

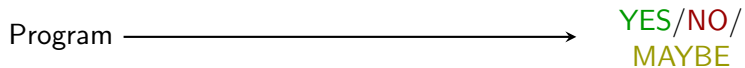
# Analysis of Arithmetic PROLOG Programs using Abstract Interpretation

Alexander Weinert

Master Thesis Presentation

June 22, 2015

# Roadmap



Question: Do all evaluations of the program terminate?

## PROLOG - Introduction

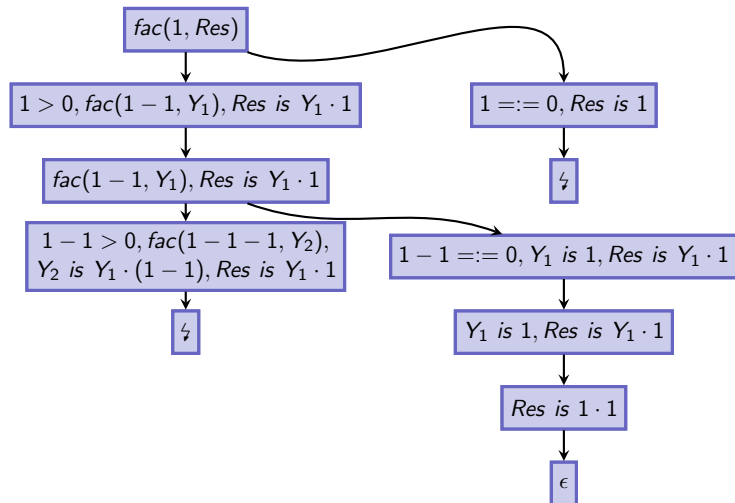
```
fac(X) =  
    if X > 0 then Y1 = fac(X-1), return Y1 * X  
    if X == 0 then return 1
```

```
fac(X, Y) :- X > 0, fac(X - 1, Y1), Y is Y1 * X.  
fac(X, Y) :- X == 0, Y is 1.
```

## PROLOG - Evaluation

`fac(X, Y) :- X > 0, fac(X - 1, Y1), Y is Y1 * X.`

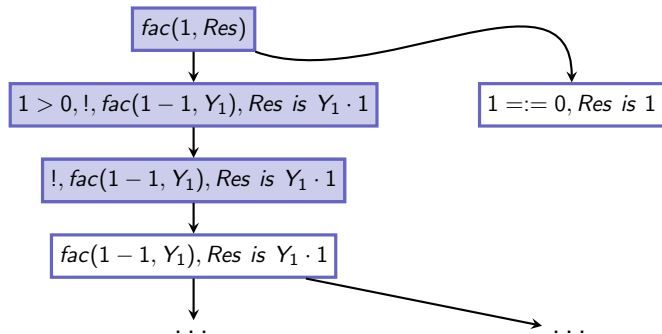
`fac(X, Y) :- X == 0, Y is 1.`



## PROLOG - Cut

```
fac(X, Y) :- X > 0, !, fac(X - 1, Y1), Y is Y1 * X.  
fac(X, Y) :- X == 0, Y is 1.
```

# PROLOG - Cut

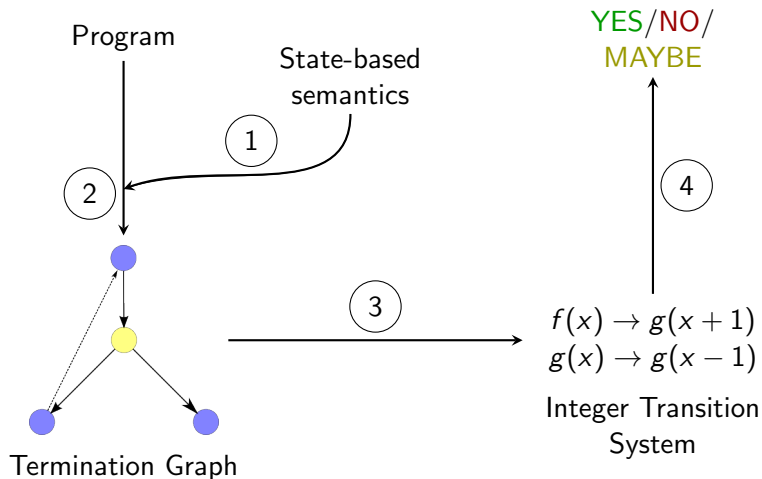


# Termination

Given: PROLOG Program, some query template

Question: For all queries matching the template: Does the inference of the query on the program eventually terminate?

