

On Compositional safety verification with Max-SMT

Fabian Böller with David Korzeniewski

RWTH Aachen

fabian.boeller@rwth-aachen.de

SS 2017

Overview

- 1 Introduction
- 2 Preliminaries
- 3 Example execution

Safety verification

Prove that an assertion is *always* true at a location

Safety verification

Prove that an assertion is *always* true at a location

Non-compositional safety verification

Safety verification where the whole program is analyzed in one step

Safety verification

Prove that an assertion is *always* true at a location

Non-compositional safety verification

Safety verification where the whole program is analyzed in one step

Compositional safety verification

Safety verification where program parts are analyzed semi-independently and composed

Scalability \leftrightarrow Loss in precision

Program

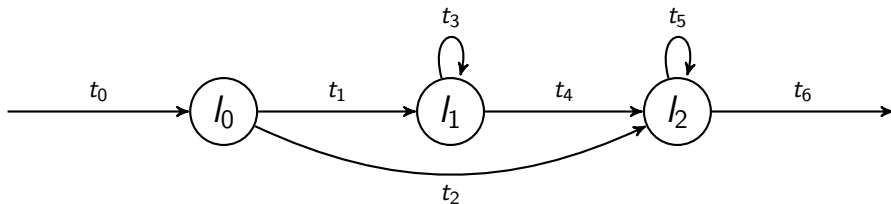
Program

$$\mathcal{L} = \{\ell_0, \ell_1, \ell_2\}$$



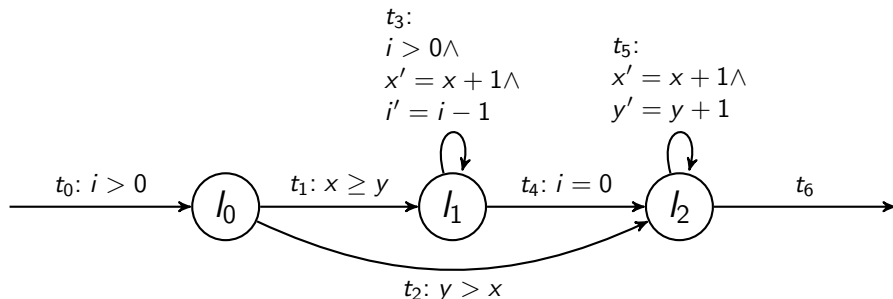
Program

$\mathcal{L} = \{l_0, l_1, l_2\}$, $\mathcal{T} = \{t_i \mid i \in \{1, \dots, 6\}\}$



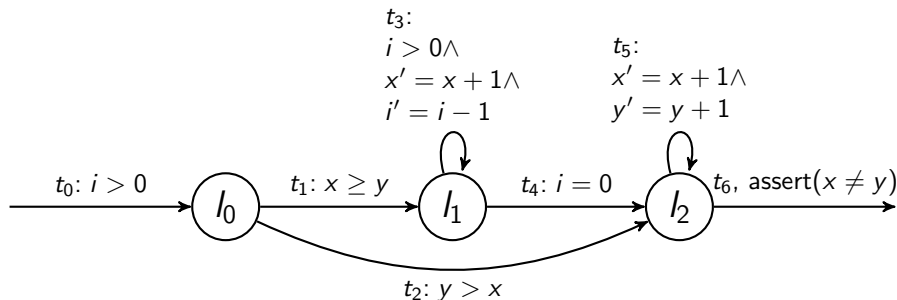
Program

$\mathcal{L} = \{\ell_0, \ell_1, \ell_2\}$, $\mathcal{T} = \{t_i \mid i \in \{1, \dots, 6\}\}$, $\mathcal{V} = \{x, y, i\}$, $\mathcal{V}' = \{x', y', i'\}$



