Proving Termination
One Loop at a Time

Michael Codish
Dept. of Computer Science
Ben-Gurion University

Joint work with
Samir Genaim

Termination Analysis

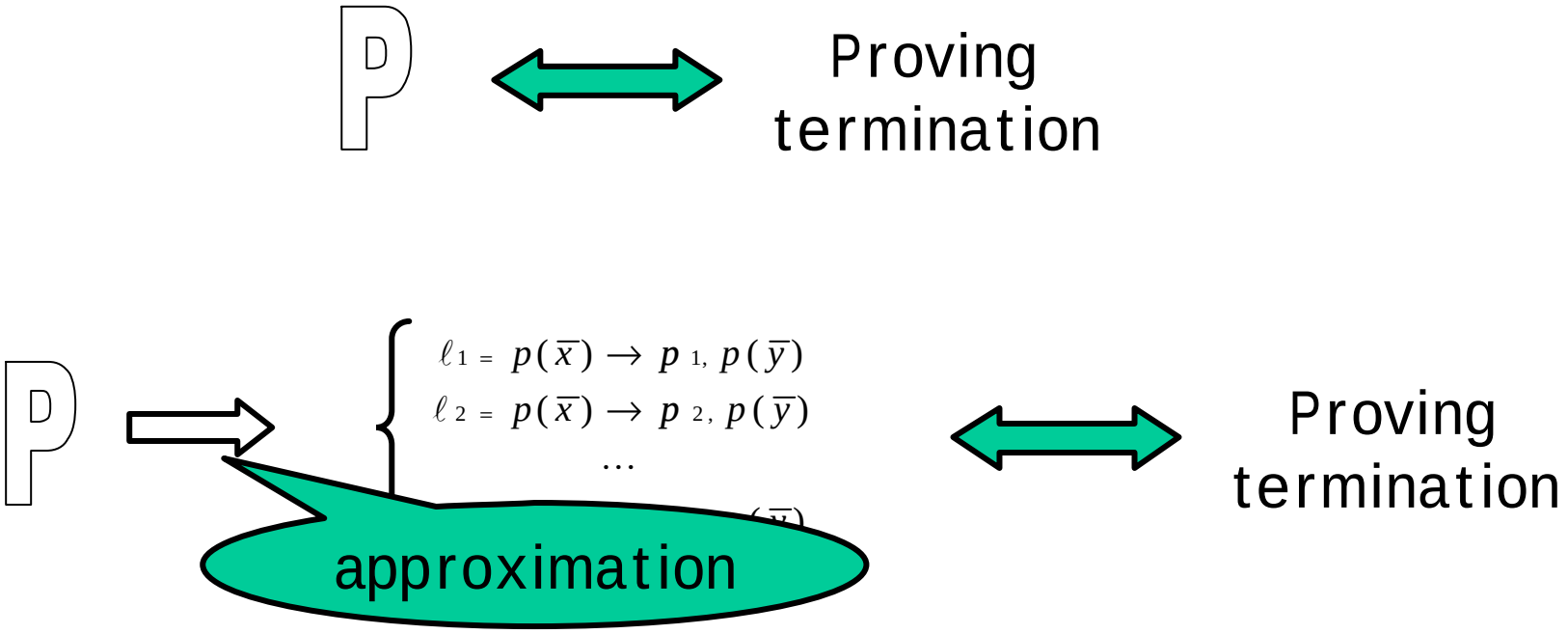
Don't try to solve the halting problem

but rather

Attempt to certify that a program terminates for a class of inputs…

or admit failure.

Program Analysis vs. Verification (or Testing)



A semantic basis for the termination analysis of logic programs; Codish & Taboch.
Sixth International Conference on Algebraic and Logic Programming (1996)
JLP, 1999.

Combining Norms to Prove Termination; Genaim, Codish, Gallagher & Lagoon; 3rd International Workshop on Verification, Model Checking and Abstract Interpretation

Reuse of Results in Termination Analysis of Typed Logic Programs; Bruynooghe, Codish, Genaim &Vanhoof; 9th International Static Analysis Symposium (SAS 2002)

Termination analysis through the combination of type based norms; Bruynooghe, Codish, Gallagher, Genaim & Vanhoof. submitted, 2003.

One Loop at a Time; Codish, Genaim, Bruynooghe, Gallagher and Vanhoof; 6th International Workshop on Termination (WST 2003)

http://www.cs.bgu.ac.il/~mcodish/

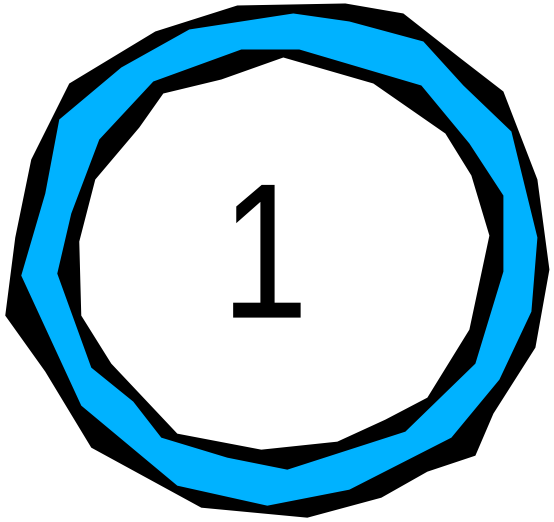
Outline of the Talk

The classic approach

Proving termination one loop at a time

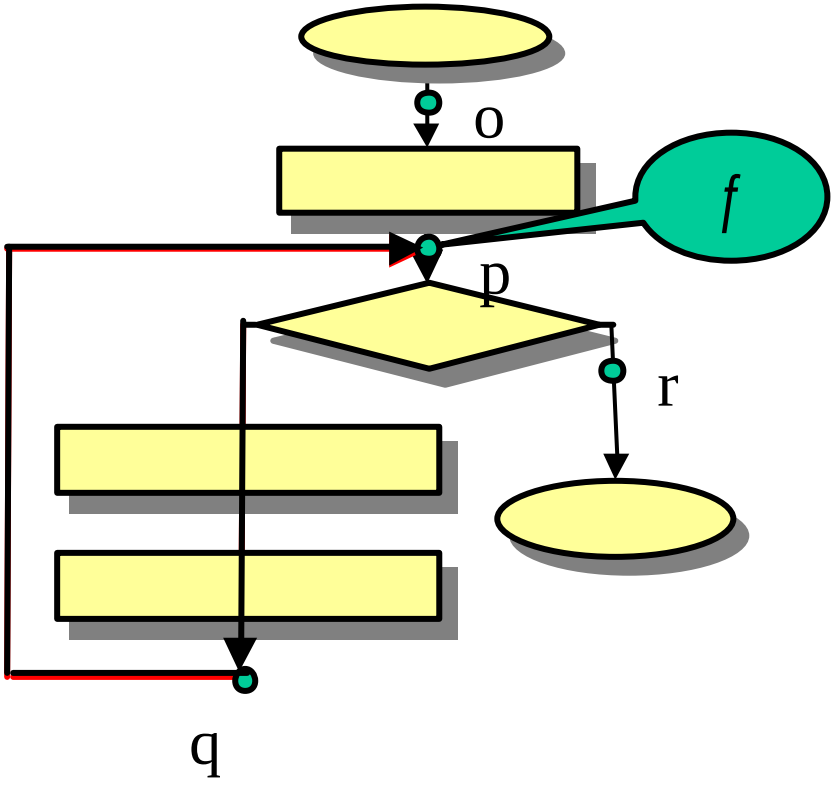
A compari

Preliminary results



The classic approach

The Classic Approach…



program point: o,p,q,r

state (at a point): p(x,y).

convergent: maps to a well-founded domain.
f : state → (D,<)

loop: a simple cycle in the graph «p,q,p»

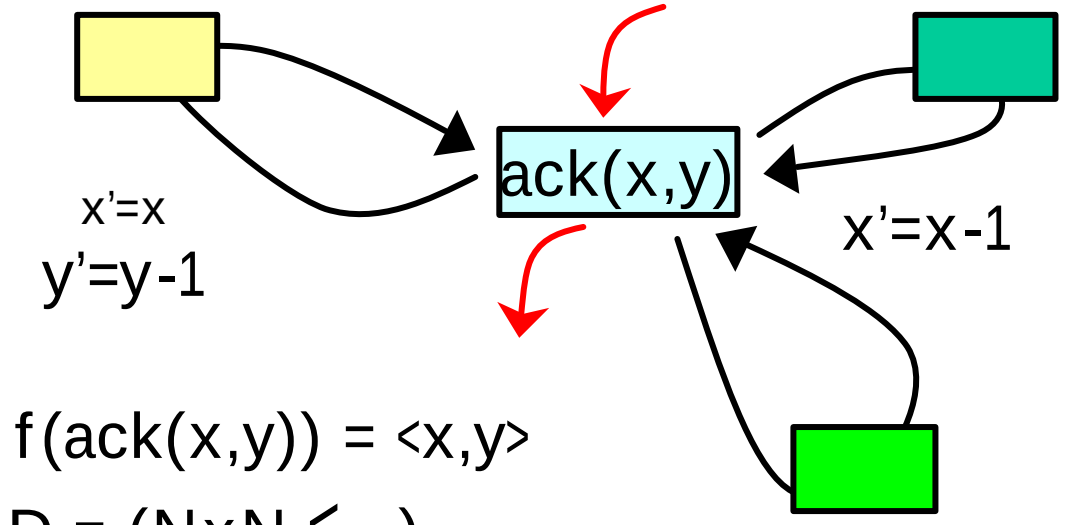
… to Proving Termination

$f." loop . f decreases on the loop

Another Example

```
int ack (int x; int y) {
/* Ackermann's function for x,y >= 0 */
   if (x == 0) return y+1
   else if (y == 0) return ack (x-1,1)
   else return ack (x-1, ack (x,y-1));
}
```

