Analysis for Trustworthy Software

David Van Horn

With support from
NSF, DARPA, CRA, & Google

DARPA

Example Apps (1:2)

★ **Invented in theater**

Works with network

- **PLI** – Blue-force tracking
- **DASH** – Messages
- **RTC** – Chat rooms
- **Collab** – Live Collaboration
- **SAR** – Coordinated Search & Rescue
- **STREAM** – Sensor streaming
- **SpeedTest** – Network performance

With or without network

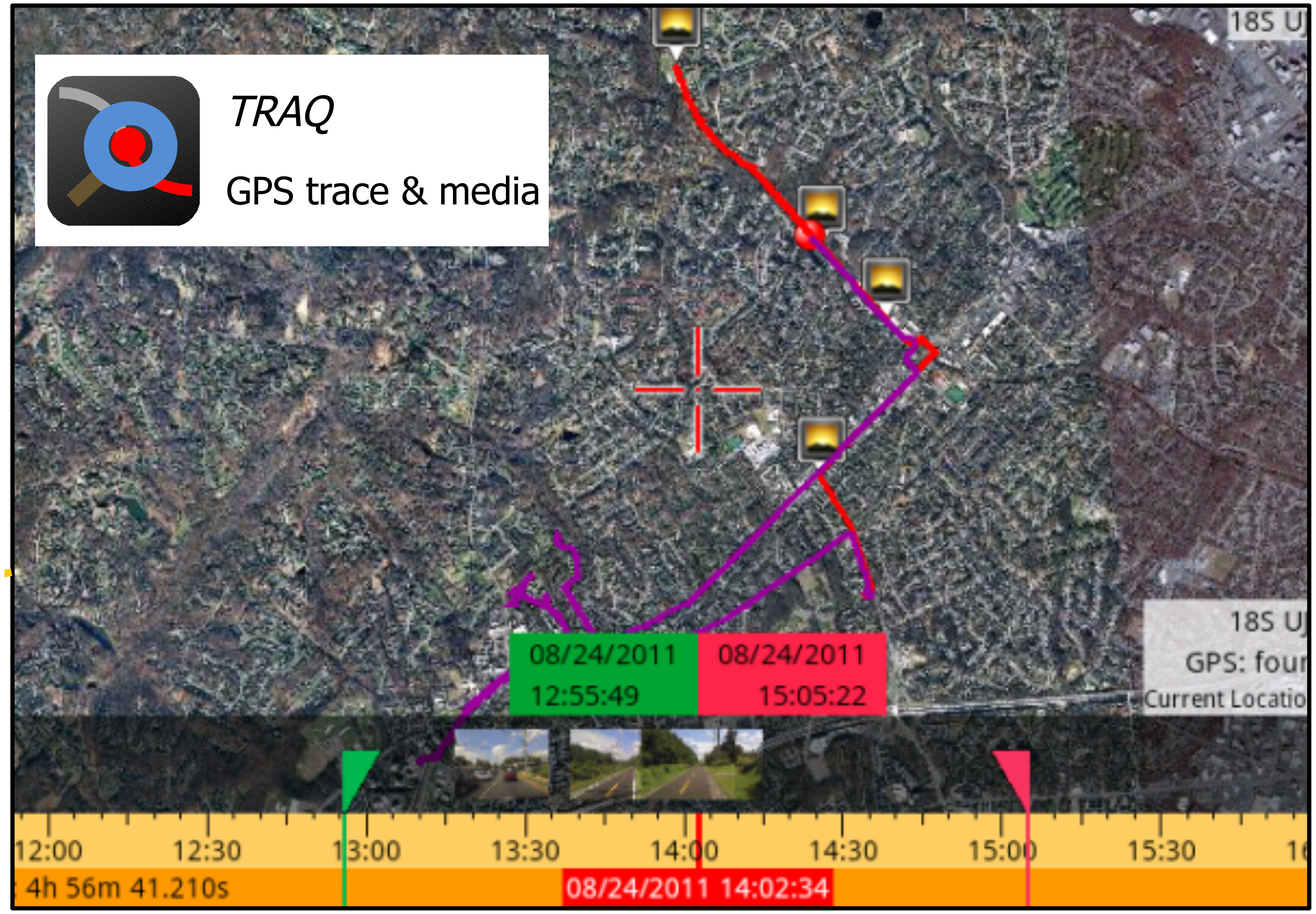
- **Maps** – High res maps navigation, layers
- **Mapdraw** – Operational graphics
- **TRAQ** – GPS trace & media
- **Transheat** ★ – Patrol heatmap
- **Debrief** ★ – Patrol review
- **TIGR** – Events, Places People, Reports

With or without network

- **Patrolview** – Collector & Viewer
- **WhoDat** ★ – Local population
- **Trip Ticket** ★ – Personnel & Eqpt mgmt
- **Agora** – Apps portal & feedback
- **Nowtu** – User empowered training
- **Paranav** – Airborne navigation

Works without network

- **WAM** ★ – Weapons and Ammunitions
- **Dimmer** ★ – Night time ops
- **Transtalk** – Pashto and Dari Translator
- **MSD** ★ – Minimum safety distance
- **Medical training modules** – Administer morphine, Apply tourniquet
- **Gammapix** – Radiation detection
- **ACOZ** ★ – Julian date converter


TRAQ
GPS trace & media
18S U
08/24/2011
12:55:49
08/24/2011
15:05:22
GPS: fou
Current Locatio
12:00
12:30
13:00
13:30
14:00
14:30
15:00
15:30
4h 56m 41.210s
08/24/2011 14:02:34


Transheat
Patrol heatmap

Example Apps (1:2)

★ **Invented in theater**

Works with network

- **PLI** – Blue-force tracking
- **DASH** – Messages
- **RTC** – Chat rooms
- **Collab** – Live Collaboration
- **SAR** – Coordinated Search & Rescue
- **STREAM** – Sensor streaming
- **SpeedTest** – Network performance

With or without network

- **Maps** – High res maps navigation, layers
- **Mapdraw** – Operational graphics
- **TRAQ** – GPS trace & media
- **Transheat** ★ – Patrol heatmap
- **Debrief** ★ – Patrol review
- **TIGR** – Events, Places People, Reports

With or without network

- **Patrolview** – Collector & Viewer
- **WhoDat** ★ – Local population
- **Trip Ticket** ★ – Personnel & Eqpt mgmt
- **Agora** – Apps portal & feedback
- **Nowtu** – User empowered training
- **Paranav** – Airborne navigation

Works without network

- **WAM** ★ – Weapons and Ammunitions
- **Dimmer** ★ – Night time ops
- **Transtalk** – Pashto and Dari Translator
- **MSD** ★ – Minimum safety distance
- **Medical training modules** – Administer morphine, Apply tourniquet
- **Gammapix** – Radiation detection
- **ACOZ** ★ – Julian date converter

When is this software trustworthy?

Example Apps (1:2)

★Invented in theater

Works with network

- PLI — Blue-force tracking
- DASH — Messages
- RTC — Chat rooms
- Collab — Live Collaboration
- SAR — Coordinated Search & Rescue
- STREAM — Sensor streaming
- SpeedTest — Network performance

With or without network

- Maps — High res maps navigation, layers
- Mapdraw — Operational graphics
- TRAQ — GPS trace & media
- Transheat ★ — Patrol heatmap
- Debrief ★ — Patrol review
- TIGR — Events, Places People, Reports

With or without network

- Patrolview — Collector & Viewer
- WhoDat ★ — Local population
- Trip Ticket ★ — Personnel & Eqpt mgmt
- Agora — Apps portal & feedback
- Nowtu — User empowered training
- Paranav — Airborne navigation

Works without network

- WAM ★ — Weapons and Ammunitions
- Dimmer ★ — Night time ops
- Transtalk — Pashto and Dari Translator
- MSD ★ — Minimum safety distance
- Medical training modules — Administer morphine, Apply tourniquet
- Gammapix — Radiation detection
- ACOZ ★ — Julian date converter

"When we trust a system, we trust it will behave as we expect it to."

— Bruce Schneier

Trust "involves the risk of failure or harm to the trustor if the trustee will not behave as desired."

— Wikipedia, *Trust (social sciences)*

When is this software trustworthy?

DARPA Example Apps (1:2)

★ **Invented in theater**

Works with network

- PLI — Blue-force tracking
- DASH — Messages
- RTC — Chat rooms
- Collab — Live Collaboration
- SAR — Coordinated Search & Rescue
- STREAM — Sensor streaming
- SpeedTest — Network performance

With or without network

- Maps — High res maps navigation, layers
- Mapdraw — Operational graphics
- TRAQ — GPS trace & media
- Transheat ★ — Patrol heatmap
- Debrief ★ — Patrol review
- TIGR — Events, Places People, Reports

With or without network

- Patrolview — Collector & Viewer
- WhoDat ★ — Local population
- Trip Ticket ★ — Personnel & Eqpt mgmt
- Agora — Apps portal & feedback
- Nowtu — User empowered training
- Paranav — Airborne navigation

Works without network

- WAM ★ — Weapons and Ammunitions
- Dimmer ★ — Night time ops
- Transtalk — Pashto and Dari Translator
- MSD ★ — Minimum safety distance
- Medical training modules — Administer morphine, Apply tourniquet
- Gammapix — Radiation detection
- ACOZ ★ — Julian date converter

When is this software trustworthy?

When we can predict it will not misbehave.

When is this software misbehaving?

Example Apps (1:2)

★ Invented in theater

Works with network

- PLI — Blue-force tracking
- DASH — Messages
- RTC — Chat rooms
- Collab — Live Collaboration
- SAR — Coordinated Search & Rescue
- STREAM — Sensor streaming
- SpeedTest — Network performance

With or without network

- Maps — High res maps navigation, layers
- Mapdraw — Operational graphics
- TRAQ — GPS trace & media
- Transheat ★ — Patrol heatmap
- Debrief ★ — Patrol review
- TIGR — Events, Places People, Reports

With or without network

- Patrolview — Collector & Viewer
- WhoDat ★ — Local population
- Trip Ticket ★ — Personnel & Eqpt mgmt
- Agora — Apps portal & feedback
- Nowtu — User empowered training
- Paranav — Airborne navigation

Works without network

- WAM ★ — Weapons and Ammunitions
- Dimmer ★ — Night time ops
- Transtalk — Pashto and Dari Translator
- MSD ★ — Minimum safety distance
- Medical training modules — Administer morphine, Apply tourniquet
- Gammapix — Radiation detection
- ACOZ ★ — Julian date converter

When is this software misbehaving?

TRAQ

GPS trace & media

When it sends GPS data
to a non-white listed URL.

When is this software misbehaving?

Gammapix

Radiation detection

When it uses the network.

When is this software misbehaving?

SpeedTest

Network performance

When it uses the camera.

When is this software misbehaving?

Paranav

Airborne navigation

When it raises an uncaught exception.

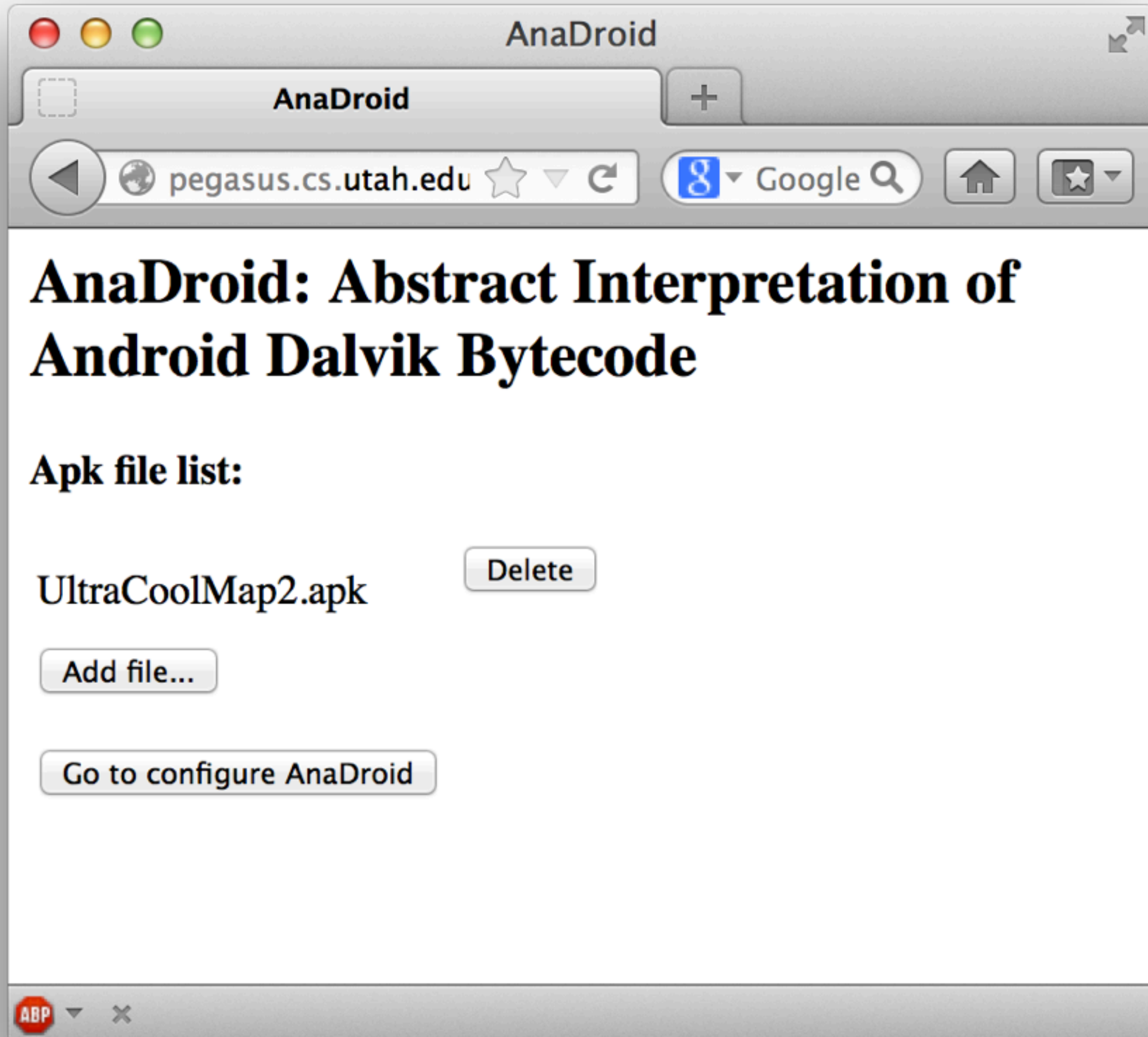
When is this software misbehaving?

To trust software,

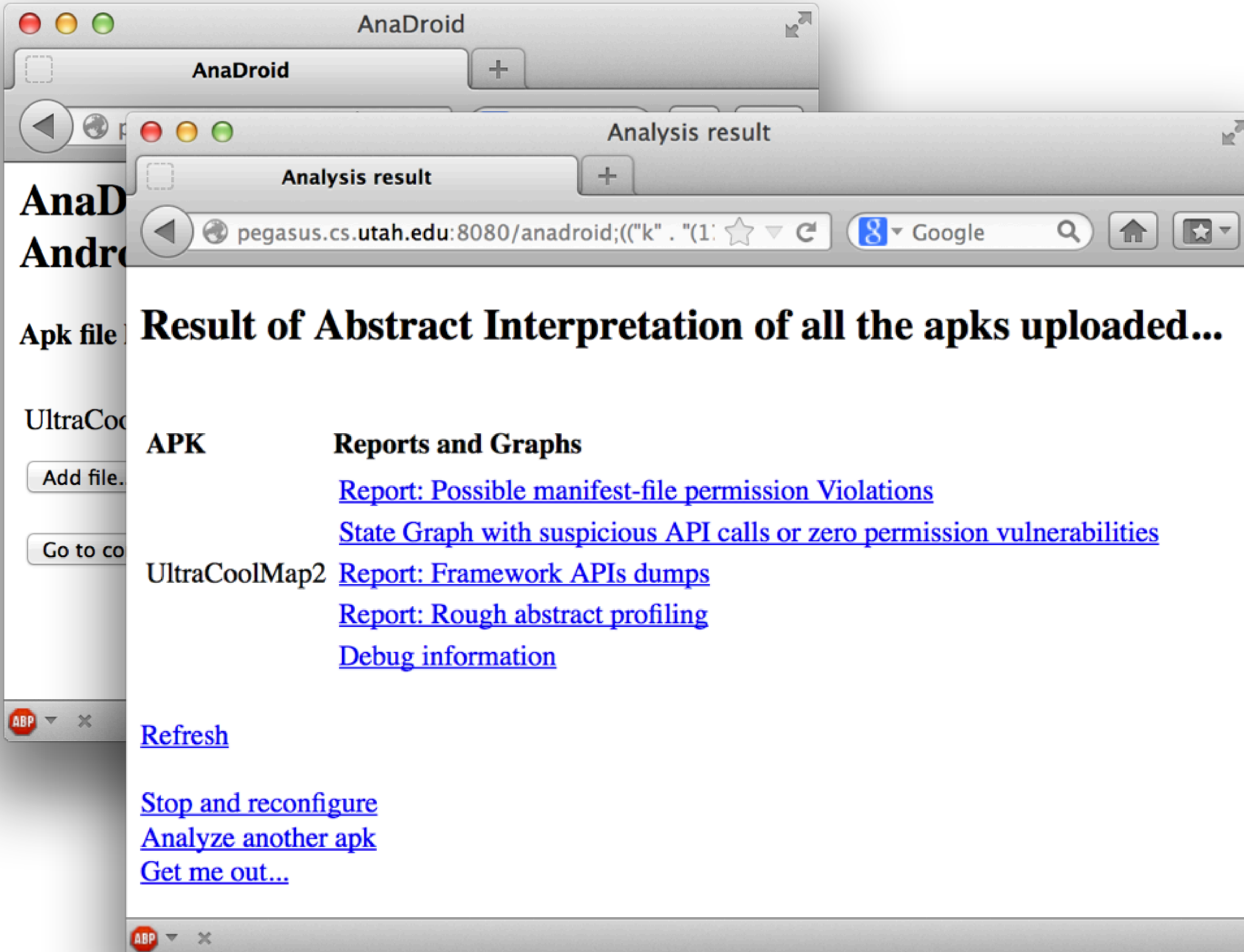
we must predict its (mis)behavior.

To trust software,
we must predict its (mis)behavior.

I build tools and techniques for
predicting the behavior of software.

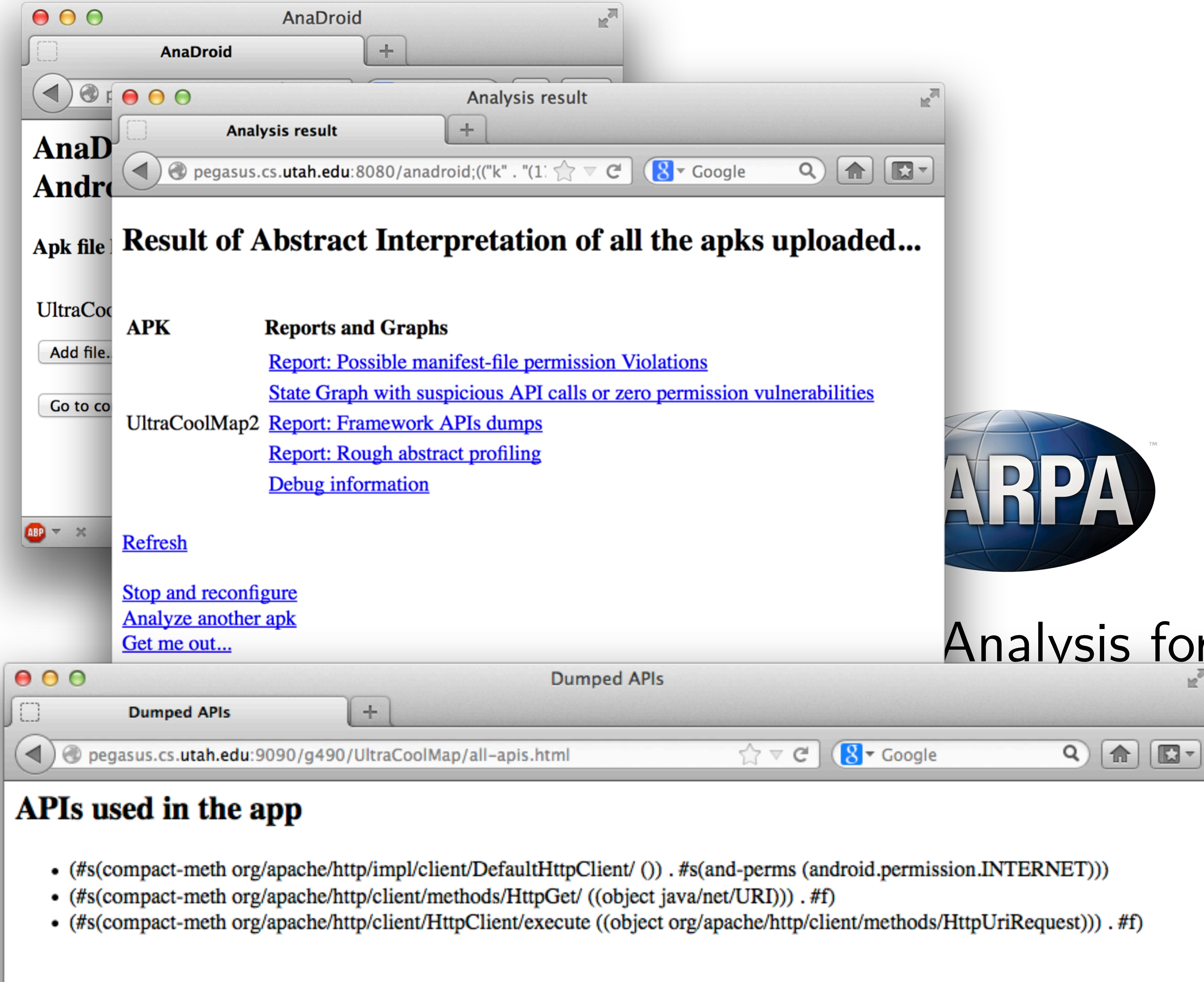


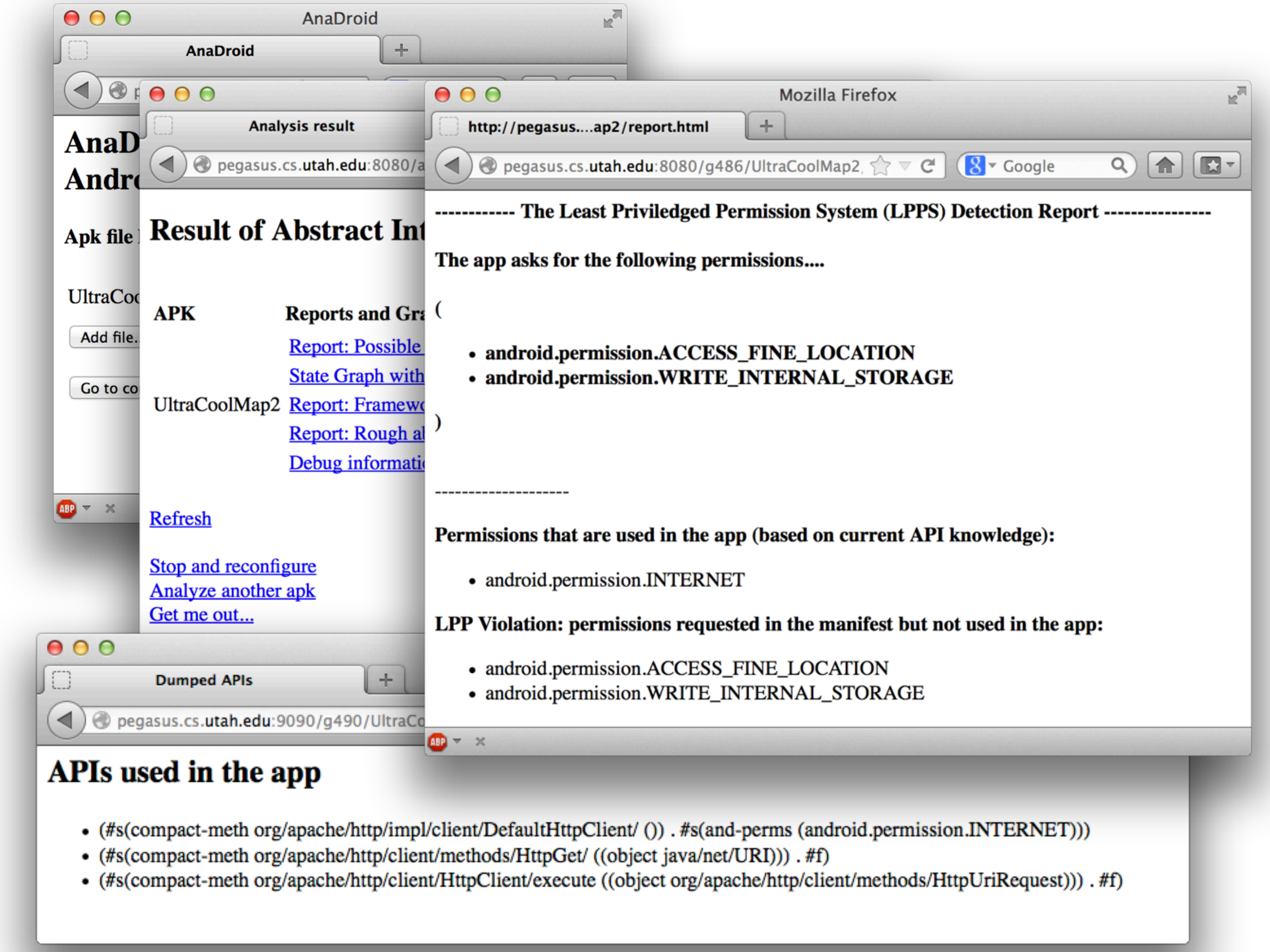
Automated Program Analysis for Cybersecurity (APAC)



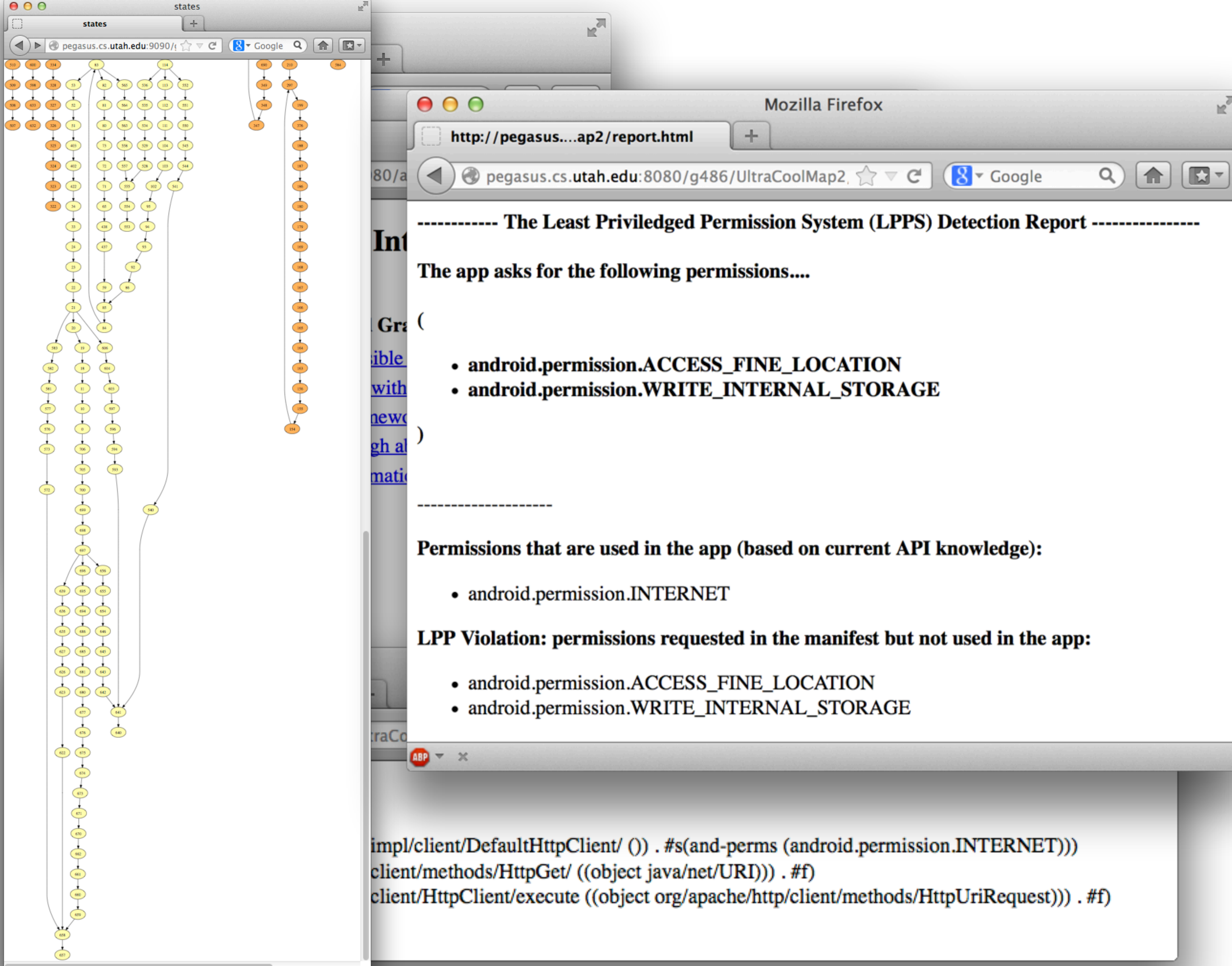
Analysis for

Cybersecurity (APAC)

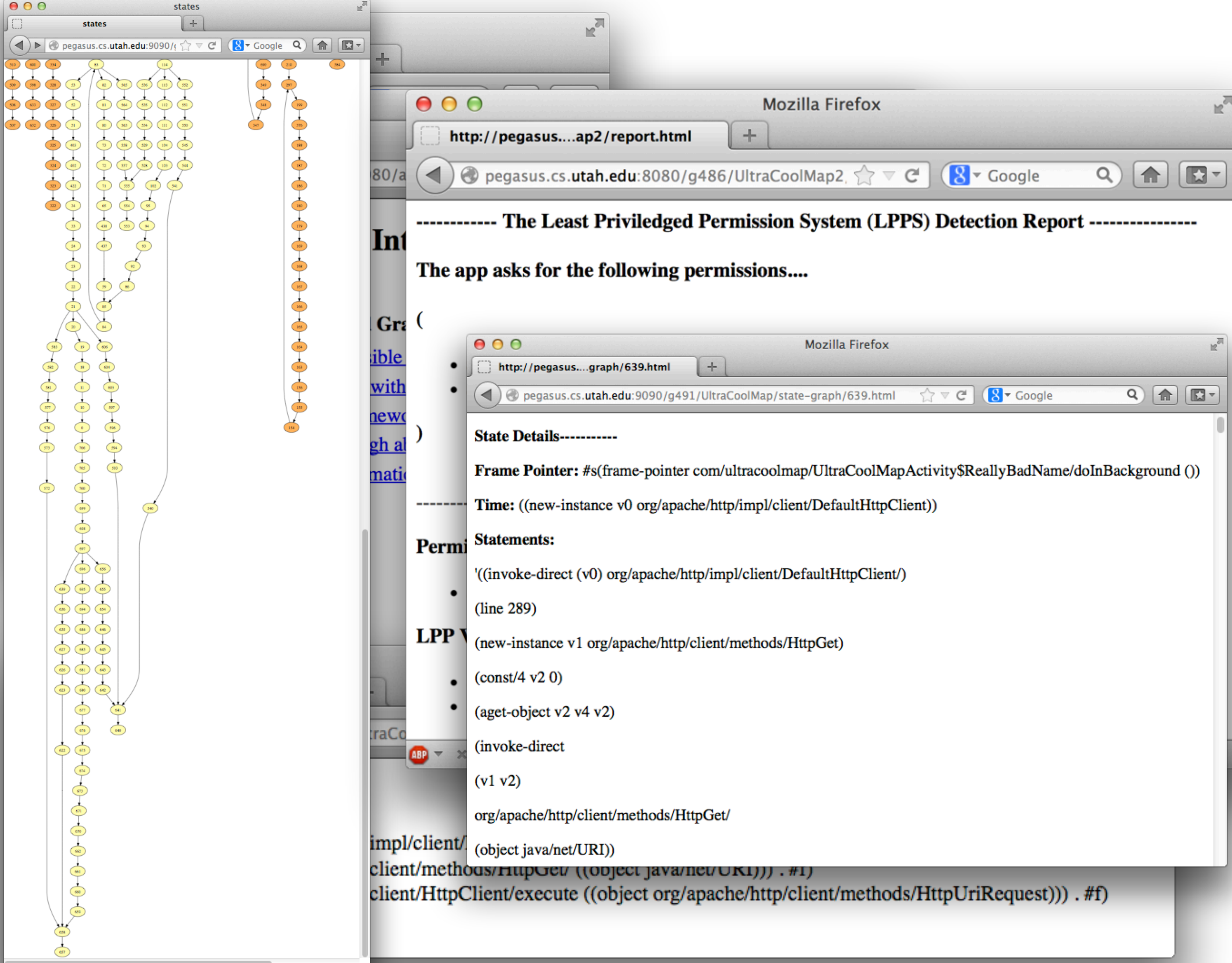

AnaDroid
Analysis result
pegasus.cs.utah.edu:8080/anadroid;(("k" . "(1:
Result of Abstract Interpretation of all the apks uploaded...
APK
Reports and Graphs
UltraCoolMap2
Report: Possible manifest-file permission Violations
State Graph with suspicious API calls or zero permission vulnerabilities
Report: Framework APIs dumps
Report: Rough abstract profiling
Debug information
Refresh
Stop and reconfigure
Analyze another apk
Get me out...
ARPA
Analysis for
Dumped APIs
pegasus.cs.utah.edu:9090/g490/UltraCoolMap/all-apis.html
APIs used in the app
(#s(compact-meth org/apache/http/impl/client/DefaultHttpClient/ ()) . #s(and-perms (android.permission.INTERNET)))
(#s(compact-meth org/apache/http/client/methods/HttpGet/ ((object java/net/URI))) . #f)
(#s(compact-meth org/apache/http/client/HttpClient/execute ((object org/apache/http/client/methods/HttpUriRequest))) . #f)


AnaDroid
AnaDroid
Analysis result
pegasus.cs.utah.edu:8080/a
Result of Abstract Int
APK
Reports and Gra
Report: Possible
State Graph with
UltraCoolMap2
Report: Framewo
Report: Rough al
Debug informati
Refresh
Stop and reconfigure
Analyze another apk
Get me out...
Apk file
UltraCoo
Add file.
Go to co
Mozilla Firefox
http://pegasus....ap2/report.html
pegasus.cs.utah.edu:8080/g486/UltraCoolMap2
Google
------------ The Least Priviledged Permission System (LPPS) Detection Report ----------------
The app asks for the following permissions....
(
android.permission.ACCESS_FINE_LOCATION
android.permission.WRITE_INTERNAL_STORAGE
)

Permissions that are used in the app (based on current API knowledge):
android.permission.INTERNET
LPP Violation: permissions requested in the manifest but not used in the app:
android.permission.ACCESS_FINE_LOCATION
android.permission.WRITE_INTERNAL_STORAGE
Dumped APIs
pegasus.cs.utah.edu:9090/g490/UltraCo
APIs used in the app
(#s(compact-meth org/apache/http/impl/client/DefaultHttpClient/ ()) . #s(and-perms (android.permission.INTERNET)))
(#s(compact-meth org/apache/http/client/methods/HttpGet/ ((object java/net/URI))) . #f)
(#s(compact-meth org/apache/http/client/HttpClient/execute ((object org/apache/http/client/methods/HttpUriRequest))) . #f)


states
pegasus.cs.utah.edu:9090/
Mozilla Firefox
http://pegasus....ap2/report.html
pegasus.cs.utah.edu:8080/g486/UltraCoolMap2,
Google
------------ The Least Priviledged Permission System (LPPS) Detection Report ----------------
The app asks for the following permissions....
(
android.permission.ACCESS_FINE_LOCATION
android.permission.WRITE_INTERNAL_STORAGE
)

Permissions that are used in the app (based on current API knowledge):
android.permission.INTERNET
LPP Violation: permissions requested in the manifest but not used in the app:
android.permission.ACCESS_FINE_LOCATION
android.permission.WRITE_INTERNAL_STORAGE
impl/client/DefaultHttpClient/ ()) . #s(and-perms (android.permission.INTERNET)))
client/methods/HttpGet/ ((object java/net/URI))) . #f)
client/HttpClient/execute ((object org/apache/http/client/methods/HttpUriRequest))) . #f)


states
pegasus.cs.utah.edu:9090/
Mozilla Firefox
http://pegasus....ap2/report.html
pegasus.cs.utah.edu:8080/g486/UltraCoolMap2,
------------ The Least Priviledged Permission System (LPPS) Detection Report ----------------
The app asks for the following permissions....
(
)
Permi
LPP
Mozilla Firefox
http://pegasus....graph/639.html
pegasus.cs.utah.edu:9090/g491/UltraCoolMap/state-graph/639.html
State Details-----------
Frame Pointer: #s(frame-pointer com/ultracoolmap/UltraCoolMapActivity$ReallyBadName/doInBackground ())
Time: ((new-instance v0 org/apache/http/impl/client/DefaultHttpClient))
Statements:
'((invoke-direct (v0) org/apache/http/impl/client/DefaultHttpClient/)
(line 289)
(new-instance v1 org/apache/http/client/methods/HttpGet)
(const/4 v2 0)
(aget-object v2 v4 v2)
(invoke-direct
(v1 v2)
org/apache/http/client/methods/HttpGet/
(object java/net/URI))
client/HttpClient/execute ((object org/apache/http/client/methods/HttpUriRequest))) . #f)

AndorsTrail	SplitTimer	SMSBackup
AndroidGame	SuperNote	SMSBlocker
AndroidPrivacyGuard_E	SuperSoduko	SMSPopup
Butane	SysMon	SysWatcherA
CalcA	SysWatcherB	SourceViewer
CalcB	TextSecure	UltraCoolMap
ConnectBot	TodoList	YARR
CountdownTimer	Word Helper	AndroidsFortune
FunDraw	AndBible	CalcC
MorseCode	AndroidPrivacyGuard_M	CalcE
MyDrawA	BatteryIndicator	ColorMatcher
MyDrawC	CalcF	FullControl
NewsCollator	MediaFun	KitteyKittey
PasswordSaver	MyDrawD	Orienteering2
PersistantAssistant	OpenGPSTracker	Sanity
SmartWebCam	Orienteering1	TomDroid
SMSReminder	PicViewer	WiFinder
SourceViewer	Collaboration ShareLoc	

AndorsTrail	SplitTimer	SMSBackup
AndroidGame	SuperNote	SMSBlocker
AndroidPrivacyGuard_E	SuperSoduko	SMSPopup
Butane	SysMon	SysWatcherA
CalcA	SysWatcherB	SourceViewer
CalcB	TextSecure	UltraCoolMap
ConnectBot	TodoList	YARR
CountdownTimer	Word Helper	AndroidsFortune
FunDraw	AndBible	CalcC
MorseCode	AndroidPrivacyGuard_M	CalcE
MyDrawA	BatteryIndicator	ColorMatcher
MyDrawC	CalcF	FullControl
NewsCollator	MediaFun	KitteyKittey
PasswordSaver	MyDrawD	Orienteering2
PersistantAssistant	OpenGPSTracker	Sanity
SmartWebCam	Orienteering1	TomDroid
SMSReminder	PicViewer	WiFinder
SourceViewer	Collaboration ShareLoc	

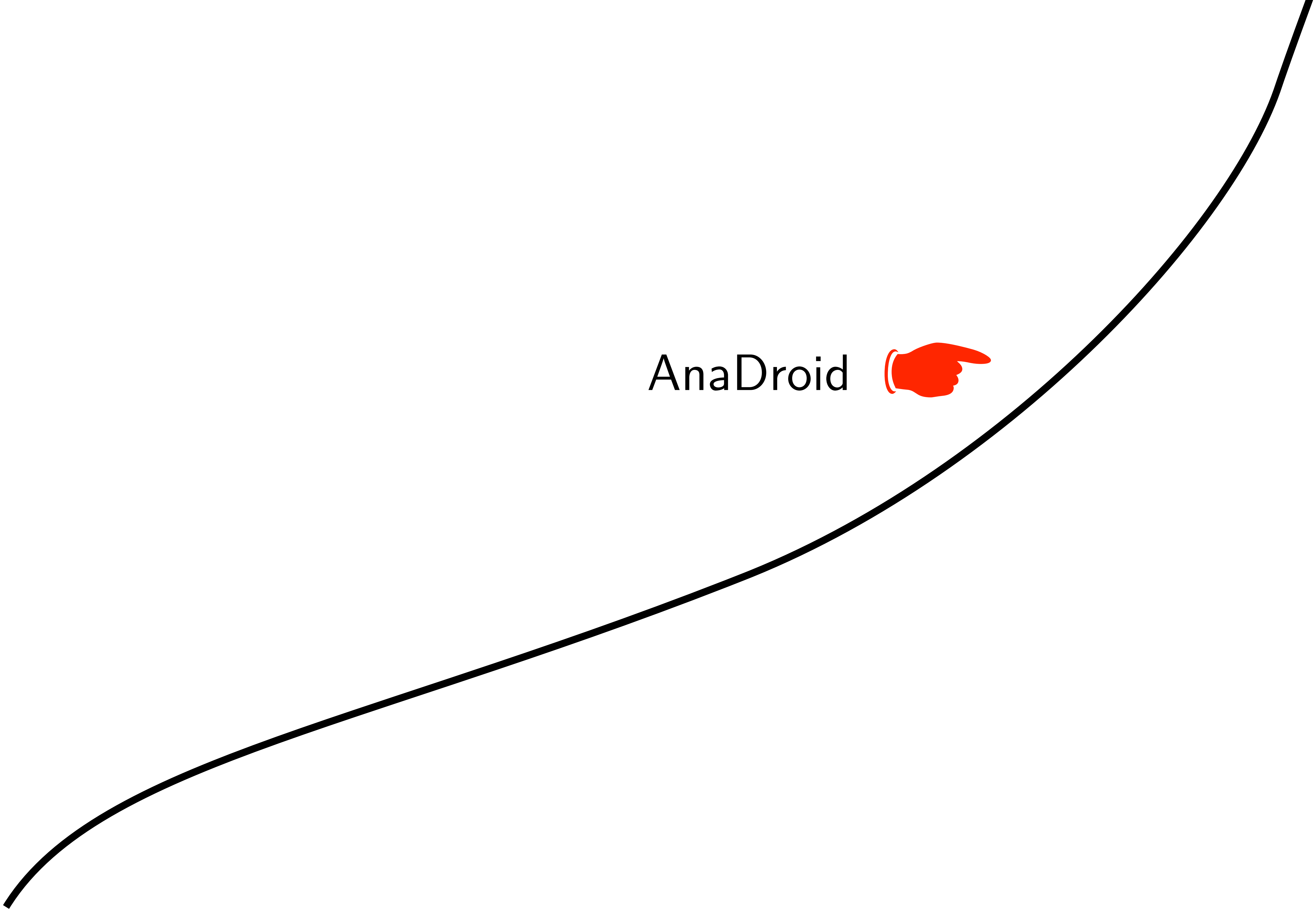
AndorsTrail	SplitTimer	SMSBackup
AndroidGame	SuperNote	SMSBlocker
AndroidPrivacyGuard_E	SuperSoduko	SMSPopup
Butane	SysMon	SysWatcherA
CalcA	SysWatcherB	SourceViewer
CalcB	TextSecure	UltraCoolMap
ConnectBot	TodoList	YARR
CountdownTimer	Word Helper	AndroidsFortune
FunDraw	AndBible	CalcC
MorseCode	AndroidPrivacyGuard_M	CalcE
MyDrawA	BatteryIndicator	ColorMatcher
MyDrawC	CalcF	FullControl
NewsCollator	MediaFun	KitteyKittey
PasswordSaver	MyDrawD	Orienteering2
PersistantAssistant	OpenGPSTracker	Sanity
SmartWebCam	Orienteering1	TomDroid
SMSReminder	PicViewer	WiFinder
SourceViewer	Collaboration ShareLoc	

DARPA

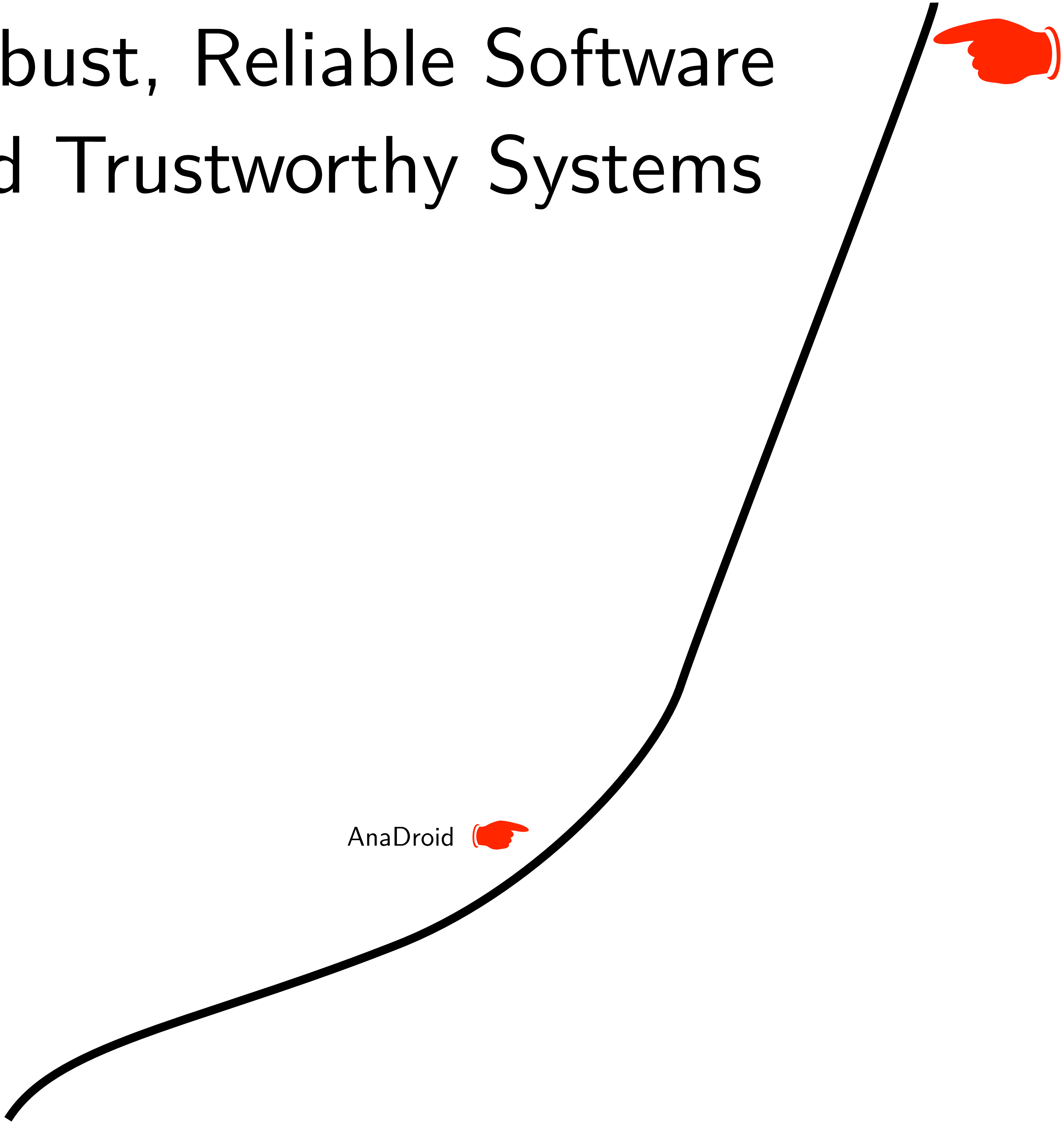
AndorsTrail	SplitTimer	SMSBackup
AndroidGame	SuperNote	SMSBlocker
AndroidPrivacyGuard_E	SuperSoduko	SMSPopup
Butane	SysMon	SysWatcherA
CalcA	SysWatcherB	SourceViewer
CalcB	TextSecure	UltraCoolMap
ConnectBot	TodoList	YARR
CountdownTimer	Word Helper	AndroidsFortune
FunDraw	AndBible	CalcC
MorseCode	AndroidPrivacyGuard_M	CalcE
MyDrawA	BatteryIndicator	ColorMatcher
MyDrawC	CalcF	FullControl
NewsCollator	MediaFun	KitteyKittey
PasswordSaver	MyDrawD	Orienteering2
PersistantAssistant	OpenGPSTracker	Sanity
SmartWebCam	Orienteering1	TomDroid
SMSReminder	PicViewer	WiFinder
SourceViewer	Collaboration ShareLoc	

I build tools and techniques for predicting the behavior of software.

I build tools and techniques for **soundly** and **effectively** predicting the behavior of software.

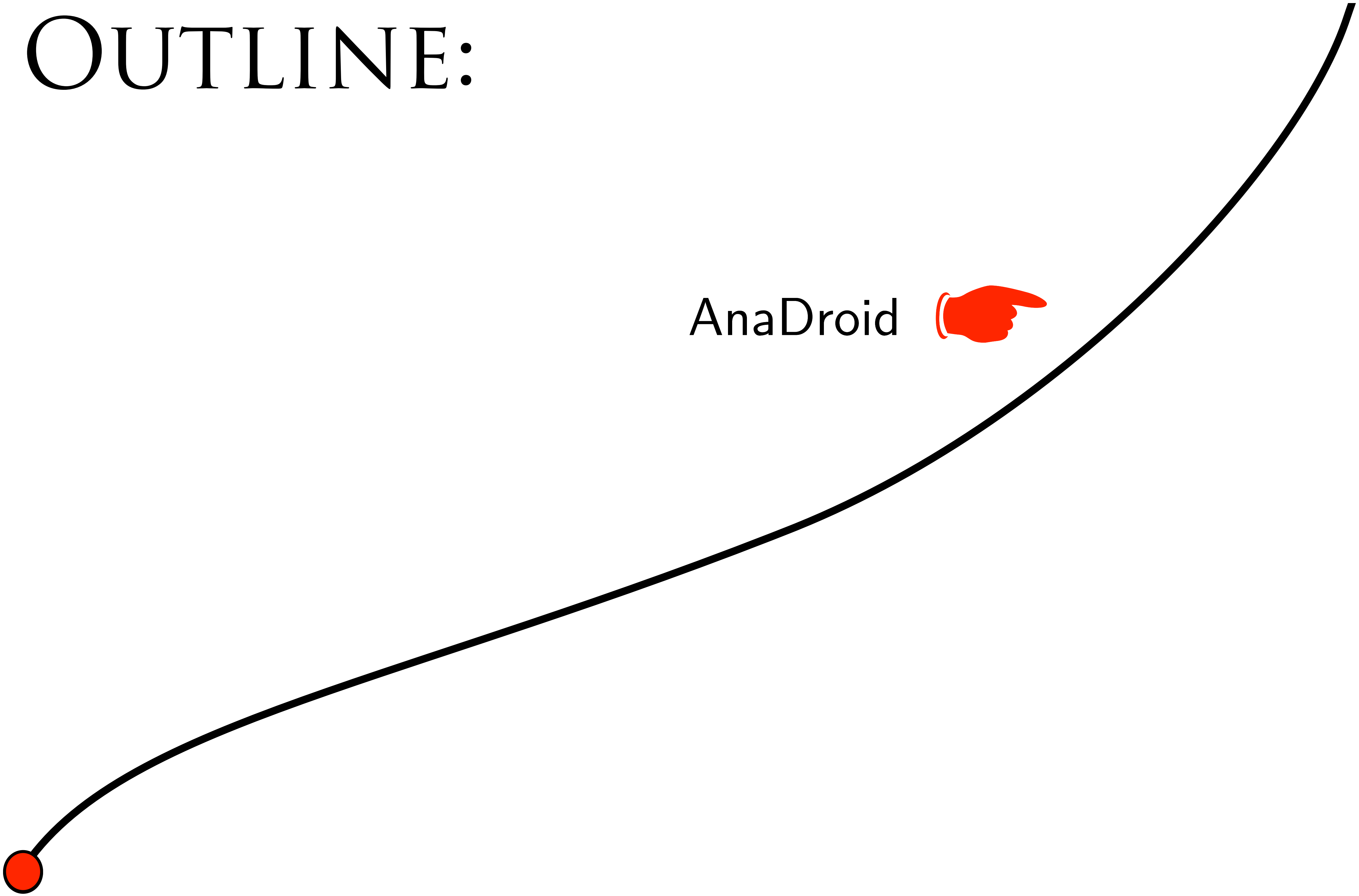

AnaDroid

Robust, Reliable Software and Trustworthy Systems

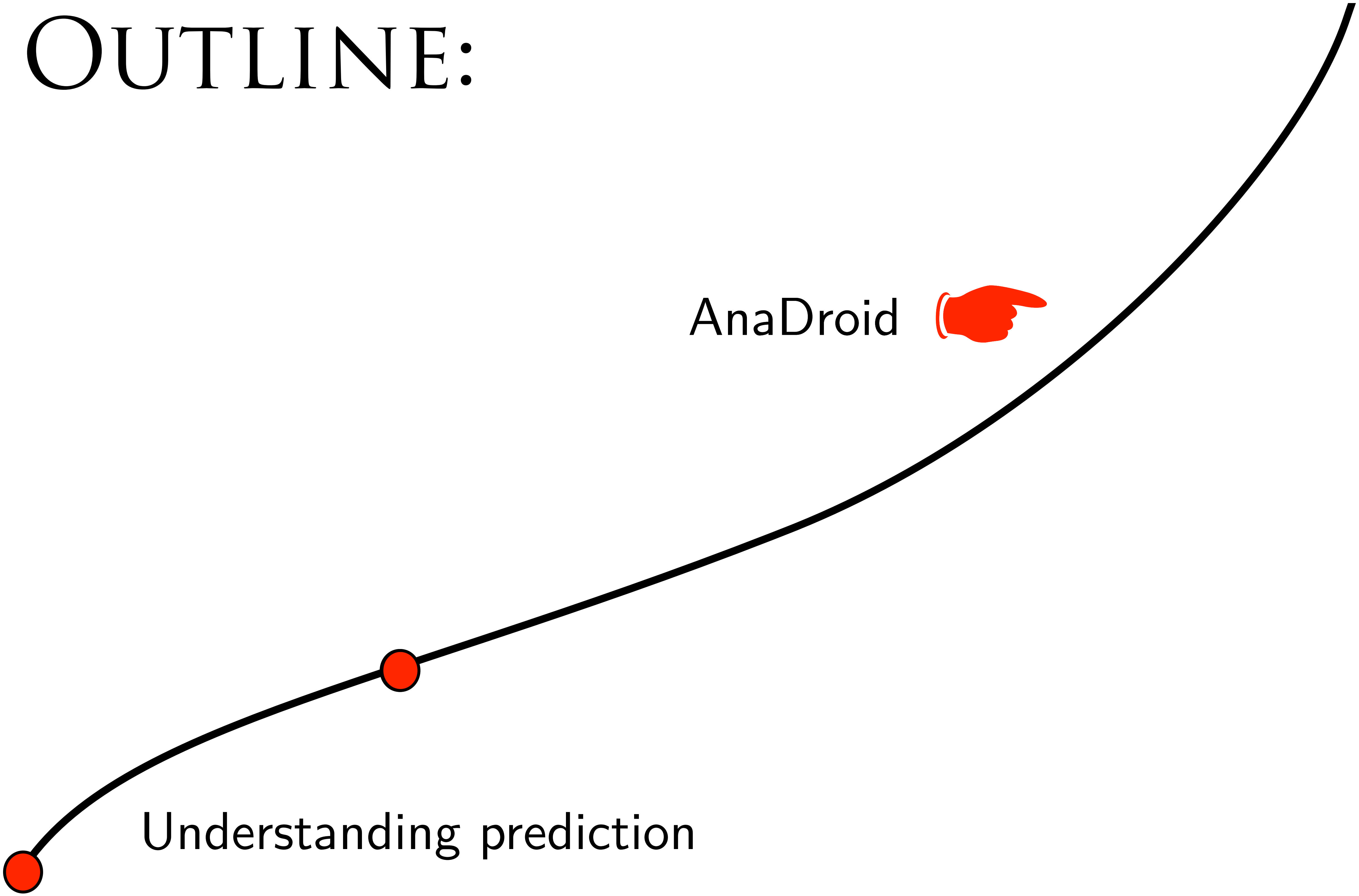


AnaDroid

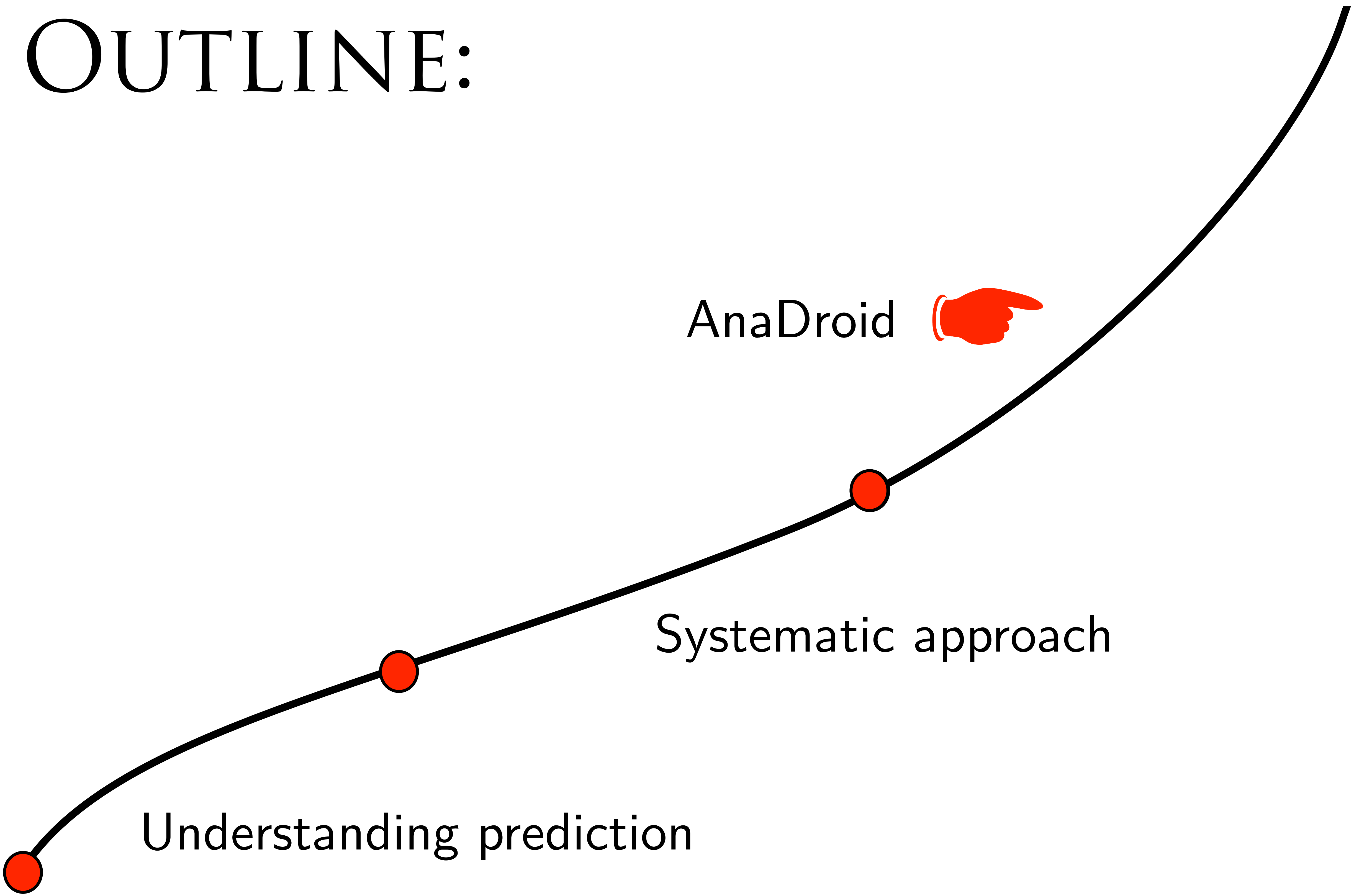
Outline:



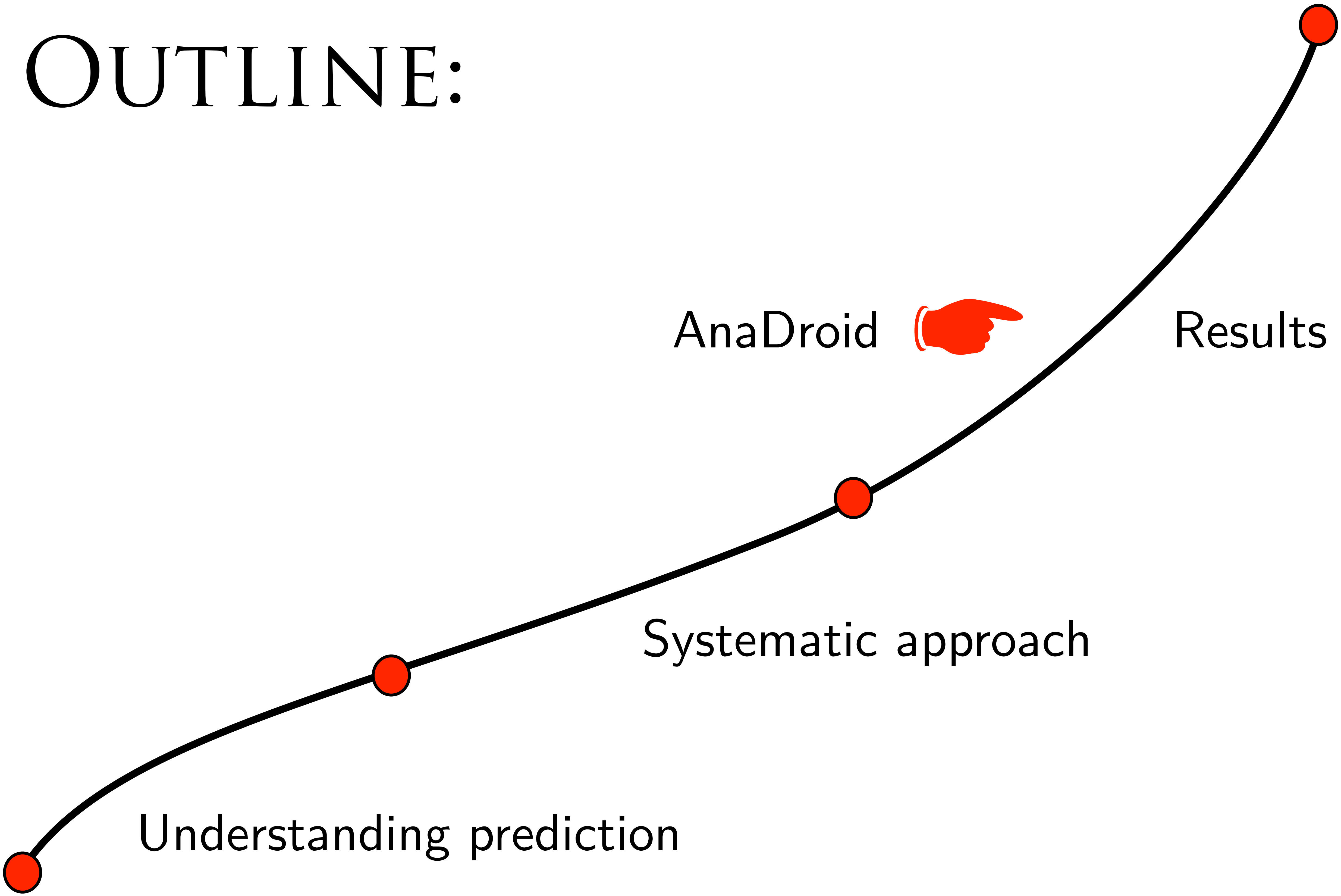
Outline:



Outline:



Outline:


AnaDroid
Results
Systematic approach
Understanding prediction

Part I: Understanding Prediction

To trust software, we must predict its (mis)behavior.

```
x.m()
```

To trust software, we must predict its (mis)behavior.

```
public void f(XYZ x) {
    return x.m();
}
```

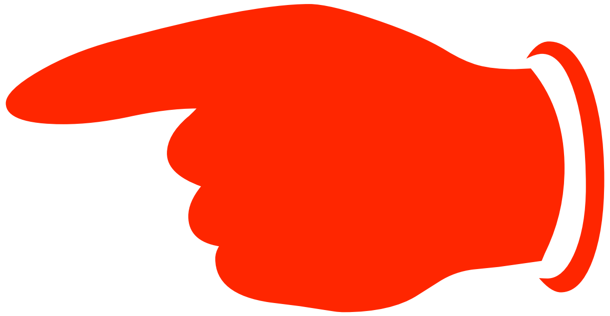
To trust software, we must predict its (mis)behavior.

```java
public void f(XYZ x) {
    return x.m();
}
```

To trust software, we must predict its (mis)behavior.

```
public void f(XYZ x) {
    return x.m();
}
```

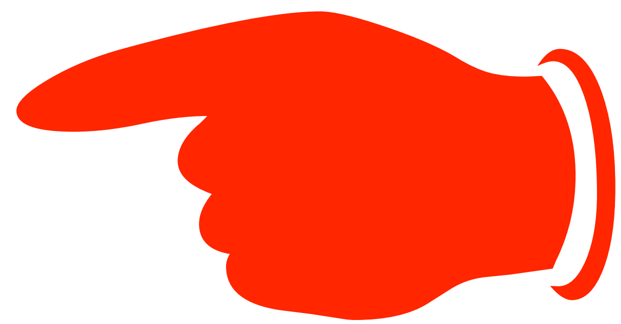
Modern software uses computational values.



To predict control flow,
we must predict data flow

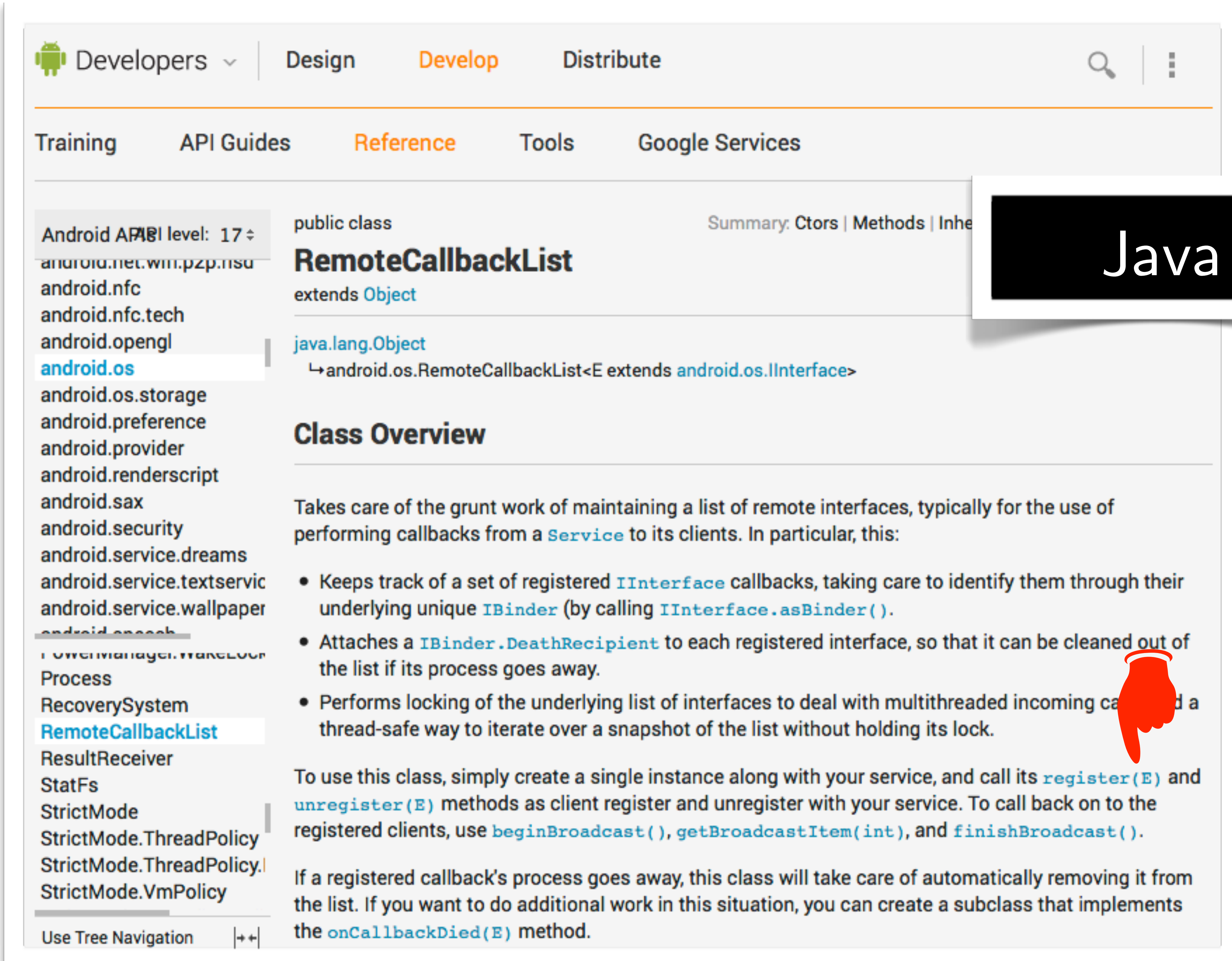
To predict data flow,
we must predict control flow

Modern software uses computational values.



To predict control flow,
we must predict data flow

To predict data flow,
we must predict control flow



Developers Design Develop Distribute

Training API Guides Reference Tools Google Services

Android API level: 17

android.nfc
android.nfc.tech
android.opengl
android.os
android.os.storage
android.preference
android.provider
android.renderscript
android.sax
android.security
android.service.dreams
android.service.textservic
android.service.wallpaper

Process
RecoverySystem
RemoteCallbackList
ResultReceiver
StatFs
StrictMode
StrictMode.ThreadPolicy
StrictMode.ThreadPolicy.
StrictMode.VmPolicy

Use Tree Navigation

public class

Summary: Ctors | Methods | Inhe

RemoteCallbackList

extends Object

java.lang.Object
↳android.os.RemoteCallbackList<E extends android.os.IInterface>

Class Overview

Takes care of the grunt work of maintaining a list of remote interfaces, typically for the use of performing callbacks from a `Service` to its clients. In particular, this:

- Keeps track of a set of registered `IInterface` callbacks, taking care to identify them through their underlying unique `IBinder` (by calling `IInterface.asBinder()`.
- Attaches a `IBinder.DeathRecipient` to each registered interface, so that it can be cleaned out of the list if its process goes away.
- Performs locking of the underlying list of interfaces to deal with multithreaded incoming ca… d a thread-safe way to iterate over a snapshot of the list without holding its lock.

To use this class, simply create a single instance along with your service, and call its `register(E)` and `unregister(E)` methods as client register and unregister with your service. To call back on to the registered clients, use `beginBroadcast()`, `getBroadcastItem(int)`, and `finishBroadcast()`.

If a registered callback's process goes away, this class will take care of automatically removing it from the list. If you want to do additional work in this situation, you can create a subclass that implements the `onCallbackDied(E)` method.

Java

java.util

Class Observable

java.lang.Object
  └ **java.util.Observable**

```
public class Observable
extends Object
```

This class represents an observable object, or "data" in the model-view paradigm. It can be subclassed to represent an object that the application wants to have observed.

An observable object can have one or more observers. An observer may be any object that implements interface `Observer`. After an observable instance changes, an application calling the `Observable`'s `notifyObservers` method causes all of its

Java

Constructor Summary

Observable()   Construct an Observable with zero Observers.

Method Summary

void	**addObserver**(Observer o) Adds an observer to the set ... ers for this object, provided that it is not the same as some observer already in the set.
protected void	**clearChanged**() Indicates that this object has no longer changed, or that it has already notified all of its observers of its most recent change, so that the `hasChanged` method will now return `false`.
int	**countObservers**() Returns the number of observers of this `Observable` object.
void	**deleteObserver**(Observer o) Deletes an observer from the set of observers of this object.
void	**deleteObservers**() Clears the observer list so that this object no longer has any observers.
boolean	**hasChanged**() Tests if this object has changed.
void	**notifyObservers**() If this object has changed, as indicated by the `hasChanged` method, then notify all of its observers and then call the `clearChanged` method to indicate that this object has no longer changed.
void	**notifyObservers**(Object arg) If this object has changed, as indicated by the `hasChanged` method, then notify all of its observers and then call the `clearChanged` method to indicate that this object has no longer changed.
protected void	**setChanged**() Marks this `Observable` object as having been changed; the `hasChanged` method will now return `true`.

W3C Candidate Recommendation

XMLHttpRequest

W3C Candidate Recommendation 3 August 2010

This Versi
http://
Latest Vers
http://
Latest Edit
http://
Previous V
http://
http://
http://
http://
http://
http://
http://
http://
http://
Editor:
Anne

Abstract

The XMLHt
functionality

JavaScript

W3C Candidate Recommendation

1. Introduction

This section is non-normative.

The XMLHttpRequest object implements an interface exposed by a scripting engine that allows scripts to perform HTTP client functionality, such as submitting form data or loading data from a server. It is the ECMAScript HTTP API.

The name of the object is XMLHttpRequest for compatibility with the Web, though each component of this name is potentially misleading. First, the object supports any text based format, including XML. Second, it can be used to make requests over both HTTP and HTTPS (some implementations support protocols in addition to HTTP and HTTPS, but that functionality is not covered by this specification). Finally, it supports "requests" in a broad sense of the term as it pertains to HTTP; namely all activity involved with HTTP requests or responses for the defined HTTP methods.

Some simple code to do something with data from an XML document fetched over the network:

```
function test(data) {
 // taking care of data
}

function handler() {
 if(this.readyState == 4 && this.status == 200) {
  // so far so good
  if(this.responseXML != null && this.responseXML.getElementById('test').firstChild.data)
     // success!
   test(this.responseXML.getElementById('test').firstChild.data);
  else
   test(null);
 } else if (this.readyState == 4 && this.status != 200) {
  // fetched the wrong page or network error...
  test(null);
 }
}

var client = new XMLHttpRequest();
client.onreadystatechange = handler;
client.open("GET", "unicorn.xml");
client.send();
```

Modern software is

JavaScript

Programming Ruby

The Pragmatic Programmer's Guide

Previous < | Contents ^ | Next >

Object-Oriented Design Libraries

One of the interesting things about Ru
between design and implementation.
design level in other languages can b

To help in this process, Ruby has sup

- **The Visitor pattern** (Design Pa
without having to know the inter
- **Delegation** is a way of composi
than can be done using standar
- **The Singleton pattern** is a way
particular class exists at a time.
- **The Observer pattern** impleme
a set of interested objects when

Normally, all four of these strategies r
implemented. With Ruby, they can be
and transparently.

Ruby

Library: observer

The Observer pattern, also known as Publish/Subscribe, provides a simple mechanism for one object to inform a set of interested third-party objects when its state changes.

In the Ruby implementation, the notifying class mixes in the `Observable` module, which provides the methods for managing the associated observer objects.

add_observer(*obj*)	Add *obj* as an observer on this object. *obj* will now receive notifications.
delete_observer(*obj*)	Delete *obj* as an observer on this object. It will no longer receive notifications.
delete_observers	Delete all observers associated with this object.
count_observers	Return the count of observers associated with this object.
changed(*newState*=`true`)	Set the changed state of this object. Notifications will be sent only if the changed state is `true`.
changed?	Query the changed state of this object.
notify_observers(**args*)	If this object's changed state is true, invoke the `update` method in each currently associated observer in turn, passing it the given arguments. The changed state is then set to `false`.

The observers must implement the `update` method to receive notifications.

1.3 Functions as values

OCaml

OCaml is a functional language: functions in the full mathematical sense are supported and can be passed around freely just as any other piece of data. For instance, here is a `deriv` function that takes any float function as argument and returns an approximation of its derivative function:

```
# let deriv f dx = function x -> (f(x +. dx) -. f(x)) /. dx;;
val deriv : (float -> float) -> float -> float -> float = <fun>

# let sin' = deriv sin 1e-6;;
val sin' : float -> float = <fun>

# sin' pi;;
- : float = -1.00000000013961143
```

Even function composition is definable:

```
# let compose f g = function x -> f(g(x));;
val compose : ('a -> 'b) -> ('c -> 'a) -> 'c -> 'b = <fun>

# let cos2 = compose square cos;;
val cos2 : float -> float = <fun>
```

Functions that take other functions as arguments are called “functionals”, or “higher-order functions”. Functionals are especially useful to provide iterators or similar generic operations over a data structure. For instance, the standard OCaml library provides a `List.map` functional that applies a given function to each element of a list, and returns the list of the results:

```
# List.map (function n -> n * 2 + 1) [0;1;2;3;4];;
- : int list = [1; 3; 5; 7; 9]
```

This functional, along with a number of other list and array functionals, is predefined because it is often useful, but there is nothing magic with it: it can easily be defined as follows.

Python

10.2. `functools` — Higher-order functions and operations on callable objects

Source code: Lib/functools.py

The `functools` module is for higher-order functions: functions that act on or return other functions. In general, any callable object can be treated as a function for the purposes of this module.

The `functools` module defines the following functions:

`functools.`**`cmp_to_key`**`(`*func*`)`

Transform an old-style comparison function to a key function. Used with tools that accept key functions (such as `sorted()`, `min()`, `max()`, `heapq.nlargest()`, `heapq.nsmallest()`, `itertools.groupby()`). This function is primarily used as a transition tool for programs being converted from Python 2 which supported the use of comparison functions.

Modern software uses
computational values.

Modern software uses
computational values.

To predict its behavior,
we need **flow analysis**.

FLOW ANALYSIS OF LAMBDA EXPRESSIONS
(Preliminary Version)

Neil D. Jones
Aarhus University, Denmark

Jones, ICALP 1981

0. INTRODUCTION

A method is described to extract from an untyped λ-expression information about the sequence of intermediate λ-expressions obtained during its evaluation. The information can be used to give "safe positive answers" to questions involving termination or nontermination of the evaluation, dependence of one subexpression on another and type errors encountered while applying δ rules, thus providing an alternative to techniques of Morris and Levy ([Mor68], [Lev75]). The method works by building a "safe description" of the set of states entered by a call-by-name interpreter and analyzing this description. A similar and more complete analysis of a call-by-value interpreter may be found in [Jon81].

From a flow analysis viewpoint these results extend existing interprocedural analysis methods to include call-by-name and the use of functions both as arguments to other functions and as the results returned by them. Further, the method naturally handles both local and global variables, extending [Cou77a] and [Sha80]. It seems clear that other traditional analyses such as available expressions, constant propagation, etc. can be carried out in this framework.

The main emphasis is on development of the framework and showing its relation to abstract interpretation, rather than on its efficient use in applications. A simplified and optimized version of the method would have applications in the efficient compilation of λ-calculus-based programming languages such as LISP, SCHEME and SASL ([McC63], [Ste75], [Tur76]).

