Size-Change Termination Analysis in k-Bits

Michael Codish[1], Vitaly Lagoon[2], Peter Schachte[2] Peter J. Stuckey[2]

mcodish@cs.bgu.ac.il, (lagoon|schachte|pjs)@cs.mu.oz.au

[1] Dept. of CS, Ben-Gurion University, Israel

[2] Dept. of CS & SE, University of Melbourne, Australia

At the Bottom Line

- Data-structure for large sets of size change graphs
- Set-based operations: union, composition, and test (for termination condition)

At the Bottom Line

Plan of the Talk

- Introduction to size-change termination
- A Constraint view (sets of solutions)
- From graphs to sets of graphs
- Representation using BDD's

- Data-structure for large sets of size change graphs
- Set-based operations: union, composition, and test (for termination condition)

At the Bottom Line

Plan of the Talk

- Introduction to size-change termination
- A Constraint view (sets of solutions) $\Leftarrow$ the key
- From graphs to sets of graphs
- Representation using BDD's

- Data-structure for large sets of size change graphs
- Set-based operations: union, composition, and test (for termination condition)

At the Bottom Line

Plan of the Talk

- Introduction to size-change termination
- A Constraint view (sets of solutions) $\Leftarrow$ the key
- From graphs to sets of graphs $\Leftarrow$ the contribution
- Representation using BDD's

- Data-structure for large sets of size change graphs
- Set-based operations: union, composition, and test (for termination condition)

At the Bottom Line

Plan of the Talk

- Introduction to size-change termination
- A Constraint view (sets of solutions) $\Leftarrow$ the key
- From graphs to sets of graphs $\Leftarrow$ the contribution
- Representation using BDD's $\Leftarrow$ the prize

- Data-structure for large sets of size change graphs
- Set-based operations: union, composition, and test (for termination condition)

At the Bottom Line

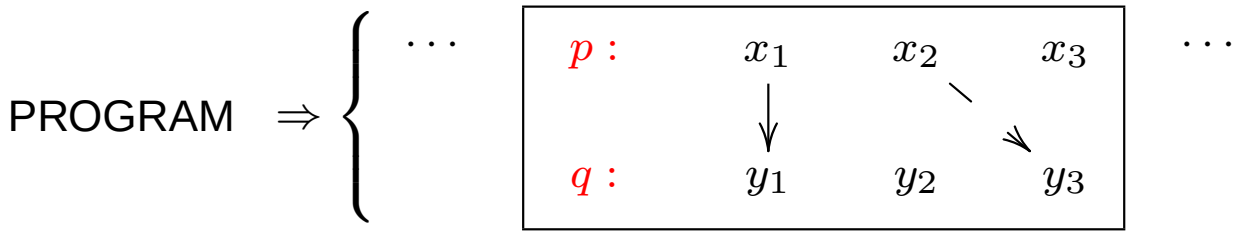
Plan of the Talk

- Introduction to size-change termination
- A Constraint view (sets of solutions) $\Leftarrow$ the key
- From graphs to sets of graphs $\Leftarrow$ the contribution
- Representation using BDD's $\Leftarrow$ the prize

- Data-structure for large sets of size change graphs
- Set-based operations: union, composition, and test (for termination condition)

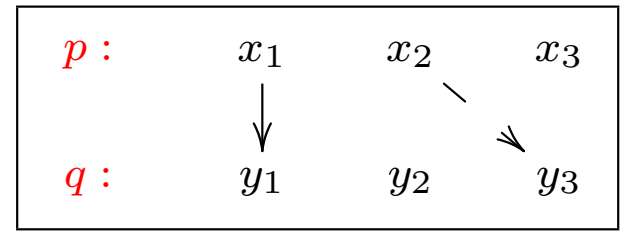
Approximation of Termination

Programs are mapped to sets of size change graphs



Approximation of Termination

Programs are mapped to sets of size change graphs

PROGRAM $\Rightarrow$ { $\cdots$  $\cdots$

Size-Change Graphs

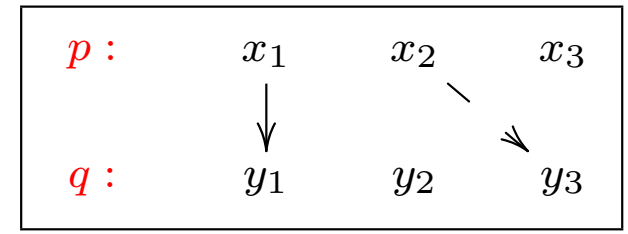
$x_i > y_j, x_i \geq y_j,$

$x_i \in \bar{x}, y_j \in \bar{y}$

(Lee,Jones,Ben-Amram 2001)

Approximation of Termination

Programs are mapped to sets of size change graphs

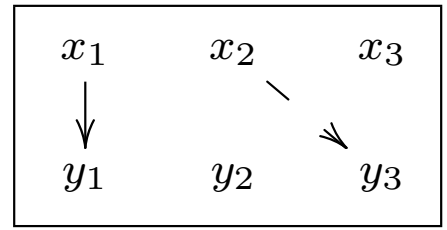
PROGRAM $\Rightarrow$ { $\cdots$  $\cdots$

Size-Change Graphs	Monotonicity Constraints
$x_i > y_j, x_i \geq y_j,$	$v > w, v \geq w,$
$x_i \in \bar{x}, y_j \in \bar{y}$	$v, w \in (\bar{x} \cup \bar{y})$
(Lee,Jones,Ben-Amram 2001)	(Lindenstrauss,Sagiv 1989)

$p(\bar{x}) \leftarrow x_1 > y_1, x_2 \geq y_3,\ q(\bar{y}).$

Approximation of Termination

Programs are mapped to sets of size change graphs

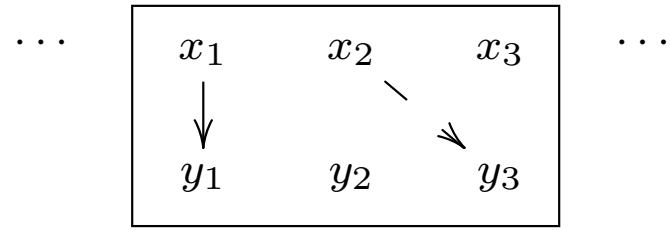
PROGRAM $\Rightarrow$ { $\cdots$  $\cdots$

Size-Change Graphs	Monotonicity Constraints
$x_i > y_j, x_i \geq y_j,$	$v > w, v \geq w,$
$x_i \in \bar{x}, y_j \in \bar{y}$	$v, w \in (\bar{x} \cup \bar{y})$
(Lee,Jones,Ben-Amram 2001)	(Lindenstrauss,Sagiv 1989)

$$p(\bar{x}) \leftarrow x_1 > y_1, x_2 \geq y_3,\ p(\bar{y}).$$

Approximation of Termination

Programs are mapped to sets of size change graphs

PROGRAM $\Rightarrow$ 

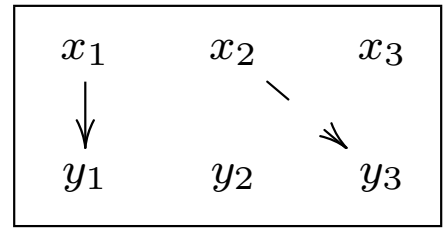
Size-Change Graphs	Monotonicity Constraints
$x_i > y_j, x_i \geq y_j,$	$v > w, v \geq w,$
$x_i \in \bar{x}, y_j \in \bar{y}$	$v, w \in (\bar{x} \cup \bar{y})$
(Lee,Jones,Ben-Amram 2001)	(Lindenstrauss,Sagiv 1989)

$$p(\bar{x}) \leftarrow x_1 > y_1, x_2 \geq y_3,\ p(\bar{y}).$$

- For size change graphs (and monotonicity constraints) termination is decidable.

Approximation of Termination

Programs are mapped to sets of size change graphs

PROGRAM $\Rightarrow$ { $\cdots$  $\cdots$

Size-Change Graphs	Monotonicity Constraints
$x_i > y_j, x_i \geq y_j,$	$v > w, v \geq w,$
$x_i \in \bar{x}, y_j \in \bar{y}$	$v, w \in (\bar{x} \cup \bar{y})$
(Lee,Jones,Ben-Amram 2001)	(Lindenstrauss,Sagiv 1989)

$$p(\bar{x}) \leftarrow x_1 > y_1, x_2 \geq y_3,\ p(\bar{y}).$$

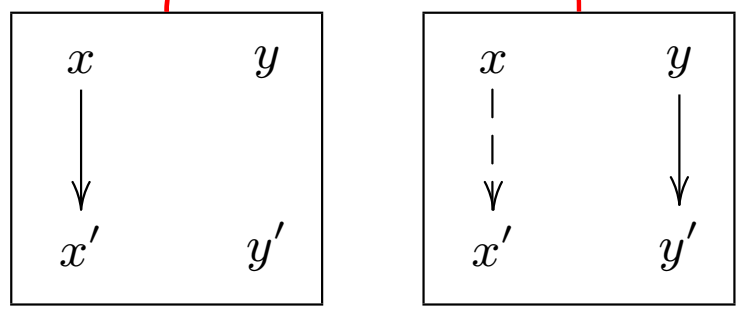
- For size change graphs (and monotonicity constraints) termination is decidable.
- It is not about solving the Halting Problem!