# Roadmap





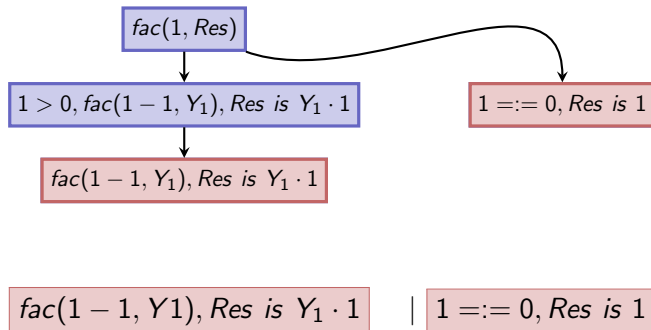
# From Trees to States

- ▶ PROLOG: Tree-based semantics
- ▶ Well-known techniques: State-based semantics

Solution: State-based semantics for PROLOG  
(Linear Operational Semantics, Ströder et al., 2012)

- ▶ Basic Idea: Leaves of tree describe state of inference
- ▶ Use front of tree as state

## From Trees to States



## From Trees to States

$\text{fac}(X, Y) \quad :- \quad X > 0, !, \text{fac}(X - 1, Y_1), Y \text{ is } Y_1 * X.$

$\text{fac}(X, Y) \quad :- \quad X ::= 0, Y \text{ is } 1.$

$\text{fac}(1, \text{Res})$

↓

$1 > 0, !, \text{fac}(1 - 1, Y_1), \text{Res is } Y_1 \cdot 1 \quad | \quad 1 ::= 0, \text{Res is } 1$

## From Trees to States

$1 > 0$ ,  $!$ ,  $fac(1 - 1, Y_1)$ ,  $Res$  is  $Y_1 \cdot 1$  |  $1 ::= 0$ ,  $Res$  is  $1$

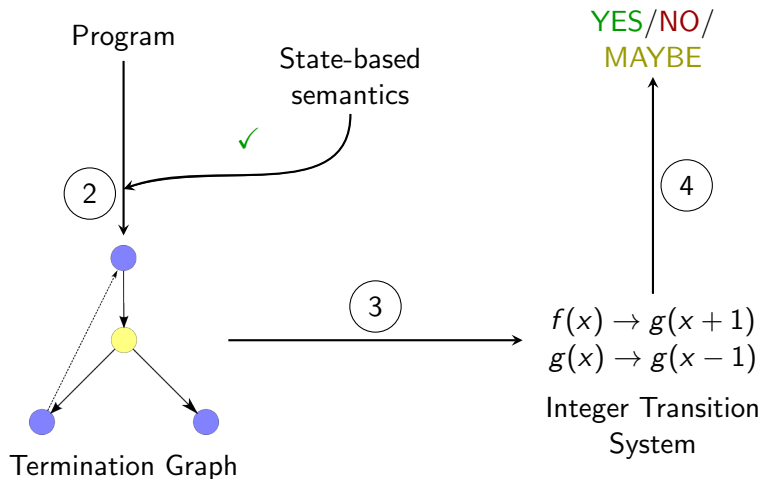
↓

$!$ ,  $fac(1 - 1, Y_1)$ ,  $Res$  is  $Y_1 \cdot 1$  |  $1 ::= 0$ ,  $Res$  is  $1$

↓

$fac(1 - 1, Y_1)$ ,  $Res$  is  $Y_1 \cdot 1$

# Roadmap



# From Programs to Graphs

Given: Some Program, some query template

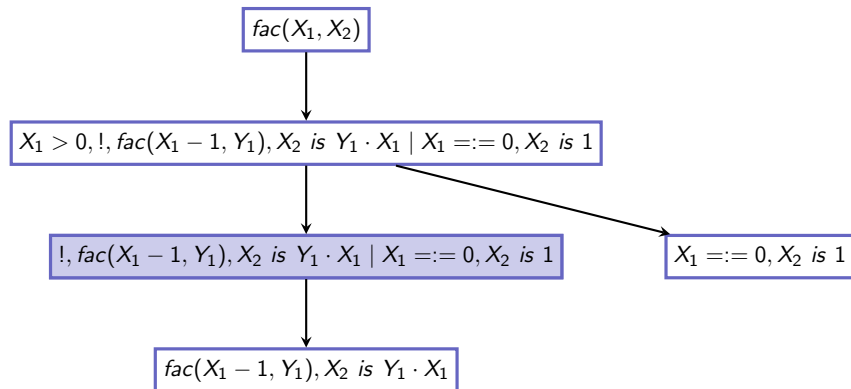
Goal: Finite representation of all possible inferences

