# A Family of Sims with Diverging Interests

POPL'26, Rennes, France

Nicolas Chappe (Post-doc @ CNRS / Verimag)

January 15, 2026

# Program comparison

Tools to *compare* programs play a major role in programming language theory

Program $p_1$

Program $p_2$

# Program comparison

Tools to *compare* programs play a major role in programming language theory

Program $p_1$

equivalent to?

Program $p_2$

# Program comparison

Tools to *compare* programs play a major role in programming language theory

Program $p_1$

equivalent to?
more general than?

Program $p_2$

# Program comparison

Tools to *compare* programs play a major role in programming language theory

Program $p_1$

equivalent to?
more general than?
just different from?

Program $p_2$

# Program comparison

Tools to *compare* programs play a major role in programming language theory

Program $p_1$

equivalent to?
more general than?
just different from?

Program $p_2$

- **Simulation** is a notion of program comparison (more specifically program *refinement*) that enables **local reasoning**

# Program comparison

Tools to *compare* programs play a major role in programming language theory

Program $p_1$

equivalent to?
more general than?
just different from?

Program $p_2$

- **Simulation** is a notion of program comparison (more specifically program *refinement*) that enables **local reasoning**
- Useful for concurrency theory, model checking, **verified compilation**, etc.

# Program comparison

Tools to *compare* programs play a major role in programming language theory

Program $p_1$

equivalent to?
more general than?
just different from?

Program $p_2$

- **Simulation** is a notion of program comparison (more specifically program *refinement*) that enables **local reasoning**
- Useful for concurrency theory, model checking, **verified compilation**, etc.
- In a verified compiler, the compiled program *refines* the source program

# Program comparison

Tools to *compare* programs play a major role in programming language theory

Program $p_1$

equivalent to?
more general than?
just different from?

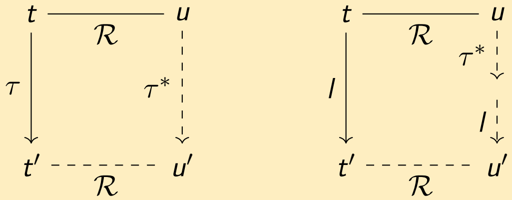
Program $p_2$

- **Simulation** is a notion of program comparison (more specifically program *refinement*) that enables **local reasoning**
- Useful for concurrency theory, model checking, **verified compilation**, etc.
- In a verified compiler, the compiled program *refines* the source program
- Programs typically modeled as labeled transition systems (LTSs)

# Which simulation? Two key properties

*Weak*

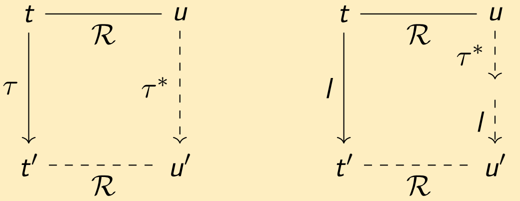
Some computations ($\tau$ transitions) are semantically invisible and do not need to be matched on the other side



# Which simulation? Two key properties

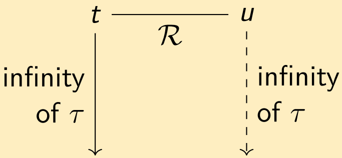
## *Weak*

Some computations ($\tau$ transitions) are semantically invisible and do not need to be matched on the other side




## *Divergence sensitive*

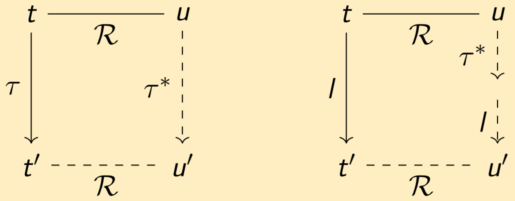
A non-diverging program should not be compiled to a possibly diverging program



# Which simulation? Two key properties

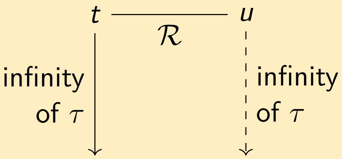
## *Weak*

Some computations ($\tau$ transitions) are semantically invisible and do not need to be matched on the other side