Size-Change Graphs by Example

```
int  Ack(int x, int y) {
        if (x==0) return y + 1
        else if (y==0) return Ack(x − 1, 1)
        else return Ack(x − 1, Ack(x, y − 1))
}
```

Size-Change Graphs by Example

```
int  Ack(int x, int y) {
        if (x==0) return y + 1
        else if (y==0) return Ack(x − 1, 1)
        else return Ack(x − 1, Ack(x, y − 1))
}
```



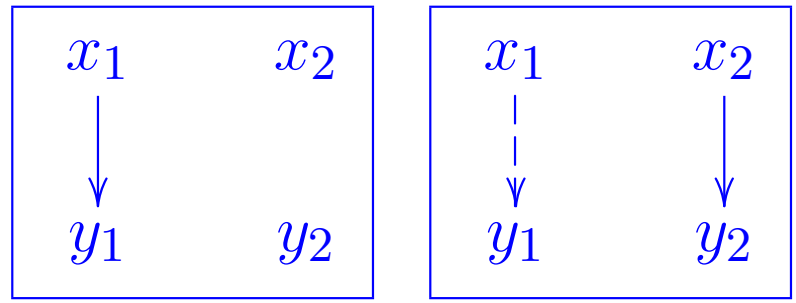
Proving Termination

Classic approach – find a *ranking function* f:

$\exists f\ \forall loop.\ f$ decreases on the $loop$

$Ack(x_1, x_2)$

$f(x, y) = \langle x, y \rangle$



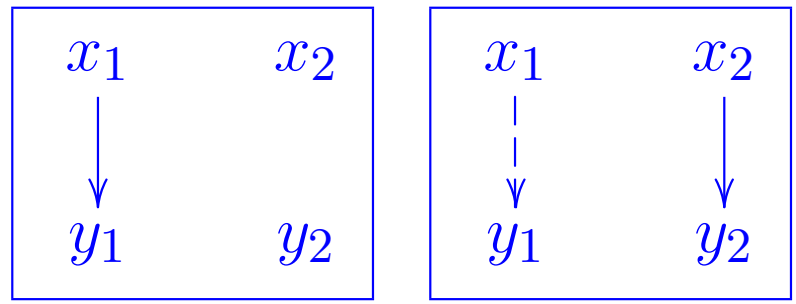
Proving Termination

Classic approach – find a *ranking function* f:

$$\exists f\ \forall loop.\ f \text{ decreases on the } loop$$

$Ack(x_1, x_2)$

$f(x, y) = \langle x, y \rangle$



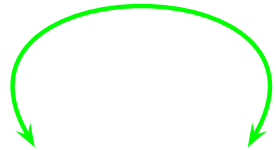
Size change termination – one loop at a time

- for a "price" it is sufficient to prove termination for each individual loop
- there is also a "prize": simpler termination conditions and easier to automate

One Loop at a Time

$\exists f\ \forall loop.\ f$ decreases on the $loop$

One Loop at a Time



$\exists f\ \forall loop.\ f$ decreases on the $loop$

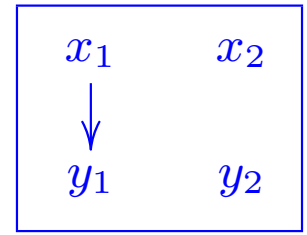
One Loop at a Time

$\forall loop\ \exists f.\ f$ decreases on the $loop$

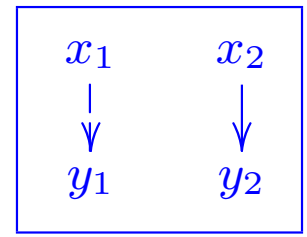
One Loop at a Time

$\forall loop\ \exists f.\ f$ decreases on the $loop$

$ack(x_1, x_2) \leftarrow [x_1 > y_1],\ ack(y_1, y_2)$



$ack(x_1, x_2) \leftarrow [x_1 \geq y_1, x_2 > y_2],\ ack(y_1, y_2)$

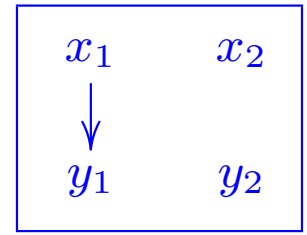


One Loop at a Time

$\forall loop\ \exists f.\ f$ decreases on the $loop$

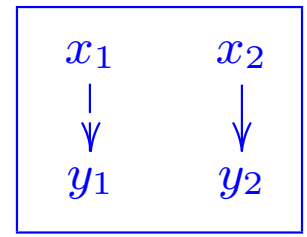
$ack(x_1, x_2) \leftarrow [x_1 > y_1],\ ack(y_1, y_2)$

$f_1(x, y) = x$



$ack(x_1, x_2) \leftarrow [x_1 \geq y_1, x_2 > y_2],\ ack(y_1, y_2)$

$f_2(x, y) = y$



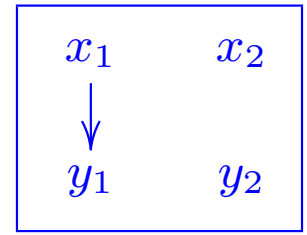
One Loop at a Time

$\forall loop\ \exists f.\ f$ decreases on the $loop$

$ack(x_1, x_2) \leftarrow [x_1 > y_1],\ ack(y_1, y_2)$

$f_1(x, y) = x$

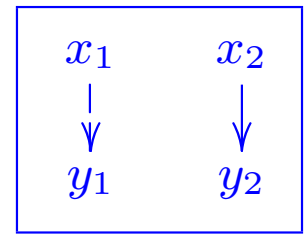
$x_1 > y_1 \models f_1(x_1, x_2) > f_1(y_1, y_2)$



$ack(x_1, x_2) \leftarrow [x_1 \geq y_1, x_2 > y_2],\ ack(y_1, y_2)$

$f_2(x, y) = y$

$x_1 \geq y_1 \wedge x_2 > y_2 \models f_2(x_1, x_2) > f_2(y_1, y_2)$



One Loop at a Time

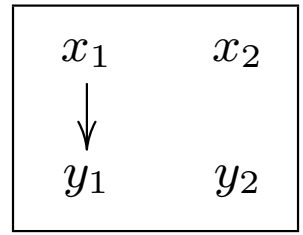
$\forall loop\ \exists f.\ f$ decreases on the $loop$

One Loop at a Time

$\forall loop\ \exists f.\ f$ decreases on the $loop$

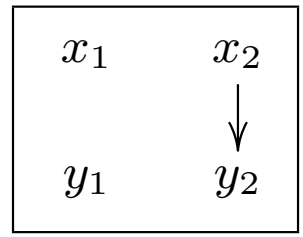
$p(x_1, x_2) \leftarrow [x_1 > y_1],\ p(y_1, y_2) :$

$f_1(x, y) = x$



$p(x_1, x_2) \leftarrow [x_2 > y_2],\ p(y_1, y_2) :$

$f_2(x, y) = y$



One Loop at a Time

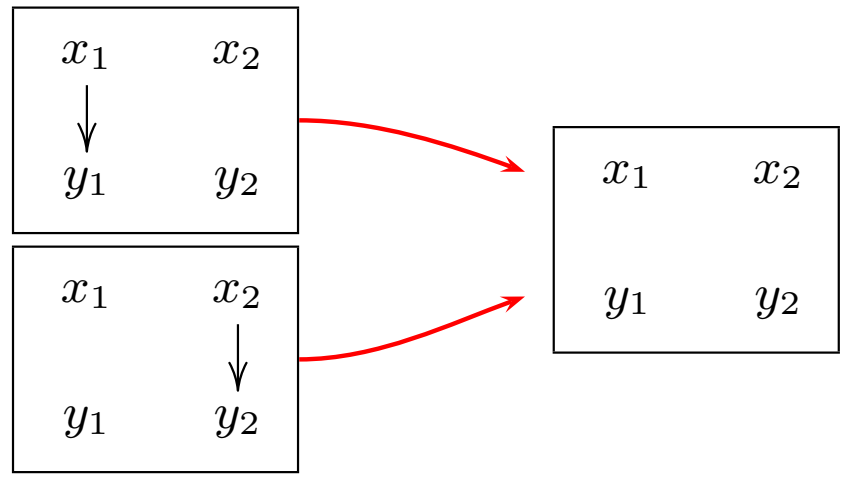
$\forall loop\ \exists f.\ f$ decreases on the $loop$

