Checking the equivalence of context-free session types

Andreia Mordido and Vasco T. Vasconcelos

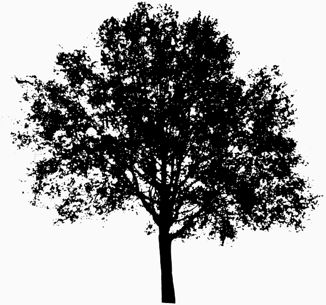
LASIGE, Faculdade de Ciências, Universidade de Lisboa

January 2019

Motivation

```
data Tree = Leaf | Node Int Tree Tree
```

```
sendTree Leaf c =
  select Leaf c
sendTree (Node x l r) c =
  let c1 = select Node c
      c2 = send x c1
      c3 = sendTree l c2
      c4 = sendTree r c3
   in c4
```



Plan

Definition (Type equivalence problem)

Given any context-free session types S and T, the type equivalence problem consists in deciding if types S and T are equivalent, i.e., $S \sim T$.

- Implement an algorithm that decides the *type equivalence problem*.
- Prove its soundness and completeness w.r.t. the metatheory of context-free session types proposed by Thiemann and Vasconcelos.
- Provide results on complexity.

Algorithm for checking the equivalence of CFST Main stages

Convert types to a grammar

Translates types into a (finite) set of productions

Prune unnormed productions

Streamlines the grammar by pruning unnormed productions

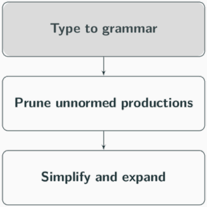
Simplify and expand

Alternates between simplification and expansion operations, until reaching a successful branch in the expansion tree or concluding that all branches are unsuccessful

Convert types to a grammar

We consider a finite set of productions $X \xrightarrow{a} \vec{Y}$ where:

- X, Y represent *non-terminal variables*
- a is a *label* in the labelled transition system



Context-free session types are seen as *simple grammars*, i.e.:

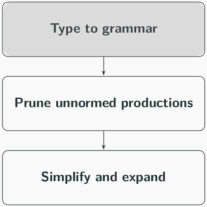
- context-free grammars in *Greibach normal form*
- s.t. for each X and a there is at most one production $X \xrightarrow{a} \vec{Y}$

Convert types to a grammar

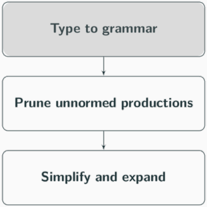
Example

$$S \triangleq (\mu x.\&\{n\colon x; x; ?\,\mathsf{int}, \ell\colon ?\,\mathsf{int}\}); (\mu z.!\,\mathsf{int}; z; z)$$

$$T \triangleq (\mu y.\&\{n\colon y; y, \ell\colon \mathsf{skip}\}; ?\,\mathsf{int}); (\mu z.!\,\mathsf{int}; z)$$



Convert types to a grammar



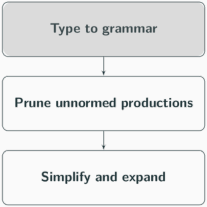
Example

$$S \triangleq (\mu x.\&\{n\colon x; x; ?\,\mathsf{int}, \ell\colon ?\,\mathsf{int}\}); (\mu z.!\,\mathsf{int}; z; z)$$

$$T \triangleq (\mu y.\&\{n\colon y; y, \ell\colon \mathsf{skip}\}; ?\,\mathsf{int}); (\mu z.!\,\mathsf{int}; z)$$

Productions for S	Productions for T
$X_1 \xrightarrow{\&n} X_1 X_1 X_2$	$Y_1 \xrightarrow{\&n} Y_1 Y_1 Y_2$
$X_1 \xrightarrow{\&\ell} X_3$	$Y_1 \xrightarrow{\&\ell} Y_2$
$X_2 \xrightarrow{?\,\mathsf{int}} \varepsilon$	$Y_2 \xrightarrow{?\,\mathsf{int}} \varepsilon$
$X_3 \xrightarrow{?\,\mathsf{int}} \varepsilon$	$Y_3 \xrightarrow{!\,\mathsf{int}} Y_3$
$X_4 \xrightarrow{!\,\mathsf{int}} X_4 X_4$	

Convert types to a grammar



Example

$$S \triangleq (\mu x.\&\{n\colon x; x; ?\,\mathsf{int}, \ell\colon ?\,\mathsf{int}\}); (\mu z.!\,\mathsf{int}; z; z)$$

$$T \triangleq (\mu y.\&\{n\colon y; y, \ell\colon \mathsf{skip}\}; ?\,\mathsf{int}); (\mu z.!\,\mathsf{int}; z)$$