The method provides a general way to find safe approximate descriptions of computations by algorithms which manipulate recursive data structures. It is thus not limited to the λ-calculus, but may be applied to analyze any programming language whose semantics can be implemented by an appropriate definitional interpreter.

Another application would be to extend the method to the flow analysis of denotational definitions of programming languages. This could be used in semantics-directed compiler generation as described in [JoS80], and provided the initial motivation for this study.

Related work

Lambda calculus evaluators have been studied in [Böh72], [Lan64], [McG70], [Plo75], [Rey72], [Sch80] and [Weg68]. Sufficient conditions for termination of

FLOW ANALYSIS OF LAMBDA EXPRESSIONS
(Preliminary Version)

Neil D. Jones
Aarhus University, Denmark

0. INTRODUCTION

A method is described to extract from an untyped λ-expression inf
the sequence of intermediate λ-expressions obtained during its evaluati
formation can be used to give "safe positive answers" to questions invo
tion or nontermination of the evaluation, dependence of one subexpress
and type errors encountered while applying δ rules, thus providing an
techniques of Morris and Levy ([Mor68], [Lev75]). The method works
"safe description" of the set of states entered by a call-by-name interp
analyzing this description. A similar and more complete analysis of a c
interpreter may be found in [Jon81].

From a flow analysis viewpoint these results extend existing interp
analysis methods to include call-by-name and the use of functions both
to other functions and as the results returned by them. Further, the me
ly handles both local and global variables, extending [Cou77a] and [Sh
clear that other traditional analyses such as available expressions, co
gation, etc. can be carried out in this framework.

The main emphasis is on development of the framework and showing
to abstract interpretation, rather than on its efficient use in applicatio
plified and optimized version of the method would have applications in t
compilation of λ-calculus-based programming languages such as LISP,
and SASL ([McC63], [Ste75], [Tur76]).

The method provides a general way to find safe approximate descr
computations by algorithms which manipulate recursive data structures.
not limited to the λ-calculus, but may be applied to analyze any progran
guage whose semantics can be implemented by an appropriate definition
Another application would be to extend the method to the flow analy
tational definitions of programming languages. This could be used in se
directed compiler generation as described in [JoS80], and provided th
tivation for this study.

Related work

Lambda calculus evaluators have been studied in [Böh72], [Lan64]
[Plo75], [Rey72], [Sch80] and [Weg68]. Sufficient conditions for ter

Control-Flow Analysis of Functional Programs

JAN MIDTGAARD, Aarhus University

We present a survey of control-flow analysis of functional programs, which has been the subject of extensive investigation throughout the past 30 years. Analyses of the control flow of functional programs have been formulated in multiple settings and have led to many different approximations, starting with the seminal works of Jones, Shivers, and Sestoft. In this article, we survey control-flow analysis of functional programs by structuring the multitude of formulations and approximations and comparing them.

Categories and Subject Descriptors: D.3.2 [**Programming Languages**]: Language Classifications—*Applicative functional languages*; F.3.1 [**Logics and Meanings of Programs**]: Specifying and Verifying and Reasoning about Programs

General Terms: Languages, Theory, Verification

Additional Key Words and Phrases: Control-flow analysis, higher-order functions

ACM Reference Format:
Midtgaard, J. 2012. Control-flow analysis of functional programs. ACM Comput. Surv. 44, 3, Article 10 (June 2012), 33 pages.
DOI = 10.1145/2187671.2187672 http://doi.acm.org/10.1145/2187671.2187672

1. INTRO

Since th
devoted
gram m
analysis
In a l
parent f

Cites over 200 papers.

called function is available at compile time. One can thus base an analysis for such a language on the textual structure of the program, since it determines the exact *control flow* of the program, for example, as a flow chart. On the other hand, in a language with higher-order functions, the operator of a function call may not be apparent from the text of the program: it can be the result of a computation and therefore, the called function may not be available until runtime. A *control-flow analysis* approximates at compile time which functions may be applied at runtime, that is, it determines an approximate control flow of a given program.

Prerequisites. We assume some familiarity with program analysis in general and with control-flow analysis in particular. For a tutorial or an introduction to the area, we refer to Nielson et al. [1999]. We also assume familiarity with functional programming and a basic acquaintance with continuation-passing style (CPS) [Steele Jr. 1978] and

Part of this work was done with support of the Carlsberg Foundation ad an INRIA post-doc grant.
Authors' addresses: J. Midtgaard, Department of Computer Science, Aarhus University, Aabogade 34, DK-8200 Aarhus N., Denmark; email: jmi@cs.au.dk.

DOI 10.1145/2187671.2187672 http://doi.acm.org/10.1145/2187671.2187672

Existing analyses
(and their complexities)

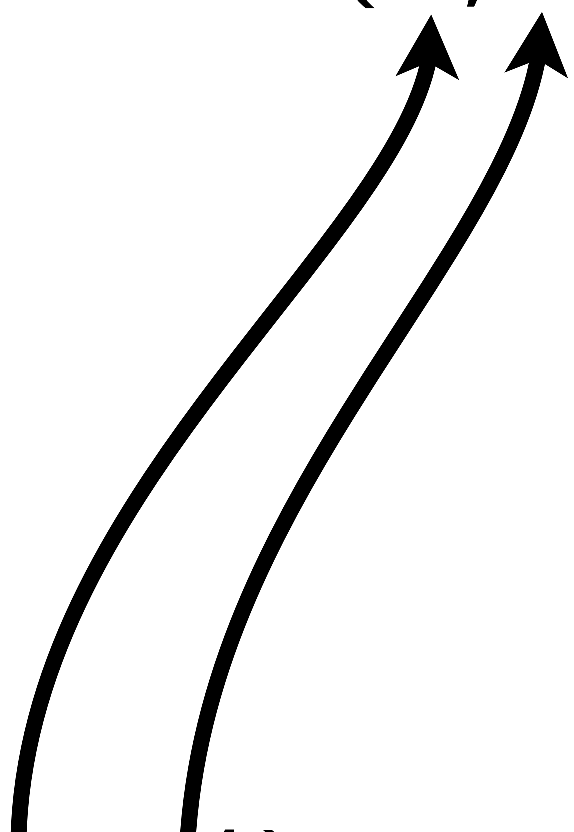
0CFA

```
function twice(f,x) { return f(f(x)); };
```

```
twice(sqr,4);            twice(dbl,5);
```

0CFA

```
function twice(f,x) { return f(f(x)); };
```



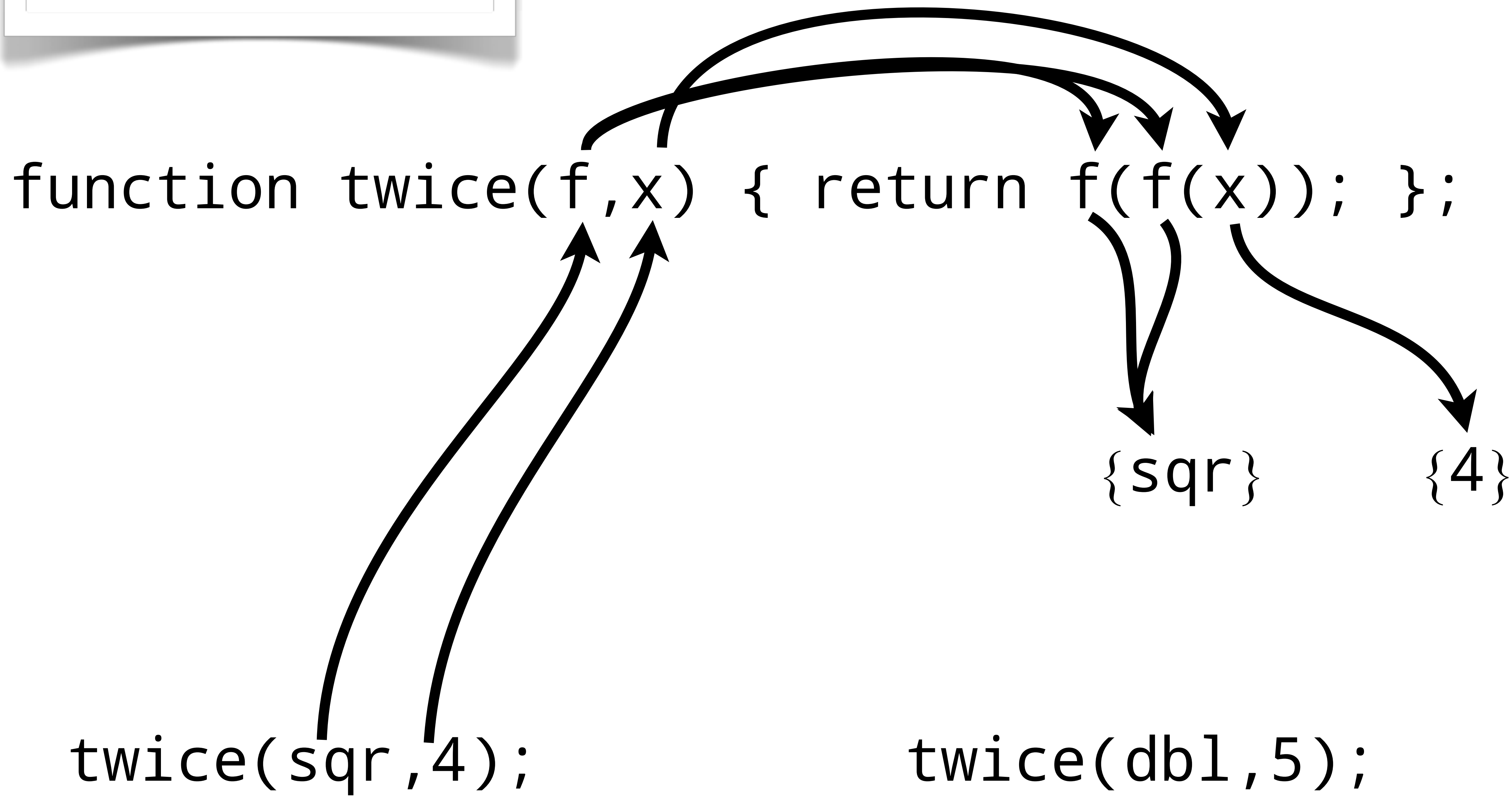
```
twice(sqr,4);                twice(dbl,5);
```

0CFA

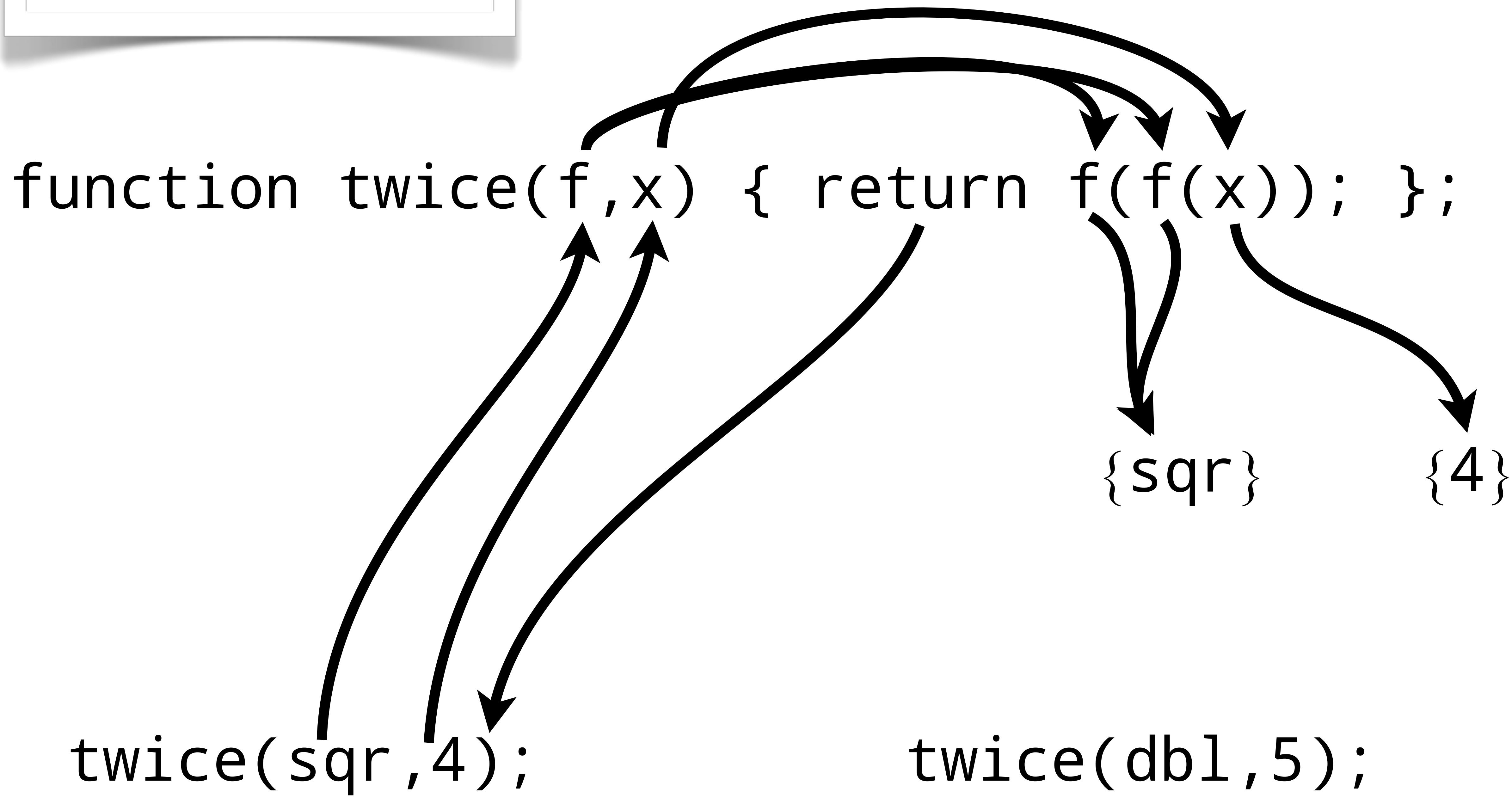
```
function twice(f,x) { return f(f(x)); };
```

```
twice(sqr,4);                    twice(dbl,5);
```

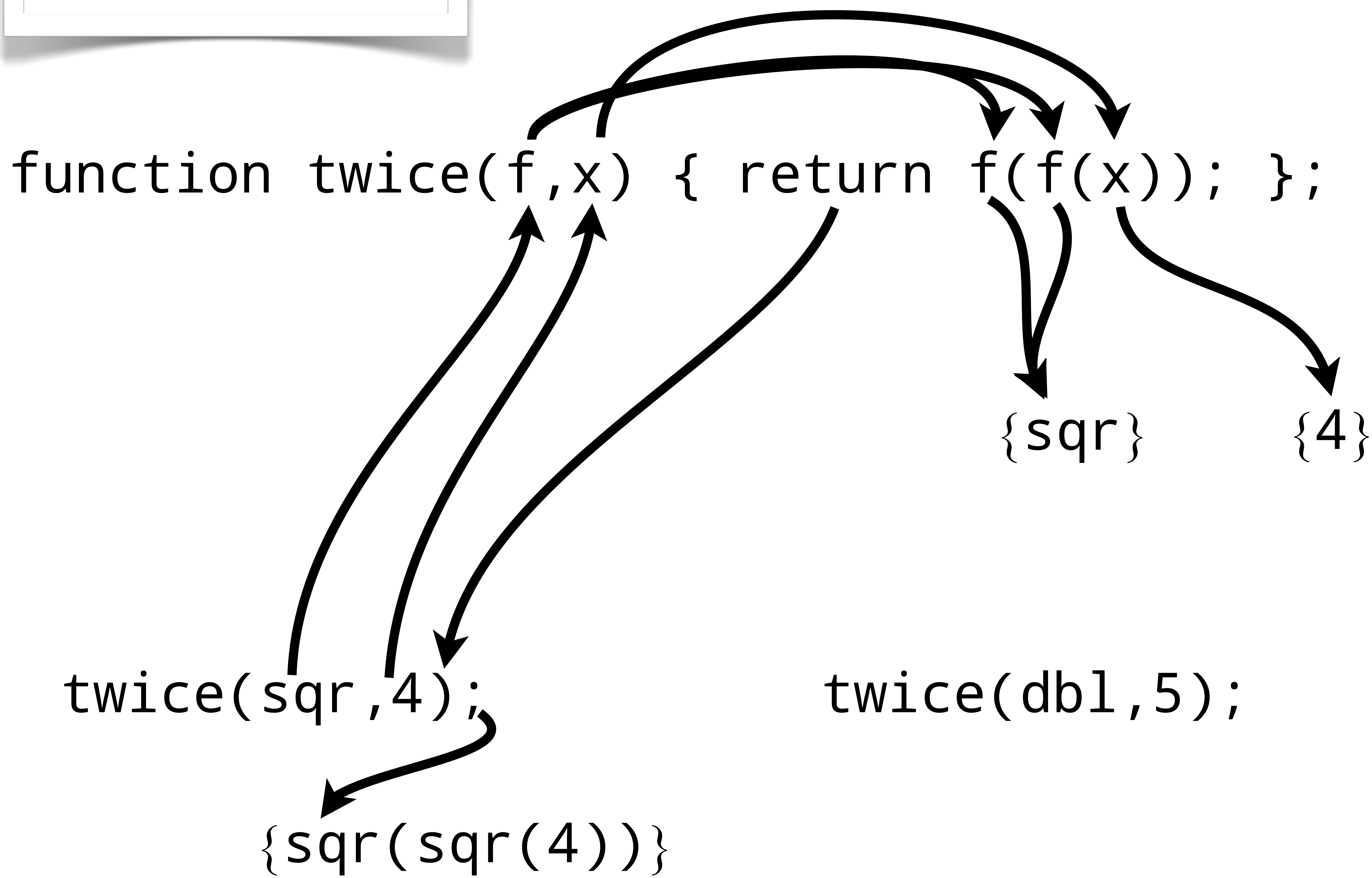
0CFA



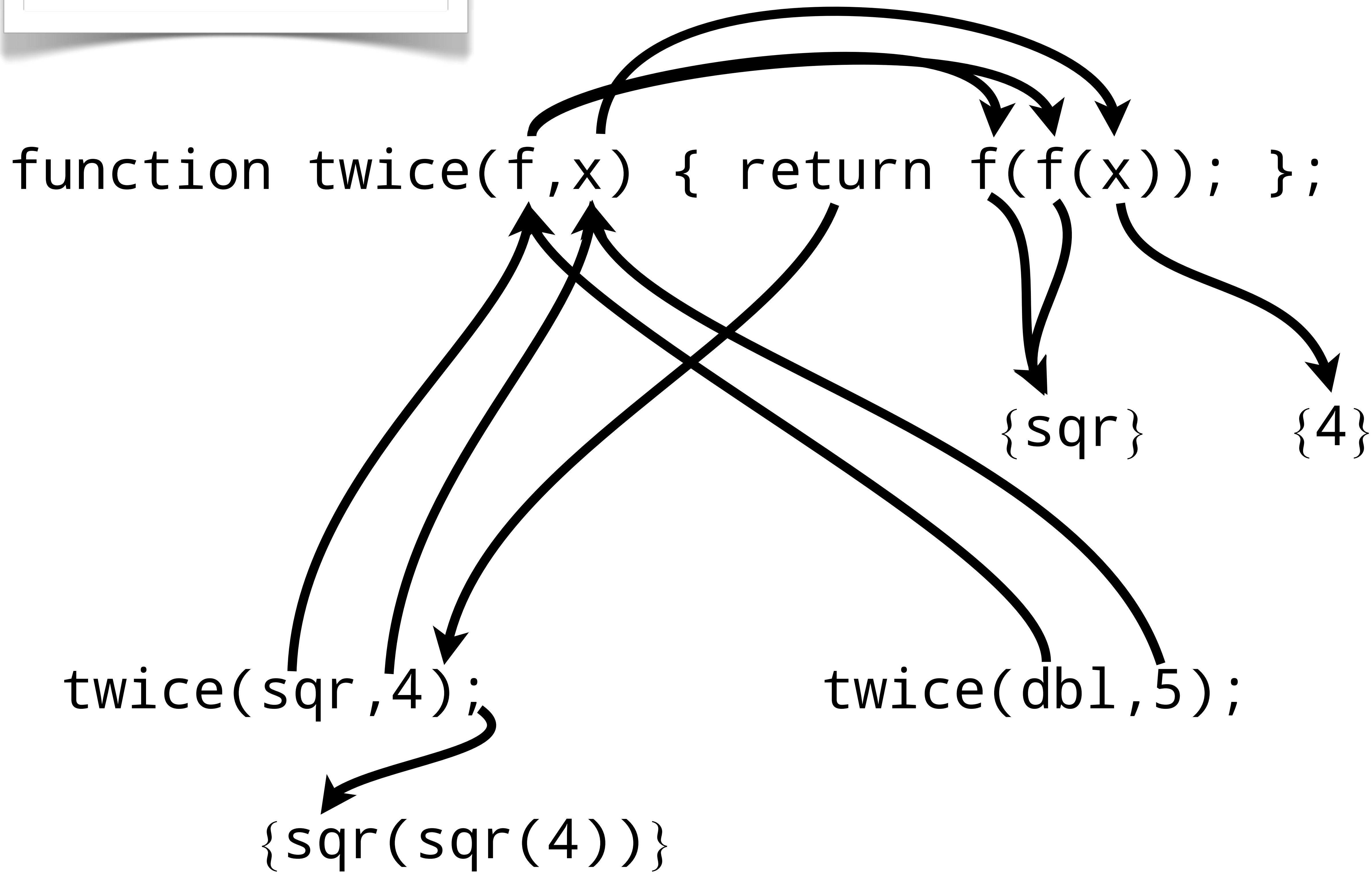
0CFA



0CFA

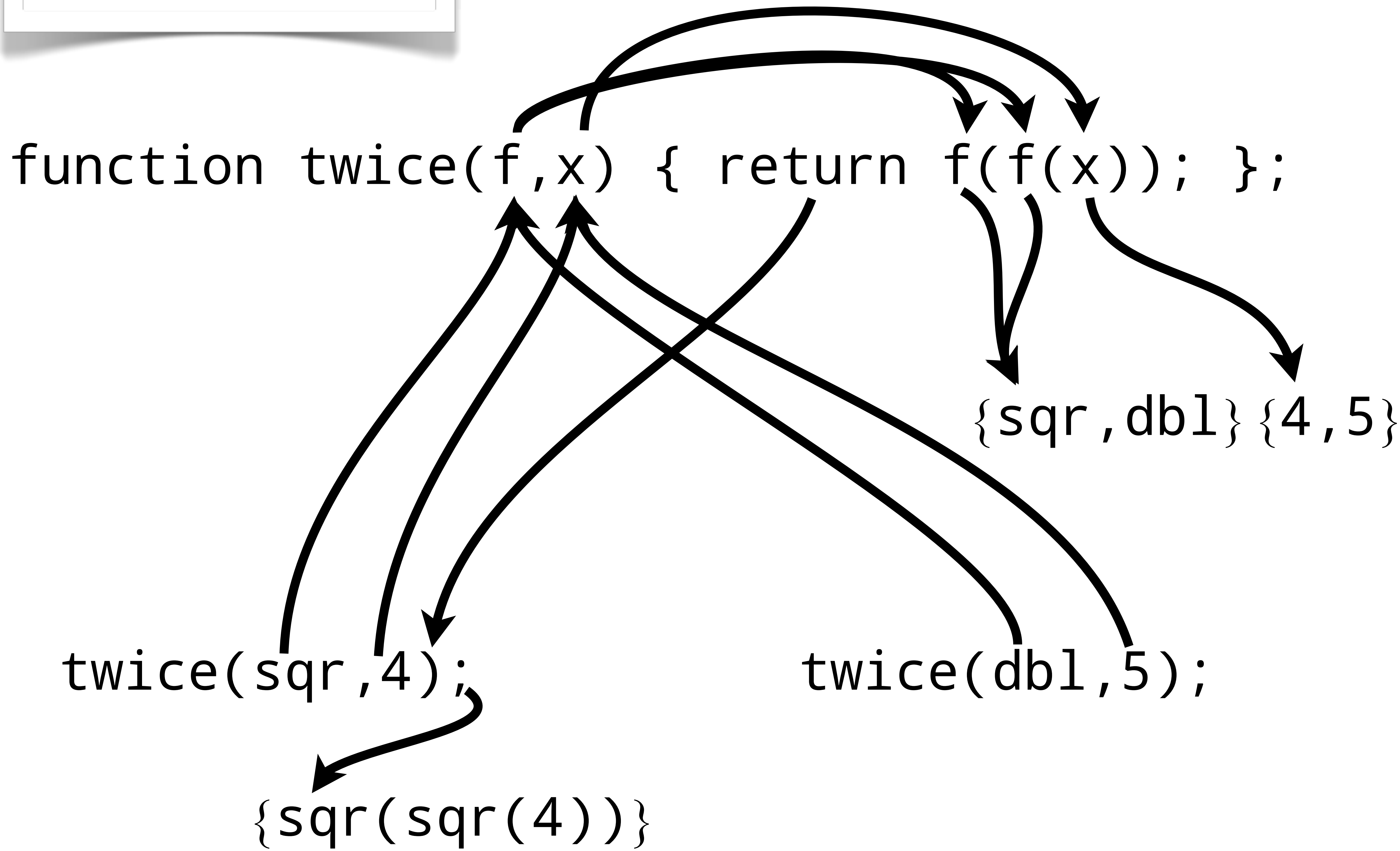

function twice(f,x) { return f(f(x)); };
{sqr}
{4}
twice(sqr,4);
twice(dbl,5);
{sqr(sqr(4))}

0CFA

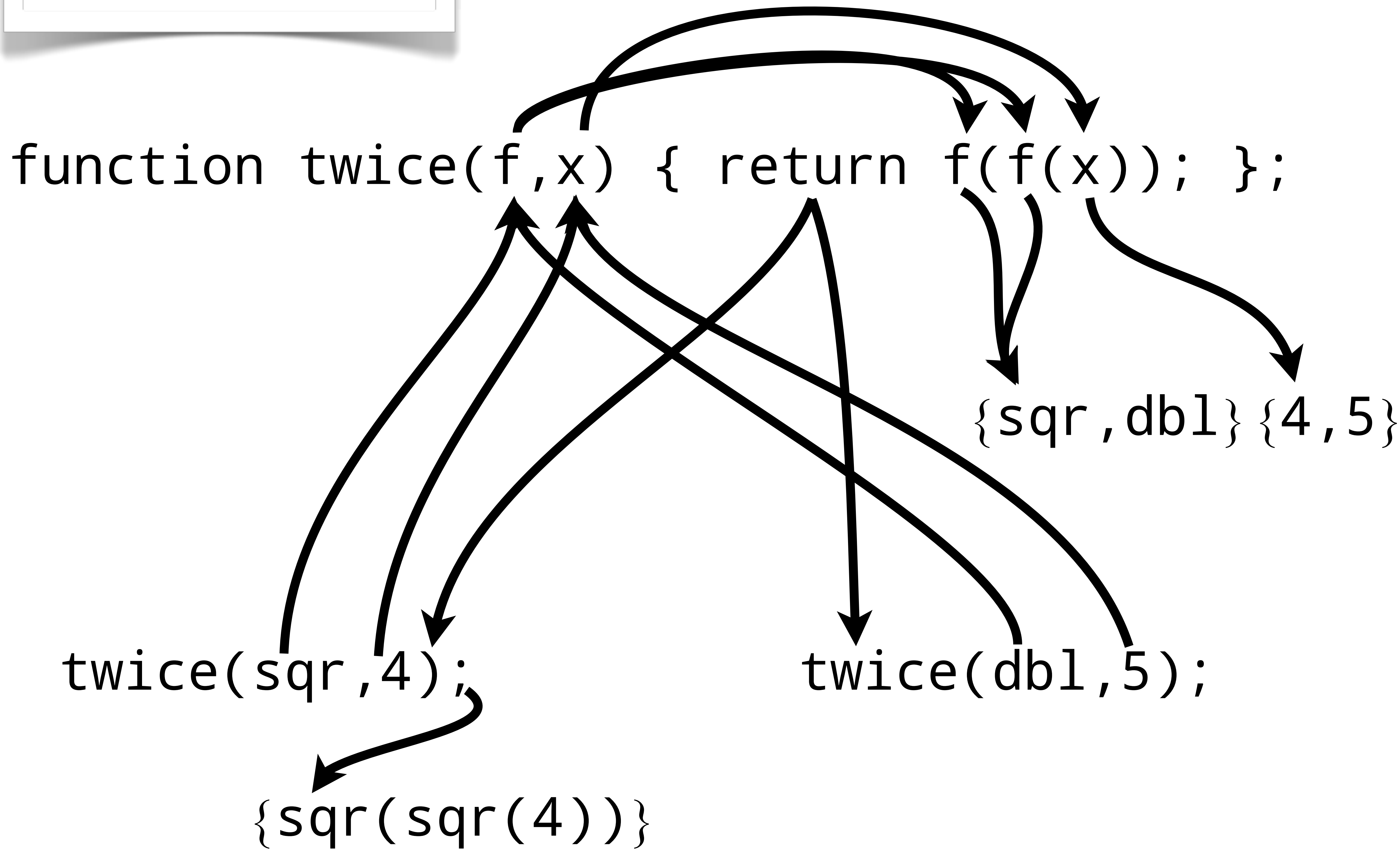


function twice(f,x) { return f(f(x)); };
{sqr}
{4}
twice(sqr,4);
twice(dbl,5);
{sqr(sqr(4))}

0CFA

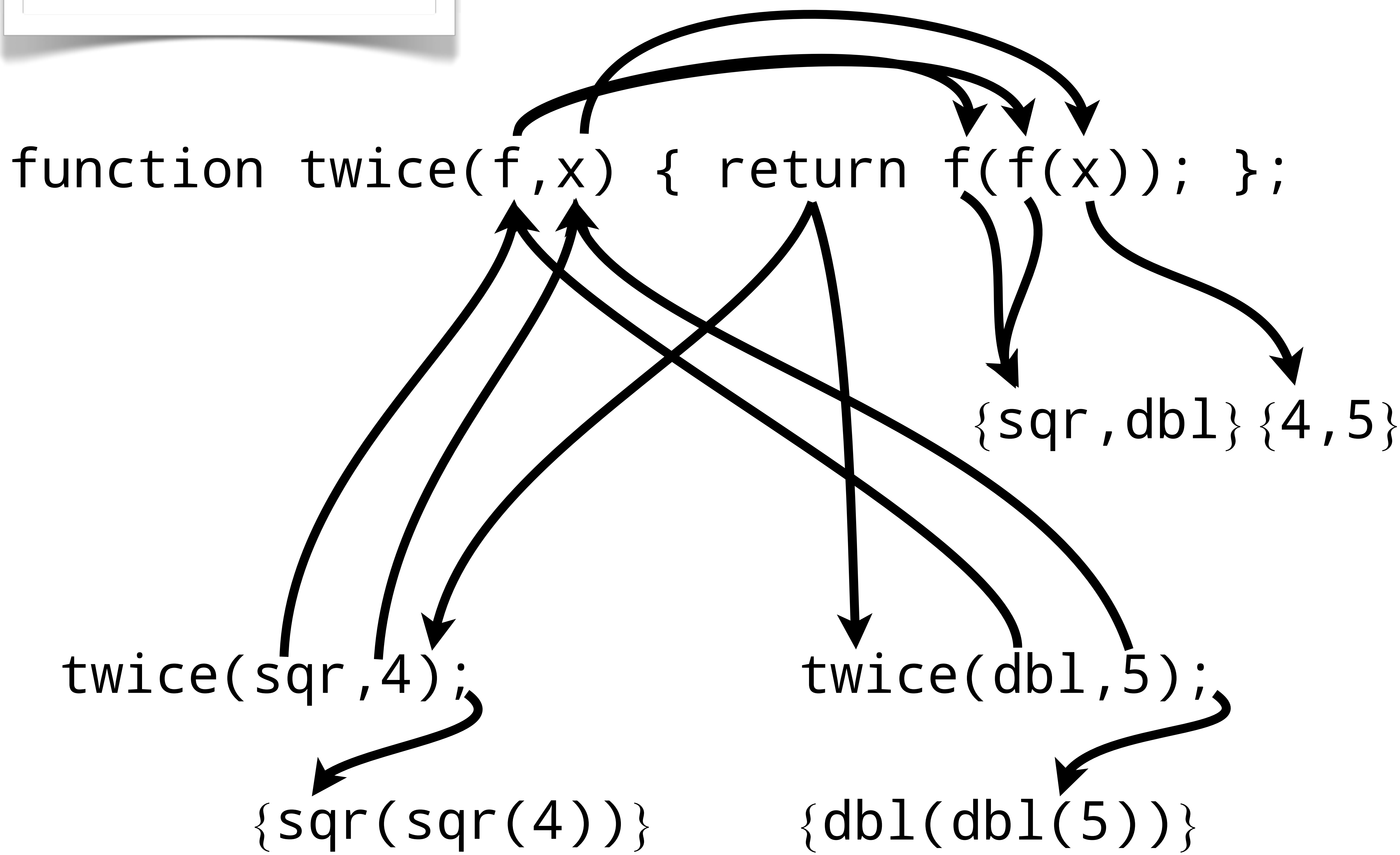

function twice(f,x) { return f(f(x)); };
{sqr,dbl}
{4,5}
twice(sqr,4);
twice(dbl,5);
{sqr(sqr(4))}

0CFA

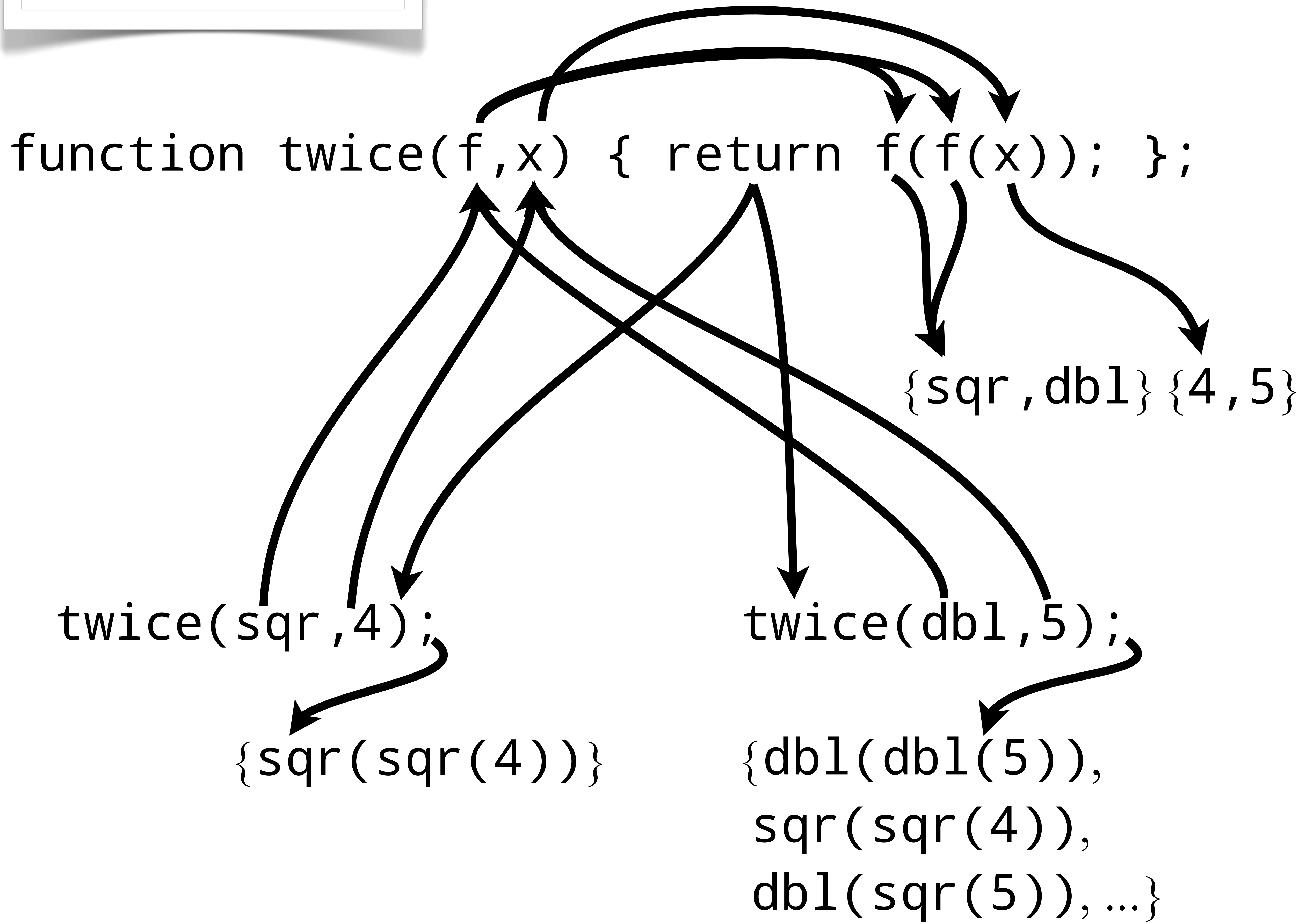


function twice(f,x) { return f(f(x)); };
{sqr,dbl}
{4,5}
twice(sqr,4);
twice(dbl,5);
{sqr(sqr(4))}

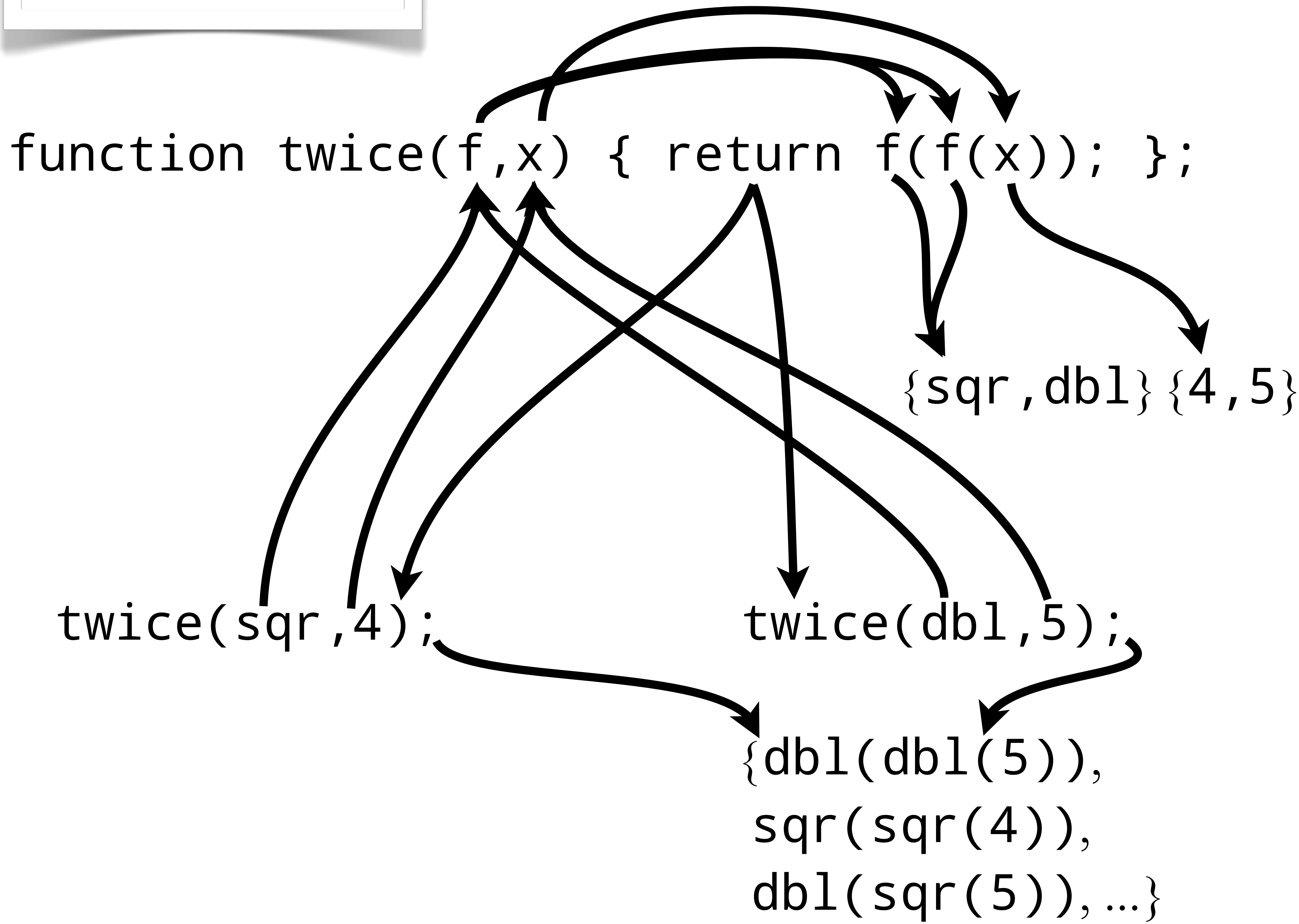
0CFA


function twice(f,x) { return f(f(x)); };
{sqr,dbl}
{4,5}
twice(sqr,4);
twice(dbl,5);
{sqr(sqr(4))}
{dbl(dbl(5))}

0CFA


function twice(f,x) { return f(f(x)); };
{sqr,dbl}
{4,5}
twice(sqr,4);
twice(dbl,5);
{sqr(sqr(4))}
{dbl(dbl(5)),
sqr(sqr(4)),
dbl(sqr(5)), ...}

0CFA

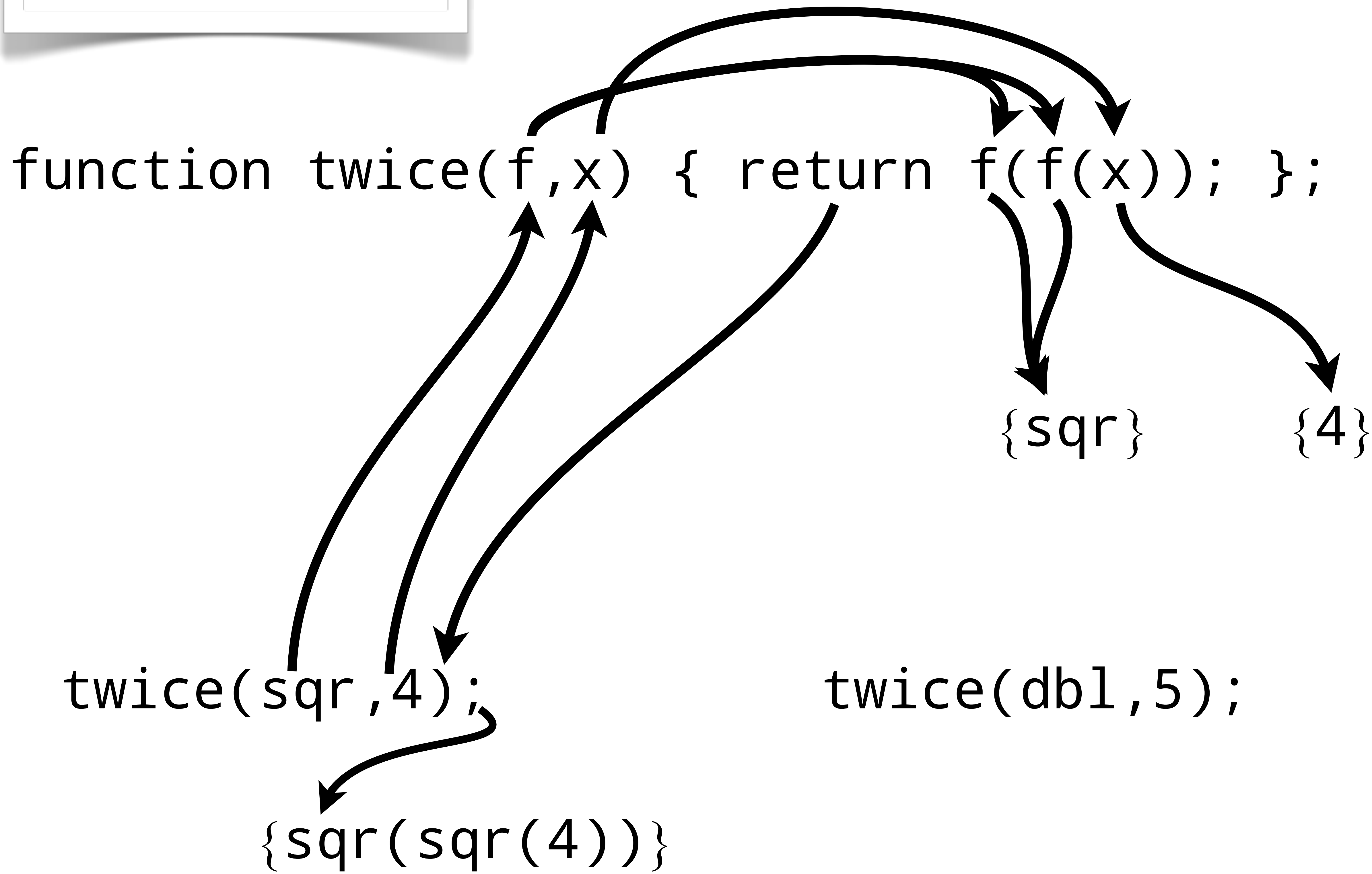

function twice(f,x) { return f(f(x)); };
{sqr,dbl} {4,5}
twice(sqr,4);
twice(dbl,5);
{dbl(dbl(5)),
sqr(sqr(4)),
dbl(sqr(5)), ...}

1CFA

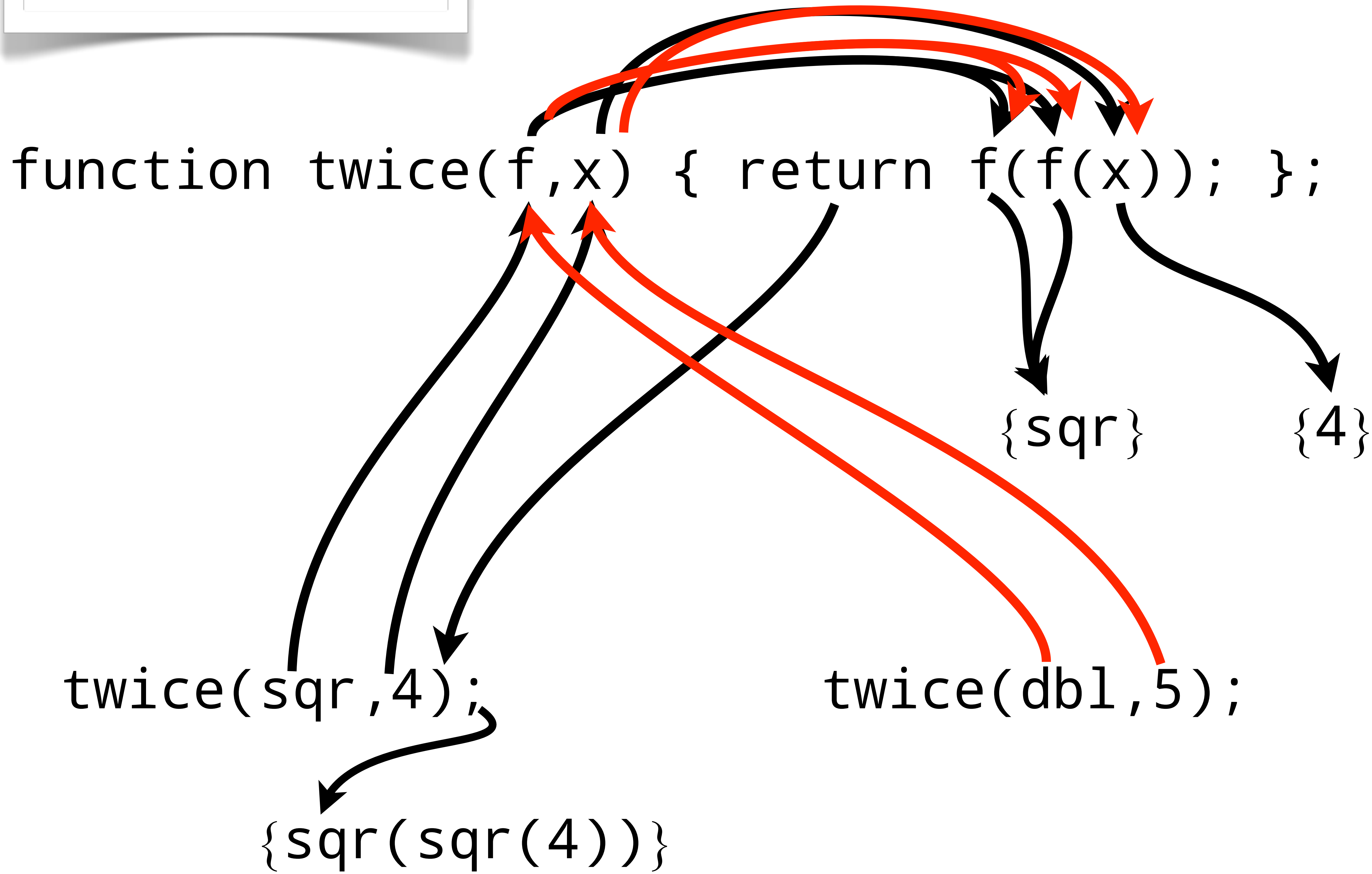
```
function twice(f,x) { return f(f(x)); };
```

```
twice(sqr,4);            twice(dbl,5);
```

1CFA

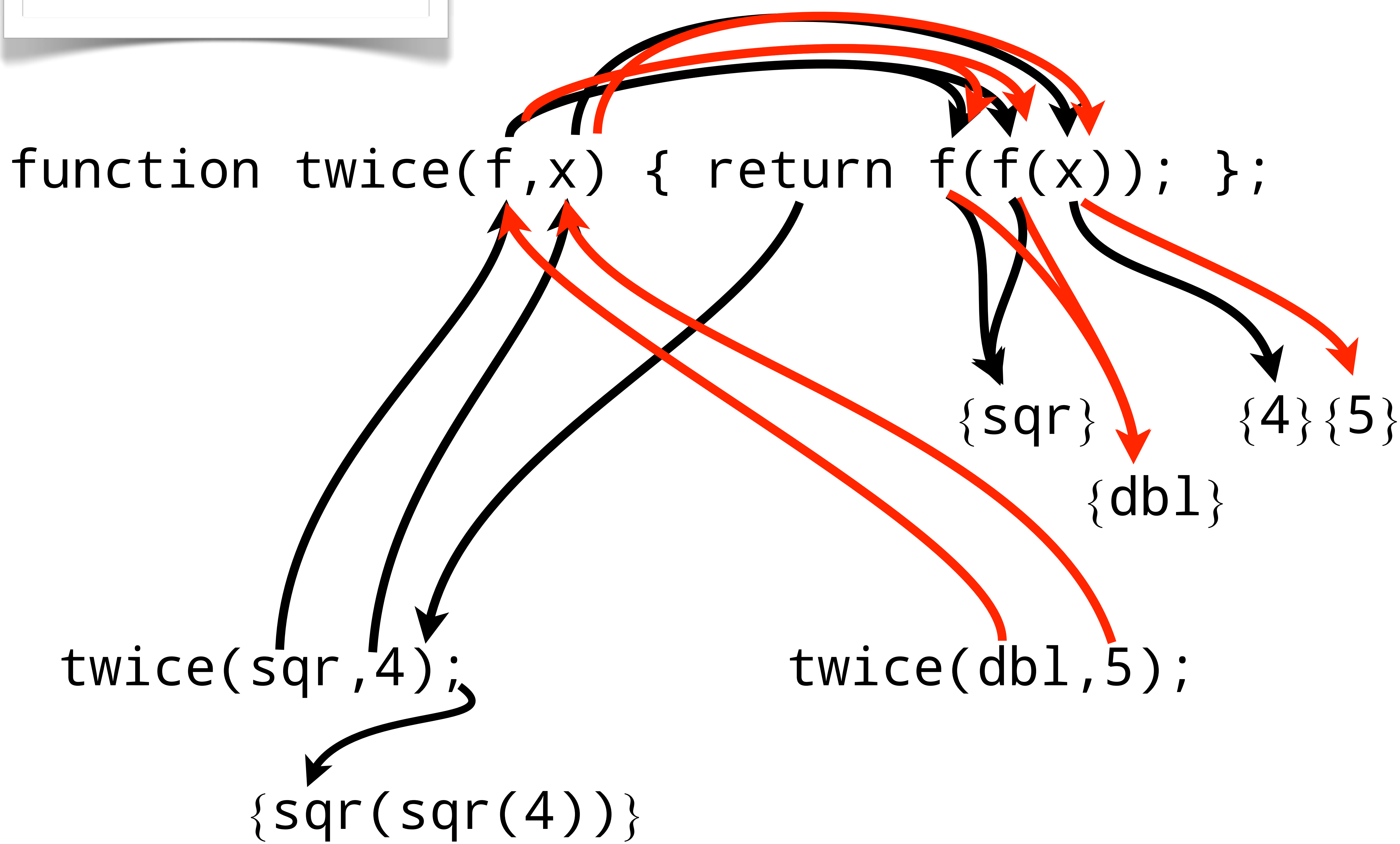

function twice(f,x) { return f(f(x)); };
{sqr}
{4}
twice(sqr,4);
twice(dbl,5);
{sqr(sqr(4))}

1CFA

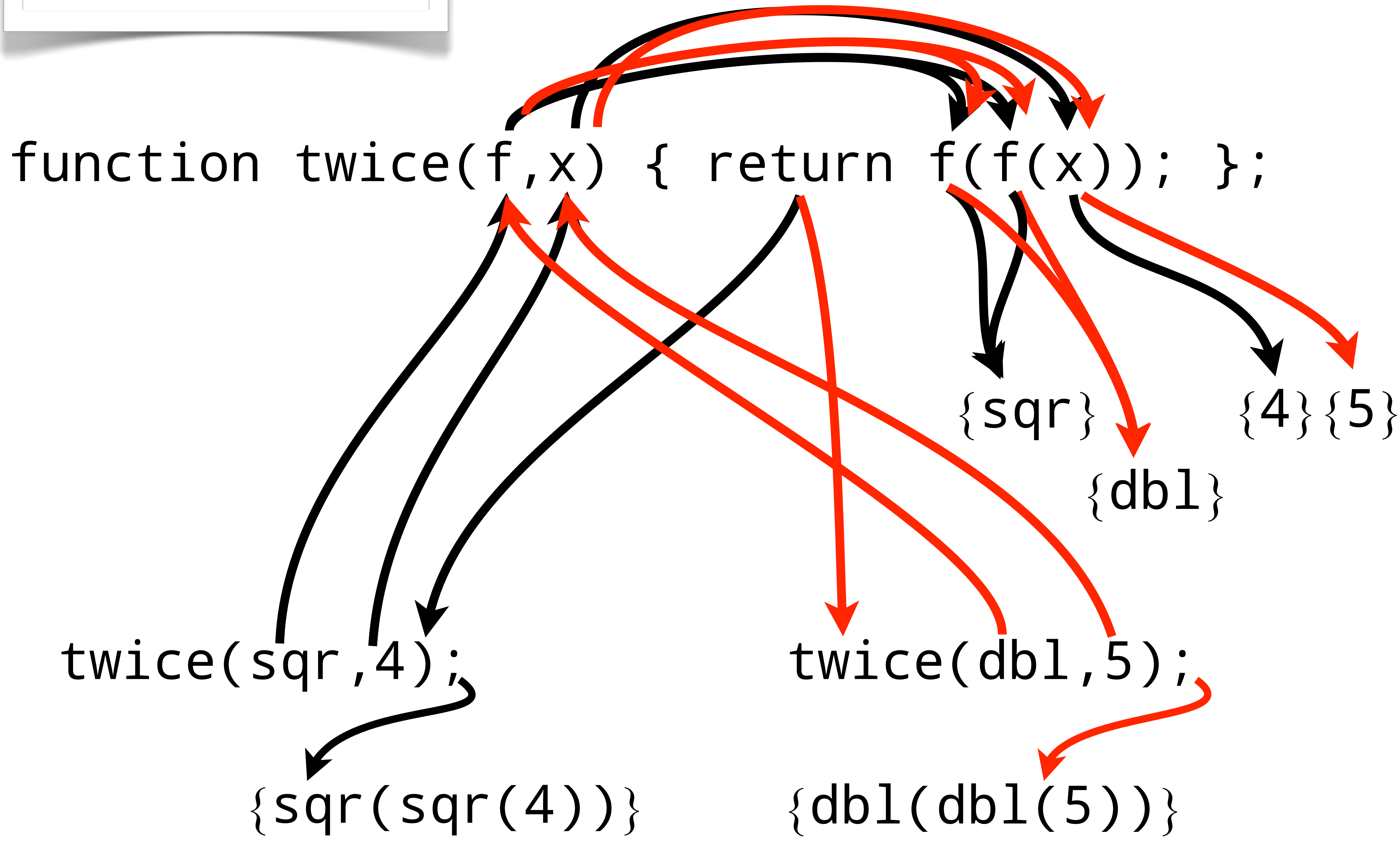


function twice(f,x) { return f(f(x)); };
{sqr}
{4}
twice(sqr,4);
twice(dbl,5);
{sqr(sqr(4))}

1CFA


function twice(f,x) { return f(f(x)); };
{sqr}
{4}{5}
{dbl}
twice(sqr,4);
twice(dbl,5);
{sqr(sqr(4))}

1CFA



function twice(f,x) { return f(f(x)); };
{sqr}
{dbl}
{4}{5}
twice(sqr,4);
twice(dbl,5);
{sqr(sqr(4))}
{dbl(dbl(5))}

Precision

*k*CFA

⋮

1CFA

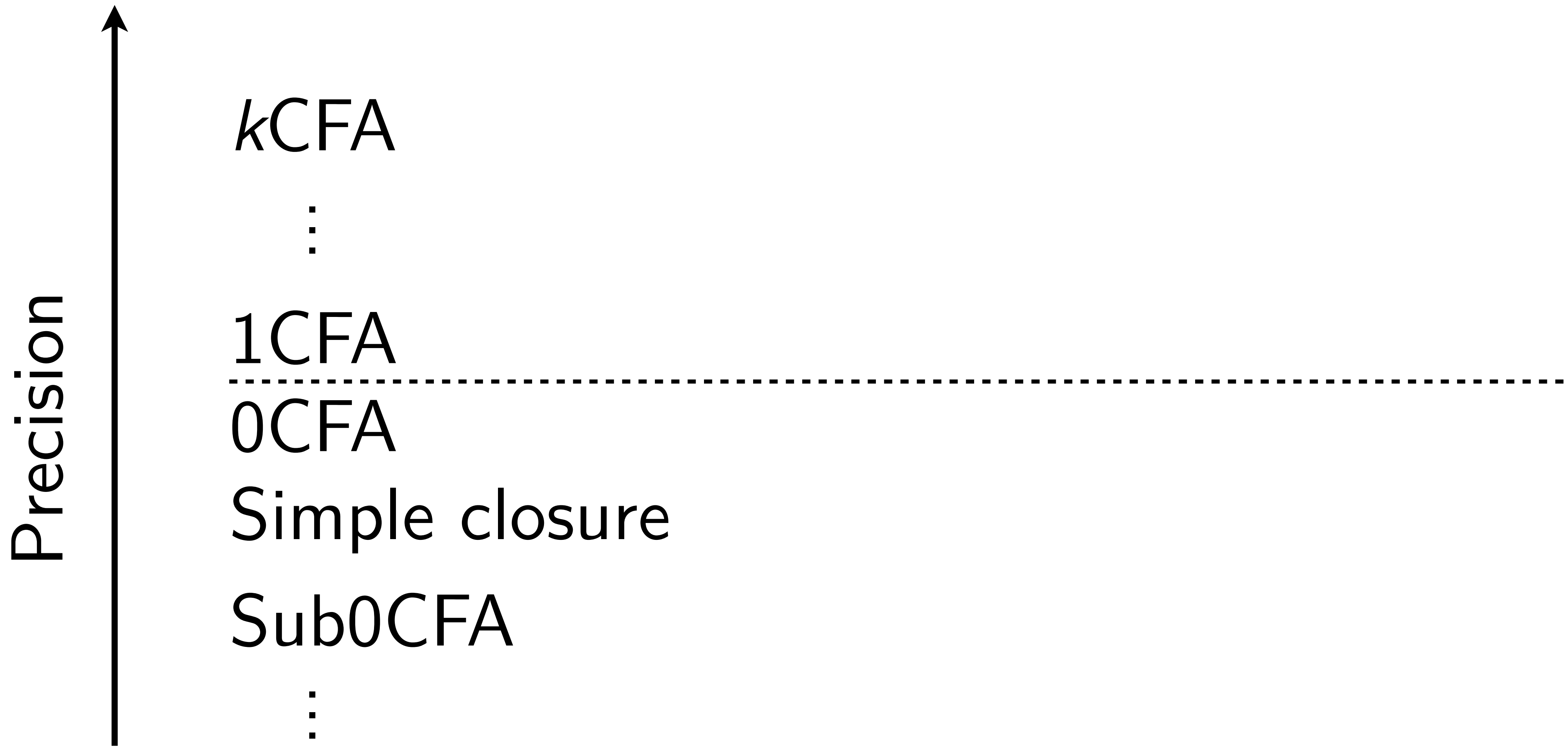
0CFA

Simple closure

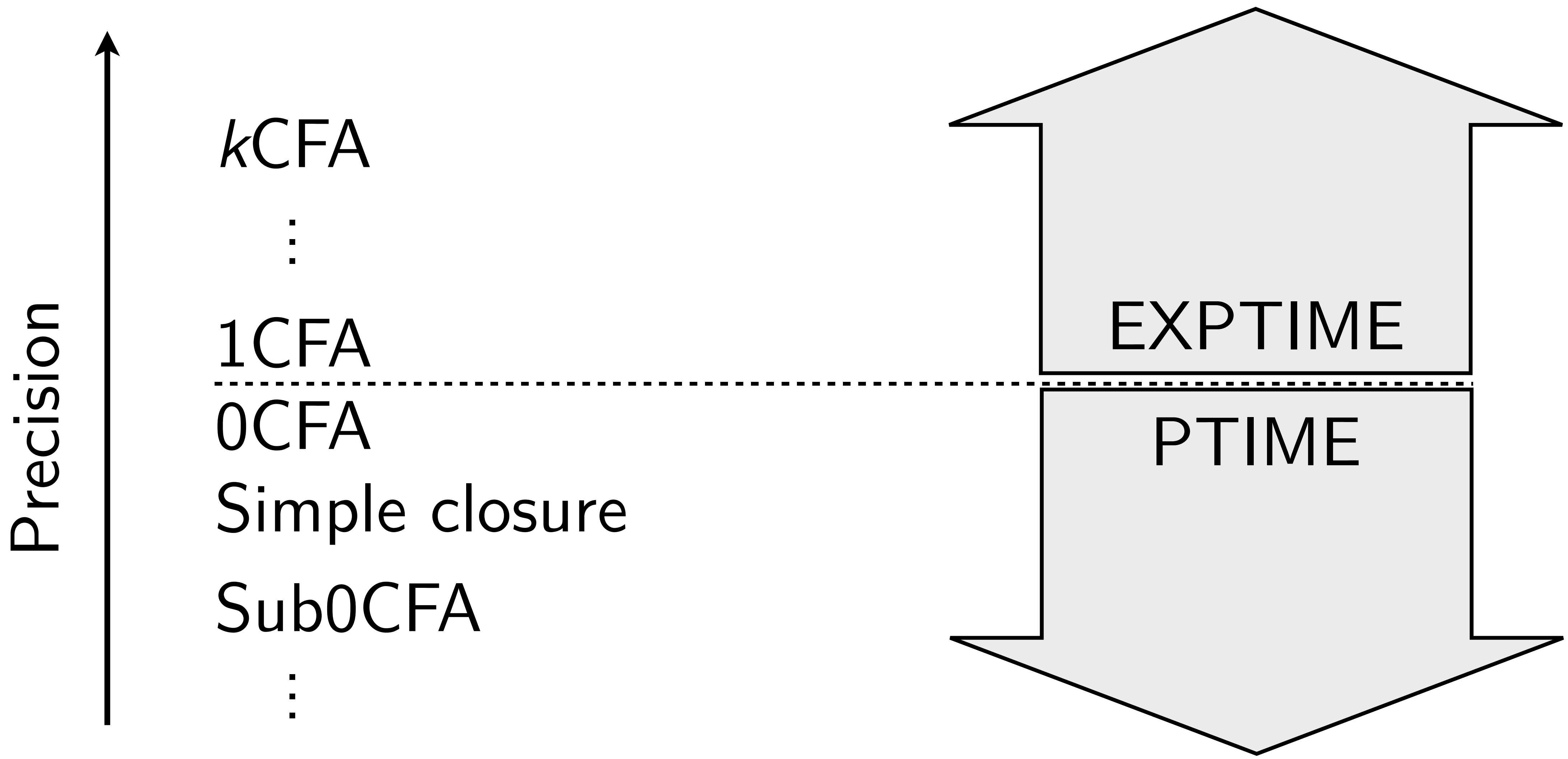
Sub0CFA

⋮

[ICFP'07, SAS'08, ICFP'08]

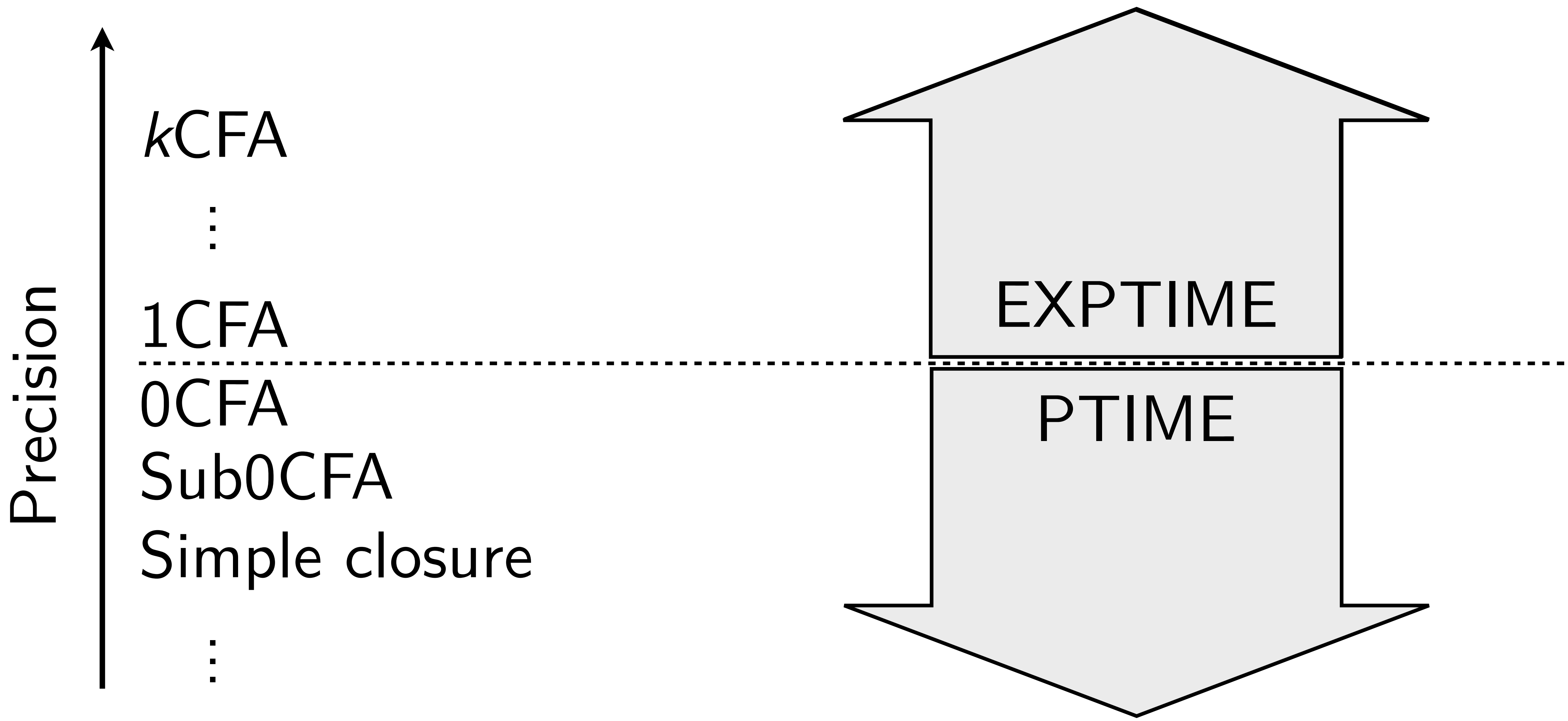


[ICFP'07, SAS'08, ICFP'08]

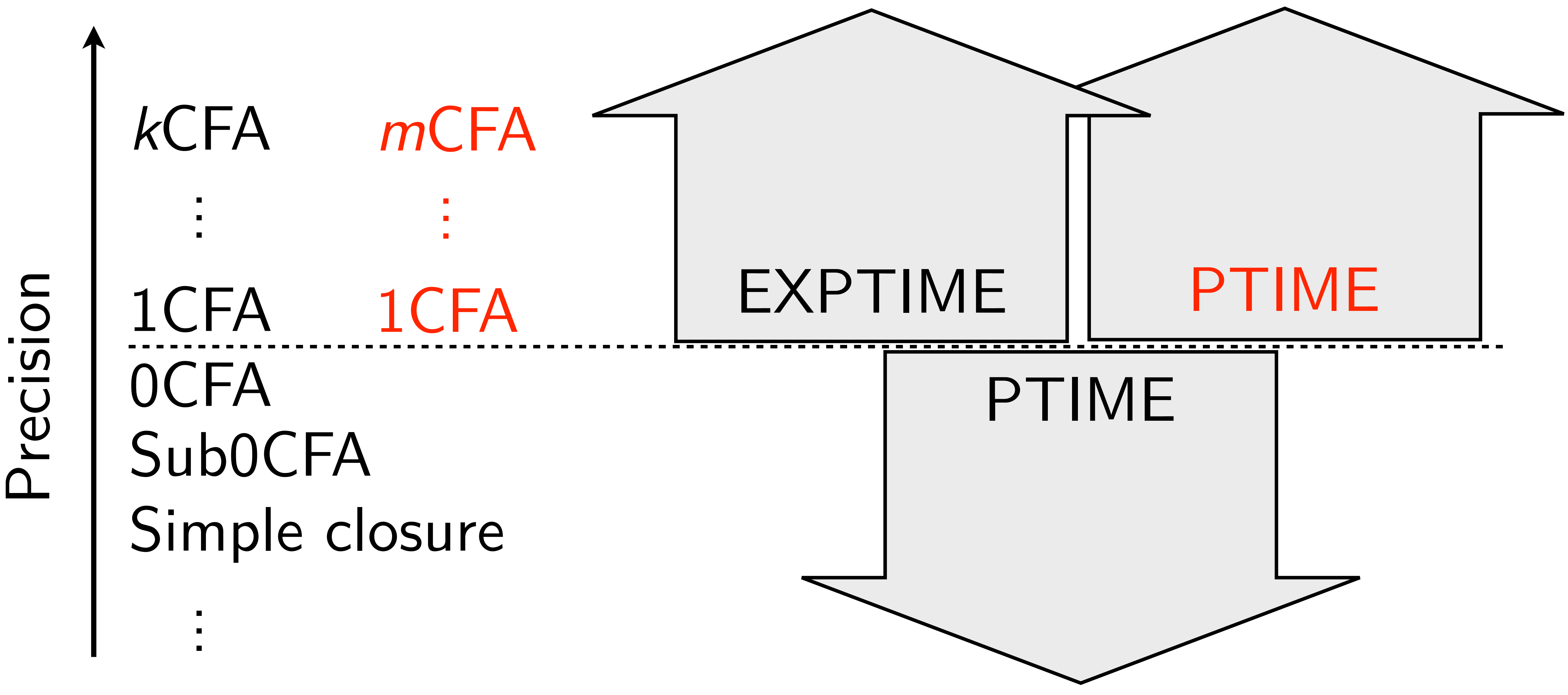


[ICFP'07, SAS'08, ICFP'08]

Key insight: analysis is a kind of evaluation



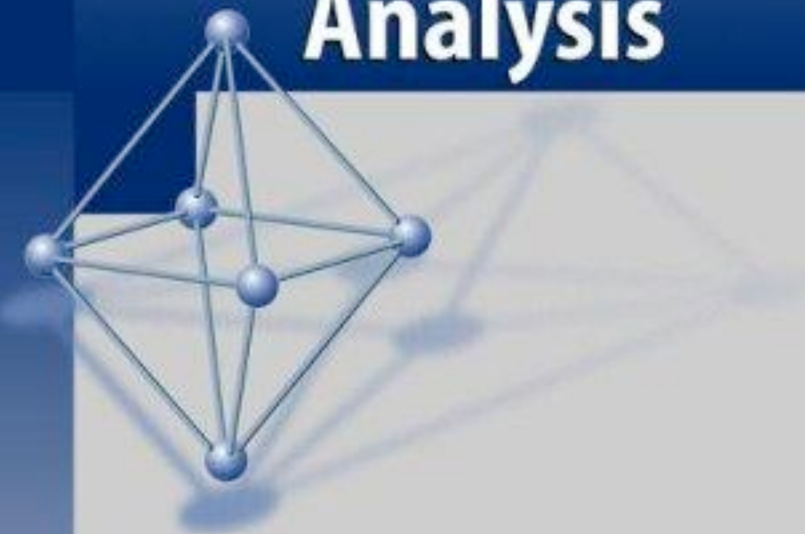
[ICFP'07, SAS'08, ICFP'08]



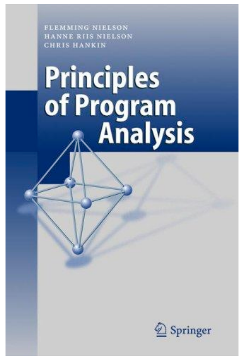
[ICFP'07, SAS'08, ICFP'08, PLDI'10]

FLEMMING NIELSON
HANNE RIIS NIELSON
CHRIS HANKIN

Principles of Program Analysis



Springer



[*con*] $(\widehat{\mathsf{C}}, \widehat{\rho}) \models c^{\ell}$ always

[*var*] $(\widehat{\mathsf{C}}, \widehat{\rho}) \models x^{\ell}$ iff $\widehat{\rho}(x) \subseteq \widehat{\mathsf{C}}(\ell)$

[*fn*] $(\widehat{\mathsf{C}}, \widehat{\rho}) \models (\texttt{fn}\ x \texttt{ => } e_0)^{\ell}$ iff $\{\texttt{fn}\ x \texttt{ => } e_0\} \subseteq \widehat{\mathsf{C}}(\ell)$

[*fun*] $(\widehat{\mathsf{C}}, \widehat{\rho}) \models (\texttt{fun}\ f\ x \texttt{ => } e_0)^{\ell}$ iff $\{\texttt{fun}\ f\ x \texttt{ => } e_0\} \subseteq \widehat{\mathsf{C}}(\ell)$

[*app*] $(\widehat{\mathsf{C}}, \widehat{\rho}) \models (t_1^{\ell_1}\ t_2^{\ell_2})^{\ell}$

$$
\begin{aligned}
\text{iff}\quad & (\widehat{\mathsf{C}}, \widehat{\rho}) \models t_1^{\ell_1} \ \wedge\ (\widehat{\mathsf{C}}, \widehat{\rho}) \models t_2^{\ell_2} \ \wedge \\
& (\forall (\texttt{fn}\ x \texttt{ => } t_0^{\ell_0}) \in \widehat{\mathsf{C}}(\ell_1) : \\
& \qquad (\widehat{\mathsf{C}}, \widehat{\rho}) \models t_0^{\ell_0} \ \wedge \\
& \qquad \widehat{\mathsf{C}}(\ell_2) \subseteq \widehat{\rho}(x) \ \wedge\ \widehat{\mathsf{C}}(\ell_0) \subseteq \widehat{\mathsf{C}}(\ell)) \ \wedge \\
& (\forall (\texttt{fun}\ f\ x \texttt{ => } t_0^{\ell_0}) \in \widehat{\mathsf{C}}(\ell_1) : \\
& \qquad (\widehat{\mathsf{C}}, \widehat{\rho}) \models t_0^{\ell_0} \ \wedge \\
& \qquad \widehat{\mathsf{C}}(\ell_2) \subseteq \widehat{\rho}(x) \ \wedge\ \widehat{\mathsf{C}}(\ell_0) \subseteq \widehat{\mathsf{C}}(\ell) \ \wedge \\
& \qquad \{\texttt{fun}\ f\ x \texttt{ => } t_0^{\ell_0}\} \subseteq \widehat{\rho}(f))
\end{aligned}
$$

[*if*] $(\widehat{\mathsf{C}}, \widehat{\rho}) \models (\texttt{if}\ t_0^{\ell_0}\ \texttt{then}\ t_1^{\ell_1}\ \texttt{else}\ t_2^{\ell_2})^{\ell}$

$$
\begin{aligned}
\text{iff}\quad & (\widehat{\mathsf{C}}, \widehat{\rho}) \models t_0^{\ell_0} \ \wedge \\
& (\widehat{\mathsf{C}}, \widehat{\rho}) \models t_1^{\ell_1} \ \wedge\ (\widehat{\mathsf{C}}, \widehat{\rho}) \models t_2^{\ell_2} \wedge \\
& \widehat{\mathsf{C}}(\ell_1) \subseteq \widehat{\mathsf{C}}(\ell) \ \wedge\ \widehat{\mathsf{C}}(\ell_2) \subseteq \widehat{\mathsf{C}}(\ell)
\end{aligned}
$$

[*let*] $(\widehat{\mathsf{C}}, \widehat{\rho}) \models (\texttt{let}\ x = t_1^{\ell_1}\ \texttt{in}\ t_2^{\ell_2})^{\ell}$

$$
\begin{aligned}
\text{iff}\quad & (\widehat{\mathsf{C}}, \widehat{\rho}) \models t_1^{\ell_1} \ \wedge\ (\widehat{\mathsf{C}}, \widehat{\rho}) \models t_2^{\ell_2} \ \wedge \\
& \widehat{\mathsf{C}}(\ell_1) \subseteq \widehat{\rho}(x) \ \wedge\ \widehat{\mathsf{C}}(\ell_2) \subseteq \widehat{\mathsf{C}}(\ell)
\end{aligned}
$$

[*op*] $(\widehat{\mathsf{C}}, \widehat{\rho}) \models (t_1^{\ell_1}\ op\ t_2^{\ell_2})^{\ell}$ iff $(\widehat{\mathsf{C}}, \widehat{\rho}) \models t_1^{\ell_1} \ \wedge\ (\widehat{\mathsf{C}}, \widehat{\rho}) \models t_2^{\ell_2}$

Table 3.1: Abstract Control Flow Analysis (Subsections 3.1.1 and 3.1.2).

