Shape Analysis via Monotonic Abstraction

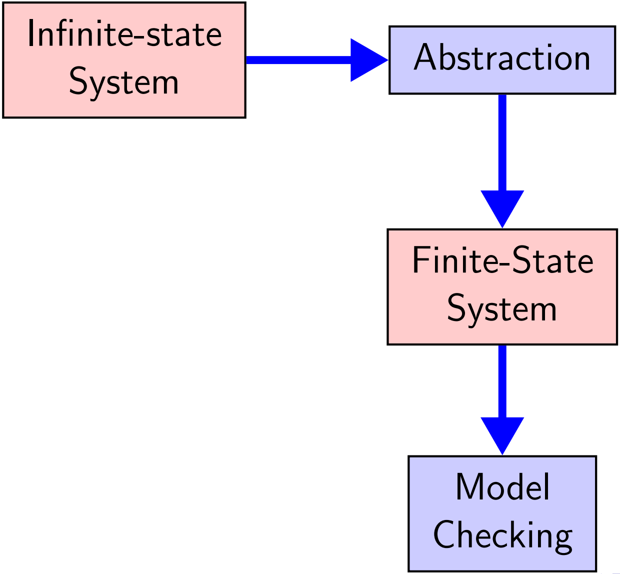
Parosh Aziz Abdulla

Uppsala University

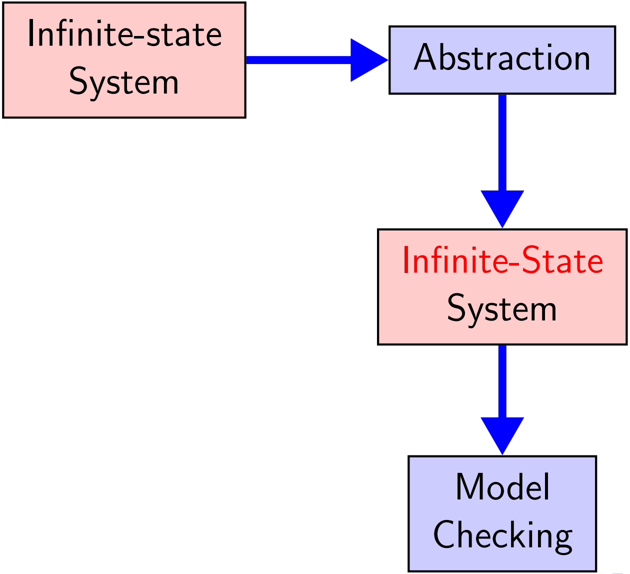
February 9, 2010

(Joint work with Ahmed Bouajjani, Jonathan Cederberg, Fédéric Haziza and Ahmed Rezine.)

Model Checking+Abstraction



Model Checking+Abstraction



Monotonic Transition Systems

Monotonic Transition System

- $\mathcal{T} = (S, \longrightarrow, \preceq)$
- S: (infinite) set of configurations
- $\longrightarrow$: transition relation
- $\preceq$: preorder on S

Monotonic Transition Systems

Monotonic Transition System

- $\mathcal{T} = (S, \longrightarrow, \preceq)$
- S: (infinite) set of configurations
- $\longrightarrow$: transition relation
- $\preceq$: preorder on S

Monotonicity

$c_1 \longrightarrow c_2$

$\curlyvee$ (rotated $\preceq$)

c_3

Monotonic Transition Systems

Monotonic Transition System

- $\mathcal{T} = (S, \longrightarrow, \preceq)$
- S: (infinite) set of configurations
- $\longrightarrow$: transition relation
- $\preceq$: preorder on S

Monotonicity

$$\begin{array}{ccc} c_1 & \longrightarrow & c_2 \\ \preceq & & \preceq \\ c_3 & \longrightarrow & c_4 \end{array}$$

Monotonic Transition Systems

Monotonic Transition System

- $\mathcal{T} = (S, \longrightarrow, \preceq)$
- S: (infinite) set of configurations
- $\longrightarrow$: transition relation
- $\preceq$: preorder on S

Monotonicity

$c_1 \longrightarrow c_2$

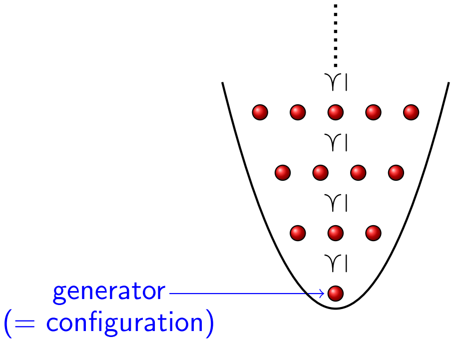
$\curlyvee\mid \qquad \curlyvee\mid$

$c_3 \longrightarrow c_4$

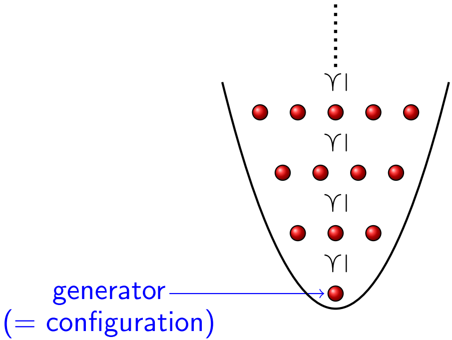
Examples

- Petri Nets.
- Lossy Channel Systems.
- Timed Petri Nets.
- Multiset Rewriting Systems.
- Broadcast Protocols.
- etc.