Productions for S	Productions for T
$X \xrightarrow{!\,()} X_1 X_4$	$Y \xrightarrow{!\,()} Y_1 Y_3$
$X_1 \xrightarrow{\&n} X_1 X_1 X_2$	$Y_1 \xrightarrow{\&n} Y_1 Y_1 Y_2$
$X_1 \xrightarrow{\&\ell} X_3$	$Y_1 \xrightarrow{\&\ell} Y_2$
$X_2 \xrightarrow{?\,\mathsf{int}} \varepsilon$	$Y_2 \xrightarrow{?\,\mathsf{int}} \varepsilon$
$X_3 \xrightarrow{?\,\mathsf{int}} \varepsilon$	$Y_3 \xrightarrow{!\,\mathsf{int}} Y_3$
$X_4 \xrightarrow{!\,\mathsf{int}} X_4 X_4$	

Convert types to a grammar

`LOHaskellCode` $\sim$ `100`

Soundness ✓

1. Conversion of types into BPAs is sound[a].
2. Conversion of BPAs to Greibach Normal Form without altering the solution is sound[b].

[a] Thiemann and Vasconcelos. Context-free session types. 2016.
[b] Baeten et al. Decidability of bisimulation equivalence for process generating context-free languages. 1993.

↓

Prune unnormed productions

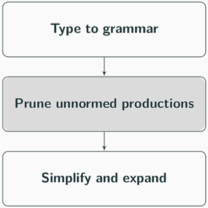
↓

Simplify and expand

Prune unnormed productions

Definition ((Un)normed terms[1])

A term α is *normed* if there is a path from α to ε. Otherwise, it is said to be *unnormed*.



Christensen, Huttel, and Stirling[1] noted that:

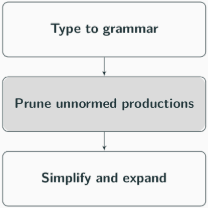
$$\text{whenever } \alpha \text{ is unnormed, } \alpha \sim \alpha\beta.$$

[1] Christensen et al. Bisimulation equivalence is decidable for all CF processes. 1995

Prune unnormed productions

Definition ((Un)normed terms[1])

A term α is *normed* if there is a path from α to ε. Otherwise, it is said to be *unnormed*.



Christensen, Huttel, and Stirling[1] noted that:

$$\text{whenever } \alpha \text{ is unnormed, } \alpha \sim \alpha\beta.$$

Example

$$S \triangleq (\mu x.\&\{n\colon x; x; ?\,\mathsf{int}, \ell\colon ?\,\mathsf{int}\}); (\mu z.!\,\mathsf{int}; z; z)$$

Productions for S
$X_1 \xrightarrow{\&n} X_1X_1X_2$
$X_1 \xrightarrow{\&\ell} X_3$
$X_2 \xrightarrow{?\,\mathsf{int}} \varepsilon$
$X_3 \xrightarrow{?\,\mathsf{int}} \varepsilon$
$X_4 \xrightarrow{!\,\mathsf{int}} X_4X_4$

[1] Christensen et al. Bisimulation equivalence is decidable for all CF processes. 1995

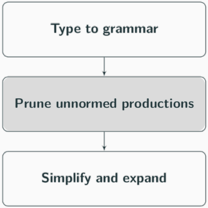
Prune unnormed productions

Definition ((Un)normed terms[1])

A term α is *normed* if there is a path from α to ε. Otherwise, it is said to be *unnormed*.

Christensen, Huttel, and Stirling[1] noted that:

whenever α is unnormed, $\alpha \sim \alpha\beta$.



Example

$$S \triangleq (\mu x.\&\{n\colon x; x; ?\,\mathsf{int}, \ell\colon ?\,\mathsf{int}\}); (\mu z.!\,\mathsf{int}; z; z)$$

Productions for S	Productions for *pruned* S
$X_1 \xrightarrow{\&n} X_1X_1X_2$	$X_1 \xrightarrow{\&n} X_1X_1X_2$
$X_1 \xrightarrow{\&\ell} X_3$	$X_1 \xrightarrow{\&\ell} X_3$
$X_2 \xrightarrow{?\,\mathsf{int}} \varepsilon$	$X_2 \xrightarrow{?\,\mathsf{int}} \varepsilon$
$X_3 \xrightarrow{?\,\mathsf{int}} \varepsilon$	$X_3 \xrightarrow{?\,\mathsf{int}} \varepsilon$
$X_4 \xrightarrow{!\,\mathsf{int}} X_4X_4$	$X_4 \xrightarrow{!\,\mathsf{int}} X_4$

[1] Christensen et al. Bisimulation equivalence is decidable for all CF processes. 1995

Convert types to a grammar

`LOHaskellCode` $\sim$ 100

Soundness ✓

Prune unnormed productions

`LOHaskellCode` $\sim$ 30

Soundness ✓

1. Pruning unnormed productions is sound[a]

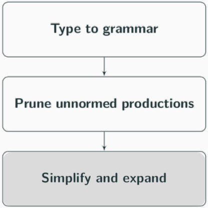
[a] Christensen, Huttel, and Stirling. Bisimulation equivalence is decidable for all context-free processes. 1995

