Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Queen Mary
University of London

LERO Software for a better world

Fondúireacht Eolaíochta Éireann sfi Dá bhfuil romhainn

Pushdown Normal-Form Bisimulation

A Nominal Context-Free Approach to Program Equivalence

Yu-Yang Lin — **Trinity College Dublin**
Vasileios Koutavas — **Trinity College Dublin & Lero**
Nikos Tzevelekos — **Queen Mary University of London**

Research supported in part by Science Foundation Ireland grant 13/RC/2094_2 (Lero), and Cisco/Silicon Valley Community Fund.

Overview

Contextual Equivalence: Are terms M, N equivalent in every program context C?

- Long history of work in PL semantics proof techniques
- Recently, **automated verification** of contextual equivalence became possible
  - e.g. our recent work [TACAS'22, LICS'23]:
    implementable verification using **NF Bisimulation/Game Semantics** and **Up-To Techniques**
  - **Problem: undecidable** verification due to **unbounded call stacks**

This Paper: decidable verification for a class of programs with **unbounded call stacks**

- Inspiration from pushdown systems:
  Replace **unbounded call stacks** with **loops in a continuation graph**

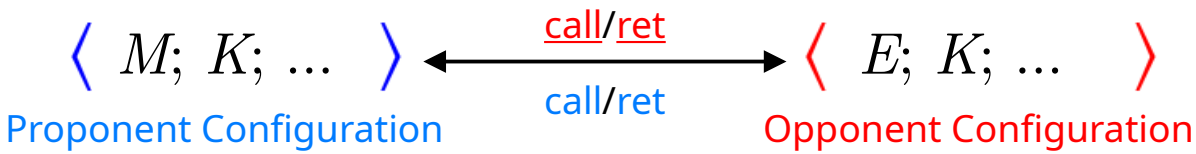
Setting: Higher-Order Stateful Programs

ML-like programming language with local state:

- **Higher-Order and Stateful:**
  - passing functions as values
  - local references

- **A problem: need to model unknown (external) code:**
  - ➔ NF Bisimulation / Game Semantics [TACAS'22]

Labelled Transition System



M: proponent term
K: call stack of continuations E

Moves that affect the stack K:

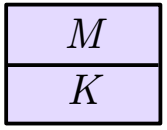
- **Opponent Call:** **PUSH** continuations on the stack
- **Proponent Return:** **POP** continuations from the stack

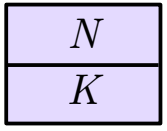
Reentrant Calls

$M =$
```
let x = ref 0 in
fun f -> f(); !x
```
$\cong$ $N =$
```
fun f -> f(); 0
```

(simplified example based on event-handler code)