[*var*] $\rho \vdash x^{\ell} \to v^{\ell}$ if $x \in dom(\rho)$ and $v = \rho(x)$

[*fn*] $\rho \vdash (\texttt{fn}\ x \texttt{ => } e_0)^{\ell} \to (\texttt{close}\ (\texttt{fn}\ x \texttt{ => } e_0)\ \texttt{in}\ \rho_0)^{\ell}$
where $\rho_0 = \rho \mid FV(\texttt{fn}\ x \texttt{ => } e_0)$

[*fun*] $\rho \vdash (\texttt{fun}\ f\ x \texttt{ => } e_0)^{\ell} \to (\texttt{close}\ (\texttt{fun}\ f\ x \texttt{ => } e_0)\ \texttt{in}\ \rho_0)^{\ell}$
where $\rho_0 = \rho \mid FV(\texttt{fun}\ f\ x \texttt{ => } e_0)$

[*app*$_1$] $$\frac{\rho \vdash ie_1 \to ie_1'}{\rho \vdash (ie_1\ ie_2)^{\ell} \to (ie_1'\ ie_2)^{\ell}}$$

[*app*$_2$] $$\frac{\rho \vdash ie_2 \to ie_2'}{\rho \vdash (v_1^{\ell_1}\ ie_2)^{\ell} \to (v_1^{\ell_1}\ ie_2')^{\ell}}$$

[*app*$_{fn}$] $\rho \vdash ((\texttt{close}\ (\texttt{fn}\ x \texttt{ => } e_1)\ \texttt{in}\ \rho_1)^{\ell_1}\ v_2^{\ell_2})^{\ell} \to$
$(\texttt{bind}\ \rho_1[x \mapsto v_2]\ \texttt{in}\ e_1)^{\ell}$

[*app*$_{fun}$] $\rho \vdash ((\texttt{close}\ (\texttt{fun}\ f\ x \texttt{ => } e_1)\ \texttt{in}\ \rho_1)^{\ell_1}\ v_2^{\ell_2})^{\ell} \to$
$(\texttt{bind}\ \rho_2[x \mapsto v_2]\ \texttt{in}\ e_1)^{\ell}$
where $\rho_2 = \rho_1[f \mapsto \texttt{close}\ (\texttt{fun}\ f\ x \texttt{ => } e_1)\ \texttt{in}\ \rho_1]$

[*bind*$_1$] $$\frac{\rho_1 \vdash ie_1 \to ie_1'}{\rho \vdash (\texttt{bind}\ \rho_1\ \texttt{in}\ ie_1)^{\ell} \to (\texttt{bind}\ \rho_1\ \texttt{in}\ ie_1')^{\ell}}$$

[*bind*$_2$] $\rho \vdash (\texttt{bind}\ \rho_1\ \texttt{in}\ v_1^{\ell_1})^{\ell} \to v_1^{\ell}$

[*if*$_1$] $$\frac{\rho \vdash ie_0 \to ie_0'}{\rho \vdash (\texttt{if}\ ie_0\ \texttt{then}\ e_1\ \texttt{else}\ e_2)^{\ell} \to (\texttt{if}\ ie_0'\ \texttt{then}\ e_1\ \texttt{else}\ e_2)^{\ell}}$$

[*if*$_2$] $\rho \vdash (\texttt{if}\ \texttt{true}^{\ell_0}\ \texttt{then}\ t_1^{\ell_1}\ \texttt{else}\ t_2^{\ell_2})^{\ell} \to t_1^{\ell}$

[*if*$_3$] $\rho \vdash (\texttt{if}\ \texttt{false}^{\ell_0}\ \texttt{then}\ t_1^{\ell_1}\ \texttt{else}\ t_2^{\ell_2})^{\ell} \to t_2^{\ell}$

[*let*$_1$] $$\frac{\rho \vdash ie_1 \to ie_1'}{\rho \vdash (\texttt{let}\ x = ie_1\ \texttt{in}\ e_2)^{\ell} \to (\texttt{let}\ x = ie_1'\ \texttt{in}\ e_2)^{\ell}}$$

[*let*$_2$] $\rho \vdash (\texttt{let}\ x = v^{\ell_1}\ \texttt{in}\ e_2)^{\ell} \to (\texttt{bind}\ \rho_0[x \mapsto v]\ \texttt{in}\ e_2)^{\ell}$
where $\rho_0 = \rho \mid FV(e_2)$

[*op*$_1$] $$\frac{\rho \vdash ie_1 \to ie_1'}{\rho \vdash (ie_1\ op\ ie_2)^{\ell} \to (ie_1'\ op\ ie_2)^{\ell}}$$

[*op*$_2$] $$\frac{\rho \vdash ie_2 \to ie_2'}{\rho \vdash (v_1^{\ell_1}\ op\ ie_2)^{\ell} \to (v_1^{\ell_1}\ op\ ie_2')^{\ell}}$$

[*op*$_3$] $\rho \vdash (v_1^{\ell_1}\ op\ v_2^{\ell_2})^{\ell} \to v^{\ell}$ if $v = v_1\ \mathbf{op}\ v_2$

Table 3.3: The Structural Operational Semantics of FUN (part 2).

[var] $\rho \vdash x^\ell \to v^\ell$ if $x \in dom(\rho)$ and $v = \rho(x)$

[fn] $\rho \vdash (\texttt{fn}\ x\ \texttt{=>}\ e_0)^\ell \to (\texttt{close}\ (\texttt{fn}\ x\ \texttt{=>}\ e_0)\ \texttt{in}\ \rho_0)^\ell$
where $\rho_0 = \rho \mid FV(\texttt{fn}\ x\ \texttt{=>}\ e_0)$

[fun] $\rho \vdash (\texttt{fun}\ f\ x\ \texttt{=>}\ e_0)^\ell \to (\texttt{close}\ (\texttt{fun}\ f\ x\ \texttt{=>}\ e_0)\ \texttt{in}\ \rho_0)^\ell$
where $\rho_0 = \rho \mid FV(\texttt{fun}\ f\ x\ \texttt{=>}\ e_0)$

[app_1] $$\frac{\rho \vdash ie_1 \to ie_1'}{\rho \vdash (ie_1\ ie_2)^\ell \to (ie_1'\ ie_2)^\ell}$$

[app_2] $$\frac{\rho \vdash ie_2 \to ie_2'}{\rho \vdash (v_1^{\ell_1}\ ie_2)^\ell \to (v_1^{\ell_1}\ ie_2')^\ell}$$

[app_{fn}] $\rho \vdash ((\texttt{close}\ (\texttt{fn}\ x\ \texttt{=>}\ e_1)\ \texttt{in}\ \rho_1)^{\ell_1}\ v_2^{\ell_2})^\ell \to$
$(\texttt{bind}\ \rho_1[x \mapsto v_2]\ \texttt{in}\ e_1)^\ell$

[app_{fun}] $\rho \vdash ((\texttt{close}\ (\texttt{fun}\ f\ x\ \texttt{=>}\ e_1)\ \texttt{in}\ \rho_1)^{\ell_1}\ v_2^{\ell_2})^\ell \to$
$(\texttt{bind}\ \rho_2[x \mapsto v_2]\ \texttt{in}\ e_1)^\ell$
where $\rho_2 = \rho_1[f \mapsto \texttt{close}\ (\texttt{fun}\ f\ x\ \texttt{=>}\ e_1)\ \texttt{in}\ \rho_1]$

[$bind_1$] $$\frac{\rho_1 \vdash ie_1 \to ie_1'}{\rho \vdash (\texttt{bind}\ \rho_1\ \texttt{in}\ ie_1)^\ell \to (\texttt{bind}\ \rho_1\ \texttt{in}\ ie_1')^\ell}$$

[$bind_2$] $\rho \vdash (\texttt{bind}\ \rho_1\ \texttt{in}\ v_1^{\ell_1})^\ell \to v_1^\ell$

[if_1] $$\frac{\rho \vdash ie_0 \to ie_0'}{\rho \vdash (\texttt{if}\ ie_0\ \texttt{then}\ e_1\ \texttt{else}\ e_2)^\ell \to (\texttt{if}\ ie_0'\ \texttt{then}\ e_1\ \texttt{else}\ e_2)^\ell}$$

[if_2] $\rho \vdash (\texttt{if}\ \texttt{true}^{\ell_0}\ \texttt{then}\ t_1^{\ell_1}\ \texttt{else}\ t_2^{\ell_2})^\ell \to t_1^\ell$

[if_3] $\rho \vdash (\texttt{if}\ \texttt{false}^{\ell_0}\ \texttt{then}\ t_1^{\ell_1}\ \texttt{else}\ t_2^{\ell_2})^\ell \to t_2^\ell$

[let_1] $$\frac{\rho \vdash ie_1 \to ie_1'}{\rho \vdash (\texttt{let}\ x\ \texttt{=}\ ie_1\ \texttt{in}\ e_2)^\ell \to (\texttt{let}\ x\ \texttt{=}\ ie_1'\ \texttt{in}\ e_2)^\ell}$$

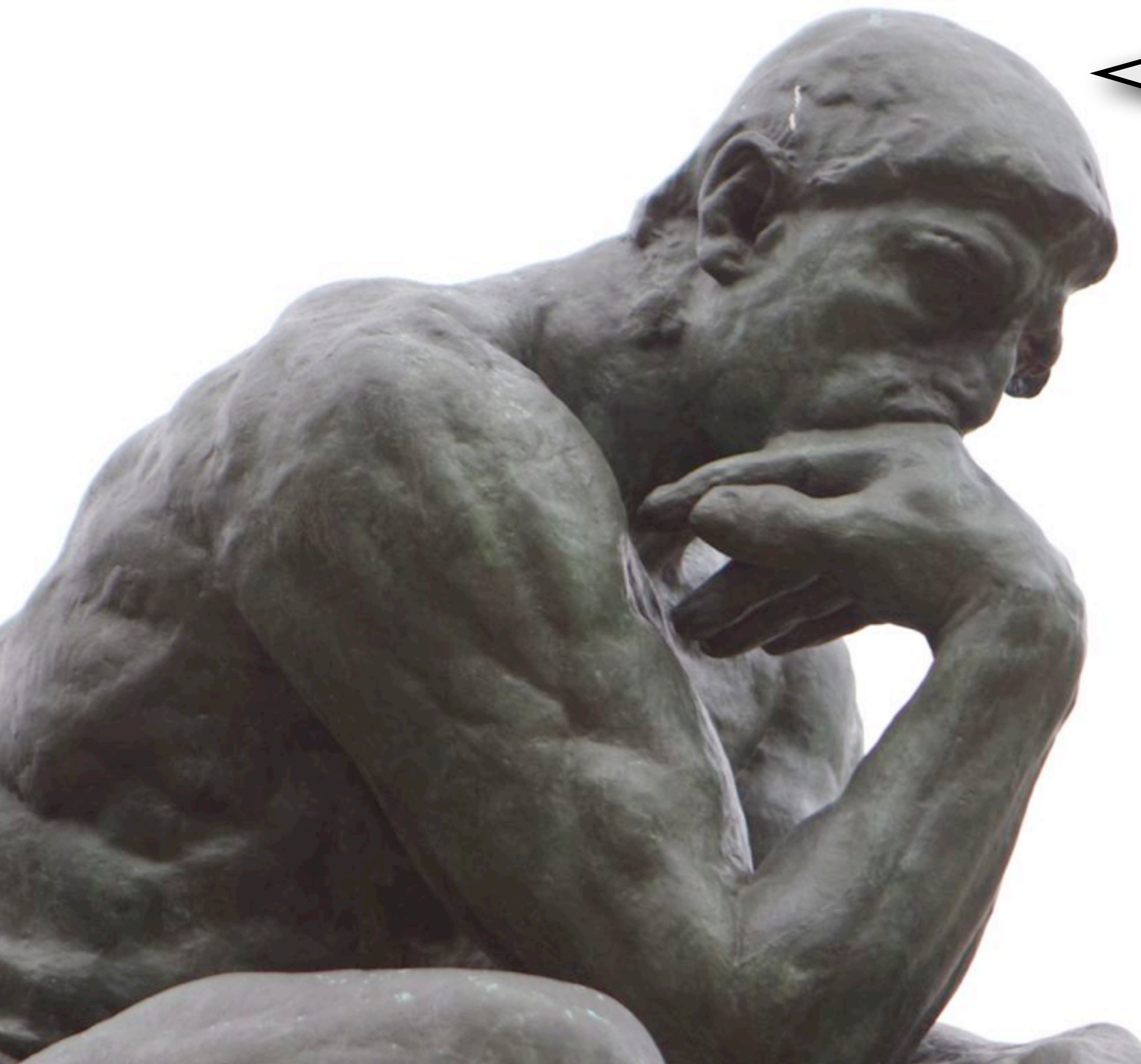
[let_2] $\rho \vdash (\texttt{let}\ x\ \texttt{=}\ v^{\ell_1}\ \texttt{in}\ e_2)^\ell \to (\texttt{bind}\ \rho_0[x \mapsto v]\ \texttt{in}\ e_2)^\ell$
where $\rho_0 = \rho \mid FV(e_2)$

[op_1] $$\frac{\rho \vdash ie_1 \to ie_1'}{\rho \vdash (ie_1\ op\ ie_2)^\ell \to (ie_1'\ op\ ie_2)^\ell}$$

[op_2] $$\frac{\rho \vdash ie_2 \to ie_2'}{\rho \vdash (v_1^{\ell_1}\ op\ ie_2)^\ell \to (v_1^{\ell_1}\ op\ ie_2')^\ell}$$

[op_3] $\rho \vdash (v_1^{\ell_1}\ op\ v_2^{\ell_2})^\ell \to v^\ell$ if $v = v_1\ \mathbf{op}\ v_2$

Table 3.3: The Structural Operational Semantics of FUN (part 2).



$[var]$ $\quad \rho \vdash x^{\ell} \rightarrow v^{\ell}$ if $x \in dom(\rho)$ and $v = \rho(x)$

$[fn]$ $\quad \rho \vdash (\texttt{fn}\ x\ \texttt{=>}\ e_0)^{\ell} \rightarrow (\texttt{close}\ (\texttt{fn}\ x\ \texttt{=>}\ e_0)\ \texttt{in}\ \rho_0)^{\ell}$ where $\rho_0 = \rho \mid FV(\texttt{fn}\ x\ \texttt{=>}\ e_0)$

$[fun]$ $\quad \rho \vdash (\texttt{fun}\ f\ x\ \texttt{=>}\ e_0)^{\ell} \rightarrow (\texttt{close}\ (\texttt{fun}\ f\ x\ \texttt{=>}\ e_0)\ \texttt{in}\ \rho_0)^{\ell}$ where $\rho_0 = \rho \mid FV(\texttt{fun}\ f\ x\ \texttt{=>}\ e_0)$

$[app_1]$ $\quad \dfrac{\rho \vdash ie_1 \rightarrow ie_1'}{\rho \vdash (ie_1\ ie_2)^{\ell} \rightarrow (ie_1'\ ie_2)^{\ell}}$

$[app_2]$ $\quad \dfrac{\rho \vdash ie_2 \rightarrow ie_2'}{\rho \vdash (v_1^{\ell_1}\ ie_2)^{\ell} \rightarrow (v_1^{\ell_1}\ ie_2')^{\ell}}$

$[app_{fn}]$ $\quad \rho \vdash ((\texttt{close}\ (\texttt{fn}\ x\ \texttt{=>}\ e_1)\ \texttt{in}\ \rho_1)^{\ell_1}\ v_2^{\ell_2})^{\ell} \rightarrow (\texttt{bind}\ \rho_1[x \mapsto v_2]\ \texttt{in}\ e_1)^{\ell}$

$[app_{fun}]$ $\quad \rho \vdash ((\texttt{close}\ (\texttt{fun}\ f\ x\ \texttt{=>}\ e_1)\ \texttt{in}\ \rho_1)^{\ell_1}\ v_2^{\ell_2})^{\ell} \rightarrow (\texttt{bind}\ \rho_2[x \mapsto v_2]\ \texttt{in}\ e_1)^{\ell}$ where $\rho_2 = \rho_1[f \mapsto \texttt{close}\ (\texttt{fun}\ f\ x\ \texttt{=>}\ e_1)\ \texttt{in}\ \rho_1]$

$[bind_1]$ $\quad \dfrac{\rho_1 \vdash ie_1 \rightarrow ie_1'}{\rho \vdash (\texttt{bind}\ \rho_1\ \texttt{in}\ ie_1)^{\ell} \rightarrow (\texttt{bind}\ \rho_1\ \texttt{in}\ ie_1')^{\ell}}$

$[bind_2]$ $\quad \rho \vdash (\texttt{bind}\ \rho_1\ \texttt{in}\ v_1^{\ell_1})^{\ell} \rightarrow v_1^{\ell}$

$[if_1]$ $\quad \dfrac{\rho \vdash ie_0 \rightarrow ie_0'}{\rho \vdash (\texttt{if}\ ie_0\ \texttt{then}\ e_1\ \texttt{else}\ e_2)^{\ell} \rightarrow (\texttt{if}\ ie_0'\ \texttt{then}\ e_1\ \texttt{else}\ e_2)^{\ell}}$

$[if_2]$ $\quad \rho \vdash (\texttt{if}\ \texttt{true}^{\ell_0}\ \texttt{then}\ t_1^{\ell_1}\ \texttt{else}\ t_2^{\ell_2})^{\ell} \rightarrow t_1^{\ell}$

$[if_3]$ $\quad \rho \vdash (\texttt{if}\ \texttt{false}^{\ell_0}\ \texttt{then}\ t_1^{\ell_1}\ \texttt{else}\ t_2^{\ell_2})^{\ell} \rightarrow t_2^{\ell}$

$[let_1]$ $\quad \dfrac{\rho \vdash ie_1 \rightarrow ie_1'}{\rho \vdash (\texttt{let}\ x\ \texttt{=}\ ie_1\ \texttt{in}\ e_2)^{\ell} \rightarrow (\texttt{let}\ x\ \texttt{=}\ ie_1'\ \texttt{in}\ e_2)^{\ell}}$

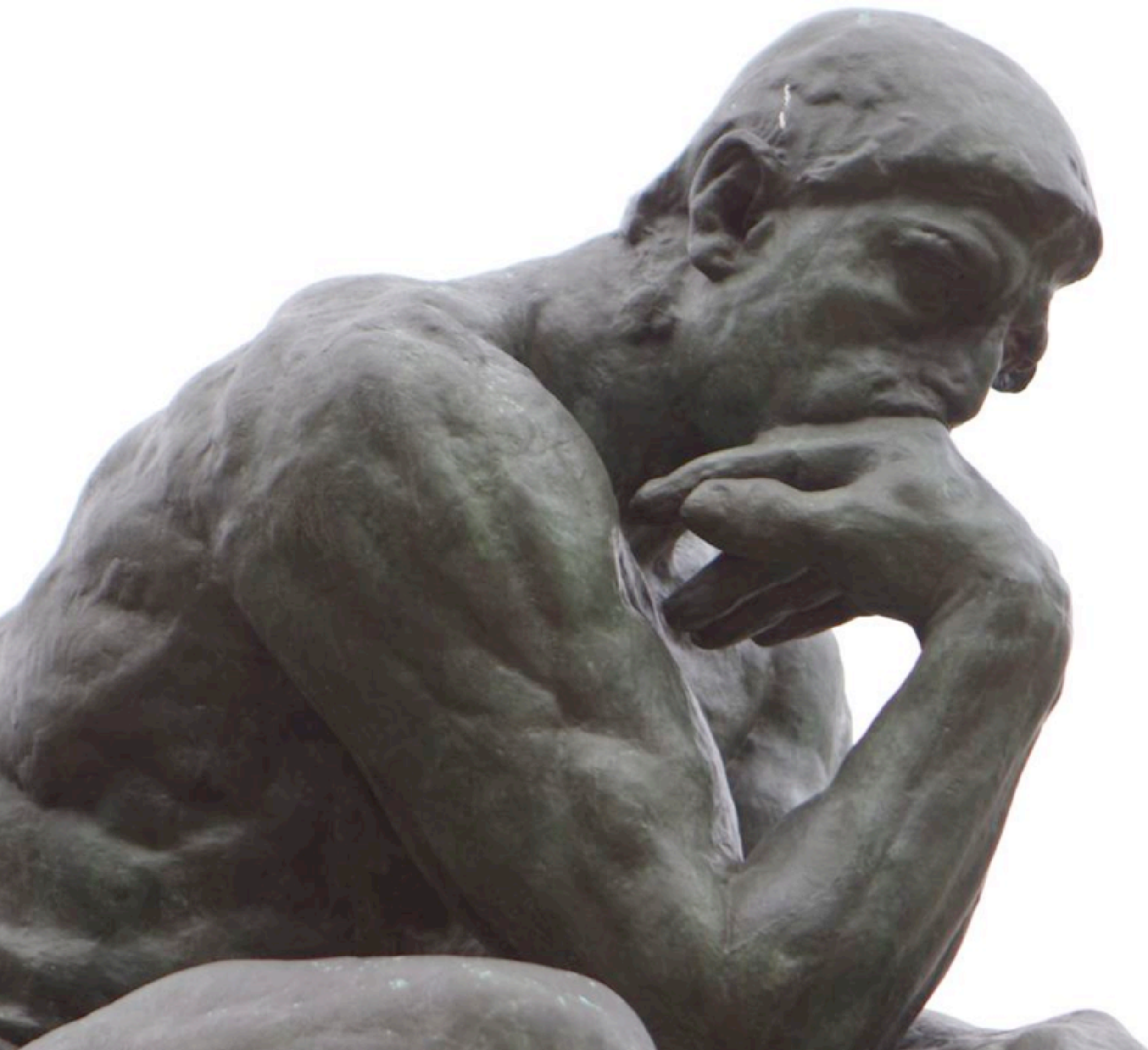
$[let_2]$ $\quad \rho \vdash (\texttt{let}\ x\ \texttt{=}\ v^{\ell_1}\ \texttt{in}\ e_2)^{\ell} \rightarrow (\texttt{bind}\ \rho_0[x \mapsto v]\ \texttt{in}\ e_2)^{\ell}$ where $\rho_0 = \rho \mid FV(e_2)$

$[op_1]$ $\quad \dfrac{\rho \vdash ie_1 \rightarrow ie_1'}{\rho \vdash (ie_1\ op\ ie_2)^{\ell} \rightarrow (ie_1'\ op\ ie_2)^{\ell}}$

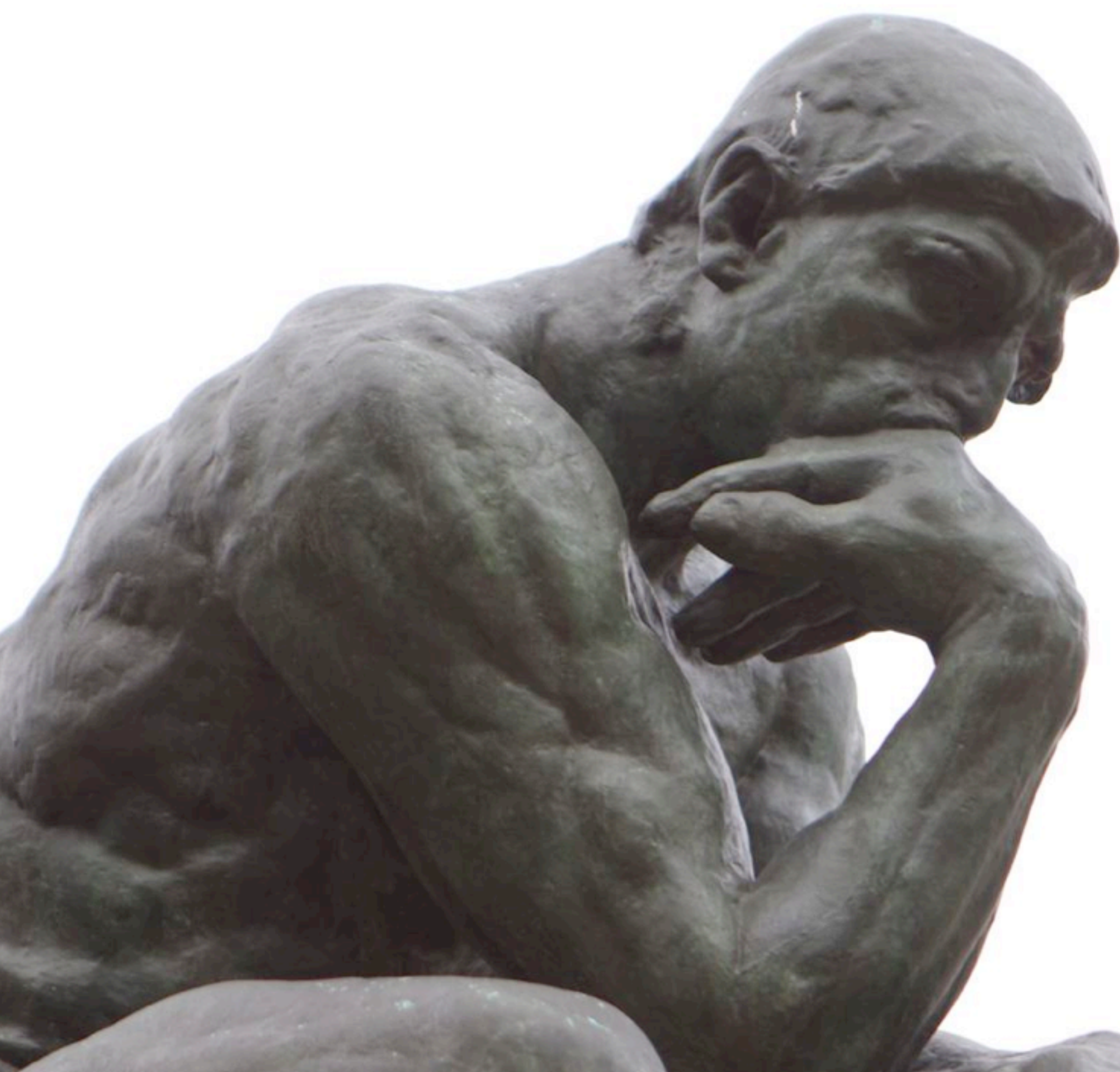
$[op_2]$ $\quad \dfrac{\rho \vdash ie_2 \rightarrow ie_2'}{\rho \vdash (v_1^{\ell_1}\ op\ ie_2)^{\ell} \rightarrow (v_1^{\ell_1}\ op\ ie_2')^{\ell}}$

$[op_3]$ $\quad \rho \vdash (v_1^{\ell_1}\ op\ v_2^{\ell_2})^{\ell} \rightarrow v^{\ell}$ $\qquad$ if $v = v_1\ \mathbf{op}\ v_2$

Table 3.3: The Structural Operational Semantics of FUN (part 2).



!



$$[con] \quad (\widehat{\mathsf{C}}, \widehat{\rho}) \models c^\ell \text{ always}$$

$$[var] \quad (\widehat{\mathsf{C}}, \widehat{\rho}) \models x^\ell \text{ iff } \widehat{\rho}(x) \subseteq \widehat{\mathsf{C}}(\ell)$$

$$[fn] \quad (\widehat{\mathsf{C}}, \widehat{\rho}) \models (\texttt{fn } x \texttt{ => } e_0)^\ell \text{ iff } \{\texttt{fn } x \texttt{ => } e_0\} \subseteq \widehat{\mathsf{C}}(\ell)$$

$$[fun] \quad (\widehat{\mathsf{C}}, \widehat{\rho}) \models (\texttt{fun } f\ x \texttt{ => } e_0)^\ell \text{ iff } \{\texttt{fun } f\ x \texttt{ => } e_0\} \subseteq \widehat{\mathsf{C}}(\ell)$$

$$\begin{aligned}
[app] \quad & (\widehat{\mathsf{C}}, \widehat{\rho}) \models (t_1^{\ell_1}\ t_2^{\ell_2})^\ell \\
& \text{iff} \quad (\widehat{\mathsf{C}}, \widehat{\rho}) \models t_1^{\ell_1} \ \wedge\ (\widehat{\mathsf{C}}, \widehat{\rho}) \models t_2^{\ell_2} \ \wedge \\
& \qquad (\forall (\texttt{fn } x \texttt{ => } t_0^{\ell_0}) \in \widehat{\mathsf{C}}(\ell_1) : \\
& \qquad\qquad (\widehat{\mathsf{C}}, \widehat{\rho}) \models t_0^{\ell_0} \ \wedge \\
& \qquad\qquad \widehat{\mathsf{C}}(\ell_2) \subseteq \widehat{\rho}(x) \ \wedge\ \widehat{\mathsf{C}}(\ell_0) \subseteq \widehat{\mathsf{C}}(\ell)) \ \wedge \\
& \qquad (\forall (\texttt{fun } f\ x \texttt{ => } t_0^{\ell_0}) \in \widehat{\mathsf{C}}(\ell_1) : \\
& \qquad\qquad (\widehat{\mathsf{C}}, \widehat{\rho}) \models t_0^{\ell_0} \ \wedge \\
& \qquad\qquad \widehat{\mathsf{C}}(\ell_2) \subseteq \widehat{\rho}(x) \ \wedge\ \widehat{\mathsf{C}}(\ell_0) \subseteq \widehat{\mathsf{C}}(\ell) \ \wedge \\
& \qquad\qquad \{\texttt{fun } f\ x \texttt{ => } t_0^{\ell_0}\} \subseteq \widehat{\rho}(f))
\end{aligned}$$

$$\begin{aligned}
[if] \quad & (\widehat{\mathsf{C}}, \widehat{\rho}) \models (\texttt{if } t_0^{\ell_0} \texttt{ then } t_1^{\ell_1} \texttt{ else } t_2^{\ell_2})^\ell \\
& \text{iff} \quad (\widehat{\mathsf{C}}, \widehat{\rho}) \models t_0^{\ell_0} \ \wedge \\
& \qquad (\widehat{\mathsf{C}}, \widehat{\rho}) \models t_1^{\ell_1} \ \wedge\ (\widehat{\mathsf{C}}, \widehat{\rho}) \models t_2^{\ell_2} \wedge \\
& \qquad \widehat{\mathsf{C}}(\ell_1) \subseteq \widehat{\mathsf{C}}(\ell) \ \wedge\ \widehat{\mathsf{C}}(\ell_2) \subseteq \widehat{\mathsf{C}}(\ell)
\end{aligned}$$

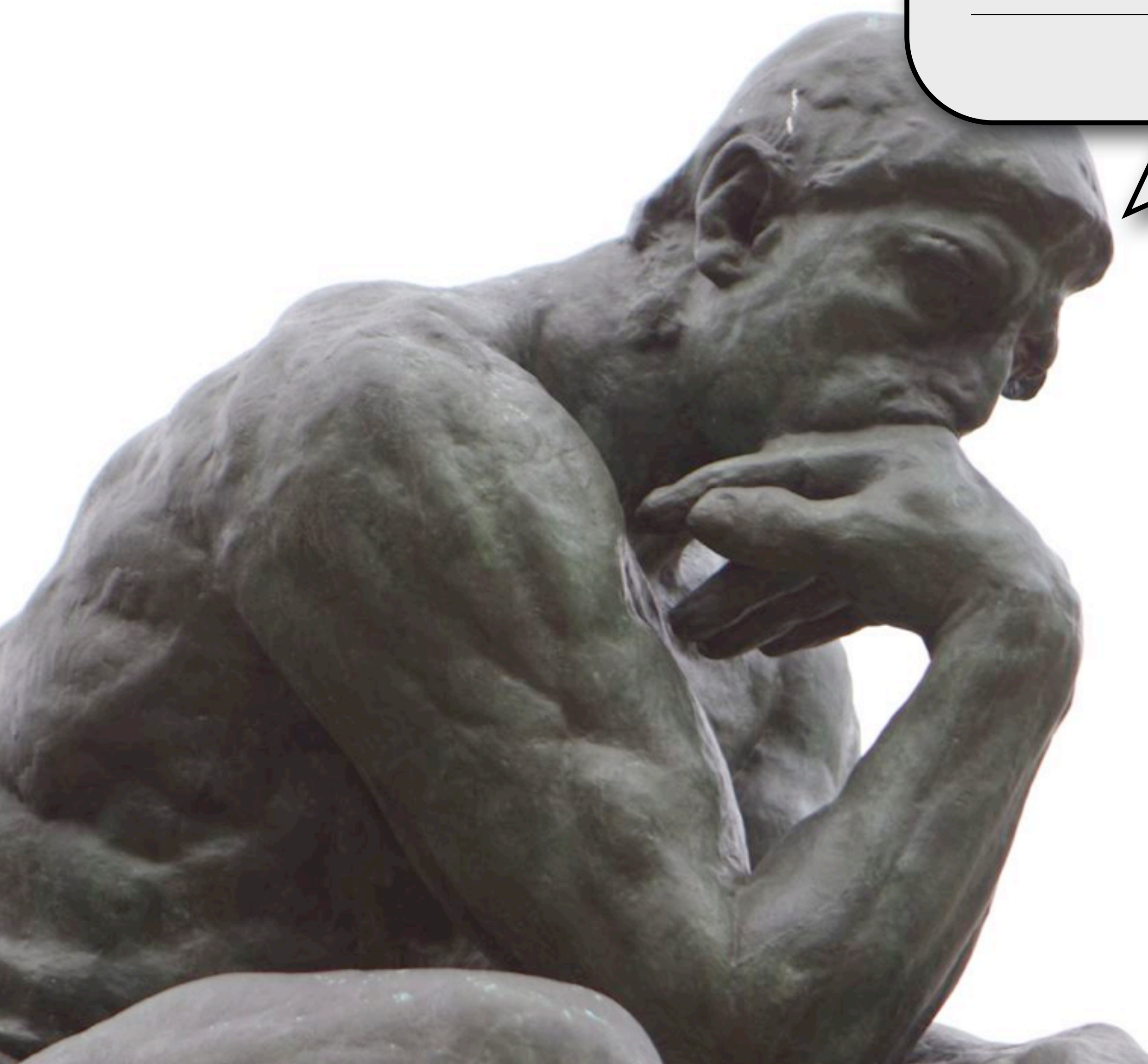
$$\begin{aligned}
[let] \quad & (\widehat{\mathsf{C}}, \widehat{\rho}) \models (\texttt{let } x \texttt{ = } t_1^{\ell_1} \texttt{ in } t_2^{\ell_2})^\ell \\
& \text{iff} \quad (\widehat{\mathsf{C}}, \widehat{\rho}) \models t_1^{\ell_1} \ \wedge\ (\widehat{\mathsf{C}}, \widehat{\rho}) \models t_2^{\ell_2} \ \wedge \\
& \qquad \widehat{\mathsf{C}}(\ell_1) \subseteq \widehat{\rho}(x) \ \wedge\ \widehat{\mathsf{C}}(\ell_2) \subseteq \widehat{\mathsf{C}}(\ell)
\end{aligned}$$

$$[op] \quad (\widehat{\mathsf{C}}, \widehat{\rho}) \models (t_1^{\ell_1}\ op\ t_2^{\ell_2})^\ell \text{ iff } \quad (\widehat{\mathsf{C}}, \widehat{\rho}) \models t_1^{\ell_1} \ \wedge\ (\widehat{\mathsf{C}}, \widehat{\rho}) \models t_2^{\ell_2}$$

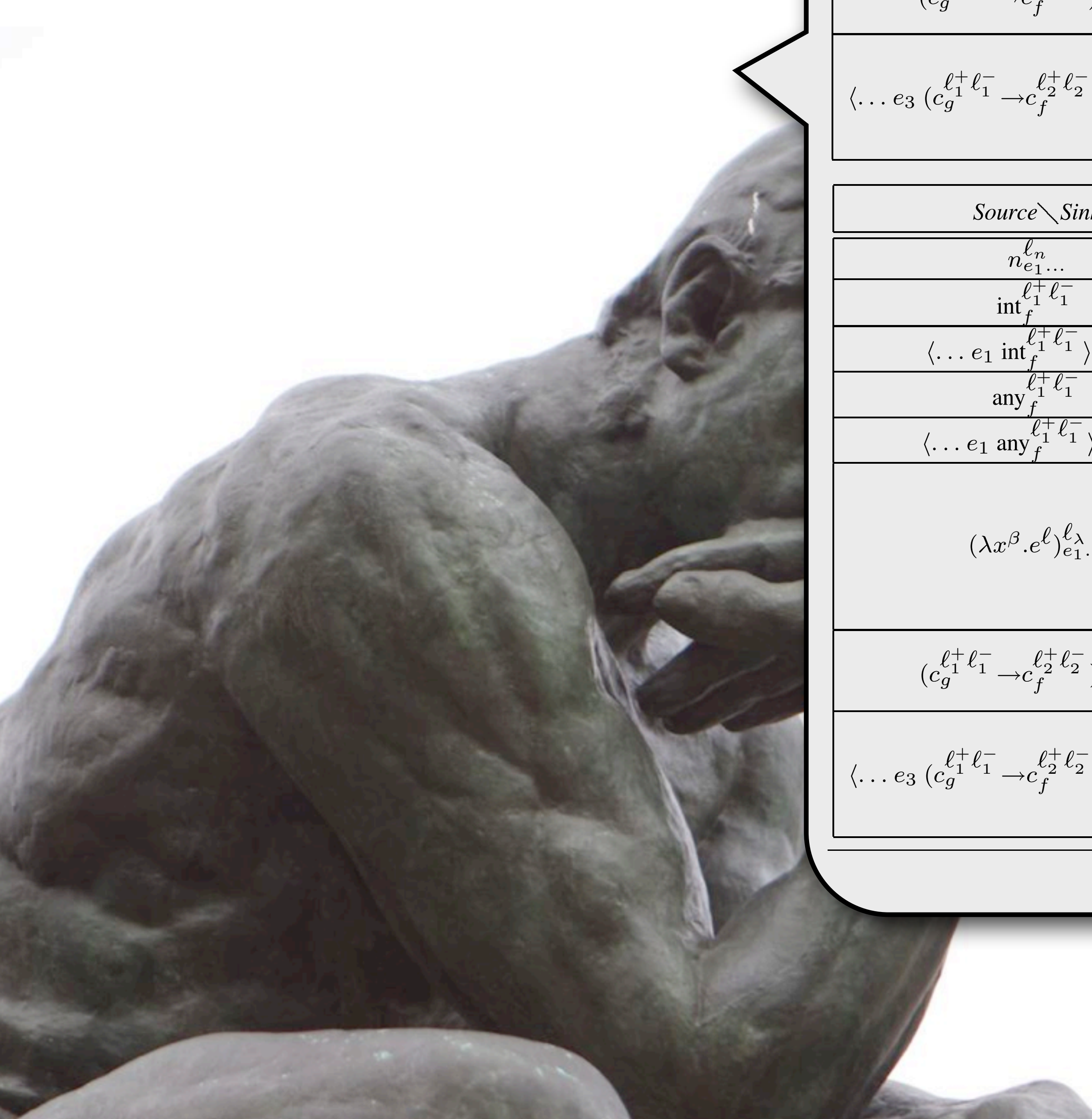
Table 3.1: Abstract Control Flow Analysis (Subsections 3.1.1 and 3.1.2).

$((\lambda x^{\beta}.e)^{\ell_\lambda}\ v^{\ell_v})^{\ell_a}$	$\longrightarrow$	$e[v^{\ell_v}/x^{\beta}]$	SUBST
$(n^{\ell_n}\ v^{\ell_v})^{\ell_a}$	$\longrightarrow$	$(\text{blame}\ \lambda\ \mathcal{R})^{\ell_a}$	APP-ERROR
$(\text{if0}\ 0^{\ell_0}\ e_1\ e_2)^{\ell}$	$\longrightarrow$	e_1	IF0-TRUE
$(\text{if0}\ v^{\ell_v}\ e_1\ e_2)^{\ell}$	$\longrightarrow$	e_2	IF0-FALSE
$(\text{any}_f^{\ell\ell'} \Leftarrow v^{\ell_v})^{\ell_c}$	$\longrightarrow$	v^{ℓ}	ANY
$(\langle e_1\ e_2\ \text{any}_f^{\ell\ell'}\rangle_f^{\ell^+\ell^-} \Leftarrow v_{e\ldots}^{\ell_v})^{\ell_c}$	$\longrightarrow$	$e_1[v_{e\ldots e_2}^{\ell}/\varepsilon^{\ell}]$	ANY-TRIP
$(\text{int}_f^{\ell\ell'} \Leftarrow n^{\ell_n})^{\ell_c}$	$\longrightarrow$	n^{ℓ}	INT-INT
$(\text{int}_f^{\ell\ell'} \Leftarrow \vec{v}^{\ell_v})^{\ell_c}$	$\longrightarrow$	$(\text{blame}\ f\ \mathcal{R})^{\ell'}$	INT-LAM
$(\langle e_1\ e_2\ \text{int}_f^{\ell\ell'}\rangle_f^{\ell^+\ell^-} \Leftarrow n_{e\ldots}^{\ell_n})^{\ell_c}$	$\longrightarrow$	$e_1[n_{e\ldots e_2}^{\ell}/\varepsilon^{\ell}]$	INT-TRIP-INT
$(\langle e_1\ e_2\ \text{int}_f^{\ell\ell'}\rangle_f^{\ell^+\ell^-} \Leftarrow \vec{v}^{\ell_v})^{\ell_c}$	$\longrightarrow$	$(\text{blame}\ f\ \mathcal{R})^{\ell'}$	INT-TRIP-LAM
$((c_1{\rightarrow}c_2)_f^{\ell\ell'} \Leftarrow \vec{v}^{\ell_v})^{\ell_c}$	$\longrightarrow$	$((c_1{\dashrightarrow}c_2)_f^{\ell\ell'} \Leftarrow \vec{v}^{\ell_v})^{\ell_c}$	LAM-LAM
$((c_1{\rightarrow}c_2)_f^{\ell\ell'} \Leftarrow n^{\ell_n})^{\ell_c}$	$\longrightarrow$	$(\text{blame}\ f\ \mathcal{R})^{\ell'}$	LAM-INT
$(\langle e_1\ e_2\ (c_1{\rightarrow}c_2)_f^{\ell\ell'}\rangle_f^{\ell^+\ell^-} \Leftarrow \vec{v}_{e\ldots}^{\ell_v})^{\ell_c}$	$\longrightarrow$	$e_1[((c_1{\dashrightarrow}c_2)_f^{\ell\ell'} \Leftarrow \vec{v}_{e\ldots e_2}^{\ell_v})^{\ell_c}/\varepsilon^{\ell}]$	LAM-TRIP-LAM
$(\langle e_1\ e_2\ (c_1{\rightarrow}c_2)_f^{\ell\ell'}\rangle_f^{\ell^+\ell^-} \Leftarrow n^{\ell_n})^{\ell_c}$	$\longrightarrow$	$(\text{blame}\ f\ \mathcal{R})^{\ell'}$	LAM-TRIP-INT
$(((c_1{\dashrightarrow}c_2)_f^{\ell\ell'} \Leftarrow \vec{v}^{\ell_v})^{\ell_c}\ w^{\ell_w})^{\ell_a}$	$\longrightarrow$	$(c_2{\Leftarrow}\ (\vec{v}^{\ell_v}\ (c_1{\Leftarrow}\ w^{\ell_w})^{lab^+(c_1)})^{lab^-(c_2)})^{lab^+(c_2)}$	SPLIT

Figure 7. Reduction rules.



Semantics of contracts



!

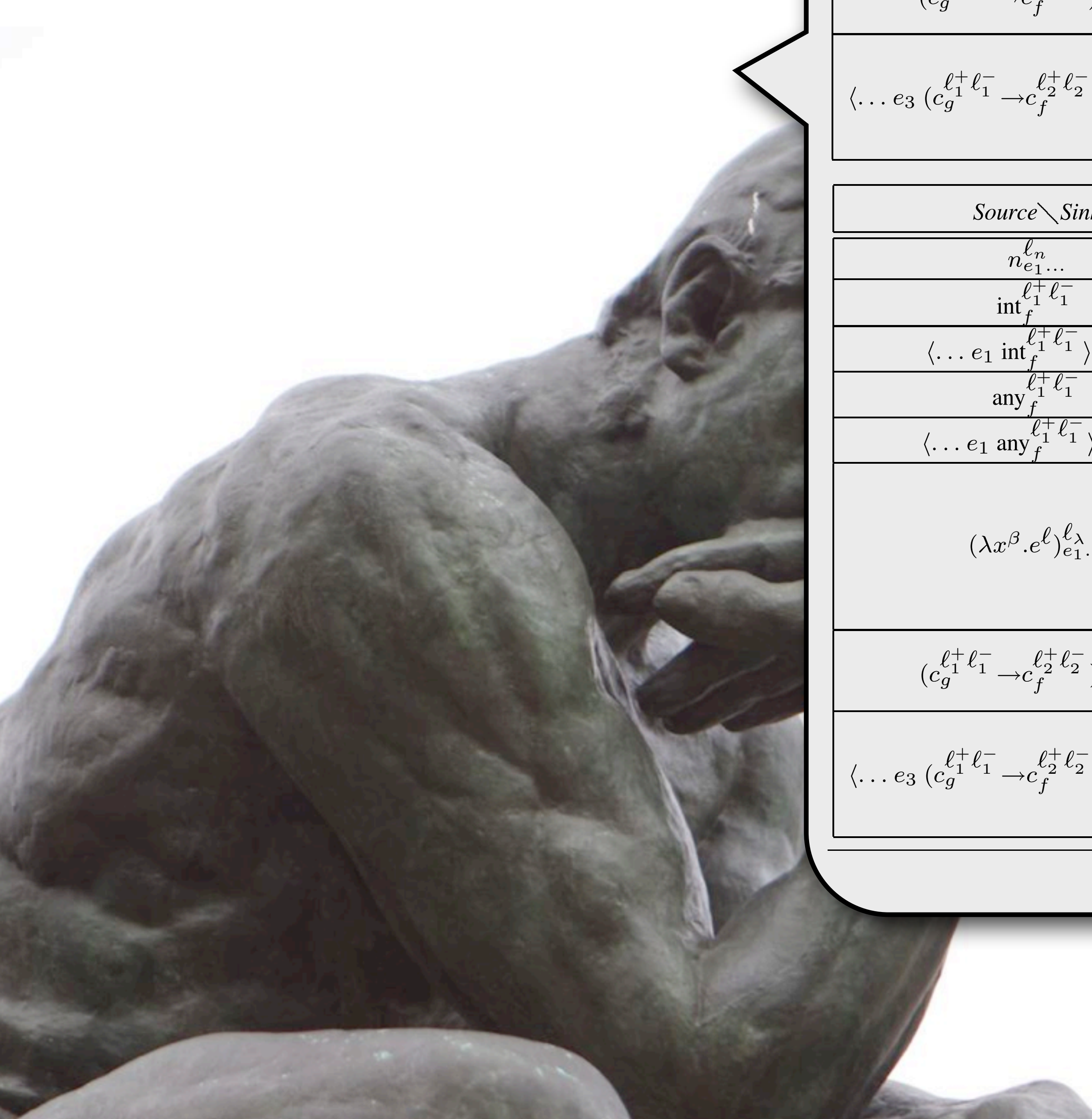
$\mathit{Source} \setminus \mathit{Sink}$	$\text{int}_h^{\ell_5^+\ell_5^-}$	$\langle \dots e_5\ \text{int}_h^{\ell_5^+\ell_5^-} \rangle_h^{\ell_6^+\ell_6^-}$	$\text{any}_h^{\ell_5^+\ell_5^-}$	$\langle \dots e_5\ \text{any}_h^{\ell_5^+\ell_5^-} \rangle_h^{\ell_6^+\ell_6^-}$
$n_{e_1\dots}^{\ell_n}$		$\{\ell_n\} \subseteq \varphi(\ell_5^-),\ e_1 \dots \not\sqsubseteq e_5 \Rightarrow \{\langle h, \mathcal{O} \rangle\} \subseteq \psi(\ell_5^-)$		$\{\ell_n\} \subseteq \varphi(\ell_5^-),\ e_1 \dots \not\sqsubseteq e_5 \Rightarrow \{\langle h, \mathcal{O} \rangle\} \subseteq \psi(\ell_5^-)$
$\text{int}_f^{\ell_1^+\ell_1^-}$		$\{\ell_1^+\} \subseteq \varphi(\ell_5^-) \Rightarrow \{\langle h, \mathcal{O} \rangle\} \subseteq \psi(\ell_5^-)$		$\{\ell_1^+\} \subseteq \varphi(\ell_5^-) \Rightarrow \{\langle h, \mathcal{O} \rangle\} \subseteq \psi(\ell_5^-)$
$\langle \dots e_1\ \text{int}_f^{\ell_1^+\ell_1^-} \rangle_f^{\ell_2^+\ell_2^-}$		$\{\ell_1^+\} \subseteq \varphi(\ell_5^-),\ e_1 \not\sqsubseteq e_5 \Rightarrow \{\langle h, \mathcal{O} \rangle\} \subseteq \psi(\ell_5^-)$		$\{\ell_1^+\} \subseteq \varphi(\ell_5^-),\ e_1 \not\sqsubseteq e_5 \Rightarrow \{\langle h, \mathcal{O} \rangle\} \subseteq \psi(\ell_5^-)$
$\text{any}_f^{\ell_1^+\ell_1^-}$	$\{\ell_1^+\} \subseteq \varphi(\ell_5^-) \Rightarrow \{\langle h, \mathcal{R} \rangle\} \subseteq \psi(\ell_5^-)$			$\{\ell_1^+\} \subseteq \varphi(\ell_5^-) \Rightarrow \{\langle h, \mathcal{O} \rangle\} \subseteq \psi(\ell_5^-)$
$\langle \dots e_1\ \text{any}_f^{\ell_1^+\ell_1^-} \rangle_f^{\ell_2^+\ell_2^-}$				$\{\ell_1^+\} \subseteq \varphi(\ell_5^-),\ e_1 \not\sqsubseteq e_5 \Rightarrow \{\langle h, \mathcal{O} \rangle\} \subseteq \psi(\ell_5^-)$
$(\lambda x^\beta . e^\ell)_{e_1\dots}^{\ell_\lambda}$	$\{\ell_\lambda\} \subseteq \varphi(\ell_5^-) \Rightarrow \{\langle h, \mathcal{R} \rangle\} \subseteq \psi(\ell_5^-)$		$\{\ell_\lambda\} \subseteq \varphi(\ell_5^-) \Rightarrow \varphi(\ell_5^+) \subseteq \varphi(\beta)$ $\{\ell_\lambda\} \subseteq \varphi(\ell_5^-) \Rightarrow \varphi(\ell) \subseteq \varphi(\ell_5^-)$	$\{\ell_\lambda\} \subseteq \varphi(\ell_5^-),\ e_1 \dots \not\sqsubseteq e_5 \Rightarrow \{\langle h, \mathcal{O} \rangle\} \subseteq \psi(\ell_5^-)$
$(c_g^{\ell_1^+\ell_1^-} \to c_f^{\ell_2^+\ell_2^-})_f^{\ell_3^+\ell_3^-}$	$\{\ell_3^+\} \subseteq \varphi(\ell_5^-) \Rightarrow \{\langle h, \mathcal{R} \rangle\} \subseteq \psi(\ell_5^-)$		$\{\ell_3^+\} \subseteq \varphi(\ell_5^-) \Rightarrow \varphi(\ell_5^+) \subseteq \varphi(\ell_1^-)$ $\{\ell_3^+\} \subseteq \varphi(\ell_5^-) \Rightarrow \varphi(\ell_2^+) \subseteq \varphi(\ell_5^-)$	$\{\ell_3^+\} \subseteq \varphi(\ell_5^-) \Rightarrow \{\langle h, \mathcal{O} \rangle\} \subseteq \psi(\ell_5^-)$ $\{\ell_3^+\} \subseteq \varphi(\ell_5^-),\ e_3 \not\sqsubseteq e_5 \Rightarrow \{\langle h, \mathcal{O} \rangle\} \subseteq \psi(\ell_5^-)$
$\langle \dots e_3\ (c_g^{\ell_1^+\ell_1^-} \to c_f^{\ell_2^+\ell_2^-})_f^{\ell_3^+\ell_3^-} \rangle_f^{\ell_4^+\ell_4^-}$				

$\mathit{Source} \setminus \mathit{Sink}$	$(e^{\ell_5}\ e^{\ell_6})^{\ell_a}$	$(c_i^{\ell_7^+\ell_7^-} \to c_h^{\ell_8^+\ell_8^-})_h^{\ell_5^+\ell_5^-}$	$\langle \dots e_5\ (c_i^{\ell_7^+\ell_7^-} \to c_h^{\ell_8^+\ell_8^-})_h^{\ell_5^+\ell_5^-} \rangle_h^{\ell_6^+\ell_6^-}$
$n_{e_1\dots}^{\ell_n}$	$\{\ell_n\} \subseteq \varphi(\ell_5) \Rightarrow \{\langle \lambda, \mathcal{R} \rangle\} \subseteq \psi(\ell_a)$	$\{\ell_n\} \subseteq \varphi(\ell_5^-) \Rightarrow \{\langle h, \mathcal{R} \rangle\} \subseteq \psi(\ell_5^-)$	
$\text{int}_f^{\ell_1^+\ell_1^-}$	$\{\ell_1^+\} \subseteq \varphi(\ell_5) \Rightarrow \{\langle \lambda, \mathcal{R} \rangle\} \subseteq \psi(\ell_a)$	$\{\ell_1^+\} \subseteq \varphi(\ell_5^-) \Rightarrow \{\langle h, \mathcal{R} \rangle\} \subseteq \psi(\ell_5^-)$	
$\langle \dots e_1\ \text{int}_f^{\ell_1^+\ell_1^-} \rangle_f^{\ell_2^+\ell_2^-}$			
$\text{any}_f^{\ell_1^+\ell_1^-}$			
$\langle \dots e_1\ \text{any}_f^{\ell_1^+\ell_1^-} \rangle_f^{\ell_2^+\ell_2^-}$			
$(\lambda x^\beta . e^\ell)_{e_1\dots}^{\ell_\lambda}$	$\{\ell_\lambda\} \subseteq \varphi(\ell_5) \Rightarrow \varphi(\ell_6) \subseteq \varphi(\beta)$ $\{\ell_\lambda\} \subseteq \varphi(\ell_5) \Rightarrow \varphi(\ell) \subseteq \varphi(\ell_a)$	$\{\ell_\lambda\} \subseteq \varphi(\ell_5^-) \Rightarrow \varphi(\ell_7^+) \subseteq \varphi(\beta)$ $\{\ell_\lambda\} \subseteq \varphi(\ell_5^-) \Rightarrow \varphi(\ell) \subseteq \varphi(\ell_8^-)$	$\{\ell_\lambda\} \subseteq \varphi(\ell_5^-),\ e_1 \dots \not\sqsubseteq e_5 \Rightarrow \{\langle h, \mathcal{O} \rangle\} \subseteq \psi(\ell_5^-)$
$(c_g^{\ell_1^+\ell_1^-} \to c_f^{\ell_2^+\ell_2^-})_f^{\ell_3^+\ell_3^-}$	$\{\ell_3^+\} \subseteq \varphi(\ell_5) \Rightarrow \varphi(\ell_6) \subseteq \varphi(\ell_1^-)$ $\{\ell_3^+\} \subseteq \varphi(\ell_5) \Rightarrow \varphi(\ell_2^+) \subseteq \varphi(\ell_a)$	$\{\ell_3^+\} \subseteq \varphi(\ell_5^-) \Rightarrow \varphi(\ell_7^+) \subseteq \varphi(\ell_1^-)$ $\{\ell_3^+\} \subseteq \varphi(\ell_5^-) \Rightarrow \varphi(\ell_2^+) \subseteq \varphi(\ell_8^-)$	$\{\ell_3^+\} \subseteq \varphi(\ell_5^-) \Rightarrow \{\langle h, \mathcal{O} \rangle\} \subseteq \psi(\ell_5^-)$ $\{\ell_3^+\} \subseteq \varphi(\ell_5^-),\ e_3 \not\sqsubseteq e_5 \Rightarrow \{\langle h, \mathcal{O} \rangle\} \subseteq \psi(\ell_5^-)$
$\langle \dots e_3\ (c_g^{\ell_1^+\ell_1^-} \to c_f^{\ell_2^+\ell_2^-})_f^{\ell_3^+\ell_3^-} \rangle_f^{\ell_4^+\ell_4^-}$			

Table 1. Constraints creation for source-sink pairs.

Analysis of contracts

&@#$!

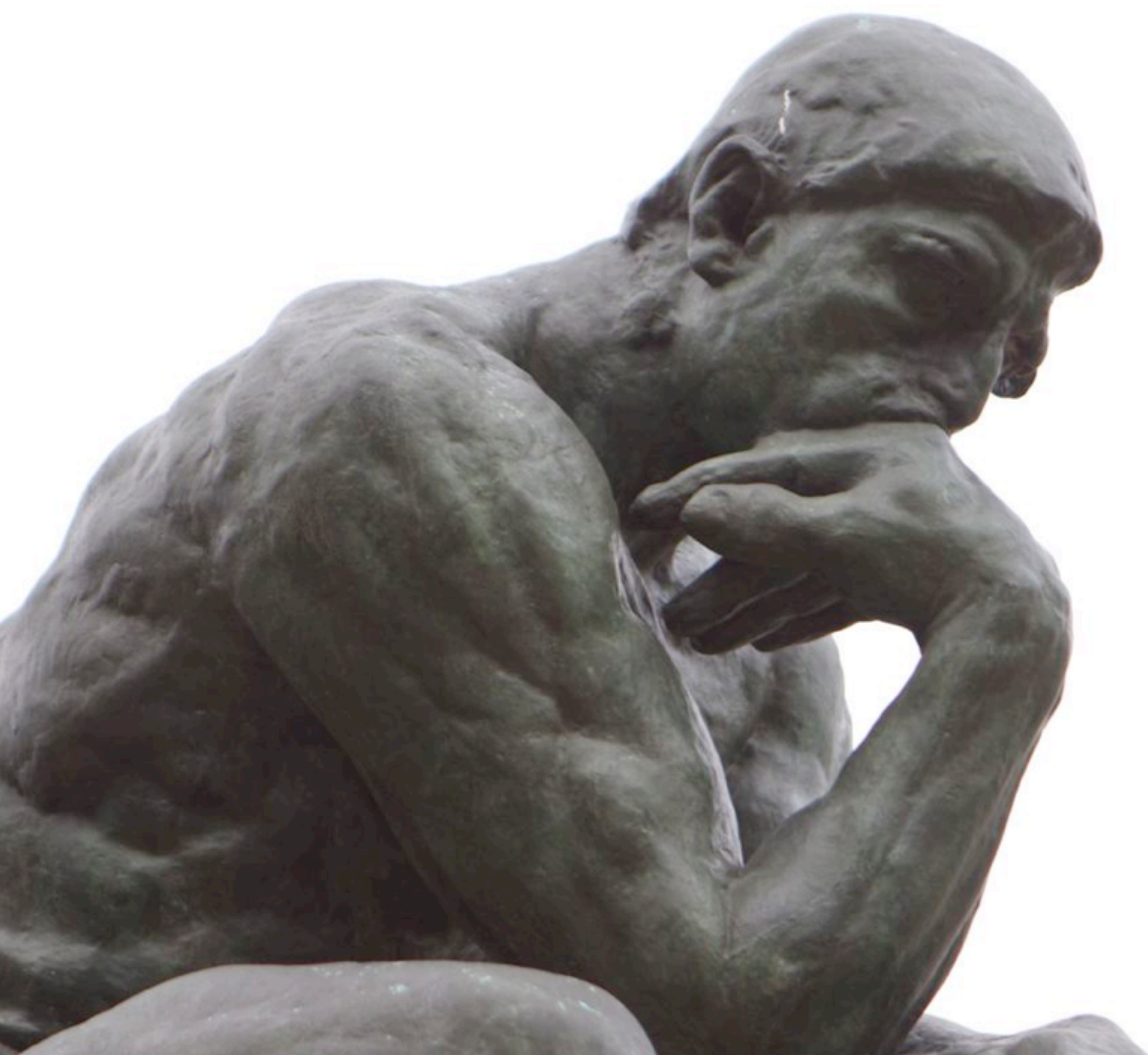


Source \ Sink	$\mathrm{int}_h^{\ell_5^+\ell_5^-}$	$\langle \ldots e_5\ \mathrm{int}_h^{\ell_5^+\ell_5^-}\rangle_h^{\ell_6^+\ell_6^-}$	$\mathrm{any}_h^{\ell_5^+\ell_5^-}$	$\langle \ldots e_5\ \mathrm{any}_h^{\ell_5^+\ell_5^-}\rangle_h^{\ell_6^+\ell_6^-}$
$n_{e_1\ldots}^{\ell_n}$		$\left.\begin{array}{c}\{\ell_n\}\subseteq\varphi(\ell_5^-)\\ e_1\ldots\not\sqsubseteq e_5\end{array}\right\}\Rightarrow\{\langle h,\mathcal{O}\rangle\}\subseteq\psi(\ell_5^-)$		$\left.\begin{array}{c}\{\ell_n\}\subseteq\varphi(\ell_5^-)\\ e_1\ldots\not\sqsubseteq e_5\end{array}\right\}\Rightarrow\{\langle h,\mathcal{O}\rangle\}\subseteq\psi(\ell_5^-)$
$\mathrm{int}_f^{\ell_1^+\ell_1^-}$		$\{\ell_1^+\}\subseteq\varphi(\ell_5^-)\Rightarrow\{\langle h,\mathcal{O}\rangle\}\subseteq\psi(\ell_5^-)$		$\{\ell_1^+\}\subseteq\varphi(\ell_5^-)\Rightarrow\{\langle h,\mathcal{O}\rangle\}\subseteq\psi(\ell_5^-)$
$\langle \ldots e_1\ \mathrm{int}_f^{\ell_1^+\ell_1^-}\rangle_f^{\ell_2^+\ell_2^-}$		$\left.\begin{array}{c}\{\ell_1^+\}\subseteq\varphi(\ell_5^-)\\ e_1\not\sqsubseteq e_5\end{array}\right\}\Rightarrow\{\langle h,\mathcal{O}\rangle\}\subseteq\psi(\ell_5^-)$		$\left.\begin{array}{c}\{\ell_1^+\}\subseteq\varphi(\ell_5^-)\\ e_1\not\sqsubseteq e_5\end{array}\right\}\Rightarrow\{\langle h,\mathcal{O}\rangle\}\subseteq\psi(\ell_5^-)$
$\mathrm{any}_f^{\ell_1^+\ell_1^-}$	$\{\ell_1^+\}\subseteq\varphi(\ell_5^-)\Rightarrow\{\langle h,\mathcal{R}\rangle\}\subseteq\psi(\ell_5^-)$			$\{\ell_1^+\}\subseteq\varphi(\ell_5^-)\Rightarrow\{\langle h,\mathcal{O}\rangle\}\subseteq\psi(\ell_5^-)$
$\langle \ldots e_1\ \mathrm{any}_f^{\ell_1^+\ell_1^-}\rangle_f^{\ell_2^+\ell_2^-}$				$\left.\begin{array}{c}\{\ell_1^+\}\subseteq\varphi(\ell_5^-)\\ e_1\not\sqsubseteq e_5\end{array}\right\}\Rightarrow\{\langle h,\mathcal{O}\rangle\}\subseteq\psi(\ell_5^-)$
$(\lambda x^\beta.e^\ell)_{e_1\ldots}^{\ell_\lambda}$	$\{\ell_\lambda\}\subseteq\varphi(\ell_5^-)\Rightarrow\{\langle h,\mathcal{R}\rangle\}\subseteq\psi(\ell_5^-)$		$\{\ell_\lambda\}\subseteq\varphi(\ell_5^-)\Rightarrow\varphi(\ell_5^+)\subseteq\varphi(\beta)$ $\{\ell_\lambda\}\subseteq\varphi(\ell_5^-)\Rightarrow\varphi(\ell)\subseteq\varphi(\ell_5^-)$ $\left.\begin{array}{c}\{\ell_\lambda\}\subseteq\varphi(\ell_5^-)\\ e_1\ldots\not\sqsubseteq e_5\end{array}\right\}\Rightarrow\{\langle h,\mathcal{O}\rangle\}\subseteq\psi(\ell_5^-)$	
$(c_g^{\ell_1^+\ell_1^-}\rightarrow c_f^{\ell_2^+\ell_2^-})_f^{\ell_3^+\ell_3^-}$	$\{\ell_3^+\}\subseteq\varphi(\ell_5^-)\Rightarrow\{\langle h,\mathcal{R}\rangle\}\subseteq\psi(\ell_5^-)$		$\{\ell_3^+\}\subseteq\varphi(\ell_5^-)\Rightarrow\{\langle h,\mathcal{O}\rangle\}\subseteq\psi(\ell_5^-)$ $\{\ell_3^+\}\subseteq\varphi(\ell_5^-)\Rightarrow\varphi(\ell_5^+)\subseteq\varphi(\ell_1^-)$ $\{\ell_3^+\}\subseteq\varphi(\ell_5^-)\Rightarrow\varphi(\ell_2^+)\subseteq\varphi(\ell_5^-)$ $\left.\begin{array}{c}\{\ell_3^+\}\subseteq\varphi(\ell_5^-)\\ e_3\not\sqsubseteq e_5\end{array}\right\}\Rightarrow\{\langle h,\mathcal{O}\rangle\}\subseteq\psi(\ell_5^-)$	
$\langle \ldots e_3\ (c_g^{\ell_1^+\ell_1^-}\rightarrow c_f^{\ell_2^+\ell_2^-})_f^{\ell_3^+\ell_3^-}\rangle_f^{\ell_4^+\ell_4^-}$				

Source \ Sink	$(e^{\ell_5}\ e^{\ell_6})^{\ell_a}$	$(c_i^{\ell_7^+\ell_7^-}\rightarrow c_h^{\ell_8^+\ell_8^-})_h^{\ell_5^+\ell_5^-}$	$\langle \ldots e_5\ (c_i^{\ell_7^+\ell_7^-}\rightarrow c_h^{\ell_8^+\ell_8^-})_h^{\ell_5^+\ell_5^-}\rangle_h^{\ell_6^+\ell_6^-}$
$n_{e_1\ldots}^{\ell_n}$	$\{\ell_n\}\subseteq$ [illegible] $\subseteq\psi(\ell_a)$	$\{\ell_n\}\subseteq\varphi(\ell_5^-)\Rightarrow\{\langle h,\mathcal{R}\rangle\}\subseteq\psi(\ell_5^-)$	
$\mathrm{int}_f^{\ell_1^+\ell_1^-}$	$\{\ell_1^+\}\subseteq\varphi(\ell_5)\Rightarrow\{\langle\lambda,\mathcal{R}\rangle\}\subseteq\psi(\ell_a)$	$\{\ell_1^+\}\subseteq\varphi(\ell_5^-)\Rightarrow\{\langle h,\mathcal{R}\rangle\}\subseteq\psi(\ell_5^-)$	
$\langle \ldots e_1\ \mathrm{int}_f^{\ell_1^+\ell_1^-}\rangle_f^{\ell_2^+\ell_2^-}$			
$\mathrm{any}_f^{\ell_1^+\ell_1^-}$			
$\langle \ldots e_1\ \mathrm{any}_f^{\ell_1^+\ell_1^-}\rangle_f^{\ell_2^+\ell_2^-}$			
$(\lambda x^\beta.e^\ell)_{e_1\ldots}^{\ell_\lambda}$	$\{\ell_\lambda\}\subseteq\varphi(\ell_5)\Rightarrow\varphi(\ell_6)\subseteq\varphi(\beta)$ $\{\ell_\lambda\}\subseteq\varphi(\ell_5)\Rightarrow\varphi(\ell)\subseteq\varphi(\ell_a)$	$\{\ell_\lambda\}\subseteq\varphi(\ell_5^-)\Rightarrow\varphi(\ell_7^+)\subseteq\varphi(\beta)$ $\{\ell_\lambda\}\subseteq\varphi(\ell_5^-)\Rightarrow\varphi(\ell)\subseteq\varphi(\ell_8^-)$	$\left.\begin{array}{c}\{\ell_\lambda\}\subseteq\varphi(\ell_5^-)\\ e_1\ldots\not\sqsubseteq e_5\end{array}\right\}\Rightarrow\{\langle h,\mathcal{O}\rangle\}\subseteq\psi(\ell_5^-)$
$(c_g^{\ell_1^+\ell_1^-}\rightarrow c_f^{\ell_2^+\ell_2^-})_f^{\ell_3^+\ell_3^-}$	$\{\ell_3^+\}\subseteq\varphi(\ell_5)\Rightarrow\varphi(\ell_6)\subseteq\varphi(\ell_1^-)$ $\{\ell_3^+\}\subseteq\varphi(\ell_5)\Rightarrow\varphi(\ell_2^+)\subseteq\varphi(\ell_a)$	$\{\ell_3^+\}\subseteq\varphi(\ell_5^-)\Rightarrow\varphi(\ell_7^+)\subseteq\varphi(\ell_1^-)$ $\{\ell_3^+\}\subseteq\varphi(\ell_5^-)\Rightarrow\varphi(\ell_2^+)\subseteq\varphi(\ell_8^-)$	$\{\ell_3^+\}\subseteq\varphi(\ell_5^-)\Rightarrow\{\langle h,\mathcal{O}\rangle\}\subseteq\psi(\ell_5^-)$ $\left.\begin{array}{c}\{\ell_3^+\}\subseteq\varphi(\ell_5^-)\\ e_3\not\sqsubseteq e_5\end{array}\right\}\Rightarrow\{\langle h,\mathcal{O}\rangle\}\subseteq\psi(\ell_5^-)$
$\langle \ldots e_3\ (c_g^{\ell_1^+\ell_1^-}\rightarrow c_f^{\ell_2^+\ell_2^-})_f^{\ell_3^+\ell_3^-}\rangle_f^{\ell_4^+\ell_4^-}$			

Table 1. Constraints creation for source-sink pairs.

Analysis of contracts



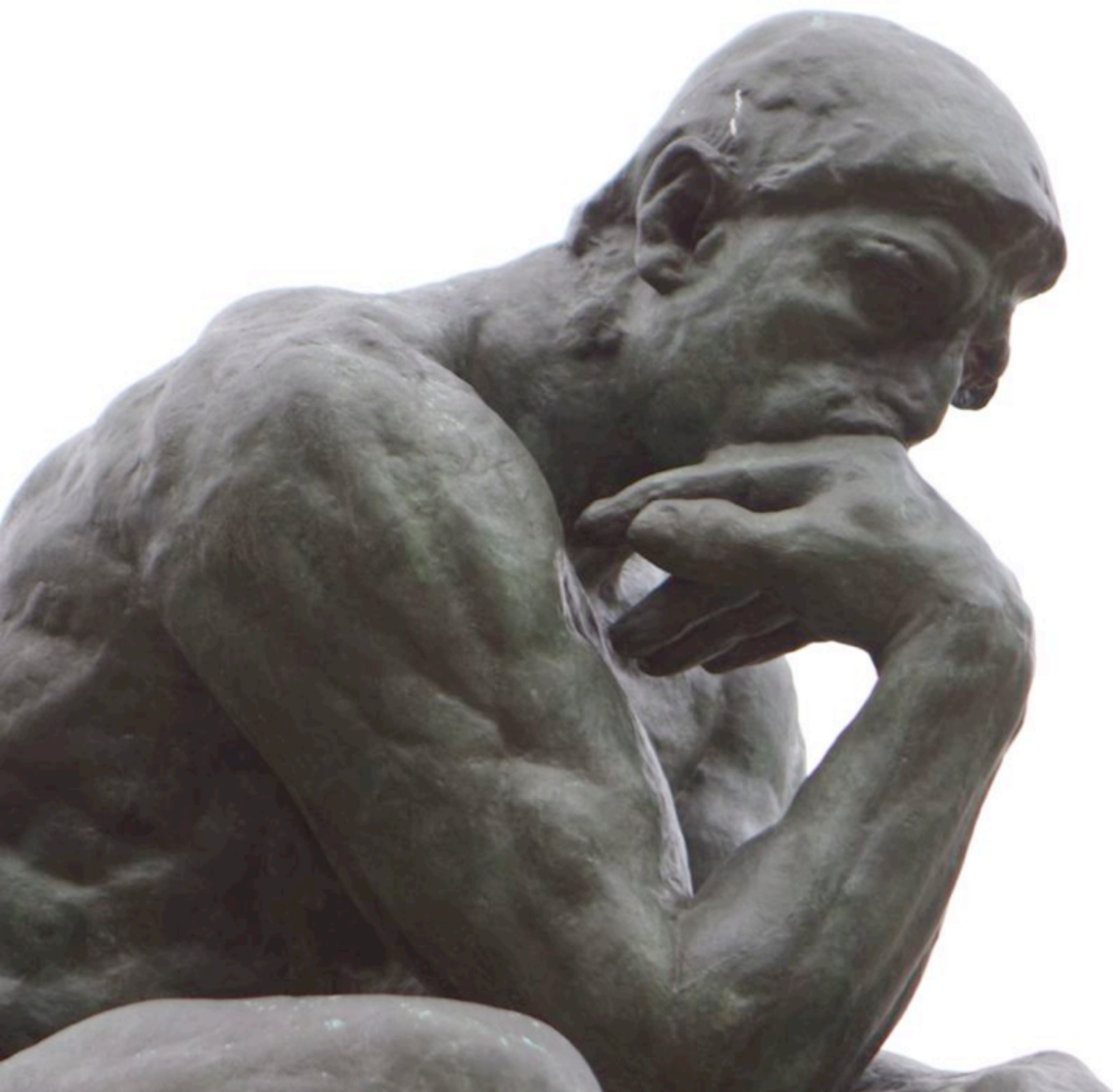
Op & Format	Mnemonic / Syntax	Arguments	Description
	76: invoke-direct/range 77: invoke-static/range 78: invoke-interface/range		
79..7a 10x	*(unused)*		*(unused)*
7b..8f 12x	*unop* vA, vB	A: destination register or pair (4	Perform the identified unary operation on

Op & Format	Mnemonic / Syntax	Arguments	Description
	be: div-long/2addr bf: rem-long/2addr c0: and-long/2addr c1: or-long/2addr c2: xor-long/2addr c3: shl-long/2addr c4: shr-long/2addr c5: ushr-long/2addr c6: add-float/2addr c7: sub-float/2addr c8: mul-float/2addr c9: div-float/2addr ca: rem-float/2addr cb: add-double/2addr cc: sub-double/2addr cd: mul-double/2addr ce: div-double/2addr cf: rem-double/2addr		
d0..d7 22s	*binop*/lit16 vA, vB, #+CCCC d0: add-int/lit16 d1: rsub-int (reverse subtract) d2: mul-int/lit16 d3: div-int/lit16 d4: rem-int/lit16 d5: and-int/lit16 d6: or-int/lit16 d7: xor-int/lit16	A: destination register (4 bits) B: source register (4 bits) C: signed int constant (16 bits)	Perform the indicated binary op on the indicated register (first argument) and literal value (second argument), storing the result in the destination register. **Note:** rsub-int does not have a suffix since this version is the main opcode of its family. Also, see below for details on its semantics.
d8..e2 22b	*binop*/lit8 vAA, vBB, #+CC d8: add-int/lit8 d9: rsub-int/lit8 da: mul-int/lit8 db: div-int/lit8 dc: rem-int/lit8 dd: and-int/lit8 de: or-int/lit8 df: xor-int/lit8 e0: shl-int/lit8 e1: shr-int/lit8 e2: ushr-int/lit8	A: destination register (8 bits) B: source register (8 bits) C: signed int constant (8 bits)	Perform the indicated binary op on the indicated register (first argument) and literal value (second argument), storing the result in the destination register. **Note:** See below for details on the semantics of rsub-int.
e3..ff 10x	*(unused)*		*(unused)*

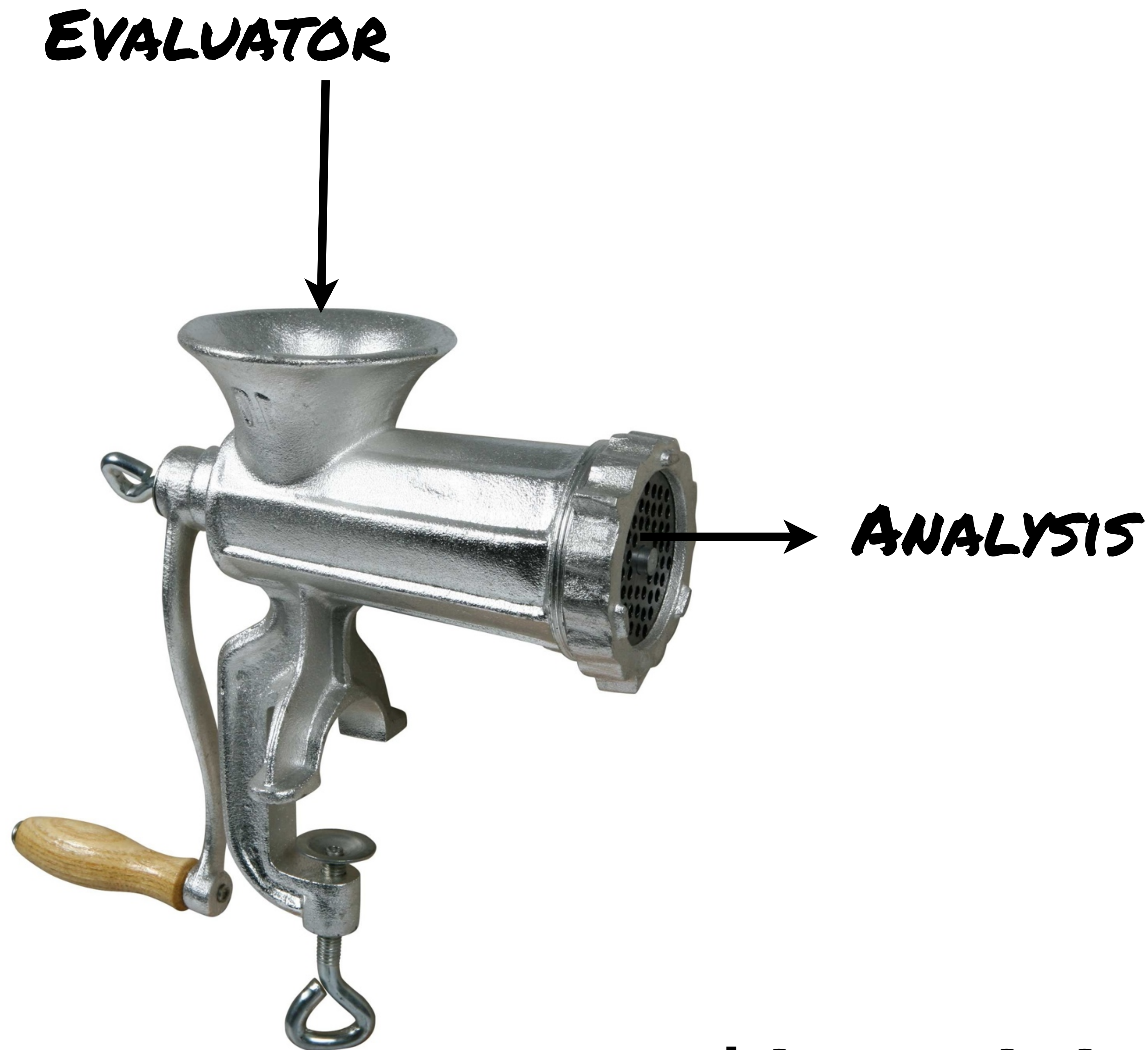
Op & Format	Mnemonic / Syntax	Arguments	Description
b0..cf 12x	*binop*/2addr vA, vB b0: add-int/2addr b1: sub-int/2addr b2: mul-int/2addr b3: div-int/2addr b4: rem-int/2addr b5: and-int/2addr b6: or-int/2addr b7: xor-int/2addr b8: shl-int/2addr b9: shr-int/2addr ba: ushr-int/2addr bb: add-long/2addr bc: sub-long/2addr bd: mul-long/2addr	A: destination and first source register or pair (4 bits) B: second source register or pair (4 bits)	Perform the identified binary operation on the two source registers, storing the result in the first source register.