$p(x_1, x_2) \leftarrow [x_1 > y_1],\ p(y_1, y_2) :$

$f_1(x, y) = x$

$p(x_1, x_2) \leftarrow [x_2 > y_2],\ p(y_1, y_2) :$

$f_2(x, y) = y$



- The composition $\mu_1(\bar{x}, \bar{y}) \circ \mu_2(\bar{x}, \bar{y})$ is a size-change graph entailed by $\exists \bar{z}.\mu_1(\bar{x}, \bar{z}) \wedge \mu_2(\bar{z}, \bar{y})$.
- Loop descriptions are closed under composition:

$$\mu_1 \in G^* \wedge \mu_2 \in G^* \ \Rightarrow\ \mu_1 \circ \mu_2 \in G^*$$

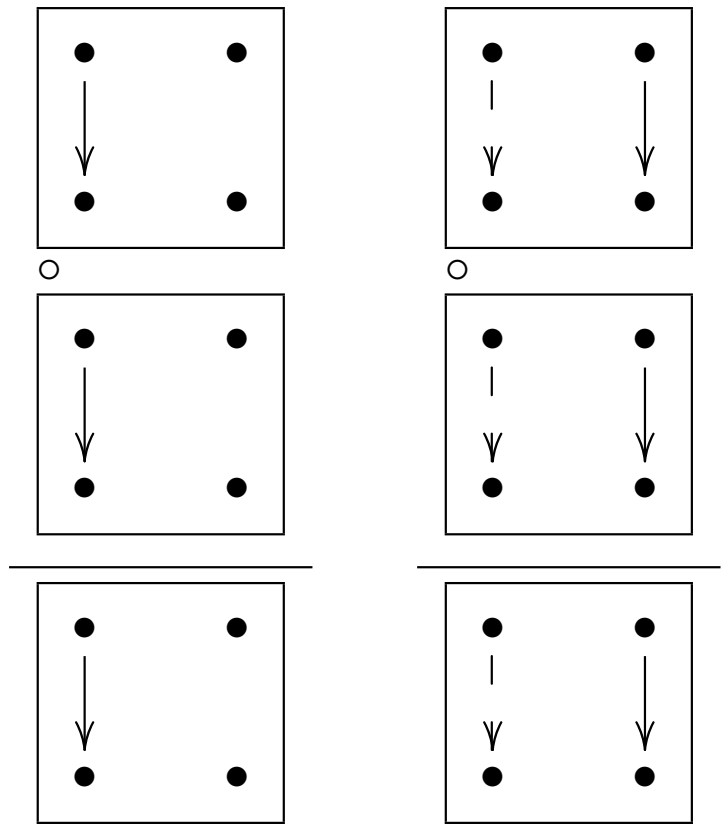
Idempotence

A size-change graph μ is idempotent if $\mu \circ \mu = \mu$.

Idempotence

A size-change graph μ is idempotent if $\mu \circ \mu = \mu$.

Example: the two graphs of $Ack(x, y)$ are idempotent

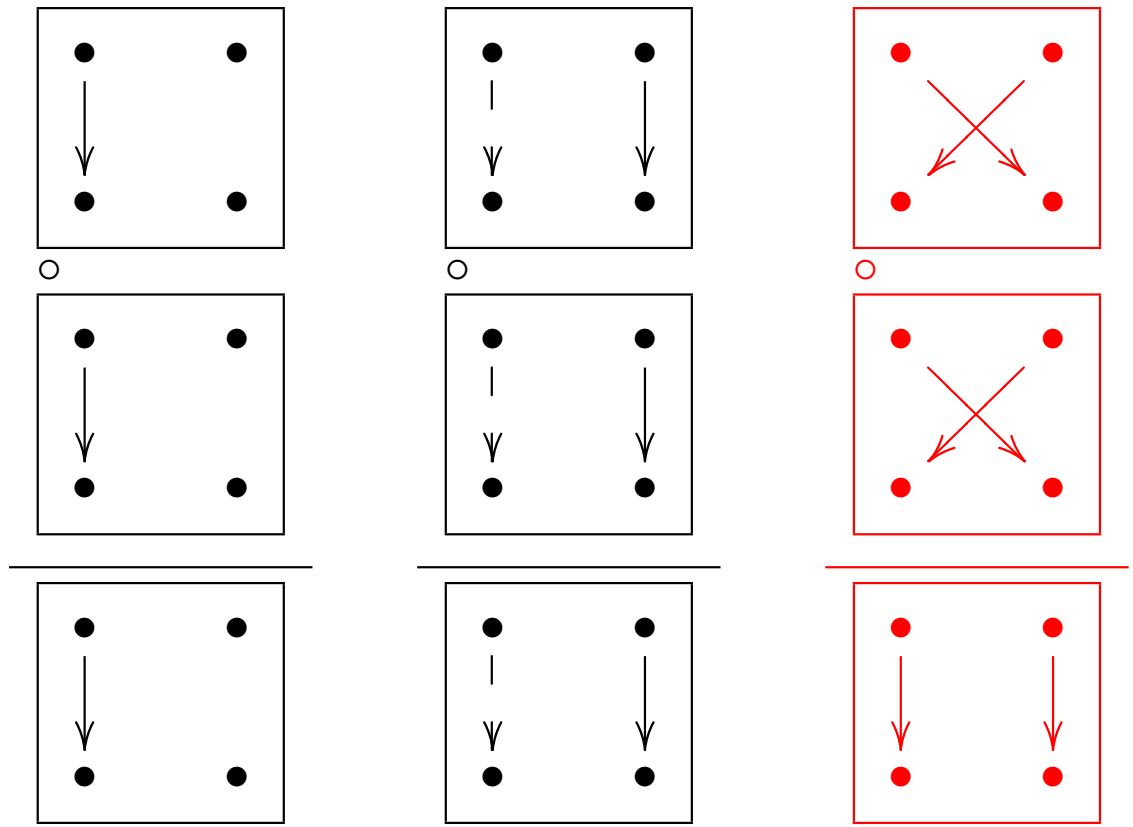


Idempotence

A size-change graph μ is idempotent if $\mu \circ \mu = \mu$.

Example: the two graphs of $Ack(x, y)$ are idempotent

Example: a non-idempotent graph



Correctness of Local Approach

Let G be a set of size-change graphs. If every recursive (idempotent) $\mu \in G^*$ has a ranking function then any program described by G terminates. (Dershowitz *et al*, 2001) (Lee,Jones,Ben-Amram 2001)

Correctness of Local Approach

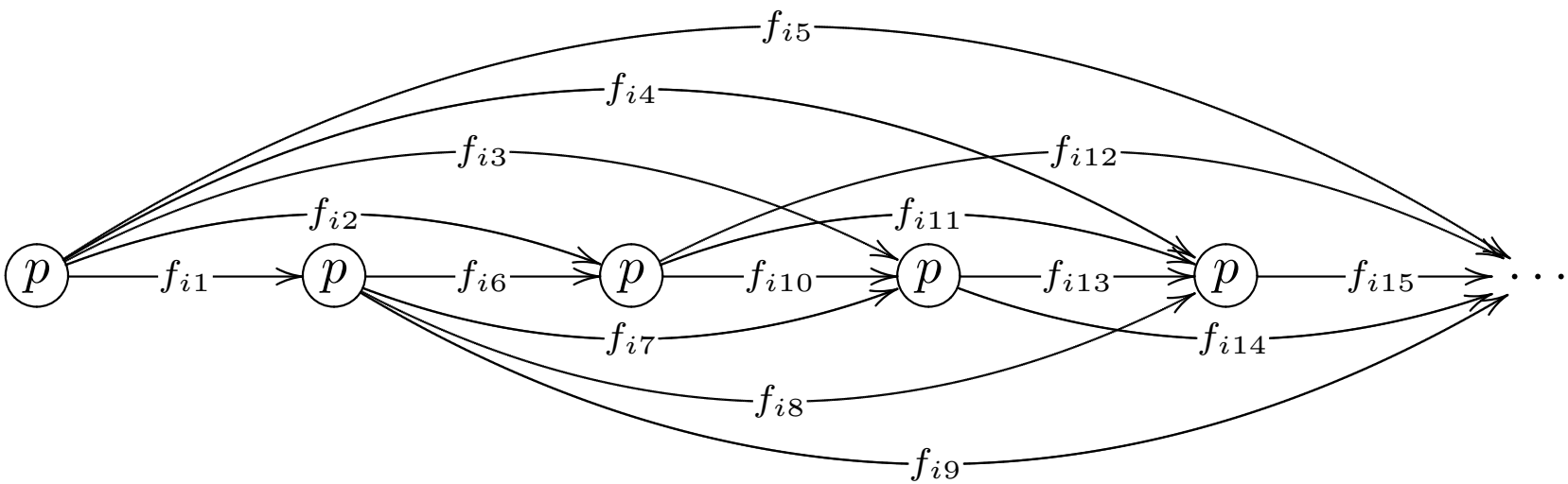
Let G be a set of size-change graphs. If every recursive (idempotent) $\mu \in G^*$ has a ranking function then any program described by G terminates. (Dershowitz *et al*, 2001) (Lee,Jones,Ben-Amram 2001)

Ramsey's theorem (1930): Let X be some countably infinite set and colour the pairs in $X \times X$ in C different colours. Then there exists some infinite $M \subset X$ such that the pairs of M all have the same colour.

