universität innsbruck

supported by FWF project I 5943-N

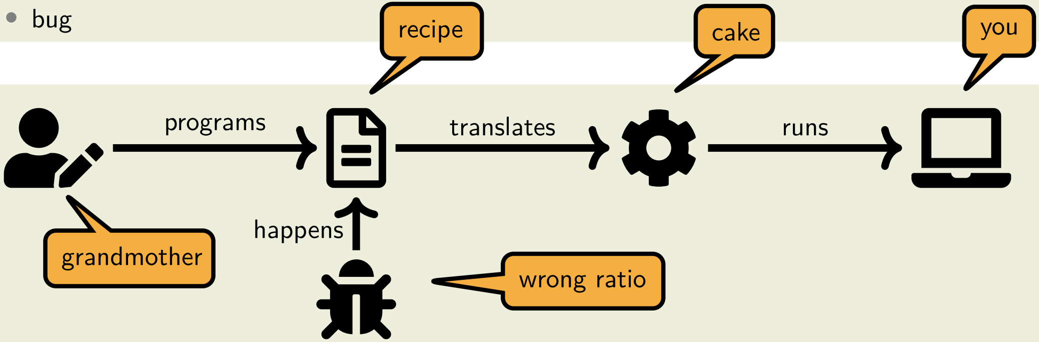
Automated Analysis of Logically Constrained Programs

Jonas Schöpf

MIP Seminar
8 January 2025

Key Concepts

- programmer
- code (program)
- executable (software)
- computer
- bug



Why do Bugs Matter?

- critical software is everywhere
- aviation, medicine, nuclear power plants, . . .
- dangerous and expensive

Figure: esa.int

Figure: howtogeek.com

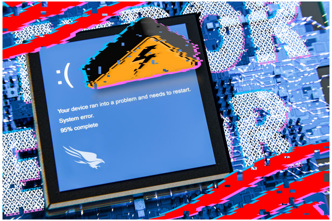
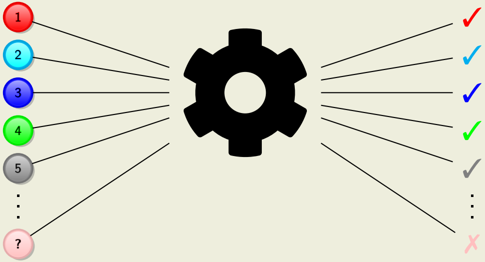


Figure: howtogeek.com

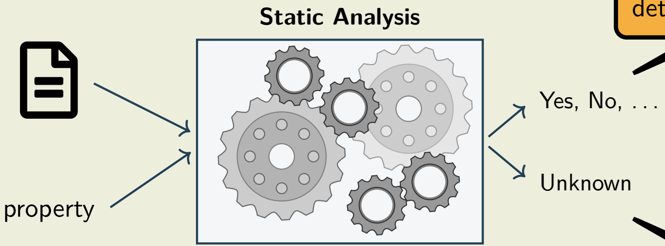
How to Avoid Bugs?

- program carefully? skilled programmers?
- bugs are not obvious
- complex (million lines of code)
- bugs may not be detected by testing

Testing?



Static Program Analysis



detailed argumentation

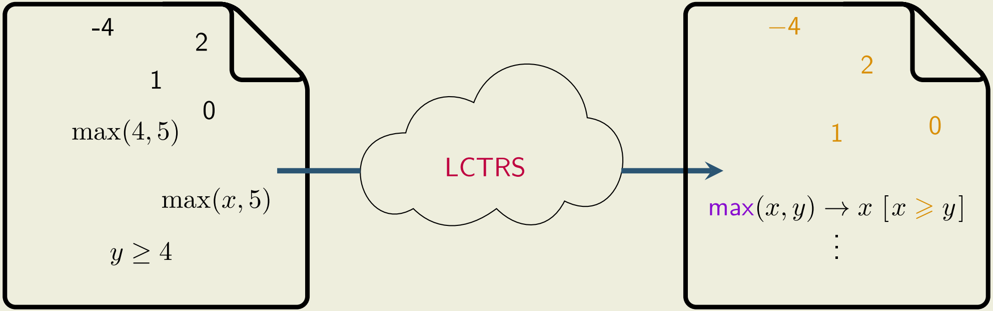
no complete way exists

Properties

termination	does the program finish in a finite amount of time?
confluence	does the program compute unique solutions?
complexity	how long does the program run?
. . .	