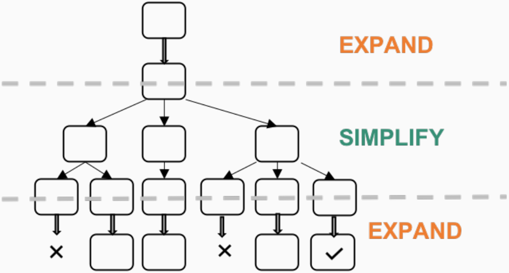
Simplify and expand

Simplify and expand

Following the ideas from Hirshfeld, Jančar, Moller, a bisimulation between processes is seen as an **expansion tree**[2,3], that alternates between:

- expansion operations
- simplification operations





[2] Hirshfeld. Bisimulation trees and the decidability of weak bisimulations. 1997

[3] Jančar and Moller. Techniques for decidability and undecidability of bisimilarity. 1999

Simplify and expand

An expansion tree alternates between:

- **expansion operations** - a single derived node results from the expansion of the parent node.

[4] Jančar, Moller. Techniques for decidability and undecidability of bisimilarity. 1999
[5] Christensen et al. Bisimulation equivalence is decidable for all CF processes. 1995

Simplify and expand

An expansion tree alternates between:

- **expansion operations** - a single derived node results from the expansion of the parent node.
- **simplification operations**
  - **reflexive rule**: omit from a node N any reflexive pair.
  - **congruence rule**: omit from a node N any pair that belongs to the least congruence containing the ancestors of N.

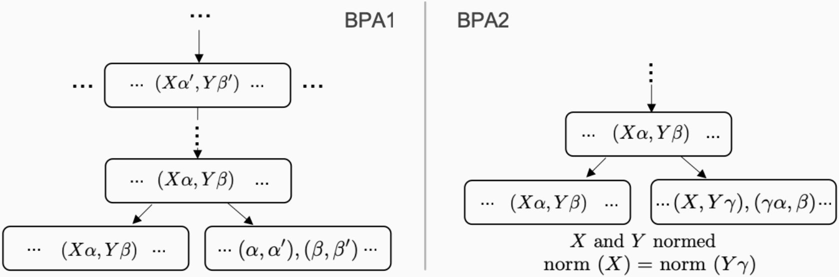
[4] Jančar, Moller. Techniques for decidability and undecidability of bisimilarity. 1999

[5] Christensen et al. Bisimulation equivalence is decidable for all CF processes. 1995

Simplify and expand

An expansion tree alternates between:

- **expansion operations** - a single derived node results from the expansion of the parent node.
- **simplification operations**
  - **reflexive rule**: omit from a node N any reflexive pair.
  - **congruence rule**: omit from a node N any pair that belongs to the least congruence containing the ancestors of N.
  - **basic process algebra rules**[4][5]



[4] Jančar, Moller. Techniques for decidability and undecidability of bisimilarity. 1999

[5] Christensen et al. Bisimulation equivalence is decidable for all CF processes. 1995

Simplify and Expand

All these transformation rules preserve the *safeness property*:

Safeness property[5]

$S \sim T$ iff the expansion tree rooted at $\{(S, T)\}$ has a successful branch.

The finite witness property holds[5]:

Finite witness property[5,6]

If $S \sim T$, then there exists a **finite successful branch** in the expansion tree.

[6] Christensen et al. Bisimulation equivalence is decidable for all CF processes. 1995

Example

$S \triangleq (\mu x.\&\{n\colon x; x; ?\,\mathsf{int}, \ell\colon ?\,\mathsf{int}\}); (\mu z.!\,\mathsf{int}; z; z)$

$T \triangleq (\mu y.\&\{n\colon y; y, \ell\colon \mathsf{skip}\}; ?\,\mathsf{int}); (\mu z.!\,\mathsf{int}; z)$

$\boxed{(X_1X_4, Y_1Y_3)}$

Productions for *pruned* S

$X_1 \xrightarrow{\&n} X_1X_1X_2$

$X_1 \xrightarrow{\&\ell} X_3$

$X_2 \xrightarrow{?\,\mathsf{int}} \varepsilon$

$X_3 \xrightarrow{?\,\mathsf{int}} \varepsilon$

$X_4 \xrightarrow{!\,\mathsf{int}} X_4$

Productions for T

$Y_1 \xrightarrow{\&n} Y_1Y_1Y_2$

$Y_1 \xrightarrow{\&\ell} Y_2$

$Y_2 \xrightarrow{?\mathsf{int}} \varepsilon$

$Y_3 \xrightarrow{!\mathsf{int}} Y_3$

Example

$$S \triangleq (\mu x.\&\{n\colon x; x; ?\,\mathsf{int}, \ell\colon ?\,\mathsf{int}\}); (\mu z.!\,\mathsf{int}; z; z)$$
$$T \triangleq (\mu y.\&\{n\colon y; y, \ell\colon \mathsf{skip}\}; ?\,\mathsf{int}); (\mu z.!\,\mathsf{int}; z)$$