Idea: Represent set of runs as graph

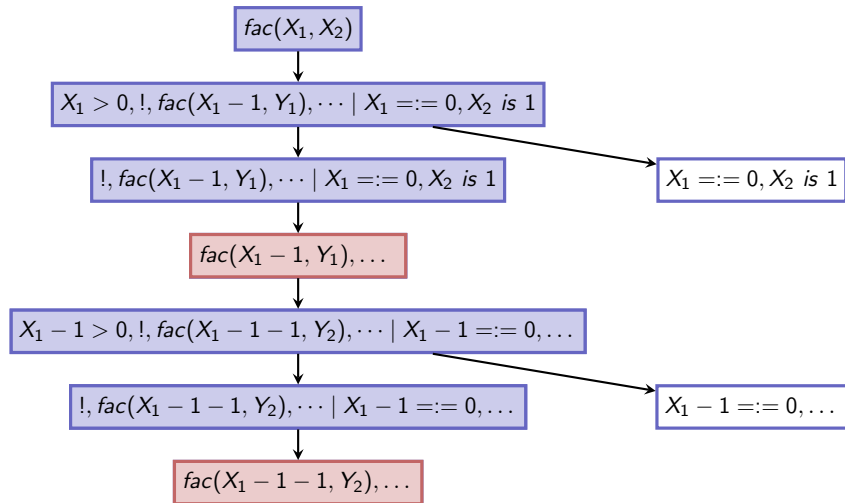
## From Programs To Graphs - Starting State

$\text{fac}(X, Y) \quad :- \quad X > 0, \text{ !}, \text{ fac}(X - 1, Y_1), Y \text{ is } Y_1 * X.$