Upward-Closed Sets (UC)



Upward-Closed Sets (UC)



Why UC?

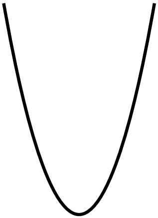
- Bad sets of states are UC
  - safety properties = reachability of UC
- Uniquely characterized by generator
  - simple representation = minimal element

Monotonicity and Upward Closedness

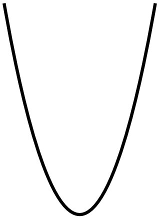
Monotonicity implies UC is closed under *Pre*

Monotonicity and Upward Closedness

Monotonicity implies UC is closed under *Pre*



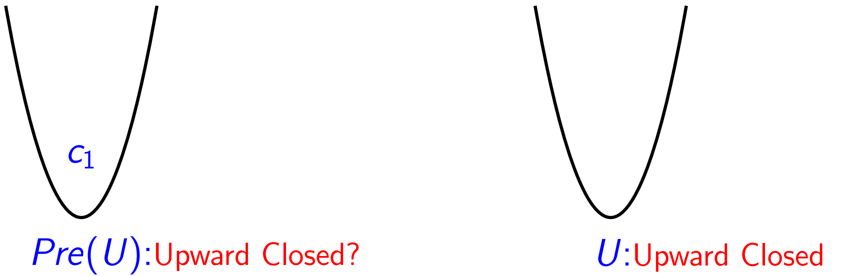
$Pre(U)$:Upward Closed?



U:Upward Closed

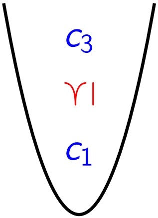
Monotonicity and Upward Closedness

Monotonicity implies UC is closed under *Pre*

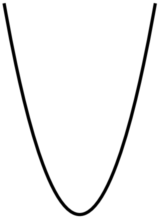


Monotonicity and Upward Closedness

Monotonicity implies UC is closed under *Pre*



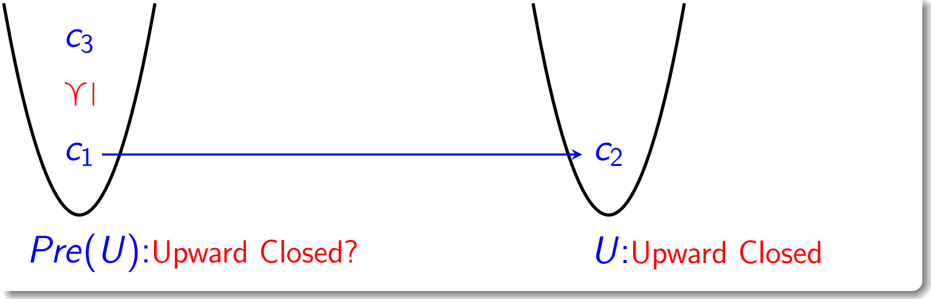
$Pre(U)$:Upward Closed?