Program

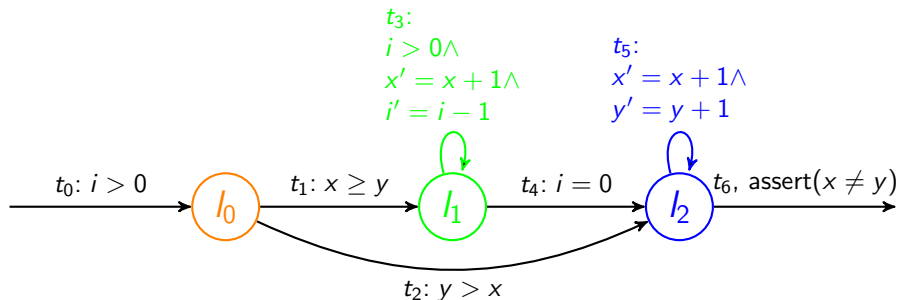
$\mathcal{L} = \{\ell_0, \ell_1, \ell_2\}$, $\mathcal{T} = \{t_i \mid i \in \{1, \dots, 6\}\}$, $\mathcal{V} = \{x, y, i\}$, $\mathcal{V}' = \{x', y', i'\}$

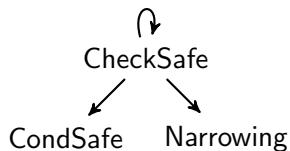


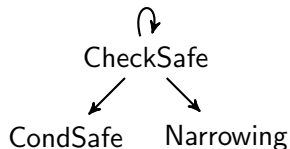
Programs

Program

$\mathcal{L} = \{\ell_0, \ell_1, \ell_2\}$, $\mathcal{T} = \{t_i \mid i \in \{1, \dots, 6\}\}$, $\mathcal{V} = \{x, y, i\}$, $\mathcal{V}' = \{x', y', i'\}$

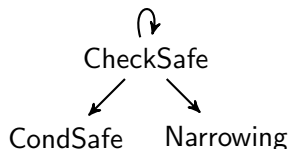






CheckSafe

Prove that an assertion is satisfied by backtracking through all entry components

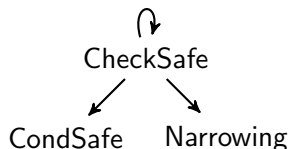


CheckSafe

Prove that an assertion is satisfied by backtracking through all entry components

CondSafe

Find a precondition for a component such that all runs satisfying the precondition always imply the postcondition



CheckSafe

Prove that an assertion is satisfied by backtracking through all entry components

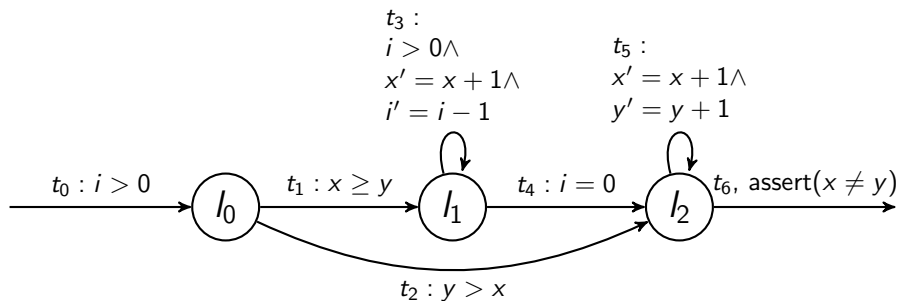
CondSafe

Find a precondition for a component such that all runs satisfying the precondition always imply the postcondition