$\text{fac}(X, Y) \quad :- \quad X == 0, Y \text{ is } 1.$

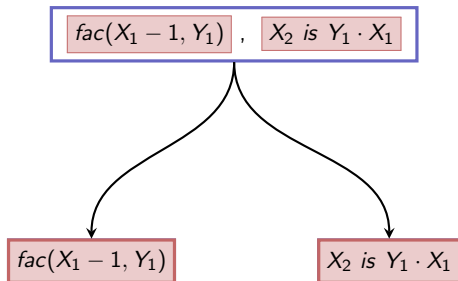


# From Programs To Graphs - Nonterminating Construction

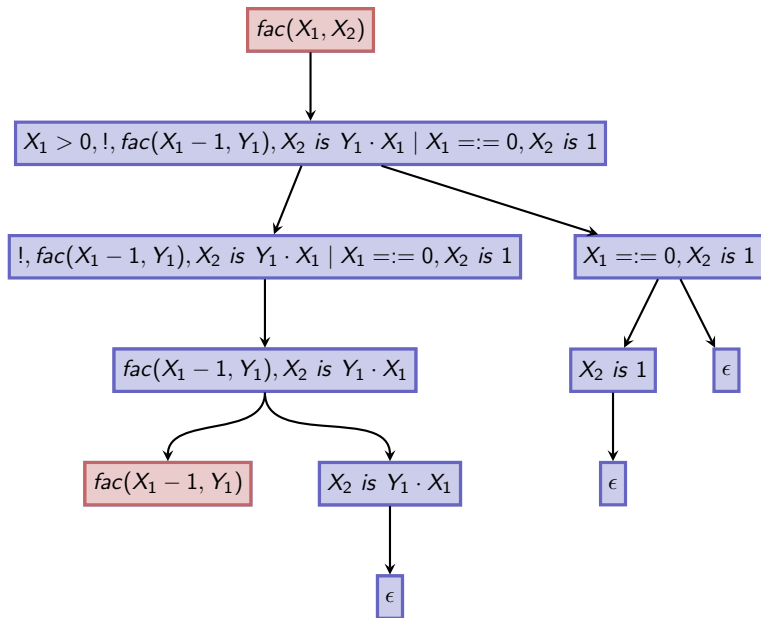




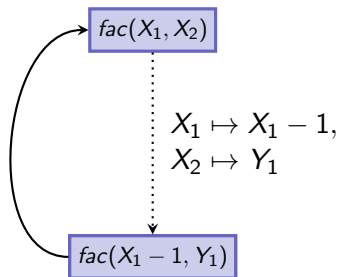
## From Programs To Graphs - Split rule



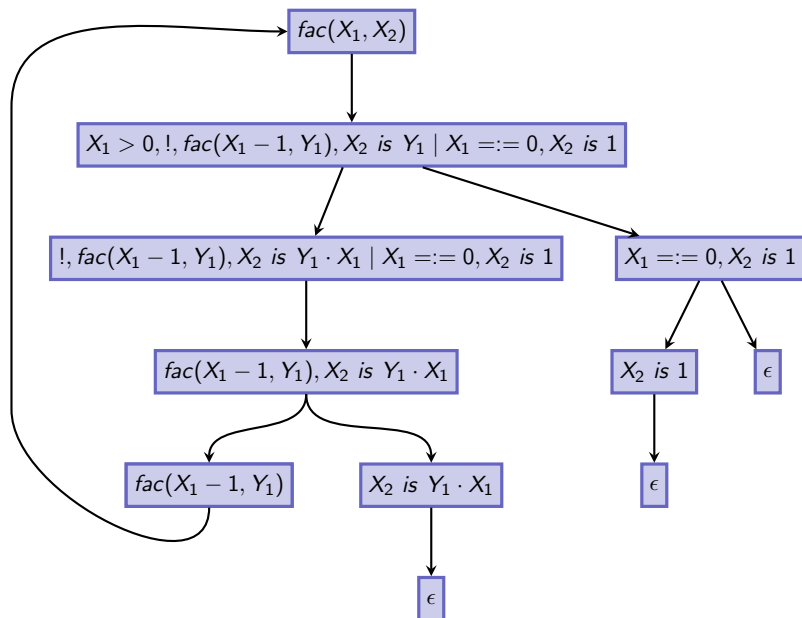
## From Programs To Graphs - Split rule



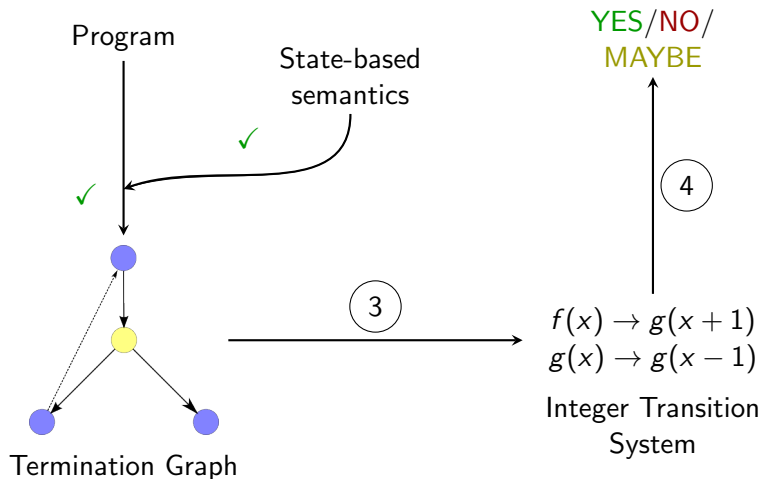
## From Programs To Graphs - Instance Rule



# From Programs To Graphs - Final Result



# Roadmap



# Integer Transition Systems

Integer Transition System:

$$f(x) \rightarrow f(x + 1) \quad | \quad x < 0$$

# From Graphs to Transition Systems

Given: Some Termination Graph

Goal: Integer Transition System that terminates if all runs described by the Termination Graph terminate

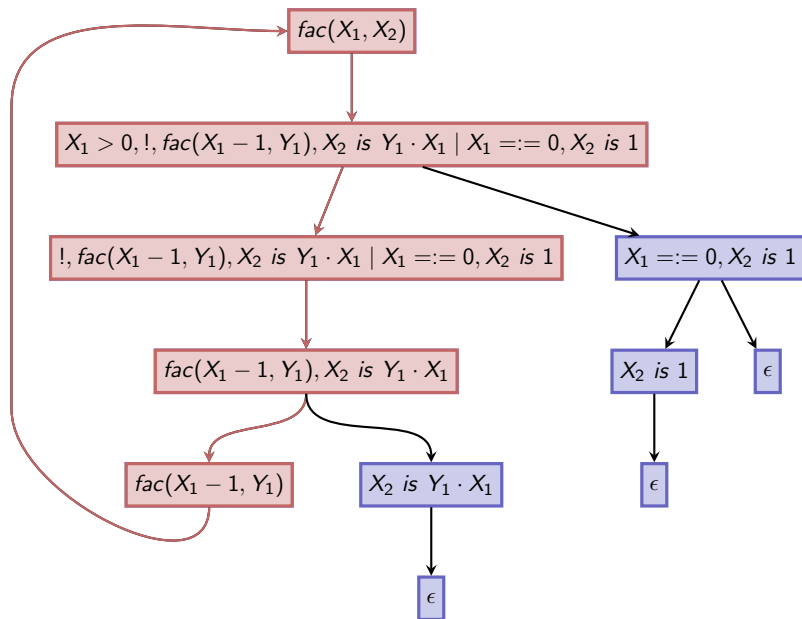
Idea: Encode graph locally, node by node