We need a systematic approach.

Evaluator



Analysis



[ICFP'10, CACM'11, JFP'12]

Part II:
A Systematic Approach

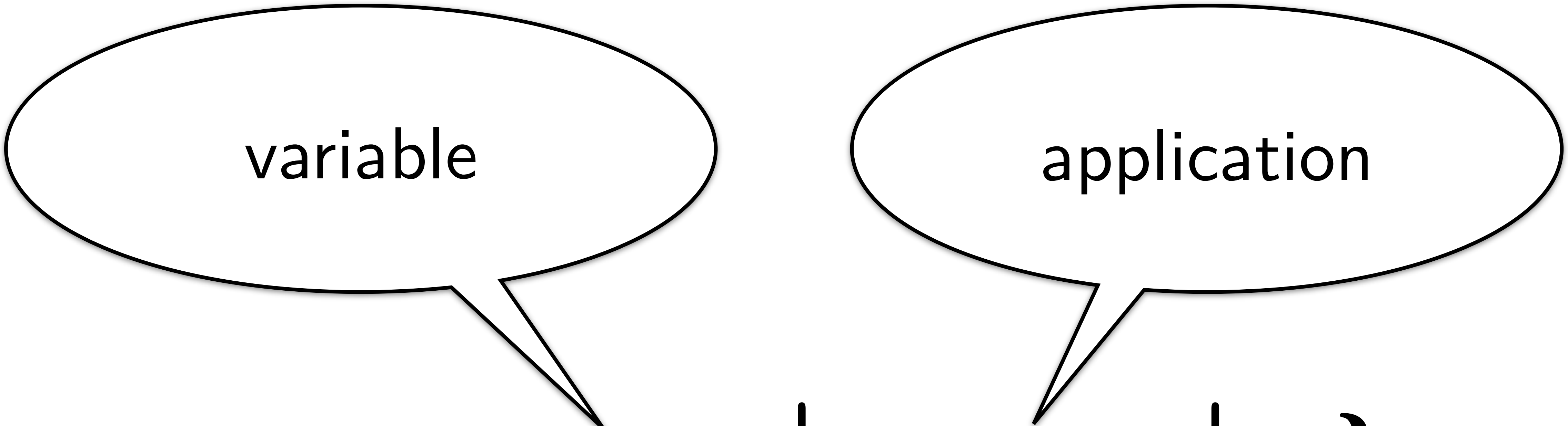
Abstracting Abstract Machines

$$e ::= x \mid e\ e \mid \lambda x.e$$

$$e ::= x \mid e\ e \mid \lambda x.e$$

variable

$$e ::= x \mid e\ e \mid \lambda x.e$$



$$e ::= x \mid e\, e \mid \lambda x.e$$

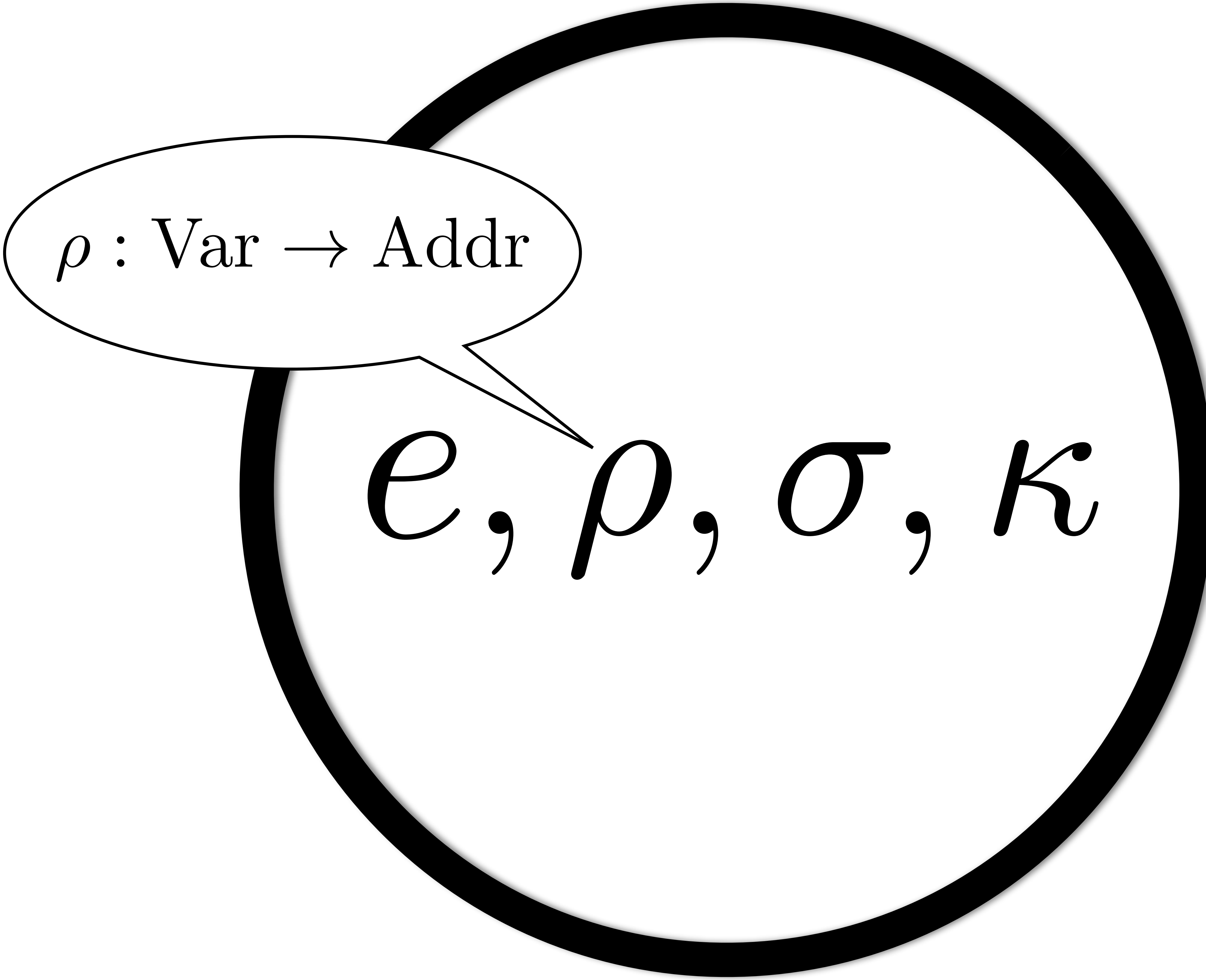
variable

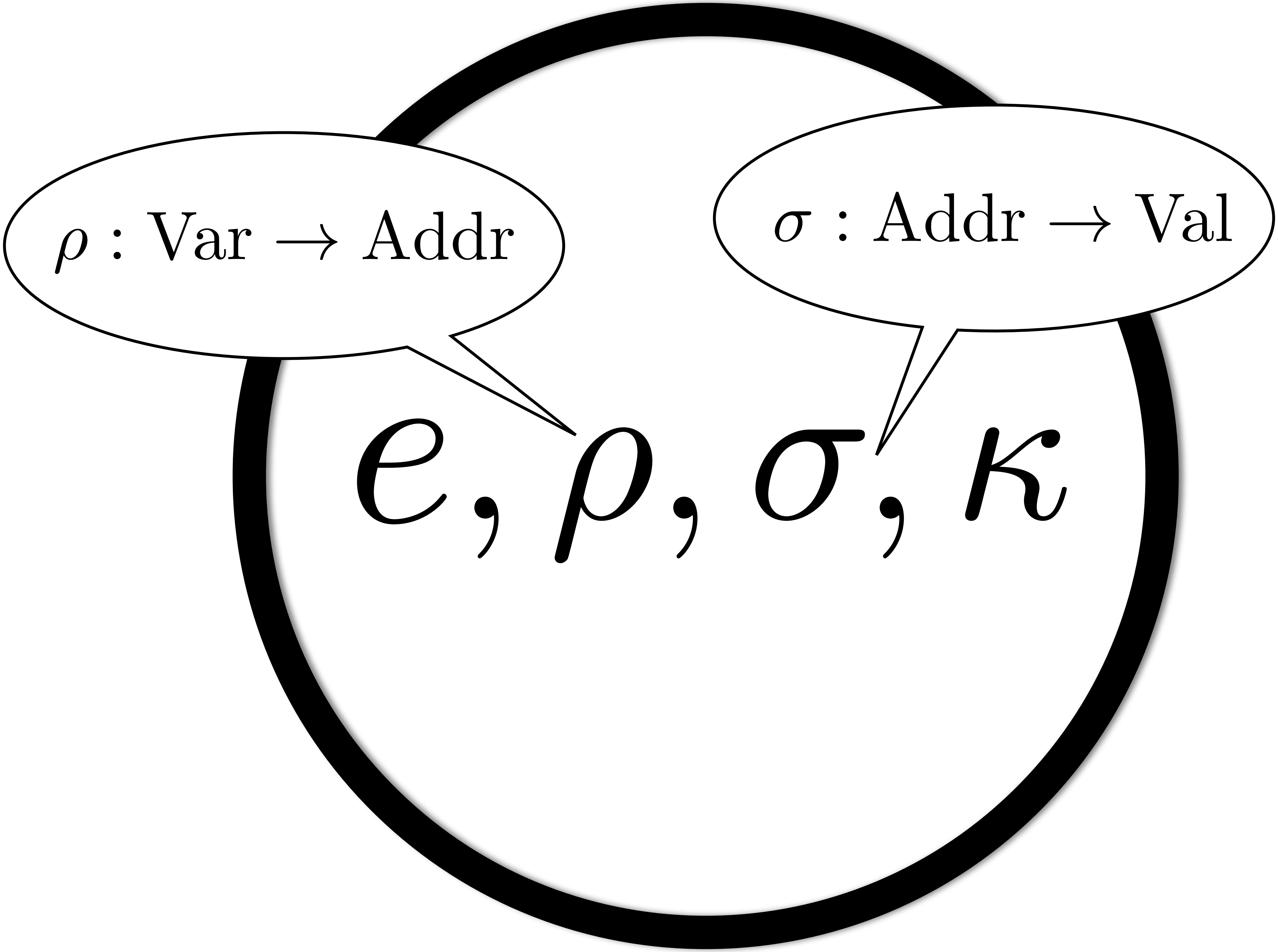
application

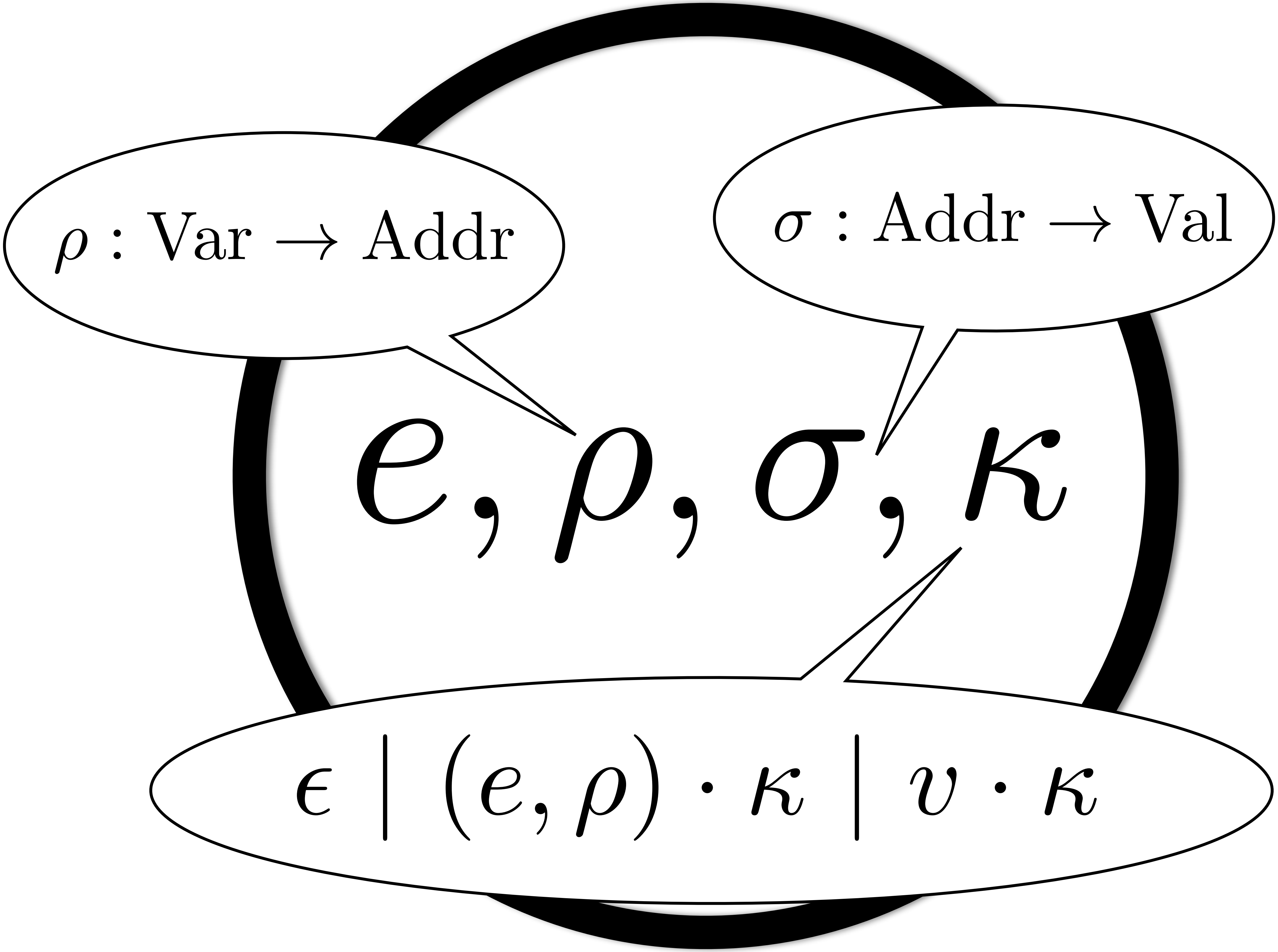
$$e ::= x \mid e\ e \mid \lambda x.e$$

anonymous
function


e, ρ, σ, κ


$\rho : \mathrm{Var} \to \mathrm{Addr}$
e, ρ, σ, κ


$\rho : \mathrm{Var} \to \mathrm{Addr}$
$\sigma : \mathrm{Addr} \to \mathrm{Val}$
e, ρ, σ, κ


e, ρ, σ, κ
$\rho : \mathrm{Var} \to \mathrm{Addr}$
$\sigma : \mathrm{Addr} \to \mathrm{Val}$
$\epsilon \mid (e, \rho) \cdot \kappa \mid v \cdot \kappa$

$$
\begin{array}{rcll}
\langle x, \rho, \sigma, \kappa \rangle & \longmapsto & \langle v, \rho, \sigma, \kappa \rangle & \text{if } v = \sigma(\rho(x)) \\
\langle e_0\ e_1, \rho, \sigma, \kappa \rangle & \longmapsto & \langle e_0, \rho, \sigma, (e_1, \rho) \cdot \kappa \rangle & \\
\langle v, \sigma, (e, \rho) \cdot \kappa \rangle & \longmapsto & \langle e, \rho, \sigma, v \cdot \kappa \rangle & \\
\langle v, \sigma, (\lambda x.e, \rho) \cdot \kappa \rangle & \longmapsto & \langle e, \rho[x \mapsto a], \sigma[a \mapsto v], \kappa \rangle &
\end{array}
$$

CESK machine

Felleisen & Friedman, ’88

$$
\begin{array}{rcl}
\langle x, \rho, \sigma, \kappa \rangle & \longmapsto & \langle v, \rho, \sigma, \kappa \rangle \quad \text{if } v = \sigma(\rho(x)) \\
\langle e_0\ e_1, \rho, \sigma, \kappa \rangle & \longmapsto & \langle e_0, \rho, \sigma, (e_1, \rho) \cdot \kappa \rangle \\
\langle v, \sigma, (e, \rho) \cdot \kappa \rangle & \longmapsto & \langle e, \rho, \sigma, v \cdot \kappa \rangle \\
\langle v, \sigma, (\lambda x.e, \rho) \cdot \kappa \rangle & \longmapsto & \langle e, \rho[x \mapsto a], \sigma[a \mapsto v], \kappa \rangle
\end{array}
$$

CESK machine

Felleisen & Friedman, '88

$$
\begin{array}{rcl}
\langle x, \rho, \sigma, \kappa \rangle & \longmapsto & \langle v, \rho, \sigma, \kappa \rangle \quad \text{if } v = \sigma(\rho(x)) \\
\langle e_0\ e_1, \rho, \sigma, \kappa \rangle & \longmapsto & \langle e_0, \rho, \sigma, (e_1, \rho) \cdot \kappa \rangle \\
\langle v, \sigma, (e, \rho) \cdot \kappa \rangle & \longmapsto & \langle e, \rho, \sigma, v \cdot \kappa \rangle \\
\langle v, \sigma, (\lambda x.e, \rho) \cdot \kappa \rangle & \longmapsto & \langle e, \rho[x \mapsto a], \sigma[a \mapsto v], \kappa \rangle
\end{array}
$$

CESK machine

Felleisen & Friedman, '88

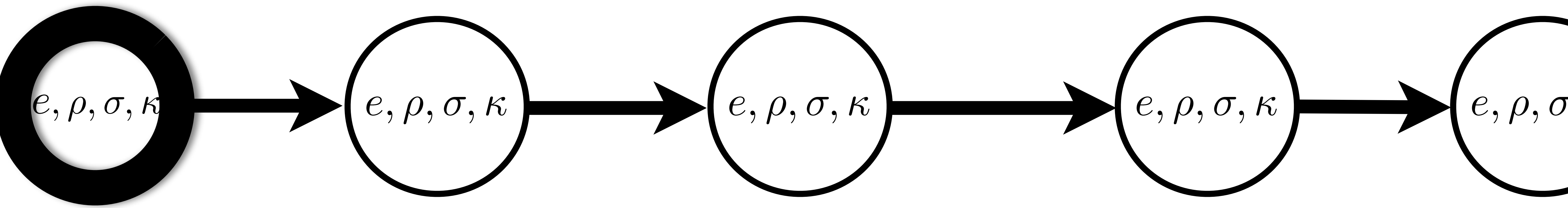
$$
\begin{array}{rcll}
\langle x, \rho, \sigma, \kappa \rangle & \longmapsto & \langle v, \rho, \sigma, \kappa \rangle & \text{if } v = \sigma(\rho(x)) \\
\langle e_0\ e_1, \rho, \sigma, \kappa \rangle & \longmapsto & \langle e_0, \rho, \sigma, (e_1, \rho) \cdot \kappa \rangle & \\
\langle v, \sigma, (e, \rho) \cdot \kappa \rangle & \longmapsto & \langle e, \rho, \sigma, v \cdot \kappa \rangle & \\
\langle v, \sigma, (\lambda x.e, \rho) \cdot \kappa \rangle & \longmapsto & \langle e, \rho[x \mapsto a], \sigma[a \mapsto v], \kappa \rangle &
\end{array}
$$

CESK machine

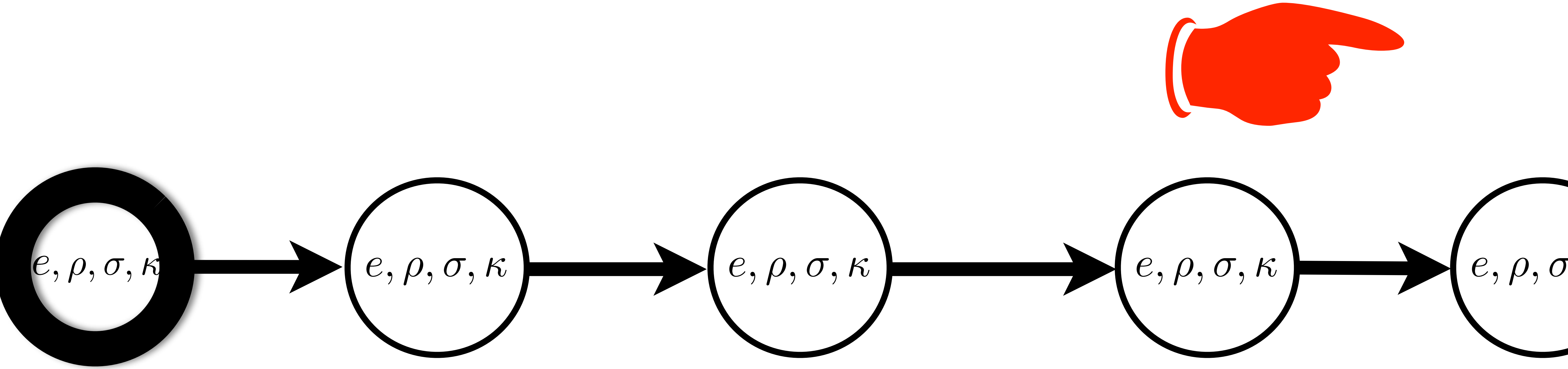
Felleisen & Friedman, '88


e, ρ, σ, κ


e, ρ, σ, κ

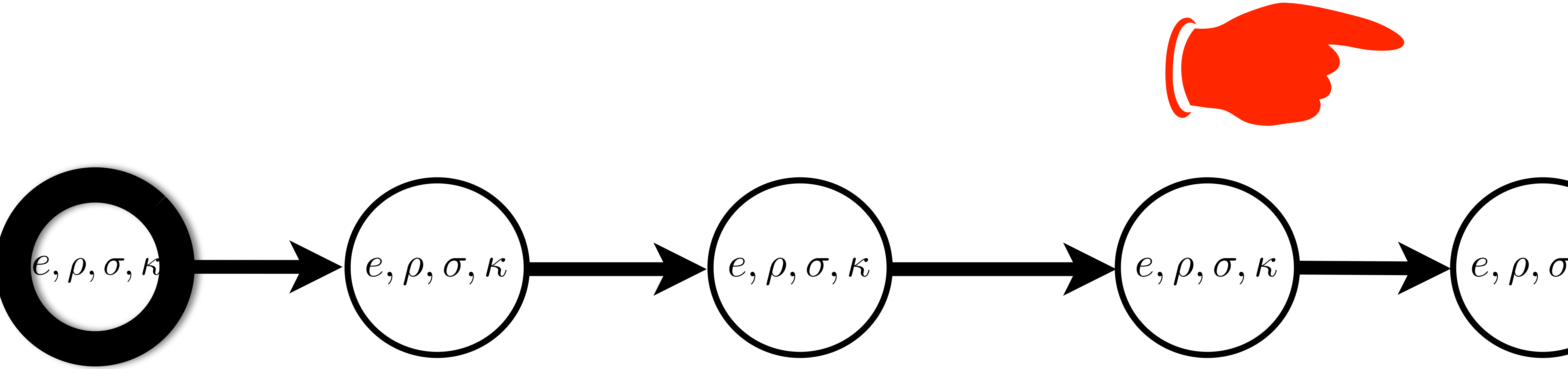


e, ρ, σ, κ
e, ρ, σ, κ
e, ρ, σ, κ
e, ρ, σ, κ

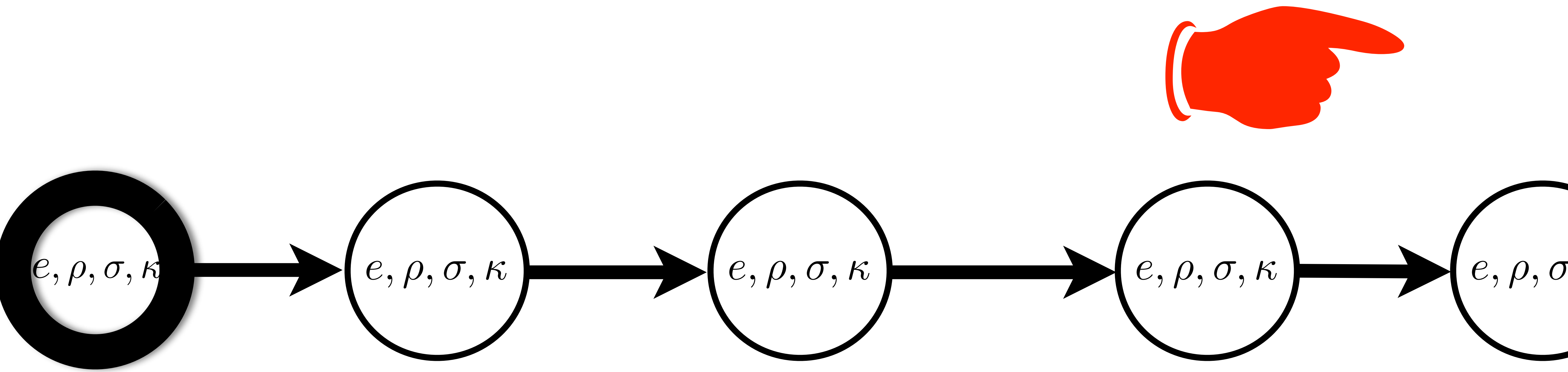


e, ρ, σ, κ
e, ρ, σ, κ
e, ρ, σ, κ
e, ρ, σ, κ

We cannot predict because the future is undecidable.



We cannot predict because the future is undecidable.

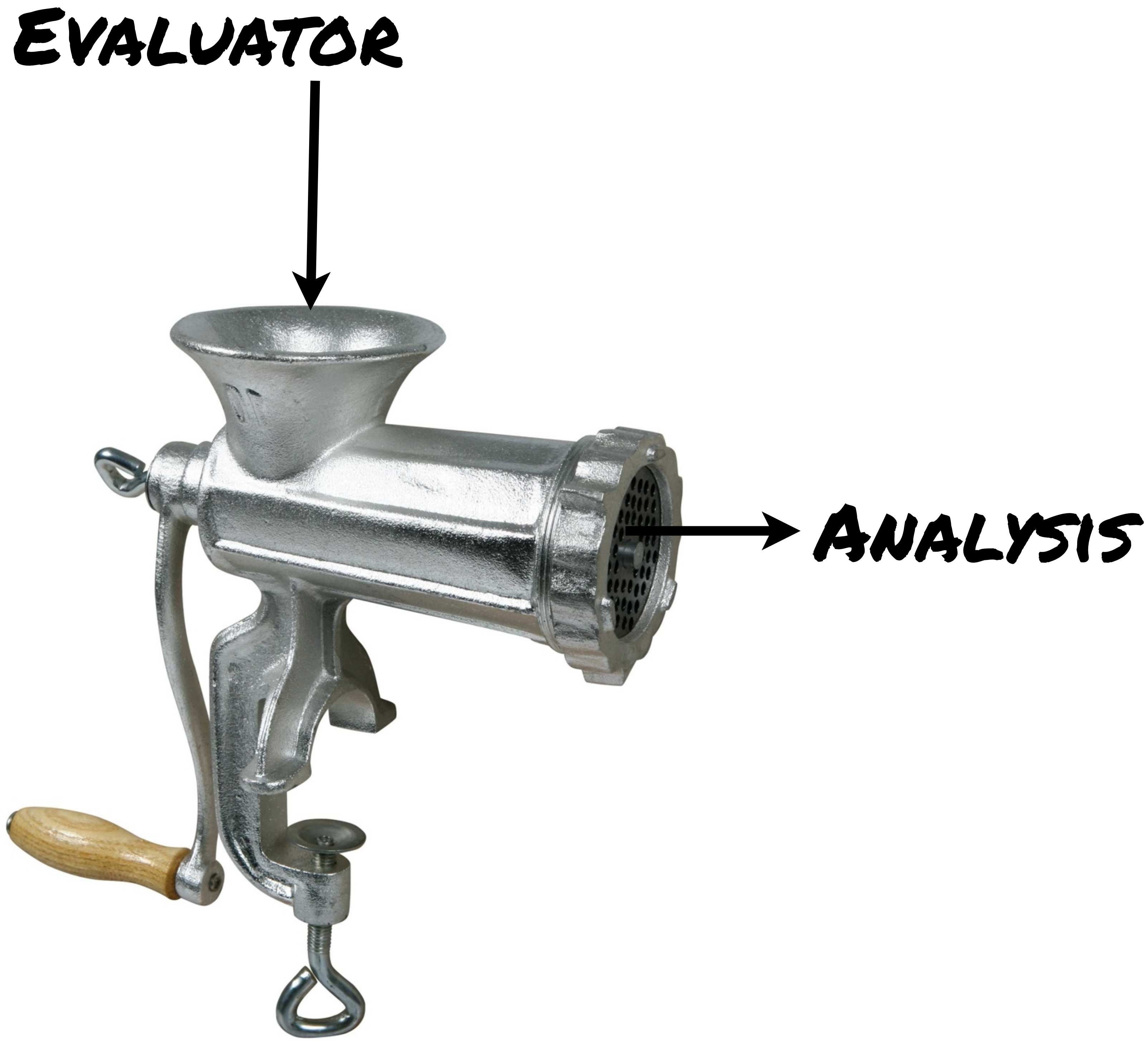


Program analysis

=

sound, computable approximations

Key idea:



Key idea:

EVALUATOR

infinite, deterministic
transition system

ANALYSIS

Key idea:

EVALUATOR

infinite, deterministic
transition system

ANALYSIS

finite, non-deterministic
transition system

$$e, \rho, \sigma, \kappa$$

Idea: make it finite

$$e, \rho, \sigma, \kappa$$

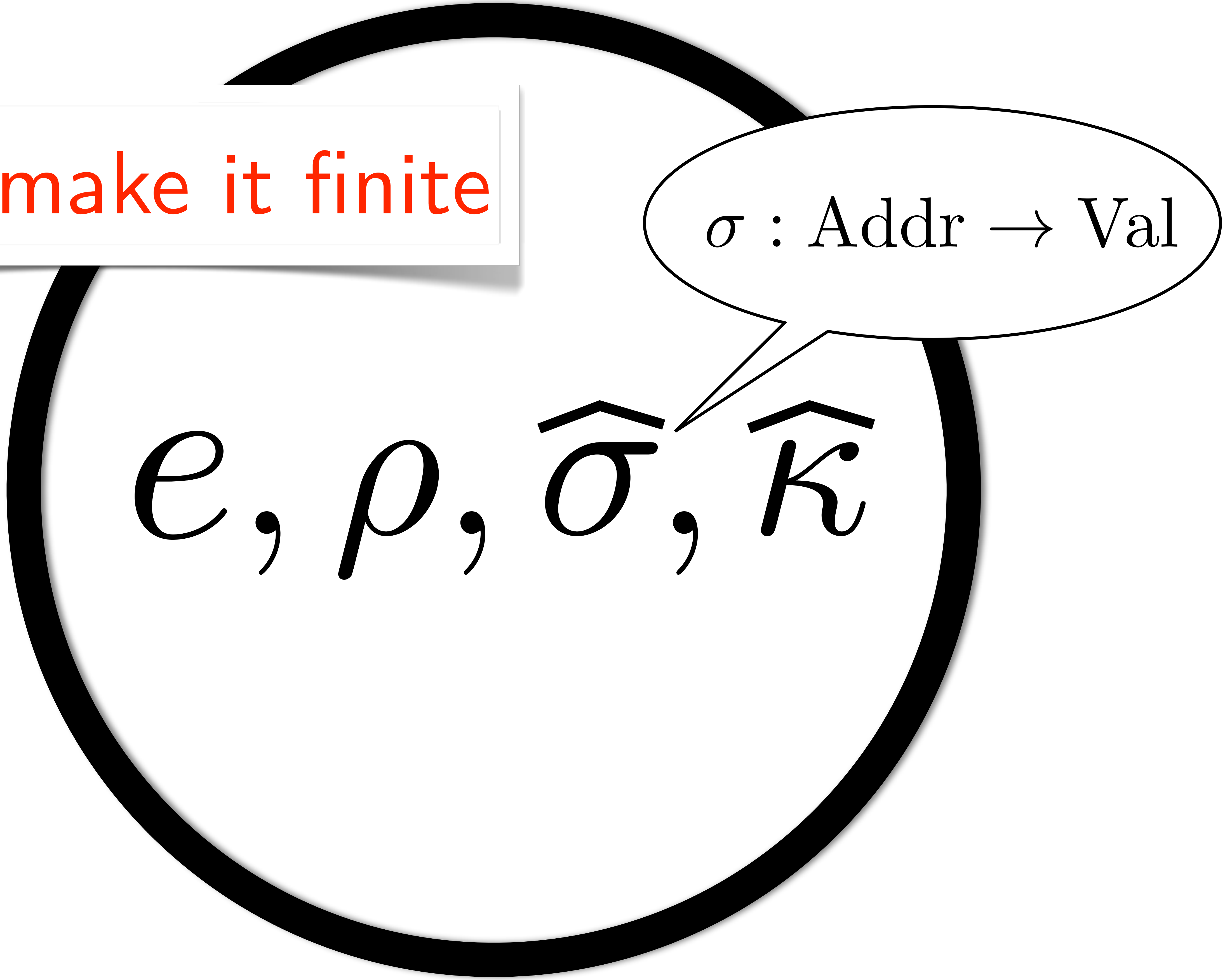
Idea: make it finite

$$e, \rho, \widehat{\sigma}, \widehat{\kappa}$$

Idea: make it finite

$$e, \rho, \widehat{\sigma}, \widehat{\kappa}$$

Idea: make it finite

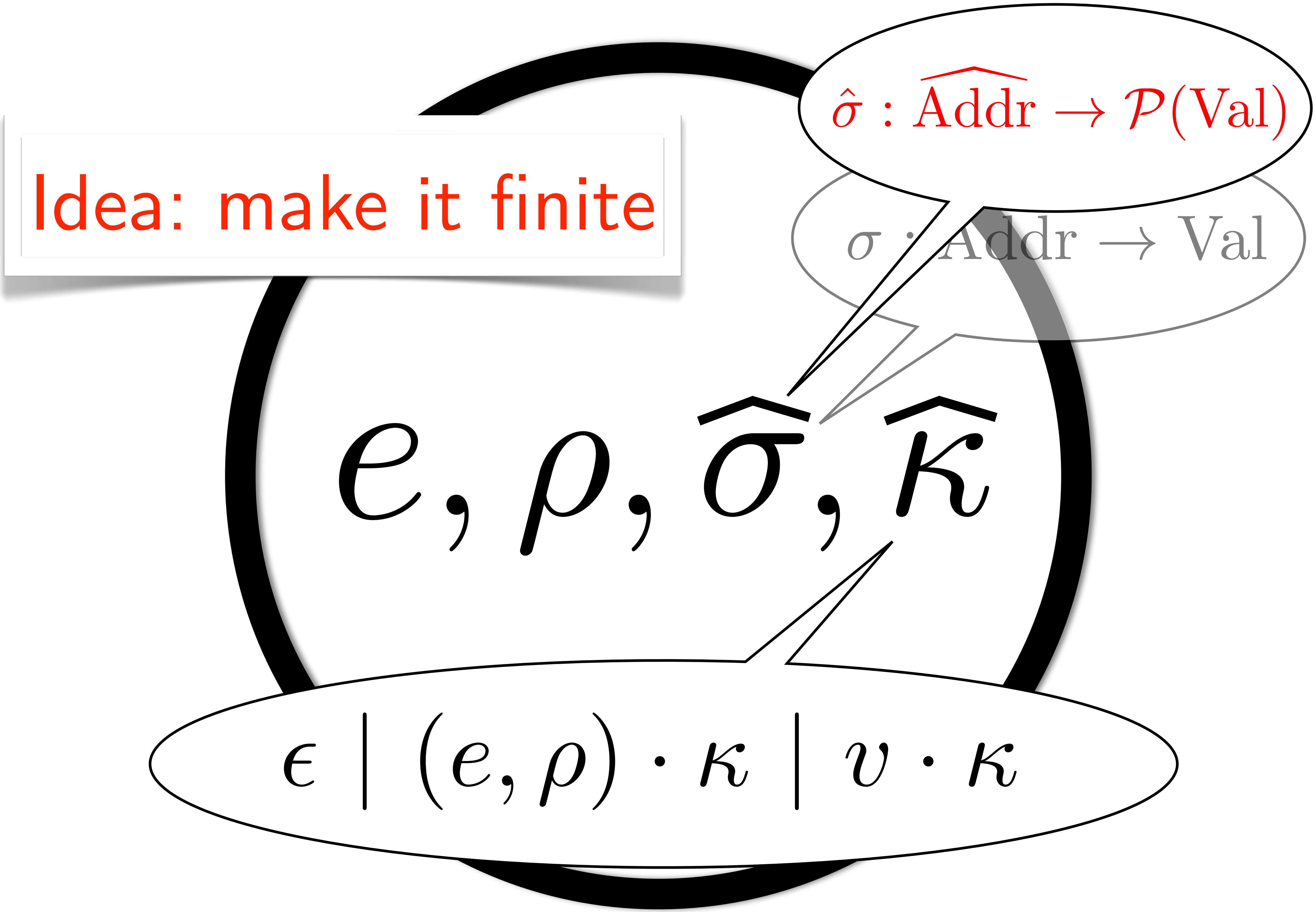


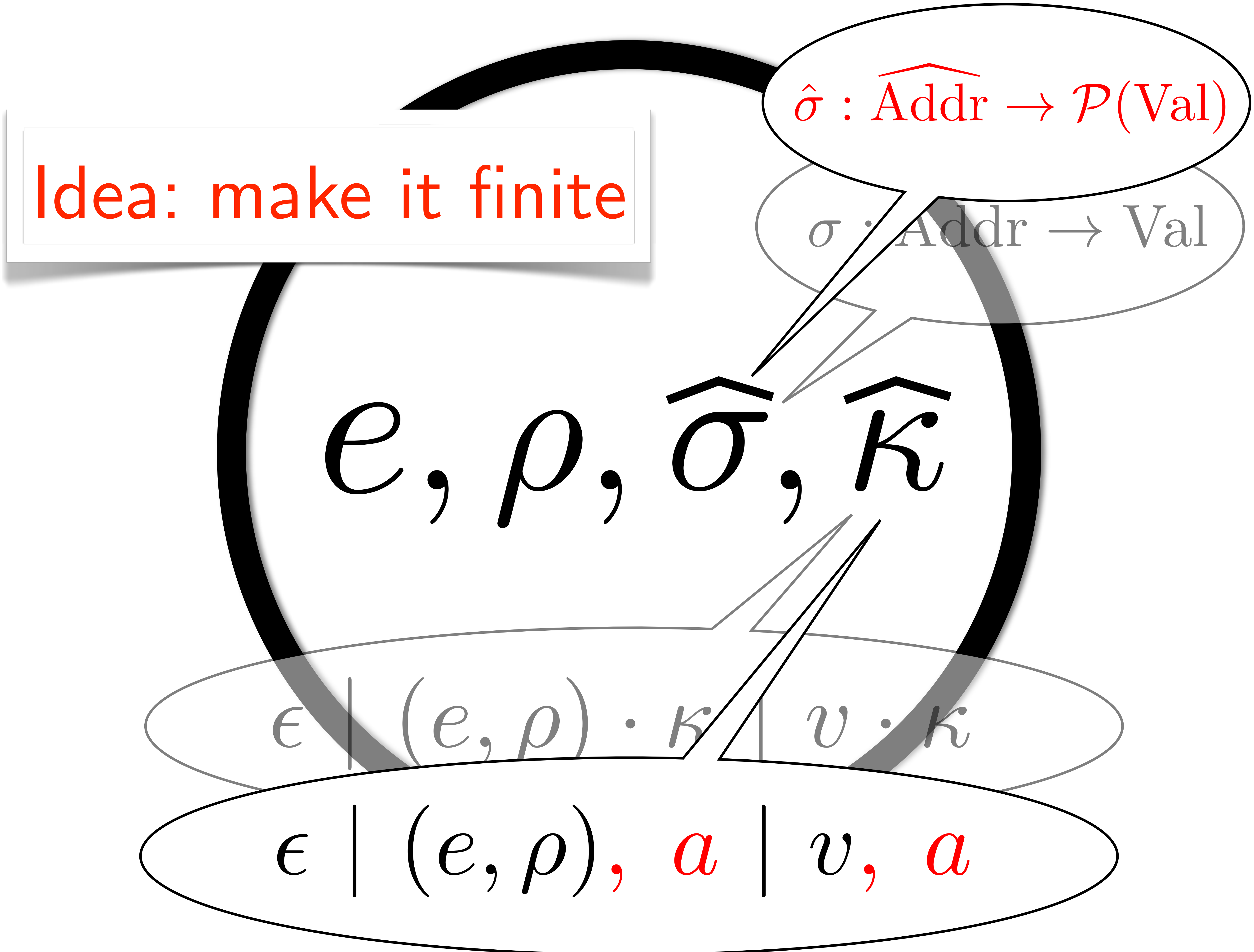
Idea: make it finite

$$e, \rho, \widehat{\sigma}, \widehat{\kappa}$$

$$\sigma : \mathrm{Addr} \to \mathrm{Val}$$

$$\epsilon \mid (e, \rho) \cdot \kappa \mid v \cdot \kappa$$


Idea: make it finite
$\hat{\sigma} : \widehat{\text{Addr}} \to \mathcal{P}(\text{Val})$
$\sigma : \text{Addr} \to \text{Val}$
$e, \rho, \widehat{\sigma}, \widehat{\kappa}$
$\epsilon \mid (e, \rho) \cdot \kappa \mid v \cdot \kappa$


Idea: make it finite
$\hat{\sigma} : \widehat{\text{Addr}} \to \mathcal{P}(\text{Val})$
$\sigma : \text{Addr} \to \text{Val}$
$e, \rho, \widehat{\sigma}, \widehat{\kappa}$
$\epsilon \mid (e, \rho) \cdot \kappa \mid v \cdot \kappa$
$\epsilon \mid (e, \rho), a \mid v, a$

$$
\begin{aligned}
\langle x, \rho, \sigma, \kappa \rangle & \longmapsto \langle v, \rho, \sigma, \kappa \rangle \quad \text{if } v = \sigma(\rho(x)) \\
\langle e_0\ e_1, \rho, \sigma, \kappa \rangle & \longmapsto \langle e_0, \rho, \sigma, (e_1, \rho) \cdot \kappa \rangle \\
\langle v, \sigma, (e, \rho) \cdot \kappa \rangle & \longmapsto \langle e, \rho, \sigma, v \cdot \kappa \rangle \\
\langle v, \sigma, (\lambda x.e, \rho) \cdot \kappa \rangle & \longmapsto \langle e, \rho[x \mapsto a], \sigma[a \mapsto v], \kappa \rangle
\end{aligned}
$$

$$\begin{aligned}
\langle x, \rho, \sigma, \kappa \rangle &\longmapsto \langle v, \rho, \sigma, \kappa \rangle \quad \text{if } v = \sigma(\rho(x)) \\
\langle e_0\ e_1, \rho, \sigma, \kappa \rangle &\longmapsto \langle e_0, \rho, \sigma, (e_1, \rho) \cdot \kappa \rangle \\
\langle v, \sigma, (e, \rho) \cdot \kappa \rangle &\longmapsto \langle e, \rho, \sigma, v \cdot \kappa \rangle \\
\langle v, \sigma, (\lambda x.e, \rho) \cdot \kappa \rangle &\longmapsto \langle e, \rho[x \mapsto a], \sigma[a \mapsto v], \kappa \rangle
\end{aligned}$$

$$\begin{aligned}
\langle x, \rho, \hat{\sigma}, \hat{\kappa} \rangle &\longmapsto \langle v, \rho, \hat{\sigma}, \hat{\kappa} \rangle \quad \text{if } v \in \hat{\sigma}(\rho(x)) \\
\langle e_0\ e_1, \rho, \hat{\sigma}, \hat{\kappa} \rangle &\longmapsto \langle e_0, \rho, \hat{\sigma} \sqcup [a \mapsto \hat{\kappa}], (e_1, \rho), a \rangle \\
\langle v, \hat{\sigma}, (e, \rho), a \rangle &\longmapsto \langle e, \rho, \hat{\sigma}, v, a \rangle \\
\langle v, \hat{\sigma}, (\lambda x.e, \rho), a \rangle &\longmapsto \langle e, \rho[x \mapsto a'], \hat{\sigma} \sqcup [a' \mapsto v], \hat{\kappa} \rangle \\
& \quad \text{if } \hat{\kappa} \in \hat{\sigma}(a)
\end{aligned}$$

*k*CFA

⋮

1CFA

0CFA

Simple closure

Sub0CFA

⋮

$$\langle x, \rho, \sigma, \kappa \rangle \quad \longmapsto \quad \langle v, \rho, \sigma, \kappa \rangle$$

$$\langle e_0\ e_1, \rho, \sigma, \kappa \rangle \quad \longmapsto \quad \langle e_0, \rho, \sigma, (e$$

$$\langle v, \sigma, (e, \rho) \cdot \kappa \rangle \quad \longmapsto \quad \langle e, \rho, \sigma, v \cdot$$

$$\langle v, \sigma, (\lambda x.e, \rho) \cdot \kappa \rangle \quad \longmapsto \quad \langle e, \rho[x \mapsto a$$

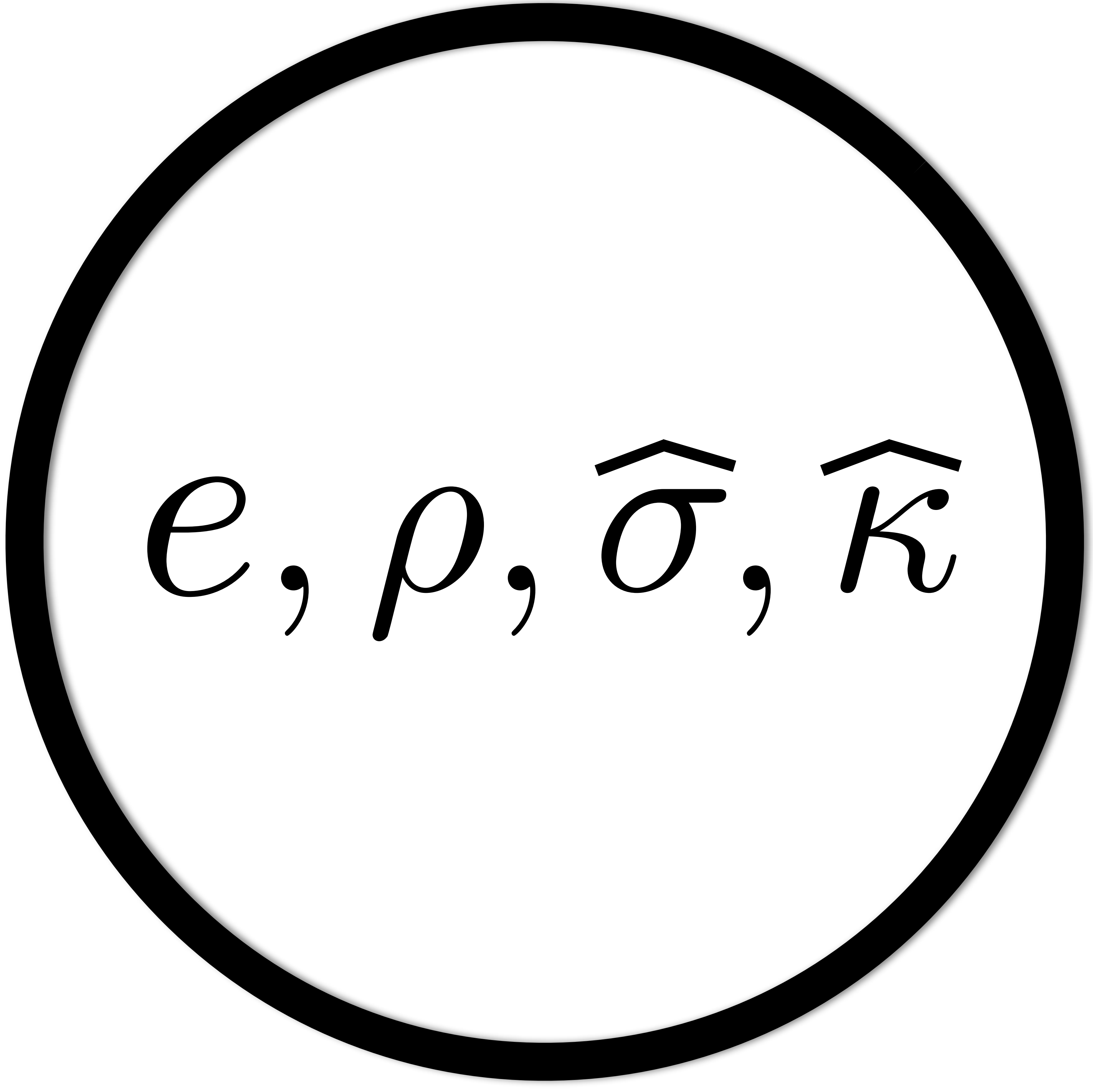
$$\langle x, \rho, \hat{\sigma}, \hat{\kappa} \rangle \quad \longmapsto \quad \langle v, \rho, \hat{\sigma}, \hat{\kappa} \rangle \quad \text{if } v \in \hat{\sigma}(\rho(x))$$

$$\langle e_0\ e_1, \rho, \hat{\sigma}, \hat{\kappa} \rangle \quad \longmapsto \quad \langle e_0, \rho, \hat{\sigma} \sqcup [a \mapsto \hat{\kappa}], (e_1, \rho), a \rangle$$

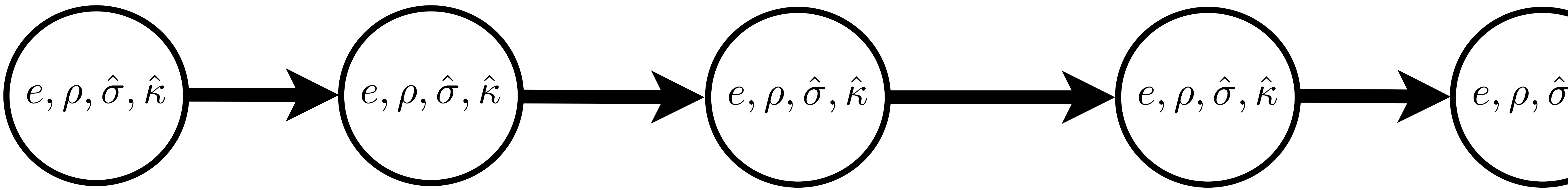
$$\langle v, \hat{\sigma}, (e, \rho), a \rangle \quad \longmapsto \quad \langle e, \rho, \hat{\sigma}, v, a \rangle$$

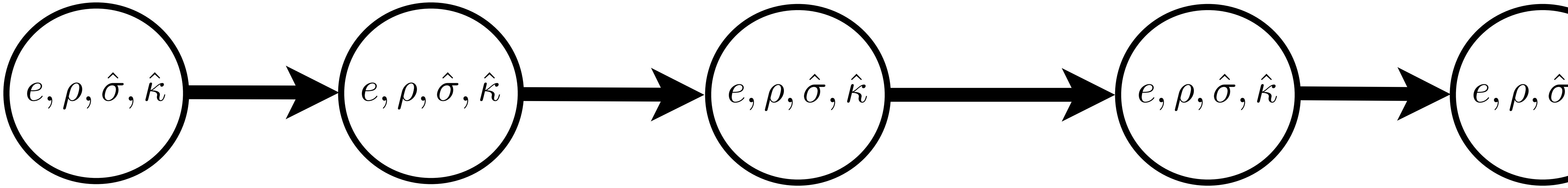
$$\langle v, \hat{\sigma}, (\lambda x.e, \rho), a \rangle \quad \longmapsto \quad \langle e, \rho[x \mapsto a'], \hat{\sigma} \sqcup [a' \mapsto v], \hat{\kappa} \rangle$$

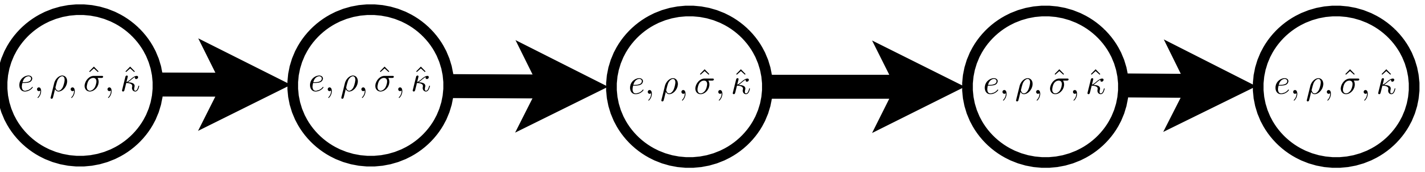
$$\text{if } \hat{\kappa} \in \hat{\sigma}(a)$$

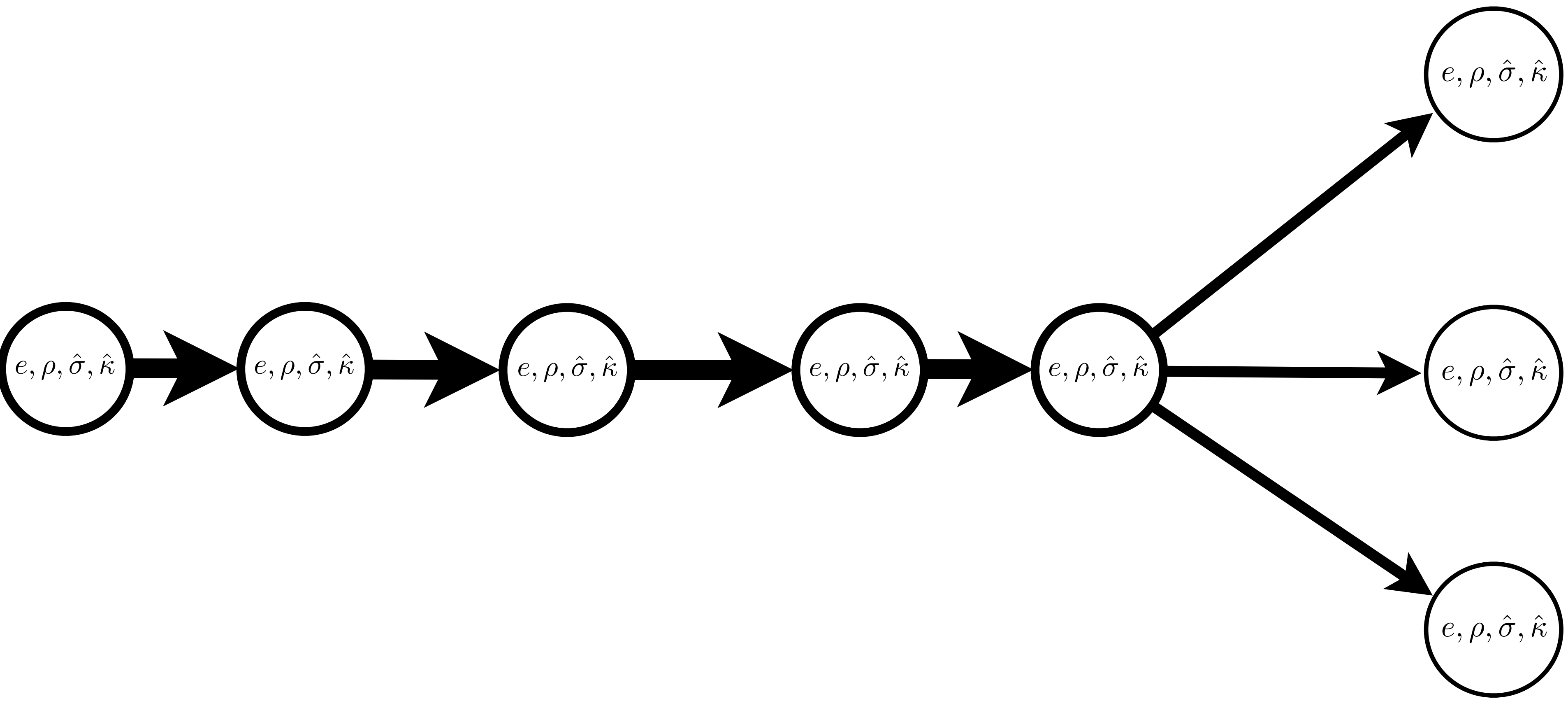

$e, \rho, \widehat{\sigma}, \widehat{\kappa}$

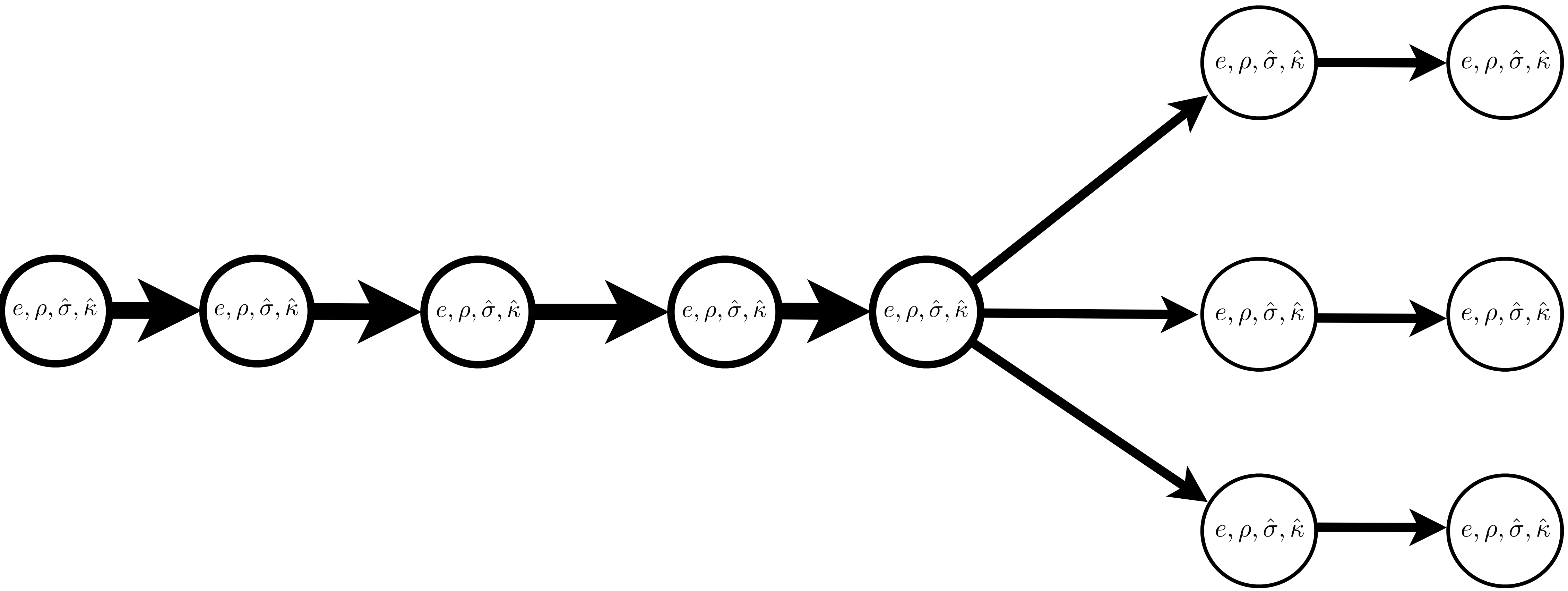

$e, \rho, \hat{\sigma}, \hat{\kappa}$


$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$

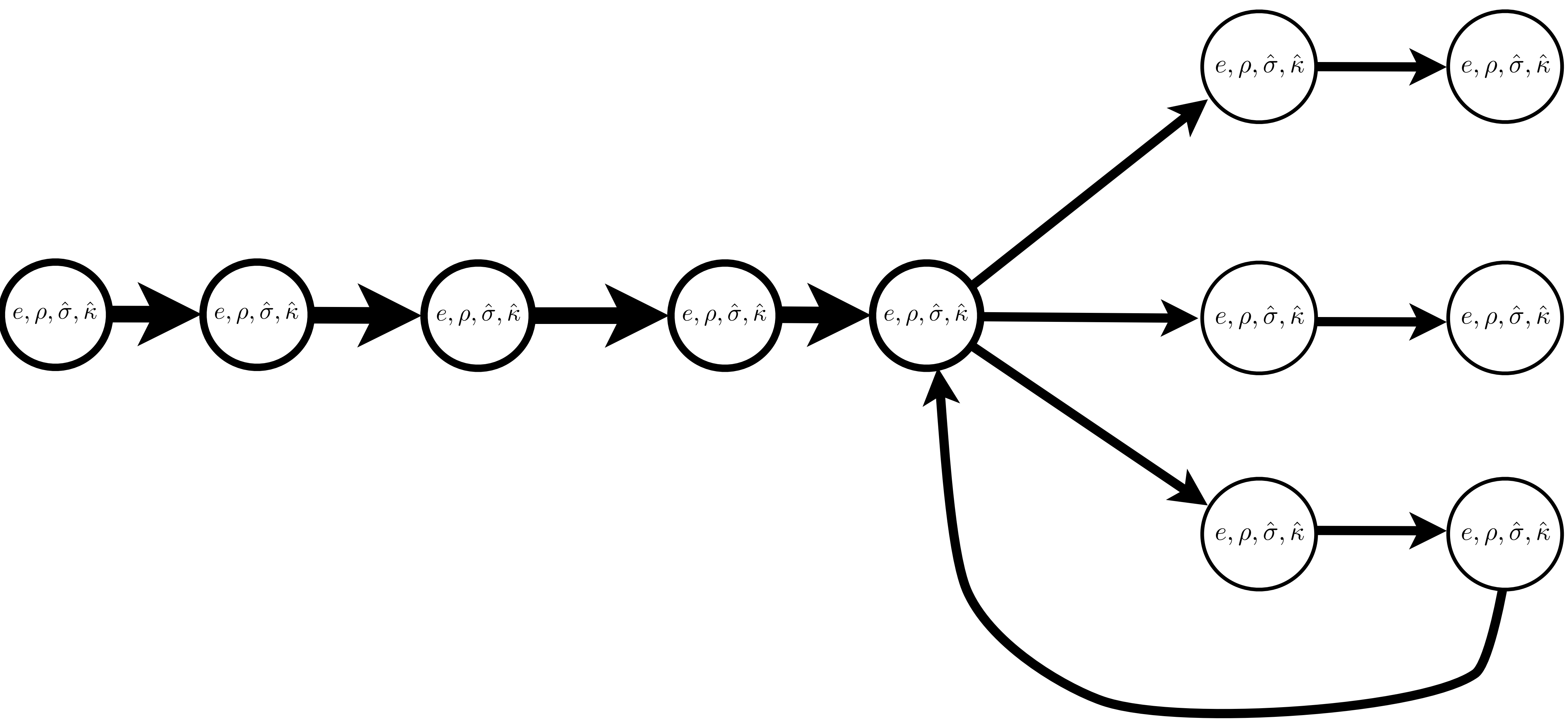

$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$

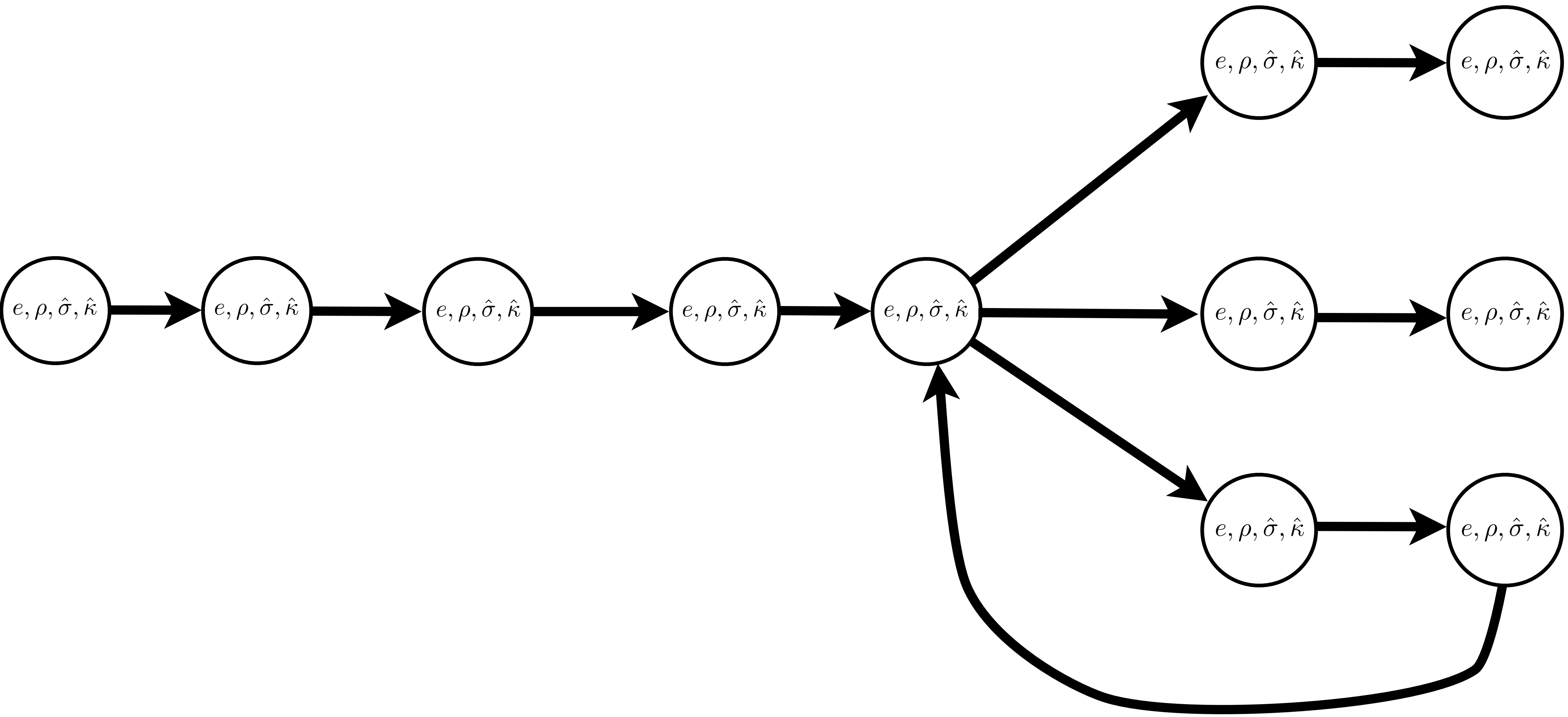

$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$

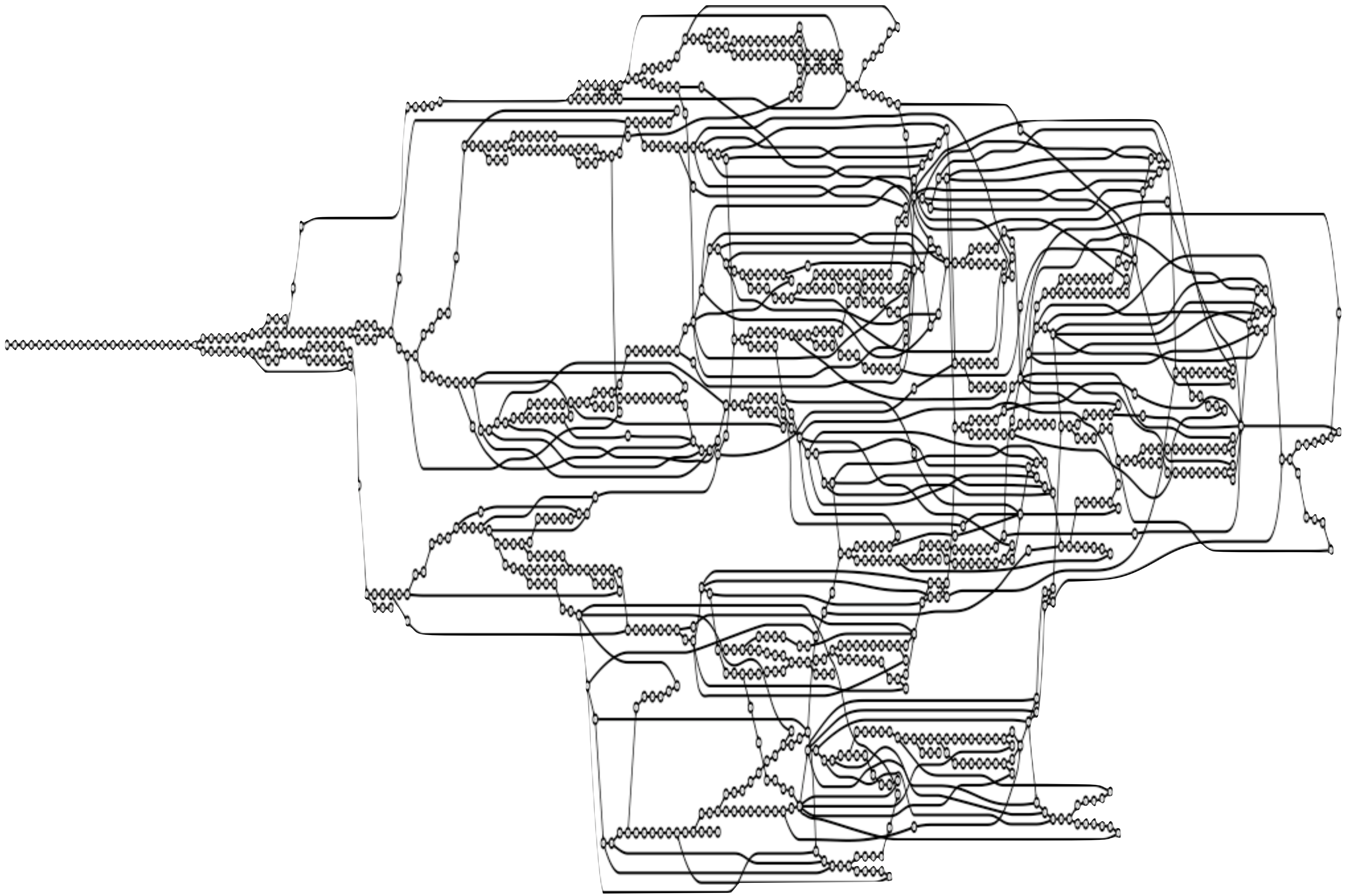

$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$



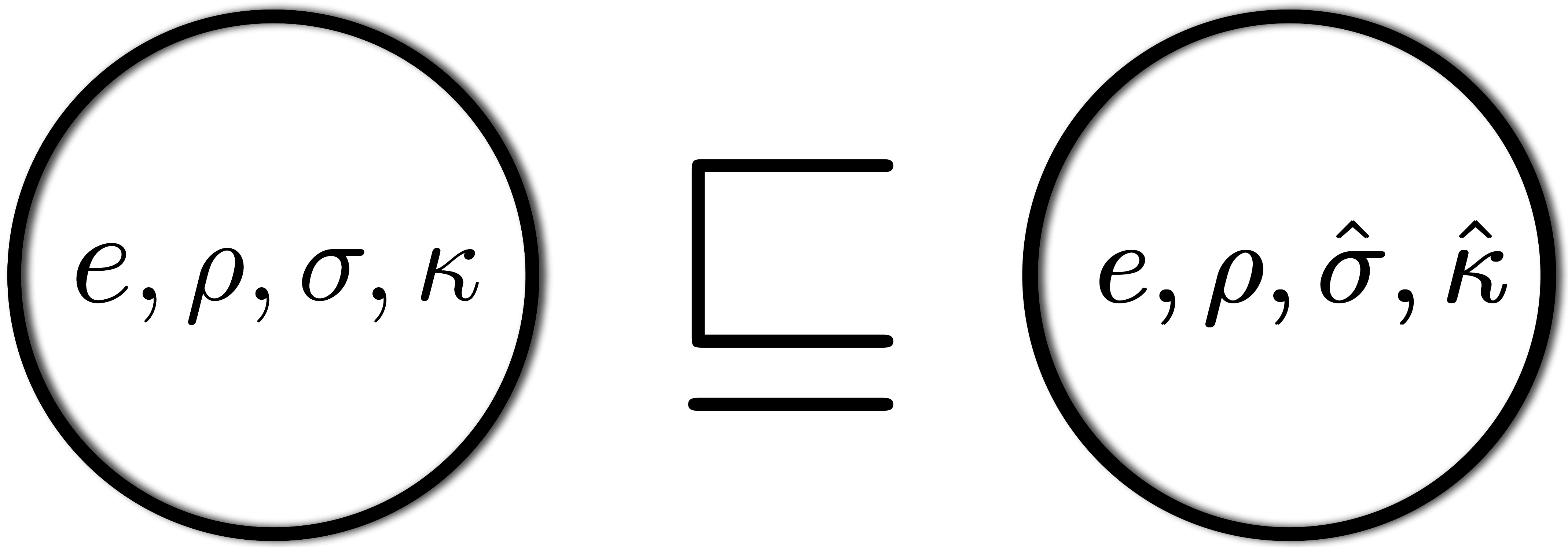
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$

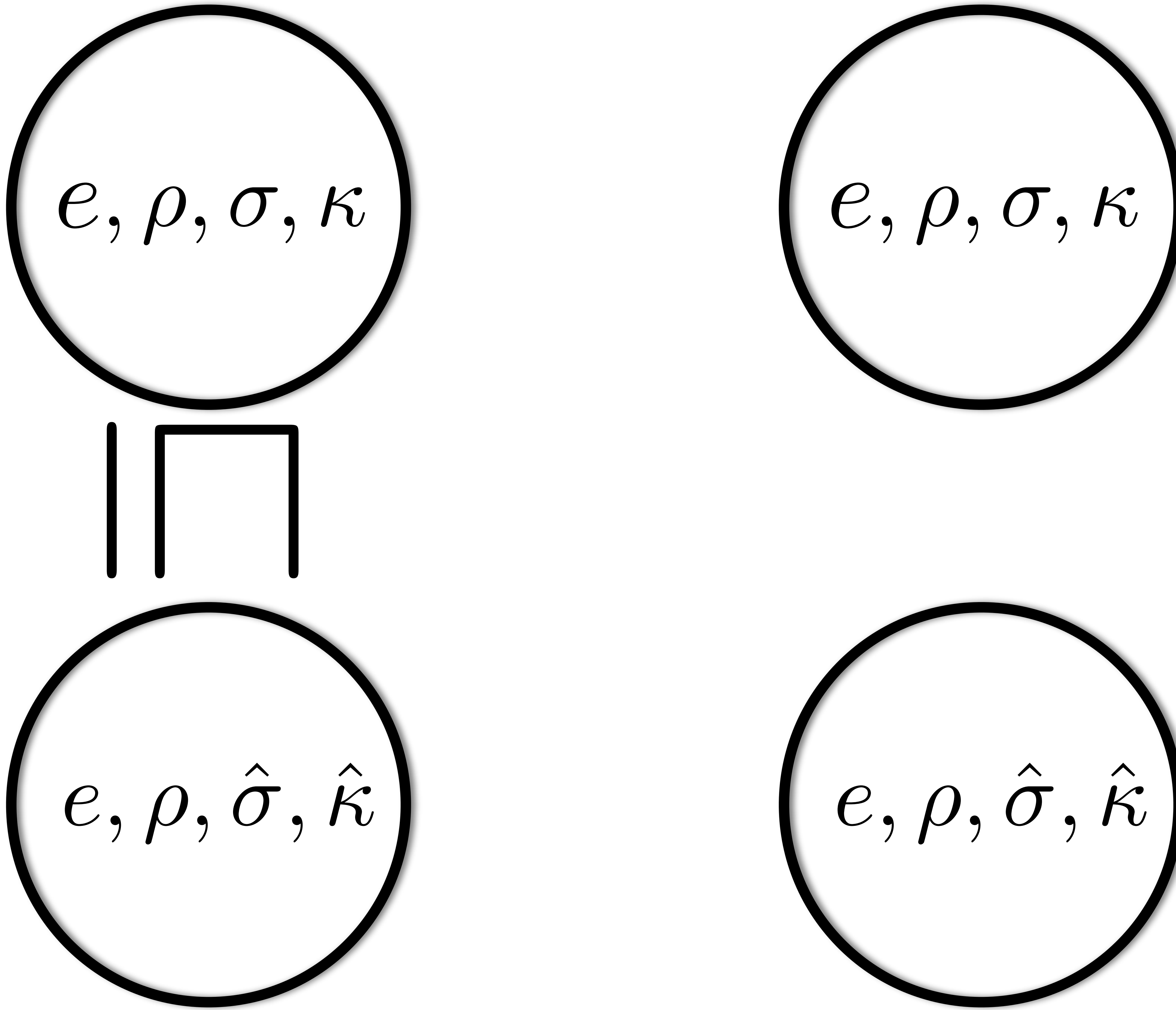

$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$

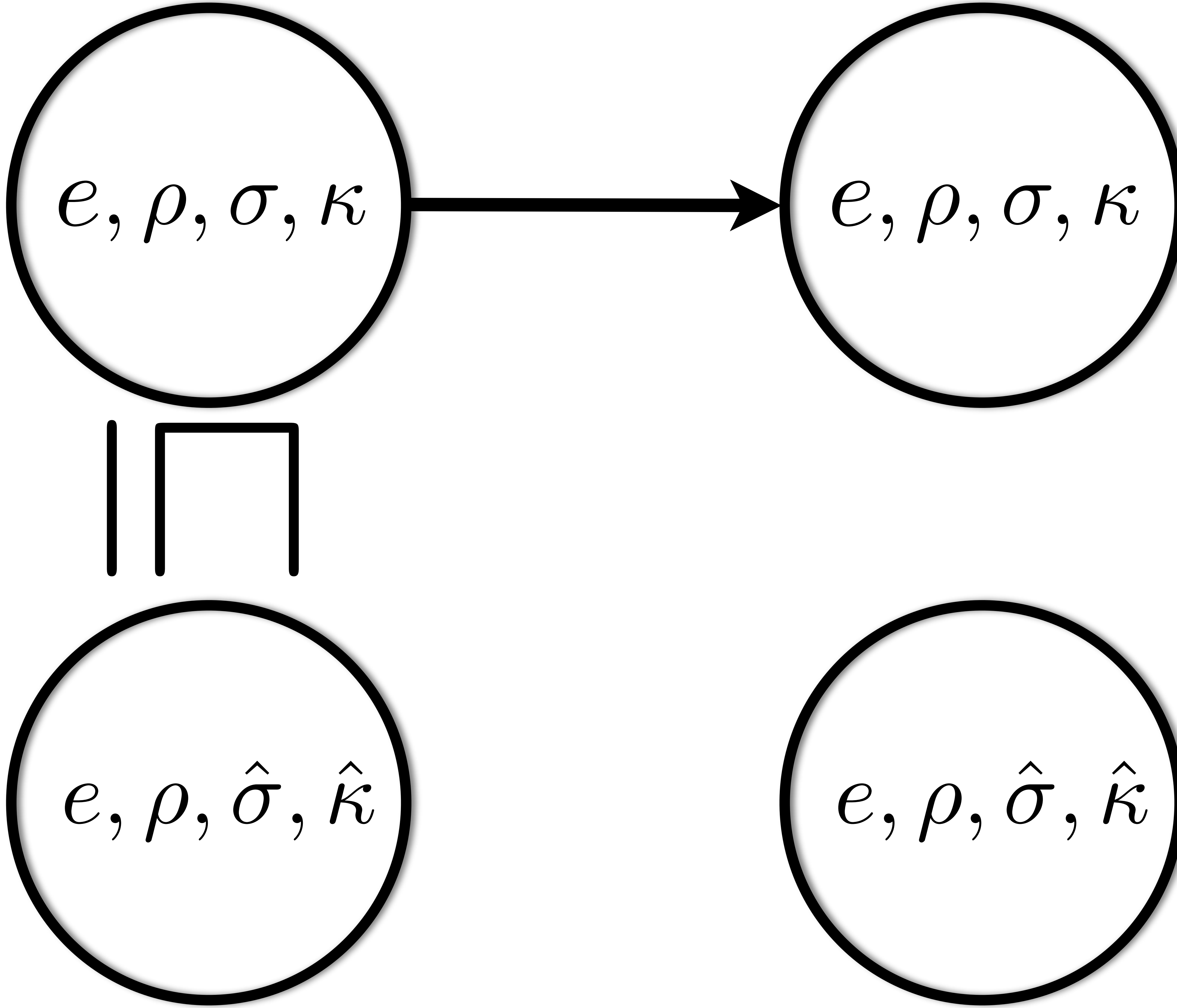

$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$



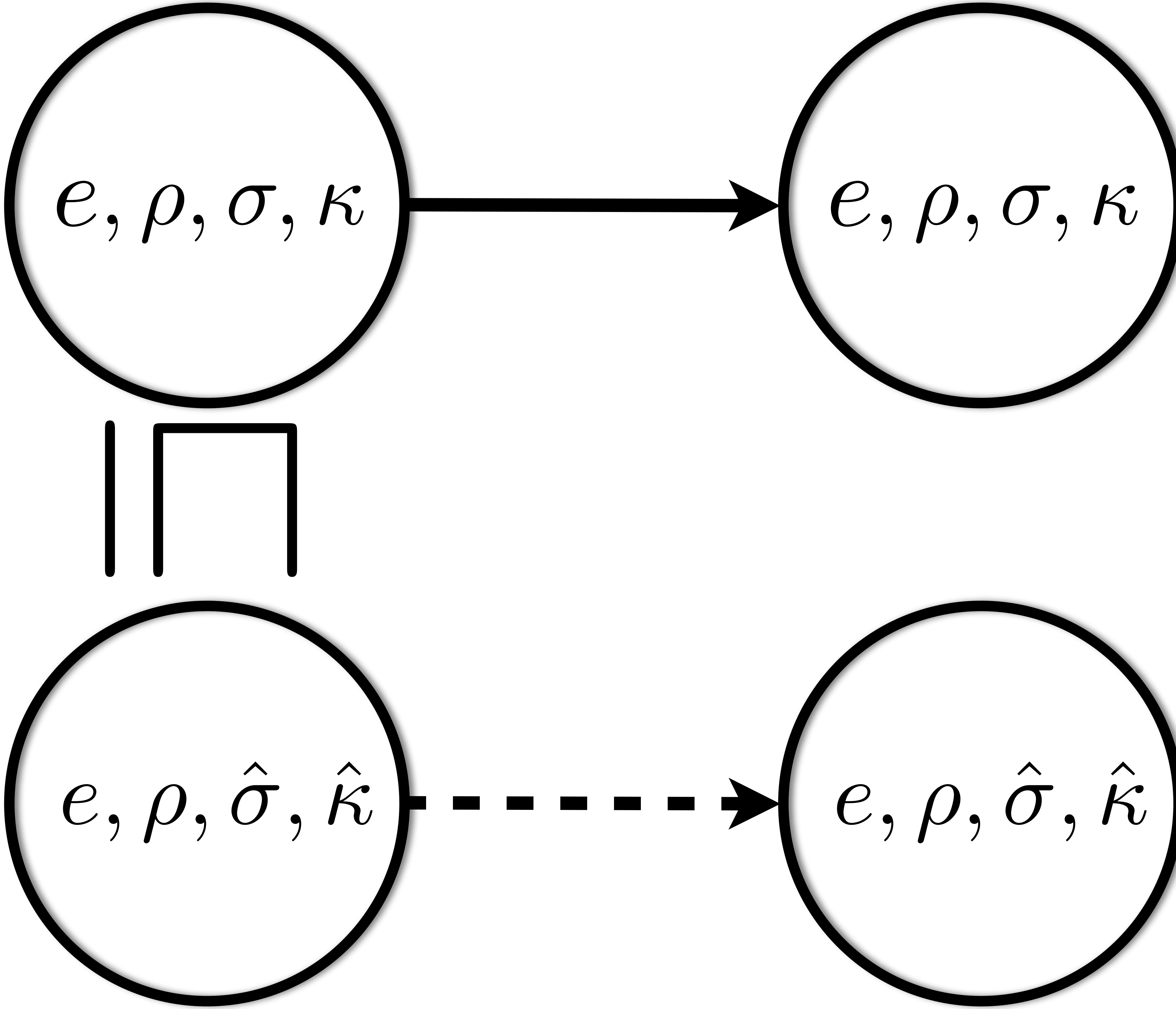
Soundness
(the safety of predictions)