$$\begin{tikzcd}
 & \boxed{(X_1X_4, Y_1Y_3)} \arrow[dl, "\text{bpa2}"'] \arrow[dr, "\text{self}"] & \\
\boxed{(X_1, Y_1), (X_4, Y_3)} & & \boxed{(X_1X_4, Y_1Y_3)}
\end{tikzcd}$$

Productions for *pruned* S

$$X_1 \xrightarrow{\&n} X_1X_1X_2$$
$$X_1 \xrightarrow{\&\ell} X_3$$
$$X_2 \xrightarrow{?\,\mathsf{int}} \varepsilon$$
$$X_3 \xrightarrow{?\,\mathsf{int}} \varepsilon$$
$$X_4 \xrightarrow{!\,\mathsf{int}} X_4$$

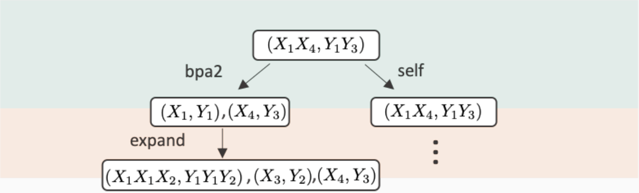
Productions for T

$$Y_1 \xrightarrow{\&n} Y_1Y_1Y_2$$
$$Y_1 \xrightarrow{\&\ell} Y_2$$
$$Y_2 \xrightarrow{?\mathsf{int}} \varepsilon$$
$$Y_3 \xrightarrow{!\mathsf{int}} Y_3$$

Example

$$S \triangleq (\mu x.\&\{n\colon x; x; ?\,\mathsf{int}, \ell\colon ?\,\mathsf{int}\}); (\mu z.!\,\mathsf{int}; z; z)$$

$$T \triangleq (\mu y.\&\{n\colon y; y, \ell\colon \mathsf{skip}\}; ?\,\mathsf{int}); (\mu z.!\,\mathsf{int}; z)$$



Productions for *pruned* S

$$X_1 \xrightarrow{\&n} X_1X_1X_2$$
$$X_1 \xrightarrow{\&\ell} X_3$$
$$X_2 \xrightarrow{?\,\mathsf{int}} \varepsilon$$
$$X_3 \xrightarrow{?\,\mathsf{int}} \varepsilon$$
$$X_4 \xrightarrow{!\,\mathsf{int}} X_4$$

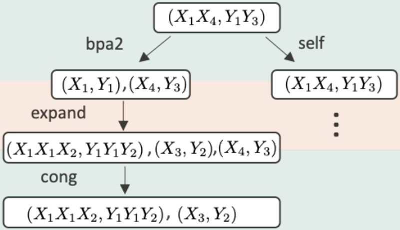
Productions for T

$$Y_1 \xrightarrow{\&n} Y_1Y_1Y_2$$
$$Y_1 \xrightarrow{\&\ell} Y_2$$
$$Y_2 \xrightarrow{?\mathsf{int}} \varepsilon$$
$$Y_3 \xrightarrow{!\mathsf{int}} Y_3$$

Example

$$S \triangleq (\mu x.\&\{n\colon x; x; ?\,\mathsf{int}, \ell\colon ?\,\mathsf{int}\}); (\mu z.!\,\mathsf{int}; z; z)$$

$$T \triangleq (\mu y.\&\{n\colon y; y, \ell\colon \mathsf{skip}\}; ?\,\mathsf{int}); (\mu z.!\,\mathsf{int}; z)$$



Productions for *pruned* S

$$X_1 \xrightarrow{\&n} X_1X_1X_2$$
$$X_1 \xrightarrow{\&\ell} X_3$$
$$X_2 \xrightarrow{?\,\mathsf{int}} \varepsilon$$
$$X_3 \xrightarrow{?\,\mathsf{int}} \varepsilon$$
$$X_4 \xrightarrow{!\,\mathsf{int}} X_4$$