Three loops at the same program point

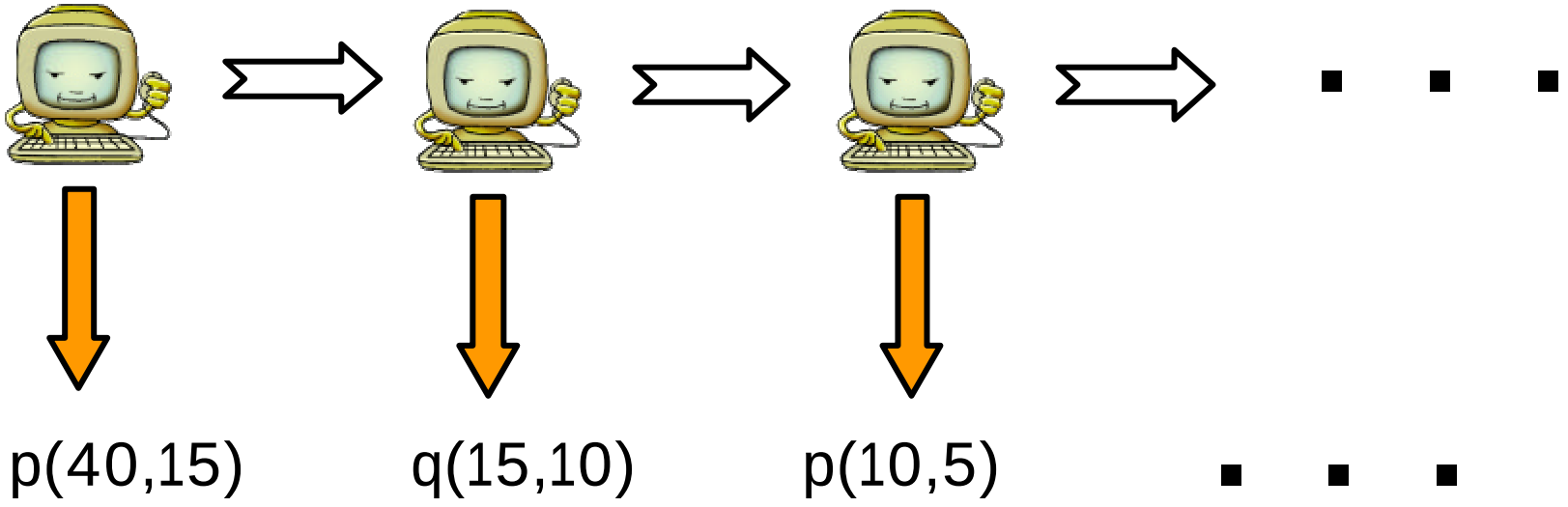


A global termination convergent

f(ack(x,y)) = <x,y>

D = (NxN,$<_{lex}$)

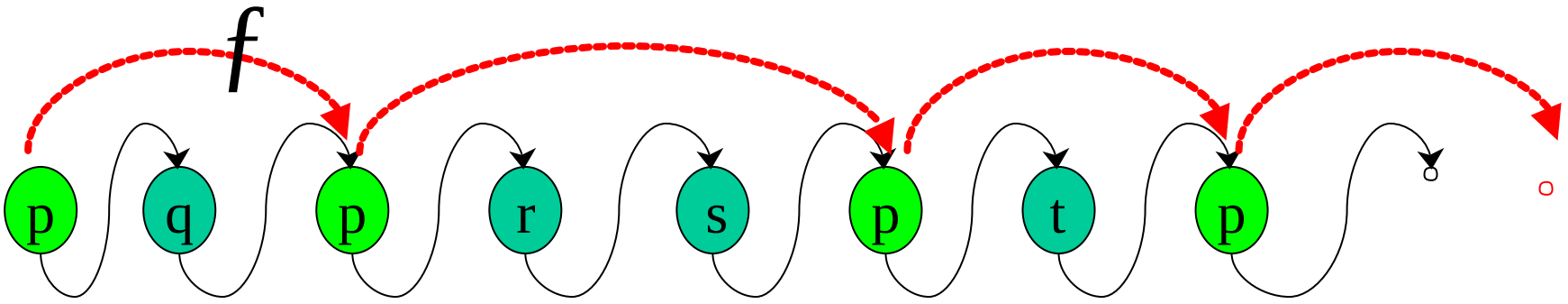
Some notion of a computation?



The sequence of program points with state of variables

Classic Correctness

$$\$f \bullet \quad '' \quad \overline{x}_p \quad \bullet \quad f(\overline{x}) \succ f(\overline{x}')$$

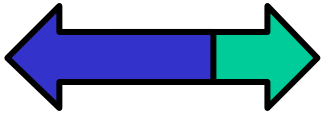


Summary (Part I)

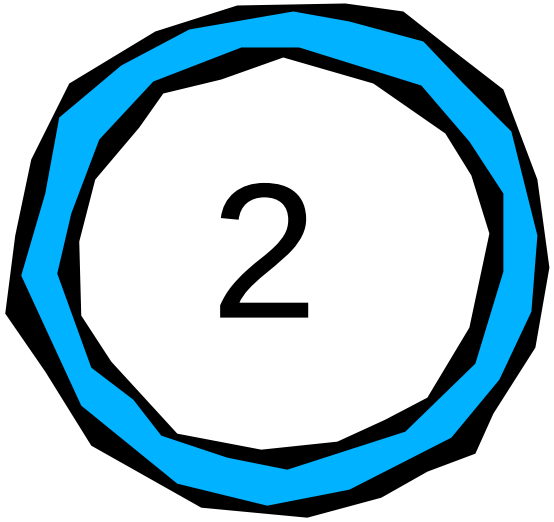
- Loops are defined syntactically
- Convergents are global
- The technique is sound
- The technique is complete

("non-constructive")