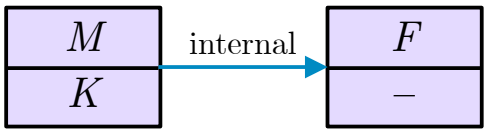
Reentrant Calls



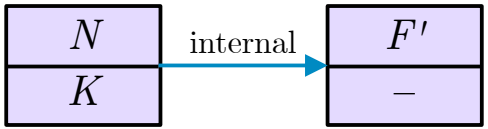


Reentrant Calls

F `= fun` α `->` α `() ; !`x

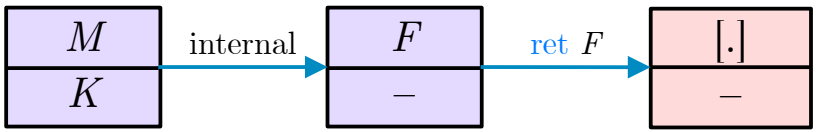


F' `= fun` α `->` α `() ; 0`

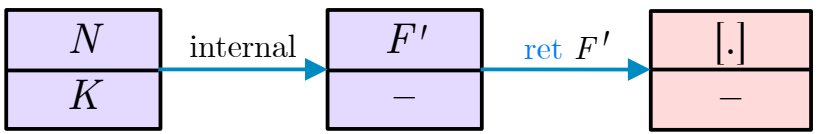


Reentrant Calls

F `= fun α -> α () ; !`x

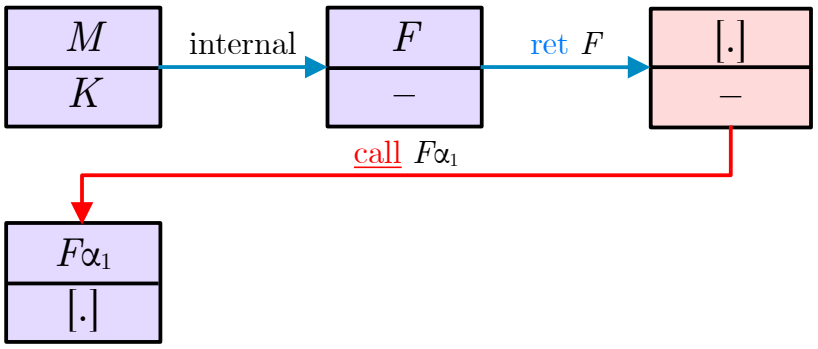


F' `= fun α -> α () ; 0`

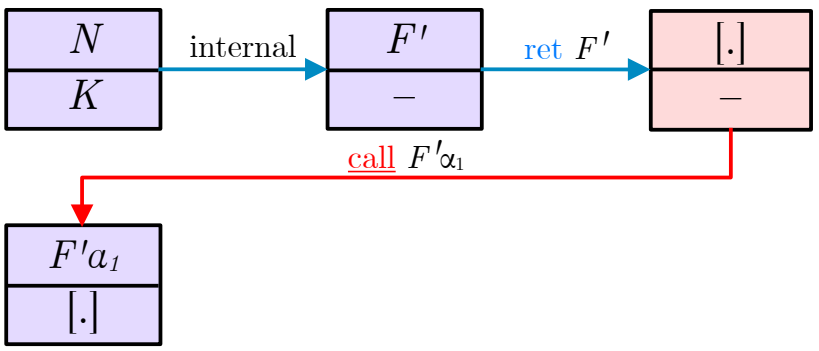