# From Graphs to Transition Systems

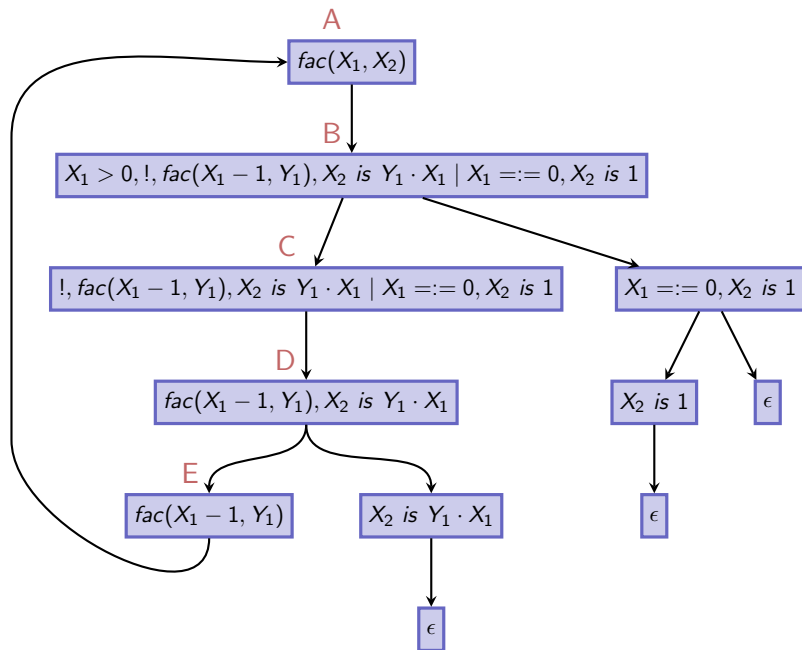
Paths in Graph  $\approx$  Evaluations



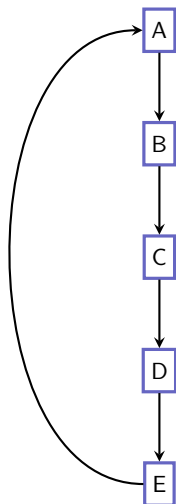
# From Graphs to Transition Systems



# From Graphs to Transition Systems



## From Graphs to Transition Systems

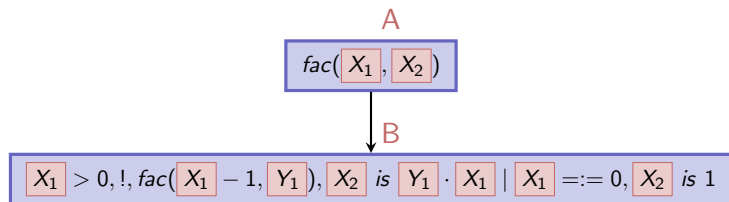


$A \rightarrow B$   
 $B \rightarrow C$   
 $C \rightarrow D$   
 $D \rightarrow E$   
 $E \rightarrow A$

↓

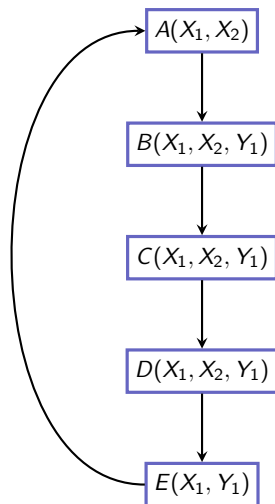
$A \rightarrow A$

# From Graphs to Transition Systems



$$A(X_1, X_2) \rightarrow B(X_1, X_2, Y_1)$$

## From Graphs to Transition Systems

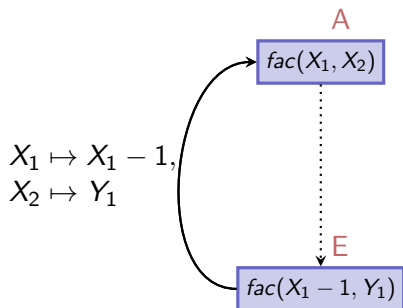


$A(X_1, X_2)$	$\rightarrow$	$B(X_1, X_2, Y_1)$
$B(X_1, X_2, Y_1)$	$\rightarrow$	$C(X_1, X_2, Y_1)$
$C(X_1, X_2, Y_1)$	$\rightarrow$	$D(X_1, X_2, Y_1)$
$D(X_1, X_2, Y_1)$	$\rightarrow$	$E(X_1, Y_1)$
$E(X_1, Y_1)$	$\rightarrow$	$A(X_1, X_2)$

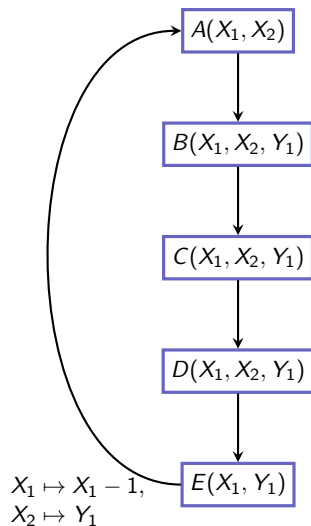
$\downarrow$

$A(X_1, X_2)$	$\rightarrow$	$A(X_1, Y_1)$
---------------	---------------	---------------