$f . " loop . f decreases on the loop

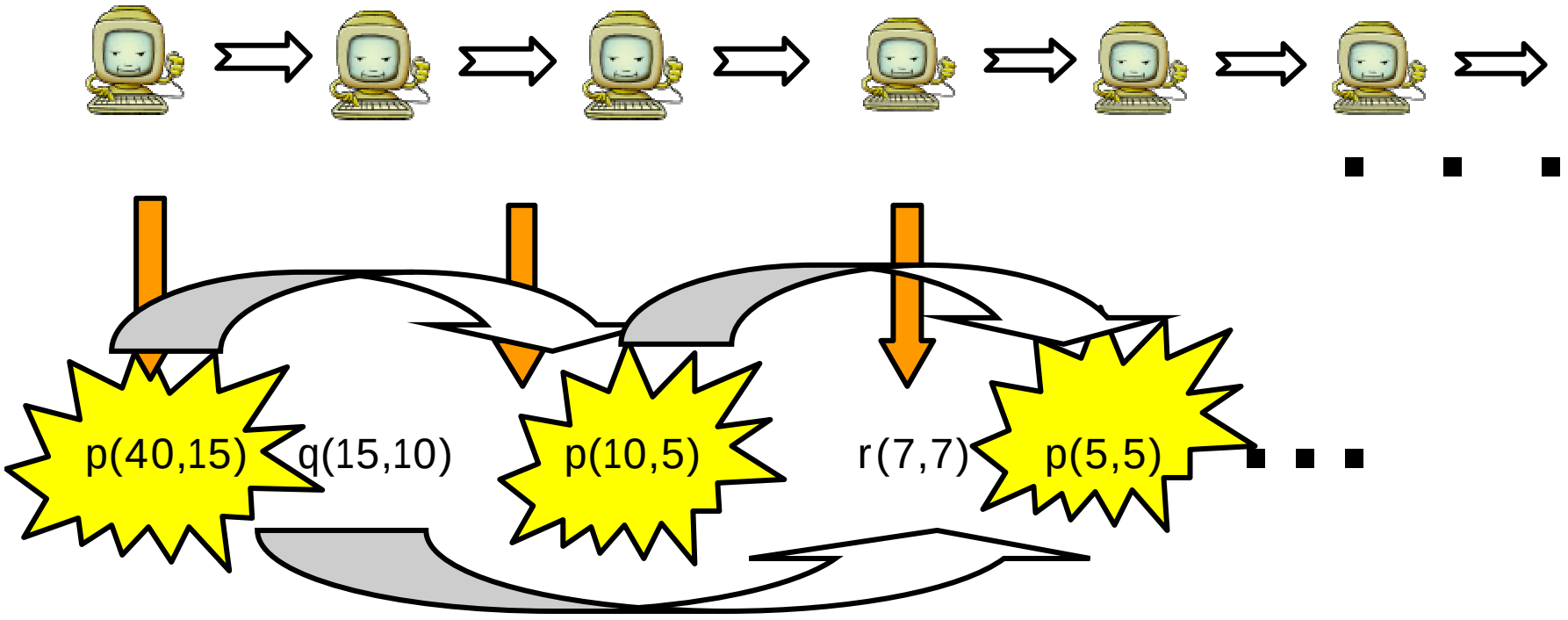


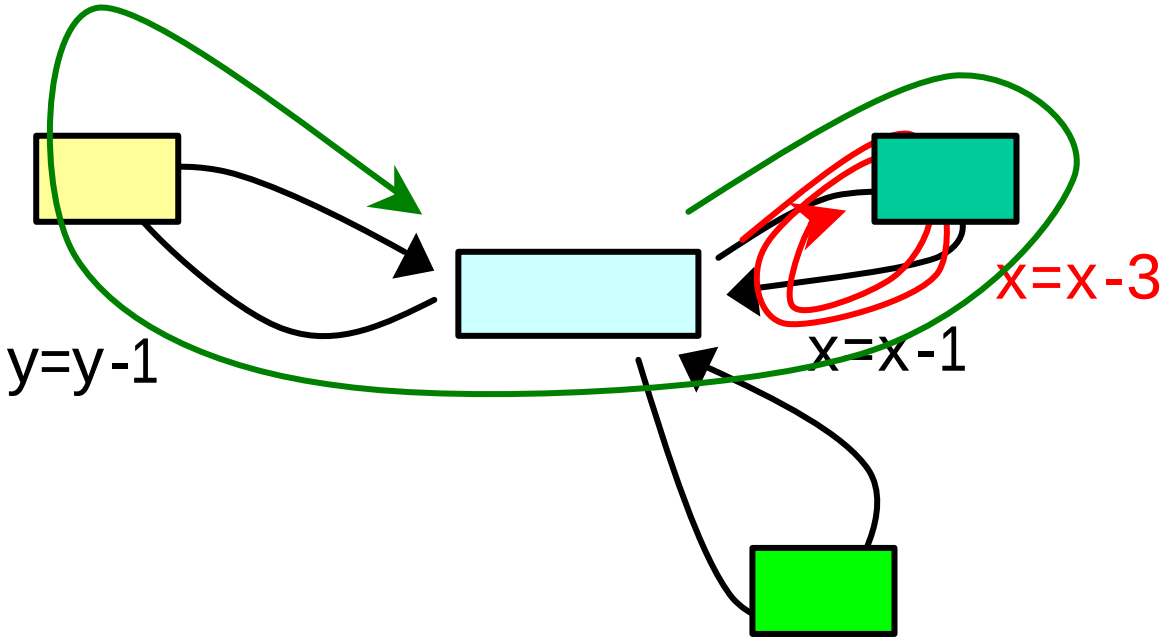
P terminates



One loop at a time

A Semantic Notion of Loop




y=y-1
x=x-1
x=x-3

A Semantic Notion of Loop

A pair of program states which visit at the same program point

Note that if $p(a) \rightarrow p(b)$ &

$p(b) \rightarrow p(c)$

are execution loops then so is

$p(a) \rightarrow p(c)$

There are typically infinitely many execution loops

Example: Execution Loops

```
int ack (int x; int y) {
/* Ackermann's function for x,y >= 0 */
    if (x == 0) return y+1
    else if (y == 0) return ack (x-1,1)
    else return ack (x-1, ack (x,y-1));
}
```

$$\dots$$

$$ack(1,2) \mapsto ack(1,1)$$

$$ack(1,1) \mapsto ack(1,0)$$

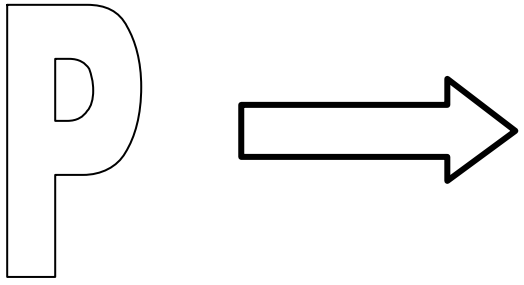
$$ack(1,0) \mapsto ack(0,1)$$

$$ack(1,2) \mapsto ack(0,3)$$

$$ack(1,1) \mapsto ack(0,2)$$

$$\dots$$

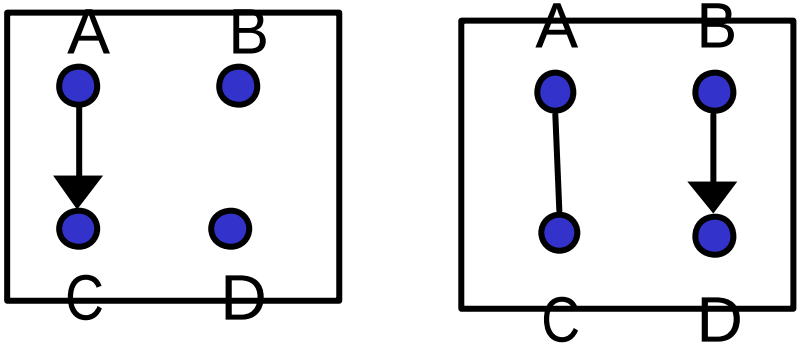
I am going to skip the part where I explain how we apply semantic based program analysis to obtain a finite approximation of a program's execution loops.



$$\begin{cases} \ell_1 = p(\bar{x}) \rightarrow p_1, p(\bar{y}) \\ \ell_2 = p(\bar{x}) \rightarrow p_2, p(\bar{y}) \\ \cdots \\ \ell_k = p(\bar{x}) \rightarrow p_k, p(\bar{y}) \end{cases}$$

ack(A,B) → [C<A], ack(C,D).
ack(A,B) → [A=C,D<B] , ack(C,D).

A,B,C,D >= 0



upper - pre-loop
lower - post-loop
edges - size relations

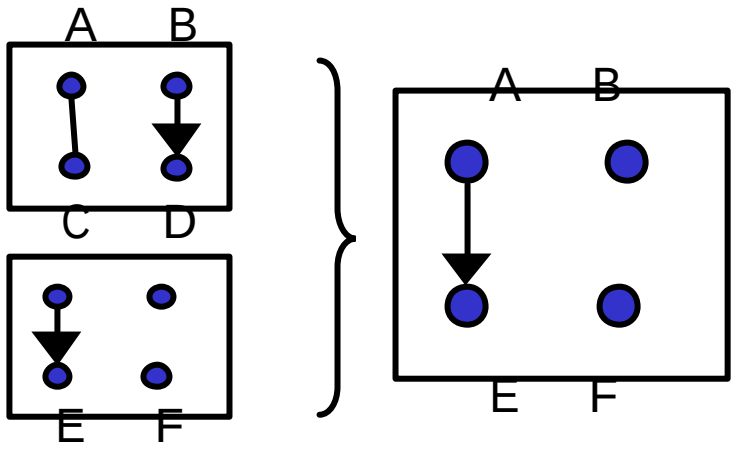
…

$$ack(1,2) \mapsto ack(1,1)$$
$$ack(1,1) \mapsto ack(1,0)$$
$$ack(1,0) \mapsto ack(0,1)$$
$$ack(1,2) \mapsto ack(0,3)$$
$$ack(1,1) \mapsto ack(0,2)$$

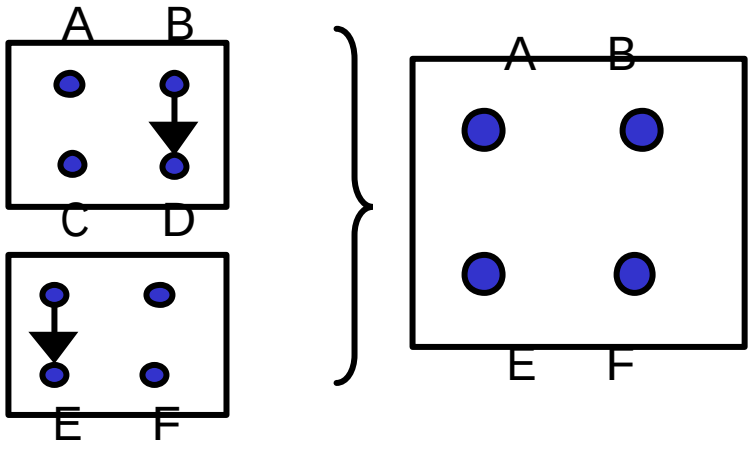
…

Monotonicity constraints (Sagiv et al.)
Size Change Graphs (Jones et al.)