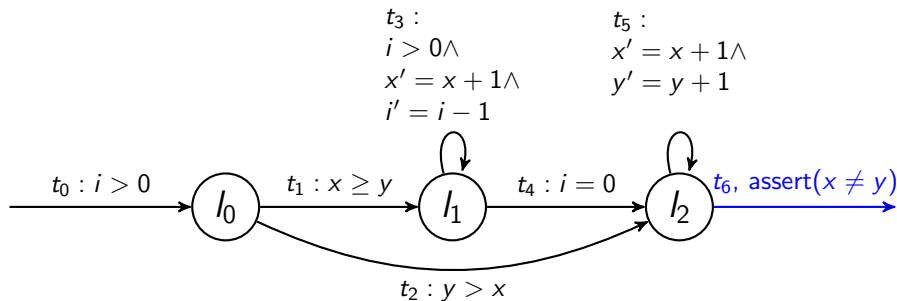
Narrowing

Manipulate the program such that new preconditions can be found

Example program



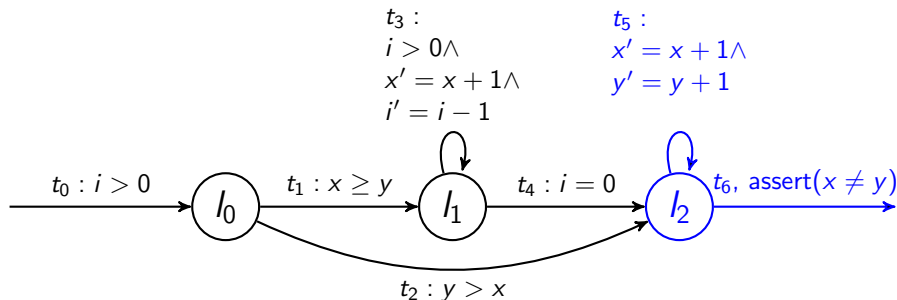
Example execution



Task

Prove that the program is safe for $x \neq y$ at t_6

Example execution



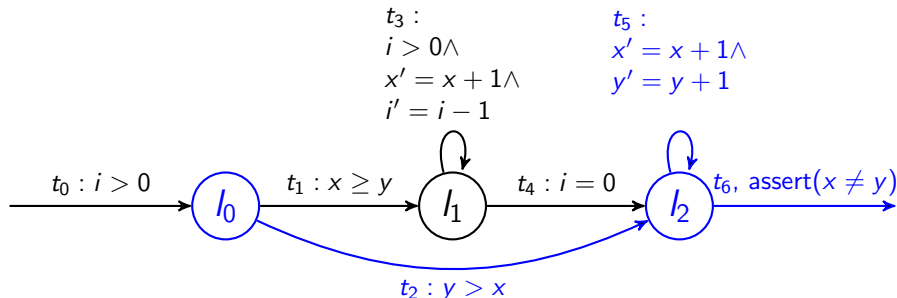
CheckSafe on $\{l_2\}$ for $x \neq y$

t_6 does not already imply $x \neq y$

t_6 is not an initial transition

Call CondSafe, get $x > y$ as precondition

Example execution



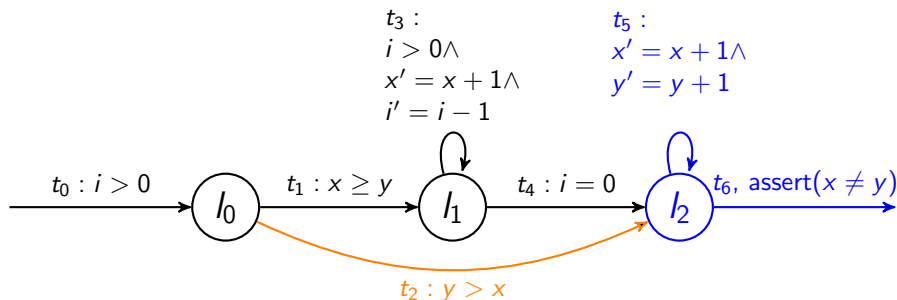
CheckSafe on $\{l_0\}$ for $x > y$

t_2 does not already imply $x > y$

t_2 is not an initial transition

Call CondSafe

Example execution

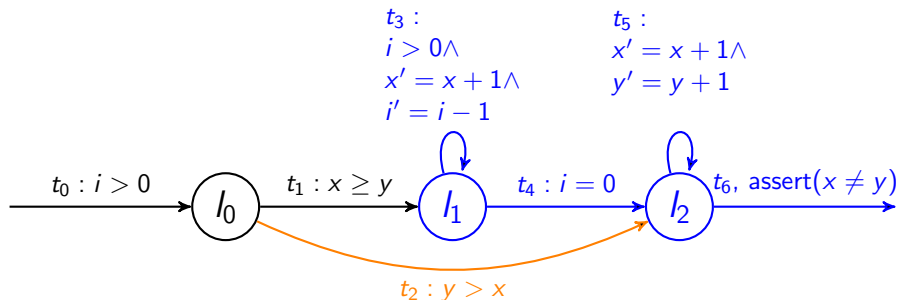