U:Upward Closed

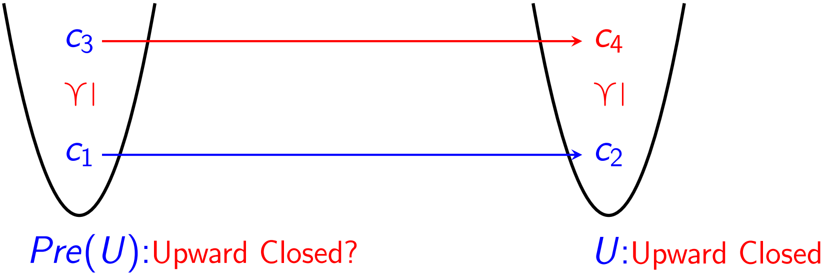
Monotonicity and Upward Closedness

Monotonicity implies UC is closed under *Pre*



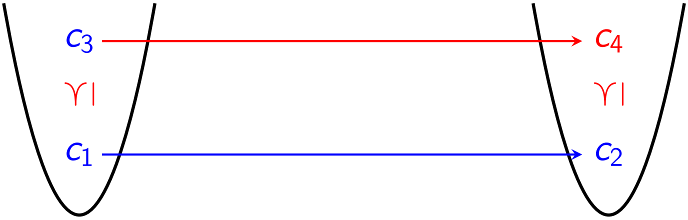
Monotonicity and Upward Closedness

Monotonicity implies UC is closed under *Pre*



Monotonicity and Upward Closedness

Monotonicity implies UC is closed under *Pre*



$Pre(U)$: Upward Closed? Yes

U: Upward Closed

Monotonic Abstraction

Problem

- When transition system not monotonic

Monotonic Abstraction

Problem

- When transition system not monotonic

Solution: Monotonic Abstraction

- Force monotonicity !
- Over-Approximation of non-monotonic transitions

Monotonic Abstraction

Problem

- When transition system not monotonic

Solution: Monotonic Abstraction

- Force monotonicity !
- Over-Approximation of non-monotonic transitions

c_1 c_2

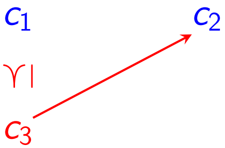
Monotonic Abstraction

Problem

- When transition system not monotonic

Solution: Monotonic Abstraction

- Force monotonicity !
- Over-Approximation of non-monotonic transitions



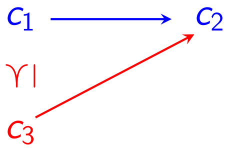
Monotonic Abstraction

Problem

- When transition system not monotonic

Solution: Monotonic Abstraction

- Force monotonicity !
- Over-Approximation of non-monotonic transitions



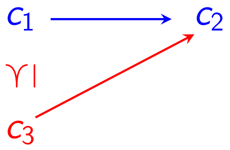
Monotonic Abstraction

Problem

- When transition system not monotonic

Solution: Monotonic Abstraction

- Force monotonicity !
- Over-Approximation of non-monotonic transitions



Examples

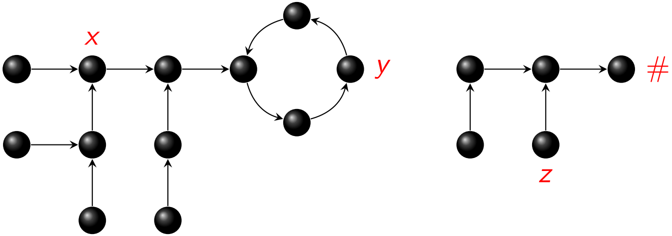
- Parameterized Systems.
- Shape Analysis.

Shape Analysis: Singly Linked Lists

Transition System $= (S, \longrightarrow, \preceq)$

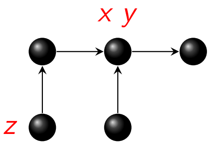
Configuration

- graph
  - node: cell
  - edge: successor
  - pointers: x, y, z, $\#$



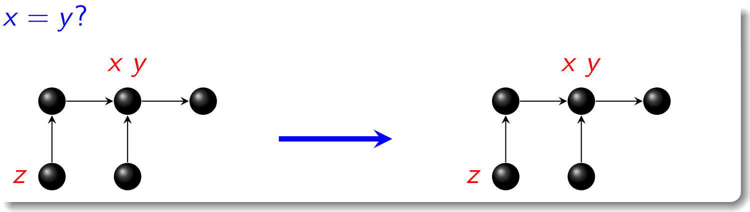
Transitions

$x = y?$



t

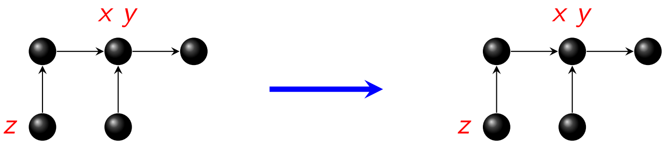
Transitions



t

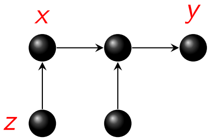
Transitions

$x = y?$



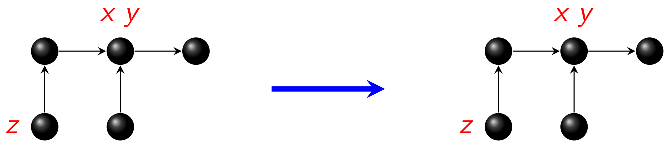
t

$x = y?$



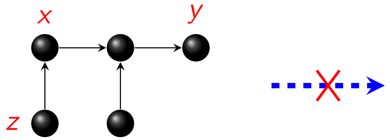
Transitions

$x = y?$



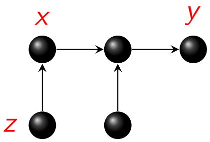
t

$x = y?$



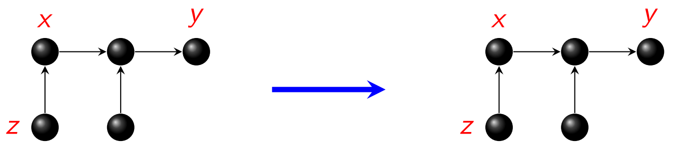
Transitions

$x \neq y$?