Loop descriptions are closed under composition

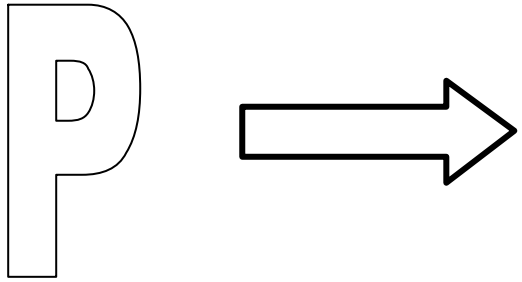


Closing under composition can increase the number of descriptions exponentially



(Lee, Jones, Ben Amram, POPL 2001)

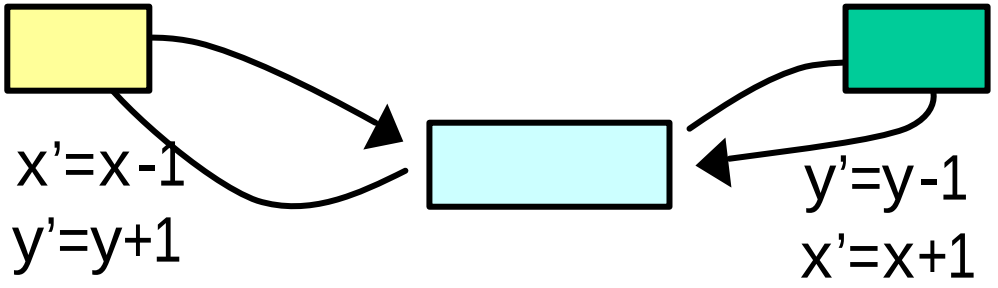
Proving termination one loop at a time



$$\begin{cases} \ell_1 = p(\bar{x}) \rightarrow p_1, p(\bar{y}) \\ \ell_2 = p(\bar{x}) \rightarrow p_2, p(\bar{y}) \\ \cdots \\ \ell_k = p(\bar{x}) \rightarrow p_k, p(\bar{y}) \end{cases}$$

$$\$f \bullet " \bar{x}_p \bullet \; f(\bar{x}) \succ f(\bar{y})$$

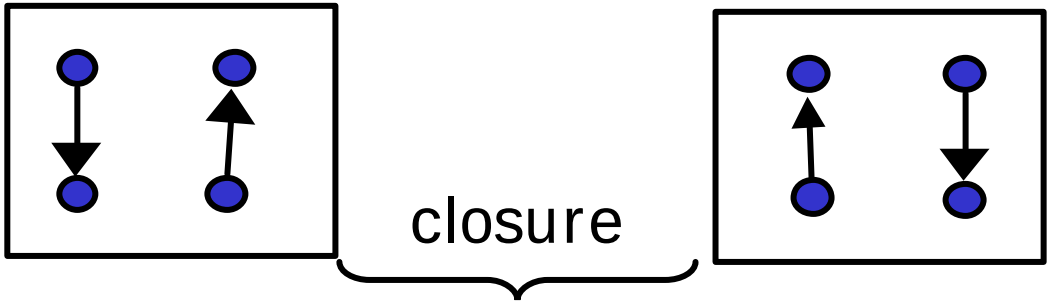
Red Warning Lights ?



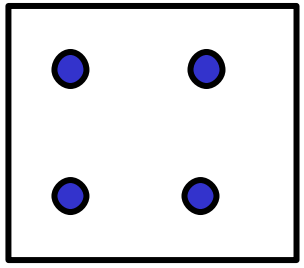
Local convergents:

$f_1(x,y) = x$

$f_2(x,y) = y$

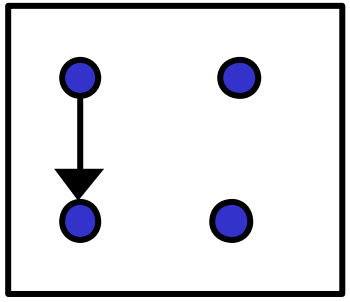


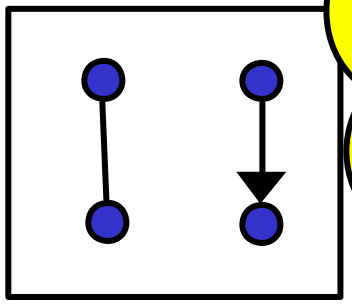
No Local convergent



Ackermann Again

```
int ack (int x; int y) {
/* Ackermann's function for x,y >= 0 */
   if (x == 0) return y+1
   else if (y == 0) return ack (x-1,1)
   else return ack (x-1, ack (x,y
end;
```



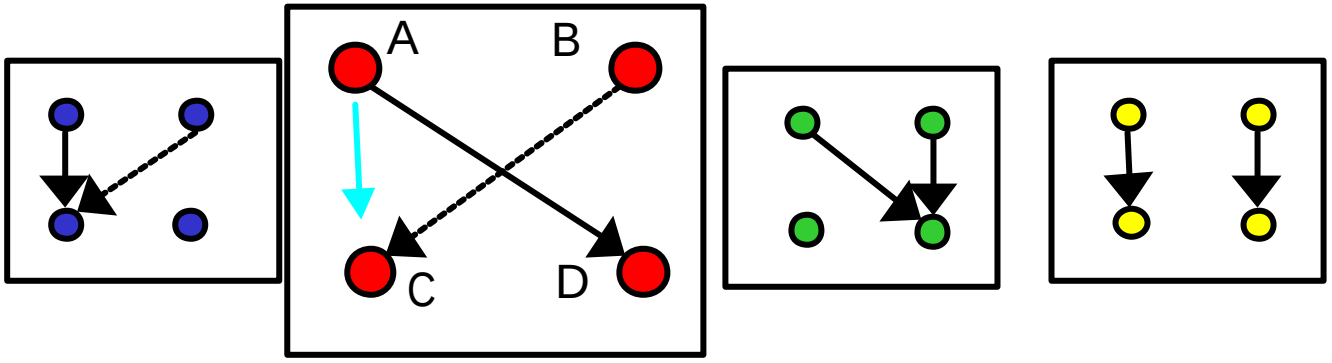


One might think that we could just tack together the local functions to get a global one.

Local convergents:
$f_1(ack(x,y)) = x$
$f_2(ack(x,y)) = y$

Global convergent:
$f(ack(x,y)) = \langle x,y \rangle$

Another Example



$f_1(x,y)=x$ $f_2(x,y)=x+y$ $f_3(x,y)=y$ $f_4(x,y)=x$

closure

A global function?

<min(x,y),?x>

min(A,B) = min(C,D) & A>D & B=C ➞ A>C

Systems based on “One Loop at a Time”

Termilog

Automatic Termination Analysis of Logic Programs (1996)

N. Lindenstrauss and Y. Sagiv
(N. Dershowitz, A. Serbrenik)

TerminWeb

A Semantic Basis for Termination Analysis of Logic Programs (1997)

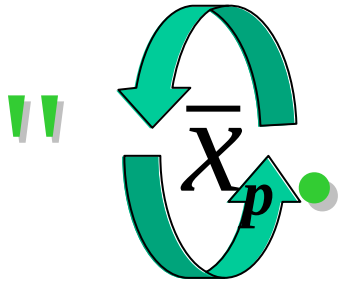
M. Codish and C. Taboch
(S. Genaim)

The Size-Change Principle for Program Termination (2001)

C. Lee, N. Jones and A. Ben-Amram

Correctness

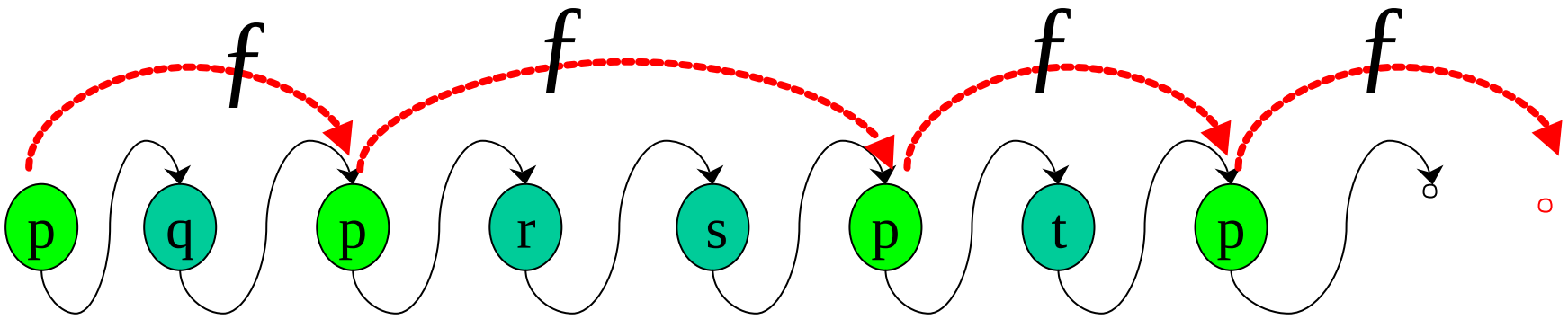
Does the condition



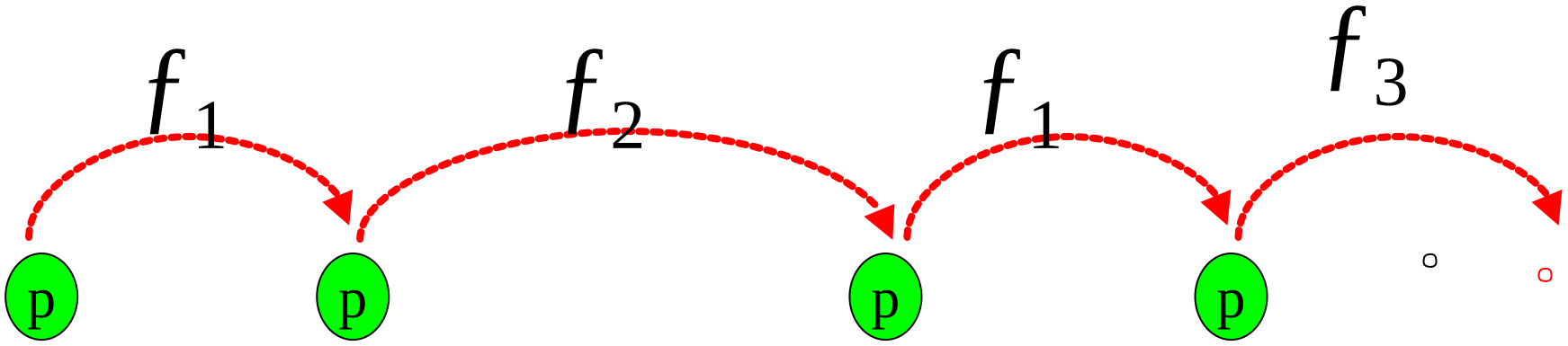
$$\forall \bar{x}_p \bullet \exists f \bullet f(\bar{x}) \succ f(\bar{x}')$$

guarantee termination ?