CheckSafe on $\{l_0\}$ for $x > y$

No precondition, since $y > x$ contradicts $x > y$

Path is maybe safe, but not for $x > y$

Example execution



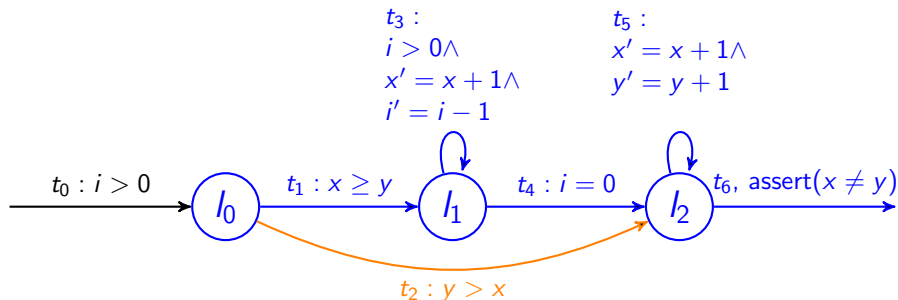
CheckSafe on $\{l_1\}$ for $x > y$

t_4 does not already imply $x > y$

t_4 is not an initial transition

Call CondSafe, get $i > 0 \wedge x \geq y$ as precondition

Example execution



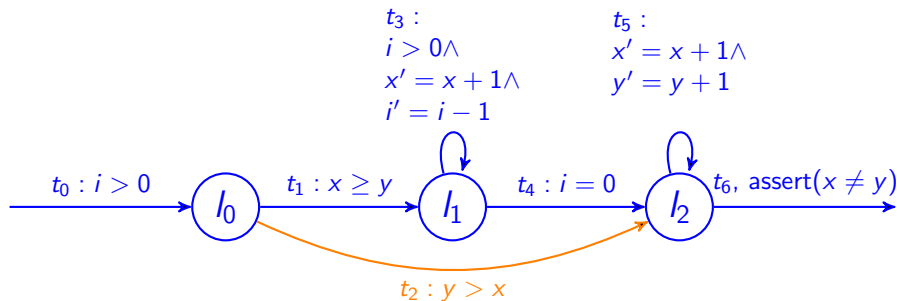
CheckSafe on $\{l_0\}$ for $i > 0$

t_1 does not already imply $i > 0$

t_1 is not an initial transition

Call CondSafe, get $i > 0$ as precondition

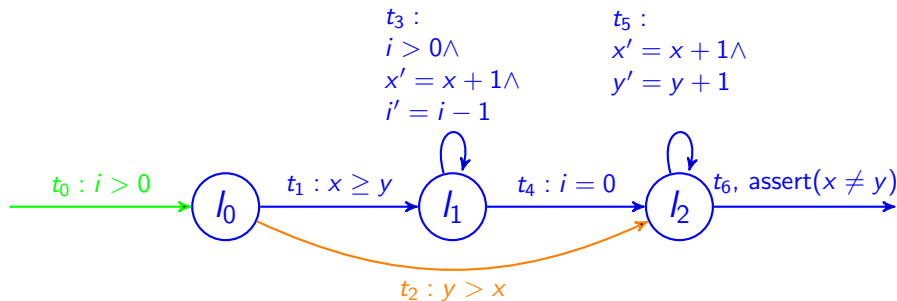
Example execution



CheckSafe on initial SCC for $i > 0$

t_0 does already imply $i > 0$

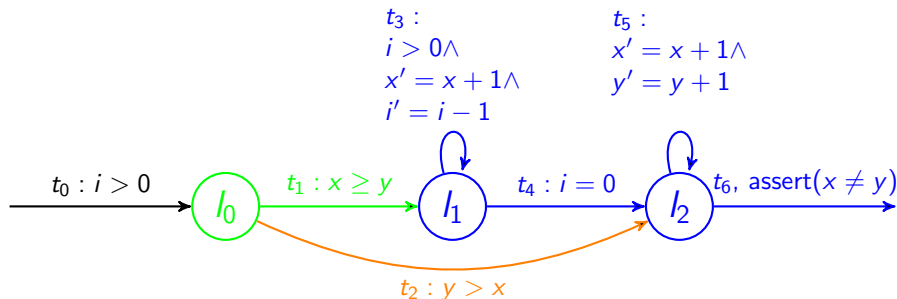