Reentrant Calls

F `= fun α -> α () ; !`x

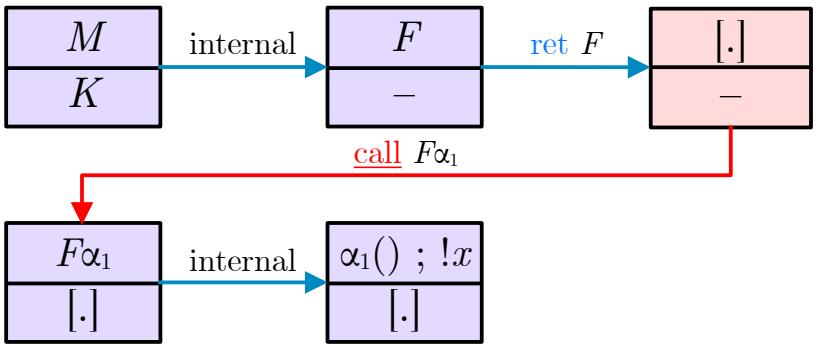


F' `= fun α -> α () ; 0`

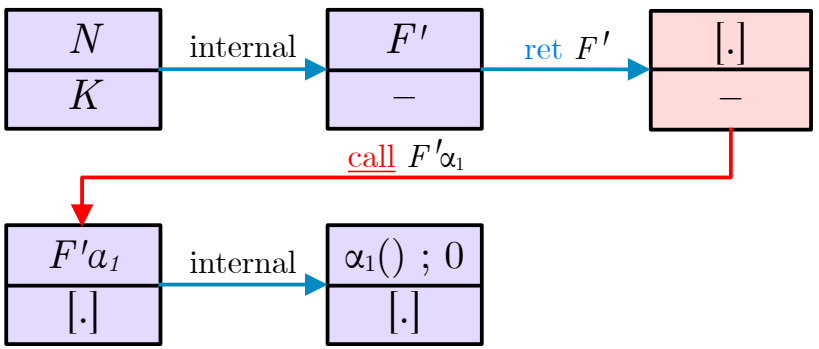


Reentrant Calls

F `= fun α -> α () ; !`x

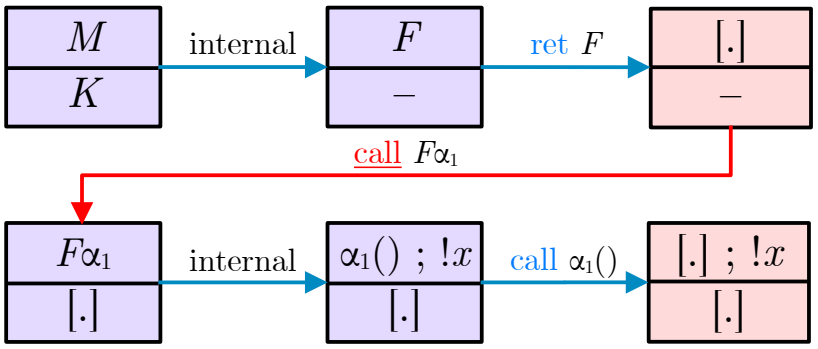


F' `= fun α -> α () ; 0`

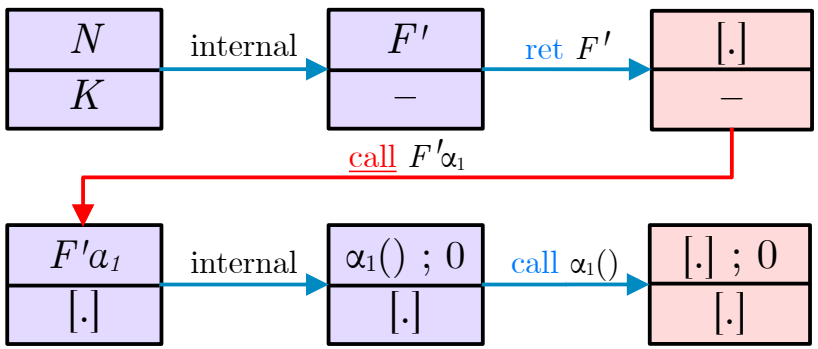


Reentrant Calls

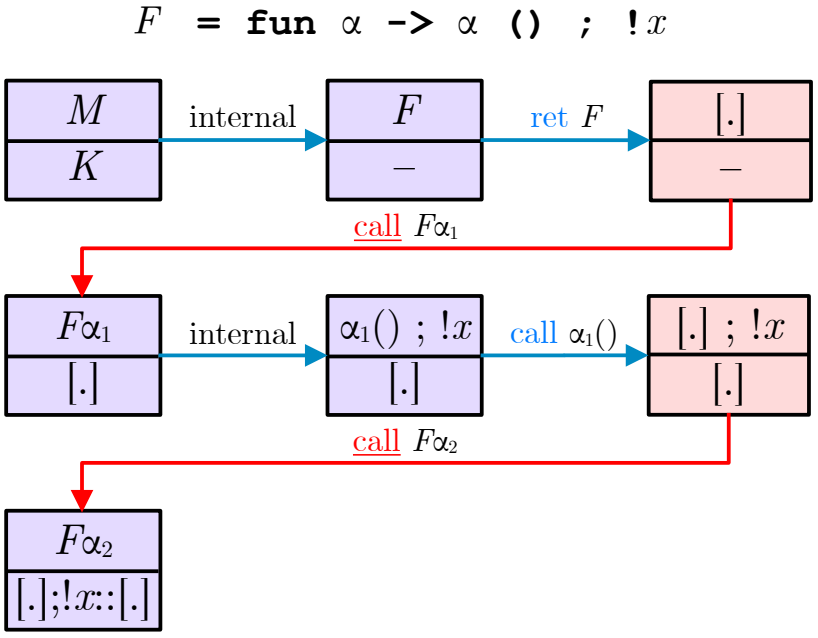
F = **fun** α -> α () ; !x