## *Divergence sensitive*

A non-diverging program should not be compiled to a possibly diverging program

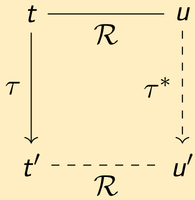


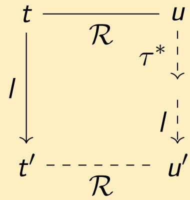

→ Straightforward answer: divergence-sensitive weak simulation

# Which simulation? Two key properties

## *Weak*

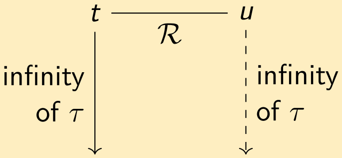
Some computations ($\tau$ transitions) are semantically invisible and do not need to be matched on the other side




## *Divergence sensitive*

A non-diverging program should not be compiled to a possibly diverging program
**This is a global condition.**




$\rightarrow$ Straightforward answer: divergence-sensitive weak simulation…maybe not

# Normed simulation

- This limitation was noted in 1998!

**Normed Simulations**

David Griffioen[1,2*] Frits Vaandrager[2]

[1] CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

[2] Computing Science Institute, University of Nijmegen
P.O. Box 9010, 6500 GL Nijmegen, The Netherlands
{davidg,fvaan}@cs.kun.nl

Thus the research program to reduce global reasoning to local reasoning has not been carried out to its completion.

# Normed simulation

- This limitation was noted in 1998!

## Normed Simulations

David Griffioen[1,2*] Frits Vaandrager[2]

[1] CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

[2] Computing Science Institute, University of Nijmegen
P.O. Box 9010, 6500 GL Nijmegen, The Netherlands
{davidg,fvaan}@cs.kun.nl

Thus the research program to reduce global reasoning to local reasoning has not been carried out to its completion.

- Normed simulation enables local reasoning through a decreasing measure
- It shaped most later notions of simulation for verified compilation
- CompCert relies on it

# Towards a better notion of simulation?

| | Introduced | Usability | Completeness |
|---|---|---|---|
| Div. weak simulation | 1981? | $\times$ | $\checkmark$ |
| Normed simulation | 1998 | $\checkmark$ | $\times$ |
| What I want | This paper | $\checkmark$ | $\checkmark$ |

Normed simulations complete *only for deterministic LTSs*

# A Family of Sims with Diverging Interests

# A modern characterization of divergence-sensitive weak simulation

## Inspiring advances from the 2010's

- *Implicit* normed simulation (used in ITrees) is based on a mixed inductive-coinductive definition.
- *Weak-tau simulation* (from CompCertTSO), made of two mutually-defined relations, relates some programs that are not related by normed simulation.
- *Coinduction up-to companion* eases the definition of powerful reasoning techniques

$\rightarrow$ I combine all of this into a mutually coinductive notion dubbed $\mu$div-simulation.

# A modern characterization of divergence-sensitive weak simulation

## Inspiring advances from the 2010's

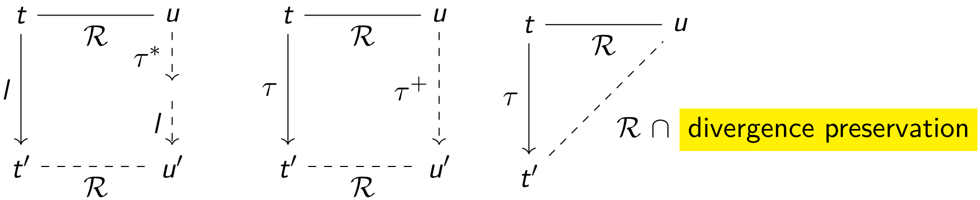
- *Implicit* normed simulation (used in ITrees) is based on a mixed inductive-coinductive definition.
- *Weak-tau simulation* (from CompCertTSO), made of two mutually-defined relations, relates some programs that are not related by normed simulation.
- *Coinduction up-to companion* eases the definition of powerful reasoning techniques

→ I combine all of this into a mutually coinductive notion dubbed $\mu$div-simulation.

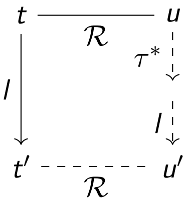
## $\mu$div-simulation

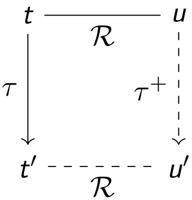
- Sound and complete wrt divergence-sensitive weak simulation
- Weaker (more complete) than variants of normed simulation
- As usable as implicit normed simulation thanks to coinduction up-to
- Defined in a generic LTS setting in ROCQ