## From Programs To Graphs - Instance Rule



## From Graphs to Transition Systems

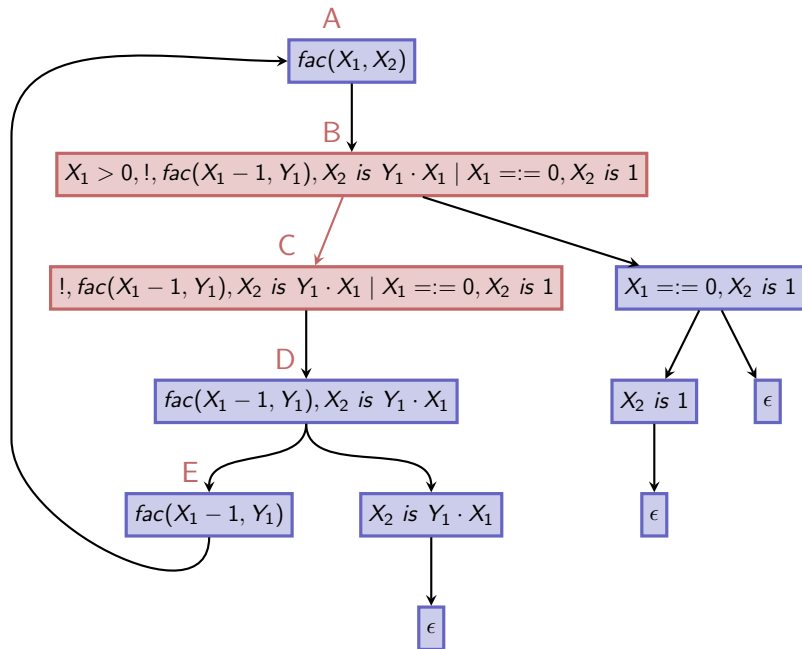


$A(X_1, X_2) \rightarrow B(X_1, X_2, Y_1)$   
 $B(X_1, X_2, Y_1) \rightarrow C(X_1, X_2, Y_1)$   
 $C(X_1, X_2, Y_1) \rightarrow D(X_1, X_2, Y_1)$   
 $D(X_1, X_2, Y_1) \rightarrow E(X_1, Y_1)$   
 $E(X_1, Y_1) \rightarrow A(X_1 - 1, Y_1)$

↓

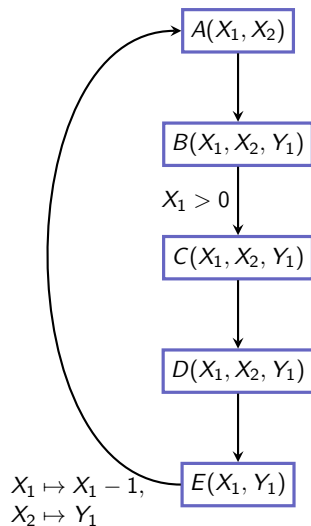
$A(X_1, X_2) \rightarrow A(X_1 - 1, Y_1)$

# From Graphs to Transition Systems





# From Graphs to Transition Systems

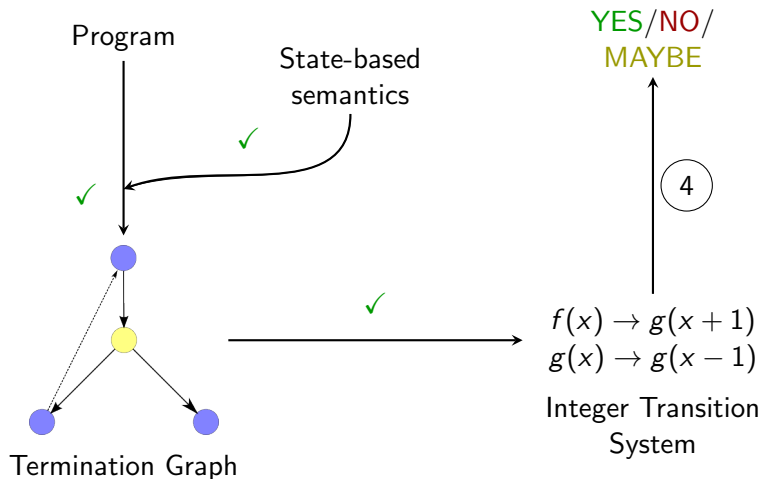


$A(X_1, X_2)$	$\rightarrow$	$B(X_1, X_2, Y_1)$	
$B(X_1, X_2, Y_1)$	$\rightarrow$	$C(X_1, X_2, Y_1)$	$  X_1 > 0$
$C(X_1, X_2, Y_1)$	$\rightarrow$	$D(X_1, X_2, Y_1)$	
$D(X_1, X_2, Y_1)$	$\rightarrow$	$E(X_1, Y_1)$	
$E(X_1, Y_1)$	$\rightarrow$	$A(X_1 - 1, Y_1)$	

