

Abstract Interpretation over Non-Lattice Abstract Domains

Graeme Gange Jorge A. Navas
Peter Schachte Harald Søndergaard
Peter J. Stuckey

Department of Computing and Information Systems,
The University of Melbourne, Victoria 3010, Australia
`{gkgange, jorge.navas, schachte, harald, pjs}@unimelb.edu.au`

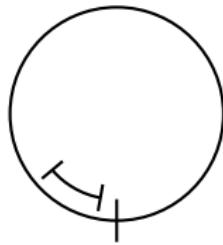
June 20, 2013

Outline

- 1 Motivation
- 2 Non-lattice Domains
- 3 Non-determinism and Precision Loss
- 4 Non-termination
- 5 Conclusion

Consider: Interval analysis

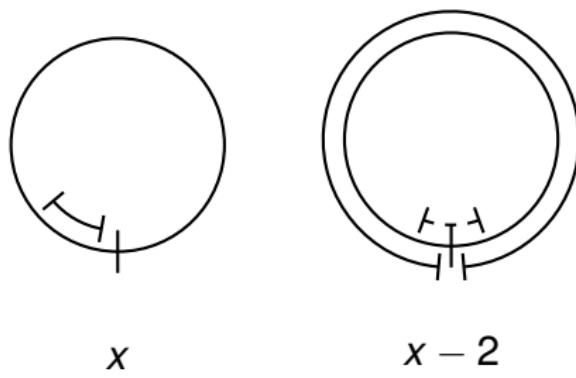
Classical interval analysis suffers from sudden information loss.



x

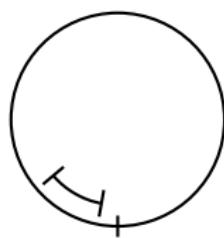
Consider: Interval analysis

Classical interval analysis suffers from sudden information loss.



Consider: 'Wrapped' intervals

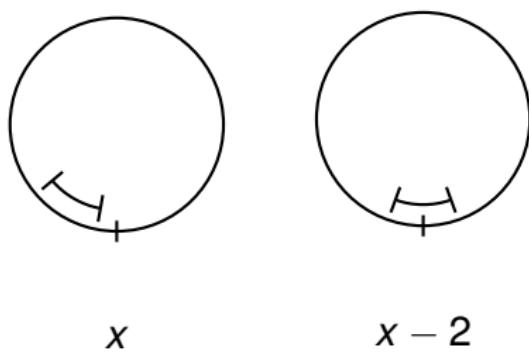
Similar to classical intervals, but without a fixed 'pole'.



x

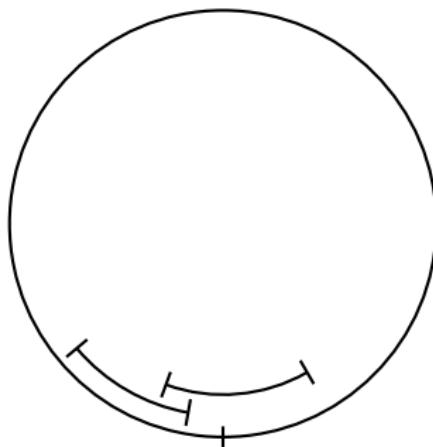
Consider: 'Wrapped' intervals

Similar to classical intervals, but without a fixed 'pole'.



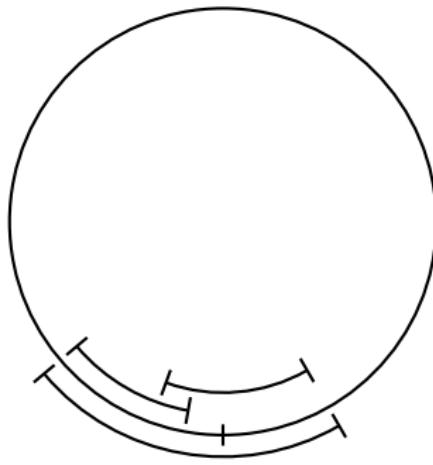
Consider: 'Wrapped' intervals

What about the join?



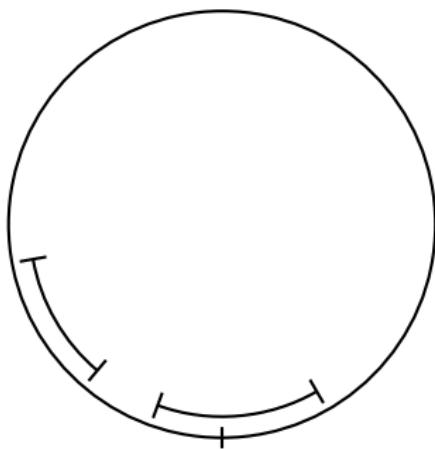
Consider: 'Wrapped' intervals

What about the join?



