gupta, Thomas A. Henzinger, Rupak Majumdar, Andrey Rybalchenko, and Ru-Gang Xu. Proving non-termination. In *Proc. POPL 2008*.

*Volume 3 of Tiny Transactions on Computer Science*