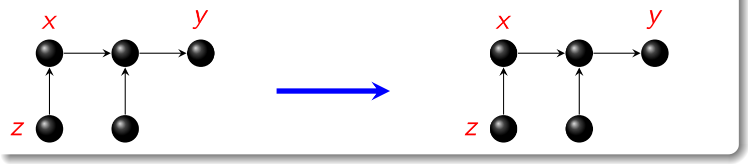
Transitions

$x \neq y$?



Transitions

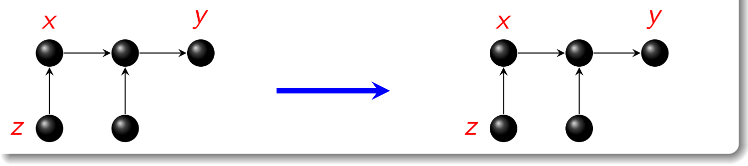
$x \neq y?$



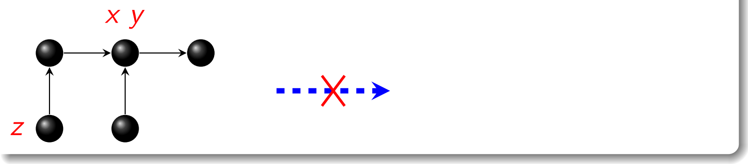
$x \neq y?$

Transitions

$x \neq y?$

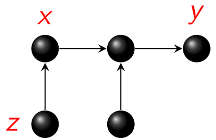


$x \neq y?$



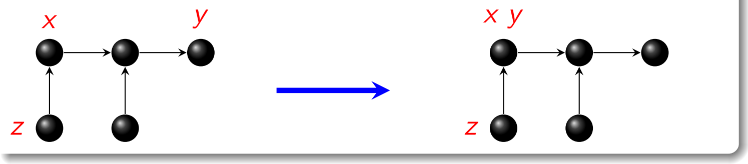
Transitions

$y := x$



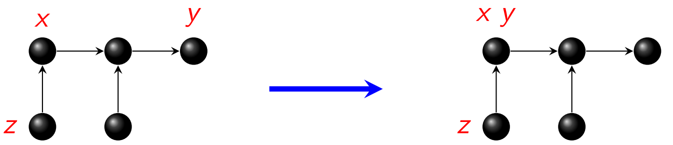
Transitions

$y := x$

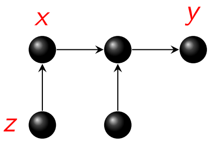


Transitions

$y := x$

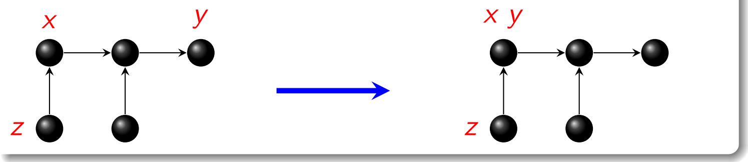


$y := x \cdot next$

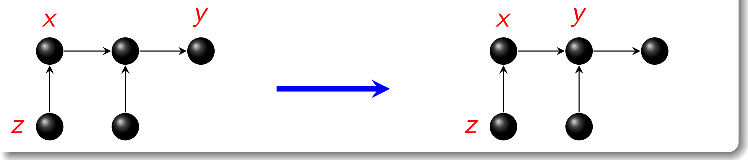


Transitions

$y := x$

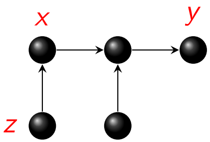


$y := x \cdot next$



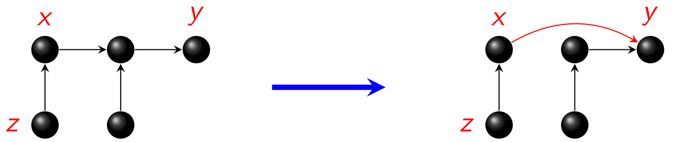
Transitions

$x \cdot next := y$



Transitions

$x \cdot next := y$



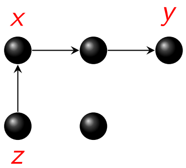
Ordering on Graphs

Variable Deletion

Ordering on Graphs

Variable Deletion

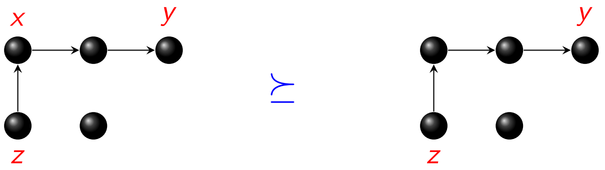
Variable Deletion



Ordering on Graphs

Variable Deletion

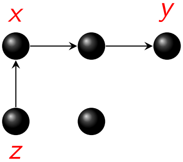
Variable Deletion



Ordering on Graphs

Edge Deletion

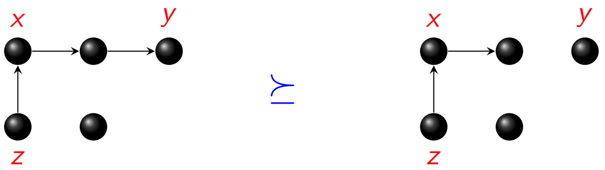