Consider: 'Wrapped' intervals

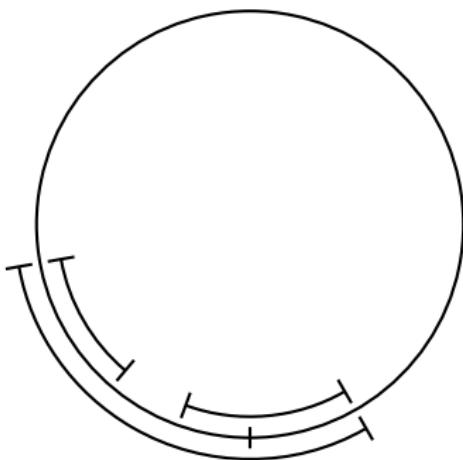
What about the join?



Pick the smaller interval...

Consider: 'Wrapped' intervals

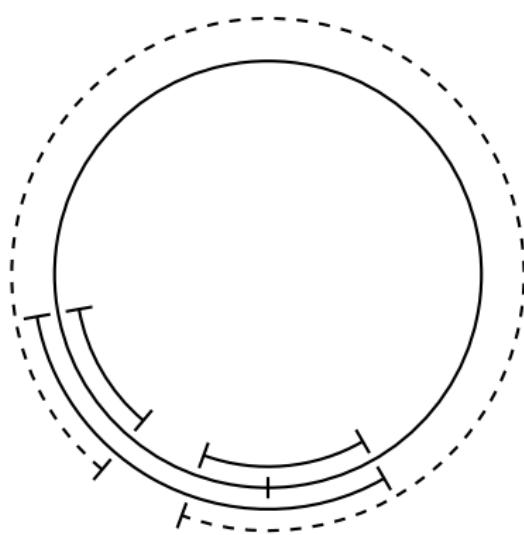
What about the join?



Pick the smaller interval...

Consider: 'Wrapped' intervals

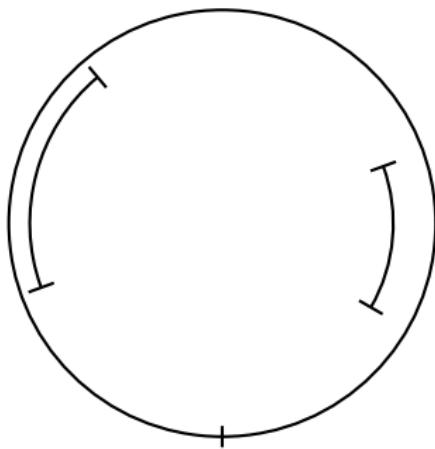
What about the join?



Pick the smaller interval...

Consider: 'Wrapped' intervals

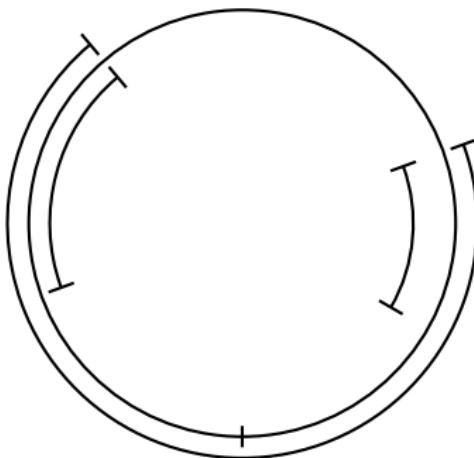
What about the join?



...and break ties deterministically.

Consider: 'Wrapped' intervals

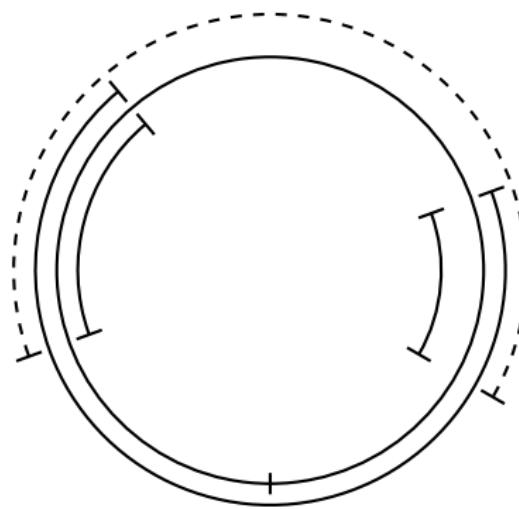
What about the join?



...and break ties deterministically.