Computational Model

- difficult on real programs
- use computational model
- many (formal) methods



-4

2

1

0

$\max(4,5)$

$\max(x,5)$

$y \geq 4$

LCTRS

-4

2

1

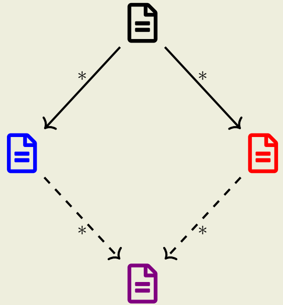
0

$\mathsf{max}(x,y) \to x \; [x \geqslant y]$

$\vdots$

Confluence

- no general way
- test this for all computations?
- extract critical parts



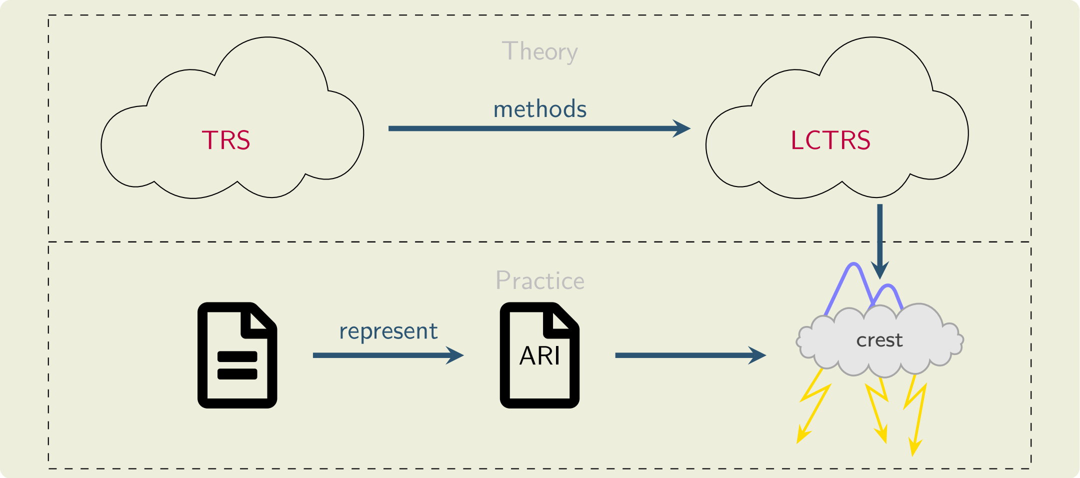
Term Rewrite System (TRS)

- many methods for confluence
- decades of research
- difficult for real programs

Logically Constrained TRS (LCTRS)

- extension of TRS
- built-in computations (including solvers)
- not many methods
- re-use existing knowledge

My Research



Example

computation rules

$$\mathsf{max}(x,y) \to x\ [x \geqslant y] \qquad \mathsf{max}(x,y) \to y\ [y \geqslant x] \qquad \mathsf{max}(x,y) \to \mathsf{max}(y,x)$$

critical parts

$$x \approx y\ [y \geqslant x \wedge x \geqslant y] \qquad \underbrace{x \approx \mathsf{max}(y,x)\ [x \geqslant y]} \qquad y \approx \mathsf{max}(y,x)\ [y \geqslant x]$$

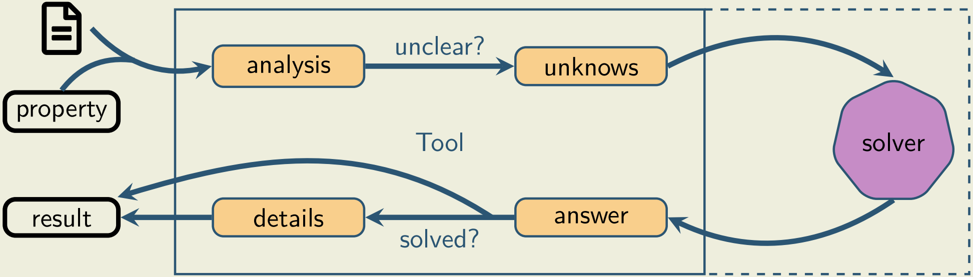
confluence criterion

$$\ldots \qquad x \approx \mathsf{max}(y,x)\ [x \geqslant y] \to x \approx x\ [x \geqslant y] \qquad \ldots$$

$\Longrightarrow$ confluence

Simplified Automation

- tedious & error-prone
- complex checks



Confluence Competititon

- annual competition since 2012
- LCTRS category 2024
- 1st place for crest

Confluence Experiments on 107 Examples

tool	✓	✗	solved	time
CRaris	58	0	54 %	14 s
crest	72	26	92 %	197 s
Ctrl	54	0	50 %	18 s

Summary

- programs have bugs & testing may not suffice
- program analysis with computational model (LCTRSs)
- methods for confluence of LCTRSs
- push-button automation

Thank you for your attention!