Correctness of Local Approach

Let G be a set of size-change graphs. If every recursive (idempotent) $\mu \in G^*$ has a ranking function then any program described by G terminates. (Dershowitz *et al*, 2001) (Lee,Jones,Ben-Amram 2001)

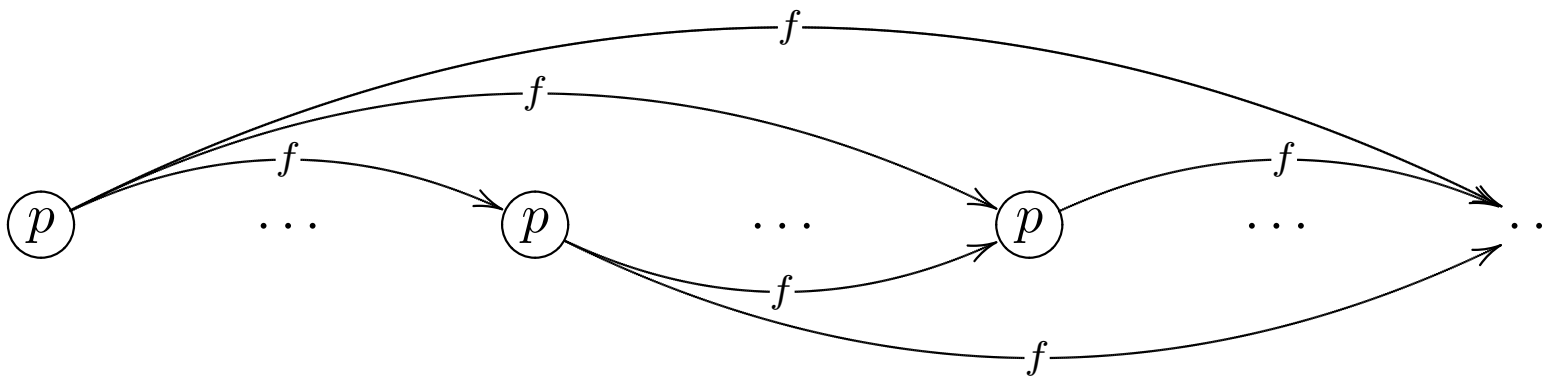
Ramsey's theorem (1930): Let X be some countably infinite set and colour the pairs in $X \times X$ in C different colours. Then there exists some infinite $M \subset X$ such that the pairs of M all have the same colour.



Correctness of Local Approach

Let G be a set of size-change graphs. If every recursive (idempotent) $\mu \in G^*$ has a ranking function then any program described by G terminates. (Dershowitz *et al*, 2001) (Lee,Jones,Ben-Amram 2001)

Ramsey's theorem (1930): Let X be some countably infinite set and colour the pairs in $X \times X$ in C different colours. Then there exists some infinite $M \subset X$ such that the pairs of M all have the same colour.



Completeness

Completness means that we can either find a ranking function or prove that none exists.

If there is ANY ranking function . . .

Completeness

Completness means that we can either find a ranking function or prove that none exists.

If there is ANY ranking function . . .

$$f(\langle x_0, \ldots, x_{n-1} \rangle) = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})}$$

Completeness

Completness means that we can either find a ranking function or prove that none exists.

If there is ANY ranking function . . .

$$f(\langle x_0, \ldots, x_{n-1} \rangle) = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})}$$

$$f(\langle x_0, \ldots x_{n-1} \rangle) = \sum_{j=0}^{n-1} \left| \sum_{k=0}^{n-1} x_k e^{-\frac{2\pi i}{n} jk} \right|$$

Completeness

Completness means that we can either find a ranking function or prove that none exists.

If there is ANY ranking function . . .

$$f(\langle x_0, \ldots, x_{n-1} \rangle) = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})}$$

$$f(\langle x_0, \ldots x_{n-1} \rangle) = \sum_{j=0}^{n-1} \left| \sum_{k=0}^{n-1} x_k e^{-\frac{2\pi i}{n} jk} \right|$$

. . . then we can find it.

Completeness

Completness means that we can either find a ranking function or prove that none exists.

If there is ANY ranking function . . .

$$f(\langle x_0, \ldots, x_{n-1} \rangle) = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})}$$

$$f(\langle x_0, \ldots x_{n-1} \rangle) = \sum_{j=0}^{n-1} \left| \sum_{k=0}^{n-1} x_k e^{-\frac{2\pi i}{n} jk} \right|$$

. . . ~~then we can find it.~~

. . . then we can find one.

Completeness for Size-Change Graphs

For an idempotent SCG if there is any ranking function f then there is one of the form $f(\bar{x}) = x_i$. (Lee *et al*, 2001)

The termination algorithm:

1. Compute the closure G^*
2. Compute the subset of idempotent graphs $I \subseteq G^*$
3. For each $\mu(\bar{x}, \bar{y}) \in I$ check that $\bigvee_i (\mu(\bar{x}, \bar{y}) \rightarrow (x_i > y_i))$

Example: $Ack(x, y)$ is terminating

$$I = G^* = \left\{ \begin{array}{|cc|} \hline x & y \\ \downarrow & \\ x' & y' \\ \hline \end{array} \quad \begin{array}{|cc|} \hline x & y \\ \vdots & \downarrow \\ x' & y' \\ \hline \end{array} \right\}$$

Completeness for Size-Change Graphs

For any SCG if there is a ranking function f then there is one of the form $f(\bar{x}) = \sum_{i \in I} x_i$. (Codish *et al*, ICLP2005)

The termination algorithm:

1. Compute the closure G^*
2. For each $\mu(\bar{x}, \bar{y}) \in G^*$ check that $\mu(\bar{x}, \bar{y}) \models \bigvee_i (x_i > y_i)$

Example:

$$G = \left\{ \begin{array}{|cc|} \hline x_1 & x_2 \\ \searrow & \swarrow \\ y_1 & y_2 \\ \hline \end{array} \right\} \qquad G^* = \left\{ \begin{array}{|cc|} \hline x_1 & x_2 \\ \searrow & \swarrow \\ y_1 & y_2 \\ \hline \end{array} \quad \begin{array}{|cc|} \hline x_1 & x_2 \\ \downarrow & \downarrow \\ y_1 & y_2 \\ \hline \end{array} \right\}$$

$$\mu(\bar{x}, \bar{y}) \models \bigvee_i (x_i > y_i)$$

Completeness for Size-Change Graphs

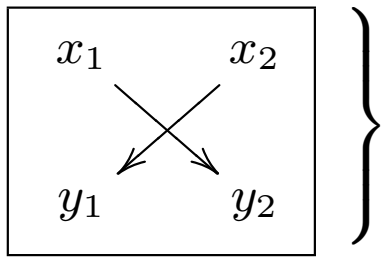
For any SCG if there is a ranking function f then there is one of the form $f(\bar{x}) = \sum_{i \in I} x_i$. (Codish *et al*, ICLP2005)

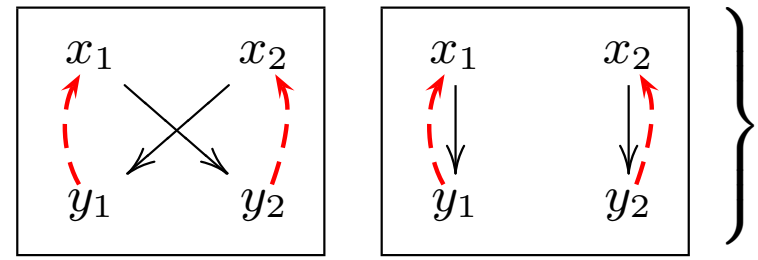
The termination algorithm:

1. Compute the closure G^*
2. For each $\mu(\bar{x}, \bar{y}) \in G^*$ check that $\mu(\bar{x}, \bar{y}) \models \bigvee_i (x_i > y_i)$

Example:

$$G = \left\{ \quad \right\} \qquad G^* = \left\{ \quad \right\}$$





$$\mu(\bar{x}, \bar{y}) \models \bigvee_i (x_i > y_i) \;\equiv\; \neg\Big(\mu(\bar{x}, \bar{y}) \wedge \bigwedge_i (x_i \leq y_i)\Big)$$

Test for Ranking Function

$\bigvee_i \mu(\bar{x}, \bar{y}) \models (x_i > y_i)$ idempotent graphs (Lee *et al*)

$\mu(\bar{x}, \bar{y}) \models \bigvee_i (x_i > y_i)$ all graphs (Codish *et al*)

Test for Ranking Function

$\bigvee_i \mu(\bar{x}, \bar{y}) \models (x_i > y_i)$ idempotent graphs (Lee *et al*)

$\mu(\bar{x}, \bar{y}) \models \bigvee_i (x_i > y_i)$ all graphs (Codish *et al*)

"The graph has down arrow" vrs

"any solution has a down arrow"

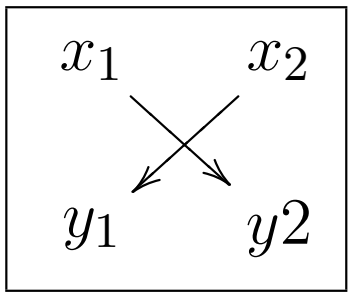
Test for Ranking Function

$\bigvee_i \mu(\bar{x}, \bar{y}) \models (x_i > y_i)$ idempotent graphs (Lee *et al*)

