Credit: Termination Proofs for Systems Code*

Terminator is a tool that proves program termination for large, multi-sectioned systems composed of more than 20,000 lines of code, such as reactive systems. A reactive system must always terminate and a reactive system is labeled as non-responsive if it does not terminate under unexpected situations. Terminator uses incremental construction of termination-argument. Termination means no cycle in the set of states of program P.

* well-founded means their intersection is empty, nothing in common

Let

- I the set of initial states,
- T the termination argument which is a binary relation and a union of well-founded relations,
- R^+_I a transitive binary reachability relation,
- τ_1, τ_2, … τ_i, τ_{i+1}, …, τ_n a non empty sequence of statements in P to be executed, linking program states s_0, s_2, … , s_i, s_{i+1} ,.. , s_n in an orderly fashion.

Then if $R^+_I \subseteq$ T that means all binary relations in R^+_I are well-defined too, implying no loop in the set of states of R^+_I, which guarantees termination.

Else there must be two execution states s_i, s_n, $(s_i, s_n) \in R^+_I$, corresponding to non-empty statements $\tau_i \neq \varnothing$ and $\tau_n \neq \varnothing$ such that $(s_i, s_n) \notin$ T, disproving $R^+_I \subseteq$ T,

counterexample: execution states s_i and s_n are both happening at one single location of a program P, proving existence of a cycle τ_{i+1}, …, τ_n.

* Reachability analysis: in a distributed system, is global state s reachable at error location E
* Binary analysis: is threat assessment and vulnerability testing at the **binary** code level.

To be able to perform binary reachability analysis, $\widehat{P}$ is created as a version of program P that implements function F, a function whose least fixpoint is a characterization of R^+_I. Let safety checker $\widehat{P_T}$ be created by a transformation of $\widehat{P}$ then an error location in $\widehat{P_T}$ is not reachable iff $R^+_I \subseteq$ T.

Optimization, which is essential for verification of large scale programs, is performed as follows:

- specialization of $\widehat{P_T}$ based of the fact that $T = T^{l_1} \wedge T^{l_2} \wedge \ldots \wedge T^{l_n}$ for each cutpoint l_i
- pre-variables for C programs, creating a pre-variable for every heap and stack location addressable using C-expressions
- structured programs,
- weak binary reachability, to reach for a path is least costly-long for the purpose of termination verification.