Consider: 'Wrapped' intervals

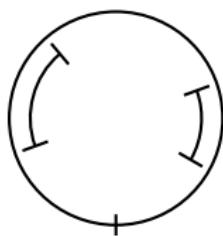
What about the join?



...and break ties deterministically.

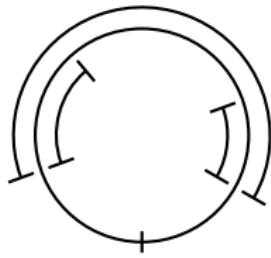
Consider: 'Wrapped' intervals

Except...



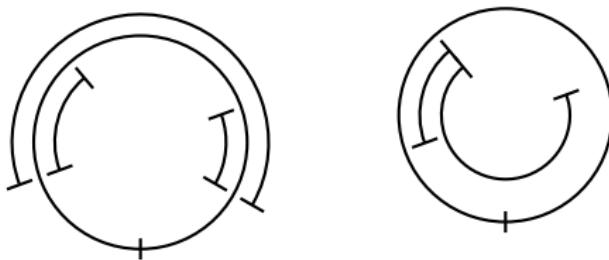
Consider: 'Wrapped' intervals

Except...



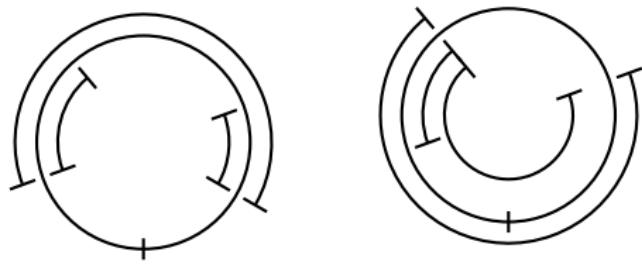
Consider: 'Wrapped' intervals

Except...



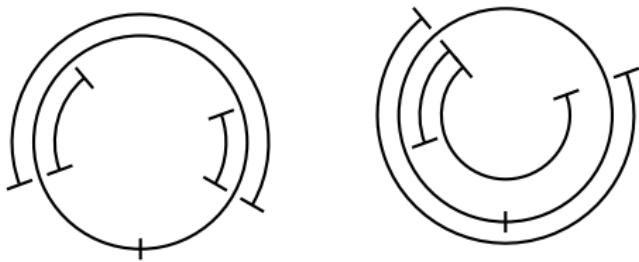
Consider: 'Wrapped' intervals

Except...



Consider: 'Wrapped' intervals

Except...



...this is non-monotone.

A recurring pattern

This sequence is increasingly common:

- A simple lattice domain is inadequate (imprecise, ...)
- A powerset domain is too expensive
- We design a new domain, which turns out not to be a lattice

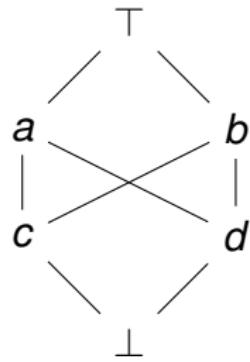
Some recent domains display similar properties:

- Wrapped Intervals
- Array segmentation domains
(Cousot, Cousot & Logozzo, 2011)
- Donut domains
(Ghorbal et.al., 2012)

Domains and butterflies

We are interested in *quasi-lattices* – pointed partial orders which lack a *least upper* (or lower) bound for some pairs elements.

A quasi-lattice necessarily contains a *butterfly* substructure:



Note that any upper bound \sqcup such that $x \sqsubseteq y \Rightarrow x \sqcup y = y$ is non-monotone.

But it gets worse

Now that \sqsubseteq is non-monotone, we have several major problems:

- Non-determinism
- Loss of precision

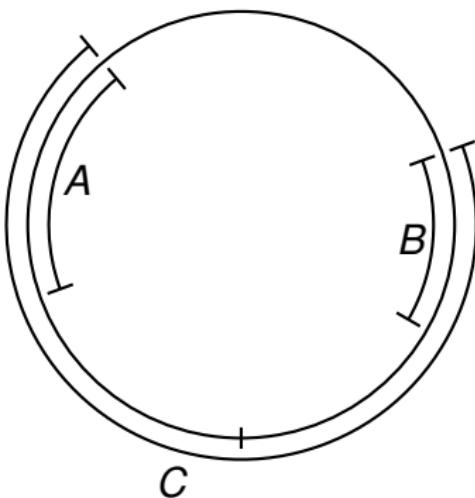
But it gets worse

Now that \sqsubset is non-monotone, we have several major problems:

- Non-determinism
- Loss of precision
- **Non-termination**

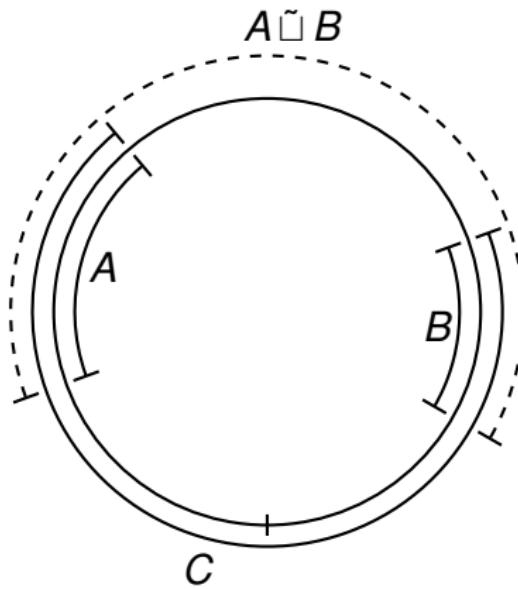
Non-determinism

Consider again:

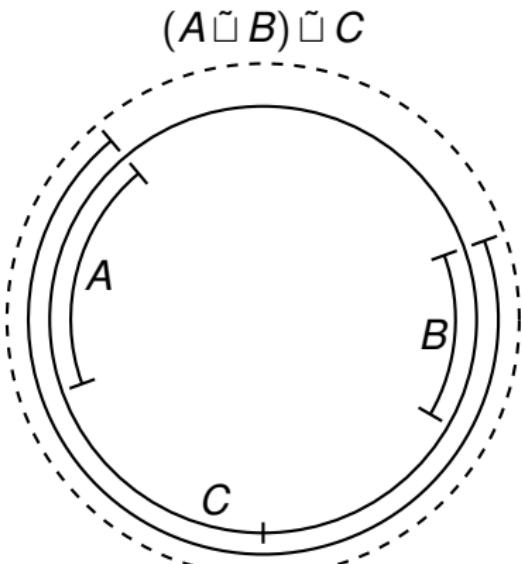


Non-determinism

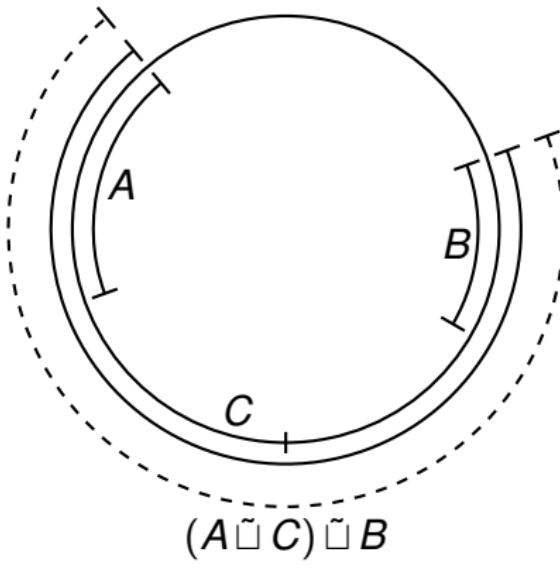
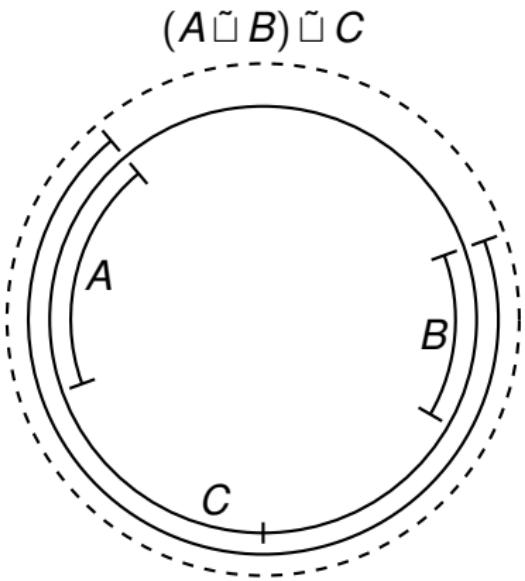
Consider again:



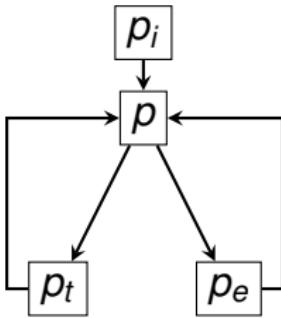
Non-determinism



Non-determinism



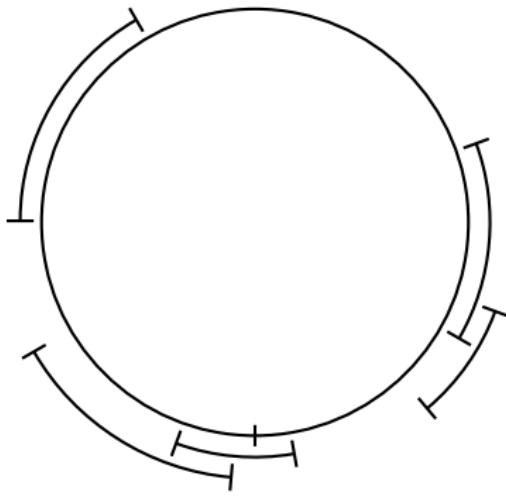
Non-determinism



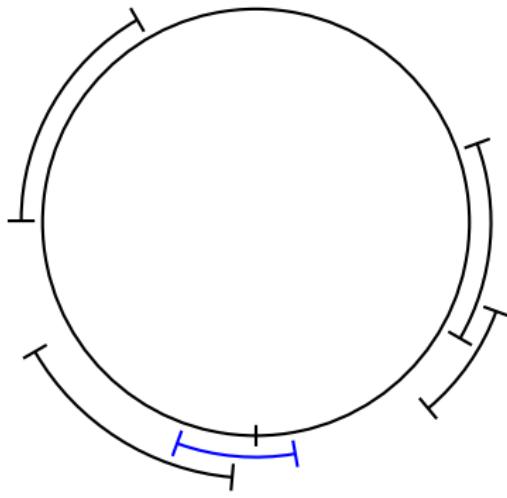
Can we ensure that we find a minimal fixpoint of p ?

- In general, no.
- We can recover some precision by defining an n-ary \sqcup , and delaying joins.
- The result is still volatile with respect to the queueing strategy.