Edge Deletion



Ordering on Graphs

Edge Deletion

Edge Deletion



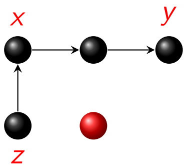
Ordering on Graphs

Vertex Deletion

Isolated Vertex

- no label
- no incoming/outgoing arcs

Vertex Deletion



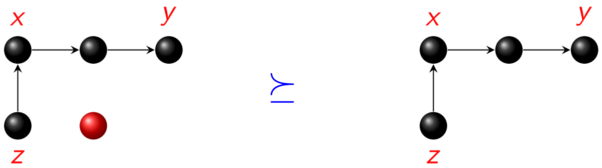
Ordering on Graphs

Vertex Deletion

Isolated Vertex

- no label
- no incoming/outgoing arcs

Vertex Deletion



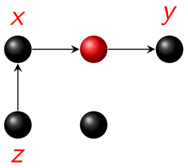
Ordering on Graphs

Contraction

SimpleVertex

- no label
- one incoming arc
- one outgoing arc

Contraction



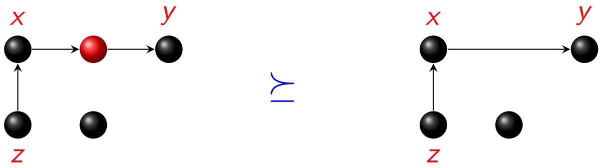
Ordering on Graphs

Contraction

SimpleVertex

- no label
- one incoming arc
- one outgoing arc

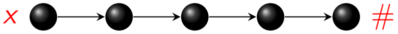
Contraction



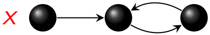
Bad Configurations

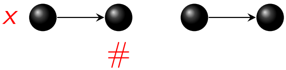
Well-formed Lists

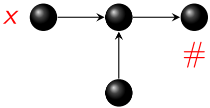
Well-Formed List:



Badly-Formed Lists:





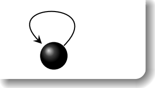


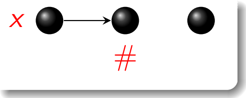
Bad Configurations

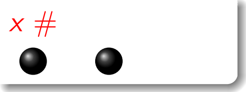
Well-formed Lists

Bad Patetrns:

- minimal elements
- finitely many
- upward closure = all badly-formed lists



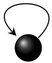




Bad Configurations

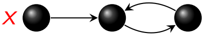
Well-formed Lists

Bad pattern



$\preceq$

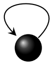
Bad configuration



Bad Configurations

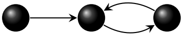
Well-formed Lists

Bad pattern



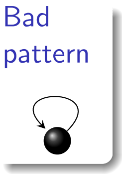
$\preceq$

Bad configuration

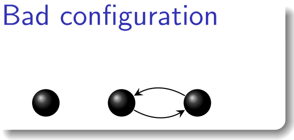


Bad Configurations

Well-formed Lists



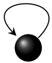
$\preceq$



Bad Configurations

Well-formed Lists

Bad pattern



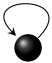
$\preceq$

Bad configuration

Bad Configurations

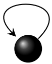
Well-formed Lists

Bad pattern



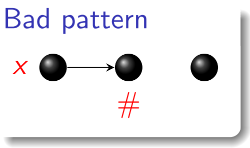
$\preceq$

Bad configuration

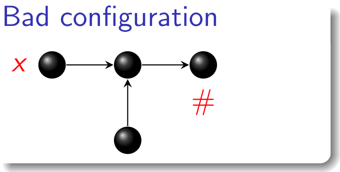


Bad Configurations

Well-formed Lists

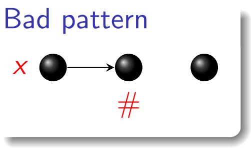


$\preceq$

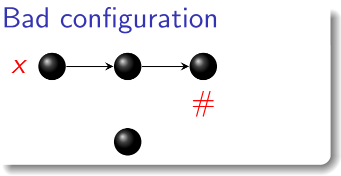


Bad Configurations

Well-formed Lists

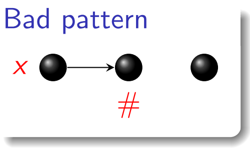


$\preceq$

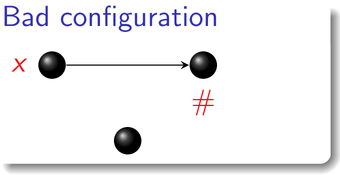


Bad Configurations

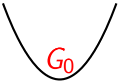
Well-formed Lists



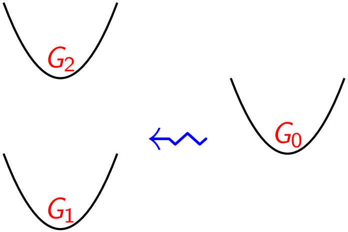
$\preceq$



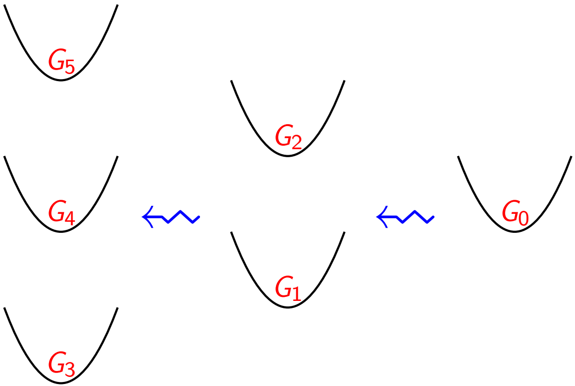