$\mu(\bar{x}, \bar{y}) \models \bigvee_i (x_i > y_i)$ all graphs (Codish *et al*)

"The graph has down arrow" vrs

"any solution has a down arrow"



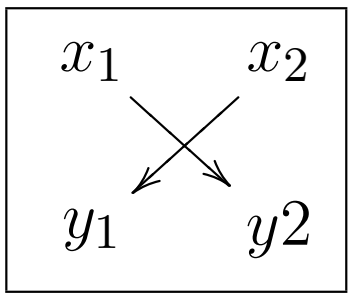
Test for Ranking Function

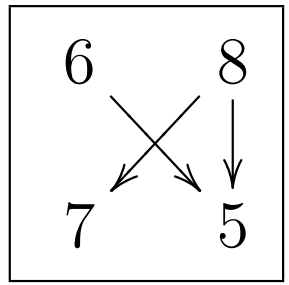
$\bigvee_i \mu(\bar{x}, \bar{y}) \models (x_i > y_i)$ idempotent graphs (Lee *et al*)

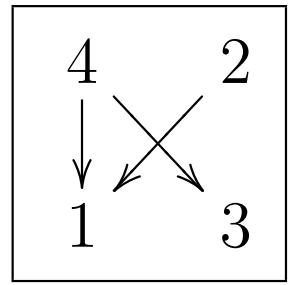
$\mu(\bar{x}, \bar{y}) \models \bigvee_i (x_i > y_i)$ all graphs (Codish *et al*)

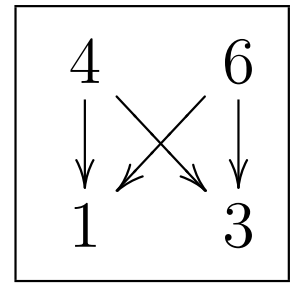
"The graph has down arrow" vrs

"any solution has a down arrow"









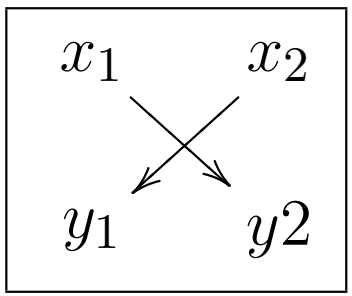
Test for Ranking Function

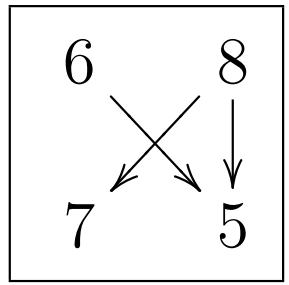
$\bigvee_i \mu(\bar{x}, \bar{y}) \models (x_i > y_i)$ idempotent graphs (Lee *et al*)

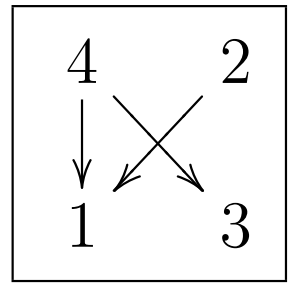
$\mu(\bar{x}, \bar{y}) \models \bigvee_i (x_i > y_i)$ all graphs (Codish *et al*)

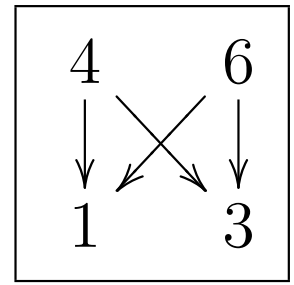
"The graph has down arrow" vrs

"any solution has a down arrow"









- Later it facilitates set-based test for termination.
- We have no set-based operation for idempotent graphs.

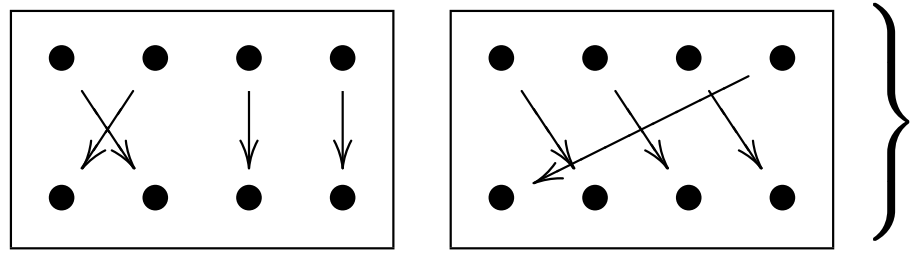
The Downside

- Size change termination is complete for PSPACE (Lee *et al* 2001). Closure under composition may introduce an exponential number of additional size change graphs.

The Downside

- Size change termination is complete for PSPACE (Lee *et al* 2001). Closure under composition may introduce an exponential number of additional size change graphs.

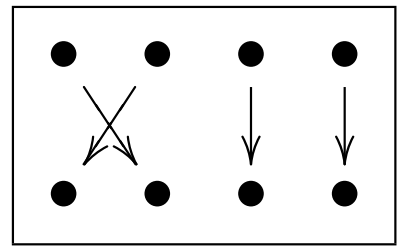
$$G_0 = \left\{ \quad \right\}$$

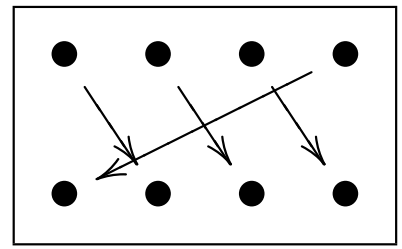


The Downside

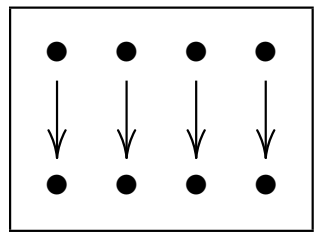
- Size change termination is complete for PSPACE (Lee *et al*2001). Closure under composition may introduce an exponential number of additional size change graphs.

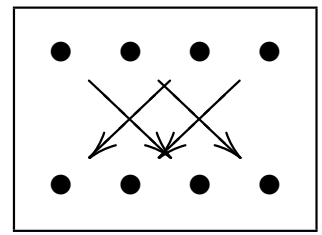
$$G_0 = \left\{ \quad \right\}$$

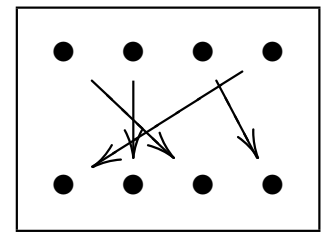


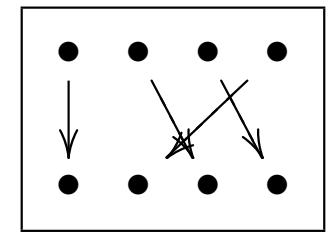


$$G_1 = G_0 \cup \left\{ \quad \right\}$$





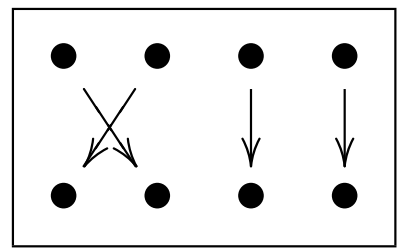


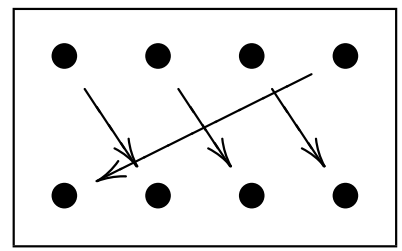


The Downside

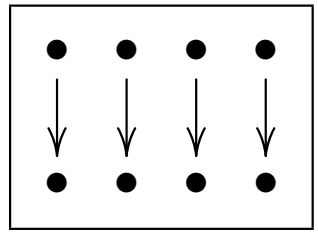
- Size change termination is complete for PSPACE (Lee *et al* 2001). Closure under composition may introduce an exponential number of additional size change graphs.

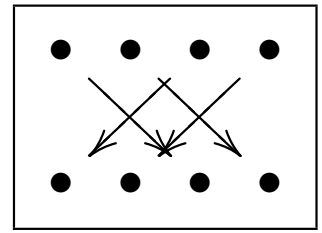
$$G_0 = \left\{ \quad \right\}$$

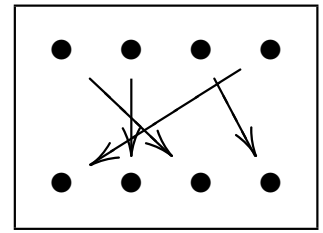


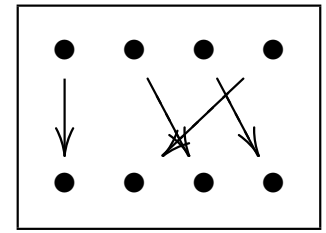


$$G_1 = G_0 \cup \left\{ \quad \right\}$$









$\bullet \quad \bullet \quad \bullet$

At the Bottom Line

- Data-structure for large sets of size change graphs
- Set-based operations: union, composition, and test (for termination condition)

At the Bottom Line

Operations are: $\bigvee$, $\bigcirc$, and test for termination.