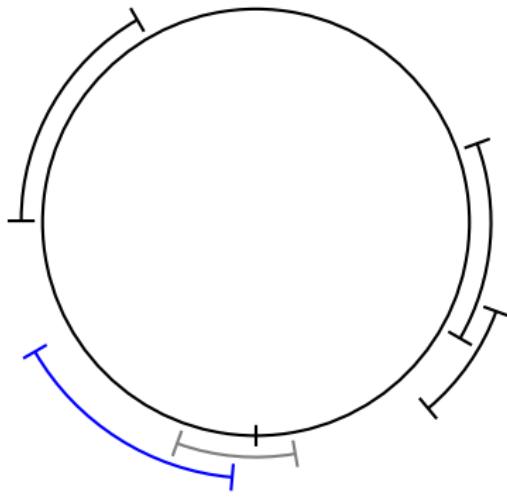
Specialized \sqcup



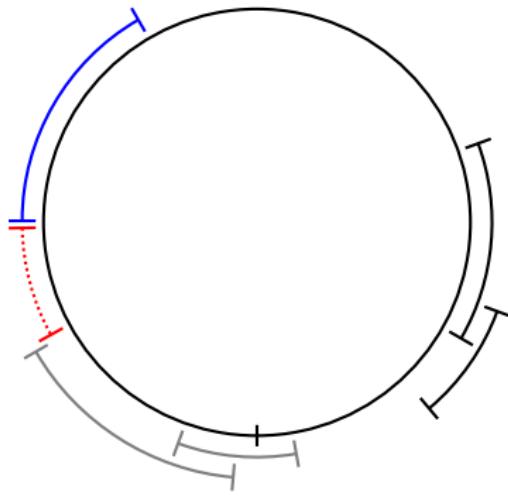
Specialized \sqcup



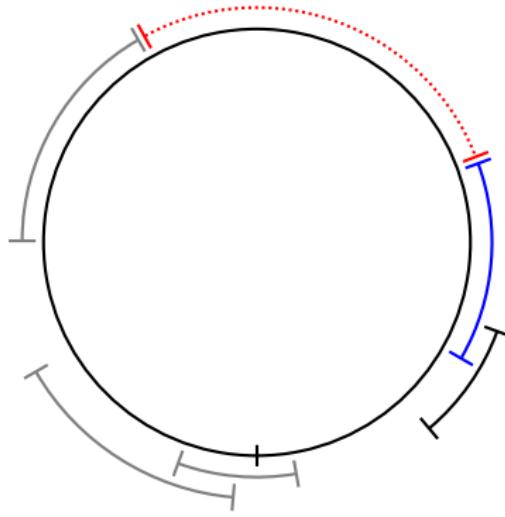
Specialized \sqcup



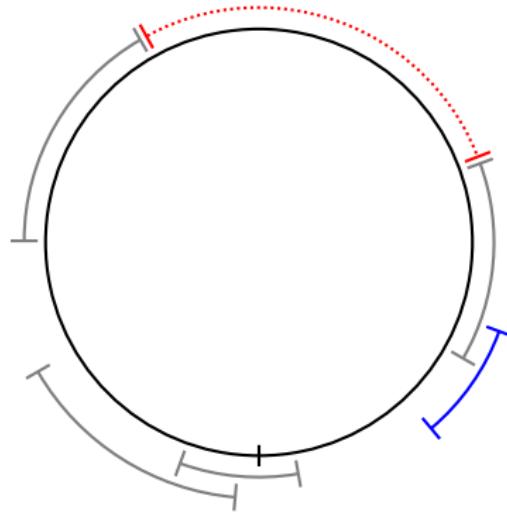
Specialized \sqcup



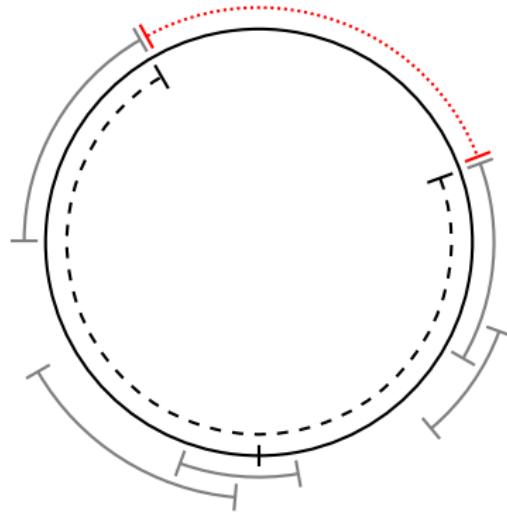
Specialized \sqcup



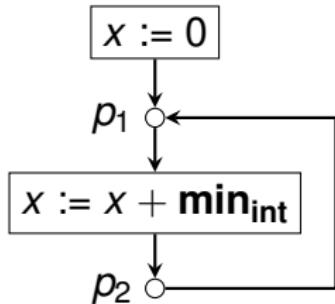
Specialized \sqcup



Specialized \sqcup



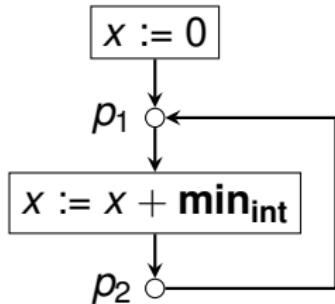
Non-termination



$p_1 \quad (0, 0)$

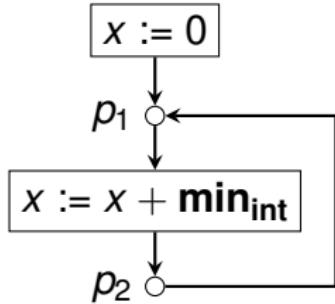
$p_2 \quad (\text{min}_{\text{int}}, \text{min}_{\text{int}})$

Non-termination



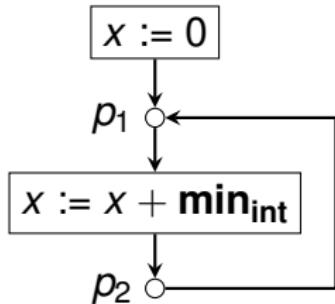
p_1	$(0, 0)$	$(0, \text{min}_{\text{int}})$
p_2	$(\text{min}_{\text{int}}, \text{min}_{\text{int}})$	$(\text{min}_{\text{int}}, 0)$

Non-termination



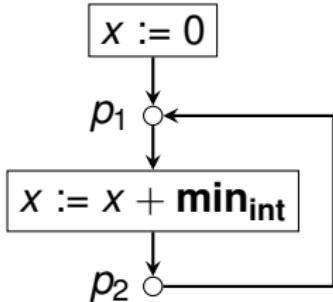
p_1	$(0, 0)$	$(0, \text{min}_{\text{int}})$	$(\text{min}_{\text{int}}, 0)$
p_2	$(\text{min}_{\text{int}}, \text{min}_{\text{int}})$	$(\text{min}_{\text{int}}, 0)$	$(0, \text{min}_{\text{int}})$

Non-termination



p_1	$(0, 0)$	$(0, \text{min}_{\text{int}})$	$(\text{min}_{\text{int}}, 0)$	$(0, \text{min}_{\text{int}})$
p_2	$(\text{min}_{\text{int}}, \text{min}_{\text{int}})$	$(\text{min}_{\text{int}}, 0)$	$(0, \text{min}_{\text{int}})$	$(\text{min}_{\text{int}}, 0)$