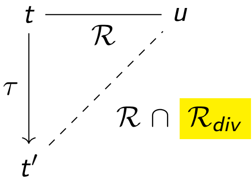
# $\mu$div-simulation, diagrammatically



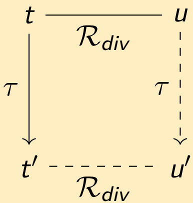
# $\mu$div-simulation, diagrammatically

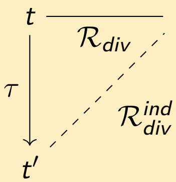



## Divergence preservation, coinductively



# How can we reason about it?

- Modern coinduction up-to
- A parameterized definition

# Coinduction up-to: Key idea

## Standard proof of simulation

- Exhibit a relation $\mathcal{R}$ between states.
- Prove that $\mathcal{R}$ is a simulation

# Coinduction up-to: Key idea

## Standard proof of simulation

- Exhibit a relation $\mathcal{R}$ between states.
- Prove that $\mathcal{R}$ is a simulation

## Proof using coinduction up-to

Start from a small $\mathcal{R}$ and lazily add pairs of states to it as needed during the proof.
$\rightarrow$ Interesting proof technique in an interactive proof assistant
Concretely, *up-to techniques* can transform the proof goal during the proof.

# Coinduction up-to: Key idea

## Standard proof of simulation

- Exhibit a relation $\mathcal{R}$ between states.
- Prove that $\mathcal{R}$ is a simulation

## Proof using coinduction up-to

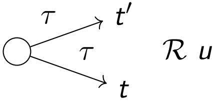
Start from a small $\mathcal{R}$ and lazily add pairs of states to it as needed during the proof.
$\rightarrow$ Interesting proof technique in an interactive proof assistant
Concretely, *up-to techniques* can transform the proof goal during the proof.

## History of coinduction up-to

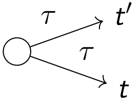
- Exists since the 80's.
- Comfortable in Rocq since the 2010's (`paco`, `coinduction`).

# Concrete up-to techniques

Consider this simulation goal:



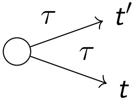
# Concrete up-to techniques

Consider this simulation goal: 

$\mathcal{R}\ u$

By the left up-to $\tau$ technique, this reduces to: $t\ \mathcal{R}\ u \wedge t'\ \mathcal{R}\ u$

# Concrete up-to techniques

Consider this simulation goal:  $\mathcal{R}\ u$

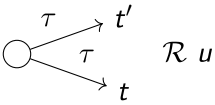


By the left up-to $\tau$ technique, this reduces to: $t\ \mathcal{R}\ u \wedge t'\ \mathcal{R}\ u$

## Asymmetric reasoning

- Left and right up-to $\tau$ and up-to $\epsilon$
- From the ITree/CTree world
- Recovers the proof rules from normed sim!

# Concrete up-to techniques

Consider this simulation goal: 


By the left up-to $\tau$ technique, this reduces to: $t \mathrel{\mathcal{R}} u \wedge t' \mathrel{\mathcal{R}} u$

## Asymmetric reasoning

- Left and right up-to $\tau$ and up-to $\epsilon$
- From the ITree/CTree world
- Recovers the proof rules from normed sim!

## Some other up-to techniques

- Transitivity and rewriting: complicated, 5 variations supported
- Well-known in the **bi**simulation literature (e.g., expansions)

# Is μdiv-simulation enough?

- We may need deadlock preservation too (easier)
- Strong simulation is easier to wield
- Some notions between weak and strong simulation can serve as proof devices

# Is µdiv-simulation enough?

- We may need deadlock preservation too (easier)
- Strong simulation is easier to wield
- Some notions between weak and strong simulation can serve as proof devices

**Problem**

That makes a lot of closely-related definitions of simulation.

# Is μdiv-simulation enough?

- We may need deadlock preservation too (easier)
- Strong simulation is easier to wield
- Some notions between weak and strong simulation can serve as proof devices