- Data-structure for large sets of size change graphs
- Set-based operations: union, composition, and test (for termination condition)

From Graphs to Sets: Union & Test

Union as disjunction —

$$\frac{\left\{\begin{array}{c} p(\bar{x}) \leftarrow \mu_1(\bar{x},\bar{y}),\ p(\bar{y}) \\ \vdots \\ p(\bar{x}) \leftarrow \mu_n(\bar{x},\bar{y}),\ p(\bar{y}) \end{array}\right\}}{p(\bar{x}) \leftarrow \bigvee_k \mu_k(\bar{x},\bar{y}),\ p(\bar{y})}$$

From Graphs to Sets: Union & Test

Union as disjunction —

$$\frac{\left\{\begin{array}{c} p(\bar{x}) \leftarrow \mu_1(\bar{x},\bar{y}),\ p(\bar{y}) \\ \vdots \\ p(\bar{x}) \leftarrow \mu_n(\bar{x},\bar{y}),\ p(\bar{y}) \end{array}\right\}}{p(\bar{x}) \leftarrow \bigvee_k \mu_k(\bar{x},\bar{y}),\ p(\bar{y})}$$

- But we lost the “disjuncts”.

From Graphs to Sets: Union & Test

Union as disjunction —

$$\frac{\left\{\begin{array}{c} p(\bar{x}) \leftarrow \mu_1(\bar{x},\bar{y}),\ p(\bar{y}) \\ \vdots \\ p(\bar{x}) \leftarrow \mu_n(\bar{x},\bar{y}),\ p(\bar{y}) \end{array}\right\}}{p(\bar{x}) \leftarrow \bigvee_k \mu_k(\bar{x},\bar{y}),\ p(\bar{y})}$$

- But we lost the “disjuncts”.

Test as entailment —

$$\boxed{\begin{array}{c} \bigwedge_k \left(\mu_k(\bar{x},\bar{y}) \models \bigvee_i (x_i > y_i)\right) \\ \Updownarrow \\ \left(\bigvee_k \mu_k(\bar{x},\bar{y})\right) \models \bigvee_i (x_i > y_i) \end{array}}$$

From Graphs to Sets: Union & Test

Union as disjunction —

$$\frac{\left\{\begin{array}{c} p(\bar{x}) \leftarrow \mu_1(\bar{x},\bar{y}),\ p(\bar{y}) \\ \vdots \\ p(\bar{x}) \leftarrow \mu_n(\bar{x},\bar{y}),\ p(\bar{y}) \end{array}\right\}}{p(\bar{x}) \leftarrow \bigvee_k \mu_k(\bar{x},\bar{y}),\ p(\bar{y})}$$

- But we lost the “disjuncts”.

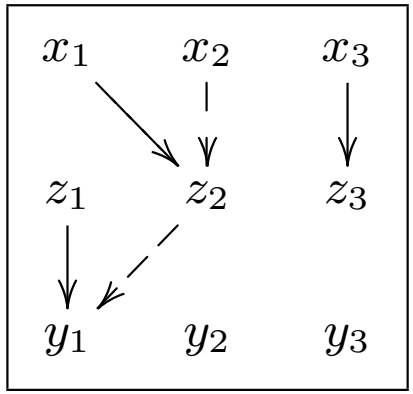
Test as entailment —

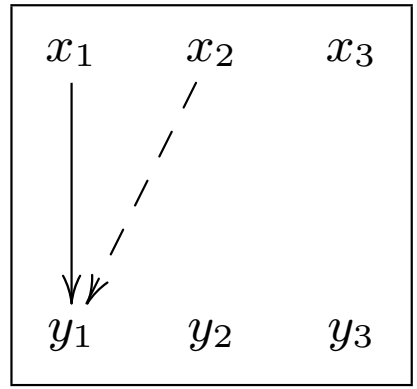
$$\boxed{\begin{array}{c} \bigwedge_k \left(\mu_k(\bar{x},\bar{y}) \models \bigvee_i (x_i > y_i)\right) \\ \Updownarrow \\ \left(\bigvee_k \mu_k(\bar{x},\bar{y})\right) \models \bigvee_i (x_i > y_i) \end{array}}$$

- What about pairwise compositions?

From Graphs to Sets: Composition

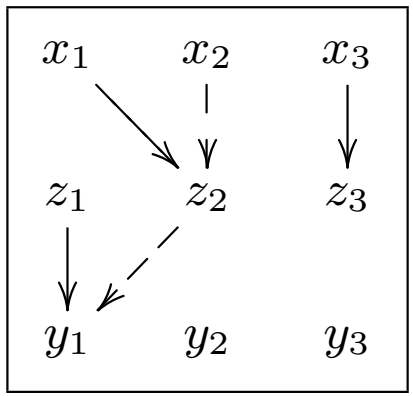
Composition as conjunction?? —

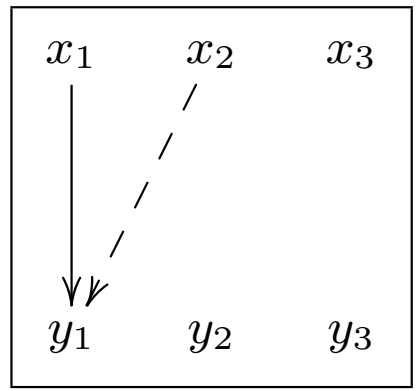




From Graphs to Sets: Composition

Composition as conjunction?? —

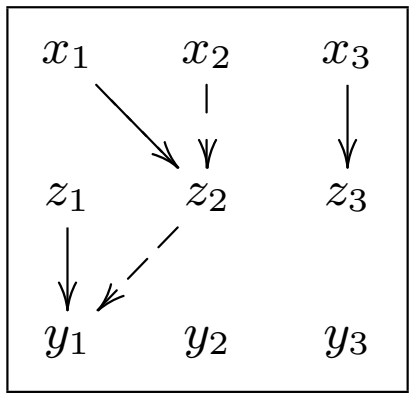


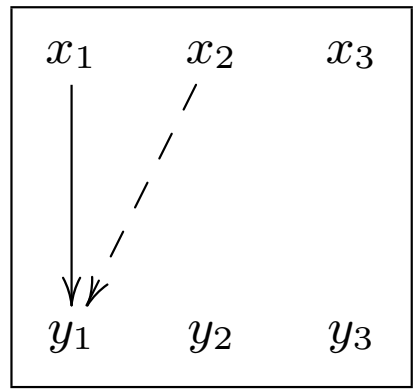


$$\exists \bar{z}. \left(\begin{array}{c} \langle x_1 > z_2, x_2 \geq z_2, x_3 > z_3 \rangle \\ \bigwedge \langle z_1 > y_1, z_2 \geq y_1 \rangle \end{array} \right) = [x_1 > y_1, x_2 \geq y_1]$$

From Graphs to Sets: Composition

Composition as conjunction?? —

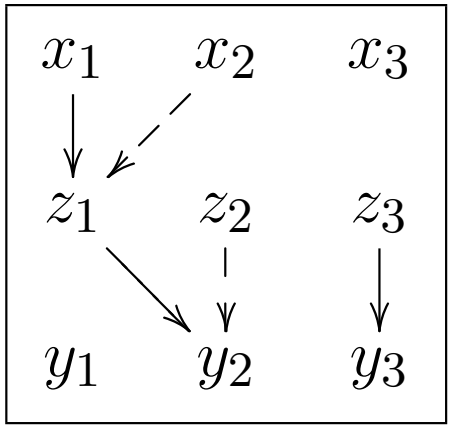


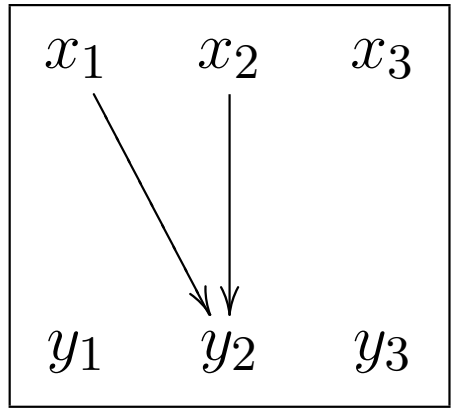


$$\exists \bar{z}. \left(\begin{array}{c} \langle x_1 > z_2, x_2 \geq z_2, x_3 > z_3 \rangle \\ \bigwedge \langle z_1 > y_1, z_2 \geq y_1 \rangle \end{array} \right) = [x_1 > y_1, x_2 \geq y_1]$$