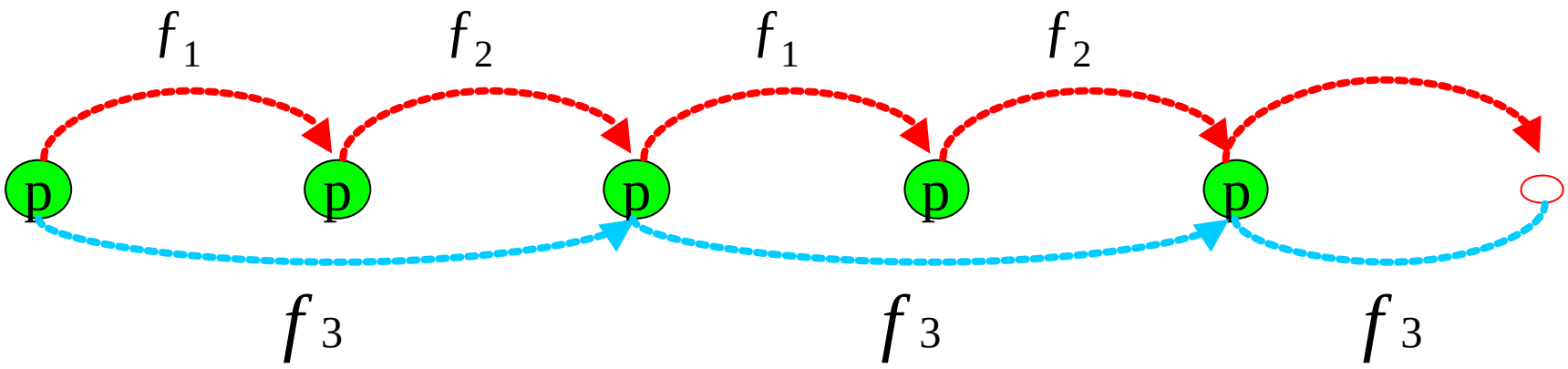
We had:



Now we have:

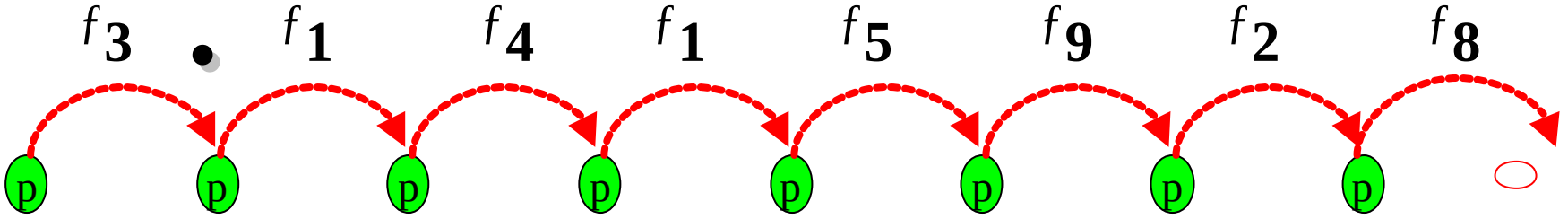


It could be simple:

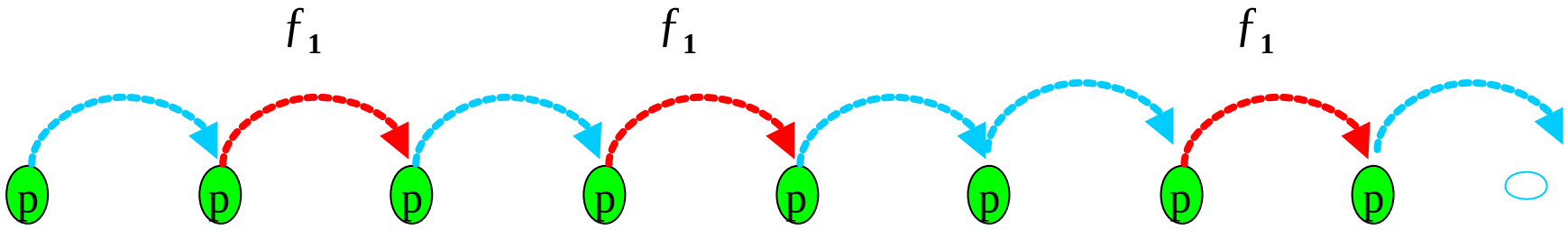


but then again … :

$\pi = 3.1415928\ldots$



It is not sufficient to know that f_1 reduces the size of the state infinitely often,…

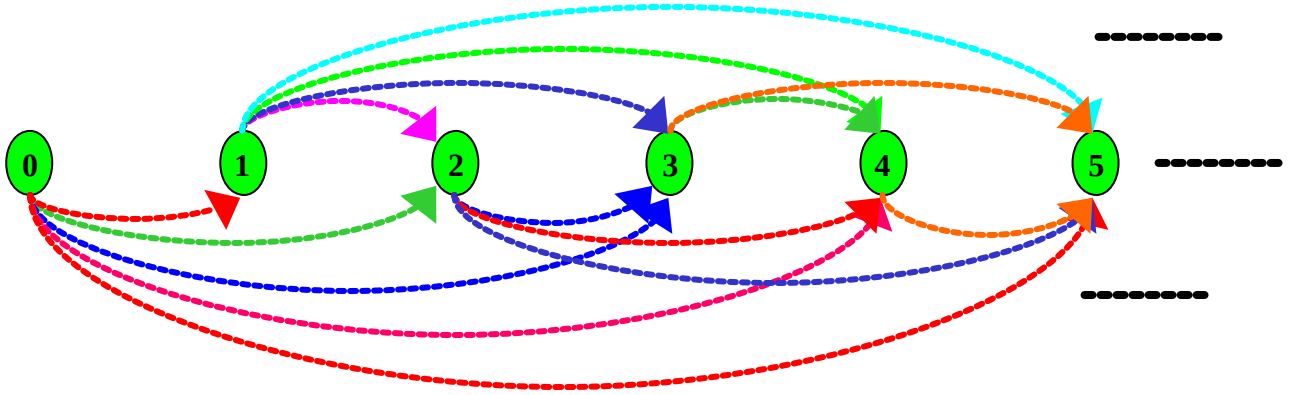


Ramsey's Theorem (1930)