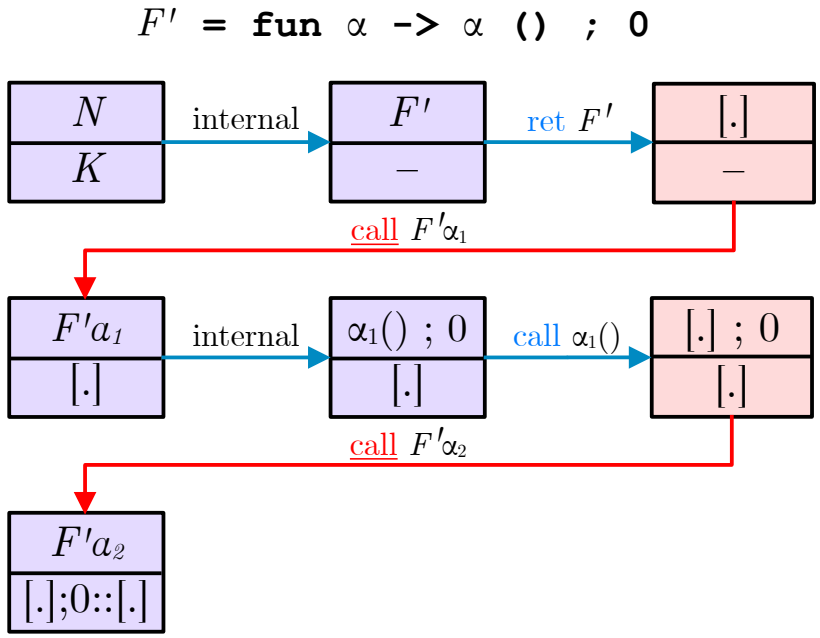


F' = **fun** α -> α () ; 0

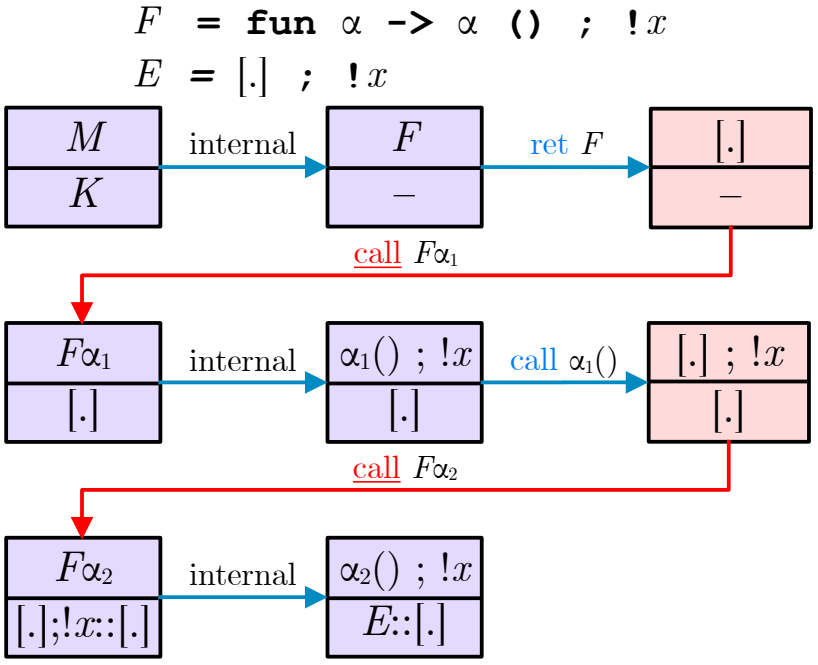


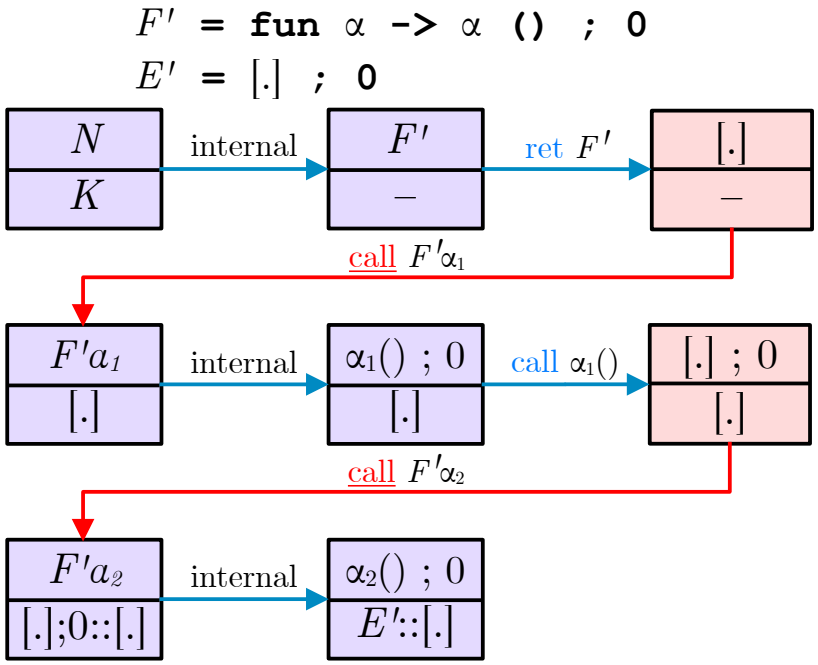
Reentrant Calls



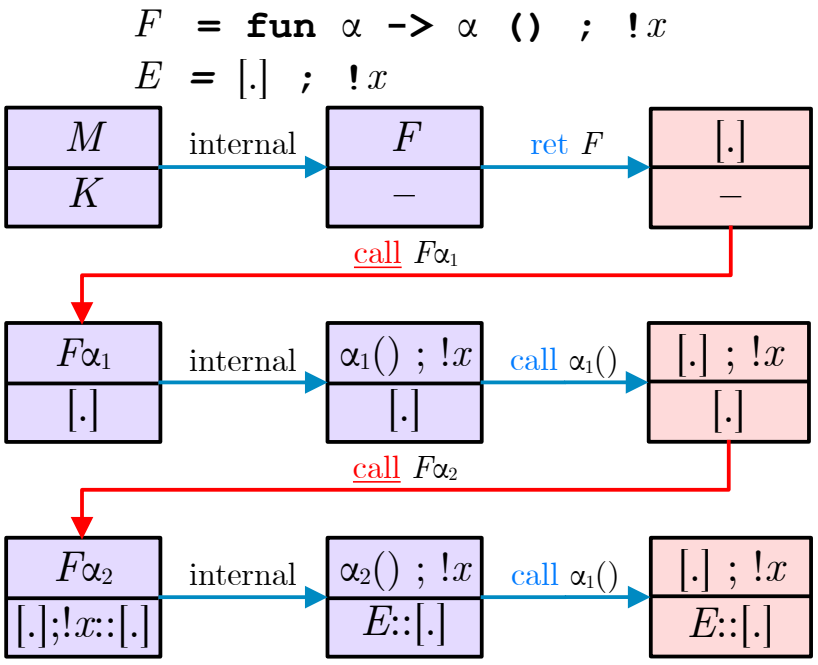


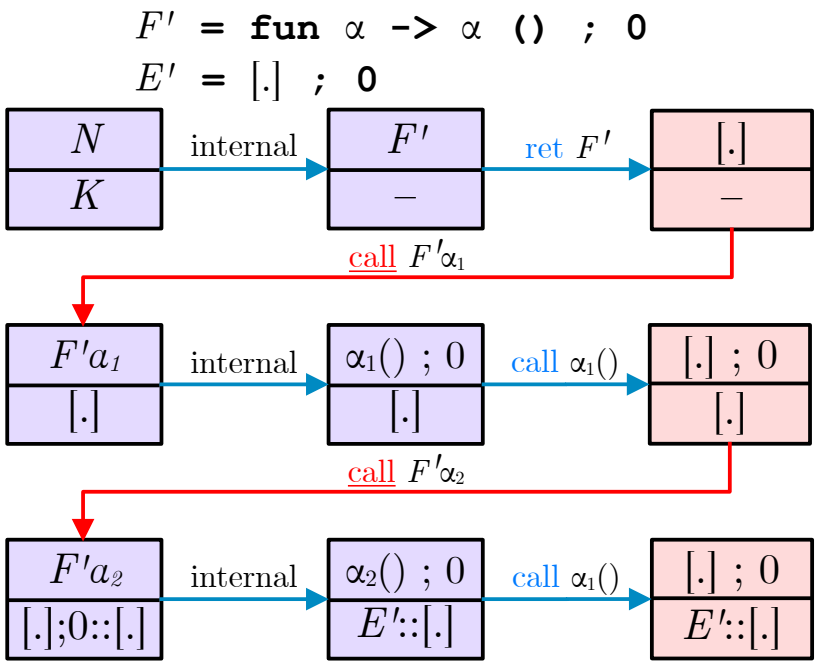
Reentrant Calls



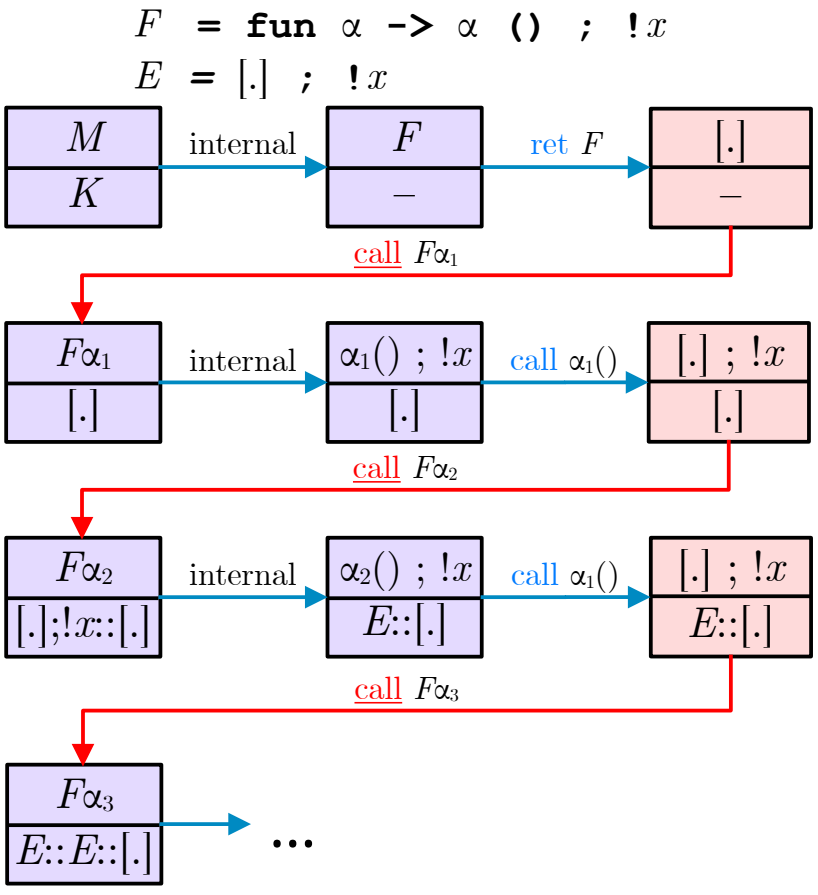


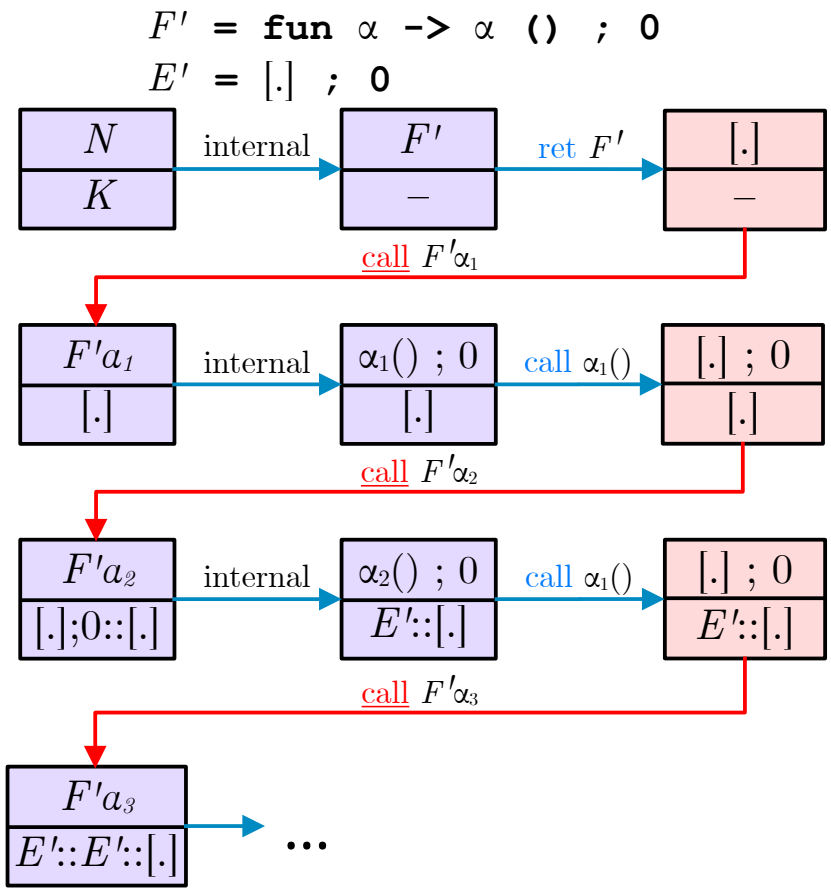
Reentrant Calls





Reentrant Calls





(Stacked) NF Bisimulation

$M \cong N$ *iff they produce the same complete traces (all calls answered)*

- we run them in parallel and see if they produce the same LTS

$$(C_1, C_2, K) \qquad \text{where } K = C_1.K_1 \ , \ C_2.K_2$$

Bisimulation Game Configurations

Can we capture reachability without losing precision?

Pushdown NF Bisimulation

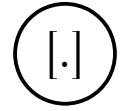
Continuation Graph (Σ):

- **Vertices (β):** proponent function *entry points* $\beta = \ulcorner C_1, C_2 \urcorner$
  - created on **PUSH** transitions (O-Call)
- **Edges ($\beta' \xrightarrow{\mathcal{E}_1, \mathcal{E}_2} \beta$):** directed and labelled with continuations
  - traversed on **POP** transitions (P-Ret)

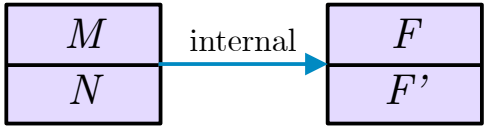
$$(C_1, C_2, \Sigma, \beta)$$

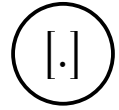
Pushdown Bisimulation Game Configurations