From Graphs to Sets: Composition

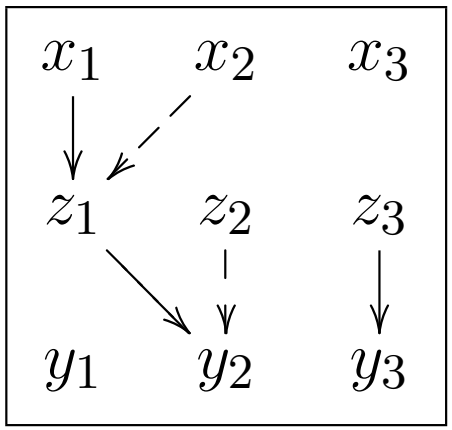
Composition as conjunction?? —

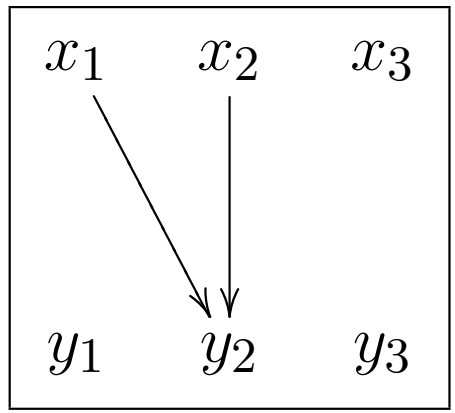




From Graphs to Sets: Composition

Composition as conjunction?? —

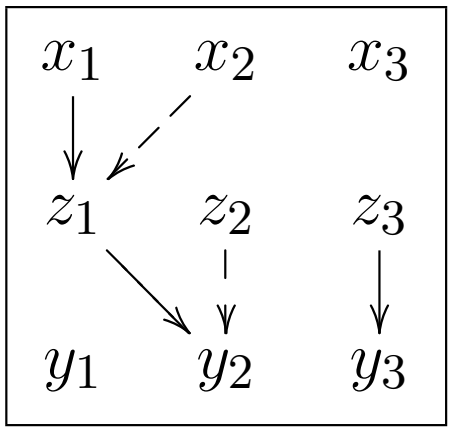


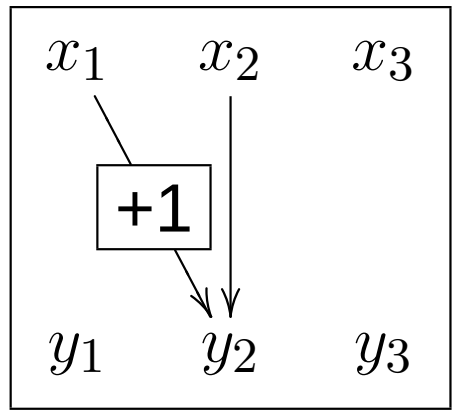


$$\exists \bar{z}. \left(\begin{array}{c} \langle x_1 > z_1, x_2 \geq z_1 \rangle \\ \bigwedge \langle z_1 > y_2, z_2 \geq y_2, z_3 > y_3 \rangle \end{array} \right) \neq [x_1 > y_1, x_2 \geq y_1]$$

From Graphs to Sets: Composition

Composition as conjunction?? —

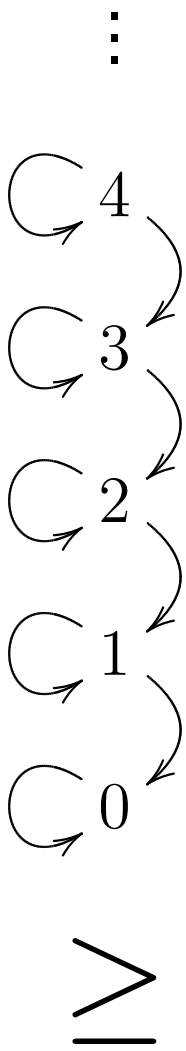




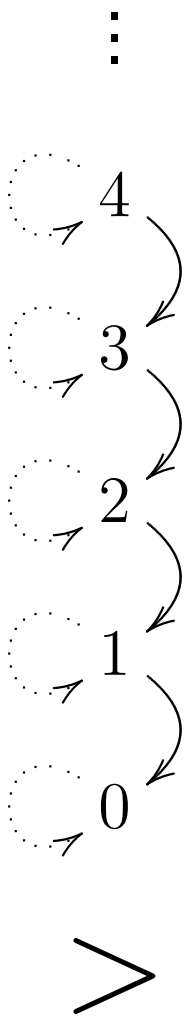
$$\exists \bar{z}. \left(\begin{array}{c} \langle x_1 > z_1, x_2 \geq z_1 \rangle \\ \bigwedge \langle z_1 > y_2, z_2 \geq y_2, z_3 > y_3 \rangle \end{array} \right) \neq [x_1 > y_1, x_2 \geq y_1]$$

Weaker Than Greater-than

$$\exists z.(x \geq z \ \wedge \ z \geq y) \ \equiv \ x \geq y$$



Weaker Than Greater-than

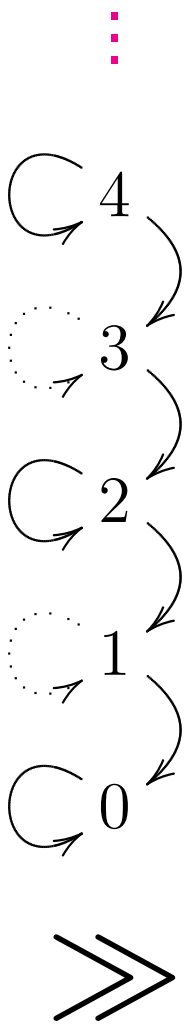


$$\exists z.(x \geq z \ \wedge \ z \geq y) \ \equiv \ x \geq y$$

$$\bigcup$$

$$\exists z.(x > z \ \wedge \ z > y) \ \not\equiv \ x > y$$

Weaker Than Greater-than



$$\exists z.(x \geq z \ \wedge \ z \geq y) \quad \equiv \quad x \geq y$$

$$\bigcup$$

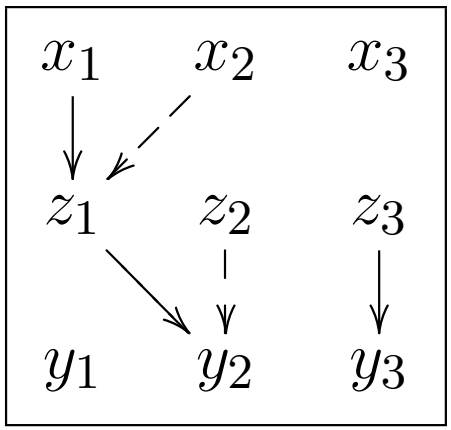
$$\exists z.(x \gg z \ \wedge \ z \gg y) \quad \equiv \quad x \gg y$$

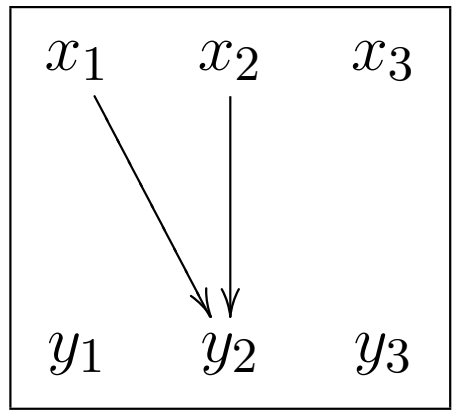
$$\bigcup$$

$$\exists z.(x > z \ \wedge \ z > y) \quad \not\equiv \quad x > y$$

From Graphs to Sets: Composition

Composition as conjunction!! —





$$\exists \bar{z}. \left(\begin{array}{c} \langle x_1 \gg z_1, x_2 \geq z_1 \rangle \\ \bigwedge \langle z_1 \gg y_2, z_2 \geq y_2, z_3 \gg y_3 \rangle \end{array} \right) = [x_1 \gg y_1, x_2 \geq y_1]$$

Set-based Operations

We now have all of the required set-based operations

- Let G be a set of scg's of the form $p(\bar{x}) \leftarrow c_i(\bar{x}, \bar{y}),\ p(\bar{y})$ and $C(\bar{x}, \bar{y}) = \vee_i c_i(\bar{x}, \bar{y})$
- To compute the closure G^* as the fixed point of $G \cup (G \circ G)$
- Compute instead the fixed point of $C(\bar{x}, \bar{y}) \vee \exists \bar{z}.C(\bar{x}, \bar{z}) \wedge C(\bar{z}, \bar{y})$
- Apply test for termination

Towards a Constraint Solver

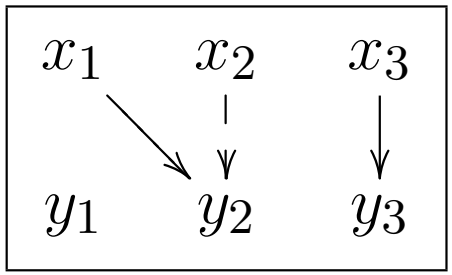
- Using general purpose constraints solvers such as CLP(R): maintain the set of disjuncts

Towards a Constraint Solver

- Using general purpose constraints solvers such as CLP(R): maintain the set of disjuncts
- We model the set of solutions (per disjunction) over a sufficiently large, finite domain (number of nodes)