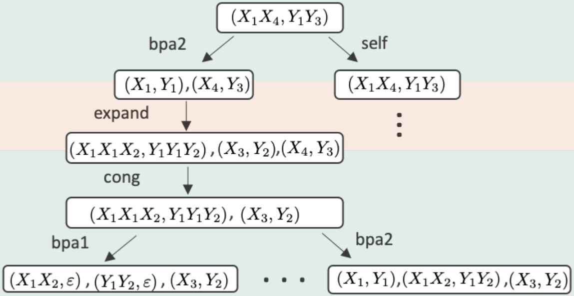
Productions for T

$$Y_1 \xrightarrow{\&n} Y_1Y_1Y_2$$
$$Y_1 \xrightarrow{\&\ell} Y_2$$
$$Y_2 \xrightarrow{?\mathsf{int}} \varepsilon$$
$$Y_3 \xrightarrow{!\mathsf{int}} Y_3$$

Example

$$S \triangleq (\mu x.\&\{n\colon x; x; ?\,\mathsf{int}, \ell\colon ?\,\mathsf{int}\}); (\mu z.!\,\mathsf{int}; z; z)$$

$$T \triangleq (\mu y.\&\{n\colon y; y, \ell\colon \mathsf{skip}\}; ?\,\mathsf{int}); (\mu z.!\,\mathsf{int}; z)$$



Productions for *pruned* S

$$X_1 \xrightarrow{\&n} X_1X_1X_2$$
$$X_1 \xrightarrow{\&\ell} X_3$$
$$X_2 \xrightarrow{?\,\mathsf{int}} \varepsilon$$
$$X_3 \xrightarrow{?\,\mathsf{int}} \varepsilon$$
$$X_4 \xrightarrow{!\,\mathsf{int}} X_4$$

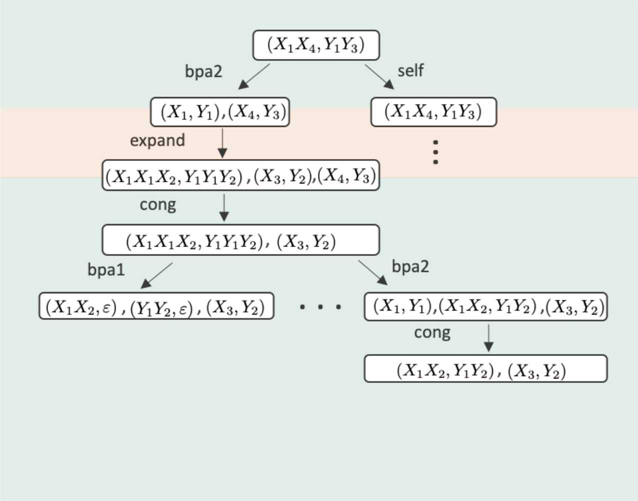
Productions for T

$$Y_1 \xrightarrow{\&n} Y_1Y_1Y_2$$
$$Y_1 \xrightarrow{\&\ell} Y_2$$
$$Y_2 \xrightarrow{?\mathsf{int}} \varepsilon$$
$$Y_3 \xrightarrow{!\mathsf{int}} Y_3$$

Example

$$S \triangleq (\mu x.\&\{n\colon x; x; ?\,\mathsf{int}, \ell\colon ?\,\mathsf{int}\}); (\mu z.!\,\mathsf{int}; z; z)$$

$$T \triangleq (\mu y.\&\{n\colon y; y, \ell\colon \mathsf{skip}\}; ?\,\mathsf{int}); (\mu z.!\,\mathsf{int}; z)$$



Productions for *pruned* S

$X_1 \xrightarrow{\&n} X_1X_1X_2$

$X_1 \xrightarrow{\&\ell} X_3$

$X_2 \xrightarrow{?\,\mathsf{int}} \varepsilon$

$X_3 \xrightarrow{?\,\mathsf{int}} \varepsilon$

$X_4 \xrightarrow{!\,\mathsf{int}} X_4$

Productions for T

$Y_1 \xrightarrow{\&n} Y_1Y_1Y_2$

$Y_1 \xrightarrow{\&\ell} Y_2$

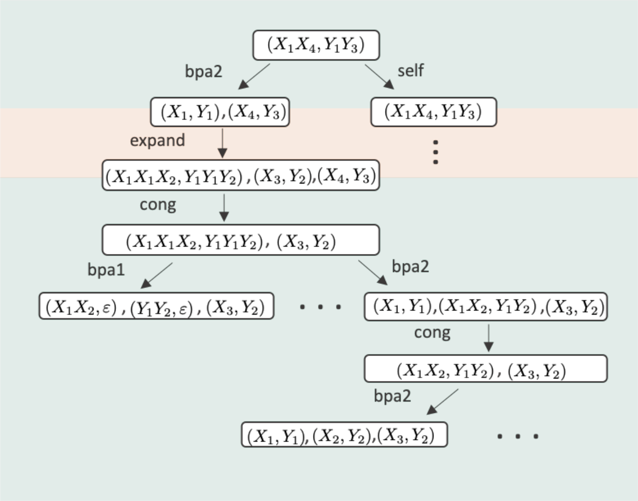
$Y_2 \xrightarrow{?\mathsf{int}} \varepsilon$