Non-termination



p_1	$(0, 0)$	$(0, \text{min}_{\text{int}})$	$(\text{min}_{\text{int}}, 0)$	$(0, \text{min}_{\text{int}})$	$(\text{min}_{\text{int}}, 0)$	\dots
p_2	$(\text{min}_{\text{int}}, \text{min}_{\text{int}})$	$(\text{min}_{\text{int}}, 0)$	$(0, \text{min}_{\text{int}})$	$(\text{min}_{\text{int}}, 0)$	$(\text{min}_{\text{int}}, 0)$	

We can abuse any other non-lattice domain in a similar fashion.
This is far from ideal.

Non-termination: Solutions

Luckily, we can several ways to avoid non-termination:

- Using widening operators
- Ensured climbing
- Loop checking

Widening

If we're using an infinite height lattice, the widening operator will *probably* take us to \top eventually.

But be careful, since widening operators are only guaranteed to stabilize for *increasing chains*, which we don't have.

Ensured climbing

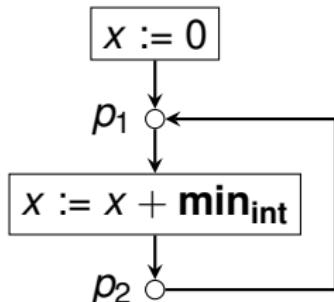
We can modify our data-flow equations:

$$p_i = f(p_1, \dots, p_n) \mapsto p_i = p_i \sqcup f(p_1, \dots, p_n)$$

Since \sqcup is an upper bound, this is guaranteed to be an ascending chain.

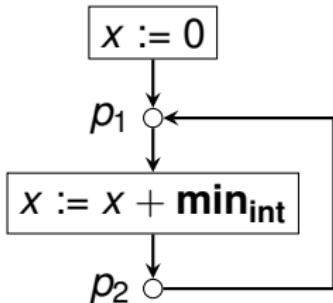
Unfortunately, we still lose precision.

Ensured climbing



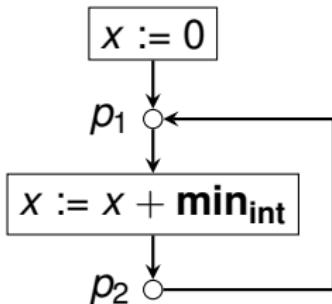
p_1	$(0, 0)$
p_2	$(\text{min}_{\text{int}}, \text{min}_{\text{int}})$

Ensured climbing



p_1	$(0, 0)$	$(0, \text{min}_{\text{int}})$
p_2	$(\text{min}_{\text{int}}, \text{min}_{\text{int}})$	$(\text{min}_{\text{int}}, 0)$

Ensured climbing



p_1	$(0, 0)$	$(0, \text{min}_{\text{int}})$	\top
p_2	$(\text{min}_{\text{int}}, \text{min}_{\text{int}})$	$(\text{min}_{\text{int}}, 0)$	\top

Well, at least it terminates.

Cycle detection

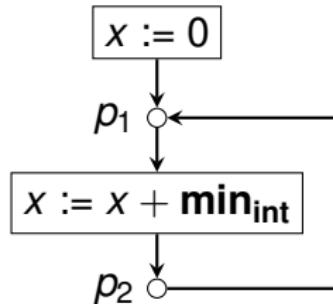
If Kleene iteration cycles, any point in the cycle is a valid approximation of fp_γ .

In fact, given a cycle $[\psi_1, \dots, \psi_n]$, the concrete fixpoint is below:

$$\gamma(\psi_1) \sqcap \dots \sqcap \gamma(\psi_n)$$

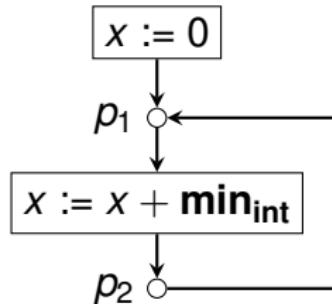
Cycle detection can be added to naive Kleene iteration with a constant-factor overhead in both time and space.