A = { [a,b] | a<b} pairs

L = {f_1, . . . f_n} colors

F : A -> L coloring



Ramsey's Theorem

$A = \{ [a,b] \in N \times N \mid a<b \}$

$L = \{f_1, \ldots f_n\}$ colors

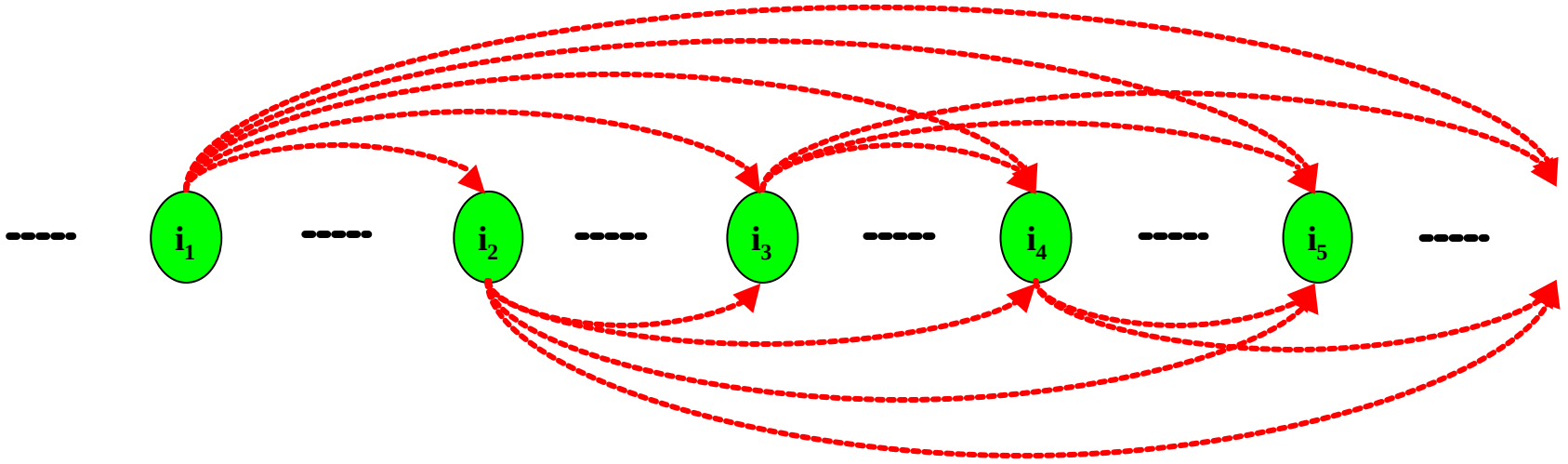
$F : A \rightarrow L$ coloring

$\Rightarrow$

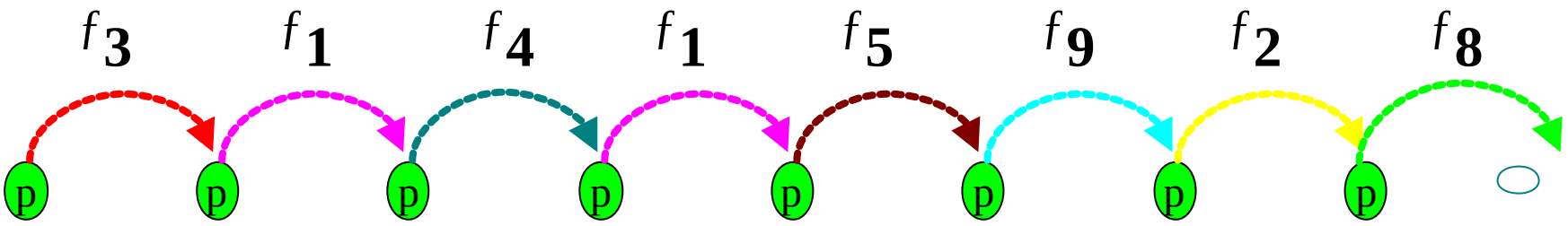
$\exists$ infinite set $X \subseteq \mathbf{N}$

$\exists$ color $f \in L$ s.t.

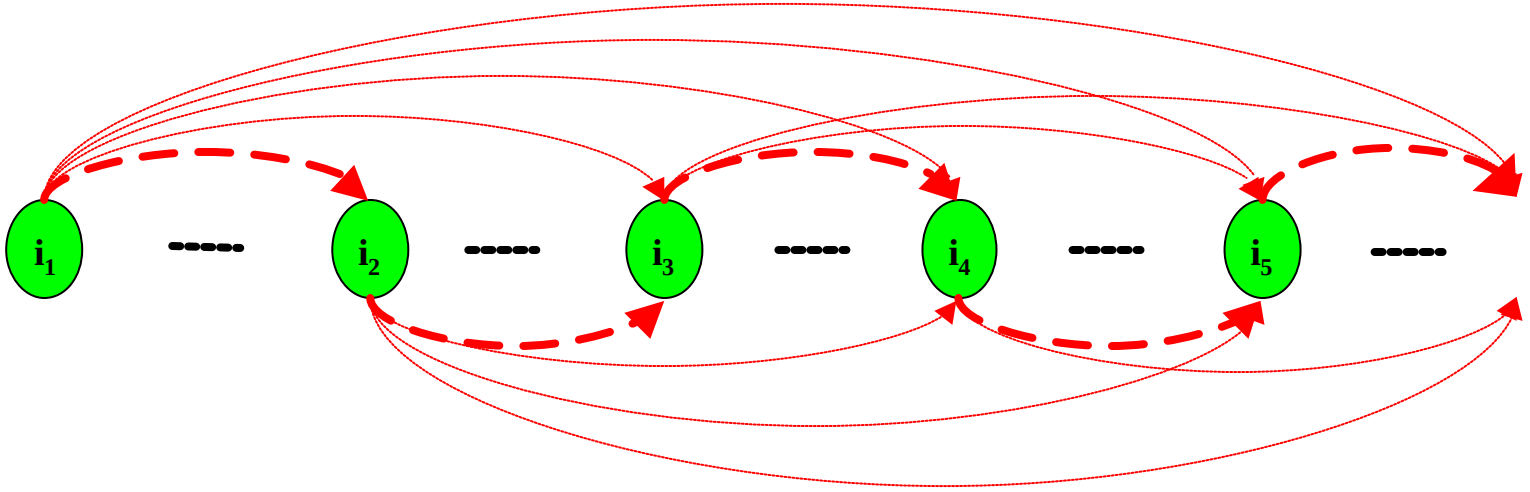
$F(a,b) = f$ forall $a<b \in X$



Applying Ramsey's Theorem



Somewhere along the line,...



And the credit goes to . . .

In the last two years all three groups independently gave proofs based on Ramsey's Theorem. The result should be credited to:

A General Framework for the Automatic Termination Analysis of Logic Programs; Nachum Dershowitz, Naomi Lindenstrauss, Yehoshua Sagiv, Alexander Serebrenik; Applicable Algebra in Engineering, Communication and Computing; 12, 117-156 (2001).

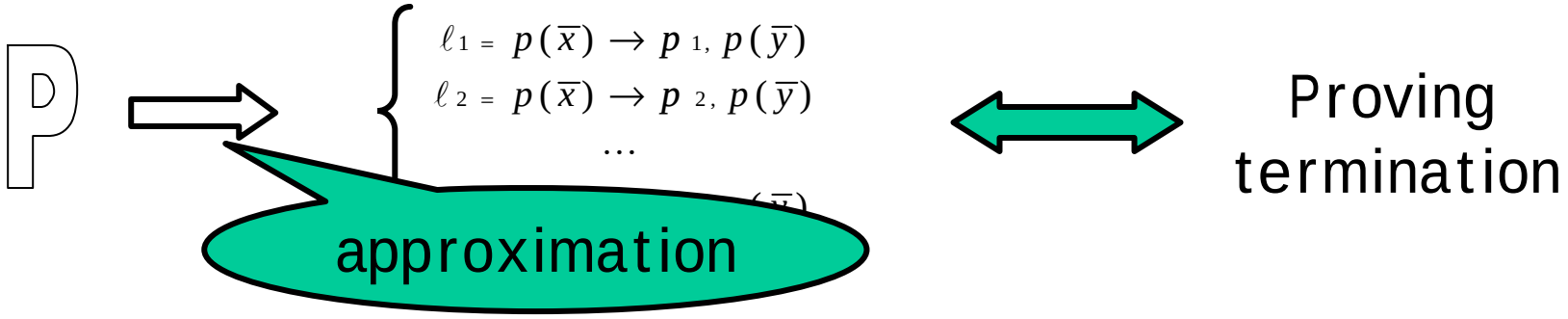
. . . Dershowitz et al.

Summary (Part II)

- Loops are defined semantically and approximated through program analysis.
- Convergents are local
- The technique is sound (assuming loops are closed under composition)

  Correctness of the program analysis &

  the argument based on Ramsey's theorem

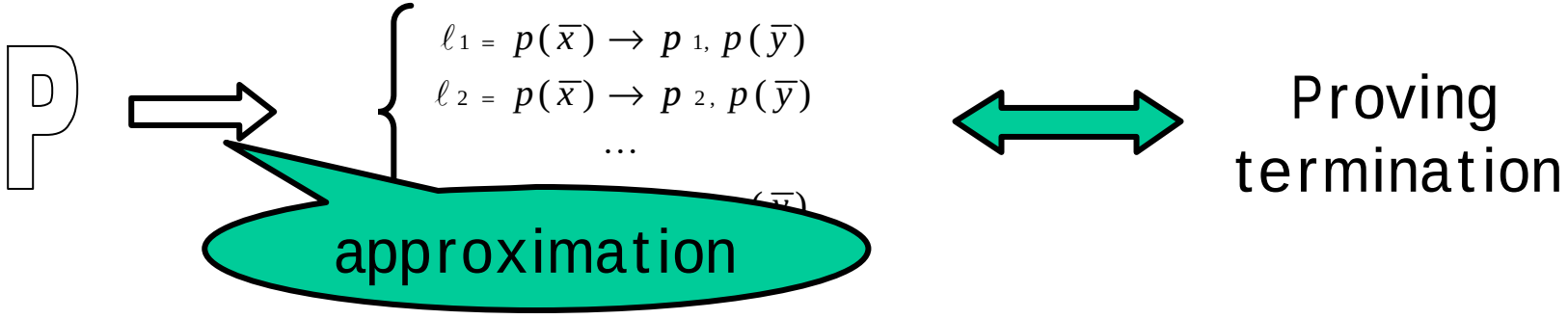


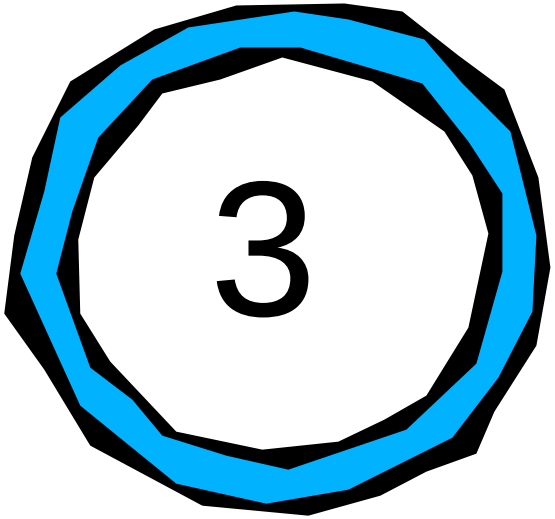
Summary (Part II - continued)

- We have lost completeness (due to approximation)

- The technique is complete **w.r.t.** the approximation

  If there exist convergents (local or global) which imply termination based on the loop descriptions then we can find them. (Constructive).