↓

$A(X) \rightarrow A(X - 1) \quad | X > 0$

# Roadmap

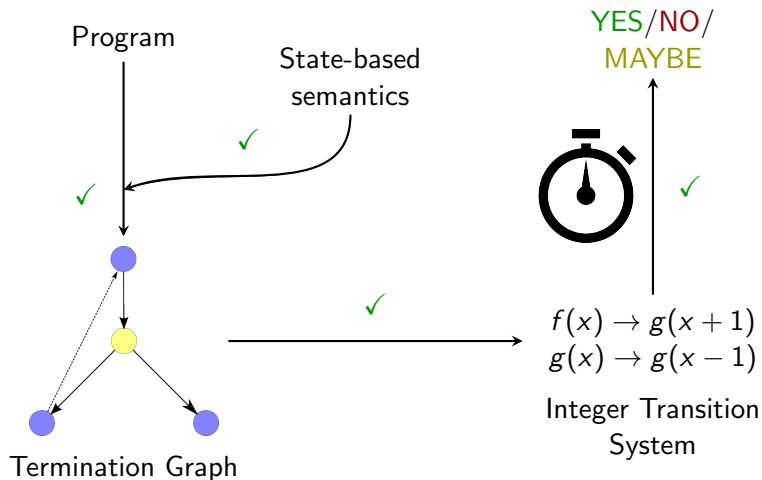


# Termination of Integer Transition Systems

Well-studied problem

Use known techniques to show termination  
(Transition Invariants, Podelski and Rybalchenko, 2004)

# Roadmap



Stopwatch: <http://icons8.com/>

## Results - Compared Approaches

Term Rewriting with Argument Filter Directly from Program  
(Termination Proofs, Schneider-Kamp et al., 2009)

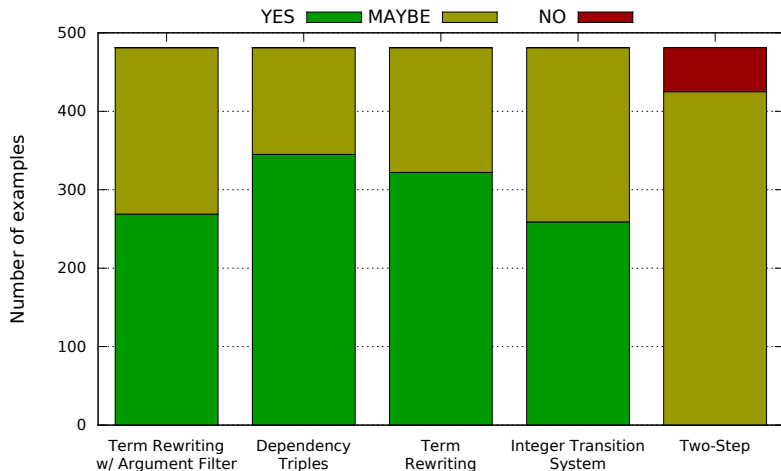
Dependency Triples Graph Construction  
(Dependency Triples, Ströder et al., 2011)

Term Rewriting Graph Construction  
(Symbolic Evaluation Graphs, Giesl et al., 2012)

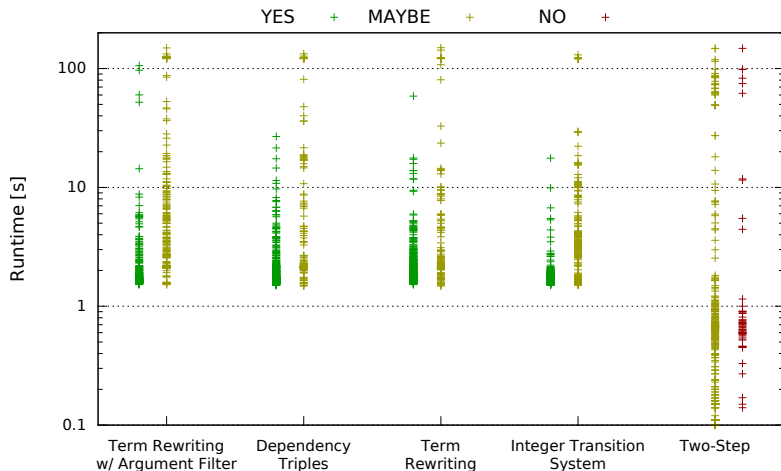
Integer Transition Systems This approach  
(Analysis of Arithmetic Prolog Programs, Weinert, 2015)