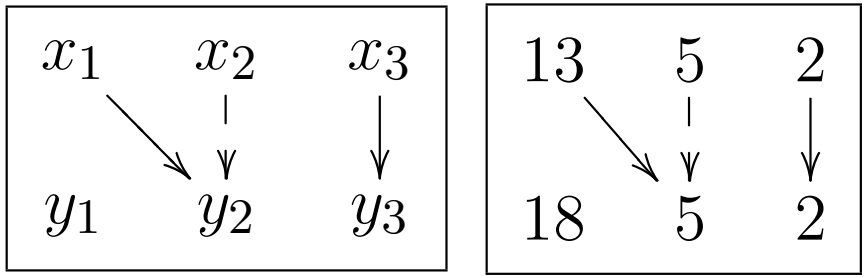
Towards a Constraint Solver

- Using general purpose constraints solvers such as CLP(R): maintain the set of disjuncts
- We model the set of solutions (per disjunction) over a sufficiently large, finite domain (number of nodes)



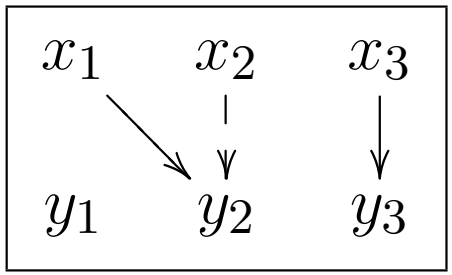
Towards a Constraint Solver

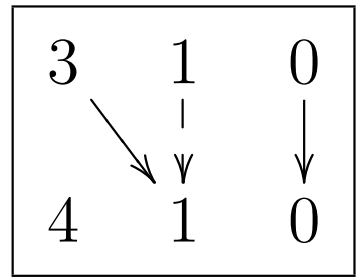
- Using general purpose constraints solvers such as CLP(R): maintain the set of disjuncts
- We model the set of solutions (per disjunction) over a sufficiently large, finite domain (number of nodes)



Towards a Constraint Solver

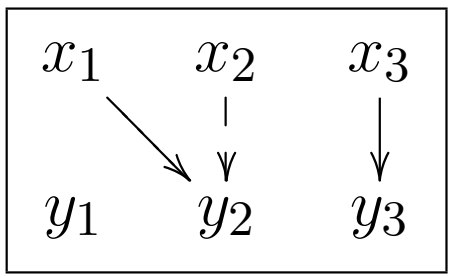
- Using general purpose constraints solvers such as CLP(R): maintain the set of disjuncts
- We model the set of solutions (per disjunction) over a sufficiently large, finite domain (number of nodes)

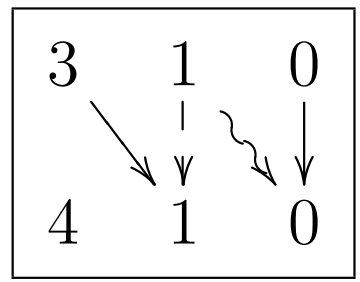




Towards a Constraint Solver

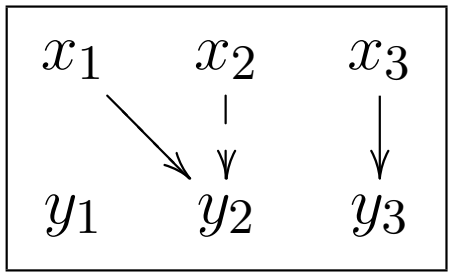
- Using general purpose constraints solvers such as CLP(R): maintain the set of disjuncts
- We model the set of solutions (per disjunction) over a sufficiently large, finite domain (number of nodes)

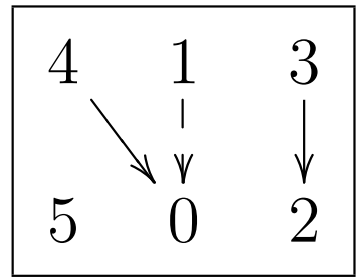




Towards a Constraint Solver

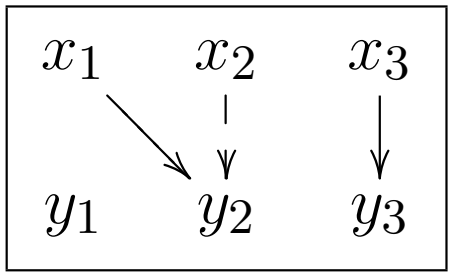
- Using general purpose constraints solvers such as CLP(R): maintain the set of disjuncts
- We model the set of solutions (per disjunction) over a sufficiently large, finite domain (number of nodes)



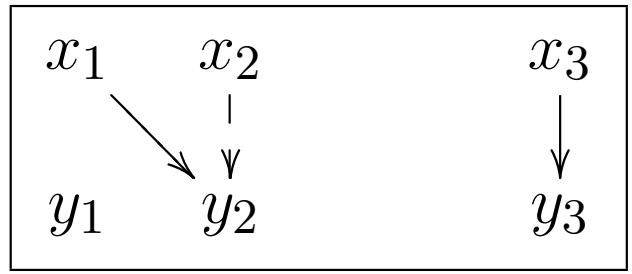


Towards a Constraint Solver

- Using general purpose constraints solvers such as CLP(R): maintain the set of disjuncts
- We model the set of solutions (per disjunction) over a sufficiently large, finite domain (number of nodes)

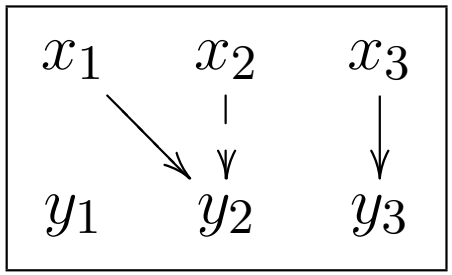


- In practice often less

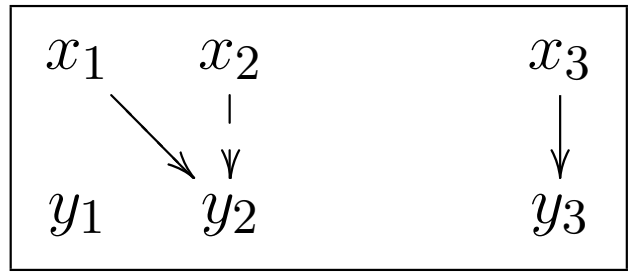


Towards a Constraint Solver

- Using general purpose constraints solvers such as CLP(R): maintain the set of disjuncts
- We model the set of solutions (per disjunction) over a sufficiently large, finite domain (number of nodes)



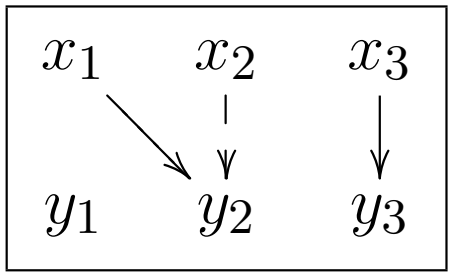
- In practice often less



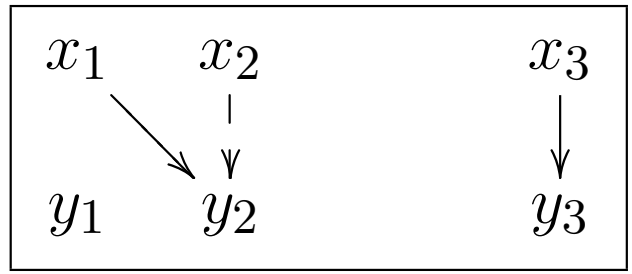
Number of nodes in largest partition

Towards a Constraint Solver

- Using general purpose constraints solvers such as CLP(R): maintain the set of disjuncts
- We model the set of solutions (per disjunction) over a sufficiently large, finite domain (number of nodes)



- In practice often less



Number of nodes in largest partition

- There are subtleties (remember $\gg$) — But it works.

And in k Bits?

- Each variable x and y is represented as $x = x_k x_{k-1} \cdots x_0$ and $y = y_k y_{k-1} \cdots y_0$;
- Each constraint $x \geq y$ and $x \gg y$ is encoded for k bits.
- (example for $k = 2$):
  $x_1 x_0 \geq y_1 y_0 \equiv (x_1 \wedge \neg y_1) \vee ((x_1 \leftrightarrow y_1) \wedge (x_0 \wedge \neg y_0))$
- Similar for $\gg$

From k Bits to BDD's

All components for size change termination analysis are represented as Boolean formula and standard Boolean operations. This facilitates an implementation based on well-studied data structures and well-engineered tools for representing and manipulating Boolean formulae.

At the Bottom Line

- Data-structure for large sets of size change graphs
- Set-based operations: union, composition, and test (for termination condition)
- Prototype is implemented using standard BDD package

At the Bottom Line

Related Work: Size change termination in Polynomial time; Ben-Amram and Lee. They show conditions which enable to avoid performing the expensive closure operation.

- Data-structure for large sets of size change graphs
- Set-based operations: union, composition, and test (for termination condition)
- Prototype is implemented using standard BDD package

At the Bottom Line

Related Work: Size change termination in Polynomial time; Ben-Amram and Lee. They show conditions which enable to avoid performing the expensive closure operation.

Future Work: Apply to richer domains: δ-SCG's and others.

- Data-structure for large sets of size change graphs
- Set-based operations: union, composition, and test (for termination condition)
- Prototype is implemented using standard BDD package