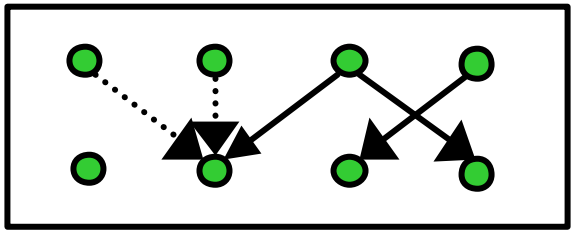




A comparison

Local Convergents are Simple

$$p(\bar{x}) \rightarrow p, p(\bar{y})$$



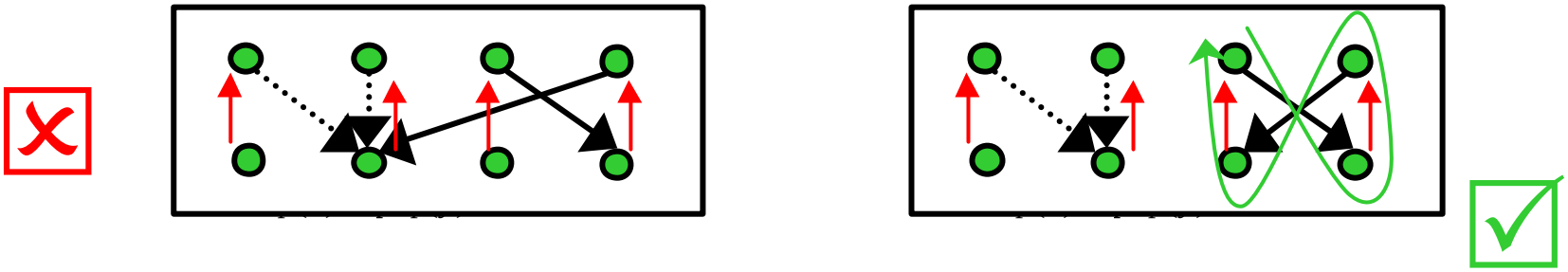
If there exists a function f such that

$$p \models f(\bar{x}) > f(\bar{y})$$

then there exists such a function of the form

$$f(\bar{x}) = \sum_{i \in I} x_i$$

Detecting the Existence of Local Convergents



There exists a function such that $p \models f(\bar{x}) > f(\bar{y})$

iff

1. adding up arrows leads to a cycle (termilog).

How hard is it to detect the existence of a global convergent?

look for local convergents instead …
(this implies the existence of a global convergent)

How hard is it to construct a global convergent ?

$O(|loops|^2)$

Both answers assume closure of loops!

Local Convergents are Simple;

Global Convergents are not too Complex

Conjecture given a finite set of loop descriptions

$$p(\bar{x}) \rightarrow p\,,\, p(\bar{y})$$

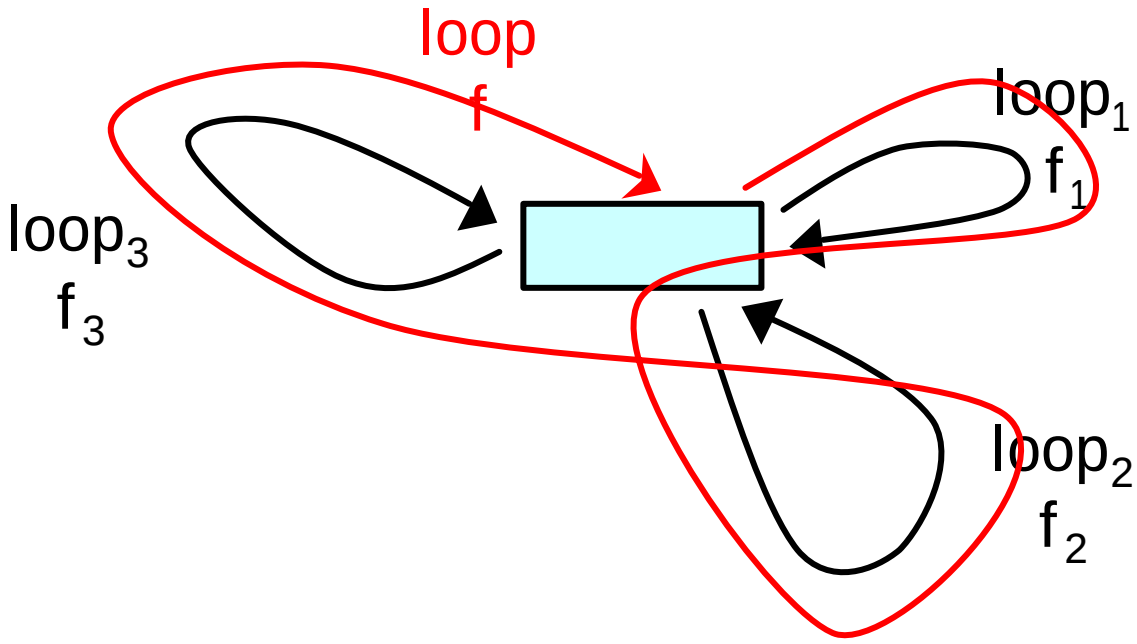
If there exists a global convergent then it is of the form

$$f(\bar{x}) = \left\langle f_1(\bar{x}), \ldots, f_k(\bar{x}) \right\rangle$$

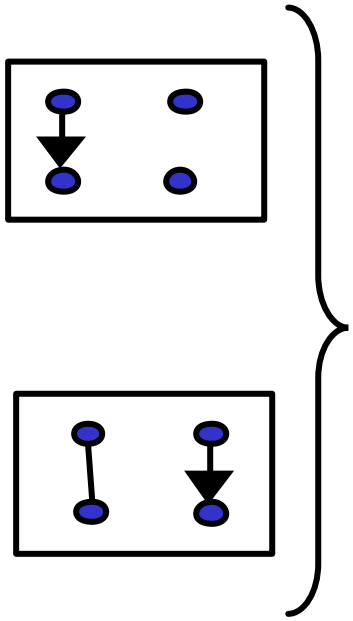
with each of the f_i a sum, min or max of arguments.

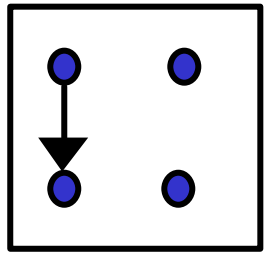
Constructing a global convergent – part I

Choose f_1, f_2, f_3 so that their composition . . .



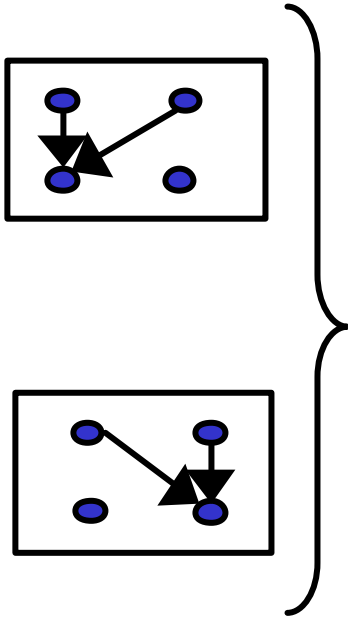
. . . satisfies that at least one of the loops decreases on f and none increase.

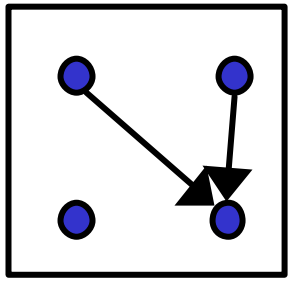


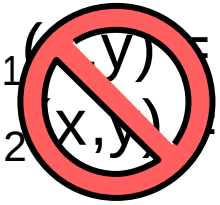


$f_1(x,y) = x$
$f_2(x,y) = y$

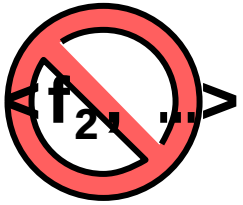
<f_1, ...>







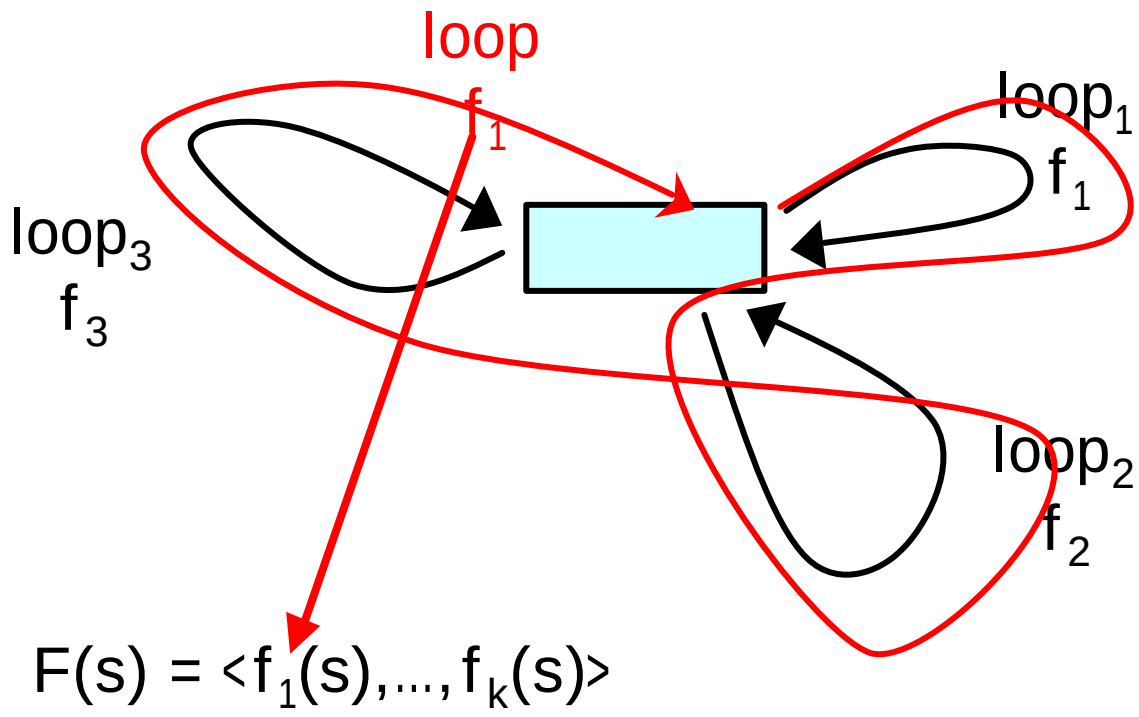
$f_1(x,y) = x$
$f_2(x,y) = y$



$f_1(x,y) = \min(x,y)$
$f_2(x,y) = \min(x,y)$

<f_1, ...>

Now a global convergent can be constructed "efficiently" by ordering (some of) the local convergents as a tuple



