

## **Florian Frohn**

# Automatic Complexity Analysis of Programs

Supervisor: Jürgen Giesl



Informatik II

LuFG

**Term Rewrite System (TRS)**  $\mathcal{R}$ 

 $plus(s(x), y) \rightarrow plus(x, s(y))$  $plus(0, y) \rightarrow y$ 

Integer Transition System (ITS)  $\mathcal{I}$  $plus(x, y) \to plus(x - 1, y + 1) | x > 0$  $plus(x, y) \rightarrow y \mid x = 0$ 

Java Program  $\mathcal{J}$ 

 $Java \rightarrow ITS$ 

# int plus(int x, int y) { while (x > 0) { X−−; Y++; return y;

- execute program symbolically
- $\sim$  symbolic evaluation graph
- translate graph to ITS
- apply techniques for ITSs
- translates  ${\mathcal J}$  to  ${\mathcal I}$
- $\sim \mathcal{O}(\mathbf{x})$

### **Recursive ITS** $\rightarrow$ **ITS**

analyze non-recursive sub-ITSs independently

approximate runtime and size of result

replace calls to analyzed program parts with obtained bounds

• currently restricted to  $\mathbb{N}$ 

FroCoS '17

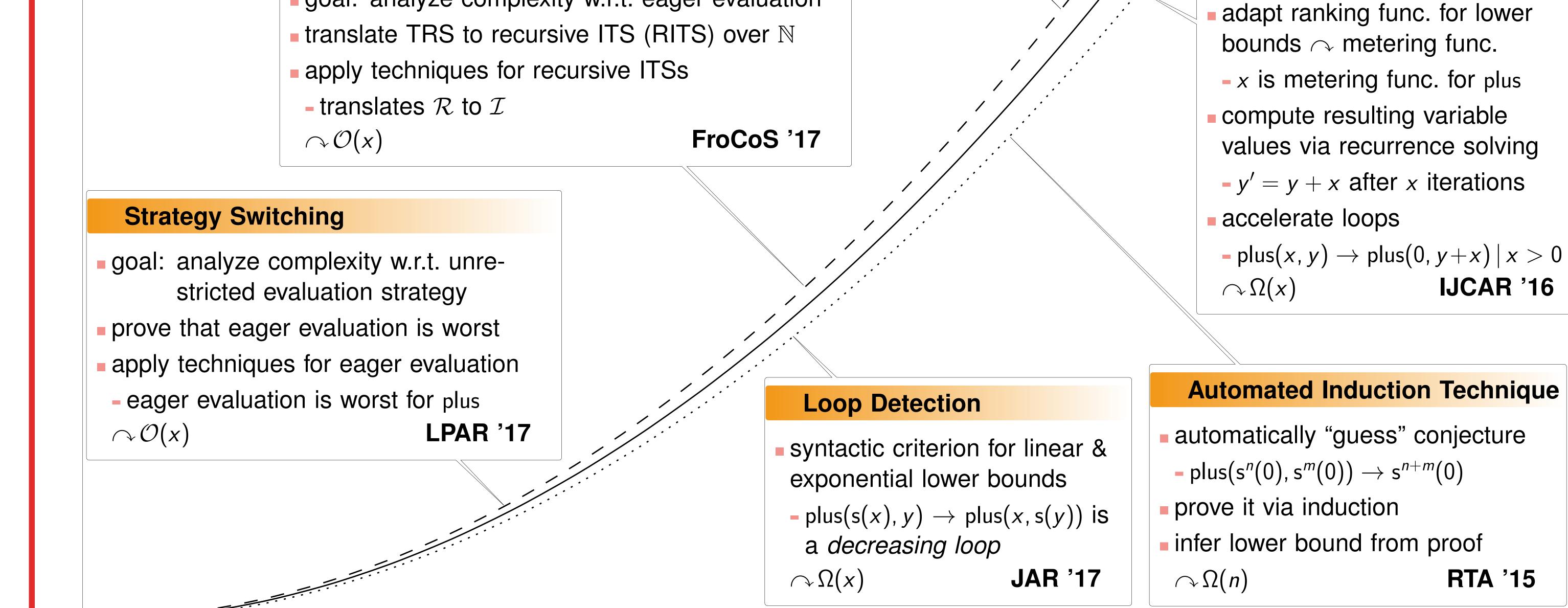
#### **TRS** $\rightarrow$ **Recursive ITS**

goal: analyze complexity w.r.t. eager evaluation

#### LoAT

iFM '17

**RTA** '15



Applic	ations
--------	--------

 worst case upper bound
 worst case runtime
 worst case lower bound

# DARPA STAC program [4]

AProVE [1] and LoAT [3] are crucial parts of the CAGE toolchain [2]

used to find or prove the absence of DoS vulnerabilities in Java programs

[1] AProVE – http://aprove.informatik.rwth-aachen.de

[2] CAGE – http://www.draper.com/news/draper-s-cage-could-spot-code-vulnerable-denial-service-attacks

[3] LoAT – https://github.com/aprove-developers/LoAT

[4] Space/Time Analysis for Cybersecurity (STAC) – https://www.darpa.mil/program/space-time-analysis-for-cybersecurity