Example execution



CheckSafe on initial SCC for $i > 0$

Path is safe for $i > 0$

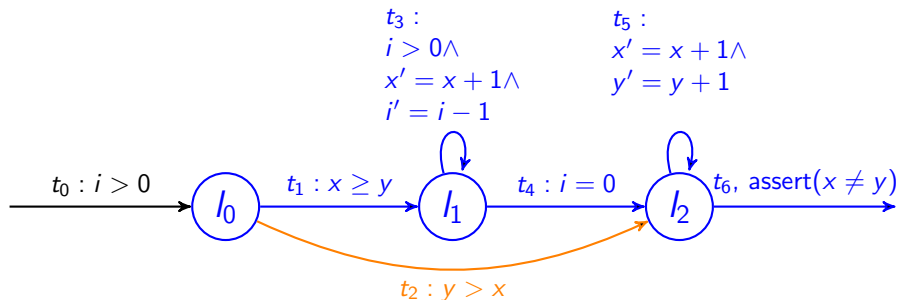
Example execution



CheckSafe on $\{l_0\}$ for $i > 0$

Path is safe for $i > 0$

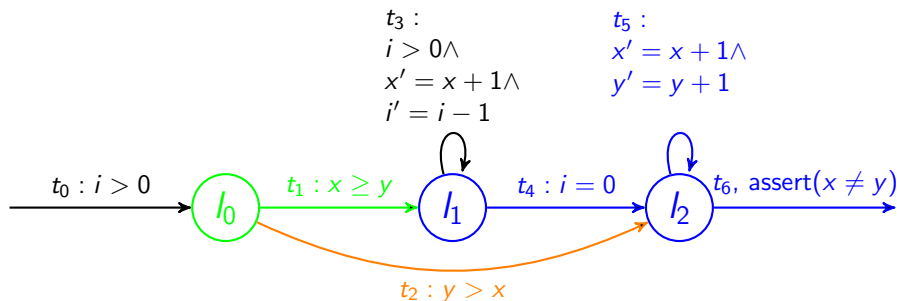
Example execution



CheckSafe on $\{l_0\}$ for $x \geq y$

t_1 does already imply $x \geq y$

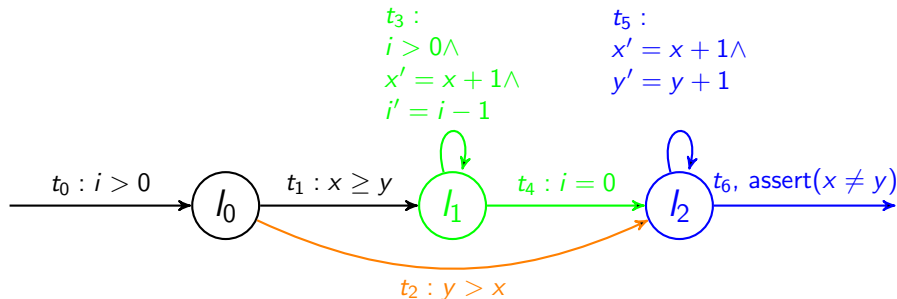
Example execution



CheckSafe on $\{l_0\}$ for $x \geq y$

Path is safe for $x \geq y$

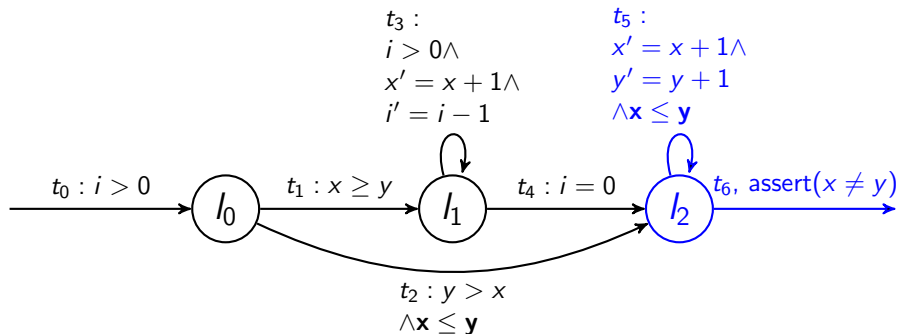
Example execution



CheckSafe on $\{l_1\}$ for $x > y$

Path is safe for $x > y$

Example execution

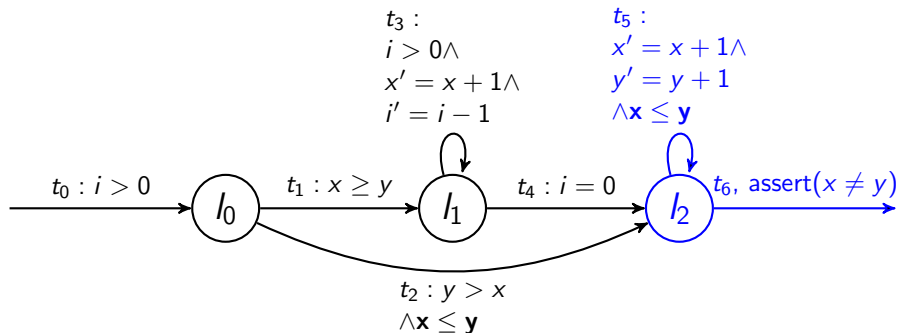


Narrow on $\{l_2\}$

Add $x \leq y$ to t_2