Start by composing all loops

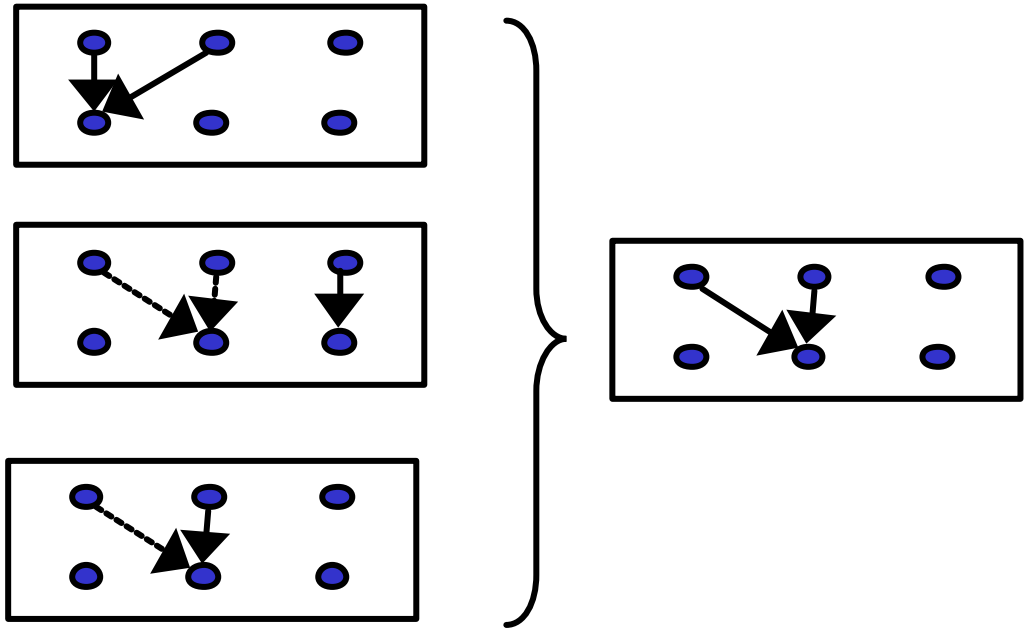
If f_1 (it has to be one of those being composed) then

put it in the tuple and remove any loop(s) which decrease for f_1

Add new constraints; Repeat until no more loops

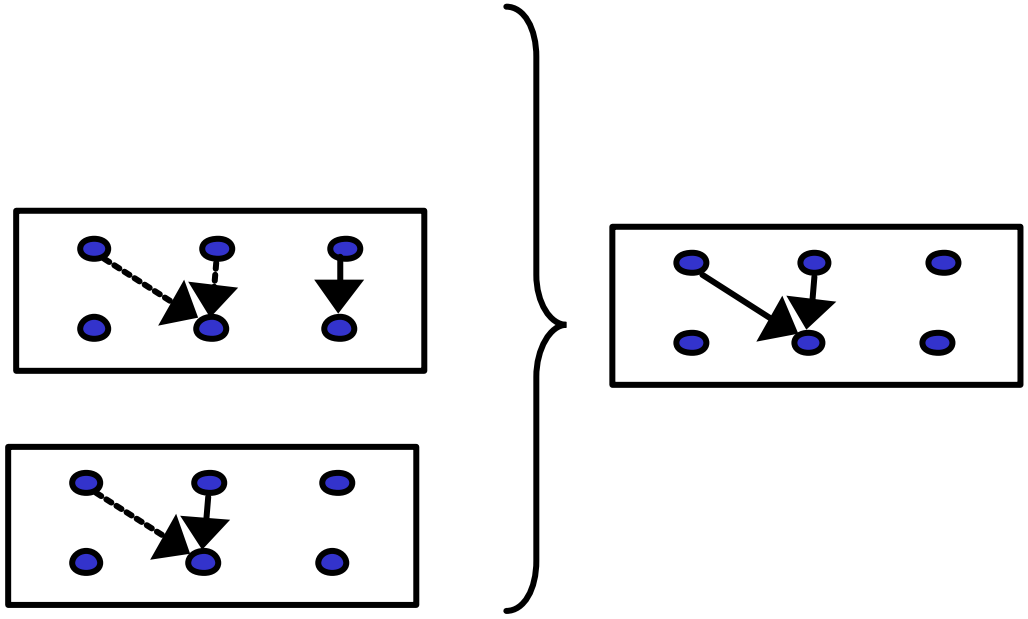
"efficently" means in the number of loops

Example



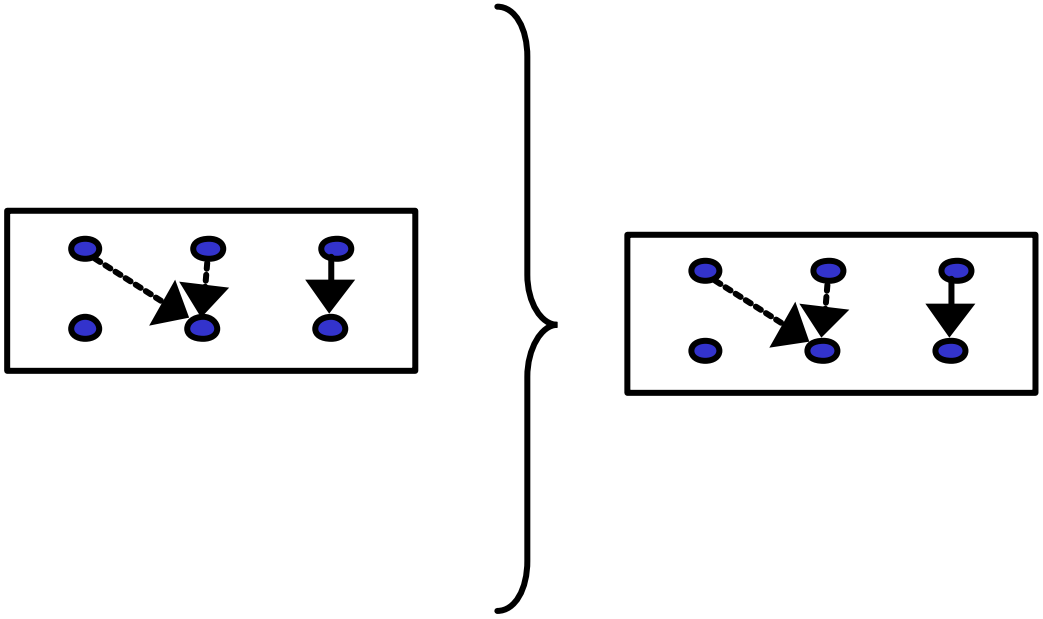
f(x,y,z) = <min(x,y), . . . >

Example



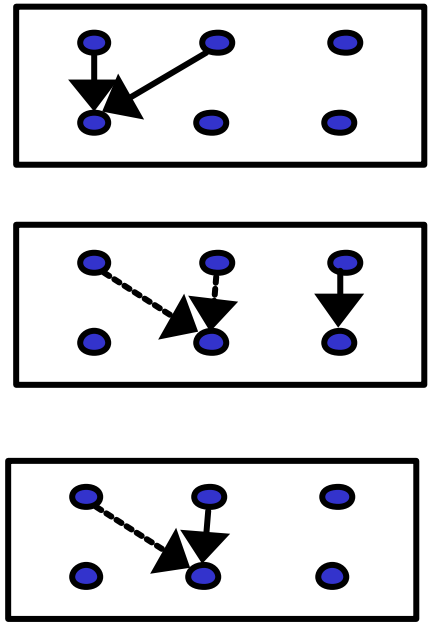
f(x,y,z) = <min(x,y), y . . . >

Example



f(x,y,z) = <min(x,y), y , z >

Example



f(x,y,z) = <min(x,y), y, z >

Conclusion (biased)

Local convergents are simplier

To find global convergents one needs also to reason about local convergents

Exponential worst case cost for closure of loops under composition is rare in practice

We have looked at the case where size is described using monotonicity constraints

What about a richer abstract domain (linear constraints)?

http://www.cs.bgu.ac.il/~mcodish/TerminWeb