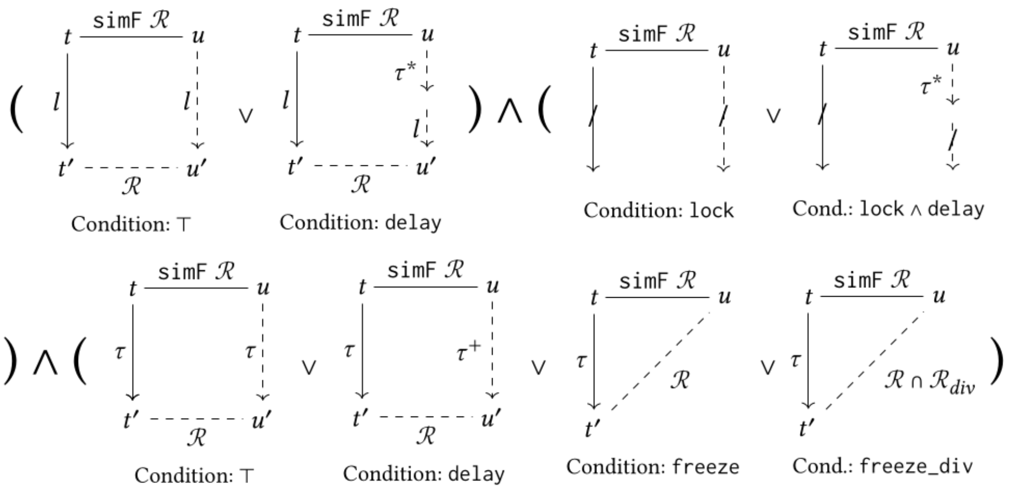
**Problem**

That makes a lot of closely-related definitions of simulation.

**Solution**

- Use a parameterized definition.
- Two Boolean parameters, one ternary parameter

- Parameterized definition: 12 notions (strong, weak, divergence-sensitive weak, dealock-sensitive, etc.) jointly studied



# Case studies

# Case study: A CompCert pass

- A few lines of Rocq to instantiate the theories
- Port of a 300-line CSE proof ($\sim$70 lines changed)
- Originally uses an *Eventually simulation*, analogous to the left up-to $\tau$ technique
- Forward $\implies$ backward simulation
- No need to explicitly build the backward simulation!

# Case study: Choice Trees

$$\frac{}{\mathsf{Ret}\ v\ \mathcal{R}\ \mathsf{Ret}\ v}\ (\mathtt{ret}) \qquad \frac{\forall x \in X, (k\ x)\ \mathcal{R}\ u \quad \mathcal{L} = \mathsf{nolock} \lor X\ \mathsf{inhabited}}{(\mathsf{Br}\ b\ k)\ \mathcal{R}\ u}\ (\mathtt{br\_l})$$

$$\frac{\exists y, t\ \mathcal{R}\ (k'\ y) \quad \mathcal{L} = \mathsf{nolock} \lor k'\ y \to \lor (\mathcal{F} = \mathsf{nofreeze} \land t \to)}{t\ \mathcal{R}\ (\mathsf{Br}\ b\ k')}\ (\mathtt{br\_r})$$

$$\frac{t\ \mathcal{R}\ u \quad \mathcal{D} = \mathsf{delay}}{t\ \mathcal{R}\ \mathsf{Step}\ u}\ (\mathtt{step\_r}) \qquad \frac{t\ \mathcal{R}\ u \quad \mathcal{F} = \mathsf{freeze} \lor (\mathcal{F} = \mathsf{freeze\_div} \land \mathsf{divpres}\ t\ u)}{\mathsf{simF}\ \mathcal{R}\ (\mathsf{Step}\ t)\ u}\ (\mathtt{step\_l})$$

$$\frac{t\ \mathcal{R}\ u \quad \mathcal{F} = \mathsf{freeze\_div}}{\mathsf{Step}\ t\ \mathcal{R}\ u}\ (\mathtt{step\_l'}) \qquad \frac{t\ \mathcal{R}\ u}{\mathsf{simF}\ \mathcal{R}\ (\mathsf{Step}\ t)\ (\mathsf{Step}\ u)}\ (\mathtt{step})$$

$$\frac{\forall v,\ (k\ v)\ \mathcal{R}\ (k'\ v)}{\mathsf{simF}\ \mathcal{R}\ (\mathsf{Vis}\ e\ k)\ (\mathsf{Vis}\ e\ k')}\ (\mathtt{vis}) \qquad \frac{t \not\to \land (\mathcal{L} = \mathsf{nolock} \lor u \not\to)}{t\ \mathcal{R}\ u}\ (\mathtt{stuck})$$

- A few lines of Rocq to instantiate the theories
- A single parameterized proof system for the 12 refinements
- Up-to bind technique

# Conclusion

# Conclusion

## Contributions

- Parameterized notion of simulation with up-to techniques
- Novel characterization of divergence preservation
- Implemented in 3.5k lines of ROCQ, using the `rocq-coinduction` library
- Version 0.2 released on opam as `rocq-sims`
- POPL'26 paper on my webpage: `https://www-verimag.imag.fr/~chappen/`

## Future work

- Extension to bisimulation
- Extension to trace inclusion

Thank you for your attention!