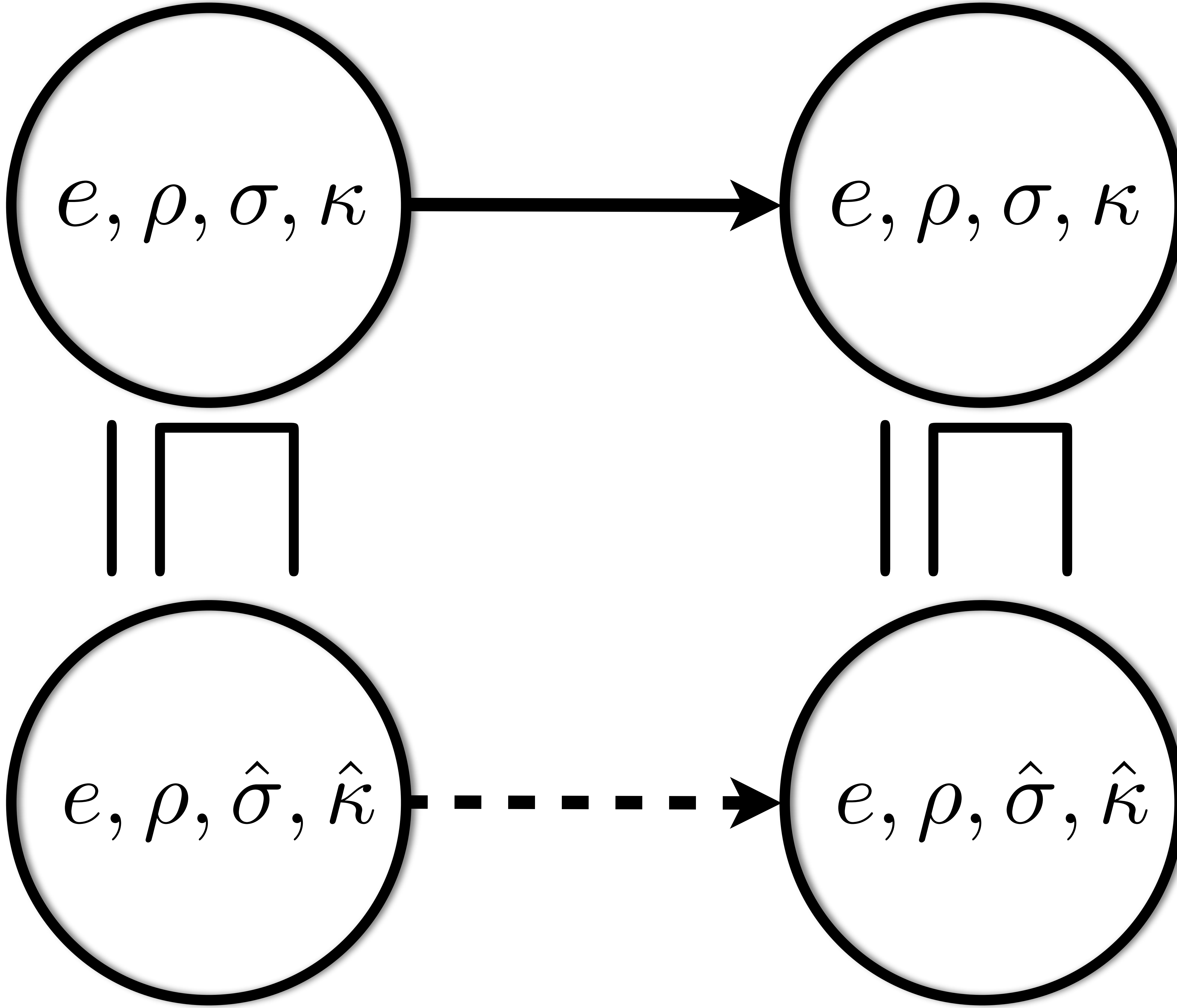

e, ρ, σ, κ
$\sqsubseteq$
$e, \rho, \hat{\sigma}, \hat{\kappa}$

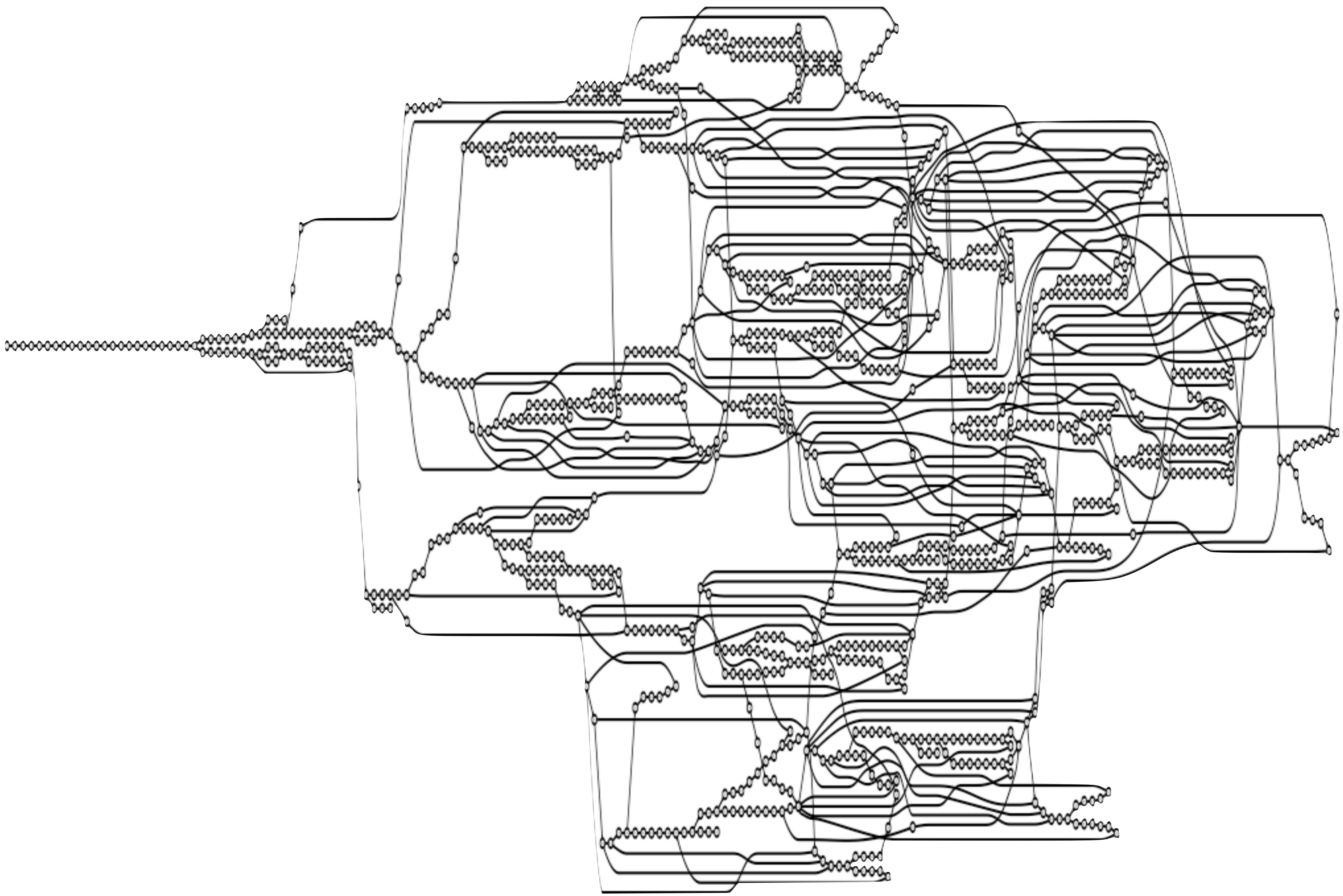

e, ρ, σ, κ
e, ρ, σ, κ
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$

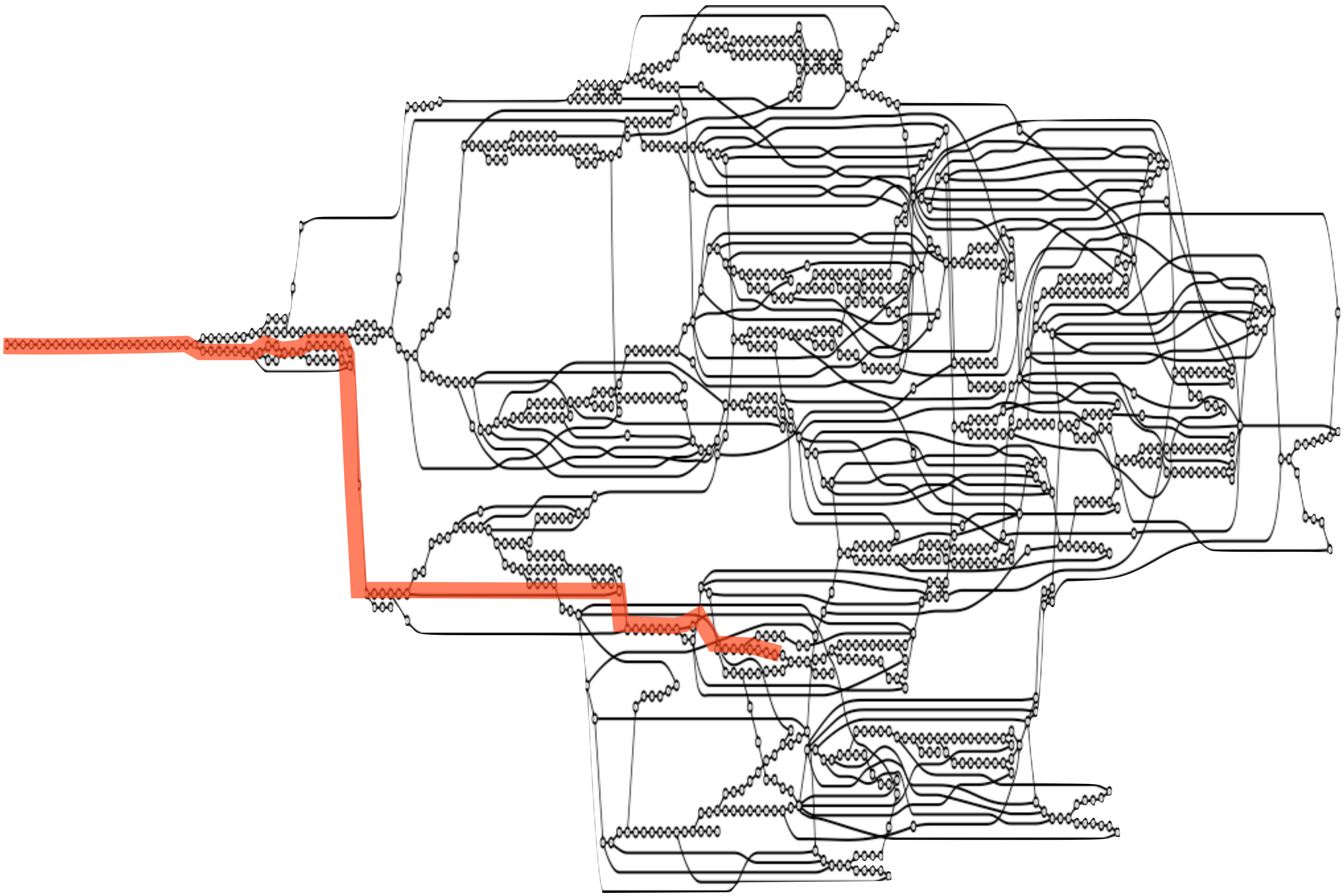


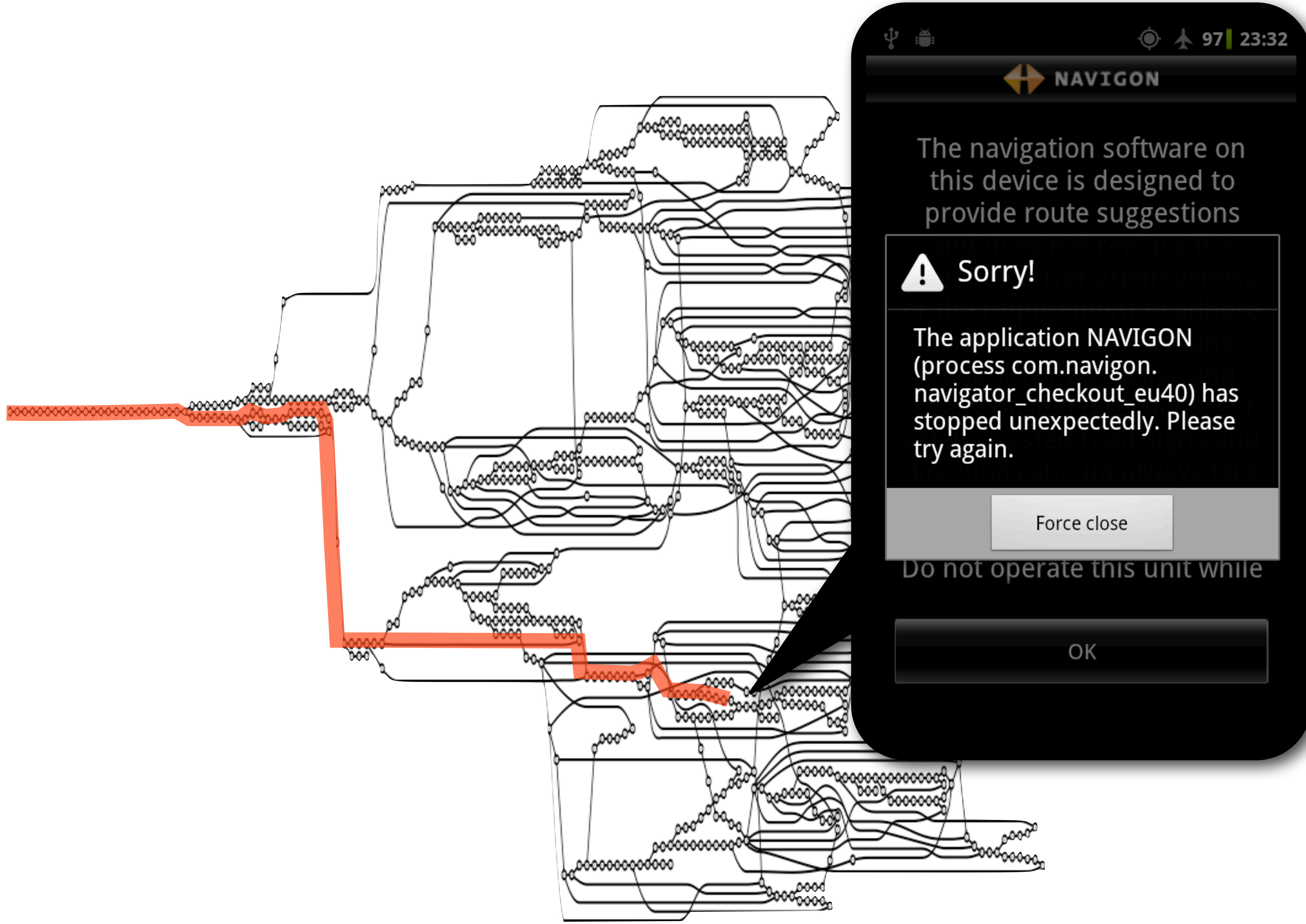
e, ρ, σ, κ
e, ρ, σ, κ
$\sqsubseteq$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$

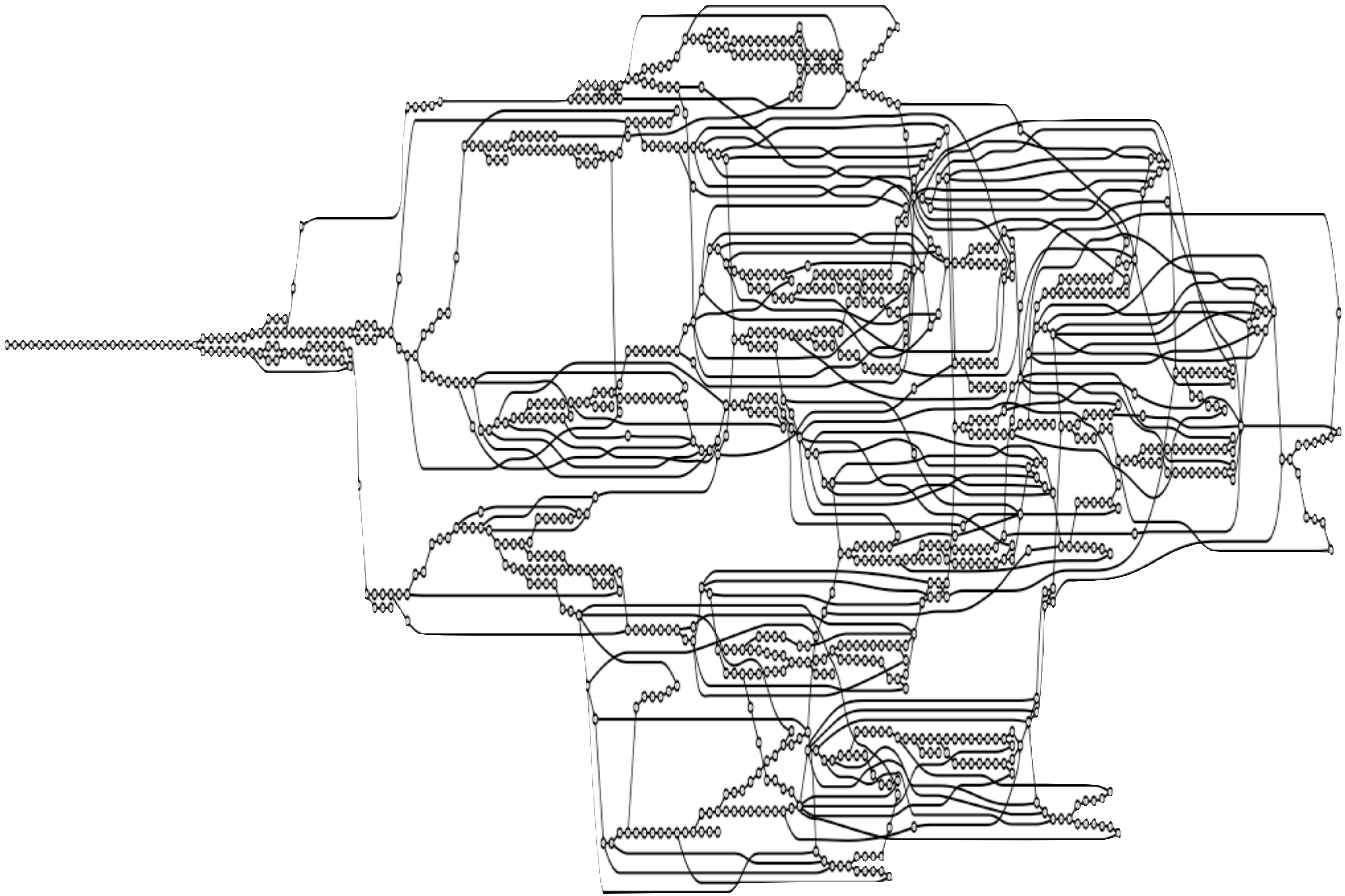

e, ρ, σ, κ
e, ρ, σ, κ
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$

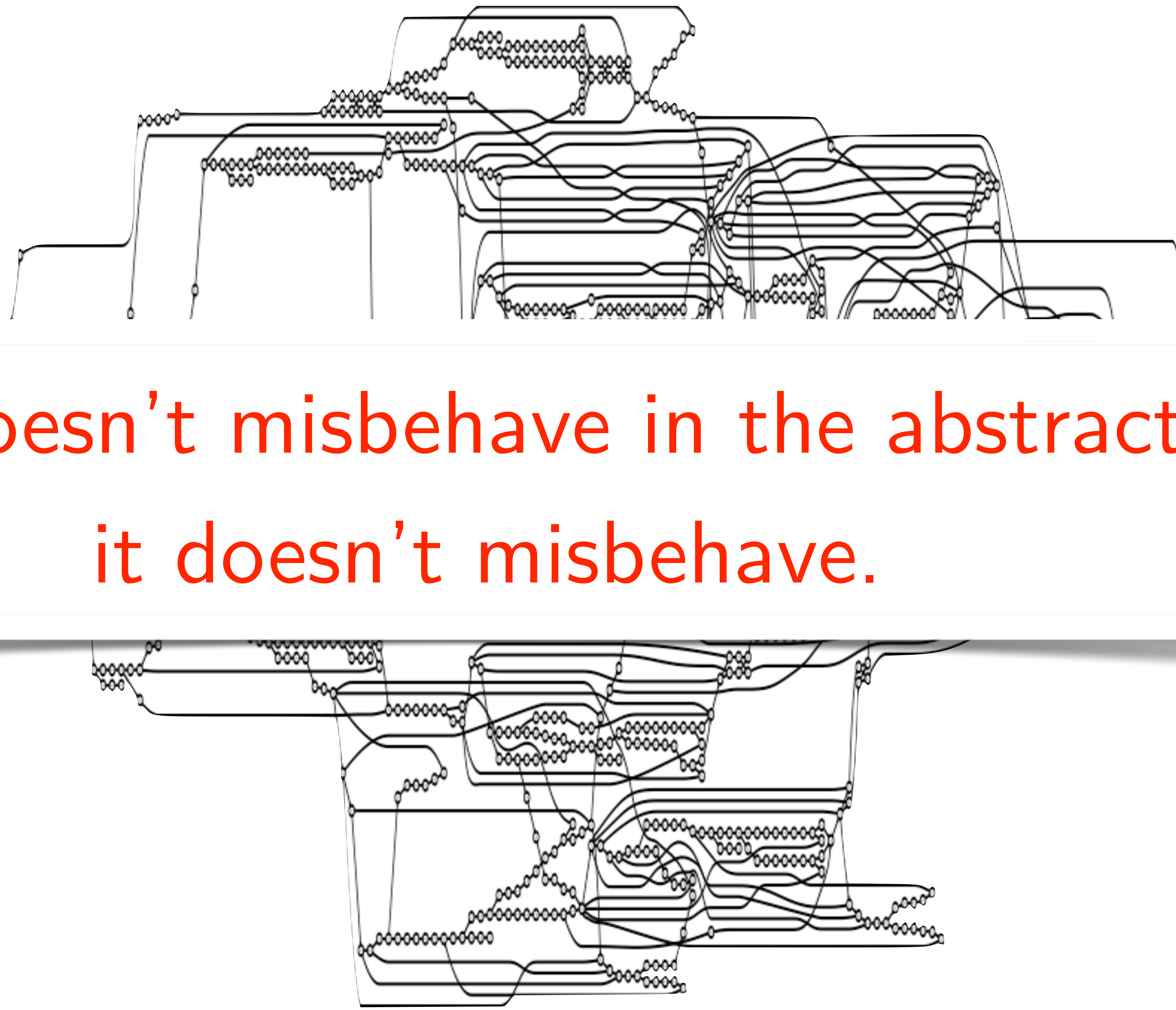

e, ρ, σ, κ
e, ρ, σ, κ
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$






97 23:32
NAVIGON
The navigation software on this device is designed to provide route suggestions
Sorry!
The application NAVIGON (process com.navigon. navigator_checkout_eu40) has stopped unexpectedly. Please try again.
Force close
Do not operate this unit while
OK

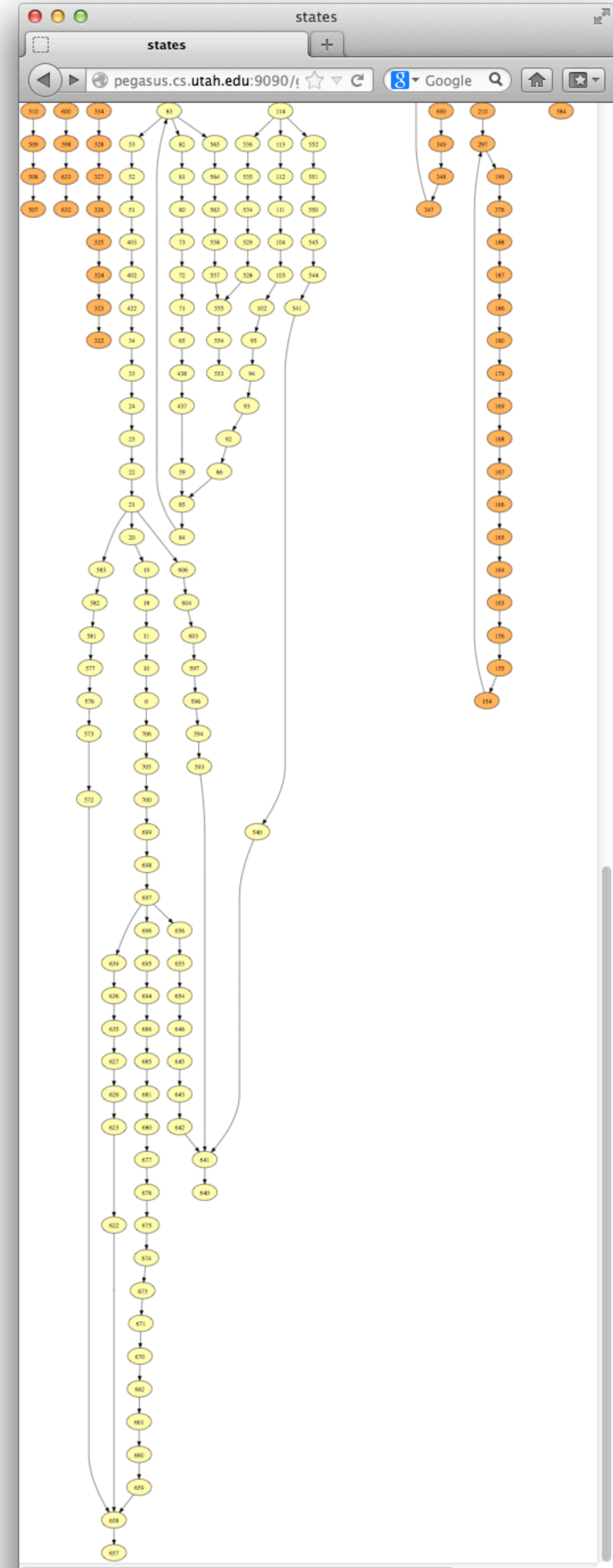


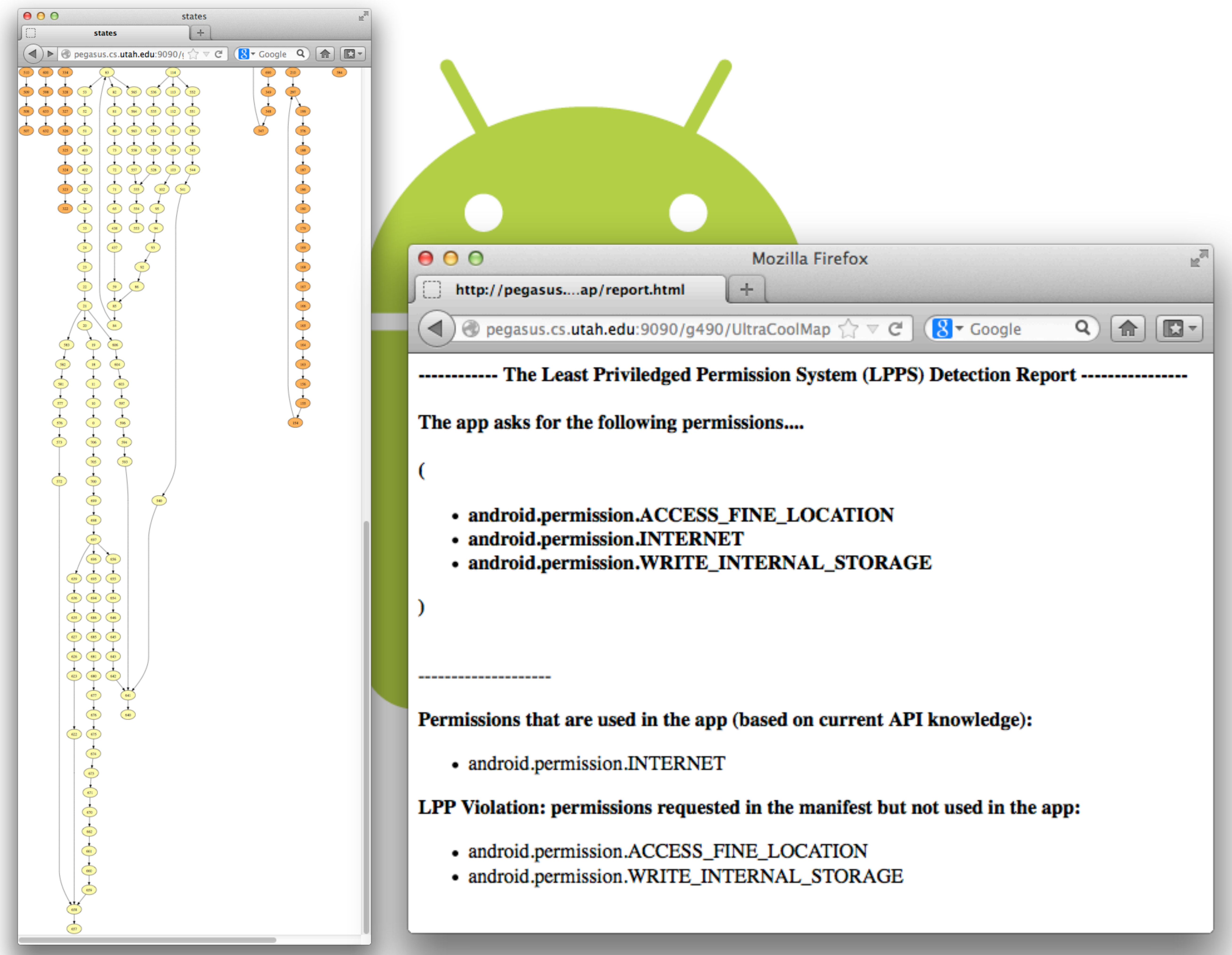


If it doesn't misbehave in the abstract,

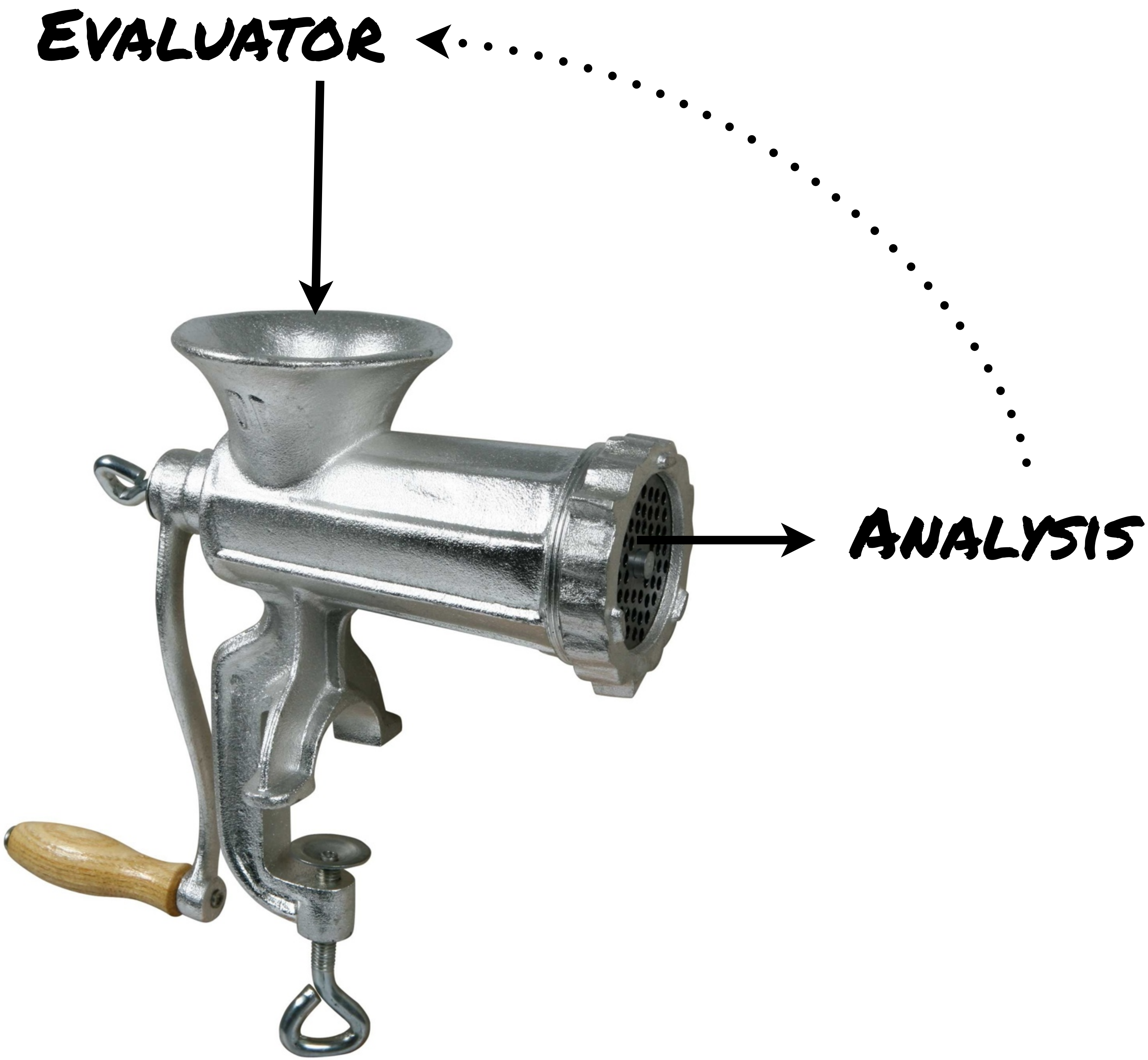
it doesn't misbehave.

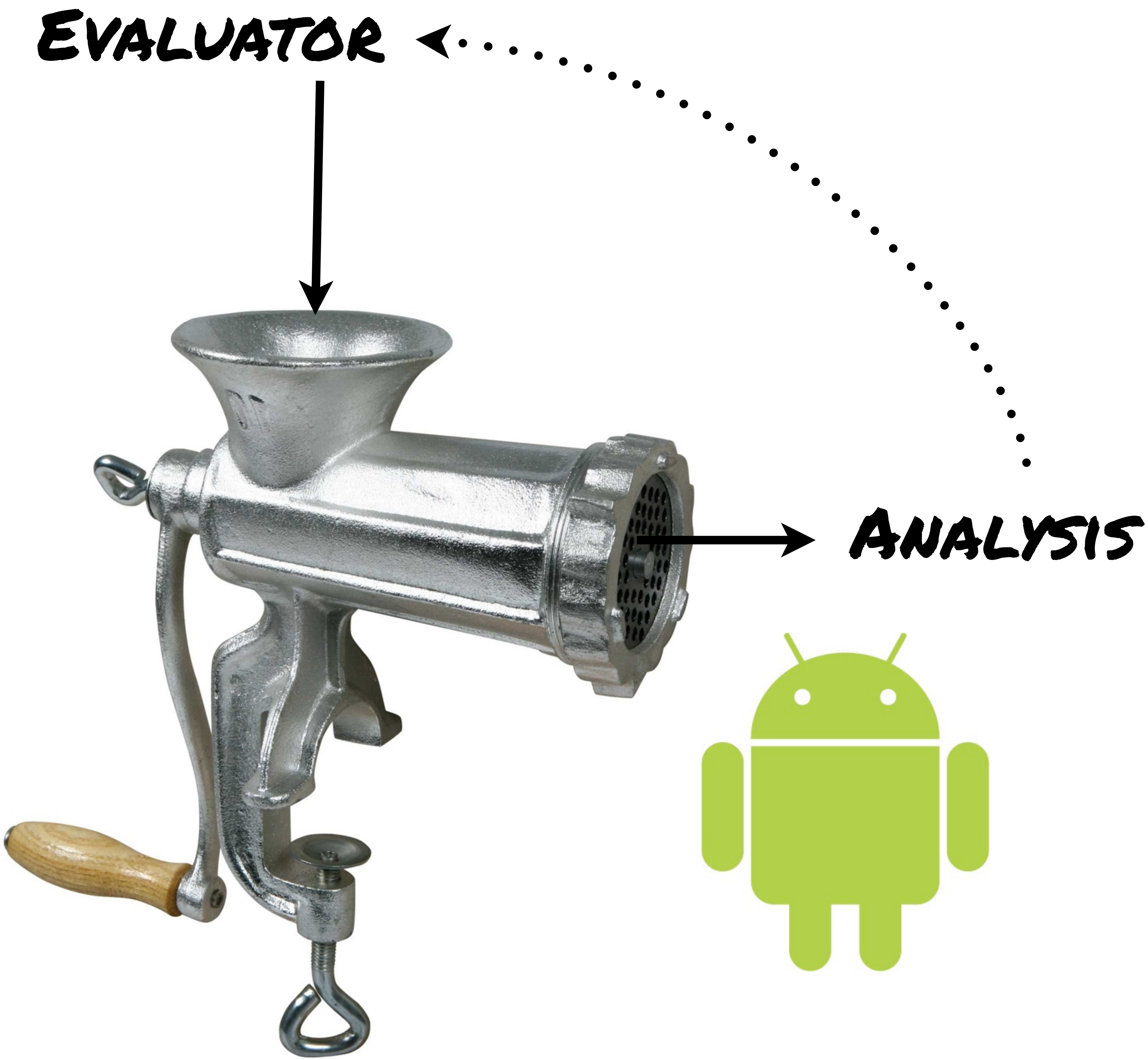
$$e ::= x \mid e\ e \mid \lambda x.e$$

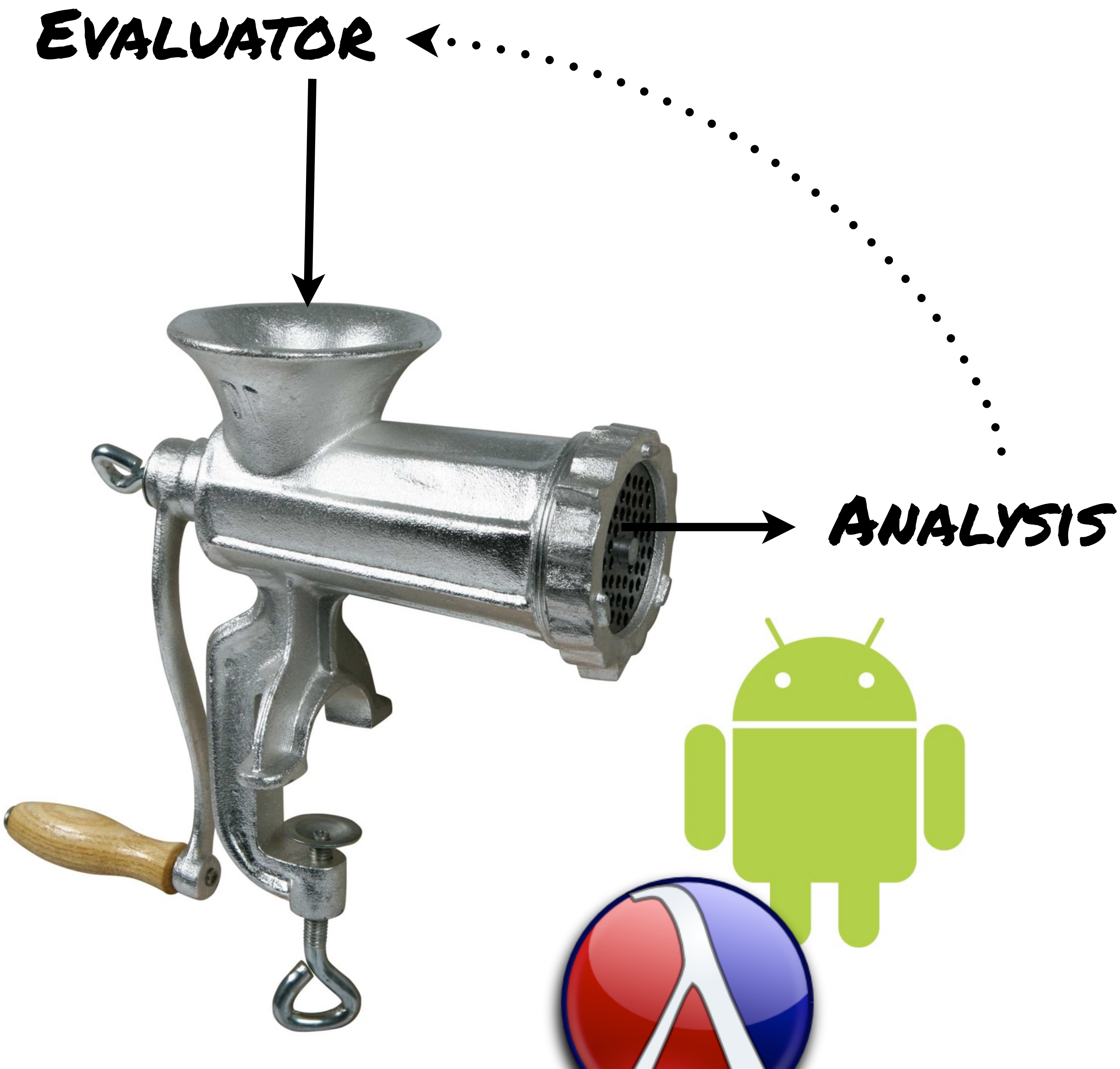

states
states
pegasus.cs.utah.edu:9090/
Google

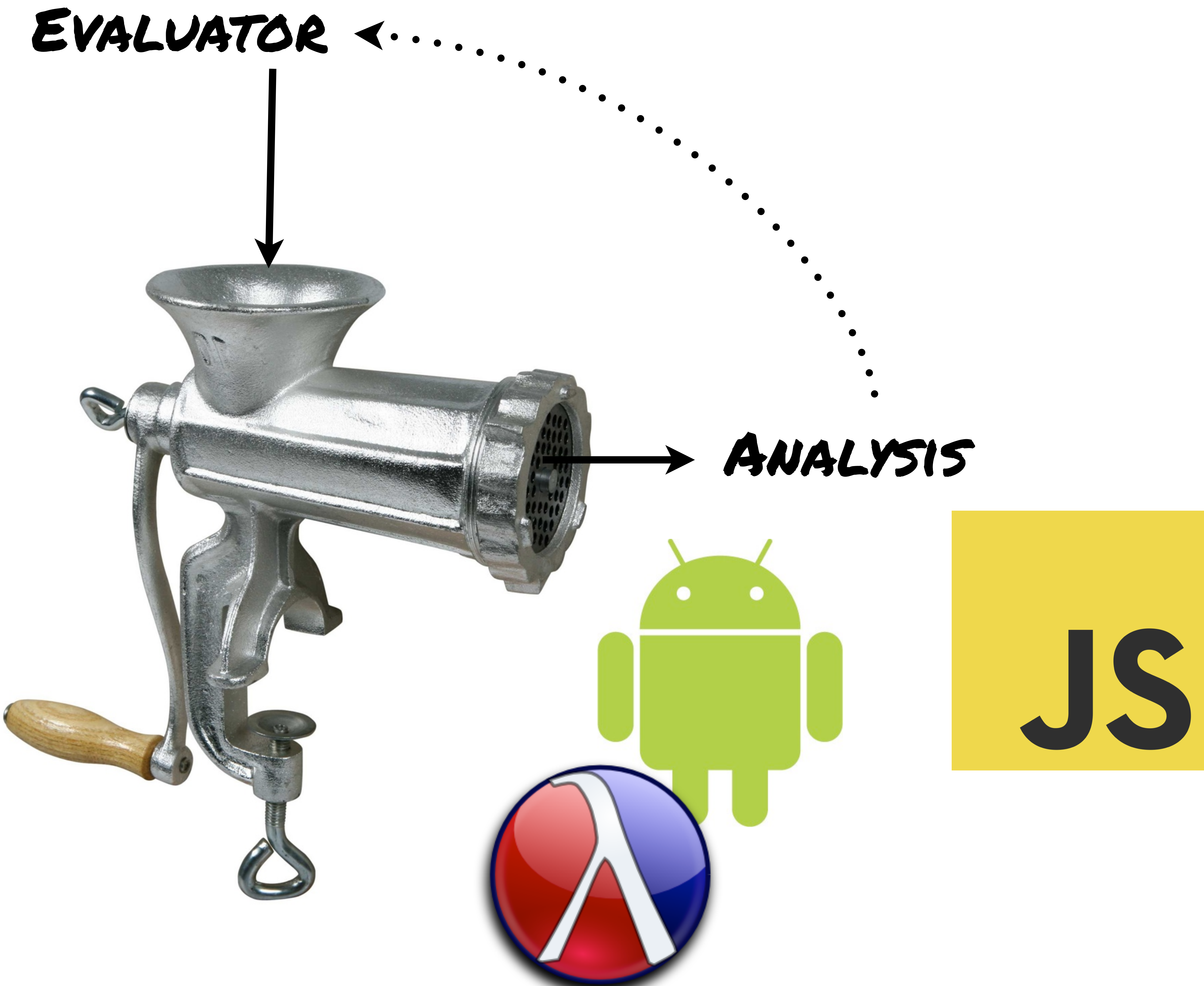

states
pegasus.cs.utah.edu:9090/
Mozilla Firefox
http://pegasus....ap/report.html
pegasus.cs.utah.edu:9090/g490/UltraCoolMap
Google
------------ The Least Priviledged Permission System (LPPS) Detection Report ----------------
The app asks for the following permissions....
(
• android.permission.ACCESS_FINE_LOCATION
• android.permission.INTERNET
• android.permission.WRITE_INTERNAL_STORAGE
)

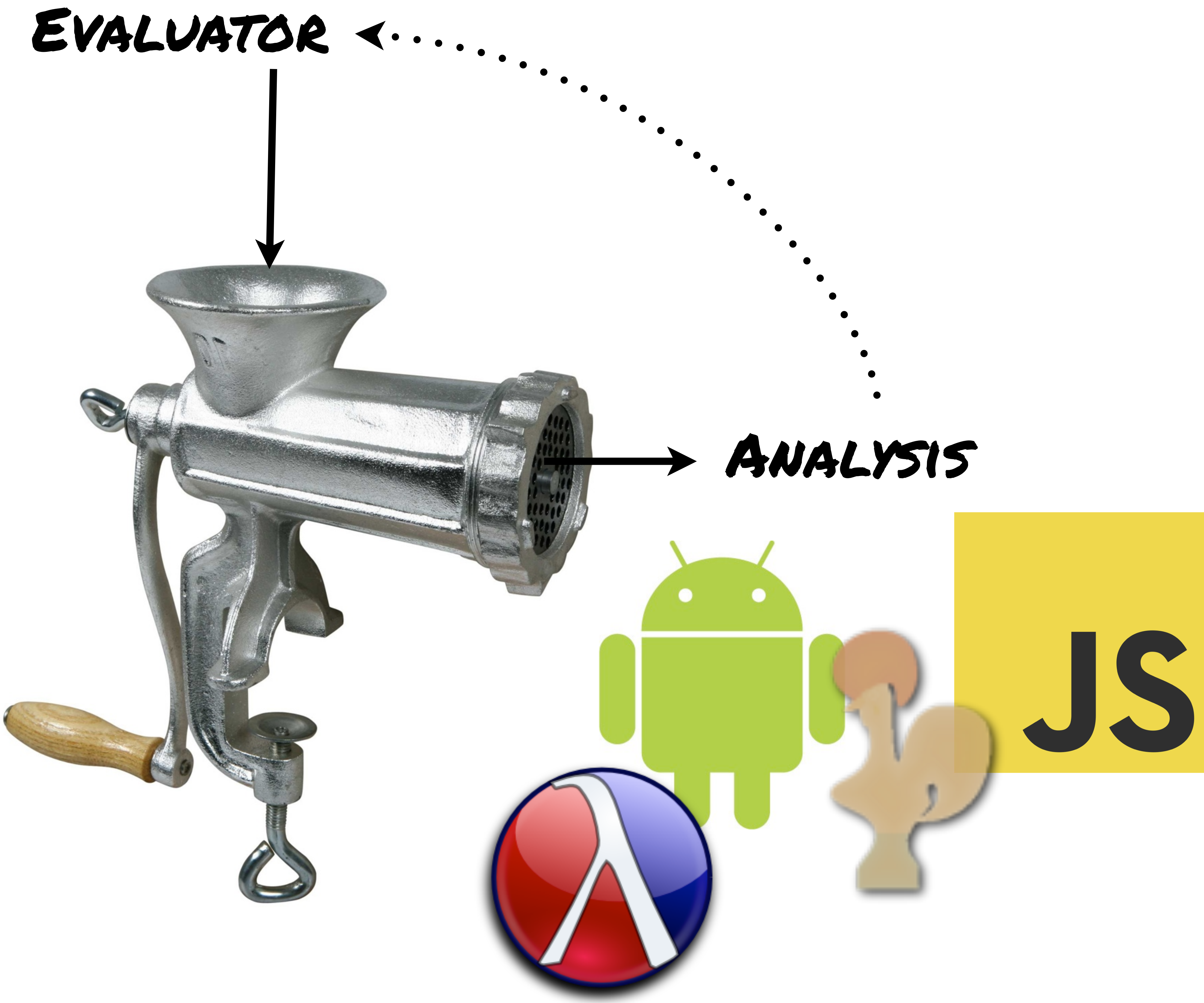
Permissions that are used in the app (based on current API knowledge):
• android.permission.INTERNET
LPP Violation: permissions requested in the manifest but not used in the app:
• android.permission.ACCESS_FINE_LOCATION
• android.permission.WRITE_INTERNAL_STORAGE

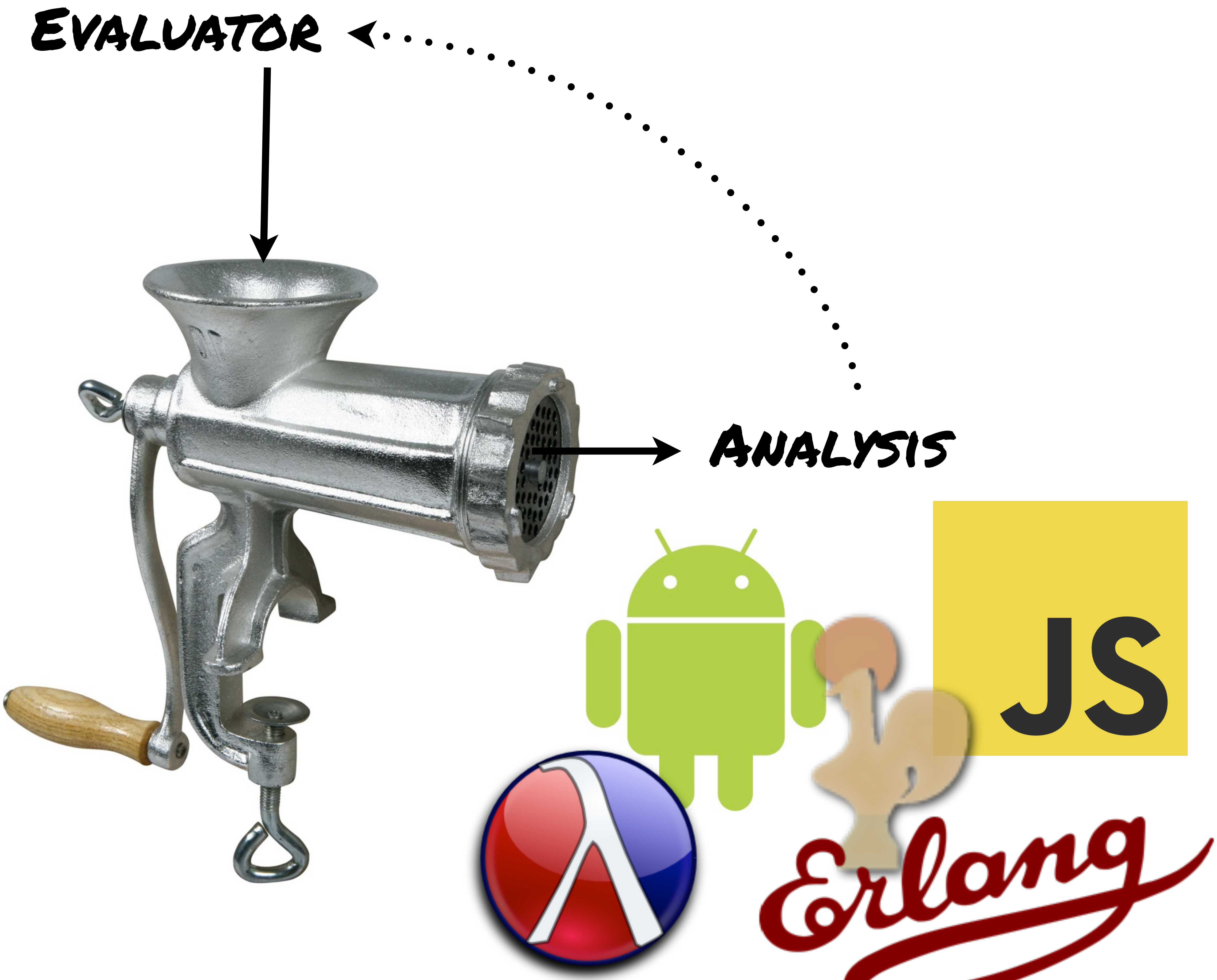

Evaluator
Analysis


Evaluator
Analysis


Evaluator
Analysis


Evaluator
Analysis
JS


Evaluator
Analysis
JS


Evaluator
Analysis
JS
Erlang

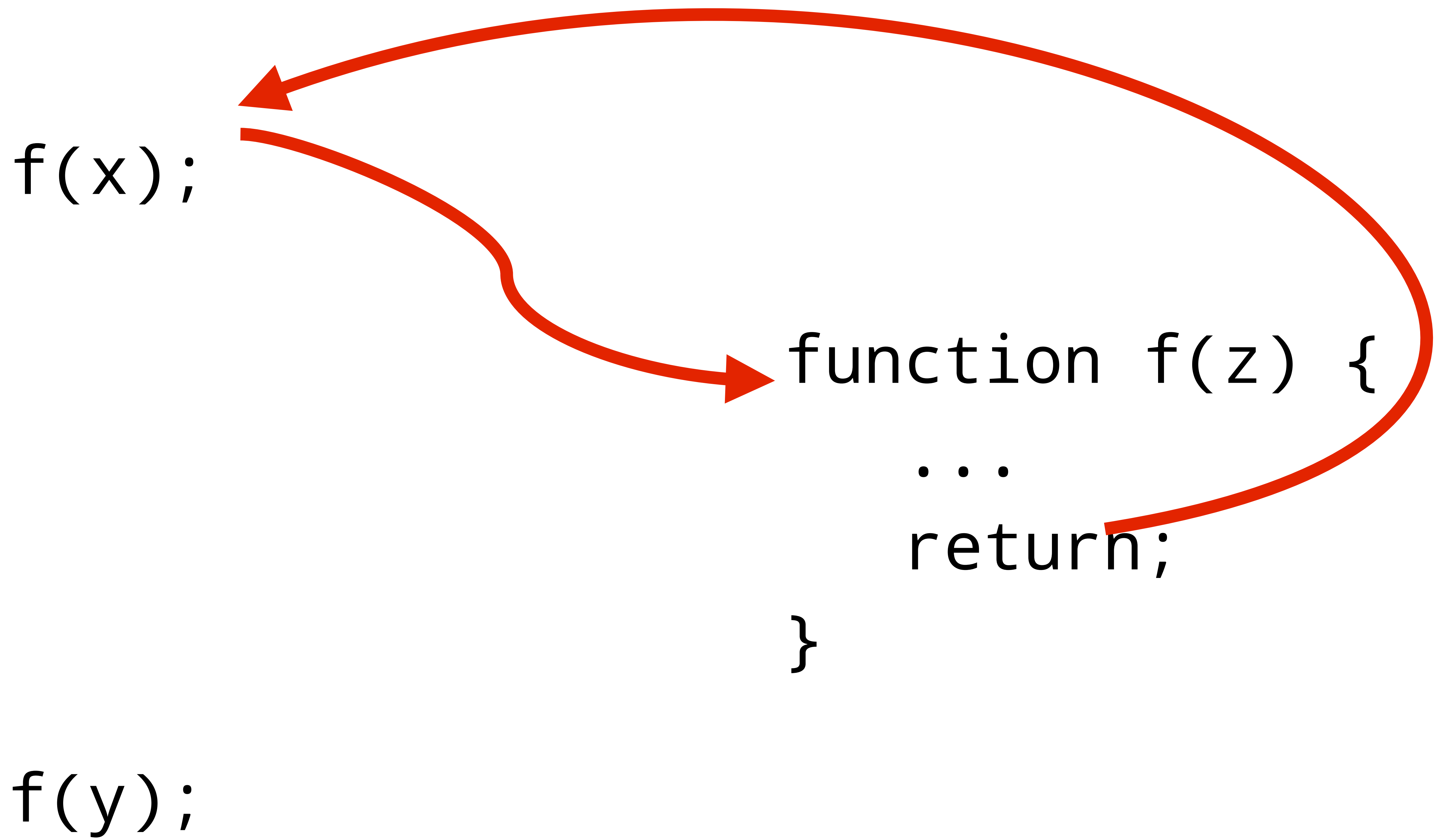
Improving precision

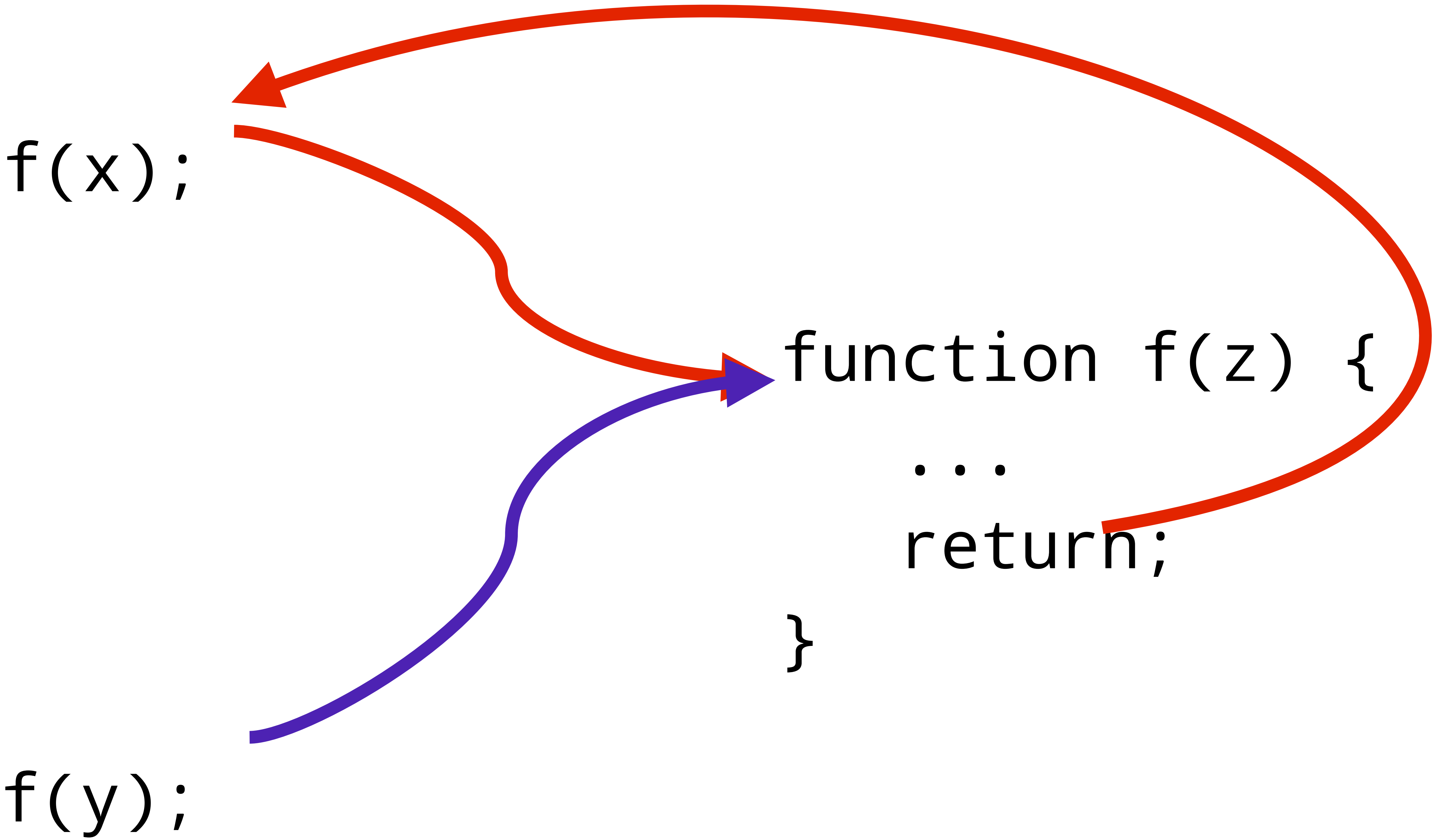
```
f(x);

                    function f(z) {
                        ...
                        return;
                    }

f(y);
```

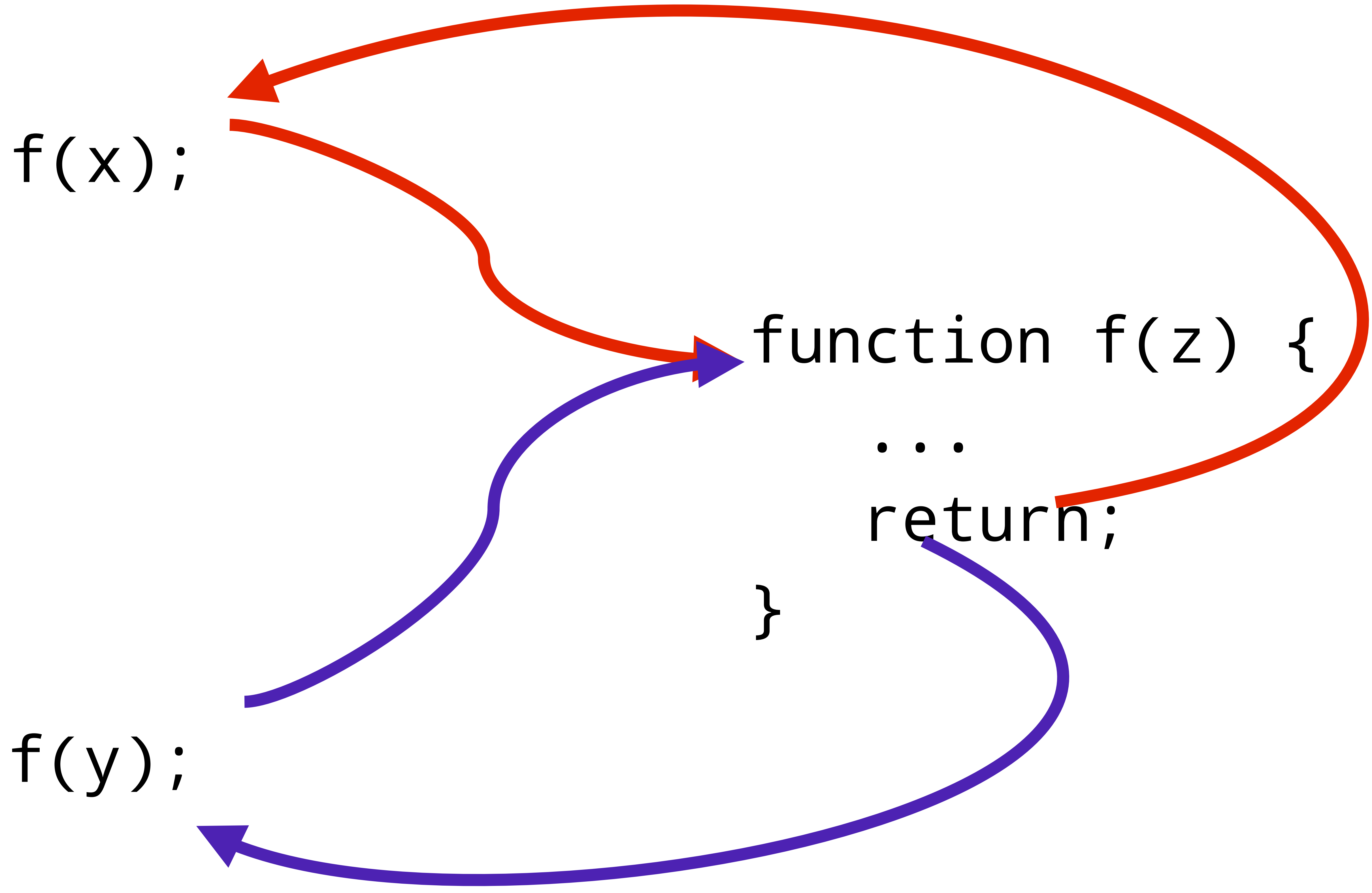
```
f(x);
```

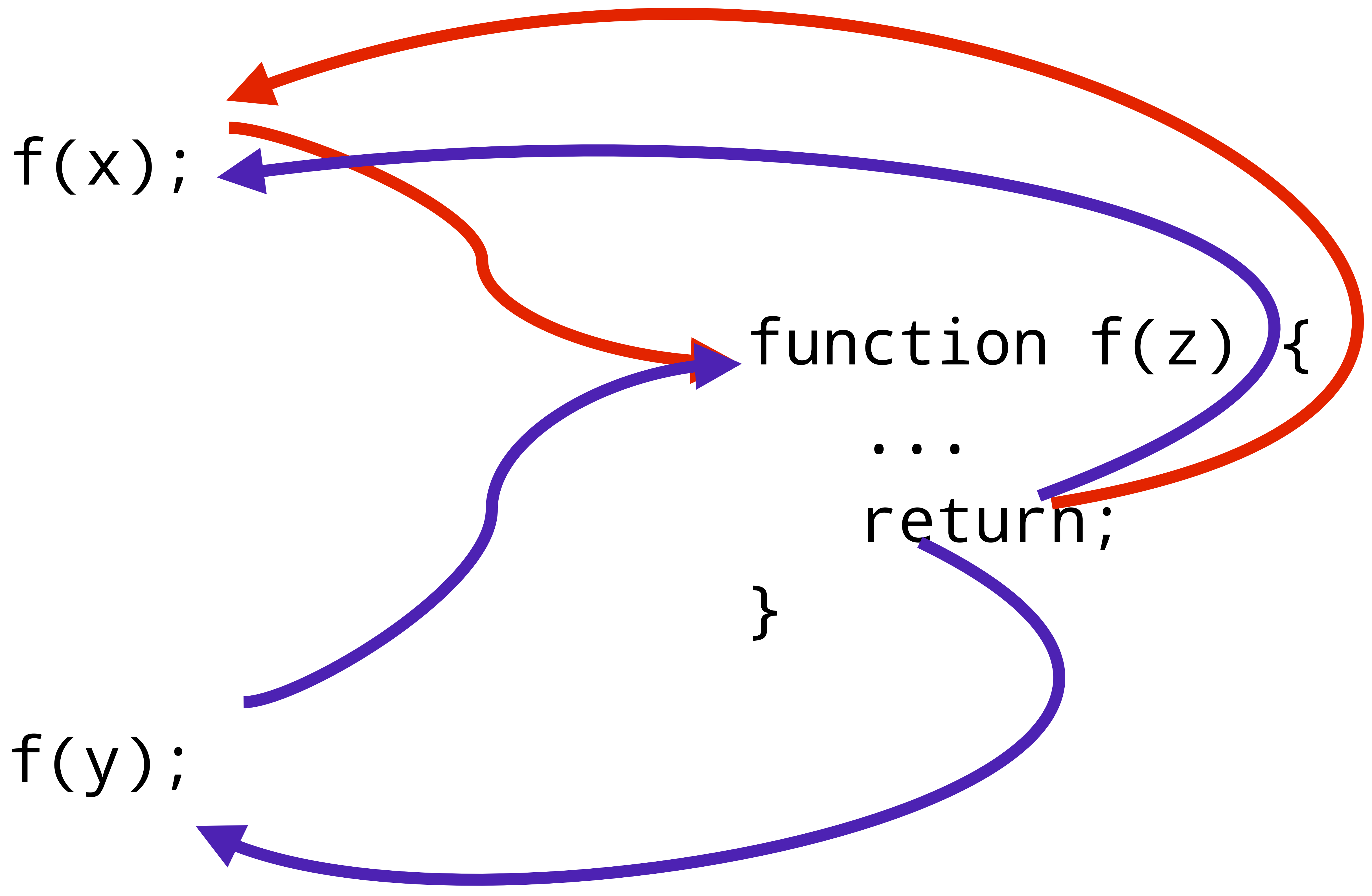
```
function f(z) {
    ...
    return;
}
```

```
f(y);
```


f(x);
function f(z) {
...
return;
}
f(y);

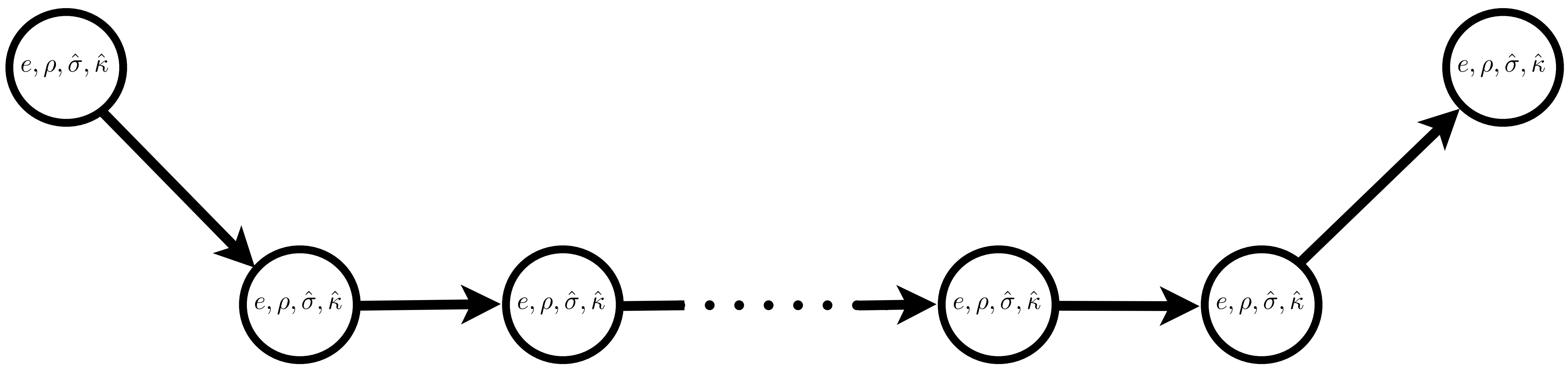

f(x);
function f(z) {
...
return;
}
f(y);


f(x);
function f(z) {
...
return;
}
f(y);

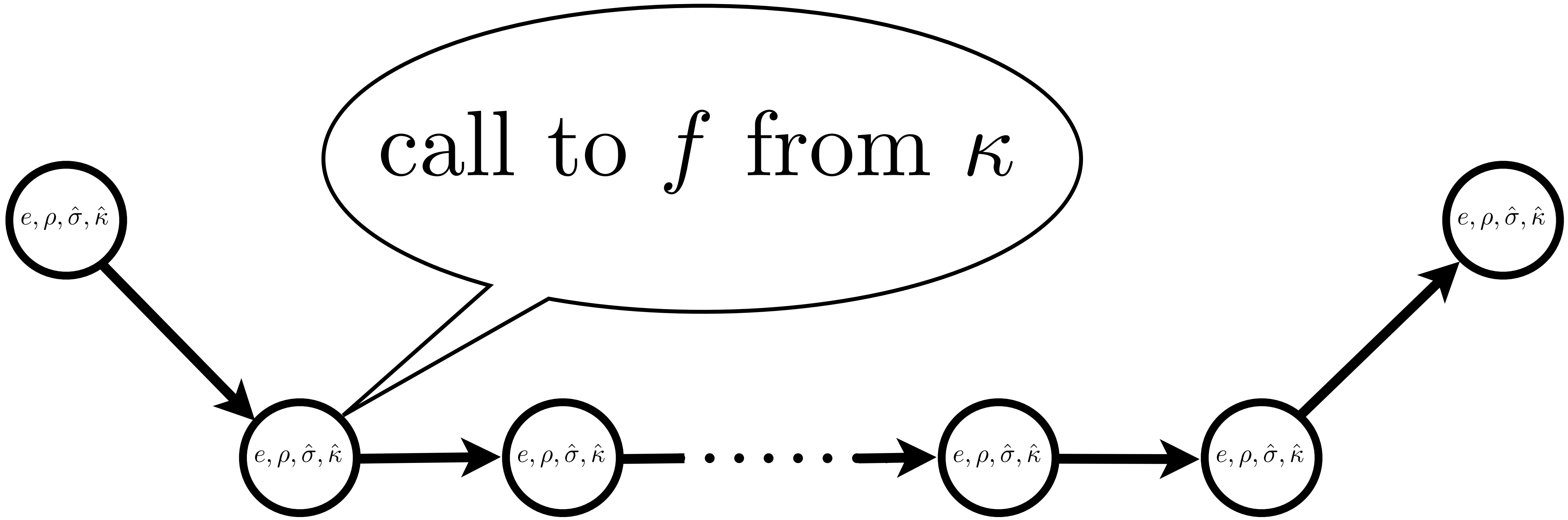


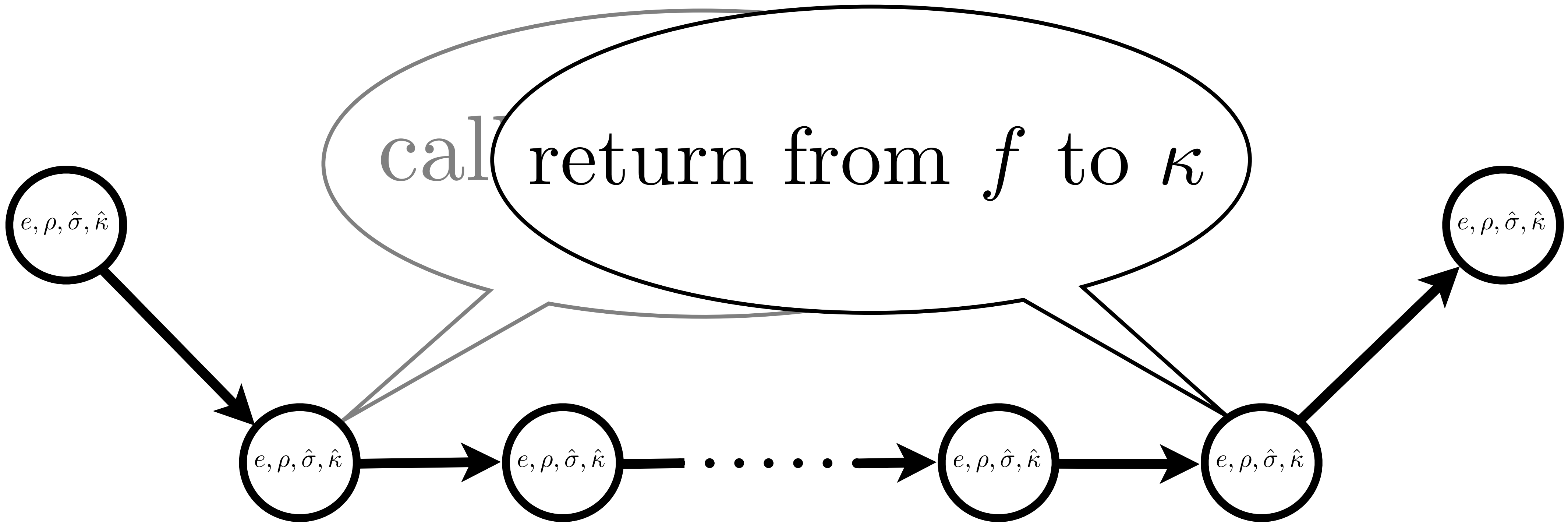
f(x);
function f(z) {
...
return;
}
f(y);