Two-Step Constraint Satisfaction Problem  
(Non-termination Analysis, Voets et al., 2011)

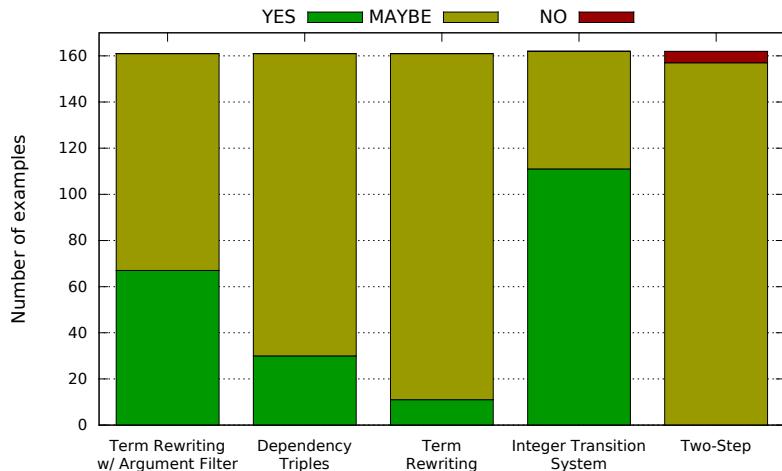
## Results - Logic Benchmarks - Power



# Results - Logic Benchmarks - Runtime

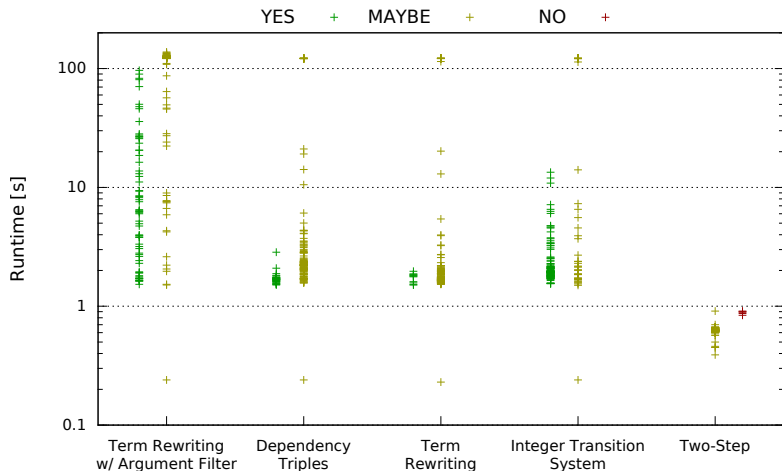


# Results - Numerical Benchmarks - Power





# Results - Numerical Benchmarks - Runtime



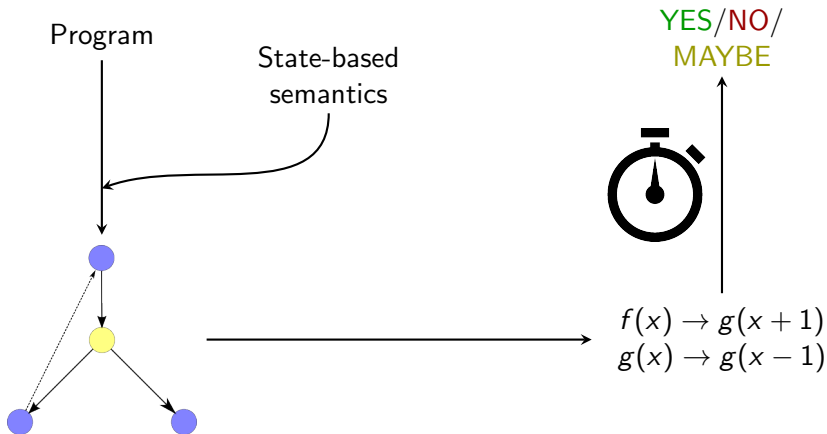
## Contributions - Theoretical

- ▶ Extended construction of Termination Graphs, taking arithmetic comparisons and evaluations into account
- ▶ Developed new construction of Integer Transition System from Termination Graphs
- ▶ Extended abstract state to store arithmetic knowledge
- ▶ Separated abstract semantics and termination analysis
- ▶ Proved soundness of all steps of the construction

## Contributions - Practical

- ▶ Implemented extension of construction of Termination Graphs, optimization through SMT solver
- ▶ Implemented construction of Integer Transition Systems from Termination Graphs
- ▶ Added 162 numerical benchmarks to benchmark suite
- ▶ Performed experiments comparing this approach to existing ones

<http://alexanderweinert.net/talks>  
alexander.weinert@rwth-aachen.de



Stopwatch: <http://icons8.com/>