$Y_3 \xrightarrow{!\mathsf{int}} Y_3$

Example

$$S \triangleq (\mu x.\&\{n\colon x; x; ?\,\mathsf{int}, \ell\colon ?\,\mathsf{int}\}); (\mu z.!\,\mathsf{int}; z; z)$$

$$T \triangleq (\mu y.\&\{n\colon y; y, \ell\colon \mathsf{skip}\}; ?\,\mathsf{int}); (\mu z.!\,\mathsf{int}; z)$$



Productions for *pruned* S

$X_1 \xrightarrow{\&n} X_1X_1X_2$

$X_1 \xrightarrow{\&\ell} X_3$

$X_2 \xrightarrow{?\,\mathsf{int}} \varepsilon$

$X_3 \xrightarrow{?\,\mathsf{int}} \varepsilon$

$X_4 \xrightarrow{!\,\mathsf{int}} X_4$

Productions for T

$Y_1 \xrightarrow{\&n} Y_1Y_1Y_2$

$Y_1 \xrightarrow{\&\ell} Y_2$

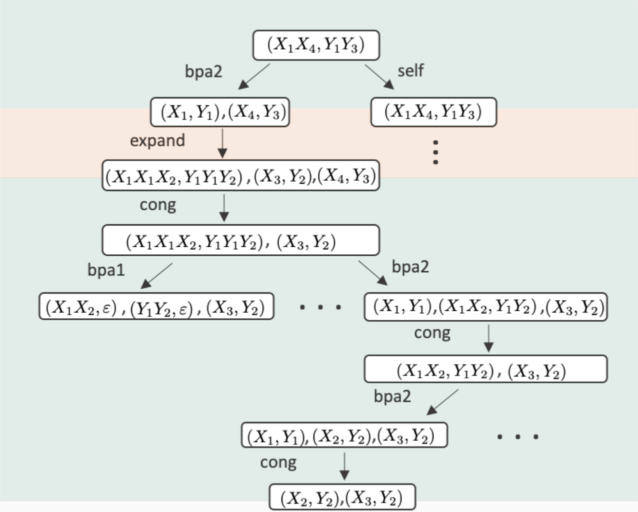
$Y_2 \xrightarrow{?\mathsf{int}} \varepsilon$

$Y_3 \xrightarrow{!\mathsf{int}} Y_3$

Example

$$S \triangleq (\mu x.\&\{n\colon x; x; ?\,\mathsf{int}, \ell\colon ?\,\mathsf{int}\}); (\mu z.!\,\mathsf{int}; z; z)$$

$$T \triangleq (\mu y.\&\{n\colon y; y, \ell\colon \mathsf{skip}\}; ?\,\mathsf{int}); (\mu z.!\,\mathsf{int}; z)$$



Productions for *pruned* S

$$X_1 \xrightarrow{\&n} X_1X_1X_2$$
$$X_1 \xrightarrow{\&\ell} X_3$$
$$X_2 \xrightarrow{?\,\mathsf{int}} \varepsilon$$
$$X_3 \xrightarrow{?\,\mathsf{int}} \varepsilon$$
$$X_4 \xrightarrow{!\,\mathsf{int}} X_4$$

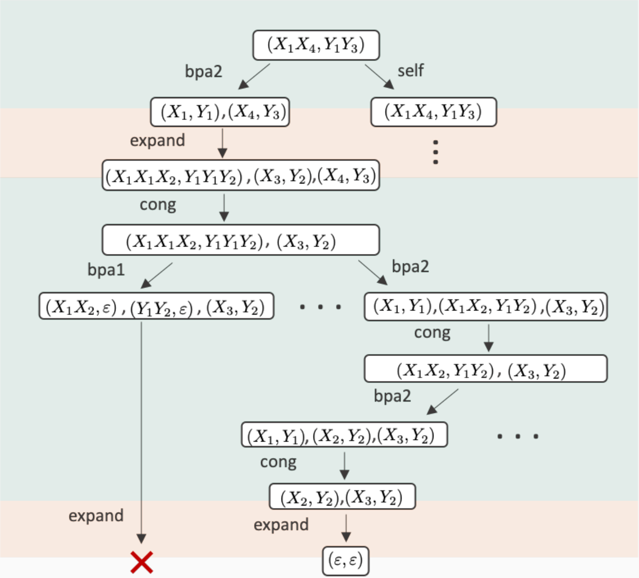
Productions for T

$$Y_1 \xrightarrow{\&n} Y_1Y_1Y_2$$
$$Y_1 \xrightarrow{\&\ell} Y_2$$
$$Y_2 \xrightarrow{?\mathsf{int}} \varepsilon$$
$$Y_3 \xrightarrow{!\mathsf{int}} Y_3$$