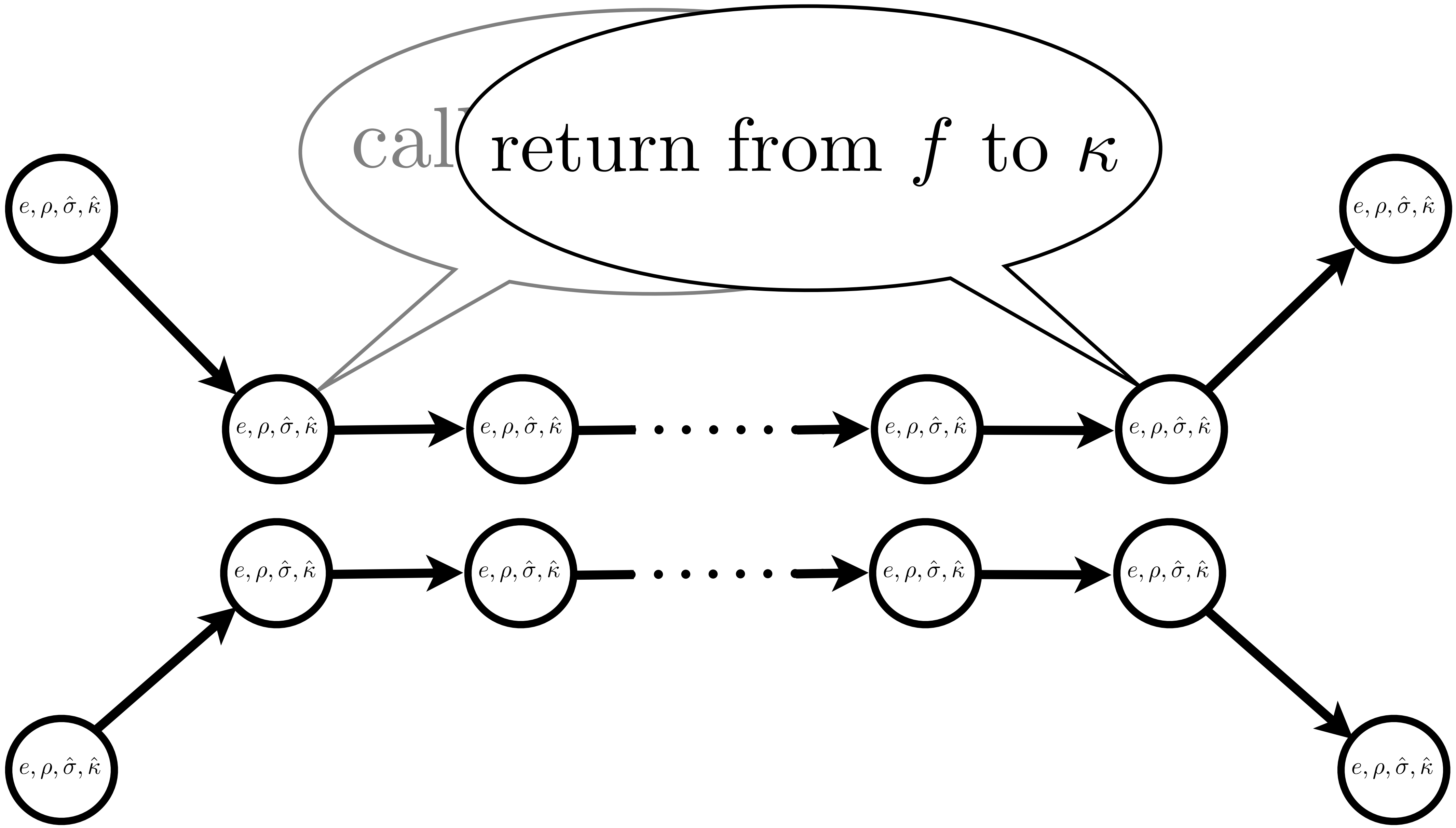

$e, \rho, \hat{\sigma}, \hat{\kappa}$



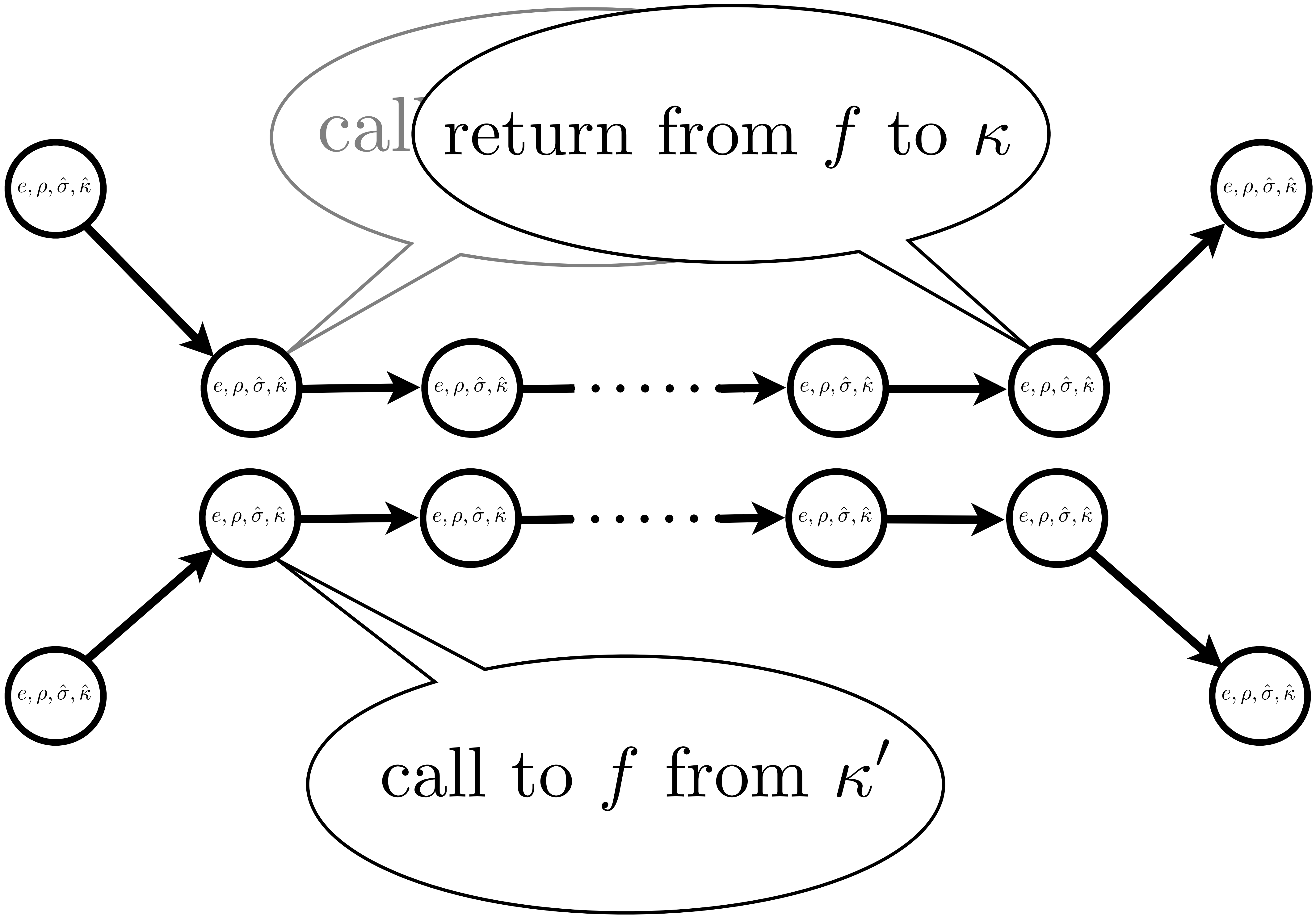
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$


call to f from κ
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$

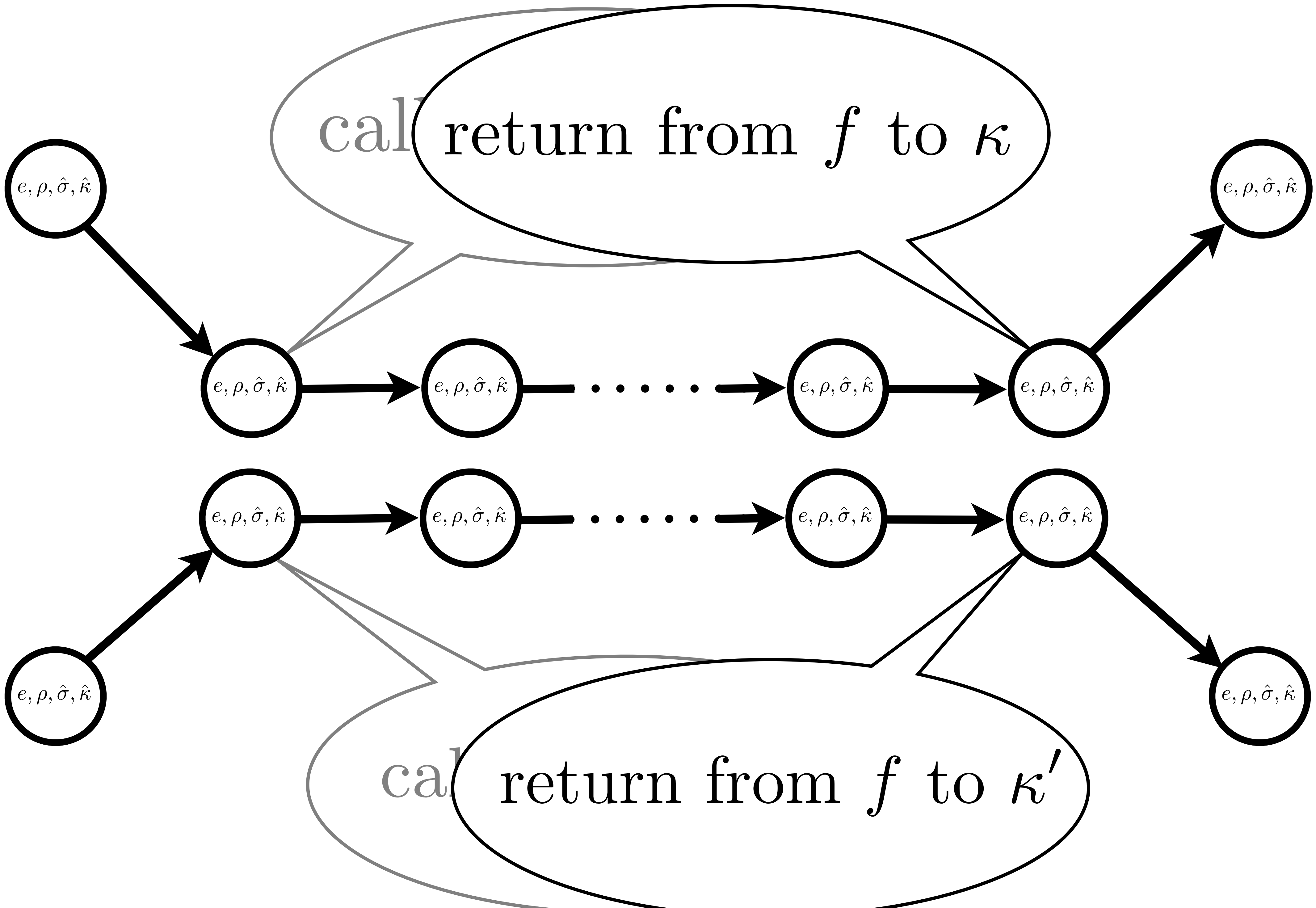

return from f to κ
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$
$e, \rho, \hat{\sigma}, \hat{\kappa}$



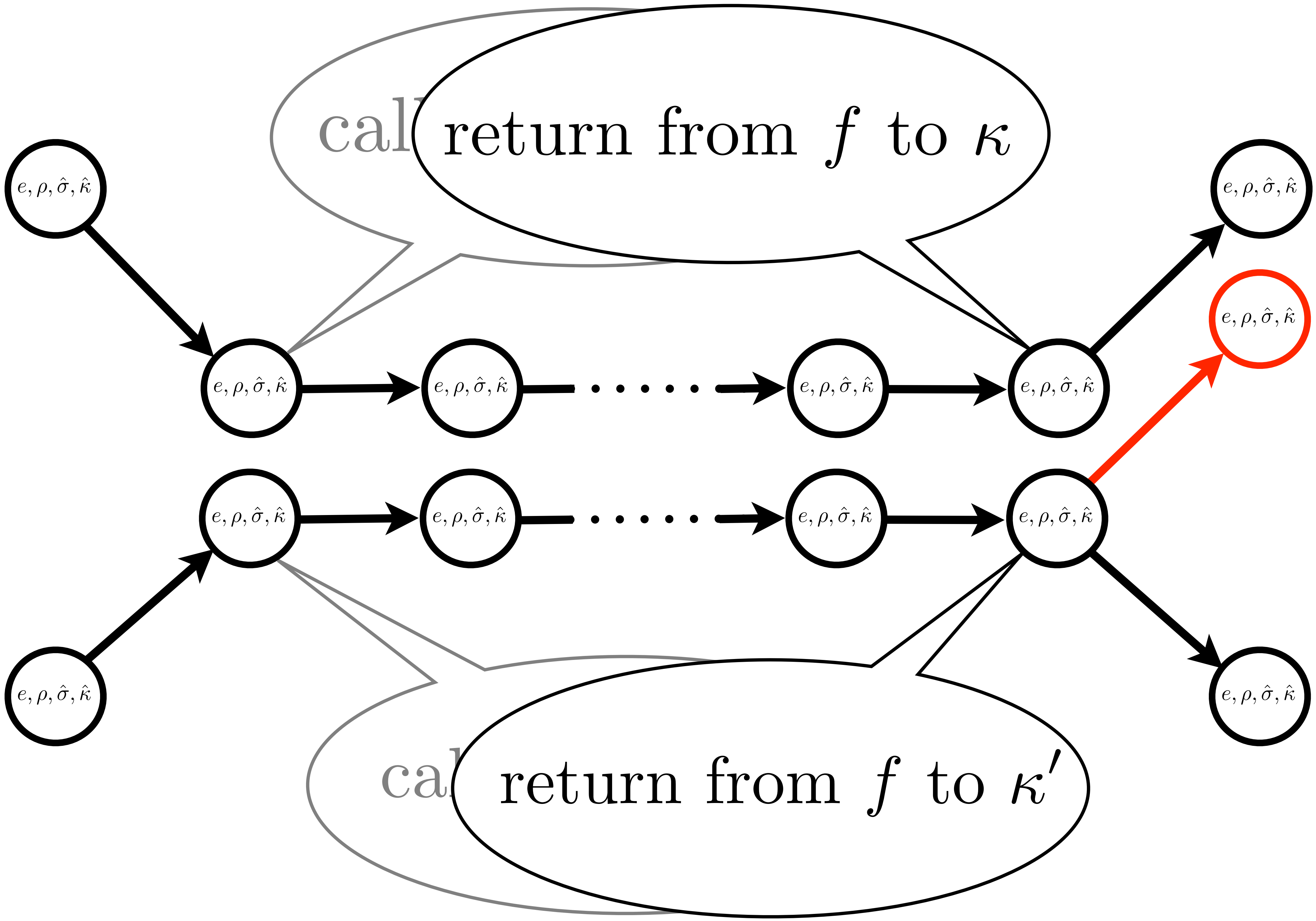
call
return from f to κ
$e, \rho, \hat{\sigma}, \hat{\kappa}$

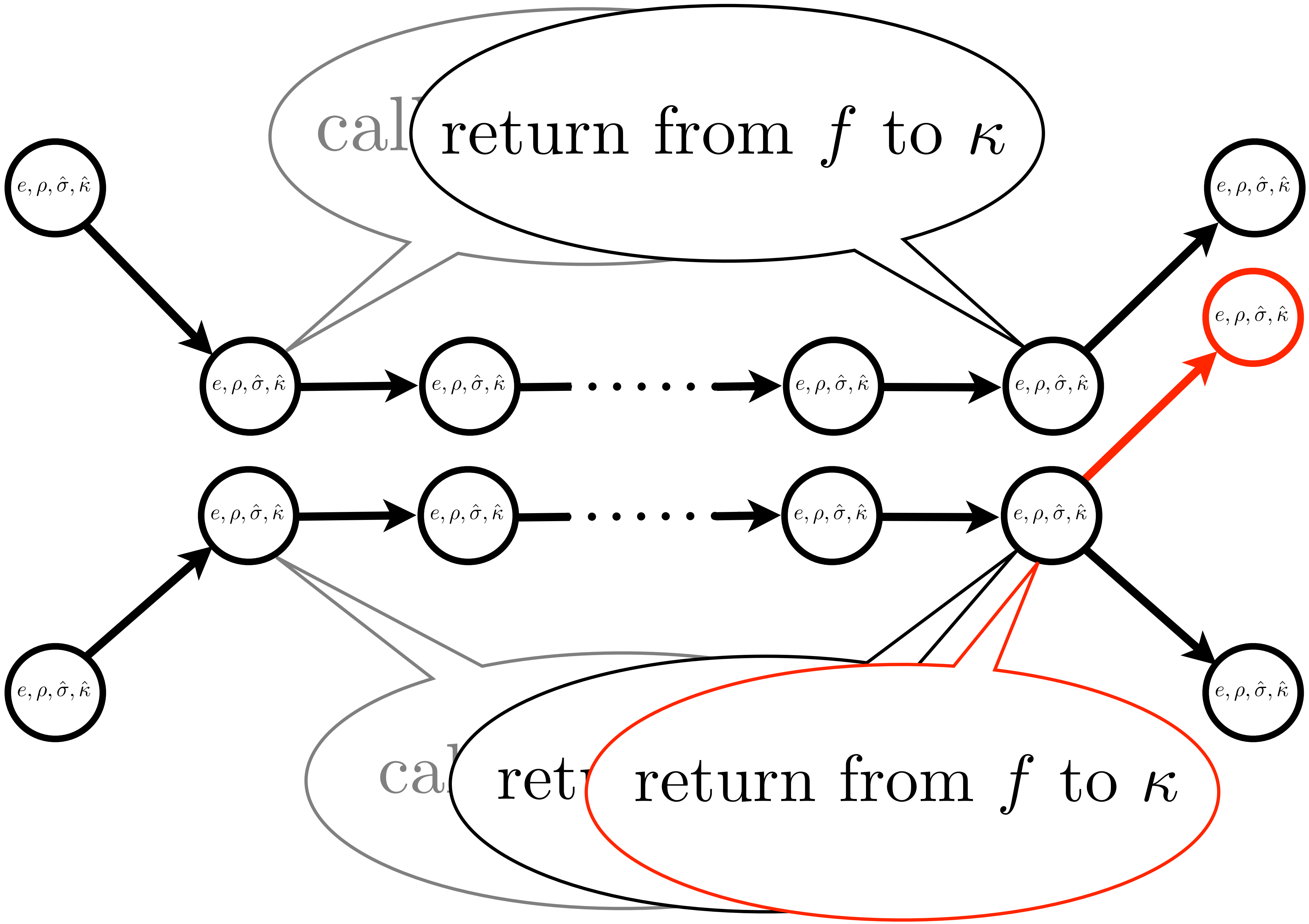


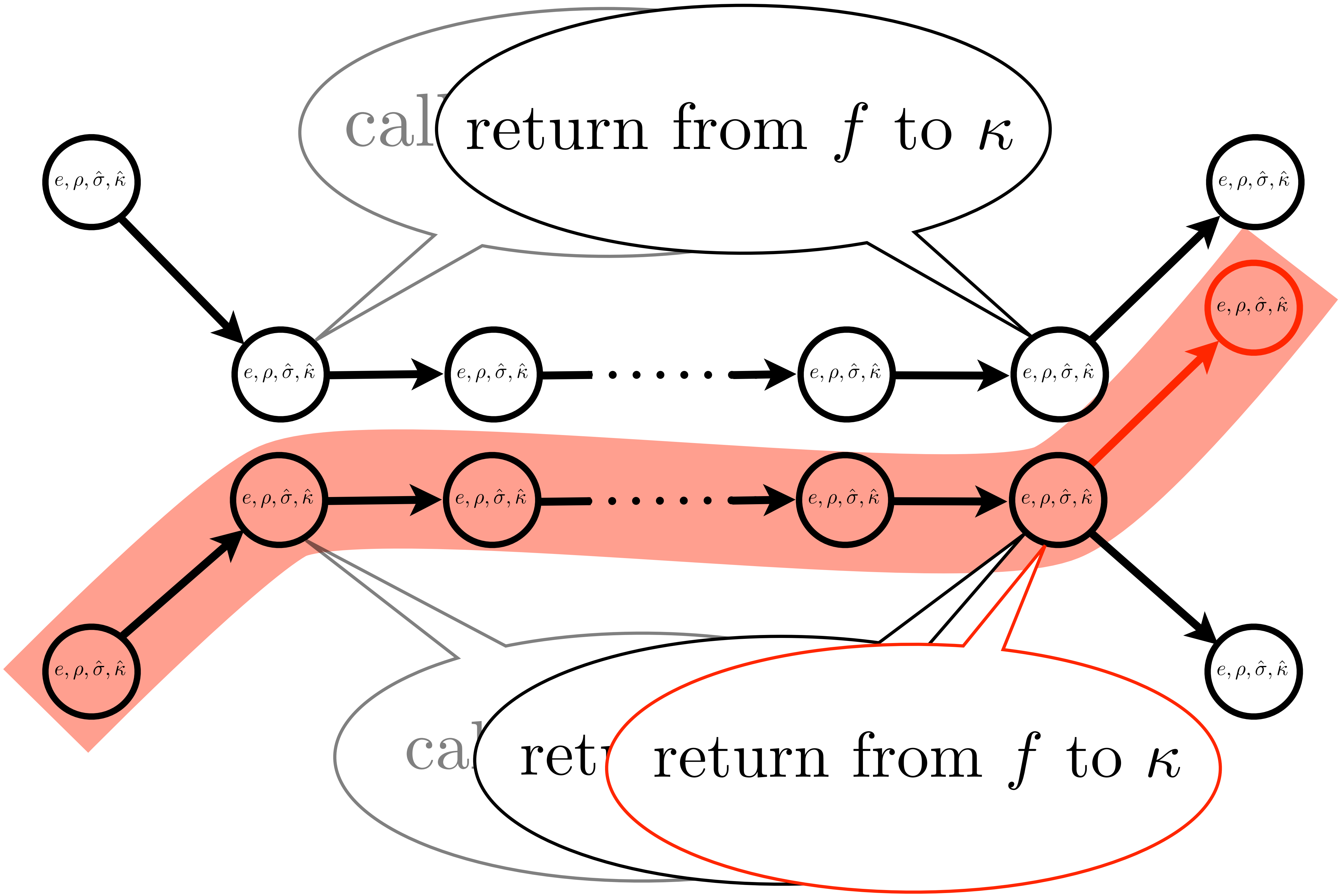
return from f to κ
call to f from κ'
$e, \rho, \hat{\sigma}, \hat{\kappa}$



return from f to κ
return from f to κ'
$e, \rho, \hat{\sigma}, \hat{\kappa}$


return from f to κ
return from f to κ'
$e, \rho, \hat{\sigma}, \hat{\kappa}$


return from f to κ
return from f to κ
$e, \rho, \hat{\sigma}, \hat{\kappa}$


return from f to κ
return from f to κ
$e, \rho, \hat{\sigma}, \hat{\kappa}$

Idea: make it finite

$$\hat{\sigma} : \widehat{\text{Addr}} \to \mathcal{P}(\text{Val})$$

$$\sigma : \text{Addr} \to \text{Val}$$

$$e, \rho, \widehat{\sigma}, \widehat{\kappa}$$

$$\epsilon \mid (e, \rho) \cdot \kappa \mid v \cdot \kappa$$

$$\epsilon \mid (e, \rho), a \mid v, a$$

Idea: make it finite

$$\hat{\sigma} : \widehat{\mathrm{Addr}} \to \mathcal{P}(\mathrm{Val})$$

$$\sigma : \mathrm{Addr} \to \mathrm{Val}$$

$$e, \rho, \widehat{\sigma}, \widehat{\kappa}$$

$$\epsilon \mid (e, \rho) \cdot \kappa \mid v \cdot \kappa$$

$$\epsilon \mid (e, \rho),\ a \mid v,\ a$$

Idea: make it [redacted]

$\hat{\sigma} : \widehat{\mathrm{Addr}} \to \mathcal{P}(\mathrm{Val})$

$\sigma : \mathrm{Addr} \to \mathrm{Val}$

$$e, \rho, \widehat{\sigma}, \widehat{\kappa}$$

$\epsilon \mid (e, \rho) \cdot \kappa \mid v \cdot \kappa$

$\epsilon \mid (e, \rho), a \mid v, a$

Idea: make it ~~finite~~ DECIDABLE

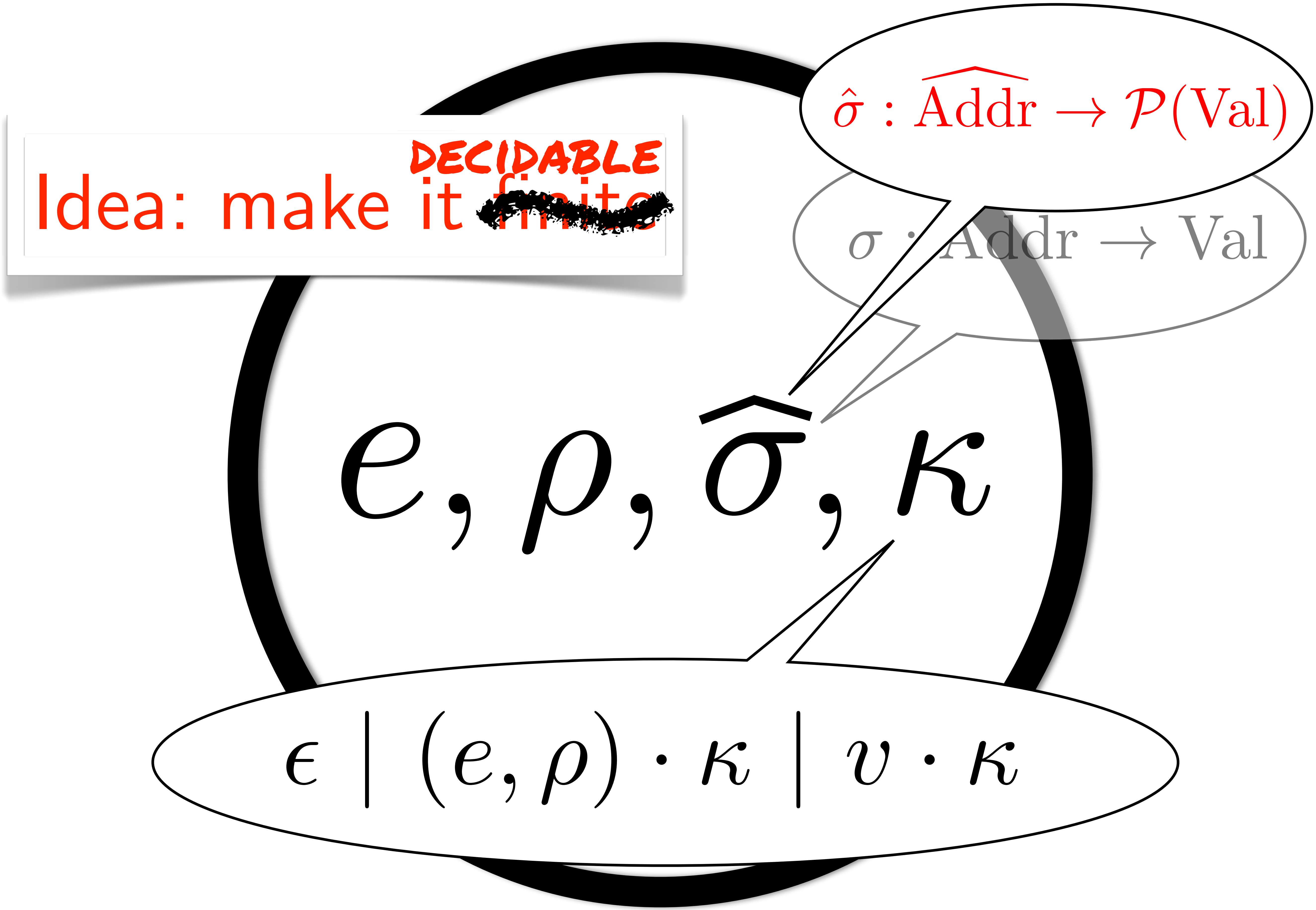
$$e, \rho, \widehat{\sigma}, \widehat{\kappa}$$

$\hat{\sigma} : \widehat{\mathrm{Addr}} \to \mathcal{P}(\mathrm{Val})$

$\sigma : \mathrm{Addr} \to \mathrm{Val}$

$\epsilon \mid (e, \rho) \cdot \kappa \mid v \cdot \kappa$

$\epsilon \mid (e, \rho),\ a \mid v,\ a$



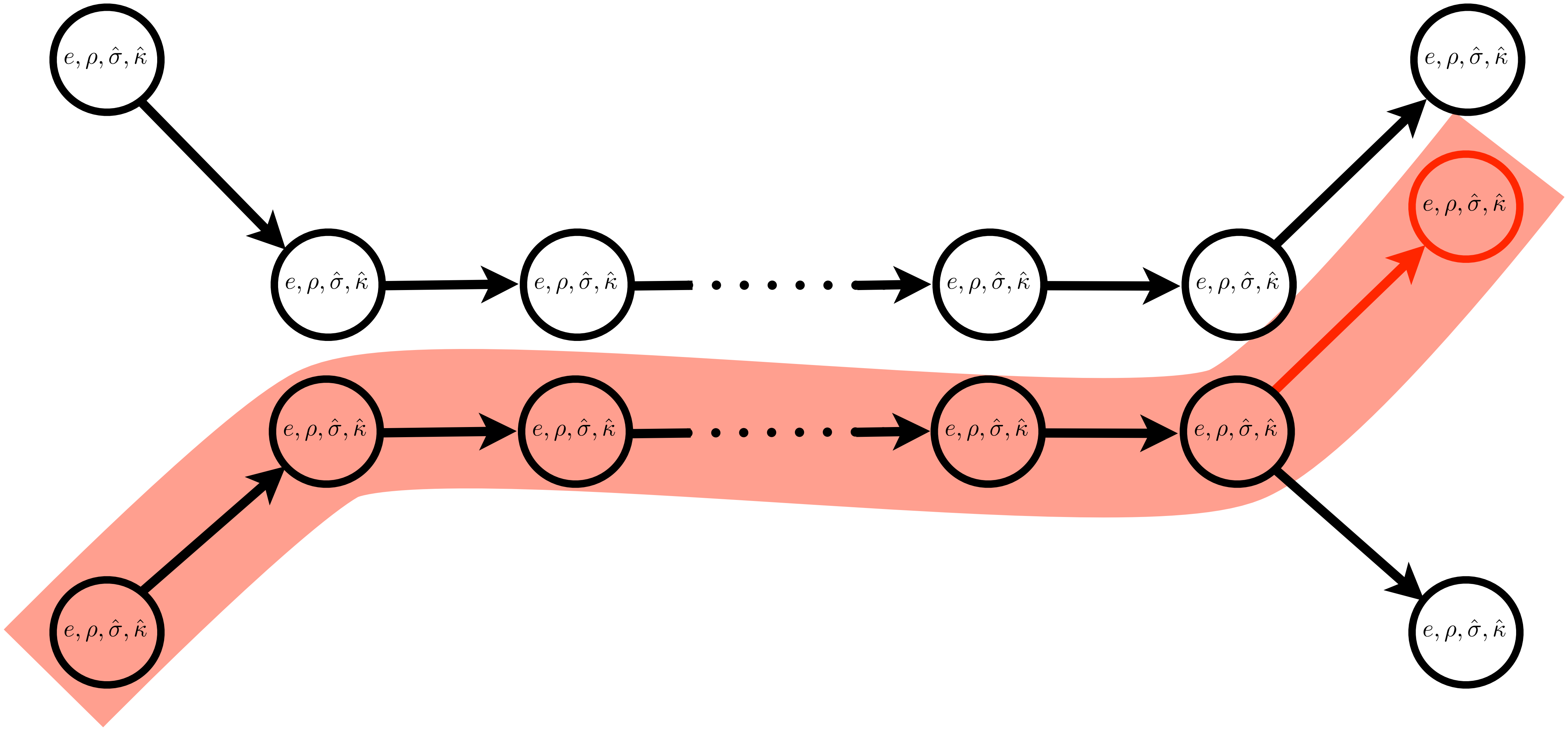
DECIDABLE
Idea: make it finite
$\hat{\sigma} : \widehat{\text{Addr}} \to \mathcal{P}(\text{Val})$
$\sigma : \text{Addr} \to \text{Val}$
$e, \rho, \widehat{\sigma}, \kappa$
$\epsilon \mid (e, \rho) \cdot \kappa \mid v \cdot \kappa$

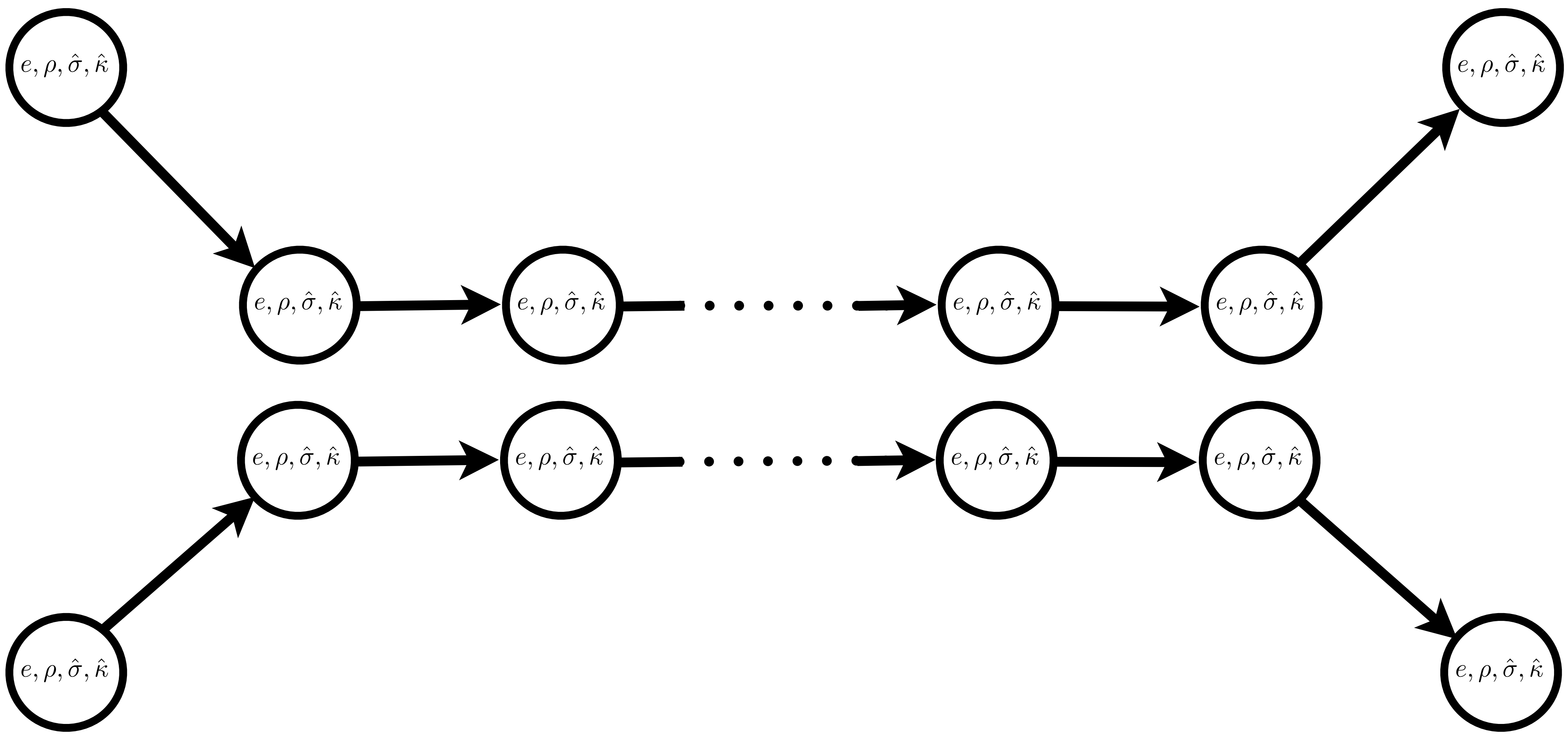
$$\begin{aligned}
\langle x, \rho, \hat{\sigma}, \hat{\kappa}\rangle &\longmapsto \langle v, \rho, \hat{\sigma}, \hat{\kappa}\rangle \quad \text{if } v \in \hat{\sigma}(\rho(x)) \\
\langle e_0\ e_1, \rho, \hat{\sigma}, \hat{\kappa}\rangle &\longmapsto \langle e_0, \rho, \hat{\sigma} \sqcup [a \mapsto \hat{\kappa}], (e_1, \rho), a\rangle \\
\langle v, \hat{\sigma}, (e, \rho), a\rangle &\longmapsto \langle e, \rho, \hat{\sigma}, v, a\rangle \\
\langle v, \hat{\sigma}, (\lambda x.e, \rho), a\rangle &\longmapsto \langle e, \rho[x \mapsto a'], \hat{\sigma} \sqcup [a' \mapsto v], \hat{\kappa}\rangle \\
&\qquad \text{if } \hat{\kappa} \in \hat{\sigma}(a)
\end{aligned}$$

$$
\begin{aligned}
\langle x, \rho, \hat{\sigma}, \hat{\kappa} \rangle &\longmapsto \langle v, \rho, \hat{\sigma}, \hat{\kappa} \rangle \quad \text{if } v \in \hat{\sigma}(\rho(x)) \\
\langle e_0\ e_1, \rho, \hat{\sigma}, \hat{\kappa} \rangle &\longmapsto \langle e_0, \rho, \hat{\sigma} \sqcup [a \mapsto \hat{\kappa}], (e_1, \rho), a \rangle \\
\langle v, \hat{\sigma}, (e, \rho), a \rangle &\longmapsto \langle e, \rho, \hat{\sigma}, v, a \rangle \\
\langle v, \hat{\sigma}, (\lambda x.e, \rho), a \rangle &\longmapsto \langle e, \rho[x \mapsto a'], \hat{\sigma} \sqcup [a' \mapsto v], \hat{\kappa} \rangle \\
&\quad\ \text{if } \hat{\kappa} \in \hat{\sigma}(a)
\end{aligned}
$$

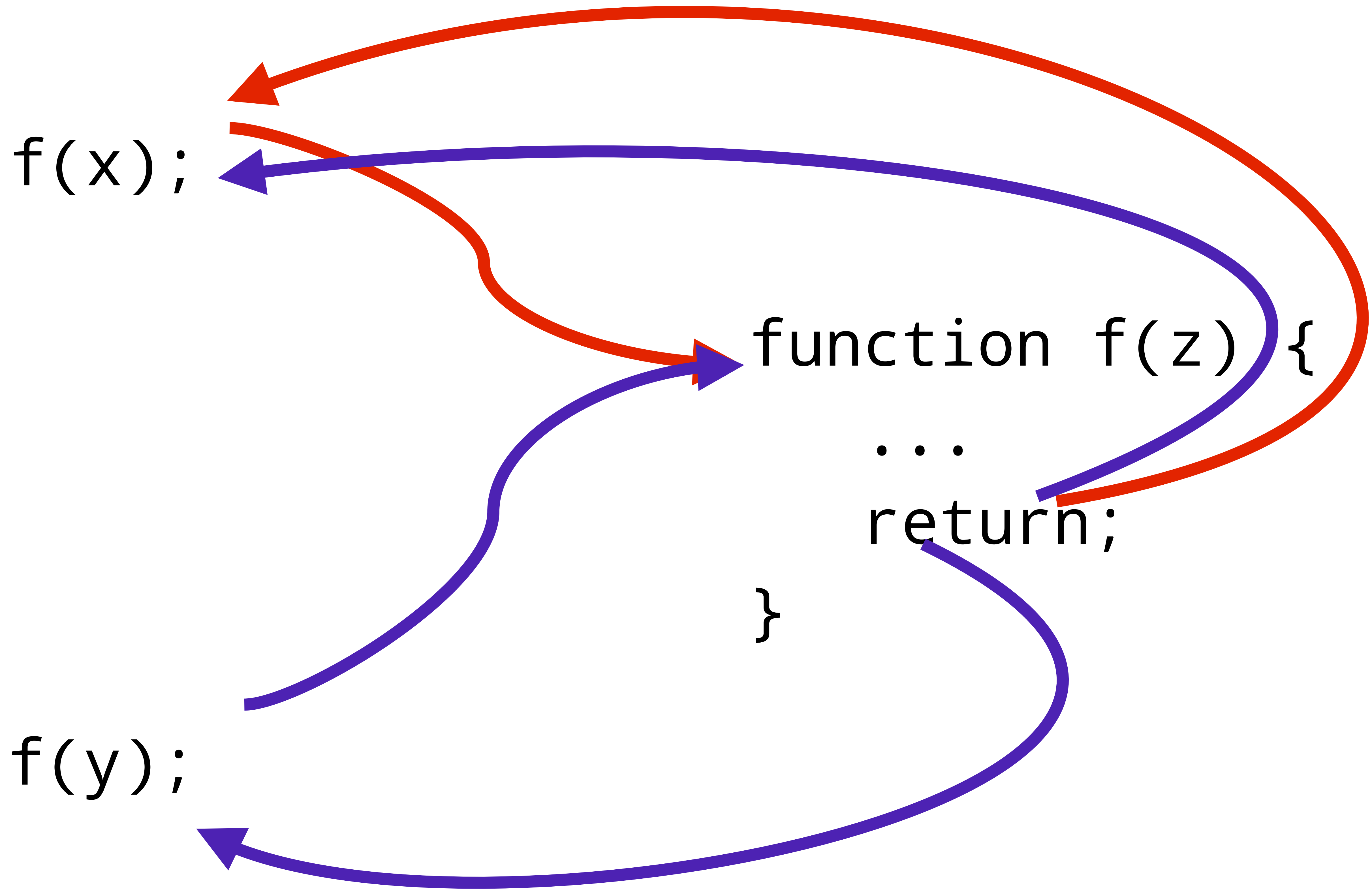
$$
\begin{aligned}
\langle x, \rho, \hat{\sigma}, \hat{\kappa} \rangle &\longmapsto \langle v, \rho, \hat{\sigma}, \hat{\kappa} \rangle \quad \text{if } v \in \hat{\sigma}(\rho(x)) \\
\langle e_0\ e_1, \rho, \hat{\sigma}, \hat{\kappa} \rangle &\longmapsto \langle e_0, \rho, \hat{\sigma} \sqcup \cancel{[a \mapsto \hat{\kappa}]}, (e_1, \rho), a \rangle \\
\langle v, \hat{\sigma}, (e, \rho), a \rangle &\longmapsto \langle e, \rho, \hat{\sigma}, v, a \rangle \\
\langle v, \hat{\sigma}, (\lambda x.e, \rho), a \rangle &\longmapsto \langle e, \rho[x \mapsto a'], \hat{\sigma} \sqcup [a' \mapsto v], \hat{\kappa} \rangle \\
& \qquad \cancel{\text{if } \hat{\kappa} \in \hat{\sigma}(a)}
\end{aligned}
$$

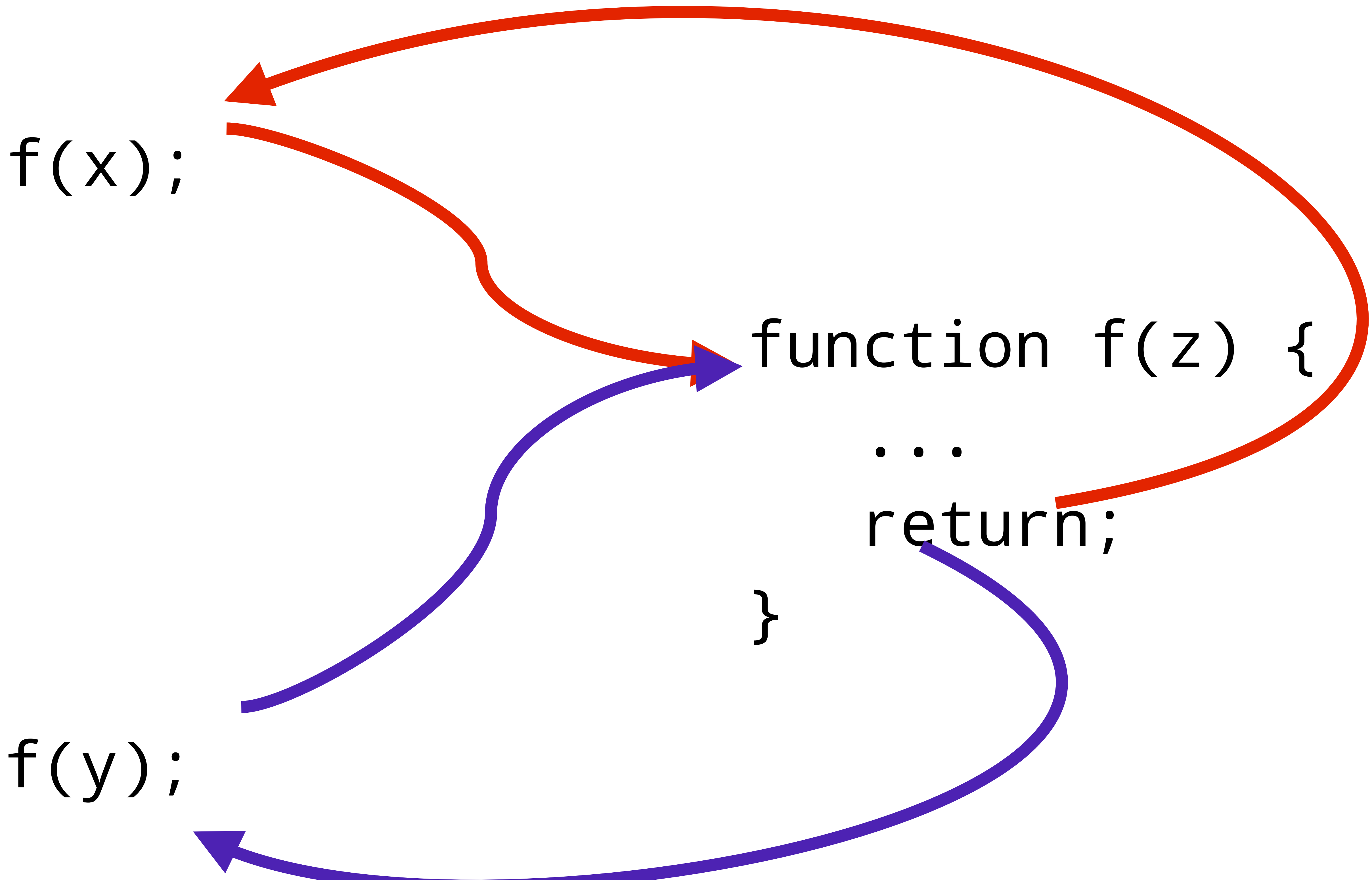
$$
\begin{aligned}
\langle x, \rho, \hat{\sigma}, \kappa \rangle &\longmapsto \langle v, \rho, \hat{\sigma}, \kappa \rangle \quad \text{if } v \in \hat{\sigma}(\rho(x)) \\
\langle e_0\ e_1, \rho, \hat{\sigma}, \kappa \rangle &\longmapsto \langle e_0, \rho, \hat{\sigma}, (e_1, \rho) \cdot \kappa \rangle \\
\langle v, \hat{\sigma}, (e, \rho) \cdot \kappa \rangle &\longmapsto \langle e, \rho, \hat{\sigma}, v \cdot \kappa \rangle \\
\langle v, \hat{\sigma}, (\lambda x.e, \rho) \cdot \kappa \rangle &\longmapsto \langle e, \rho[x \mapsto a], \hat{\sigma} \sqcup [a \mapsto v], \kappa \rangle
\end{aligned}
$$

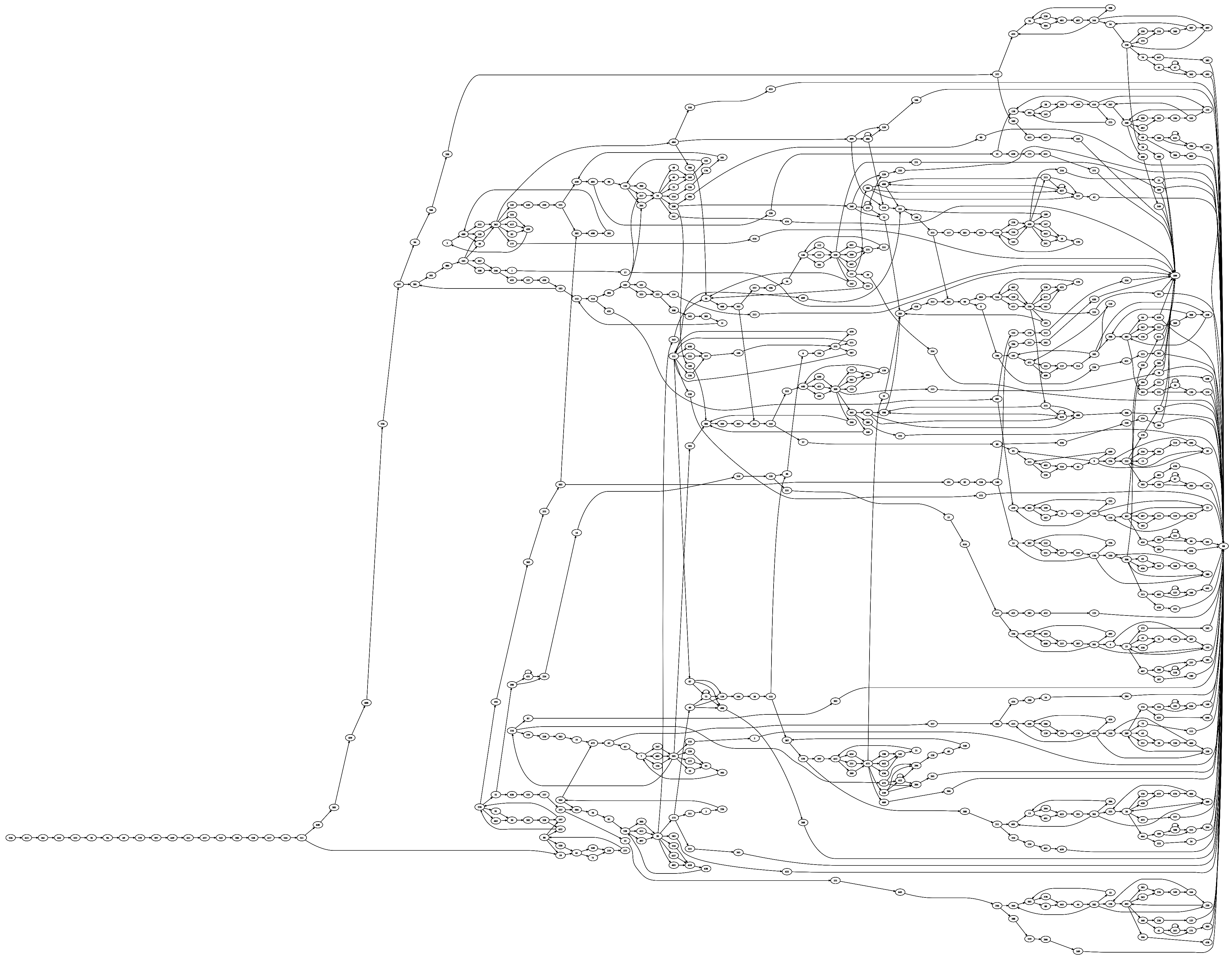

$e, \rho, \hat{\sigma}, \hat{\kappa}$

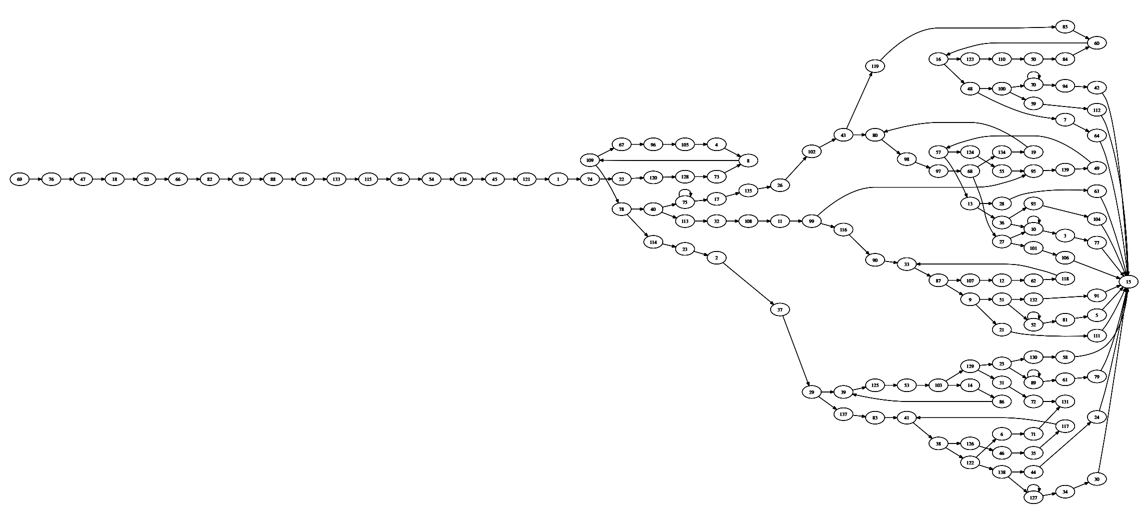


$e, \rho, \hat{\sigma}, \hat{\kappa}$

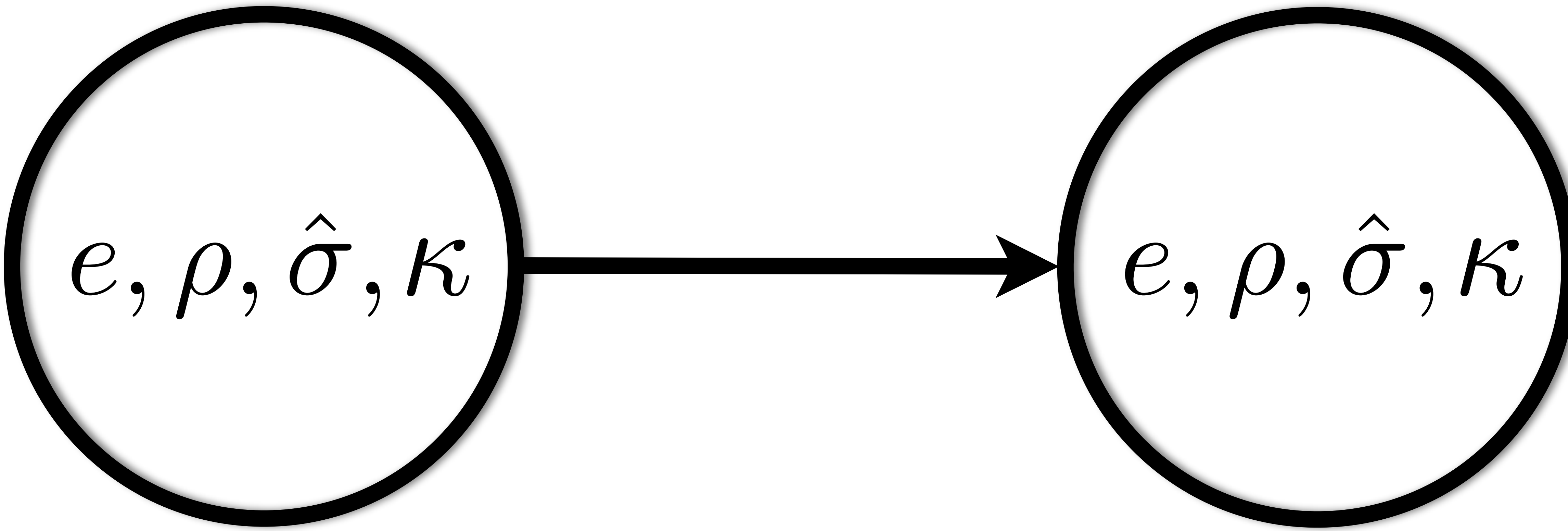

f(x);
function f(z) {
...
return;
}
f(y);

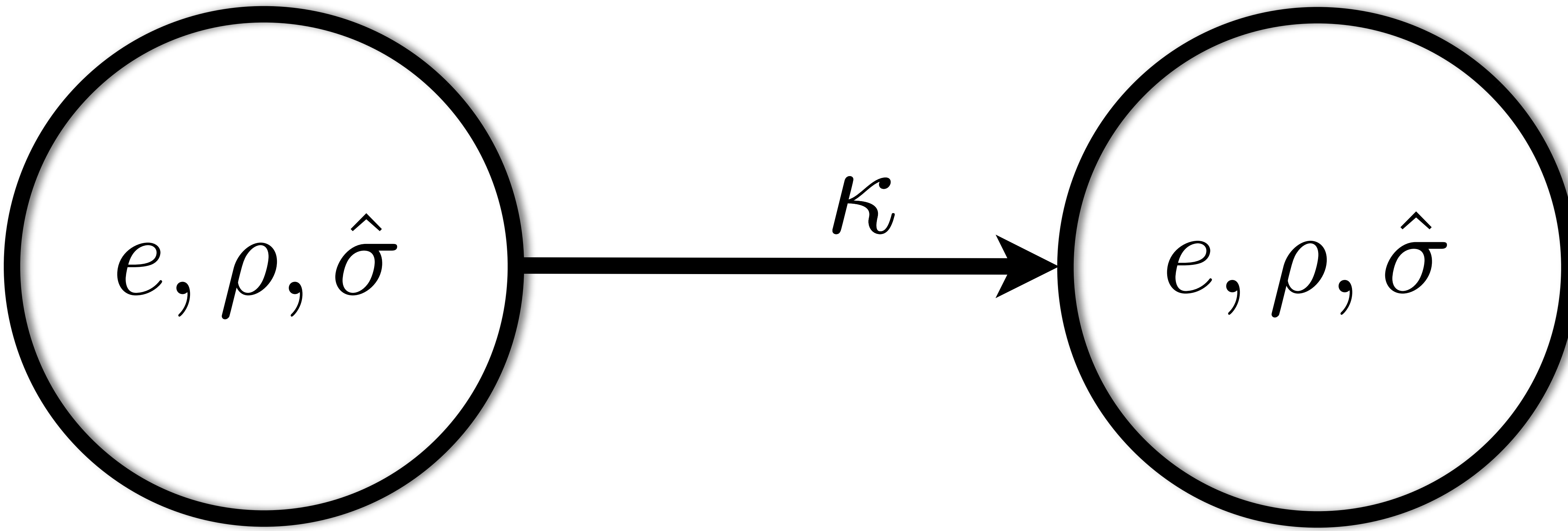

f(x);
function f(z) {
...
return;
}
f(y);



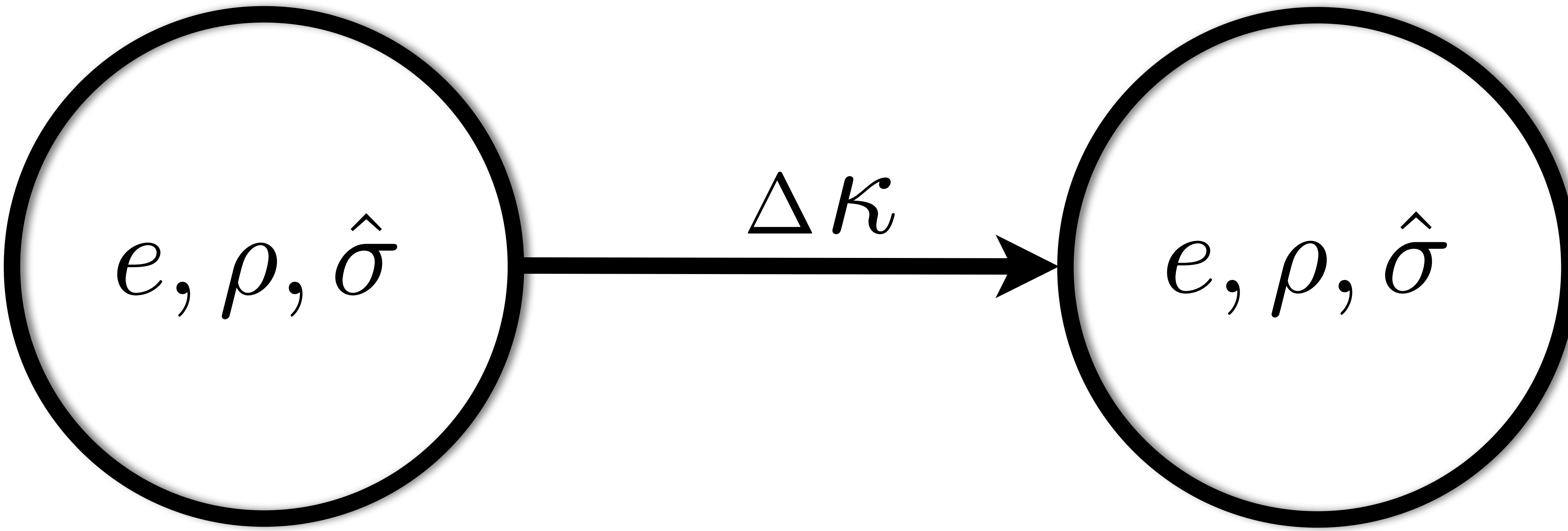


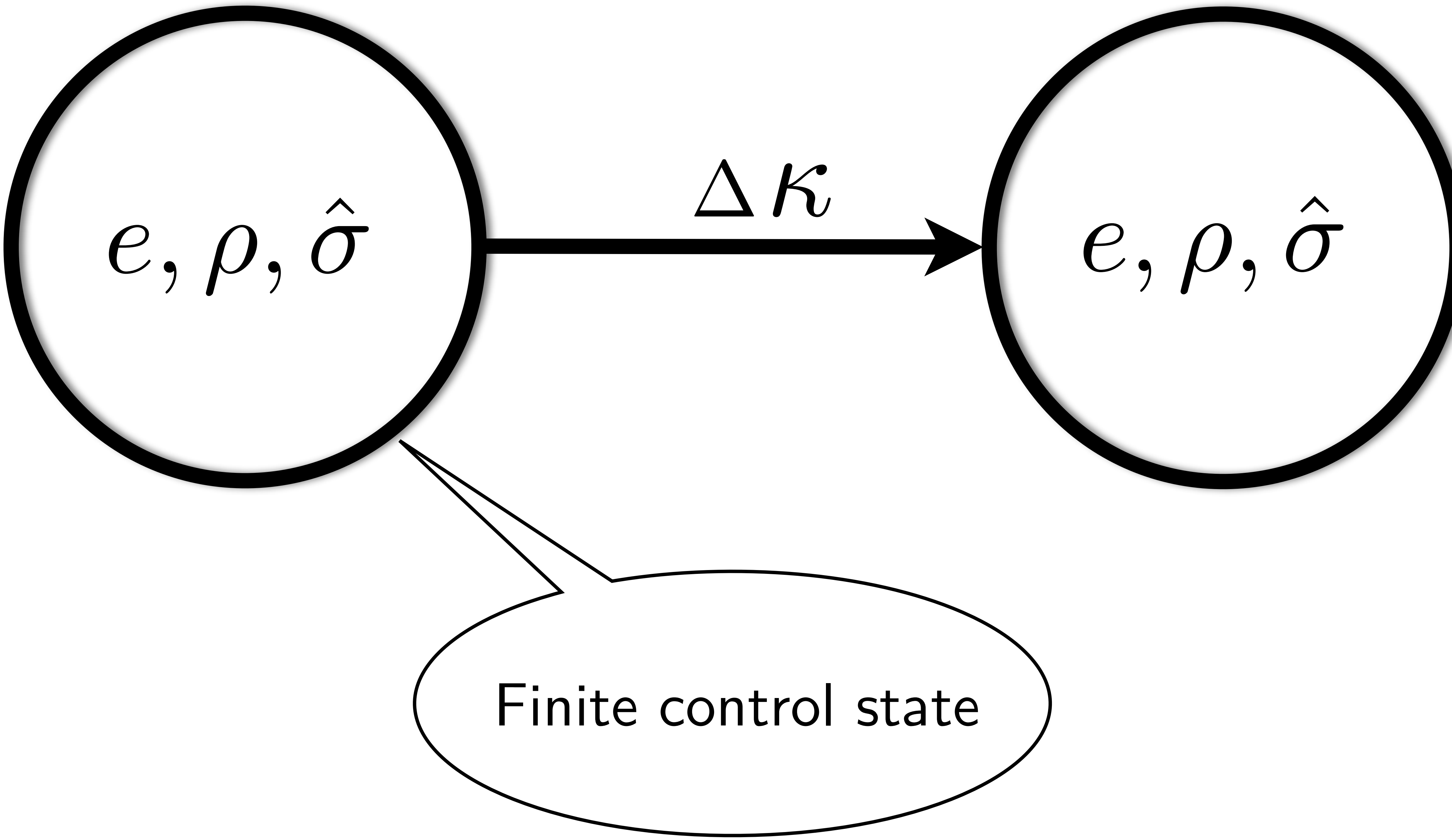

$e, \rho, \widehat{\sigma}, \kappa$

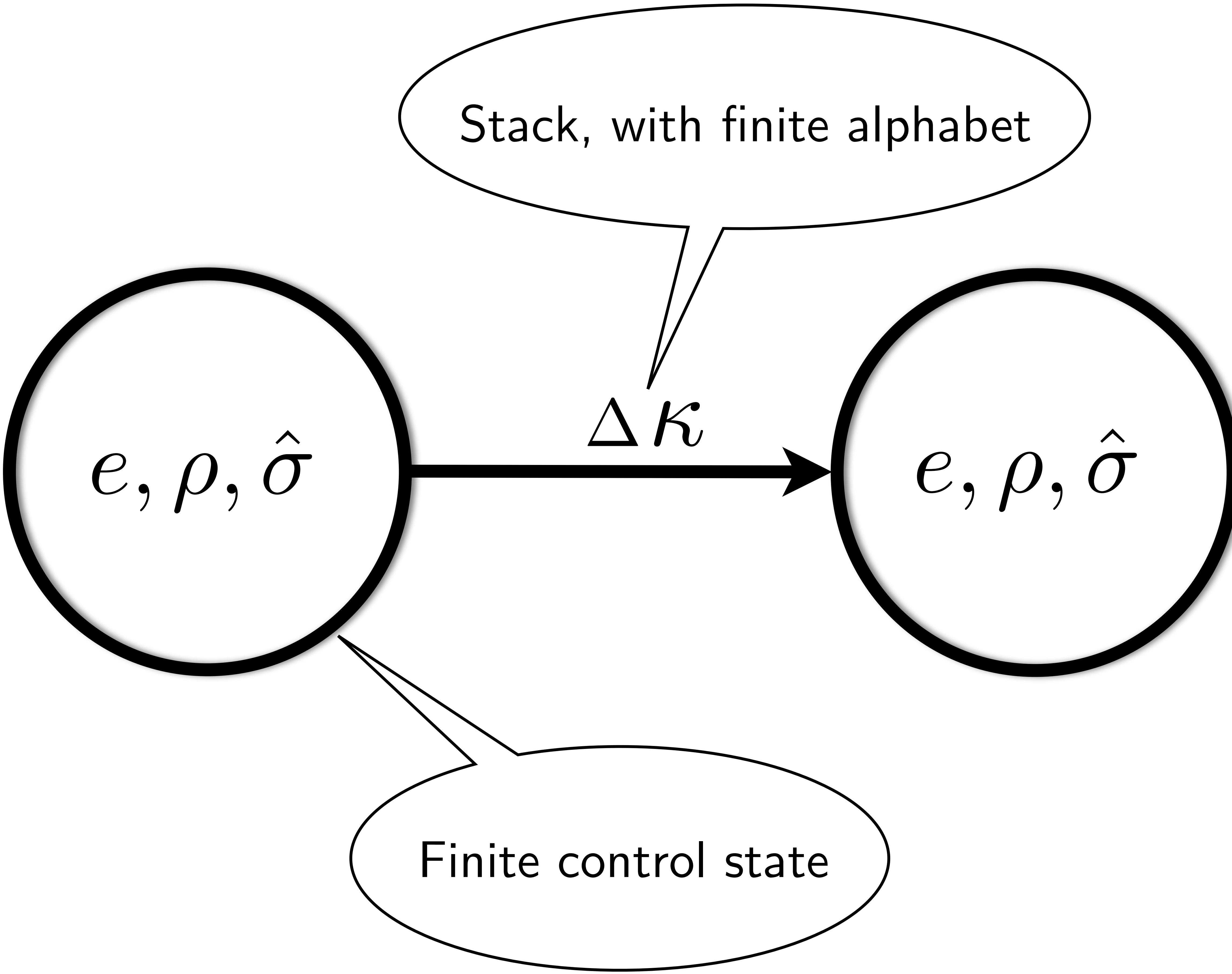

$e, \rho, \hat{\sigma}, \kappa$
$e, \rho, \hat{\sigma}, \kappa$

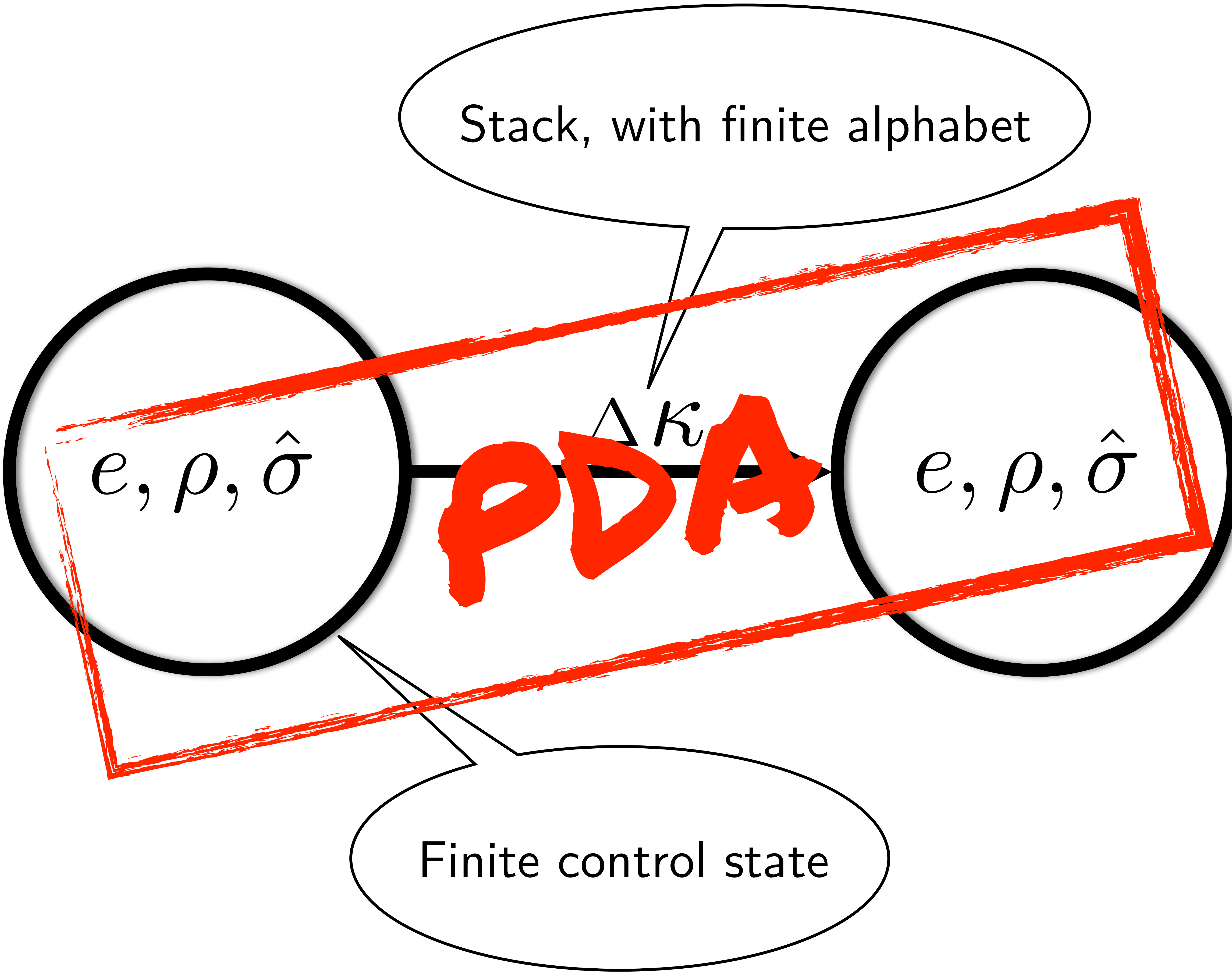


$e, \rho, \hat{\sigma}$
κ
$e, \rho, \hat{\sigma}$


$e, \rho, \hat{\sigma}$
$\Delta\kappa$
$e, \rho, \hat{\sigma}$


$e, \rho, \hat{\sigma}$
$\Delta\kappa$
$e, \rho, \hat{\sigma}$
Finite control state


Stack, with finite alphabet
$\Delta\kappa$
$e, \rho, \hat{\sigma}$
$e, \rho, \hat{\sigma}$
Finite control state



Stack, with finite alphabet
$\Delta\kappa$
$e, \rho, \hat{\sigma}$
PDA
$e, \rho, \hat{\sigma}$
Finite control state

Part III: Results

AndorsTrail	SplitTimer	SMSBackup
AndroidGame	SuperNote	SMSBlocker
AndroidPrivacyGuard_E	SuperSoduko	SMSPopup
Butane	SysMon	SysWatcherA
CalcA	SysWatcherB	SourceViewer
CalcB	TextSecure	UltraCoolMap
ConnectBot	TodoList	YARR
CountdownTimer	Word Helper	AndroidsFortune
FunDraw	AndBible	CalcC
MorseCode	AndroidPrivacyGuard_M	CalcE
MyDrawA	BatteryIndicator	ColorMatcher
MyDrawC	CalcF	FullControl
NewsCollator	MediaFun	KitteyKittey
PasswordSaver	MyDrawD	Orienteering2
PersistantAssistant	OpenGPSTracker	Sanity
SmartWebCam	Orienteering1	TomDroid
SMSReminder	PicViewer	WiFinder
SourceViewer	Collaboration ShareLoc	

DARPA™

Improving Exception-flow analysis

```
a.foo()
```

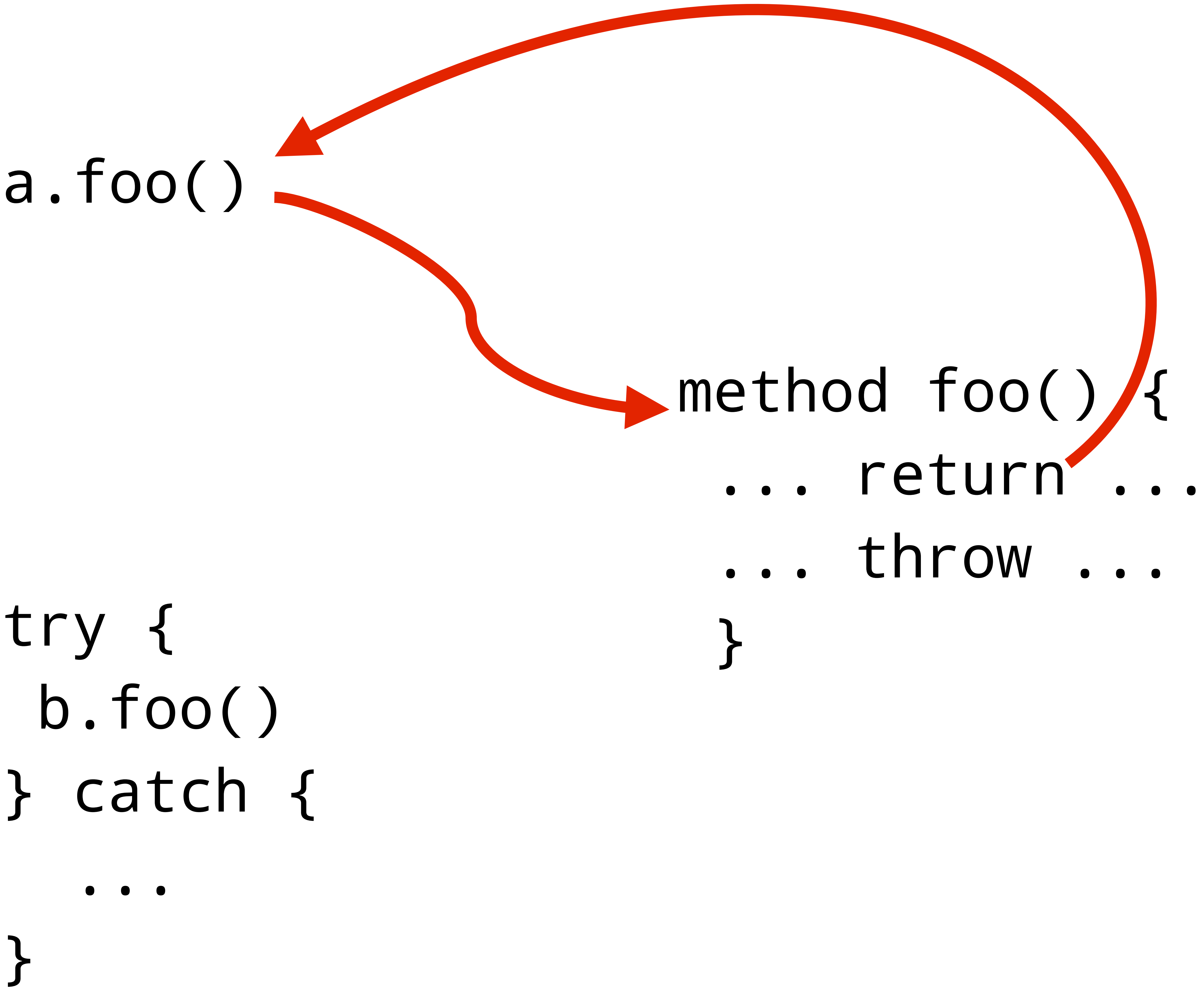
```
method foo() {
 ... return ...
 ... throw ...
}
```