Backward Reachability Analysis



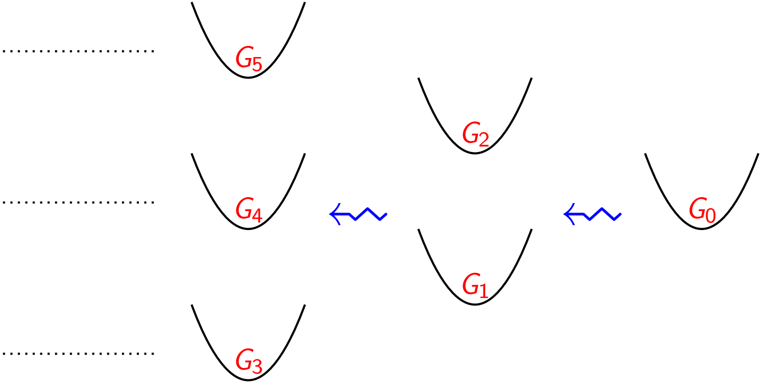
Backward Reachability Analysis



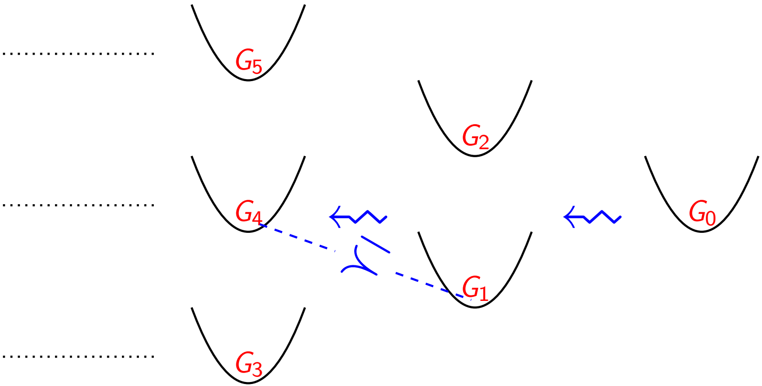
Backward Reachability Analysis



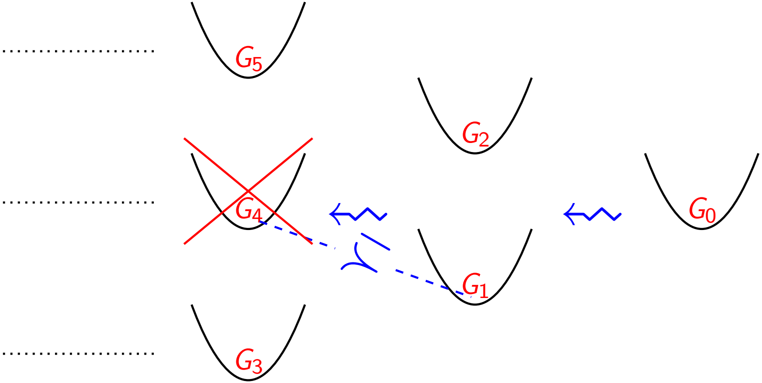
Backward Reachability Analysis



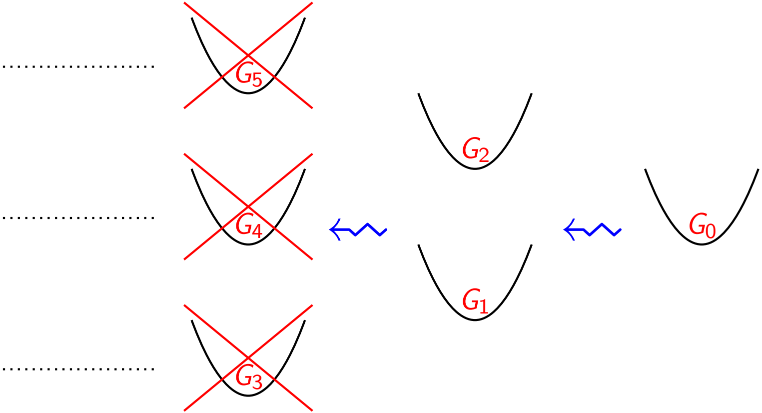
Backward Reachability Analysis



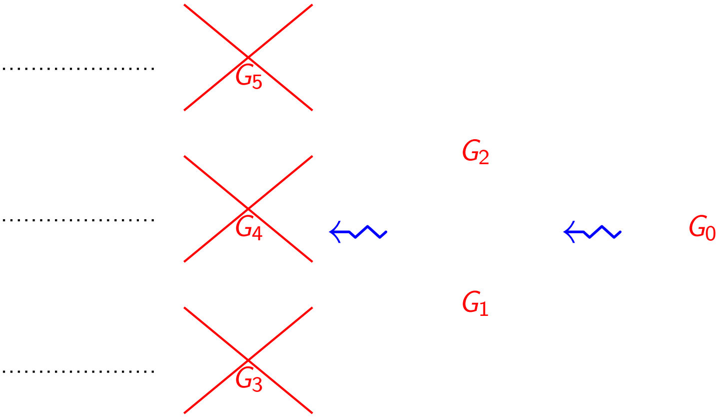
Backward Reachability Analysis



Backward Reachability Analysis



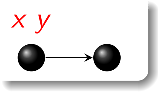
Backward Reachability Analysis



- symbolic representation = graphs
- $\preceq$ WQO implies termination

Computing predecessors

Testing Equality: $x = y$?

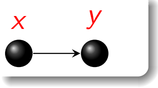


Computing predecessors

Testing Equality: $x = y$?

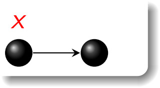
Computing predecessors

Testing Equality: $x = y$?



Computing predecessors

Testing Equality: $x = y$?

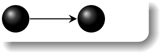


Computing predecessors

Testing Equality: $x = y$?

Computing predecessors

Testing Equality: $x = y$?

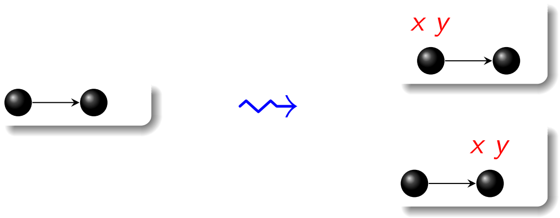


Computing predecessors

Testing Equality: $x = y$?

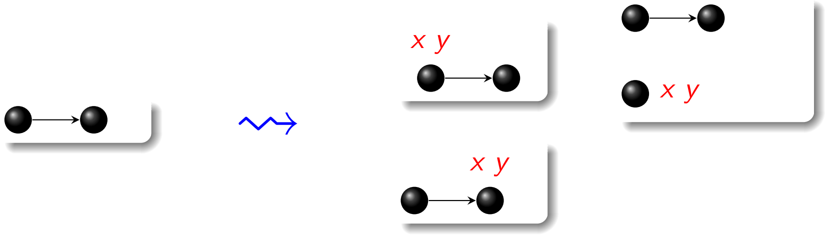
Computing predecessors

Testing Equality: $x = y$?



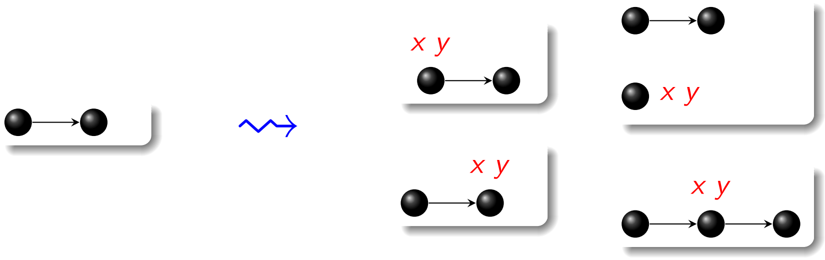
Computing predecessors

Testing Equality: $x = y$?



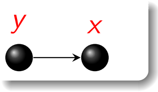
Computing predecessors

Testing Equality: $x = y$?