Reentrant Example Revisited

$\Sigma =$ 

Reentrant Example Revisited

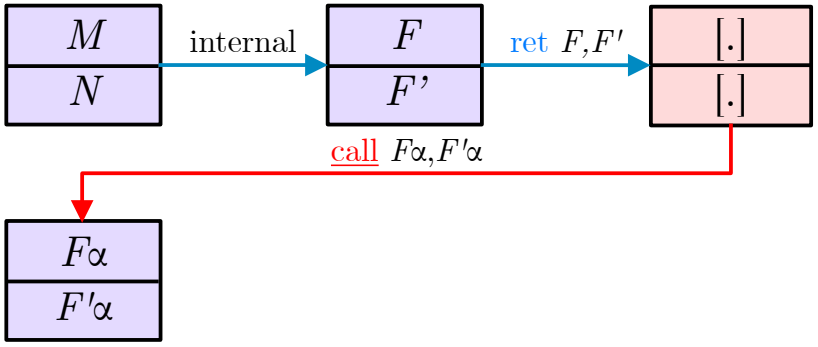


$\Sigma =$ 

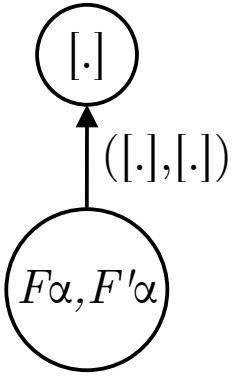
Reentrant Example Revisited

$\Sigma =$ [.]