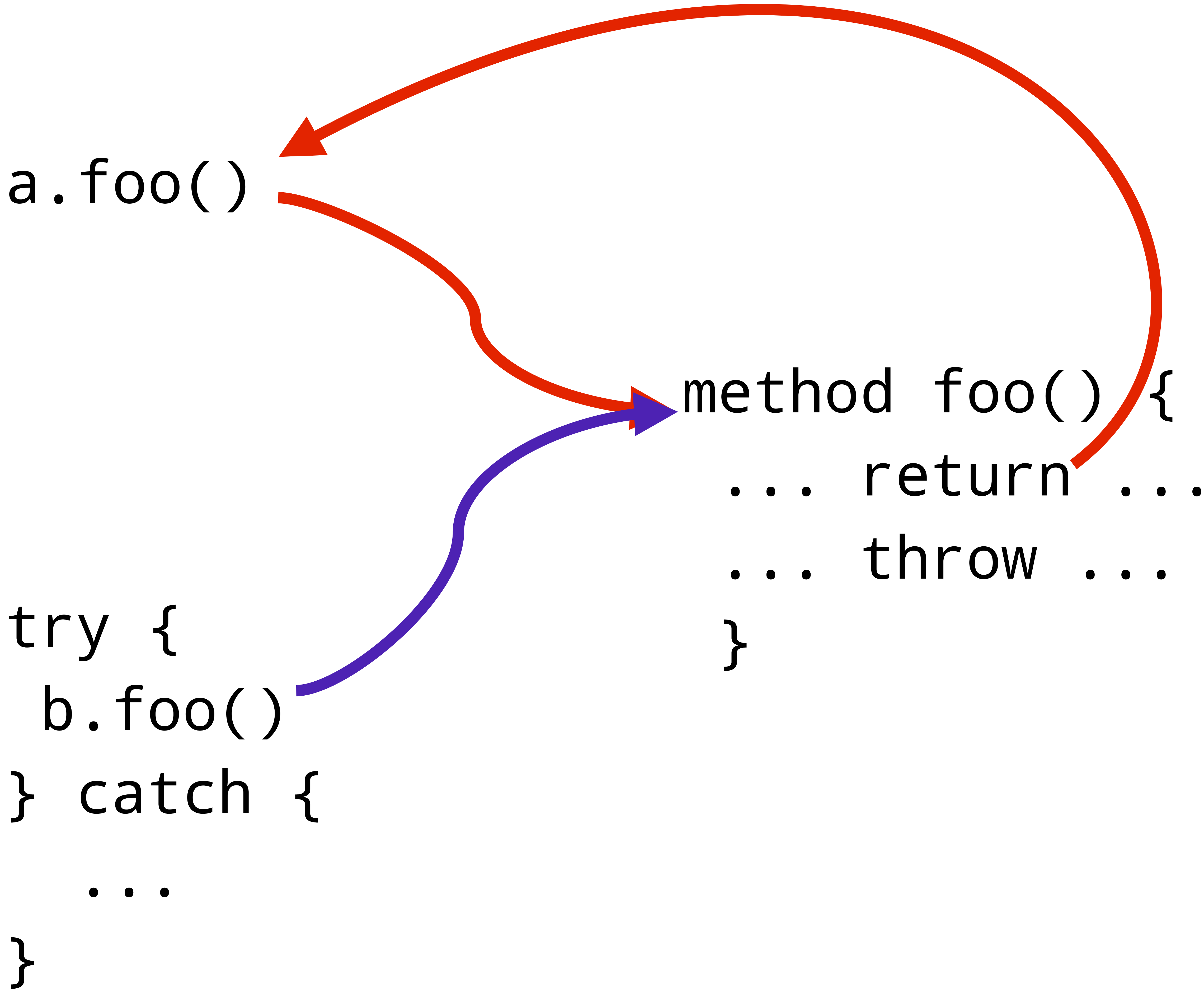
```
try {
 b.foo()
} catch {
  ...
}
```

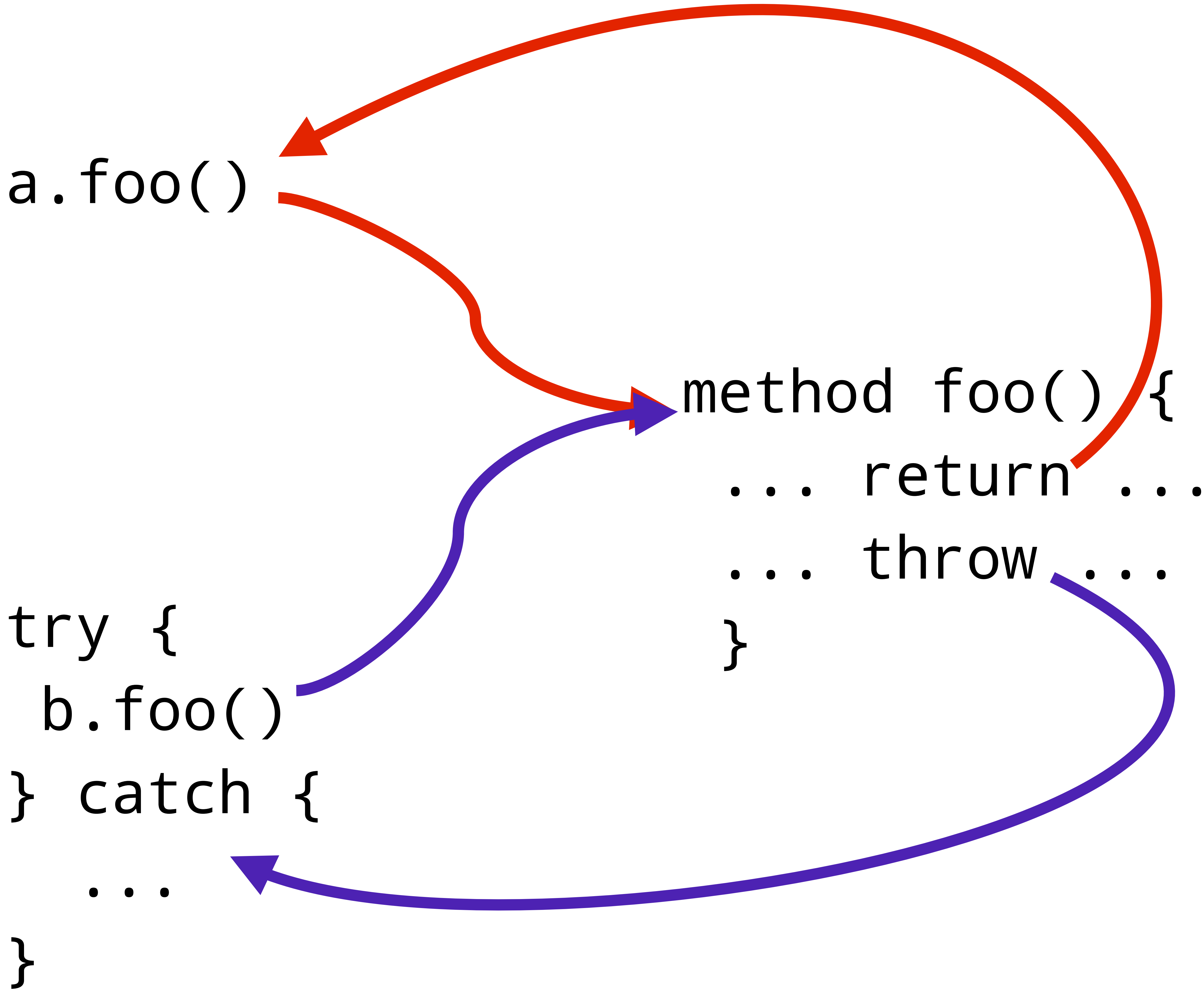
```
a.foo()
```

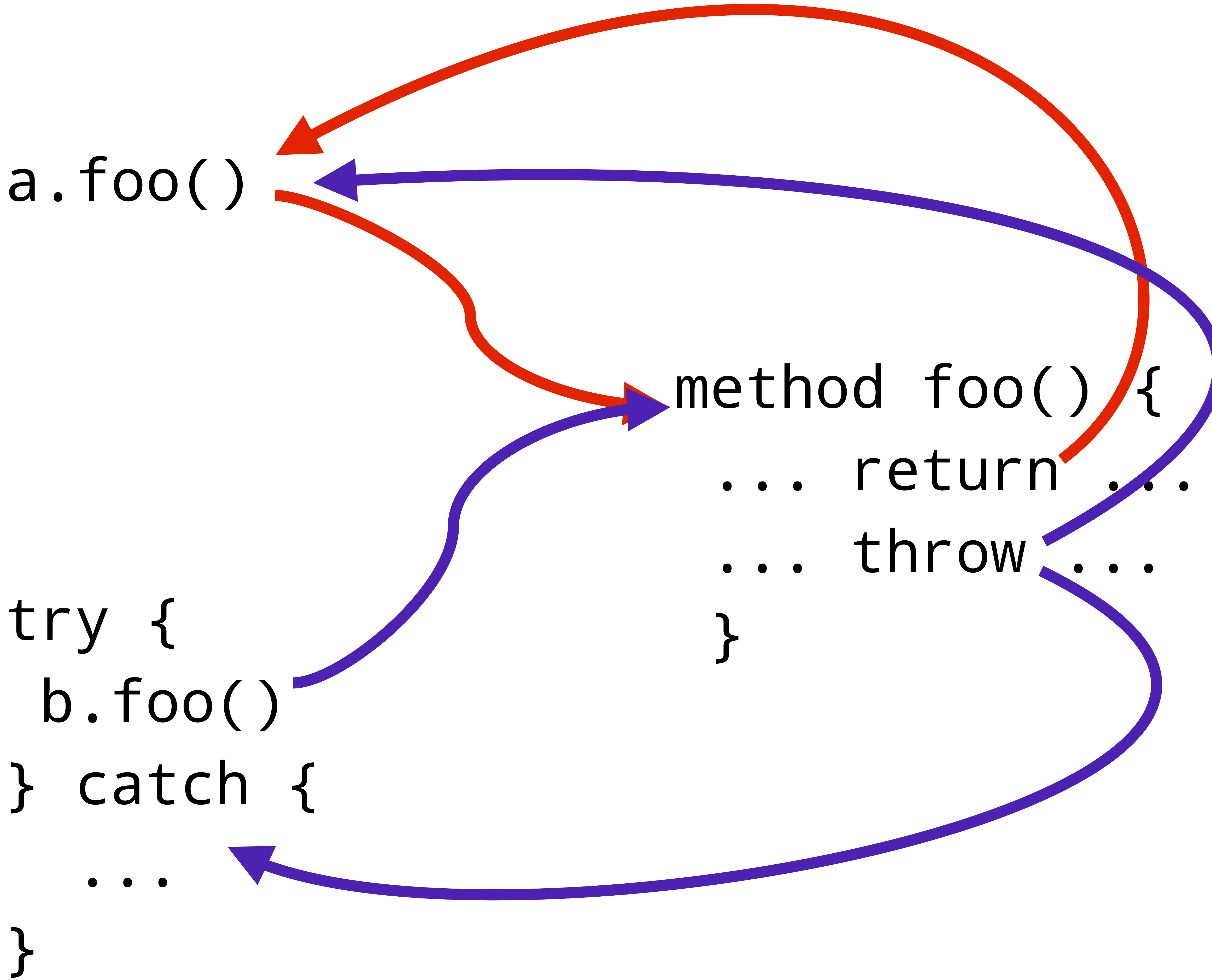
```
method foo() {
  ... return ...
  ... throw ...
}
```

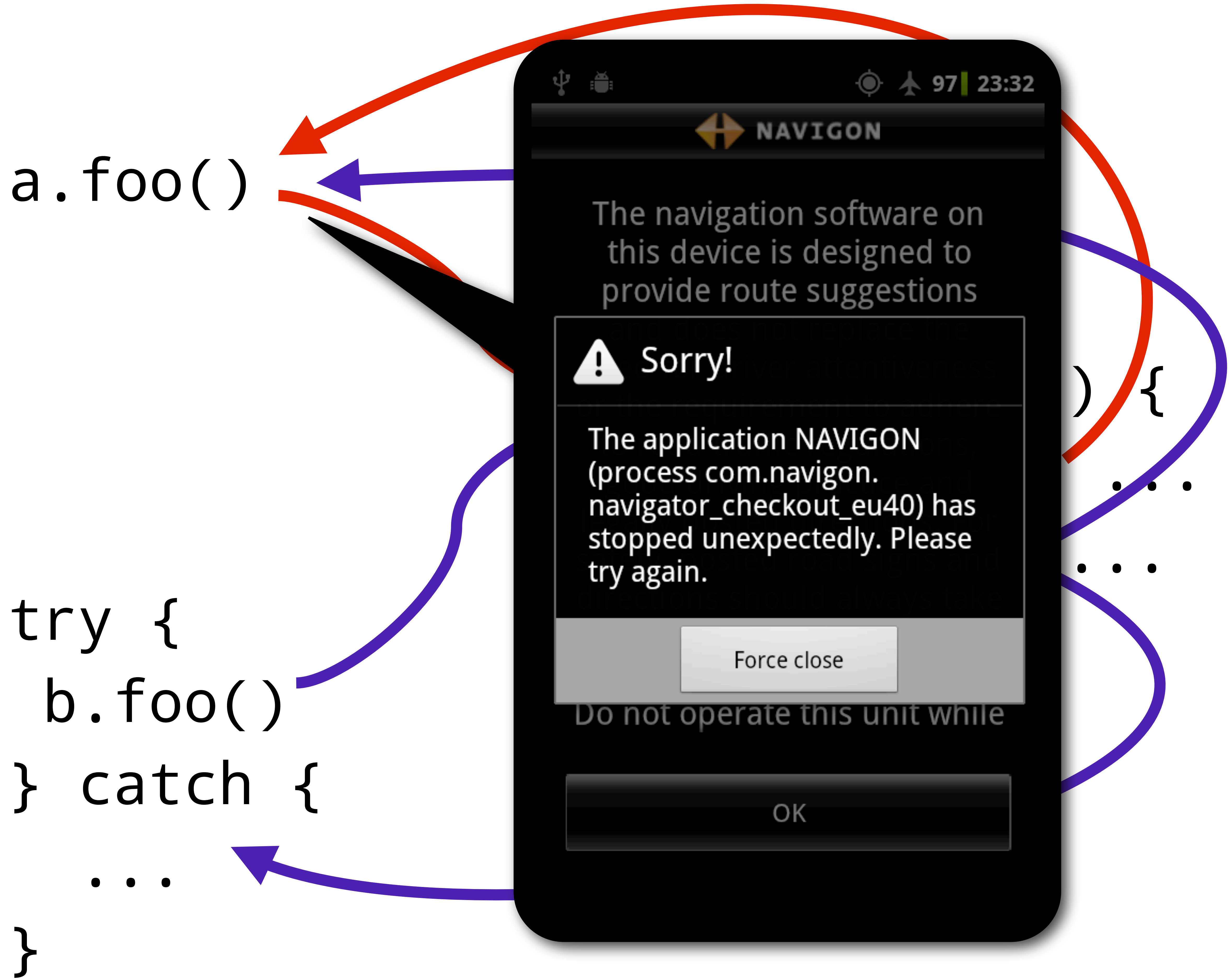
```
try {
 b.foo()
} catch {
  ...
}
```

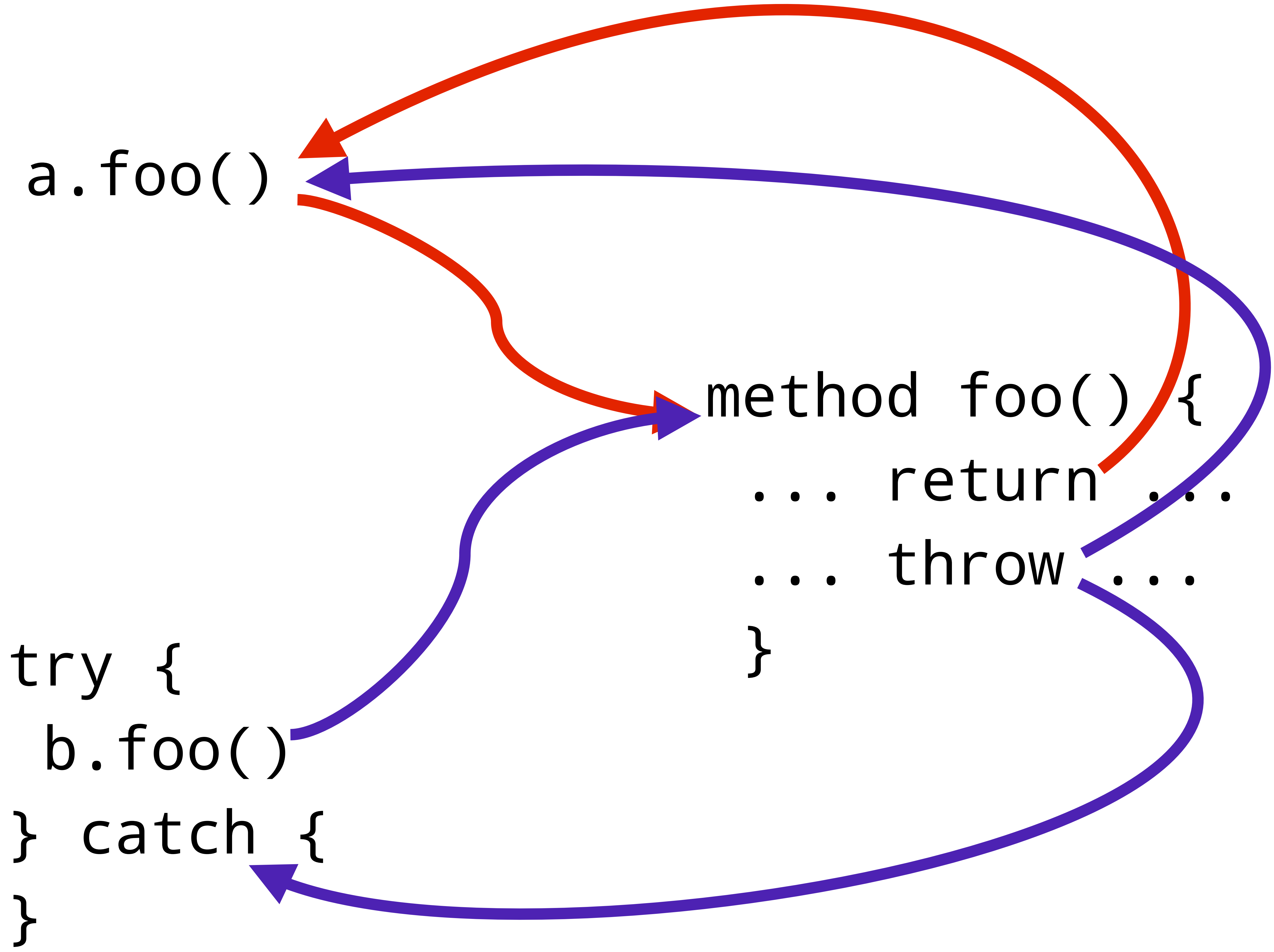

a.foo()
method foo() {
... return ...
... throw ...
}
try {
b.foo()
} catch {
...
}


a.foo()
method foo() {
... return ...
... throw ...
}
try {
b.foo()
} catch {
...
}

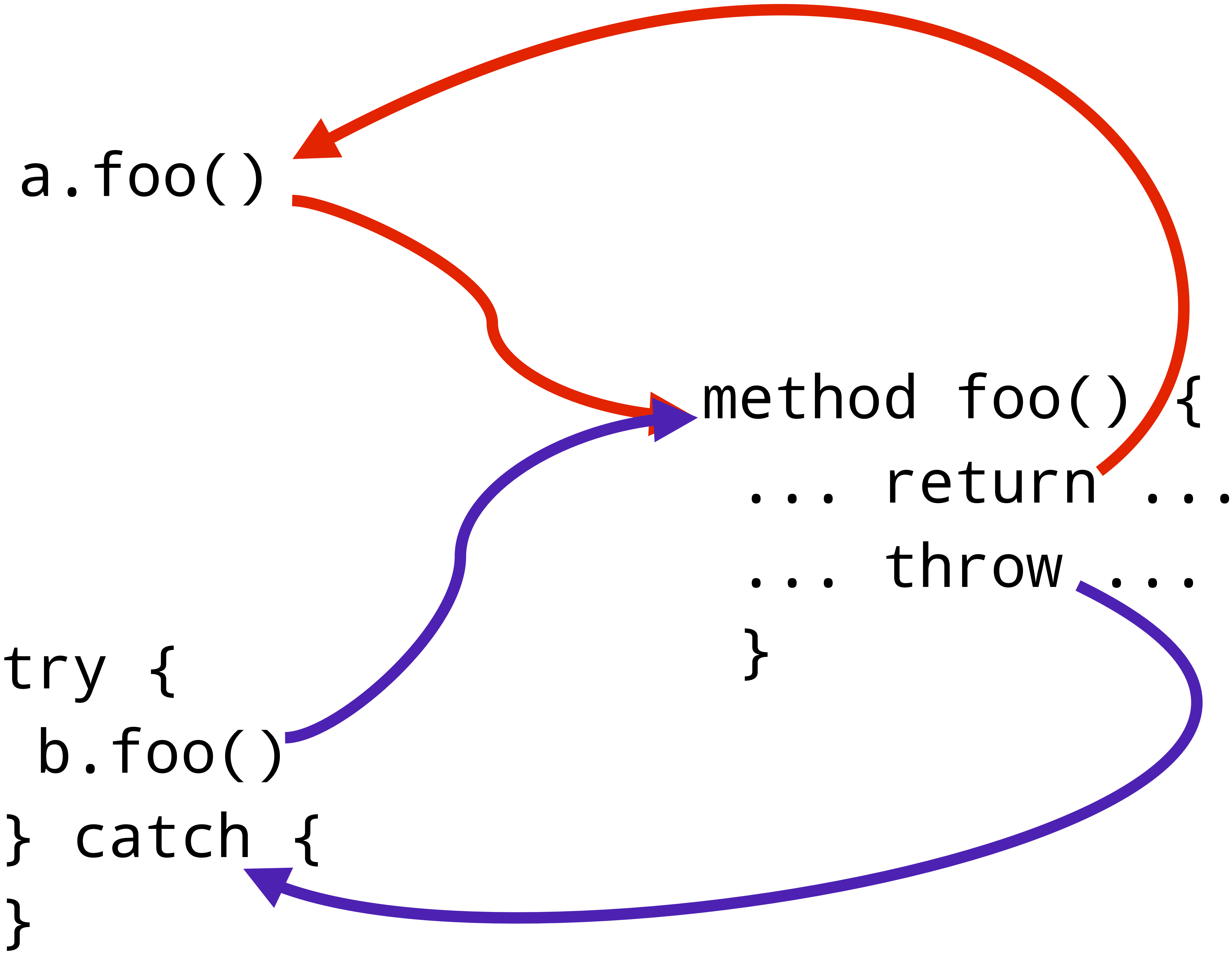

a.foo()
method foo() {
... return ...
... throw ...
}
try {
b.foo()
} catch {
...
}


a.foo()
method foo() {
... return ...
... throw ...
}
try {
b.foo()
} catch {
...
}


a.foo()
try {
b.foo()
} catch {
...
}
) {
...
...
97 23:32
NAVIGON
The navigation software on this device is designed to provide route suggestions
Sorry!
The application NAVIGON (process com.navigon. navigator_checkout_eu40) has stopped unexpectedly. Please try again.
Force close
Do not operate this unit while
OK



a.foo()
method foo() {
... return ...
... throw ...
}
try {
b.foo()
} catch {
}


a.foo()
method foo() {
... return ...
... throw ...
}
try {
b.foo()
} catch {
}

```
a.foo()
```

```
method foo() {
 ... return ...
 ... throw ...
}
```

```
try {
 b.foo()
} catch {
}
```

Program	Variable points-to	Throw-Catch edges	Time
antlr 35KLOC	614	2277	>4 hours
lusearch 87KLOC	348	2378	46 minutes
pmd 55KLOC	343	2284	56 minutes

Bravenboer & Smaragdakis, ISSTA'09

Program	Variable points-to	Throw-Catch edges	Time
antlr 35KLOC	614 2	2277 65	>4 hours 1.1 hours
lusearch 87KLOC	348 2	2378 59	46 minutes 46 minutes
pmd 55KLOC	343 2	2284 38	56 minutes 22 minutes

Pushdown Exception Flow Analysis

Run-time Techniques at Analysis-time

Abstract Models of Memory Management*

Greg Morrisett — Carnegie Mellon — jgmorris@cs.cmu.edu

Matthias Felleisen — Rice University — matthias@cs.rice.edu

Robert Harper — Carnegie Mellon — rwh@cs.cmu.edu

Abstract

Most specifications of garbage collectors concentrate on the low-level algorithmic details of *how* to find and preserve accessible objects. Often, they focus on bit-level manipulations such as "scanning stack frames," "marking objects," "tagging data," *etc.* While these details are important in some contexts, they often obscure the more fundamental aspects of memory management: *what* objects are garbage and *why*?

We develop a series of calculi that are just low-level enough that we can express allocation and garbage collection, yet are sufficiently abstract that we may formally prove the correctness of various memory management strategies. By making the heap of a program syntactically apparent, we can specify memory actions as rewriting rules that allocate values on the heap and automatically dereference pointers to such objects when needed. This formulation permits the specification of garbage collection as a relation that removes portions of the heap without affecting the outcome of the evaluation.

Our high-level approach allows us to specify in a compact manner a wide variety of memory management techniques, including standard trace-based garbage collection (*i.e.*, the family of copying and mark/sweep collection algorithms), generational collection, and type-based, tag-free collection. Furthermore, since the definition of garbage is based on the *semantics* of the underlying language instead of the conservative approximation of inaccessibility, we are able to specify and prove the idea that type inference can be used to collect some objects that are accessible but never used.

*This work was sponsored in part by the Advanced Research Projects Agency (ARPA), CSTO, under the title "The Fox Project: Advanced Development of Systems Software," ARPA Order No. 8313, issued by ESD/AVS under Contract No. F19628–91–C–0168, Wright Laboratory, Aeronautical Systems Center, Air Force Materiel Command, USAF, and ARPA grant No. F33615-93-1-1330. Views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing official policies or endorsements, either expressed or implied, of Wright Laboratory or the United States Government.

1 Memory Safety

Advanced programming languages manage memory allocation and deallocation automatically. Automatic memory managers, or garbage collectors, significantly facilitate the programming process because programmers can rely on the language implementation for the delicate tasks of finding and freeing unneeded objects. Indeed, the presence of a garbage collector ensures *memory safety* in the same way that a type system guarantees *type safety*: no program written in an advanced programming language will crash due to dangling pointer problems while allocation, access, and deallocation are transparent. However, in contrast to type systems, memory management strategies and particularly garbage collectors rarely come with a compact formulation and a formal proof of soundness. Since garbage collectors work on the machine representations of abstract values, the very idea of providing a proof of memory safety sounds unrealistic given the lack of simple models of memory operations.

The recently developed syntactic approaches to the specification of language semantics by Felleisen and Hieb [11] and Mason and Talcott [18, 19] are the first execution models that are intensional enough to permit the specification of memory management actions and yet are sufficiently abstract to permit compact proofs of important properties. Starting from the λ_v-**S** calculus of Felleisen and Hieb, we design compact specifications of a number of memory management ideas and prove several correctness theorems.

The basic idea underlying the development of our garbage collection calculi is the representation of a program's run-time memory as a global series of syntactic declarations. The program evaluation rules allocate large objects in the global declaration, which represents the heap, and automatically dereference pointers to such objects when needed. As a result, garbage collection can be specified as any relation that removes portions of the current heap without affecting the result of a program's execution.

In Section 2, we present a small functional programming language, λgc, with a rewriting semantics that makes allocation explicit. We define a semantic notion of garbage collection for λgc and prove that there is no *optimal* collection strategy that is computable. In Section 3, we specify the "free-variable" garbage collection rule which models trace-based collectors including mark/sweep and copying collectors. We prove that the free-variable rule is correct and provide two "implementations" at the syntactic level: the first corresponds to a copying collector, the second to a generational one.

In Section 4, we formalize so-called "tag-free" collection algorithms for explicitly-typed, monomorphic languages such as Pascal and Algol [7, 29, 8]. We show how to *recover*

Abstract Models of Memory Management*

Greg Morrisett, Carnegie Mellon, jgmorris@cs.cmu.edu
Matthias Felleisen, Rice University, matthias@cs.rice.edu
Robert Harper, Carnegie Mellon, rwh@cs.cmu.edu

Abstract

Most specifications of garbage collectors concentrate on the low-level algorithmic details of *how* to find and preserve accessible objects. Often, they focus on bit-level manipulations such as "scanning stack frames," "marking objects," "tagging data," *etc.* While these details are important in some contexts, they often obscure the more fundamental aspects of memory management: *what* objects are garbage and *why*?

We develop a series of calculi that are just low-level enough that we can express allocation and garbage collection, yet are sufficiently abstract that we may formally prove the correctness of various memory management strategies. By making the heap of a program syntactically apparent, we can specify memory actions as rewriting rules that allocate values on the heap and automatically dereference pointers to such objects when needed. This formulation permits the specification of garbage collection as a relation that removes portions of the heap without affecting the outcome of the evaluation.

Our high-level approach allows us to specify in a compact manner a wide variety of memory management techniques, including standard trace-based garbage collection (*i.e.*, the family of copying and mark/sweep collection algorithms), generational collection, and type-based, tag-free collection. Furthermore, since the definition of garbage is based on the *semantics* of the underlying language instead of the conservative approximation of inaccessibility, we are able to specify and prove the idea that type inference can be used to collect some objects that are accessible but never used.

*This work was sponsored in part by the Advanced Research Projects Agency (ARPA), CSTO, under the title "The Fox Project: Advanced Development of Systems Software," ARPA Order No. 8313, issued by ESD/AVS under Contract No. F19628–91–C–0168, Wright Laboratory, Aeronautical Systems Center, Air Force Materiel Command, USAF, and ARPA grant No. F33615-93-1-1330. Views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing official policies or endorsements, either expressed or implied, of Wright Laboratory or the United States Government.

1 Memory Safety

Advanced programming languages manage memory allocation and deallocation automatically. Automatic memory managers, or garbage collectors, significantly facilitate the programming process because programmers can rely on the language implementation for the delicate tasks of finding and freeing unneeded objects. Indeed, the presence of a garbage collector ensures *memory safety* in the same way that a type system guarantees *type safety*: no program written in an advanced programming language will crash due to dangling pointer problems while allocation, access, and deallocation are transparent. However, in contrast to type systems, memory management strategies and particularly garbage collectors rarely come with a compact formulation and a formal proof of soundness. Since garbage collectors work on the machine representations of abstract values, the very idea of providing a proof of memory safety sounds unrealistic given the lack of simple models of memory operations.

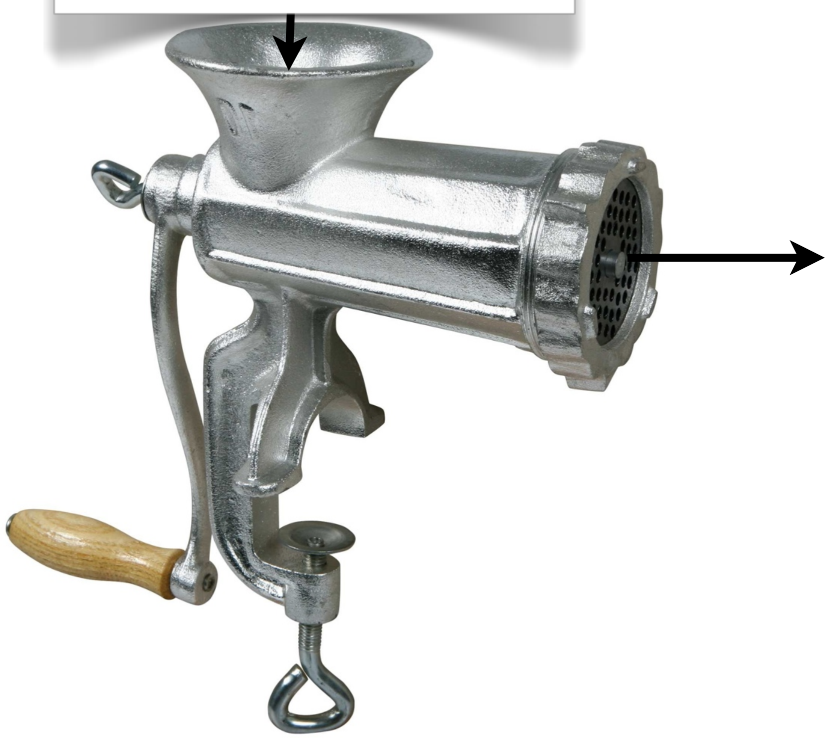
The recently developed syntactic approaches to the specification of language semantics by Felleisen and Hieb [11] and Mason and Talcott [18, 19] are the first execution models that are intensional enough to permit the specification of memory management actions and yet are sufficiently abstract to permit compact proofs of important properties. Starting from the λ_v-**S** calculus of Felleisen and Hieb, we design compact specifications of a number of memory management ideas and prove several correctness theorems.

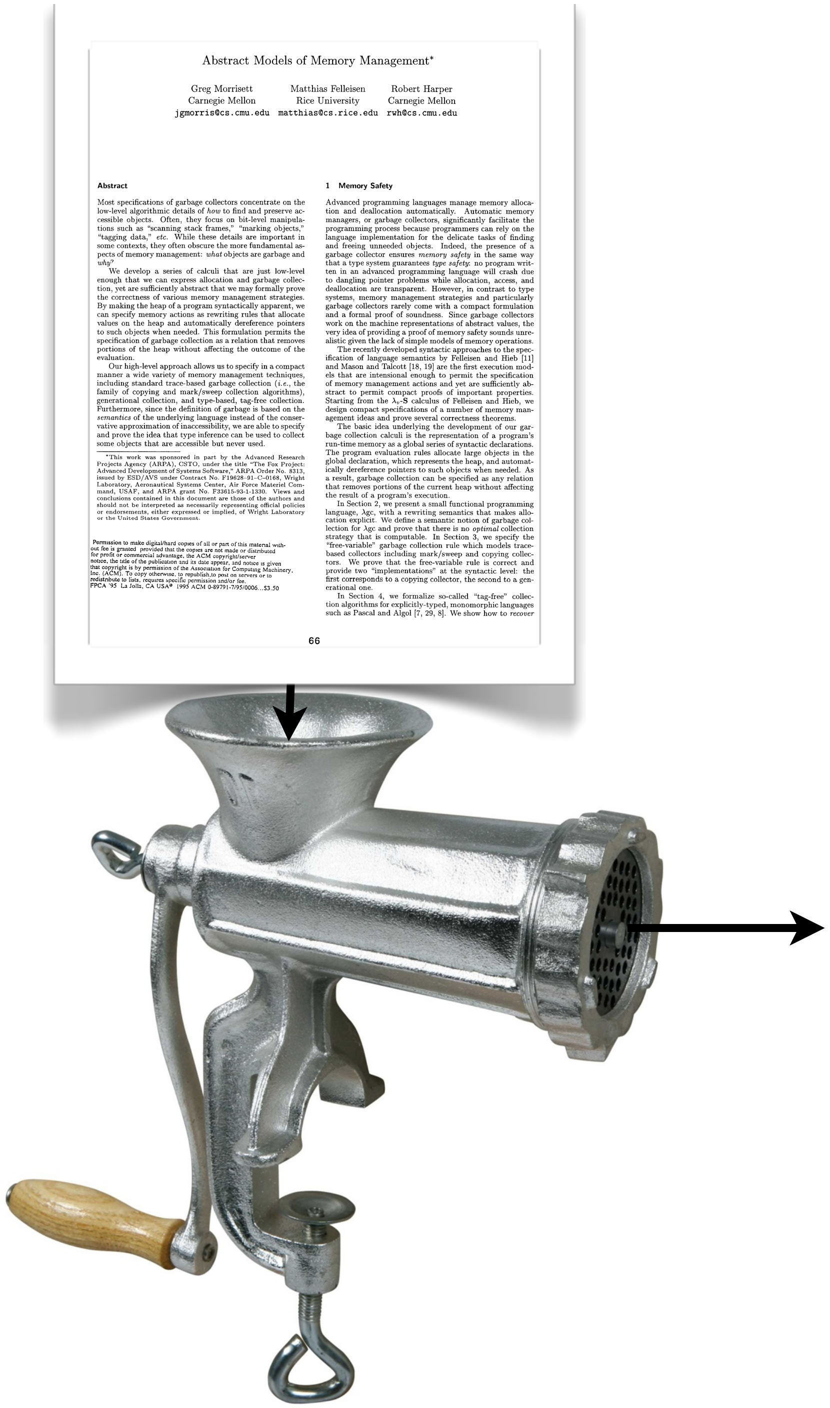
The basic idea underlying the development of our garbage collection calculi is the representation of a program's run-time memory as a global series of syntactic declarations. The program evaluation rules allocate large objects in the global declaration, which represents the heap, and automatically dereference pointers to such objects when needed. As a result, garbage collection can be specified as any relation that removes portions of the current heap without affecting the result of a program's execution.

In Section 2, we present a small functional programming language, λgc, with a rewriting semantics that makes allocation explicit. We define a semantic notion of garbage collection for λgc and prove that there is no *optimal* collection strategy that is computable. In Section 3, we specify the "free-variable" garbage collection rule which models trace-based collectors including mark/sweep and copying collectors. We prove that the free-variable rule is correct and provide two "implementations" at the syntactic level: the first corresponds to a copying collector, the second to a generational one.

In Section 4, we formalize so-called "tag-free" collection algorithms for explicitly-typed, monomorphic languages such as Pascal and Algol [7, 29, 8]. We show how to *recover*

66





Improved precision and efficiency via abstract GC

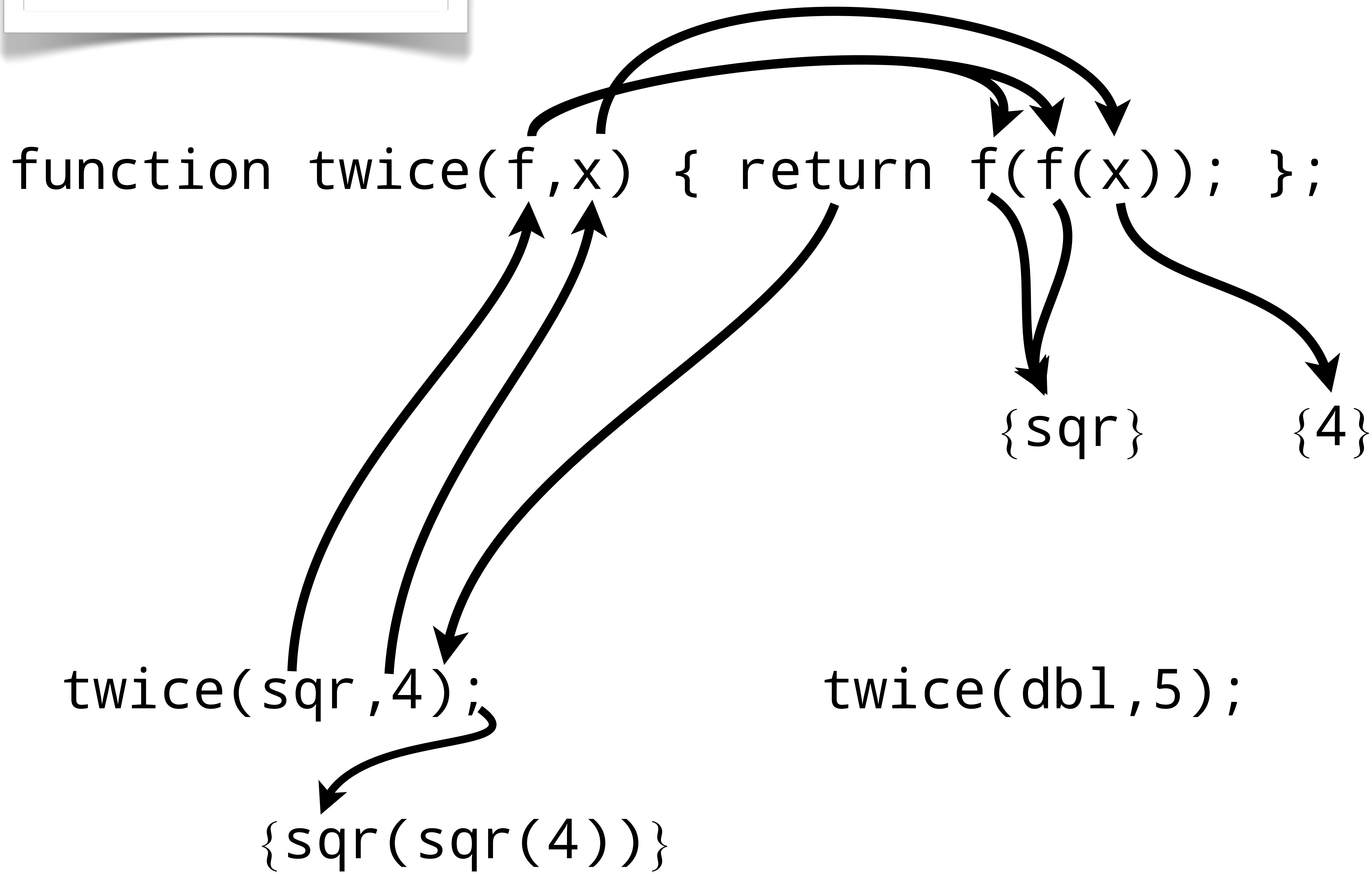
0CFA + GC

```
function twice(f,x) { return f(f(x)); };
```

```
twice(sqr,4);
```

```
twice(dbl,5);
```

0CFA + GC


function twice(f,x) { return f(f(x)); };
{sqr}
{4}
twice(sqr,4);
twice(dbl,5);
{sqr(sqr(4))}

0CFA + GC

```
function twice(f,x) { return f(f(x)); };
```

```
twice(sqr,4);                twice(dbl,5);
```

{sqr(sqr(4))}

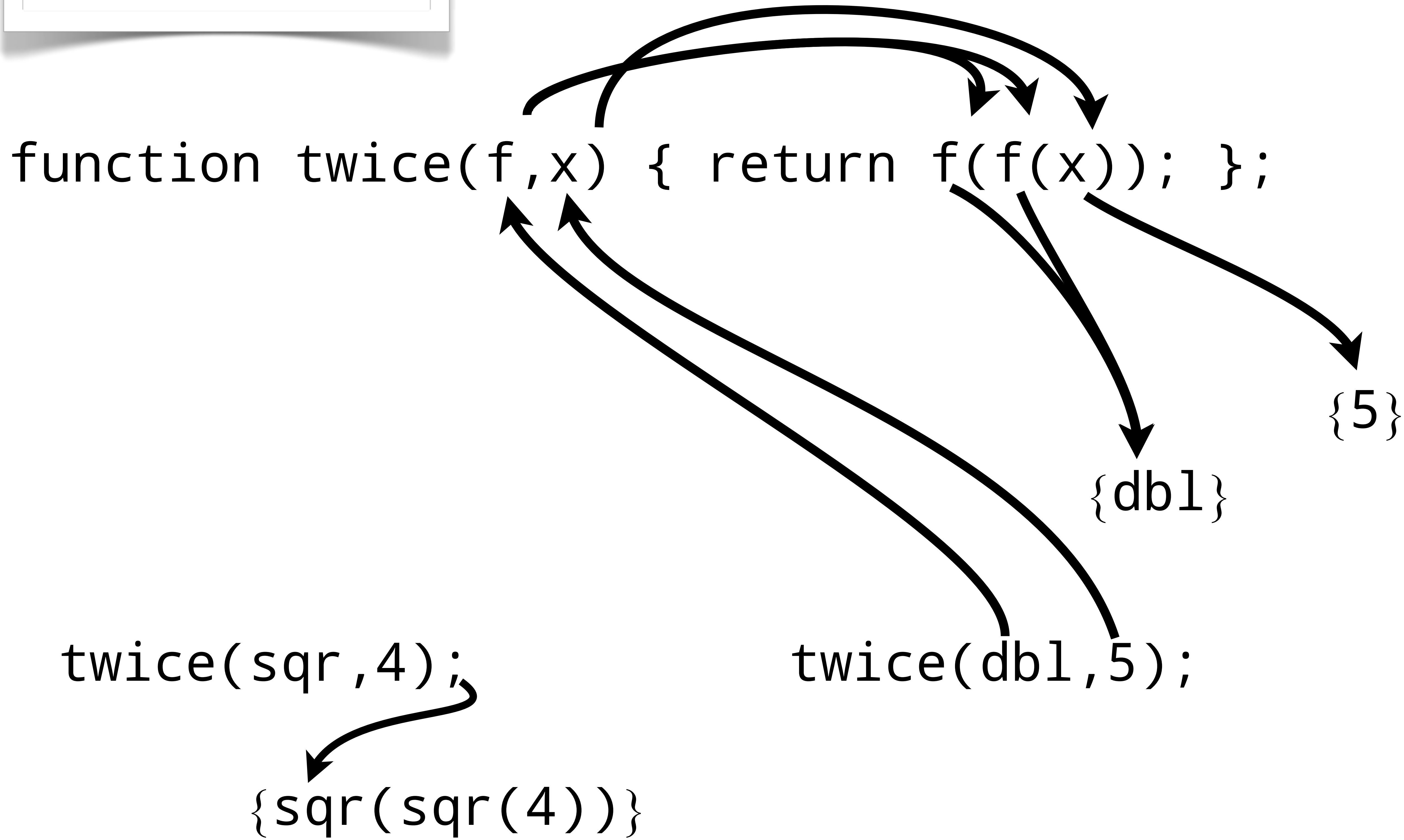
0CFA + GC

```
function twice(f,x) { return f(f(x)); };
```

```
twice(sqr,4);                twice(dbl,5);
```

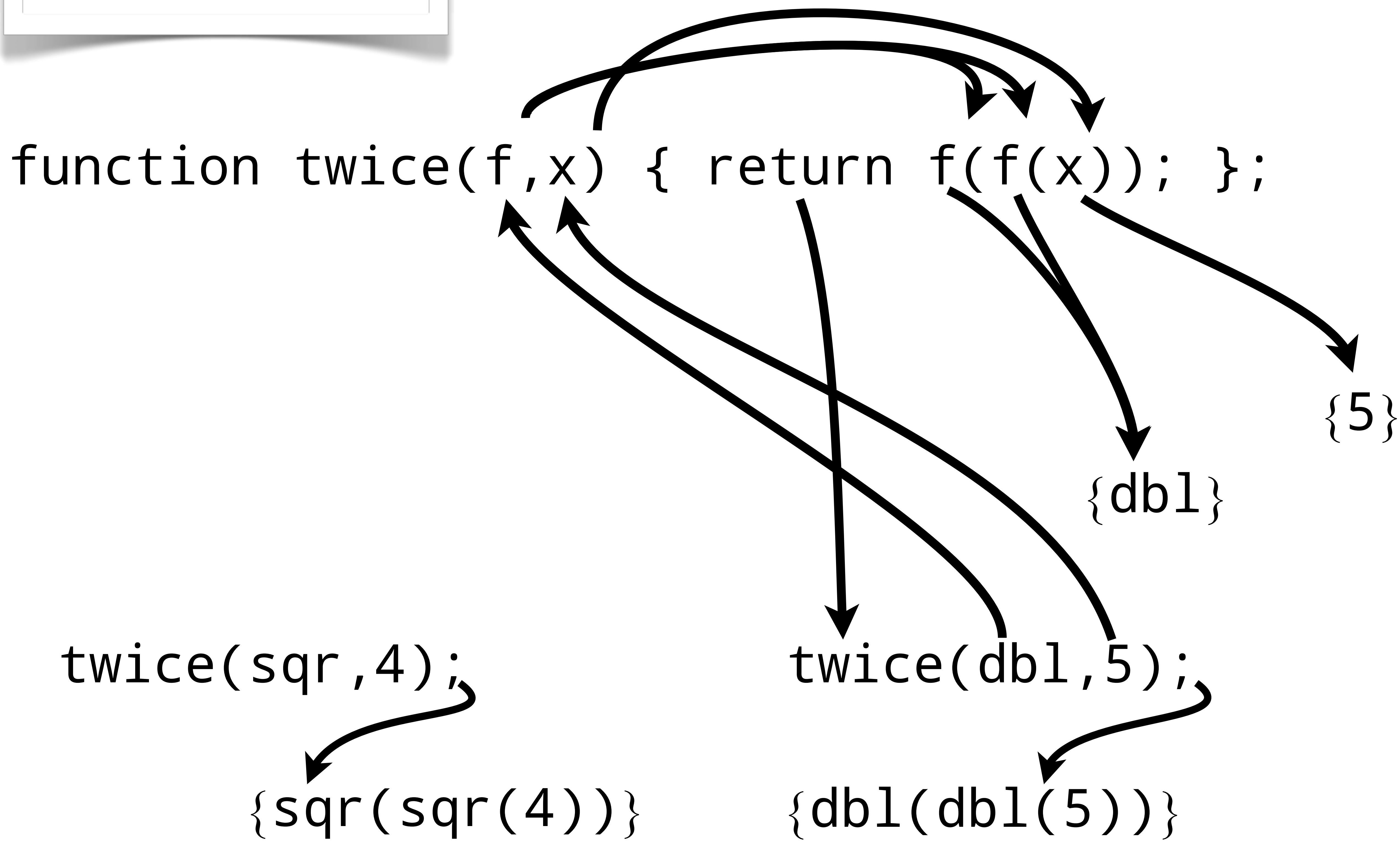
{sqr(sqr(4))}

0CFA + GC

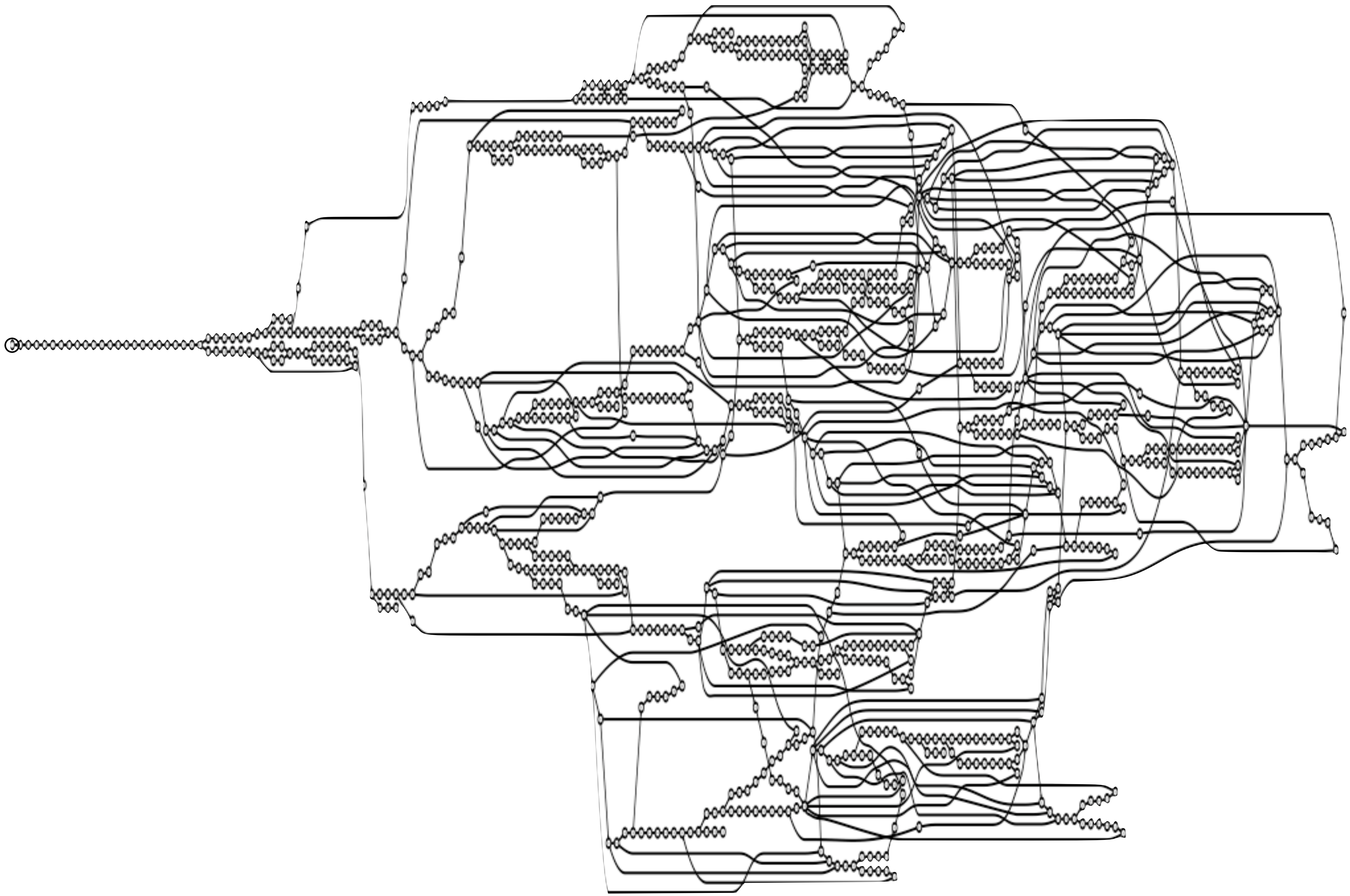


function twice(f,x) { return f(f(x)); };
{5}
{dbl}
twice(sqr,4);
twice(dbl,5);
{sqr(sqr(4))}

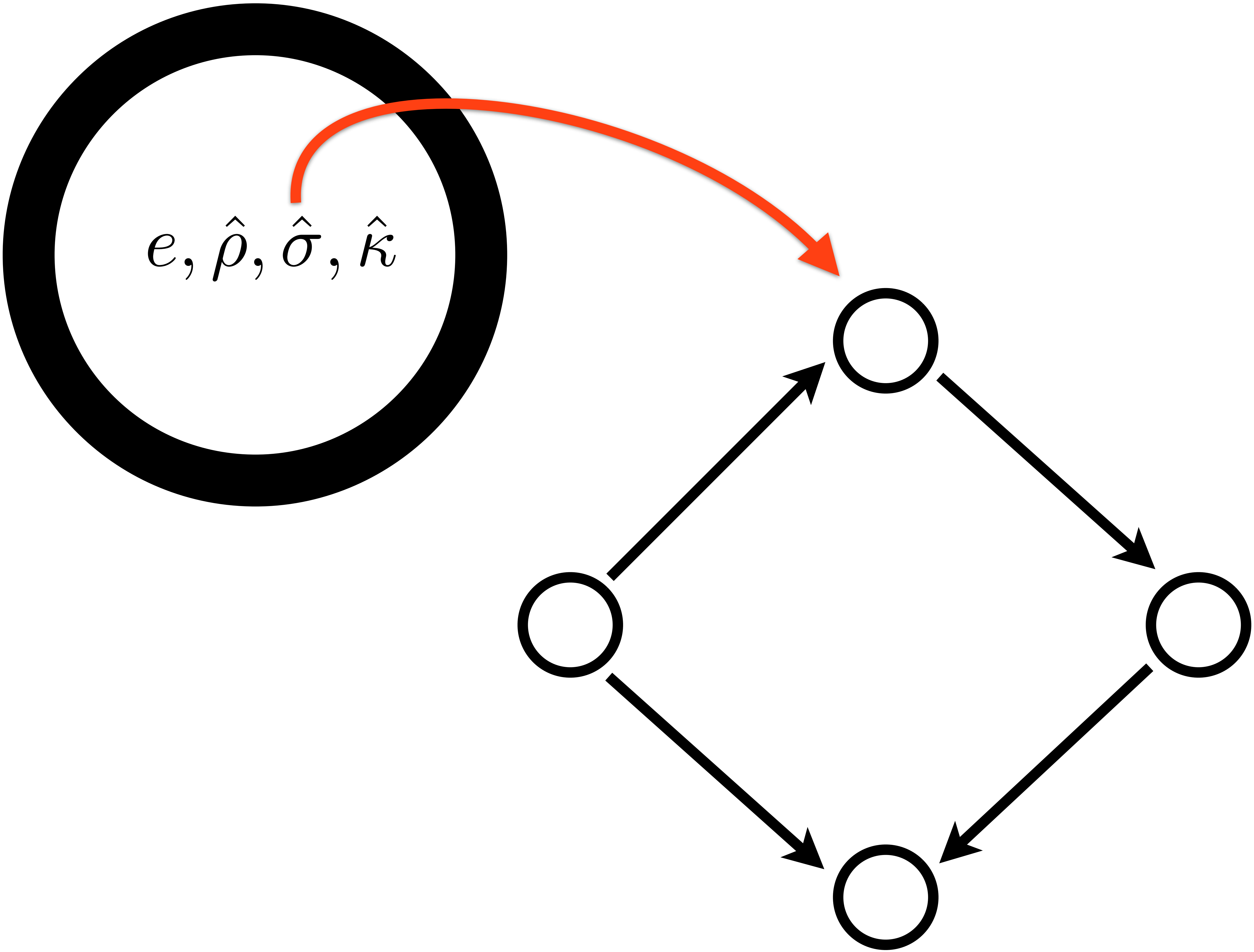
0CFA + GC

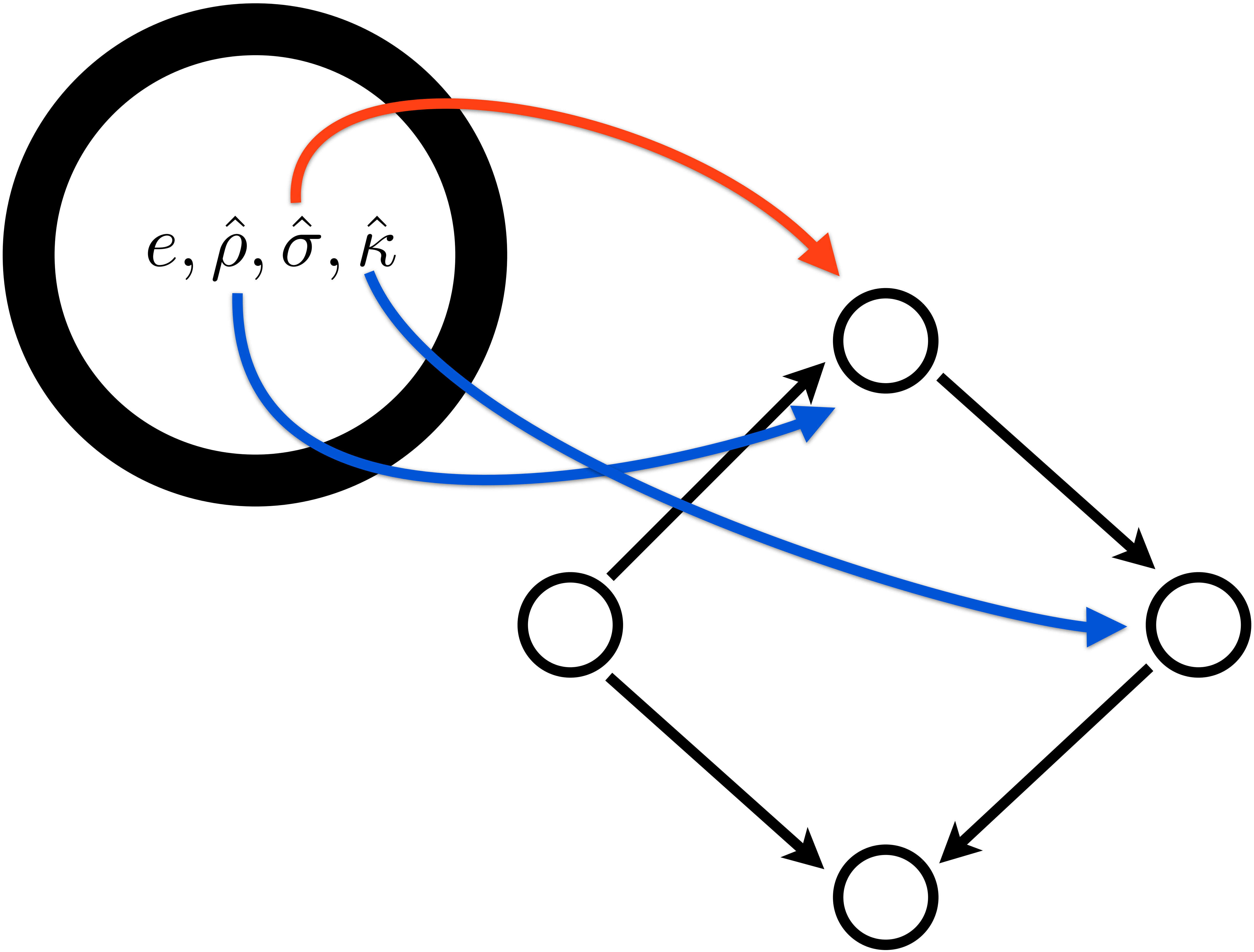


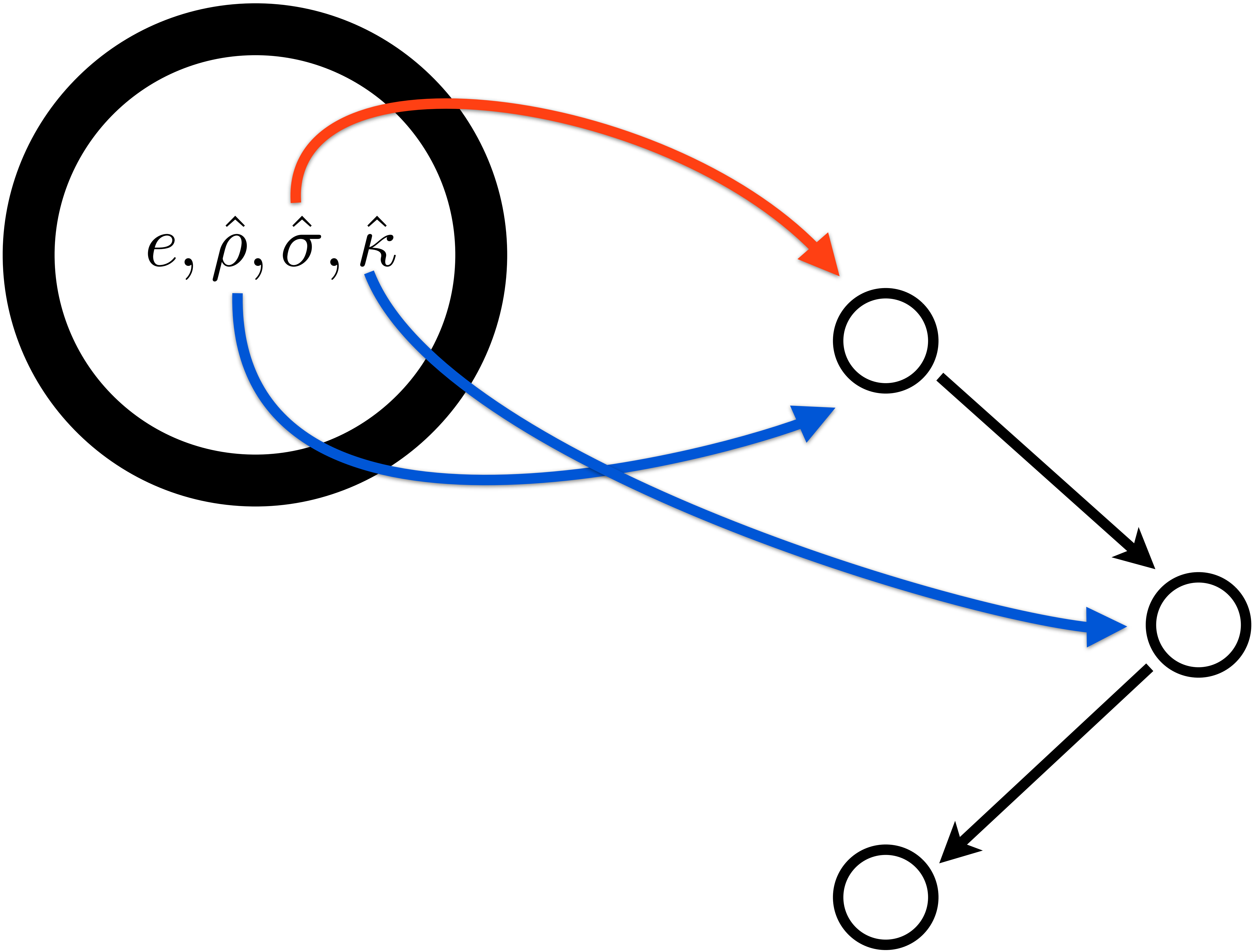
function twice(f,x) { return f(f(x)); };
{5}
{dbl}
twice(sqr,4);
twice(dbl,5);
{sqr(sqr(4))}
{dbl(dbl(5))}