Example

$$S \triangleq (\mu x.\&\{n\colon x; x; ?\,\mathsf{int}, \ell\colon ?\,\mathsf{int}\}); (\mu z.!\,\mathsf{int}; z; z)$$

$$T \triangleq (\mu y.\&\{n\colon y; y, \ell\colon \mathsf{skip}\}; ?\,\mathsf{int}); (\mu z.!\,\mathsf{int}; z)$$



Productions for *pruned* S

$$X_1 \xrightarrow{\&n} X_1X_1X_2$$
$$X_1 \xrightarrow{\&\ell} X_3$$
$$X_2 \xrightarrow{?\,\mathsf{int}} \varepsilon$$
$$X_3 \xrightarrow{?\,\mathsf{int}} \varepsilon$$
$$X_4 \xrightarrow{!\,\mathsf{int}} X_4$$

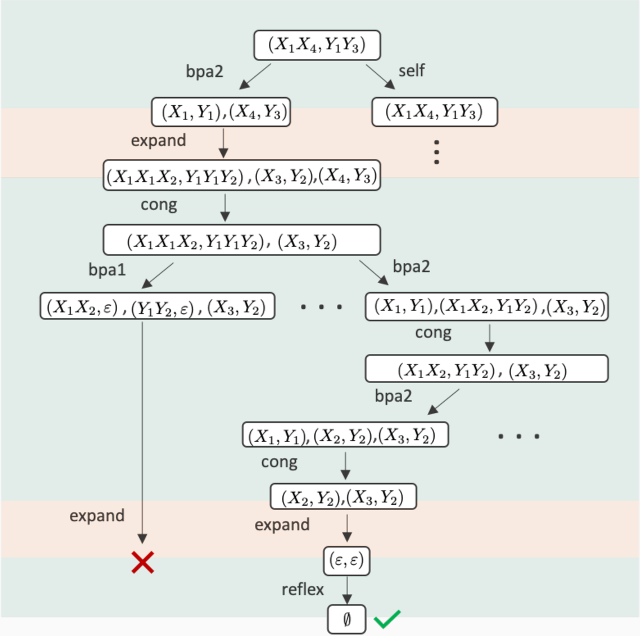
Productions for T

$$Y_1 \xrightarrow{\&n} Y_1Y_1Y_2$$
$$Y_1 \xrightarrow{\&\ell} Y_2$$
$$Y_2 \xrightarrow{?\mathsf{int}} \varepsilon$$
$$Y_3 \xrightarrow{!\mathsf{int}} Y_3$$

Example

$$S \triangleq (\mu x.\&\{n\colon x; x; ?\,\mathsf{int}, \ell\colon ?\,\mathsf{int}\}); (\mu z.!\,\mathsf{int}; z; z)$$

$$T \triangleq (\mu y.\&\{n\colon y; y, \ell\colon \mathsf{skip}\}; ?\,\mathsf{int}); (\mu z.!\,\mathsf{int}; z)$$



Productions for *pruned* S

$X_1 \xrightarrow{\&n} X_1X_1X_2$

$X_1 \xrightarrow{\&\ell} X_3$

$X_2 \xrightarrow{?\,\mathsf{int}} \varepsilon$

$X_3 \xrightarrow{?\,\mathsf{int}} \varepsilon$

$X_4 \xrightarrow{!\,\mathsf{int}} X_4$

Productions for T

$Y_1 \xrightarrow{\&n} Y_1Y_1Y_2$

$Y_1 \xrightarrow{\&\ell} Y_2$

$Y_2 \xrightarrow{?\mathsf{int}} \varepsilon$

$Y_3 \xrightarrow{!\mathsf{int}} Y_3$

Algorithm for checking the equivalence of CFST Main stages

Convert types to a grammar

`LOHaskellCode` ∼ 100

Soundness ✓

↓

Prune unnormed productions

`LOHaskellCode` ∼ 30

Soundness ✓

↓

Simplify and expand

`LOHaskellCode` ∼ 170

Soundness ✓

1. The *safeness property ensures soundness*[a]

[a] Jančar, Moller. Techniques for decidability and undecidability of bisimilarity. 1999

Implementation strategies

Implementation choice:

- Breadth-first search on the tree

Implementation strategies

Implementation choice:

- Breadth-first search on the tree

Strategic options that can enhance performance:

- Instead of looking for a fixed point, iterate the simplification phase
- Apply BPA rules wrapped with blocks of reflexive and congruence rules

Implementation strategies

Implementation choice:

- Breadth-first search on the tree

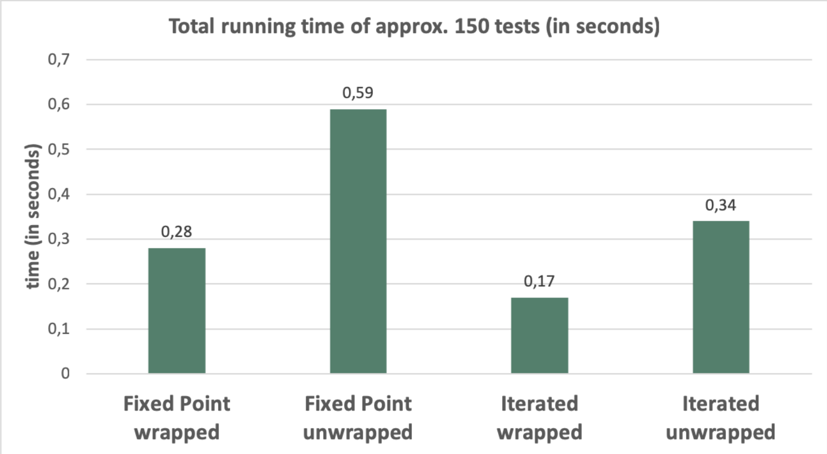
Strategic options that can enhance performance:

- Instead of looking for a fixed point, iterate the simplification phase
- Apply BPA rules wrapped with blocks of reflexive and congruence rules

We showcase runtimes for four scenarios:

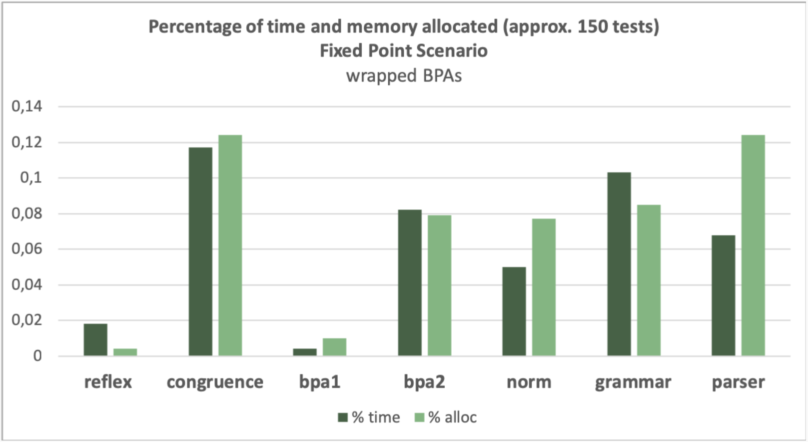
	Fixed Point Scenario		**Iterated Scenario**	
	wrapped	unwrapped	wrapped	unwrapped
Simplification	find fixed point		iterate	
BPAs wrapped	✓	×	✓	×

Running times ($\sim$ 150 tests)

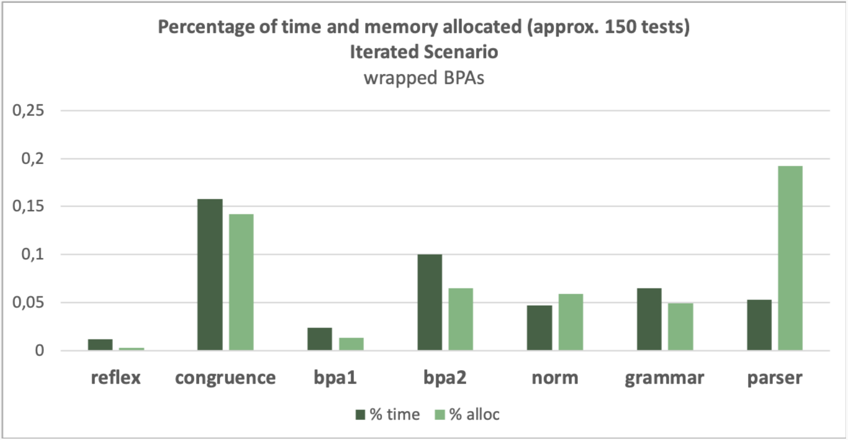


Specs:	MacBook Air
	1.8 GHz Intel Core i5
	8GB of memory

Time and memory allocated



Time and memory allocated



Towards completeness...

Finite witness property[7][8]

If $S \sim T$, then there exists a **finite successful branch** in the expansion tree.

Implementation choices aiming to achieve completeness:

- Instead of looking for a fixed point, iterate the simplification phase (there may not be a fixed point)
- Double ended enqueue, prepending *promising* nodes, as opposed to queuing all new nodes

[7] Christensen et al. Bisimulation equivalence is decidable for all CF processes. 1995
[8] Jančar, Moller. Techniques for decidability and undecidability of bisimilarity. 1999

Ongoing Work

- Soundness ✓
- Completeness ? ... on our way to achieve it.
- Complexity ? ... in practice seems to take not much than parsing!
- Lines of Haskell code: approx.\ 300

Coming soon:

FreeST, a compiler for context-free session types!

(demos on demand)

Thank you!