Computing predecessors

$x := y \cdot next$

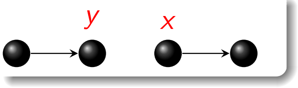


Computing predecessors

$x := y \cdot next$

Computing predecessors

$x := y \cdot next$

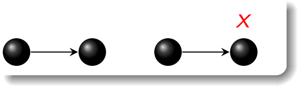


Computing predecessors

$x := y \cdot next$

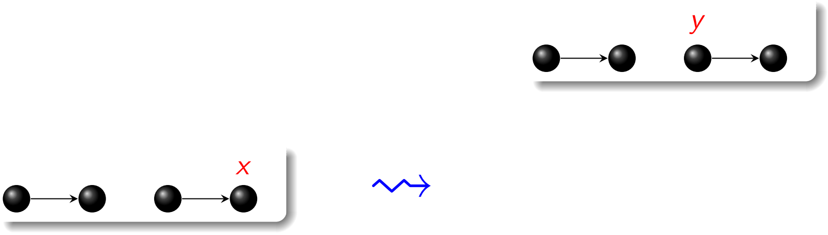
Computing predecessors

$x := y \cdot next$



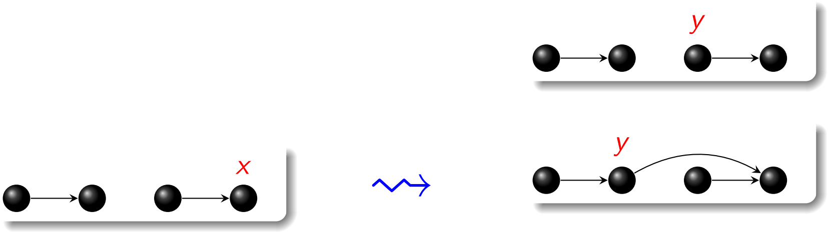
Computing predecessors

$x := y \cdot next$



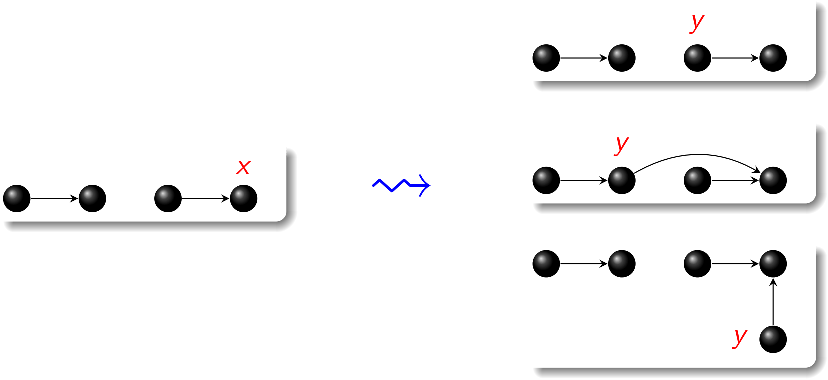
Computing predecessors

$x := y \cdot next$



Computing predecessors

$x := y \cdot next$



WQO

Degree

Degree

$deg(G) :=$ # unlabeled leafs

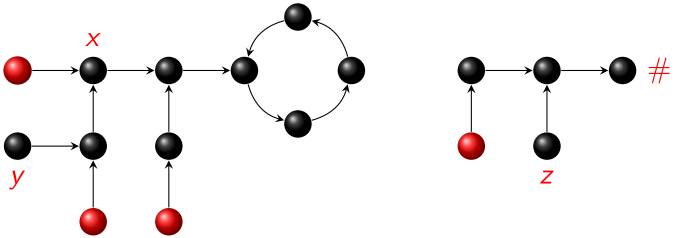
WQO

Degree

Degree

$deg(G) :=$ # unlabeled leafs

Example: $deg(G) = 4$



WQO

Block

Block

maximal subgraph which is connected

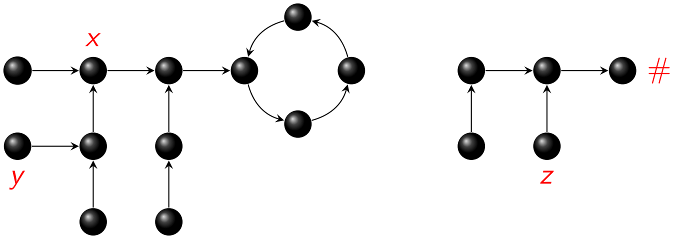
WQO

Block

Block

maximal subgraph which is connected

Example: Two blocks



WQO

Proof

$\preceq$ WQO:

- $g_1 \rightsquigarrow g_2$ implies $deg(g_1) \geq deg(g_2)$
- In back reachability scheme:
    - generated graphs have bounded degree
    - contain finitely many types of blocks (modulo contraction)
    - each graph can be encoded by a vector of multisets of vectors of natural numbers !
    - $\preceq$ WQO by Higman's lemma.

Experiments

Prog.	Prop.	Time	$\#C^{ons.}$	#Iter.	Prog.	Prop.	Time	$\#C^{ons}$
Concat	Deref	0.4 s	7	3	Delete	Deref	0.4 s	8
Fumble	Deref	0.3 s	3	2	Reverse	Deref	0.3 s	2
Walk	Deref	0.4 s	9	3	Zip	Deref	1.9 s	206
Fumble	Garbage	0.7 s	38	14	Reverse	Garbage	0.8 s	55
Reverse	Well-form.	1.7 s	48	20				