Add $x \leq y$ to t_5

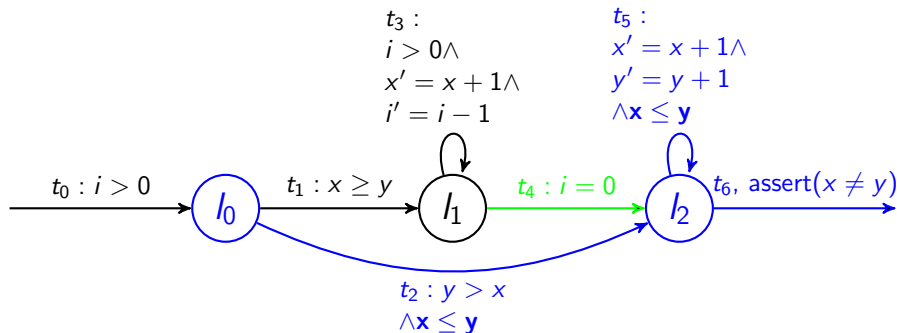
Example execution



CheckSafe on $\{l_2\}$ for $x \neq y$

Call CondSafe, get $y > x$ instead of $x > y$ as precondition

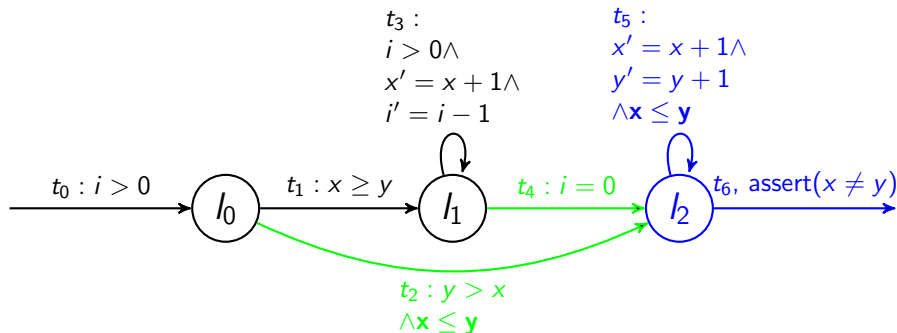
Example execution



CheckSafe on $\{l_0\}$ for $y > x$

t_2 does already imply $y > x$

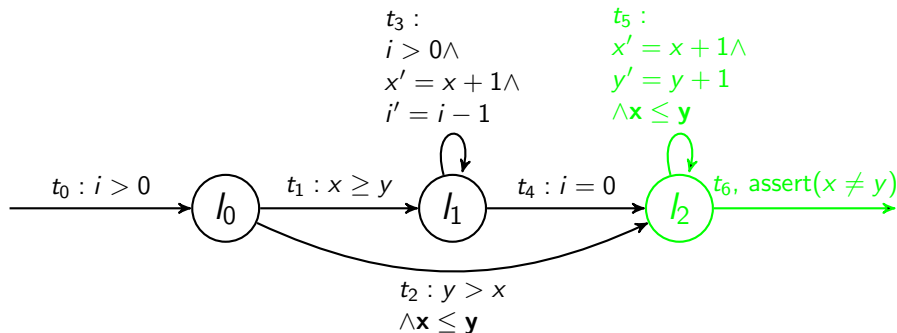
Example execution



CheckSafe on $\{l_0\}$ for $y > x$

Path is safe for $y > x$

Example execution



CheckSafe on $\{l_0\}$ for $y > x$

Program is safe for $x \neq y$



Brockschmidt, Marc and Larraz, Daniel and Oliveras, Albert and Rodriguez-Carbonell, Enric and Rubio, Albert (2015)

Compositional Safety Verification with Max-SMT

Proceedings of FMCAD'15

The End