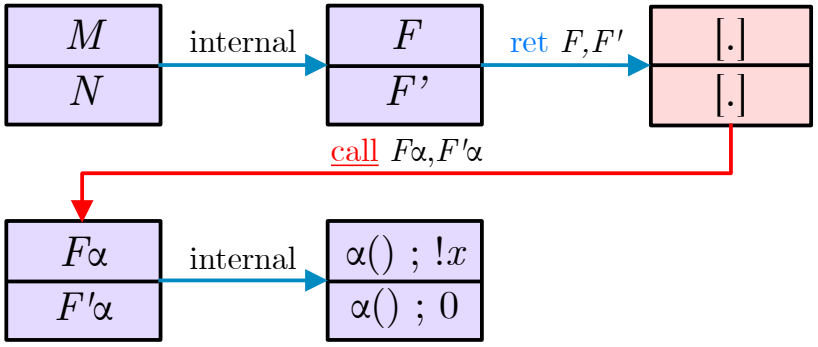
Reentrant Example Revisited



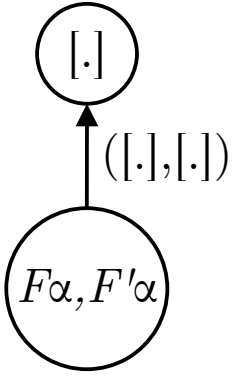
$\Sigma =$



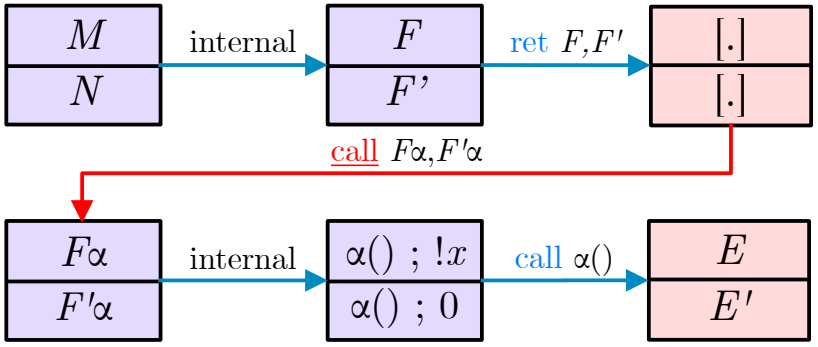
Reentrant Example Revisited



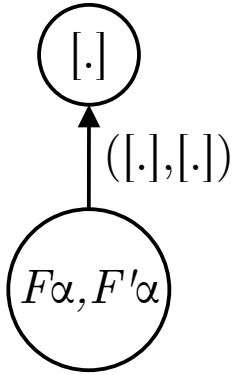
$\Sigma =$



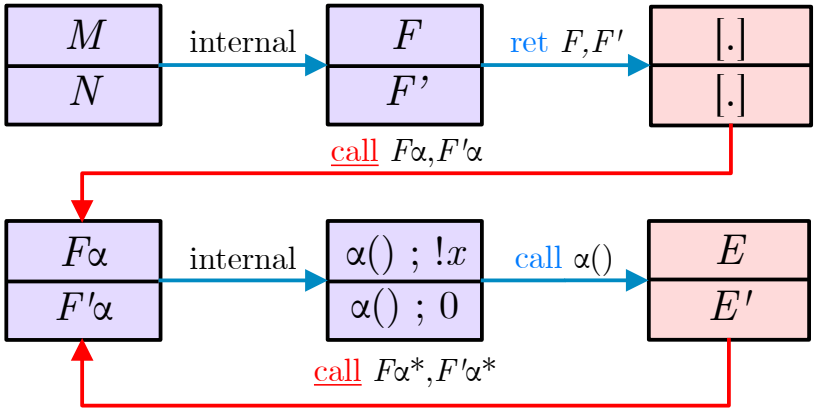
Reentrant Example Revisited



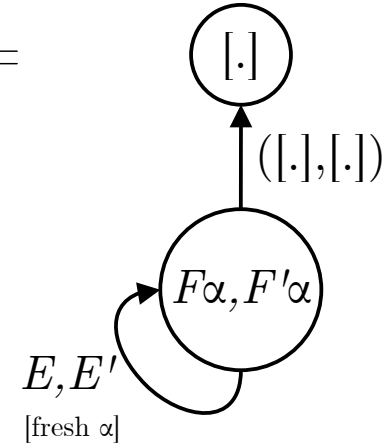
$\Sigma =$



Reentrant Example Revisited



$\Sigma =$



Theoretical Results

Pushdown NF Bisimilarity is **Sound** and **Complete** (Fully Abstract) wrt Contextual Equivalence

- There are spurious paths, but they only go through real reachable states

Decidability (Dynamic Property)**:**

$M \cong N$ is decidable if M, N reach a finite* number of configurations in the *stackless* LTS

(*): up-to permutation of names

Implementation

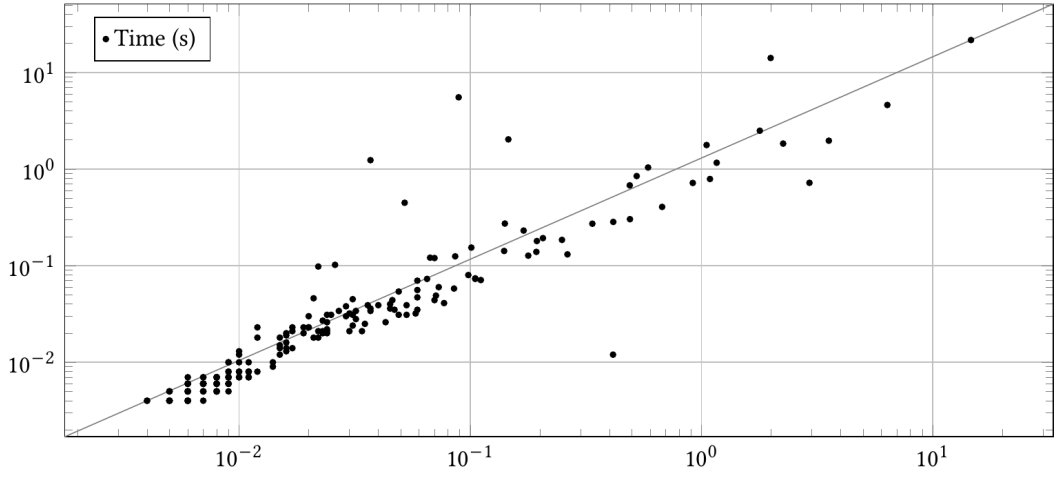
Hobbit-PDNF: https://github.com/LaifsV1/Hobbit-PDNF

DOI: 10.5281/zenodo.11128050

Two Test Suites:

1) Hobbit's original test suite
2) Instances of well-bracketed state
   - Based on Event Handlers and a handoff protocol

		(1)		(2)	
		PDNF	Hobbit	PDNF	Hobbit
Eq.	Proven	72	62	11	0
Ineq.	Proven	77	78	N/A	N/A



(X) Hobbit vs. (Y) Hobbit-PDNF over Hobbit's test suite

Conclusion

Novel Technique:

- Sound and complete with respect to contextual equivalence
- Decidable for a class of programs with unbounded call stack
- Abstracts away stacks without losing precision

Experimental Evaluation:

- Proves strictly more equivalences than previous work
- Mostly no or little overhead
- More complex LTS may cause some inequivalences to take longer