Cycle detection



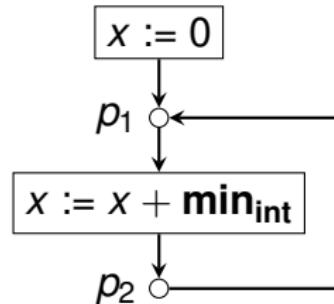
	H	T
p_1	$(0, 0)$	$(0, 0)$
p_2	$(\text{min}_{\text{int}}, \text{min}_{\text{int}})$	$(\text{min}_{\text{int}}, \text{min}_{\text{int}})$

Cycle detection



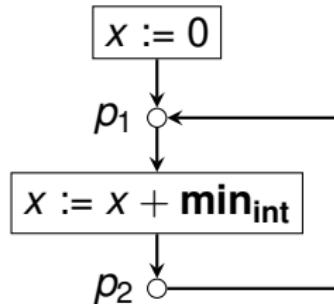
	H	T
p_1	$(0, \mathbf{min}_{\mathbf{int}})$	$(0, 0)$
p_2	$(\mathbf{min}_{\mathbf{int}}, 0)$	$(\mathbf{min}_{\mathbf{int}}, \mathbf{min}_{\mathbf{int}})$

Cycle detection



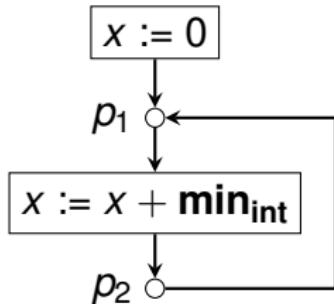
	H	T
p_1	$(\mathbf{min}_{\mathbf{int}}, 0)$	$(0, 0)$
p_2	$(0, \mathbf{min}_{\mathbf{int}})$	$(\mathbf{min}_{\mathbf{int}}, \mathbf{min}_{\mathbf{int}})$

Cycle detection



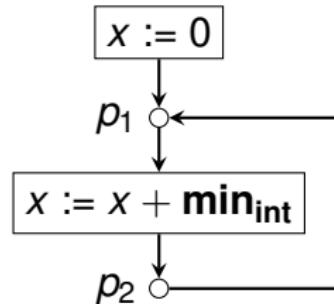
	H	T
p_1	$(\text{min}_{\text{int}}, 0)$	$(0, \text{min}_{\text{int}})$
p_2	$(0, \text{min}_{\text{int}})$	$(\text{min}_{\text{int}}, 0)$

Cycle detection



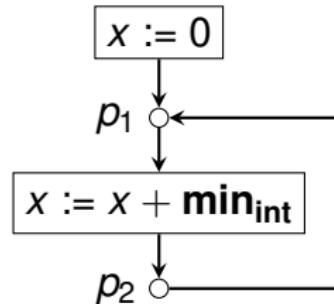
	H	T
p_1	$(0, \mathbf{min}_{\mathbf{int}})$	$(0, \mathbf{min}_{\mathbf{int}})$
p_2	$(\mathbf{min}_{\mathbf{int}}, 0)$	$(\mathbf{min}_{\mathbf{int}}, 0)$

Cycle detection



	H	T
p_1	$(\mathbf{min}_{\mathbf{int}}, 0)$	$(0, \mathbf{min}_{\mathbf{int}})$
p_2	$(0, \mathbf{min}_{\mathbf{int}})$	$(\mathbf{min}_{\mathbf{int}}, 0)$

Cycle detection



	H	T
p_1	$(\mathbf{min}_{\mathbf{int}}, 0)$	$(\mathbf{min}_{\mathbf{int}}, 0)$
p_2	$(0, \mathbf{min}_{\mathbf{int}})$	$(0, \mathbf{min}_{\mathbf{int}})$

Cycle detection

The elements of the cycle are:

$$\begin{pmatrix} p_1 \\ p_2 \end{pmatrix} = \left\{ \begin{array}{l} (\mathbf{min}_{int}, 0) \\ (0, \mathbf{min}_{int}) \\ (0, \mathbf{min}_{int}) \\ (\mathbf{min}_{int}, 0) \end{array} \right\}$$

Cycle detection

The elements of the cycle are:

$$\begin{pmatrix} p_1 \\ p_2 \end{pmatrix} = \left\{ \begin{array}{l} (\mathbf{min}_{\text{int}}, 0) \\ (0, \mathbf{min}_{\text{int}}) \end{array}, \begin{array}{l} (0, \mathbf{min}_{\text{int}}) \\ (\mathbf{min}_{\text{int}}, 0) \end{array} \right\}$$

If we take the meet in the concrete domain, we get:

$$\begin{pmatrix} p_1 \\ p_2 \end{pmatrix} = \begin{pmatrix} \{0, \mathbf{min}_{\text{int}}\} \\ \{0, \mathbf{min}_{\text{int}}\} \end{pmatrix}$$

Summary

When designing/using non-lattice domains, we must be wary.

Non-monotone joins cause:

- Volatile fixpoints
- Non-termination

We can reduce volatility, but not cure it in general.

Non-termination can be solved by:

- Using widening
- Transforming the data-flow equations
- Adding cycle detection