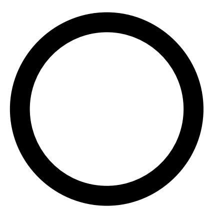

$e, \hat{\rho}, \hat{\sigma}, \hat{\kappa}$

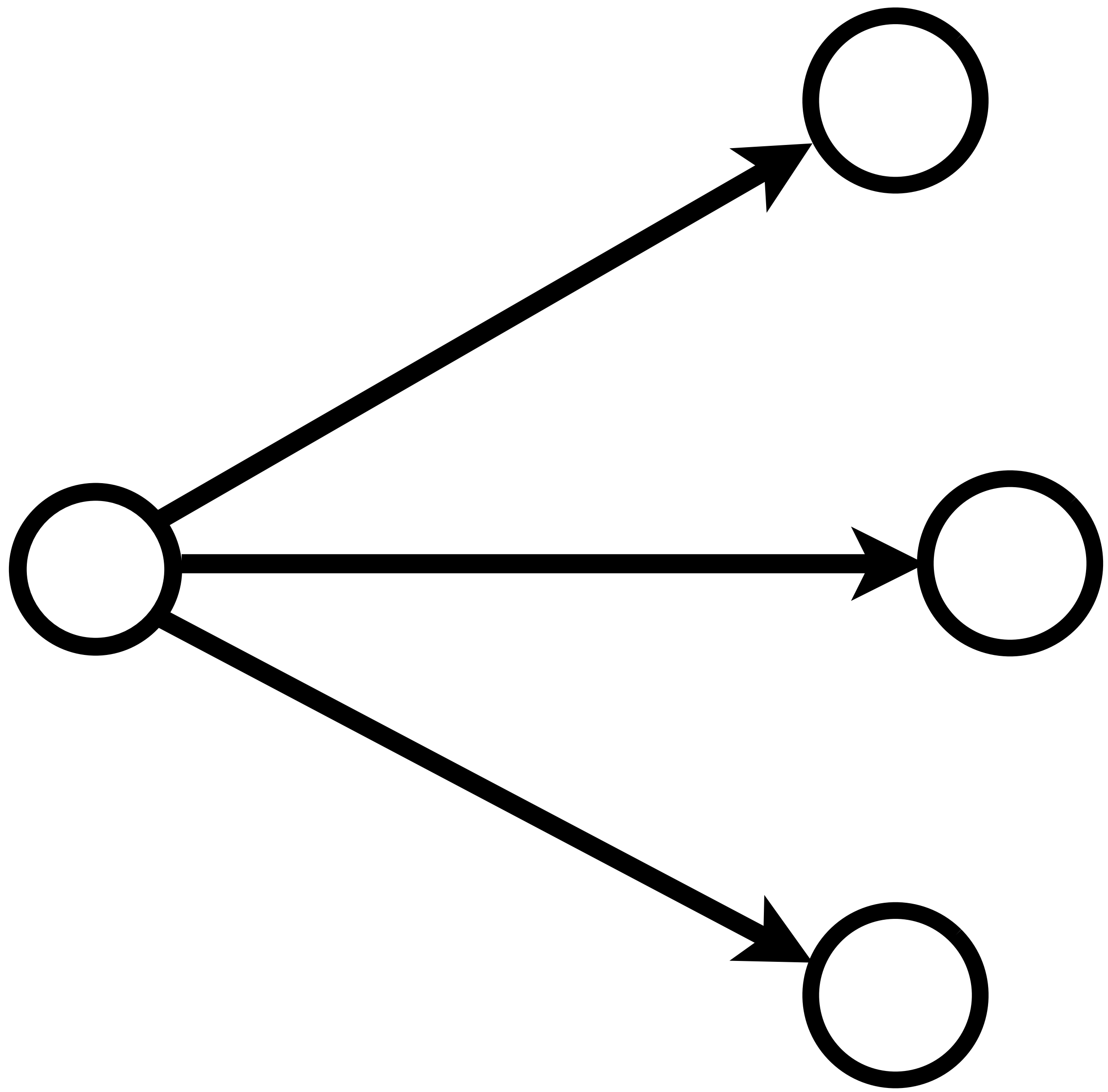

$e, \hat{\rho}, \hat{\sigma}, \hat{\kappa}$

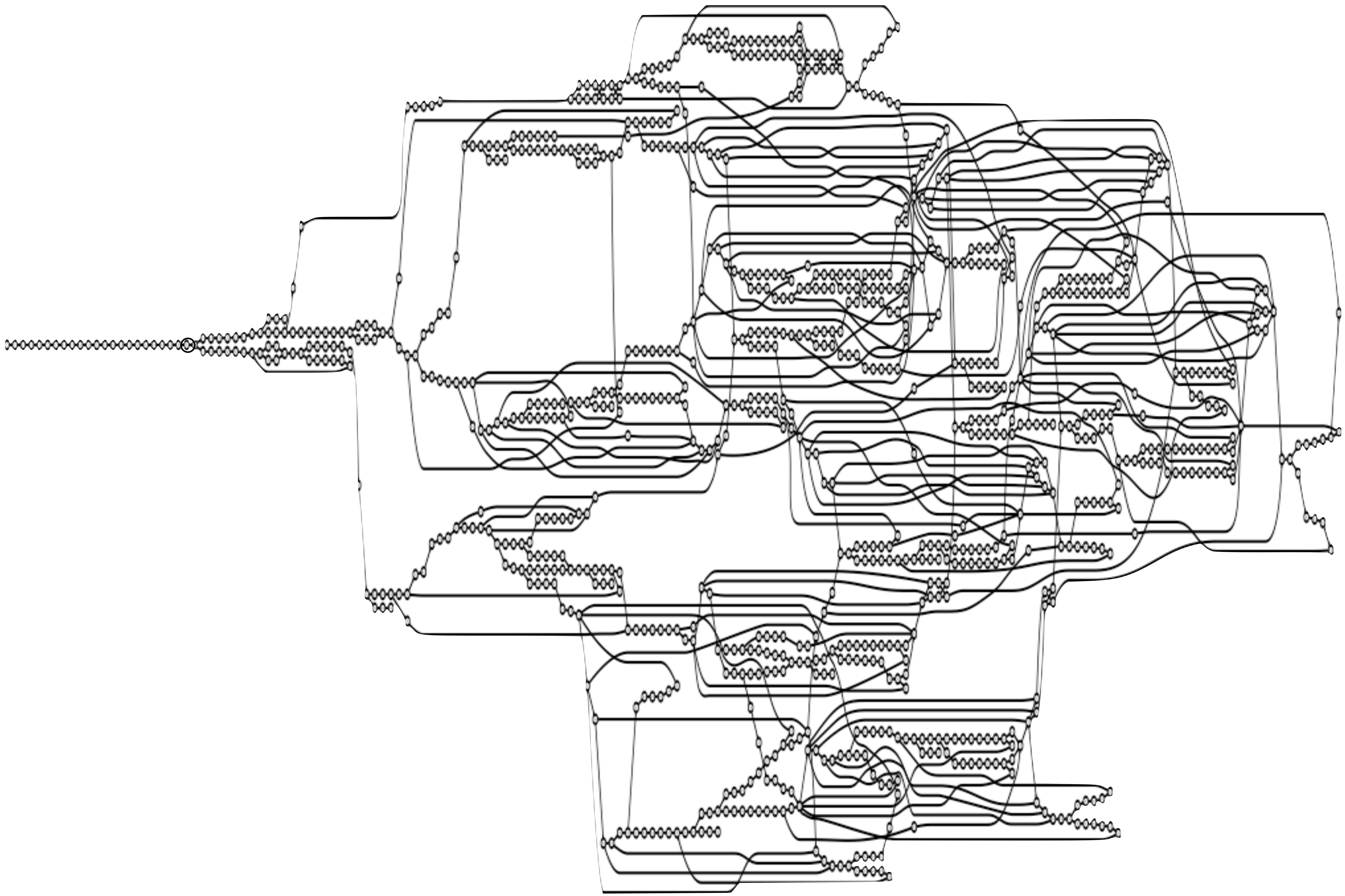

$e, \hat{\rho}, \hat{\sigma}, \hat{\kappa}$

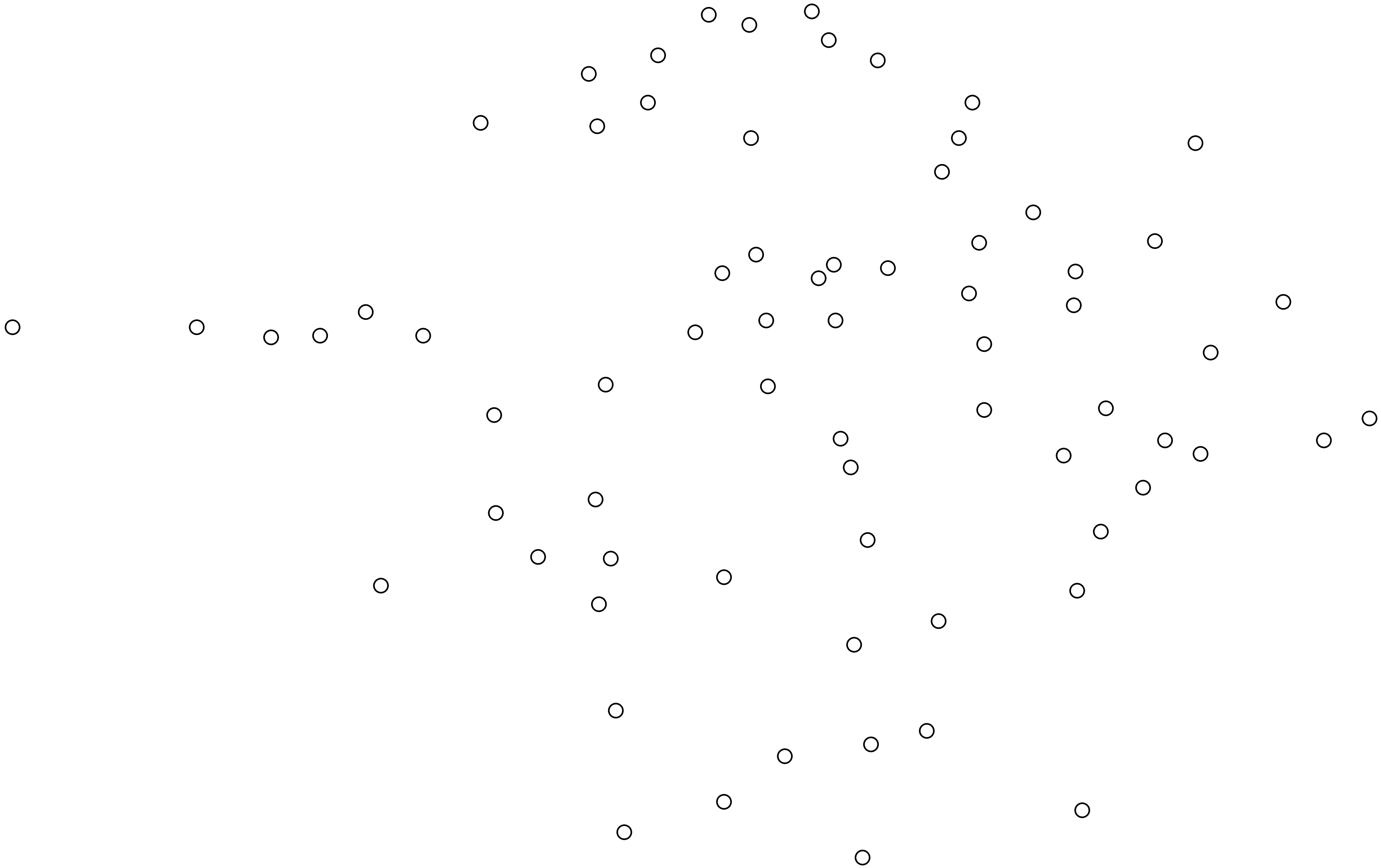

$e, \hat{\rho}, \hat{\sigma}, \hat{\kappa}$

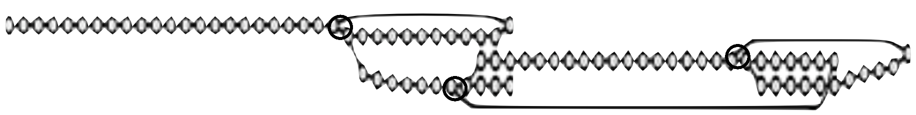

$e, \hat{\rho}, \hat{\sigma}', \hat{\kappa}$





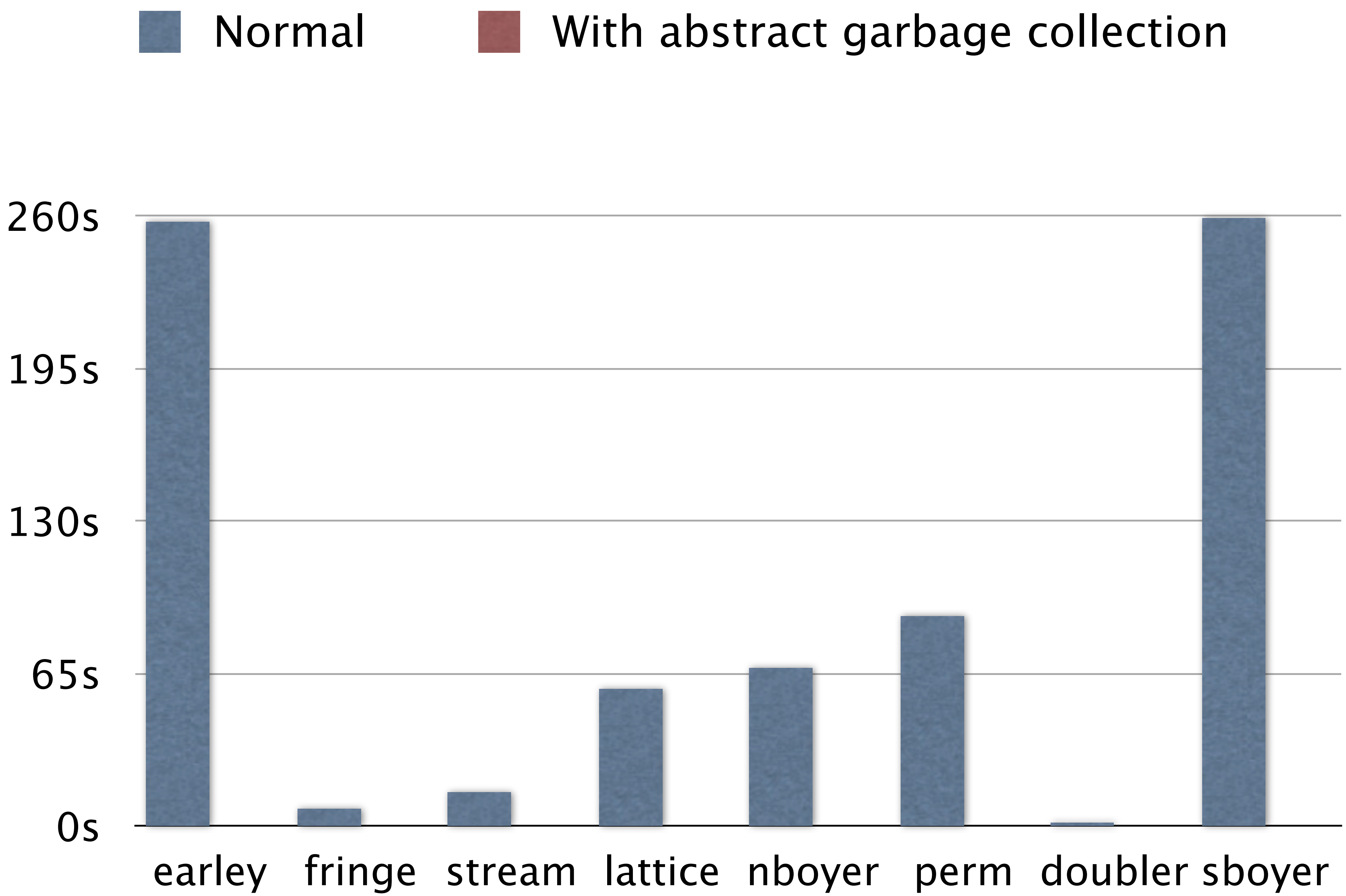




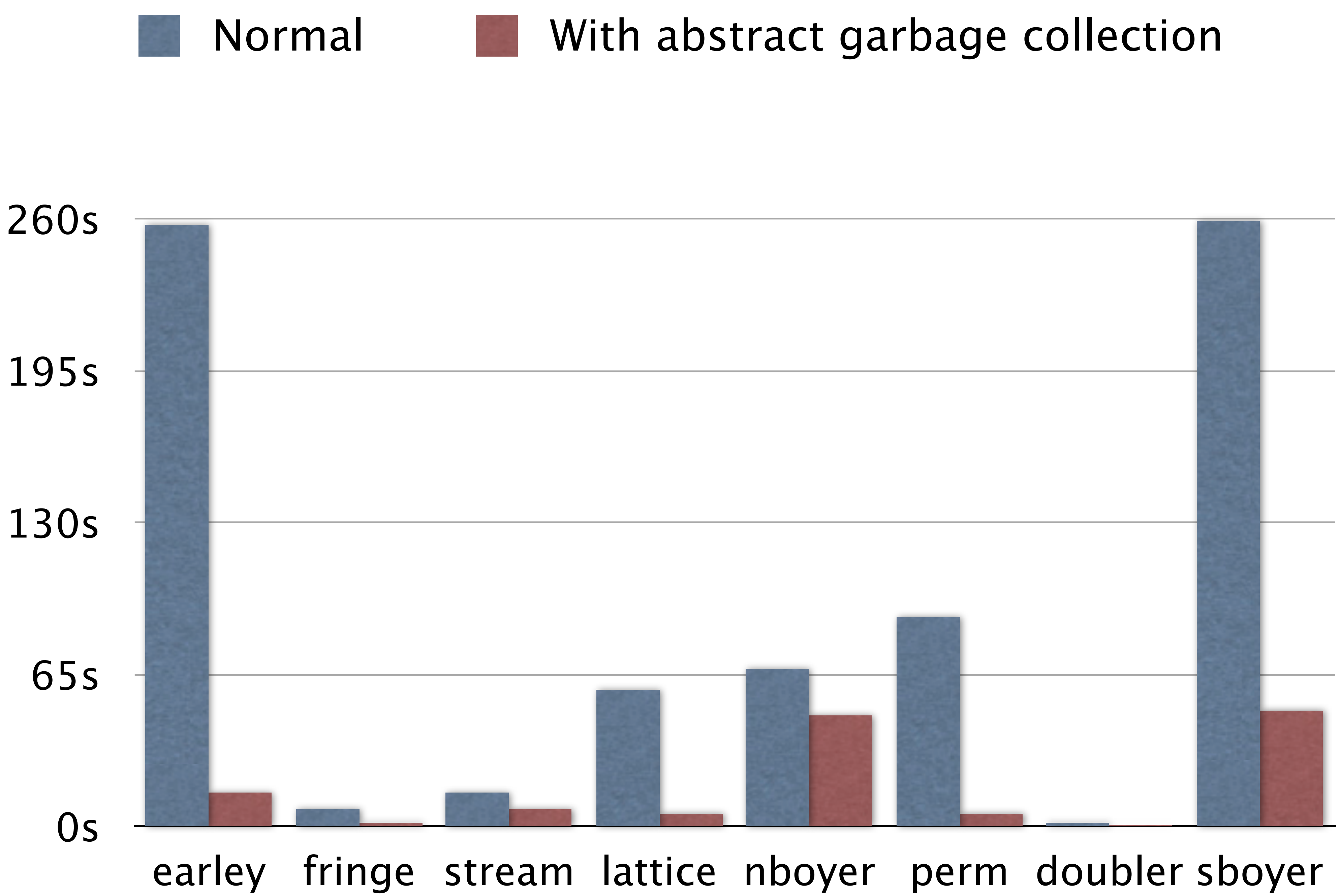


Normal   With abstract garbage collection

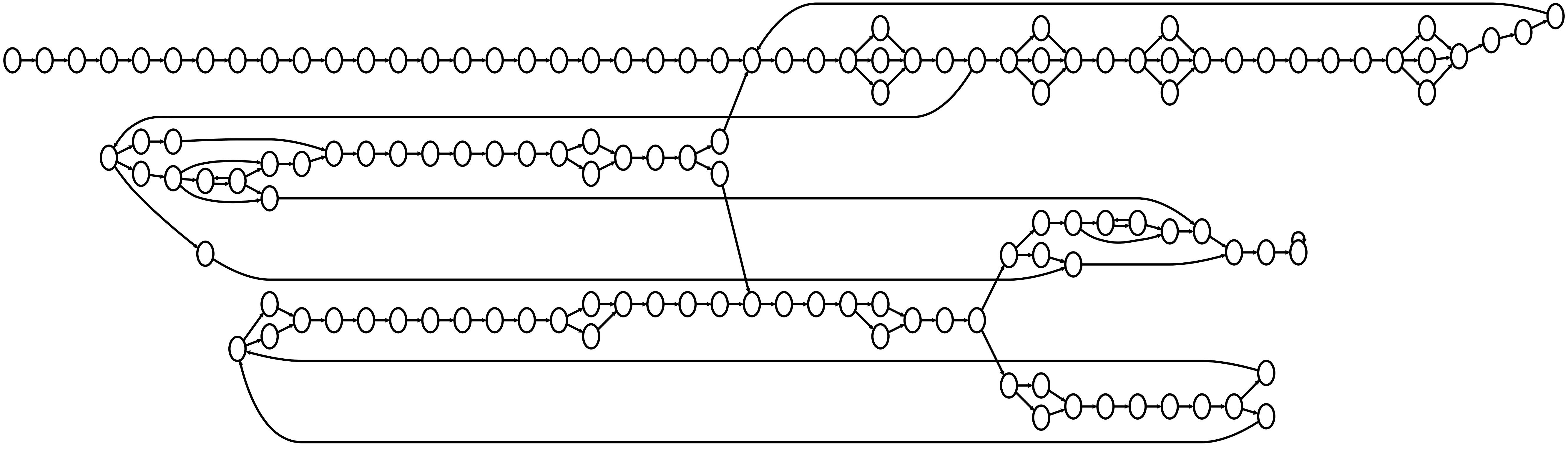
Analysis time



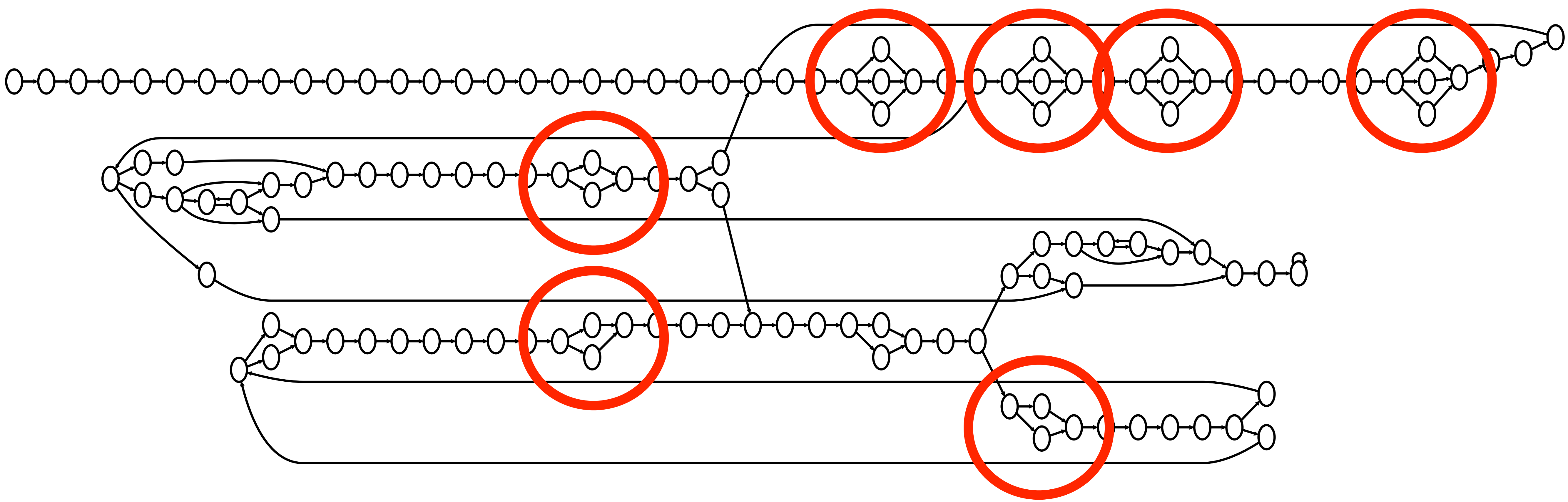
Analysis time



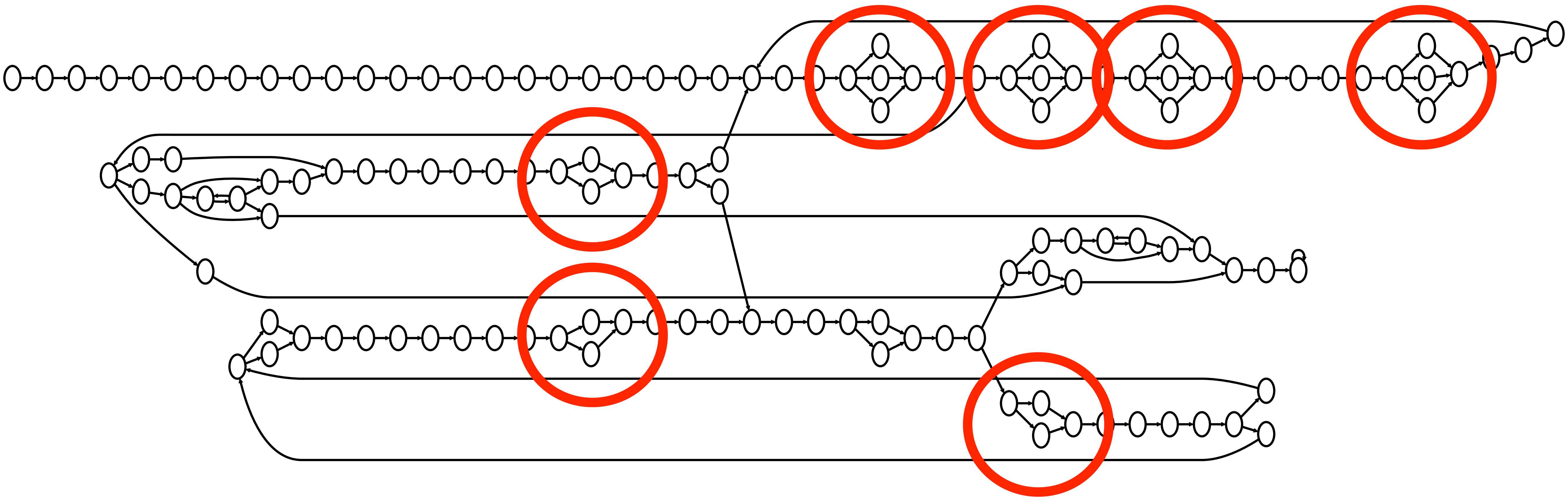
Analysis time



Problem: needless non-determinism

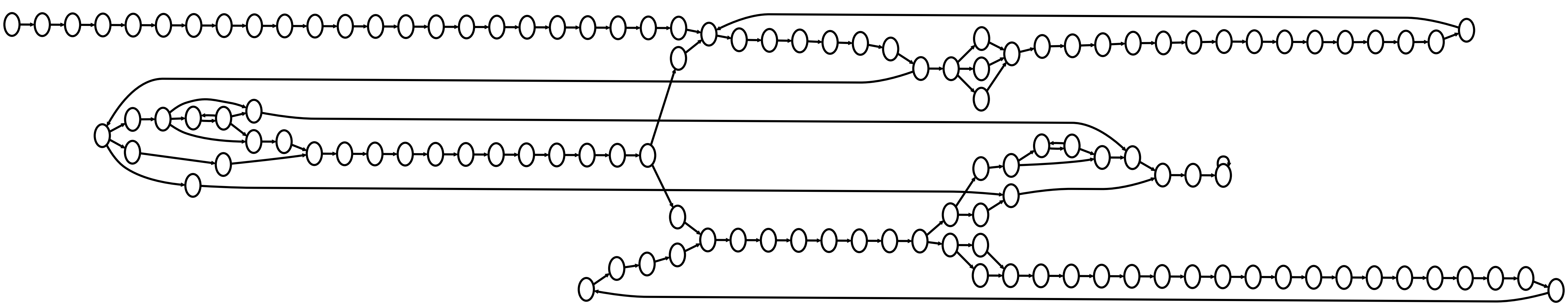


Problem: needless non-determinism

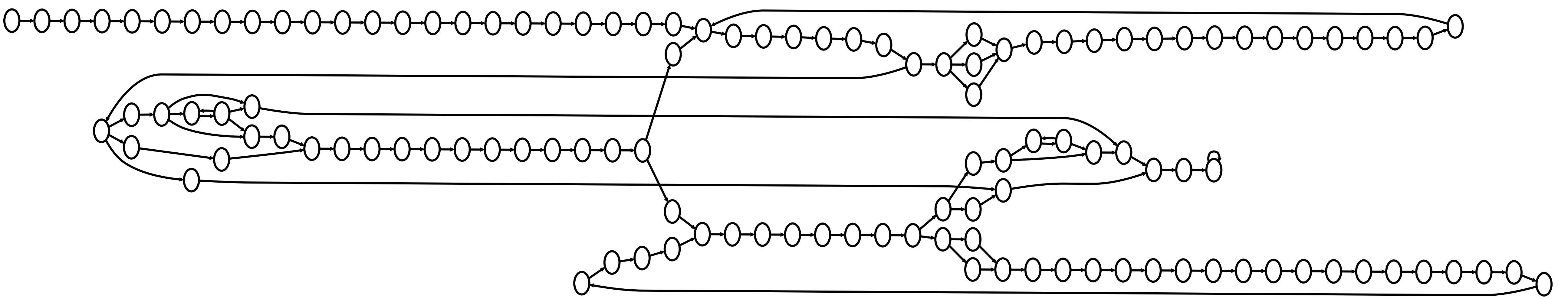


Solution: Lazy non-determinism

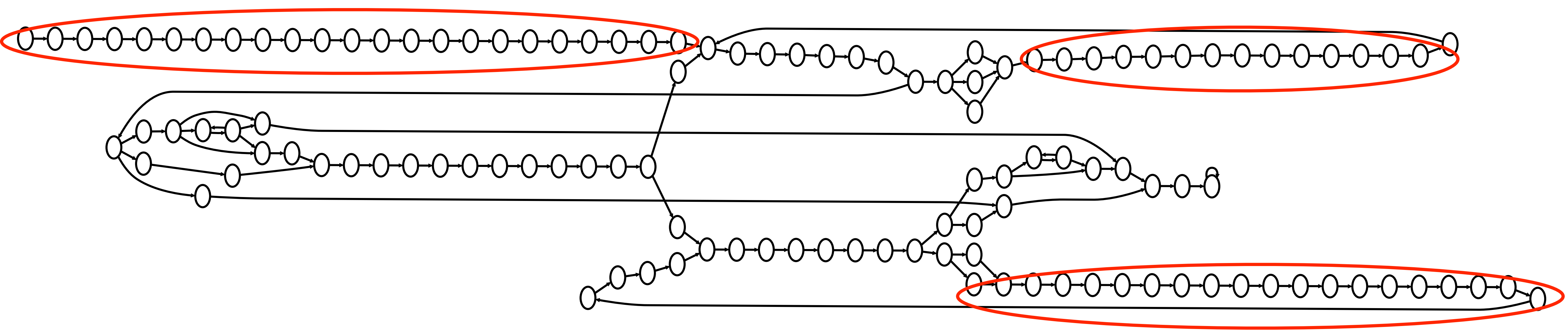
Problem: needless non-determinism



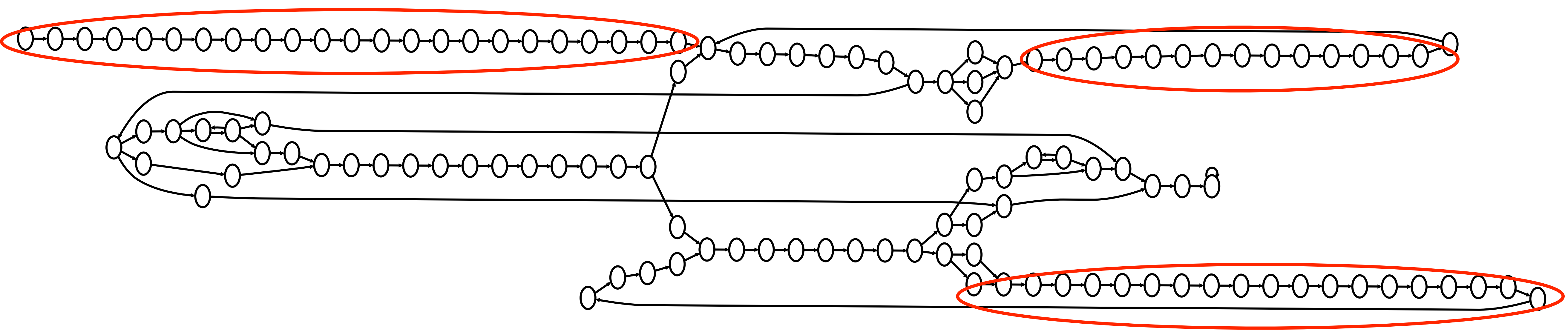
Solution: Lazy non-determinism



Problem: Long Corridors

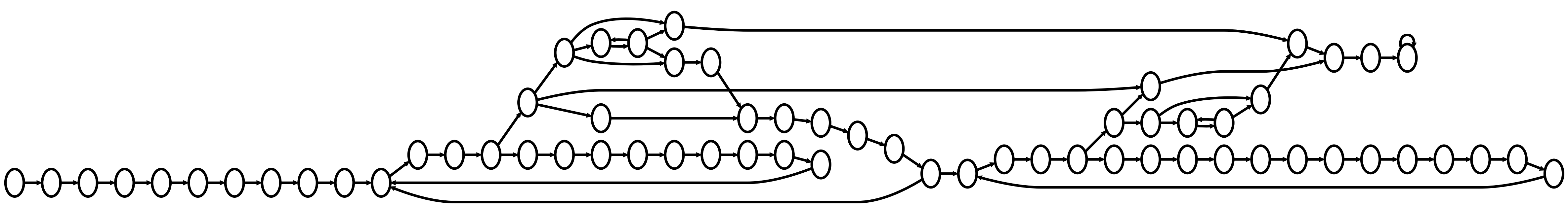


Problem: Long Corridors

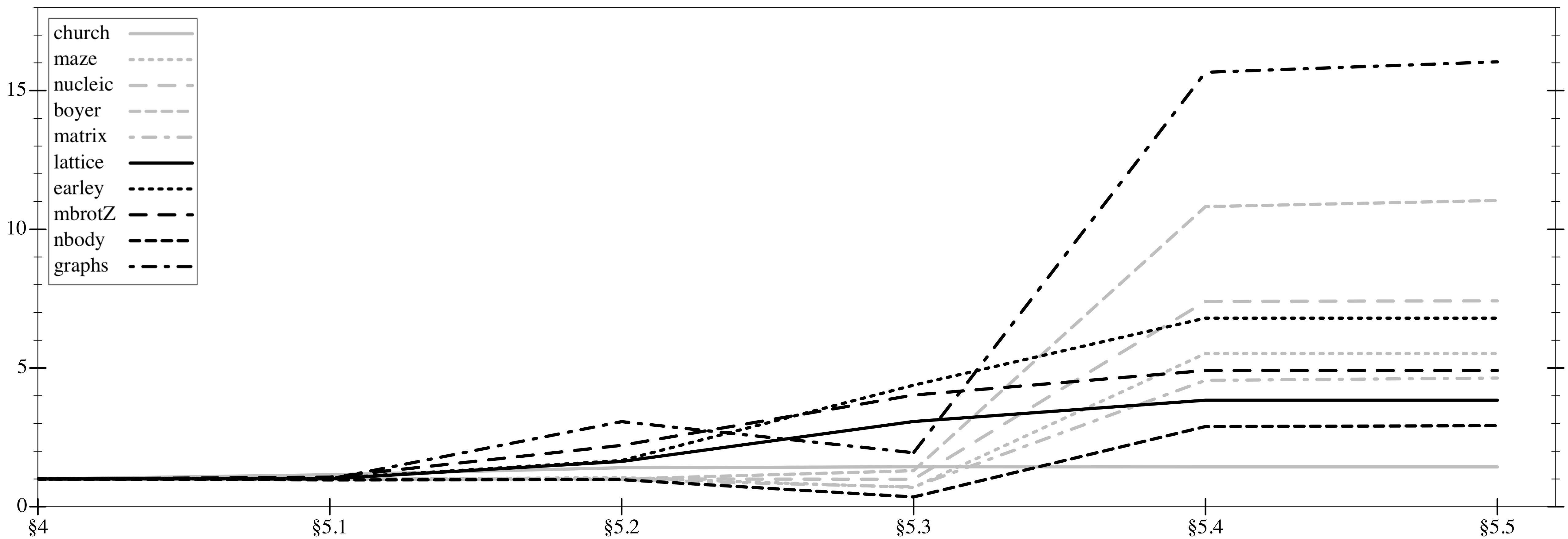


Solution: Abstract Compilation

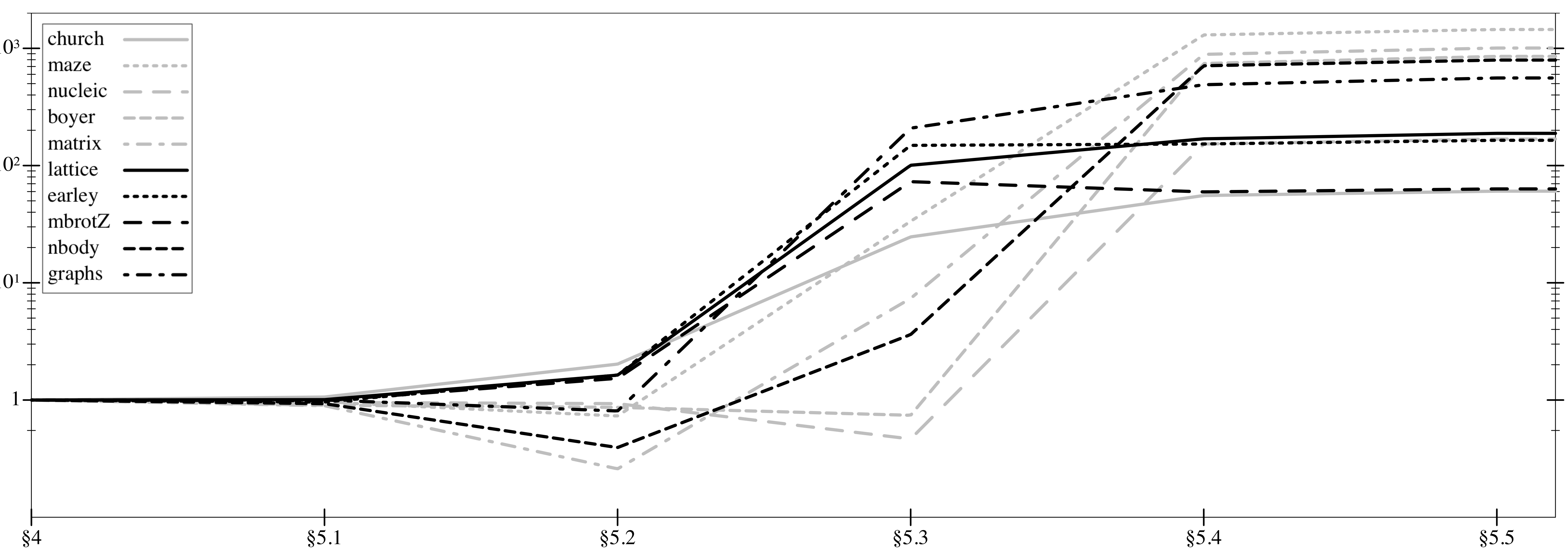
Problem: Long Corridors



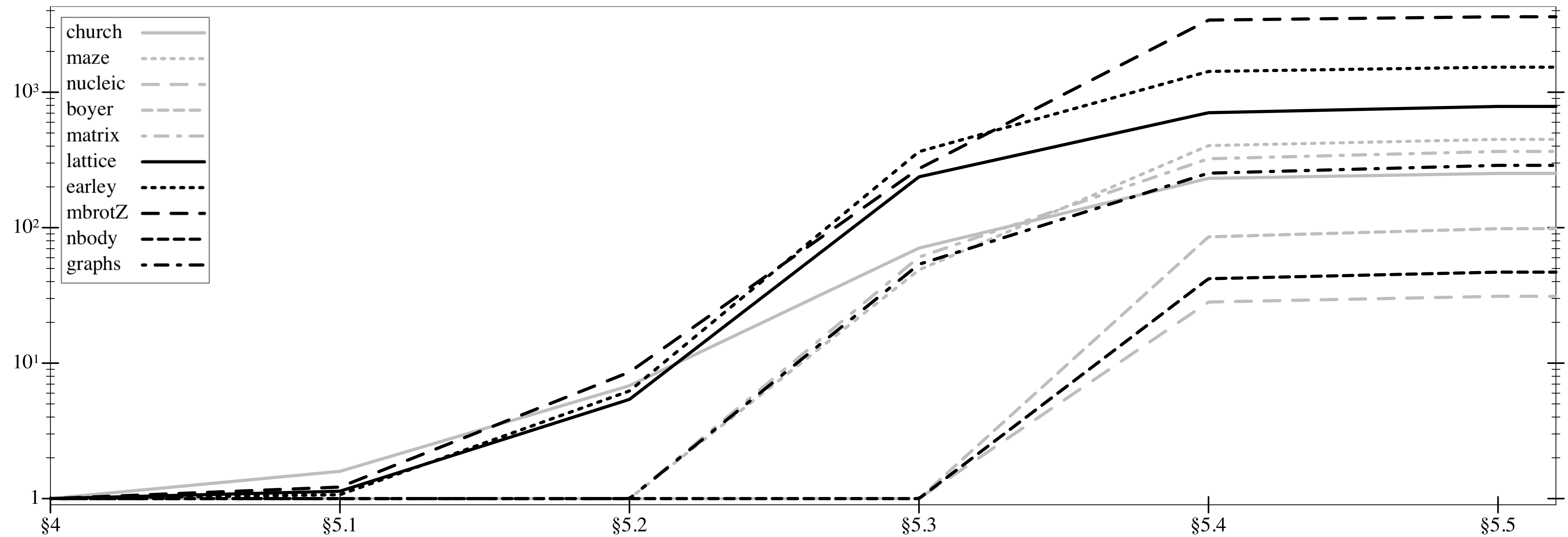
Solution: Abstract Compilation



Factor improvement of Peak Memory usage



Factor improvement of Speed of transitions

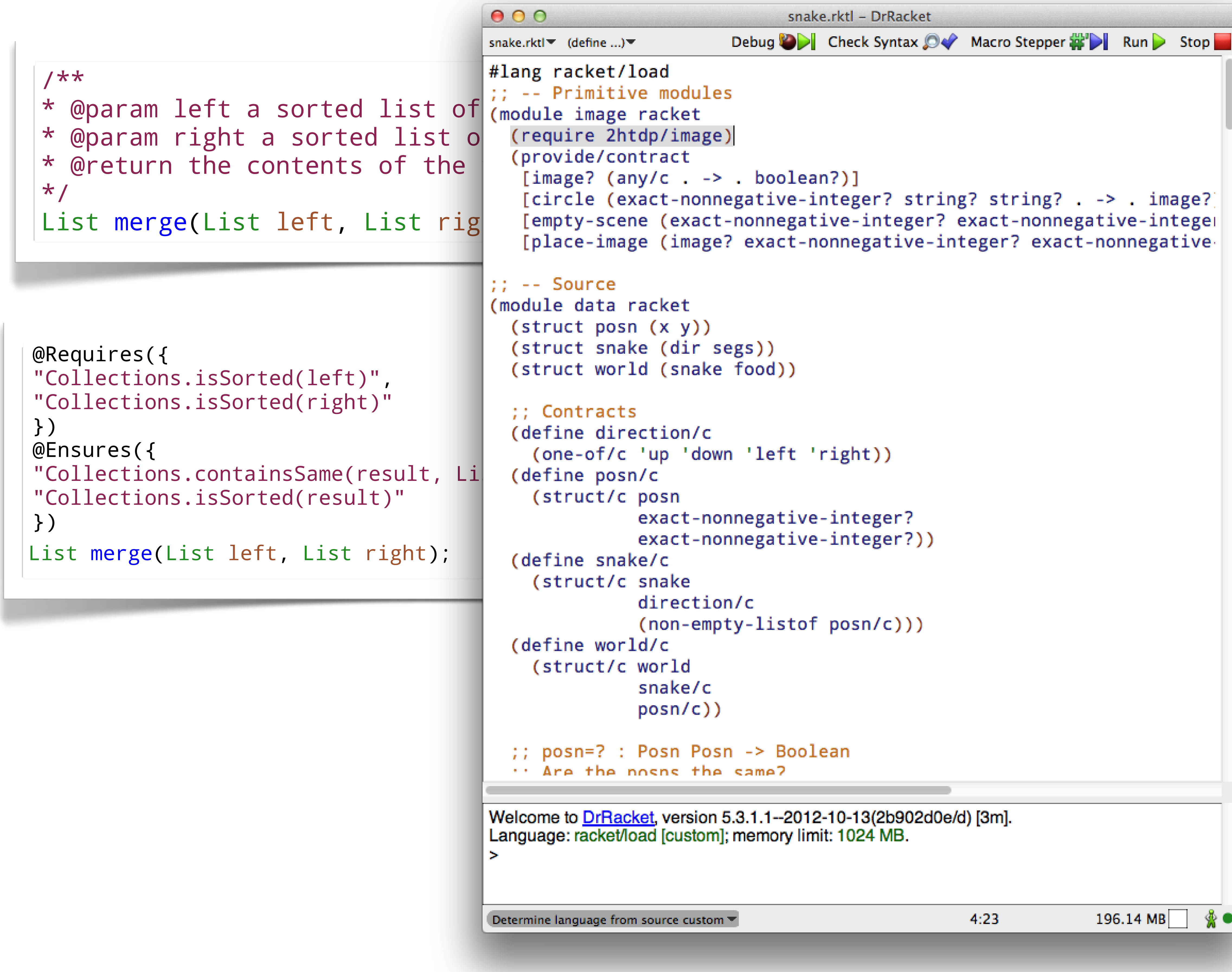


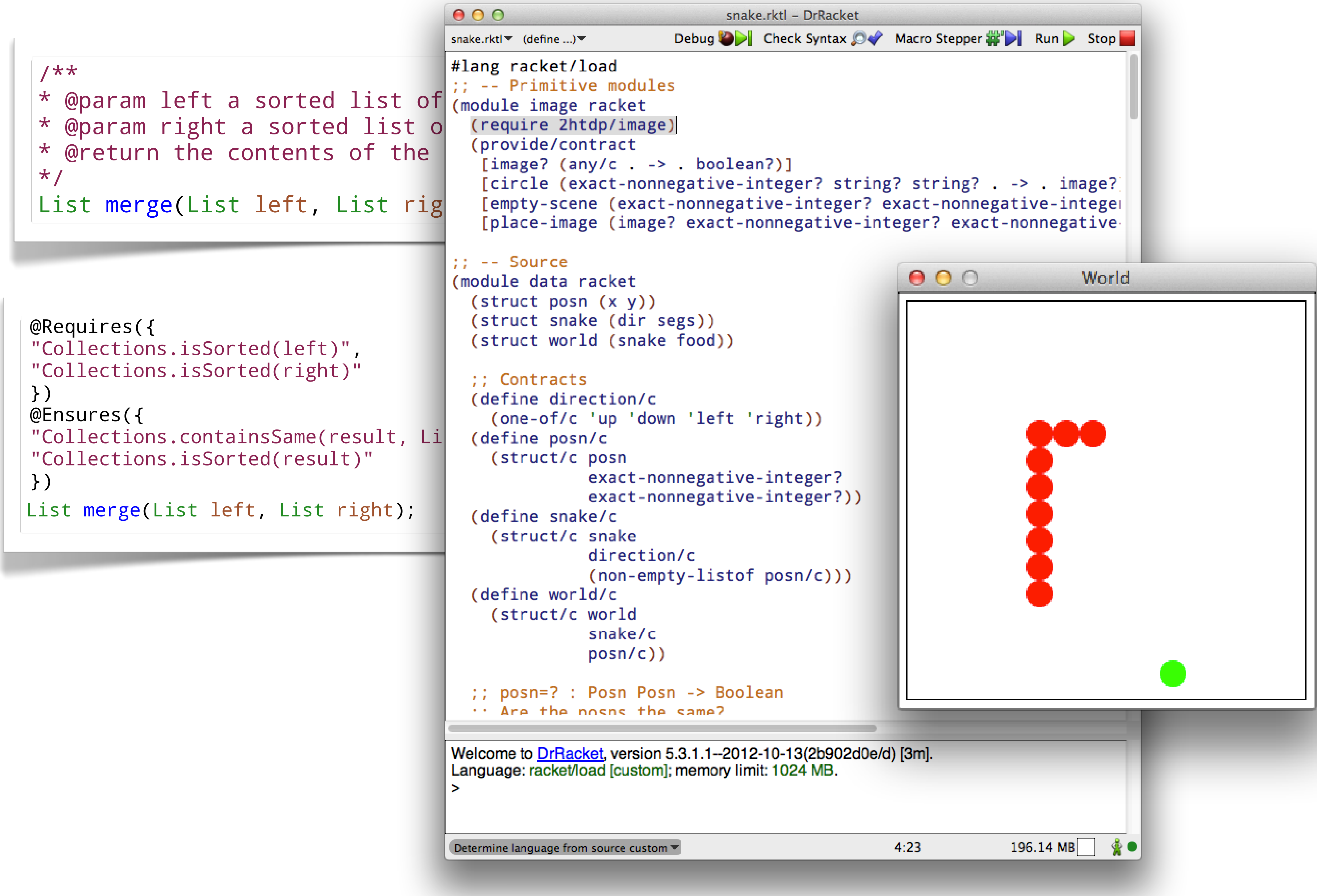
Factor improvement of Analysis Time

Behavioral Software Contract Verification

```
/**
* @param left a sorted list of elements
* @param right a sorted list of elements
* @return the contents of the two lists, merged, sorted
*/
List merge(List left, List right);
```

```
@Requires({
"Collections.isSorted(left)",
"Collections.isSorted(right)"
})
@Ensures({
"Collections.containsSame(result, Lists.concatenate(left, right))",
"Collections.isSorted(result)"
})
List merge(List left, List right);
```


/**
* @param left a sorted list of
* @param right a sorted list o
* @return the contents of the
*/
List merge(List left, List rig
@Requires({
"Collections.isSorted(left)",
"Collections.isSorted(right)"
})
@Ensures({
"Collections.containsSame(result, Li
"Collections.isSorted(result)"
})
List merge(List left, List right);
snake.rktl – DrRacket
snake.rktl (define ...)
Debug
Check Syntax
Macro Stepper
Run
Stop
#lang racket/load
;; -- Primitive modules
(module image racket
(require 2htdp/image)
(provide/contract
[image? (any/c . -> . boolean?)]
[circle (exact-nonnegative-integer? string? string? . -> . image?
[empty-scene (exact-nonnegative-integer? exact-nonnegative-intege
[place-image (image? exact-nonnegative-integer? exact-nonnegative
;; -- Source
(module data racket
(struct posn (x y))
(struct snake (dir segs))
(struct world (snake food))
;; Contracts
(define direction/c
(one-of/c 'up 'down 'left 'right))
(define posn/c
(struct/c posn
exact-nonnegative-integer?
exact-nonnegative-integer?))
(define snake/c
(struct/c snake
direction/c
(non-empty-listof posn/c)))
(define world/c
(struct/c world
snake/c
posn/c))
;; posn=? : Posn Posn -> Boolean
Welcome to DrRacket, version 5.3.1.1--2012-10-13(2b902d0e/d) [3m].
Language: racket/load [custom]; memory limit: 1024 MB.
>
Determine language from source custom
4:23
196.14 MB


/**
* @param left a sorted list of
* @param right a sorted list o
* @return the contents of the
*/
List merge(List left, List rig
@Requires({
"Collections.isSorted(left)",
"Collections.isSorted(right)"
})
@Ensures({
"Collections.containsSame(result, Li
"Collections.isSorted(result)"
})
List merge(List left, List right);
snake.rktl – DrRacket
snake.rktl (define ...)
Debug
Check Syntax
Macro Stepper
Run
Stop
#lang racket/load
;; -- Primitive modules
(module image racket
(require 2htdp/image)
(provide/contract
[image? (any/c . -> . boolean?)]
[circle (exact-nonnegative-integer? string? string? . -> . image?
[empty-scene (exact-nonnegative-integer? exact-nonnegative-intege
[place-image (image? exact-nonnegative-integer? exact-nonnegative
;; -- Source
(module data racket
(struct posn (x y))
(struct snake (dir segs))
(struct world (snake food))
;; Contracts
(define direction/c
(one-of/c 'up 'down 'left 'right))
(define posn/c
(struct/c posn
exact-nonnegative-integer?
exact-nonnegative-integer?))
(define snake/c
(struct/c snake
direction/c
(non-empty-listof posn/c)))
(define world/c
(struct/c world
snake/c
posn/c))
;; posn=? : Posn Posn -> Boolean
;; Are the posns the same?
World
Welcome to DrRacket, version 5.3.1.1--2012-10-13(2b902d0e/d) [3m].
Language: racket/load [custom]; memory limit: 1024 MB.
>
Determine language from source custom
4:23
196.14 MB

Contracts for Higher-Order Functions

Robert Bruce Findler[1] Matthias Felleisen
Northeastern University
College of Computer Science
Boston, Massachusetts 02115, USA

Abstract

Assertions play an important role in the construction of robust software. Their use in programming languages dates back to the 1970s. Eiffel, an object-oriented programming language, wholeheartedly adopted assertions and developed the "Design by Contract" philosophy. Indeed, the entire object-oriented community recognizes the value of assertion-based contracts on methods.

In contrast, languages with higher-order functions do not support assertion-based contracts. Because predicates on functions are, in general, undecidable, specifying such predicates appears to be meaningless. Instead, the functional languages community developed type systems that statically approximate interesting predicates.

In this paper, we show how to support higher-order function contracts in a theoretically well-founded and practically viable manner. Specifically, we introduce λ^{CON}, a typed lambda calculus with assertions for higher-order functions. The calculus models the assertion monitoring system that we employ in DrScheme. We establish basic properties of the model (type soundness, etc.) and illustrate the usefulness of contract checking with examples from DrScheme's code base.

We believe that the development of an assertion system for higher-order functions serves two purposes. On one hand, the system has strong practical potential because existing type systems simply cannot express many assertions that programmers would like to state. On the other hand, an inspection of a large base of invariants may provide inspiration for the direction of practical future type system research.

Categories & Subject Descriptors: D.3.3, D.2.1; **General Terms:** Design, Languages, Reliability; **Keywords:** Contracts, Higher-order Functions, Behavioral Specifications, Predicate Typing, Software Reliability

[1] Work partly conducted at Rice University, Houston TX. Address as of 9/2002: University of Chicago; 1100 E 58th Street; Chicago, IL 60637

This is a technical report version of a paper that appeared in ICFP in 2002 [6]. This version includes everything that the conference version does, but also includes the complete proofs in an appendix.

1 Introduction

Dynamically enforced pre- and post-condition contracts have been widely used in procedural and object-oriented languages [11, 14, 17, 20, 21, 22, 25, 31]. As Rosenblum [27] has shown, for example, these contracts have great practical value in improving the robustness of systems in procedural languages. Eiffel [22] even developed an entire philosophy of system design based on contracts ("Design by Contract"). Although Java [12] does not support contracts, it is one of the most requested extensions.[1]

With one exception, higher-order languages have mostly ignored assertion-style contracts. The exception is Bigloo Scheme [28], where programmers can write down first-order, type-like constraints on procedures. These constraints are used to generate more efficient code when the compiler can prove they are correct and are turned into runtime checks when the compiler cannot prove them correct.

First-order procedural contracts have a simple interpretation. Consider this contract, written in an ML-like syntax:

f : **int**[$>$ 9] $\rightarrow$**int**[0,99]
val rec $f = \lambda\, x. \cdots$

It states that the argument to f must be an **int** greater than 9 and that f produces an **int** between 0 and 99. To enforce this contract, a contract compiler inserts code to check that x is in the proper range when f is called and that f's result is in the proper range when f returns. If x is not in the proper range, f's caller is blamed for a contractual violation. Symmetrically, if f's result is not in the proper range, the blame falls on f itself. In this world, detecting contractual violations and assigning blame merely means checking appropriate predicates at well-defined points in the program's evaluation.

This simple mechanism for checking contracts does not generalize to languages with higher-order functions. Consider this contract:

g : (**int**[$>$ 9] $\rightarrow$**int**[0,99]) $\rightarrow$**int**[0,99]
val rec $g = \lambda$ *proc*. $\cdots$

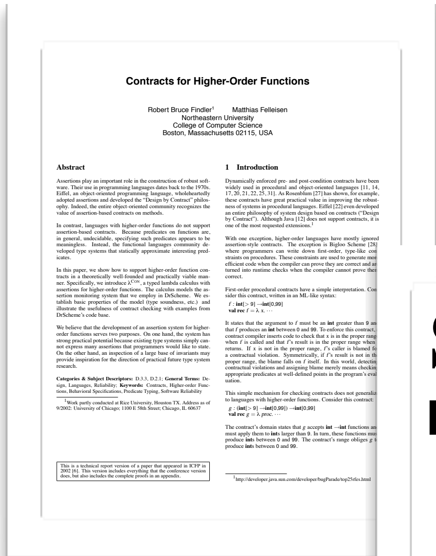
The contract's domain states that g accepts **int** $\rightarrow$**int** functions and must apply them to **int**s larger than 9. In turn, these functions must produce **int**s between 0 and 99. The contract's range obliges g to produce **int**s between 0 and 99.

[1] http://developer.java.sun.com/developer/bugParade/top25rfes.html

[OOPSLA'12]


Contracts for Higher-Order Functions
Robert Bruce Findler
Matthias Felleisen
Northeastern University
College of Computer Science
Boston, Massachusetts 02115, USA
Abstract
1 Introduction

[OOPSLA'12]


Contracts for Higher-Order Functions

Robert Bruce Findler   Matthias Felleisen
Northeastern University
College of Computer Science
Boston, Massachusetts 02115, USA

Abstract

1 Introduction

Semantics for Symbolic PCF with Contracts $\quad E \longmapsto E'$

$$\mathsf{if}\ V\ E_1\ E_2 \longmapsto E_1\ \mathsf{if}\ \delta(\mathsf{false?}, V) \ni \mathsf{ff}$$

$$\mathsf{if}\ V\ E_1\ E_2 \longmapsto E_2\ \mathsf{if}\ \delta(\mathsf{false?}, V) \ni \mathsf{tt}$$

$$(\lambda X\!:\!T.E)\ V \longmapsto [V/X]E$$

$$\mu X\!:\!T.E \longmapsto [\mu X\!:\!T.E/X]E$$

$$O(\vec{V}) \longmapsto A\ \mathsf{if}\ \delta(O, \vec{V}) \ni A$$

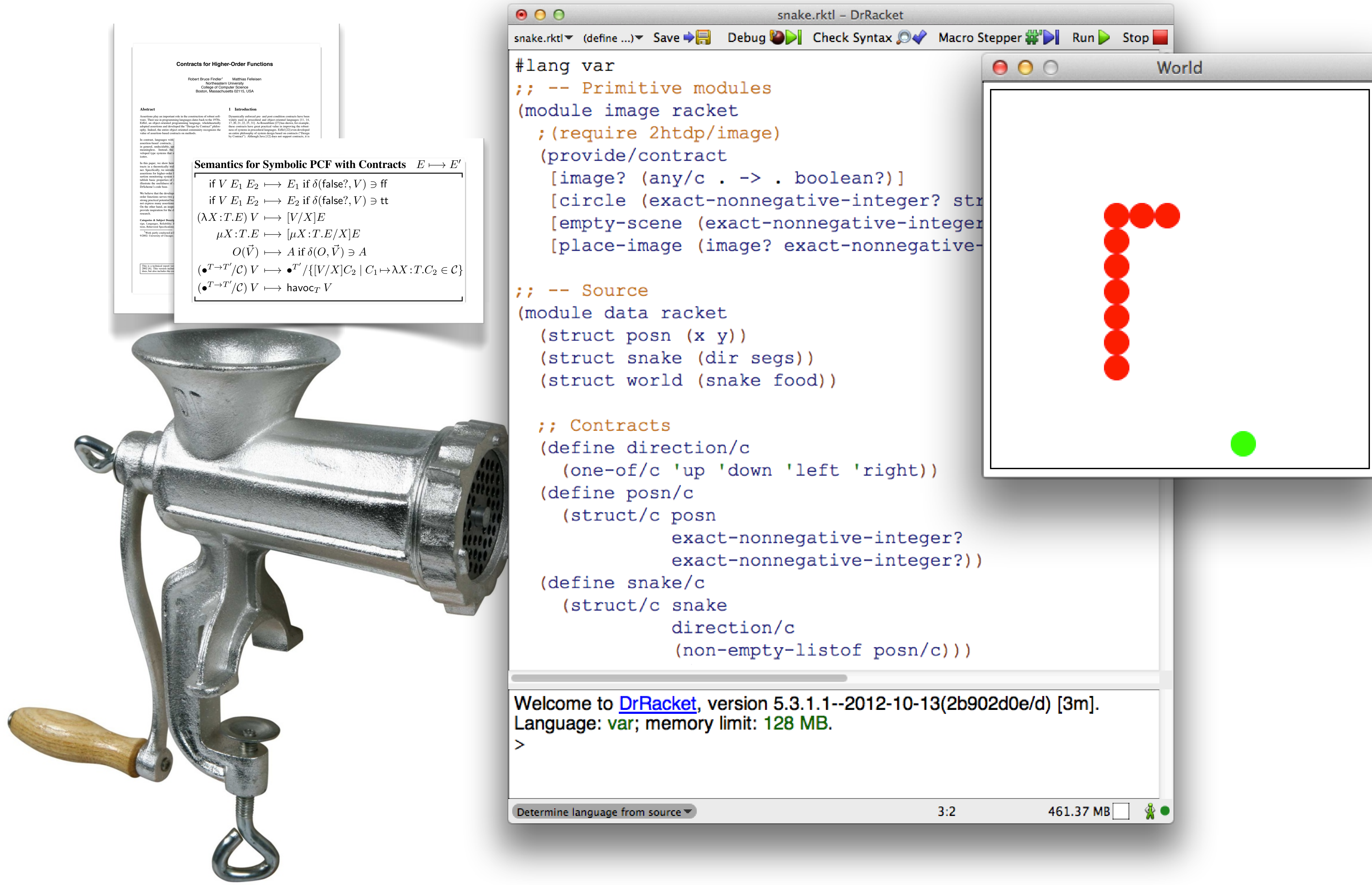
$$(\bullet^{T\to T'}/\mathcal{C})\ V \longmapsto \bullet^{T'}/\{[V/X]C_2 \mid C_1 \mapsto \lambda X\!:\!T.C_2 \in \mathcal{C}\}$$

$$(\bullet^{T\to T'}/\mathcal{C})\ V \longmapsto \mathsf{havoc}_T\ V$$

[OOPSLA'12]


Contracts for Higher-Order Functions
Robert Bruce Findler Matthias Felleisen
Northeastern University
College of Computer Science
Boston, Massachusetts 02115, USA
Abstract
1 Introduction
Semantics for Symbolic PCF with Contracts $E \longmapsto E'$
$\mathsf{if}\ V\ E_1\ E_2 \longmapsto E_1\ \mathsf{if}\ \delta(\mathsf{false?}, V) \ni \mathsf{ff}$
$\mathsf{if}\ V\ E_1\ E_2 \longmapsto E_2\ \mathsf{if}\ \delta(\mathsf{false?}, V) \ni \mathsf{tt}$
$(\lambda X\!:\!T.E)\ V \longmapsto [V/X]E$
$\mu X\!:\!T.E \longmapsto [\mu X\!:\!T.E/X]E$
$O(\vec{V}) \longmapsto A\ \mathsf{if}\ \delta(O, \vec{V}) \ni A$
$(\bullet^{T \to T'}/\mathcal{C})\ V \longmapsto \bullet^{T'}/\{[V/X]C_2 \mid C_1 \mapsto \lambda X\!:\!T.C_2 \in \mathcal{C}\}$
$(\bullet^{T \to T'}/\mathcal{C})\ V \longmapsto \mathsf{havoc}_T\ V$

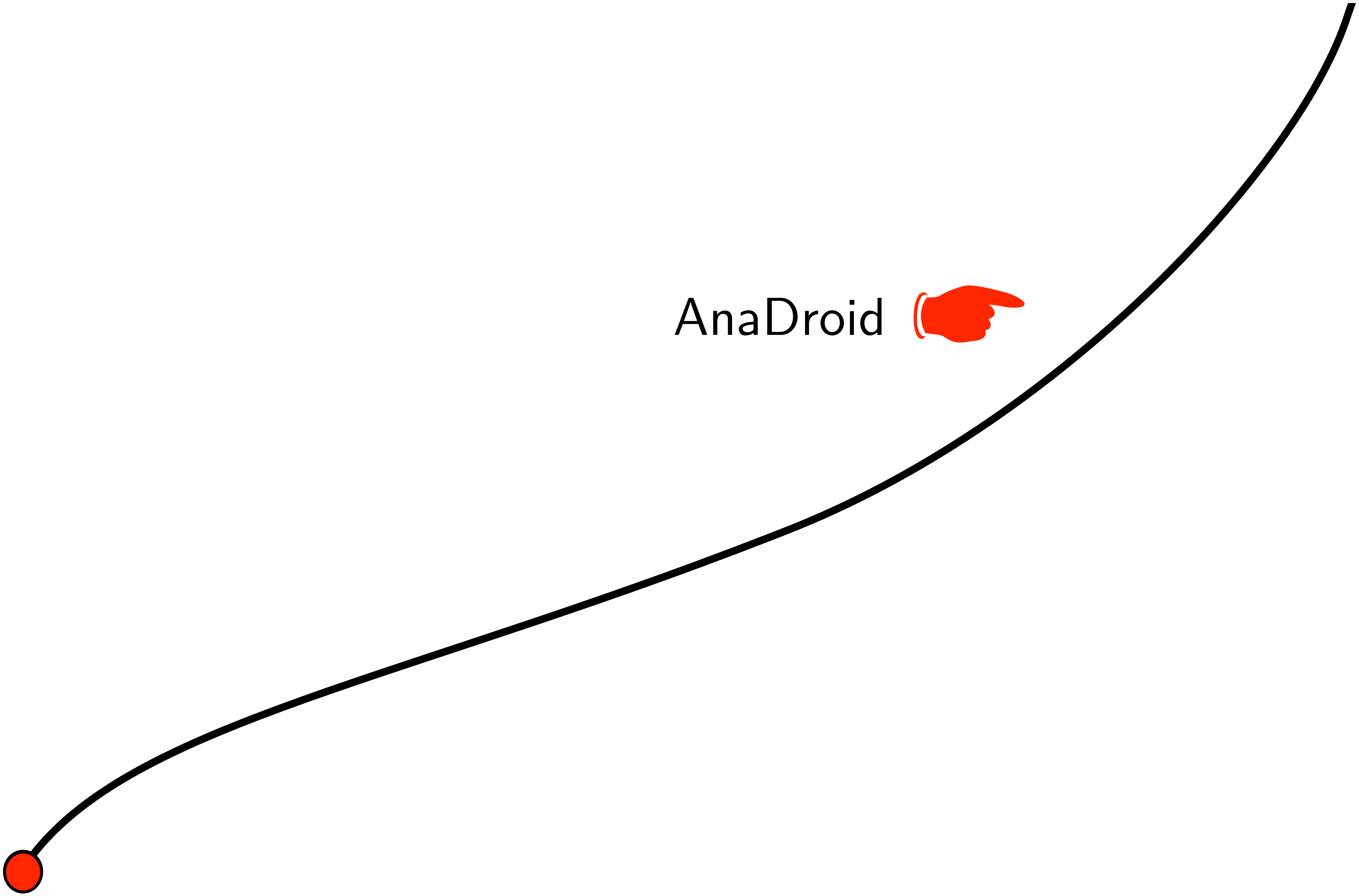
[OOPSLA'12]

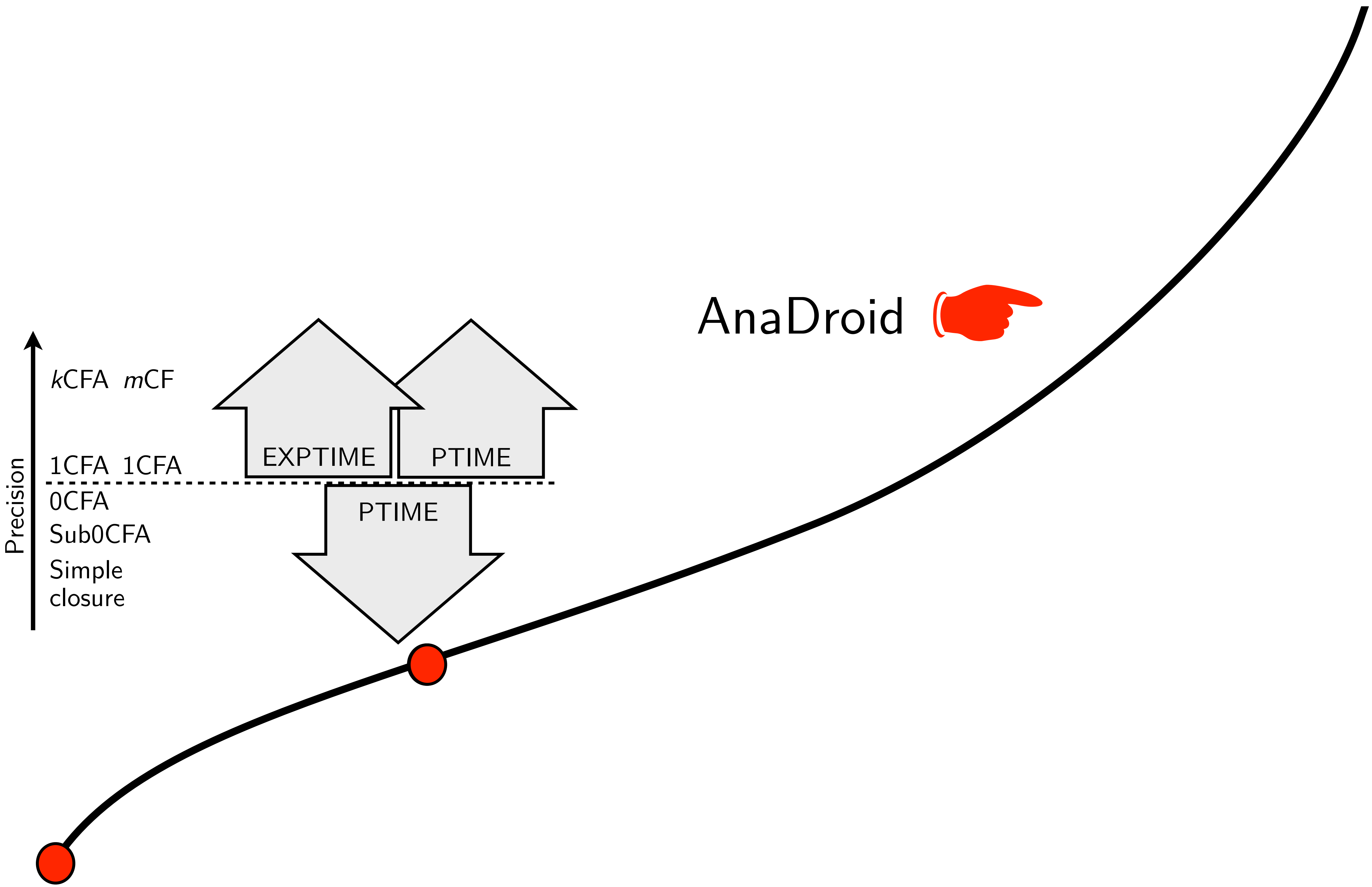

Contracts for Higher-Order Functions
Semantics for Symbolic PCF with Contracts E ⟼ E′
snake.rktl – DrRacket
snake.rktl (define ...) Save Debug Check Syntax Macro Stepper Run Stop
#lang var
;; -- Primitive modules
(module image racket
;(require 2htdp/image)
(provide/contract
[image? (any/c . -> . boolean?)]
[circle (exact-nonnegative-integer? str
[empty-scene (exact-nonnegative-integer
[place-image (image? exact-nonnegative-
;; -- Source
(module data racket
(struct posn (x y))
(struct snake (dir segs))
(struct world (snake food))
;; Contracts
(define direction/c
(one-of/c 'up 'down 'left 'right))
(define posn/c
(struct/c posn
exact-nonnegative-integer?
exact-nonnegative-integer?))
(define snake/c
(struct/c snake
direction/c
(non-empty-listof posn/c)))
Welcome to DrRacket, version 5.3.1.1--2012-10-13(2b902d0e/d) [3m].
Language: var; memory limit: 128 MB.
>
Determine language from source
3:2
461.37 MB
World

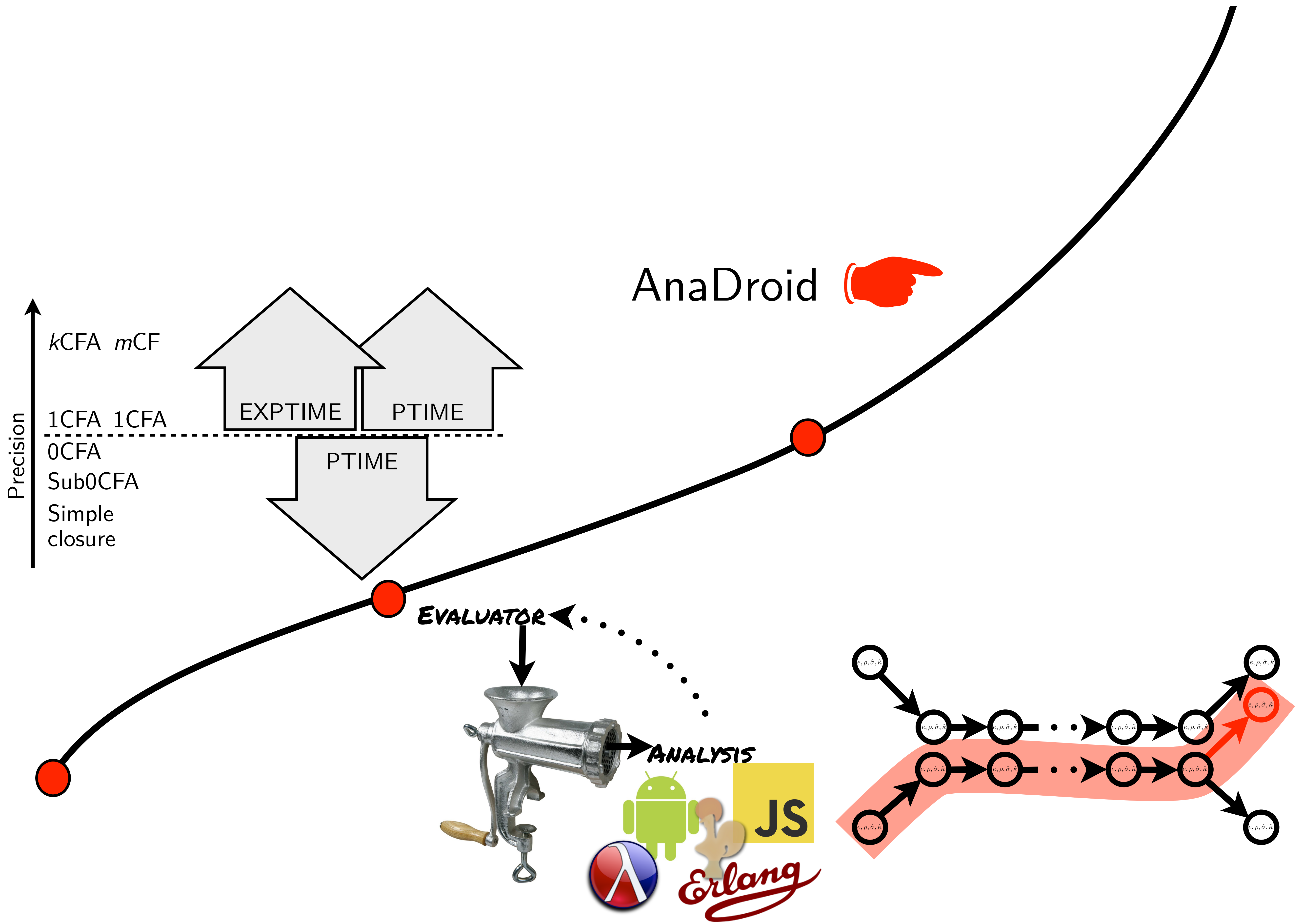
[OOPSLA'12]

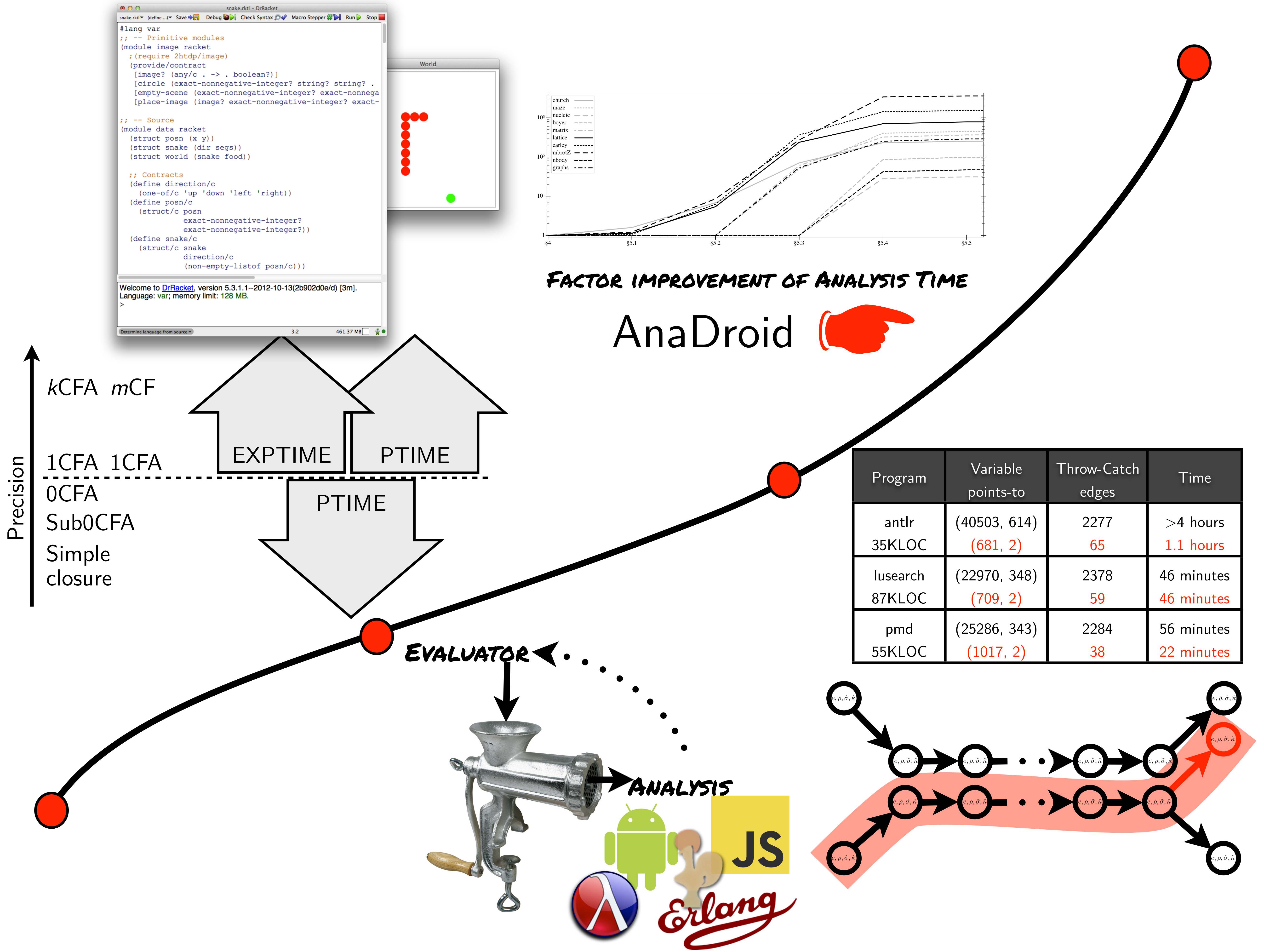
Conclusion & Perspective

To trust software, we must predict its (mis)behavior.


AnaDroid

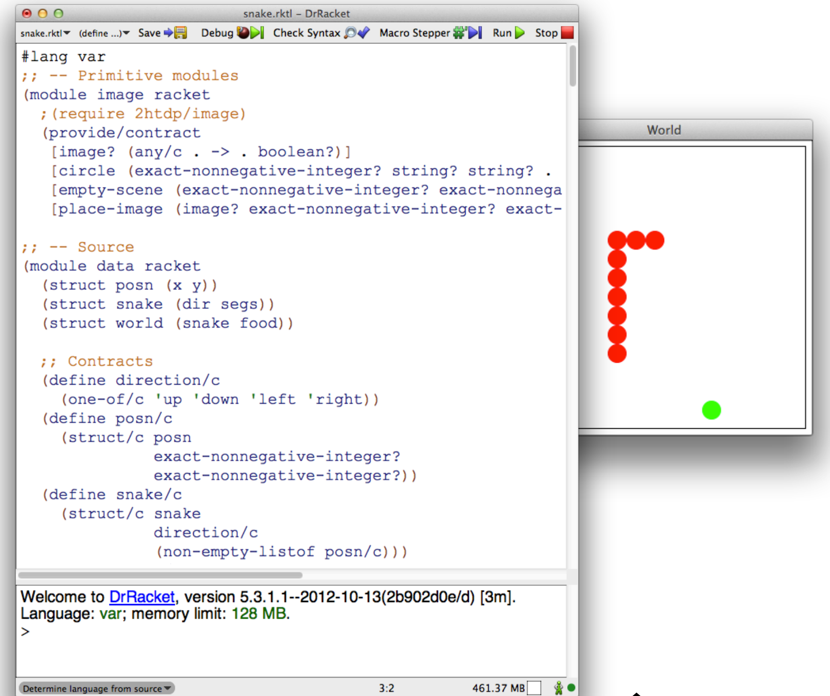

AnaDroid
Precision
kCFA mCF
1CFA 1CFA
0CFA
Sub0CFA
Simple
closure
EXPTIME
PTIME
PTIME

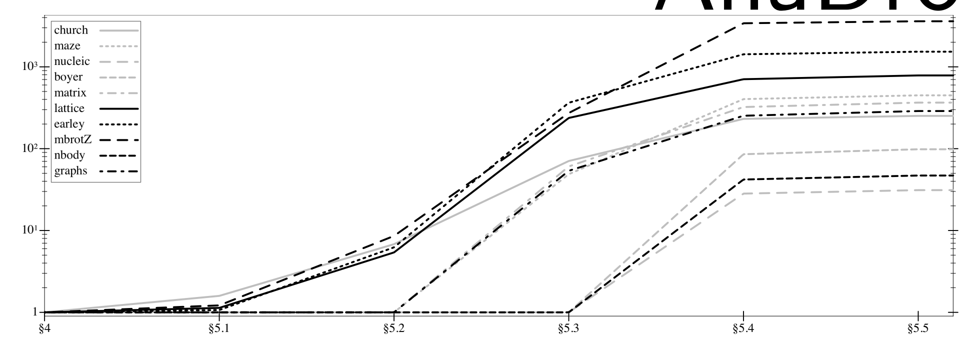

AnaDroid
Precision
kCFA mCF
1CFA 1CFA
0CFA
Sub0CFA
Simple
closure
EXPTIME
PTIME
PTIME
Evaluator
Analysis
JS
Erlang



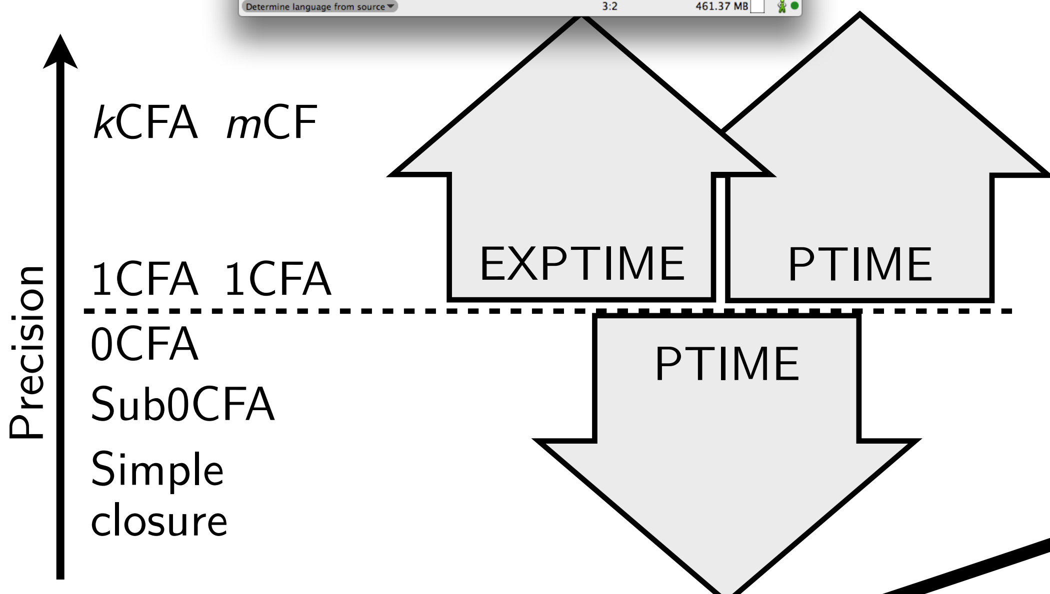
Program	Variable points-to	Throw-Catch edges	Time
antlr 35KLOC	(40503, 614) (681, 2)	2277 65	>4 hours 1.1 hours
lusearch 87KLOC	(22970, 348) (709, 2)	2378 59	46 minutes 46 minutes
pmd 55KLOC	(25286, 343) (1017, 2)	2284 38	56 minutes 22 minutes

AnaDroid

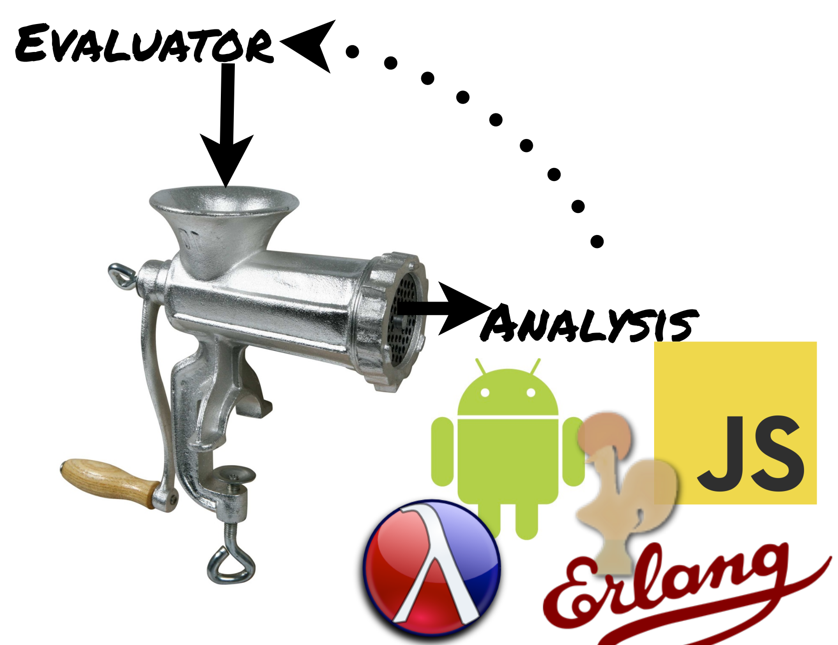


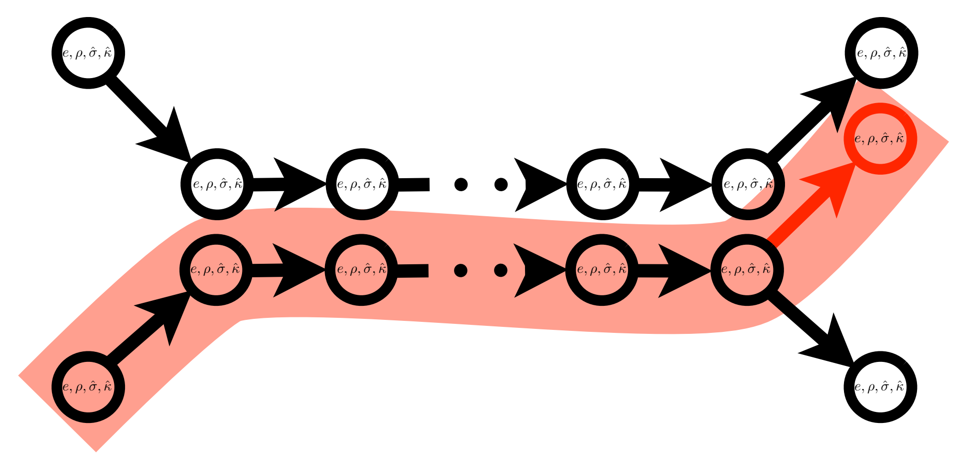


FACTOR IMPROVEMENT OF ANALYSIS TIME

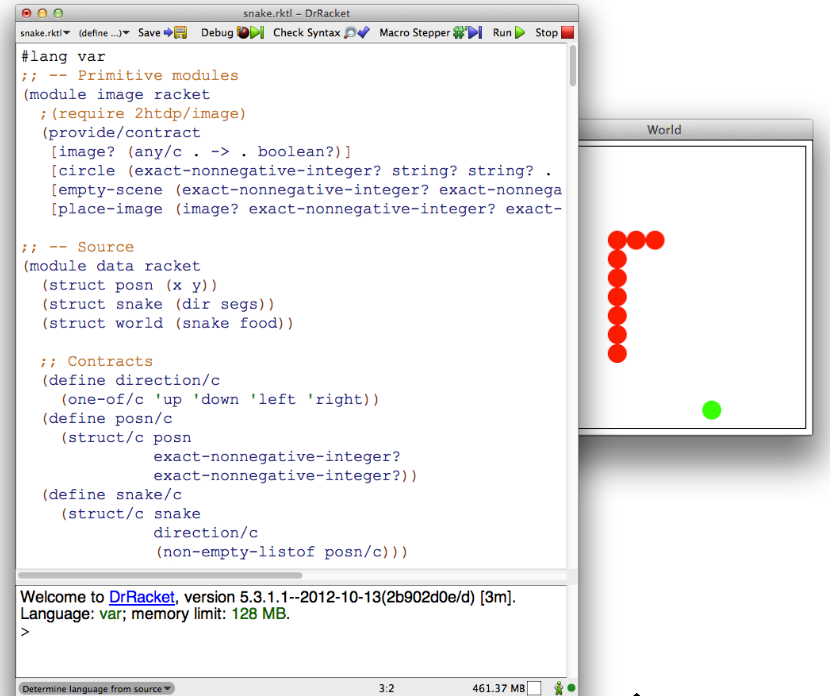


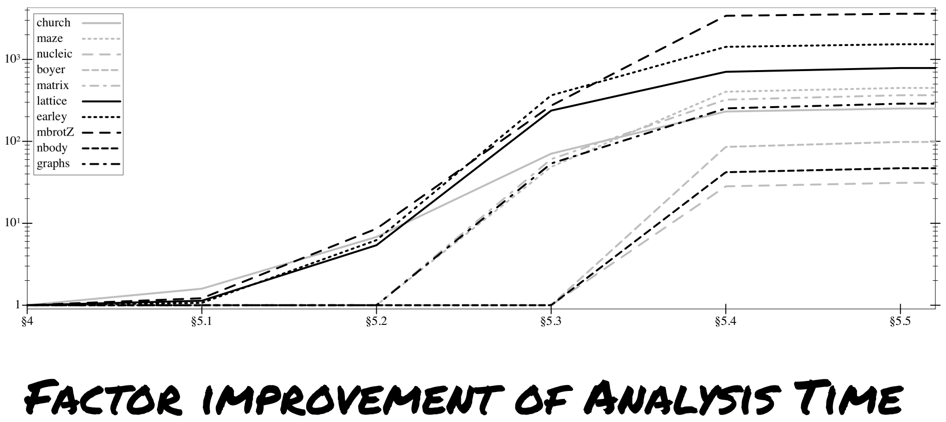
Program	Variable points-to	Throw-Catch edges	Time
antlr 35KLOC	(40503, 614) (681, 2)	2277 65	>4 hours 1.1 hours
lusearch 87KLOC	(22970, 348) (709, 2)	2378 59	46 minutes 46 minutes
pmd 55KLOC	(25286, 343) (1017, 2)	2284 38	56 minutes 22 minutes

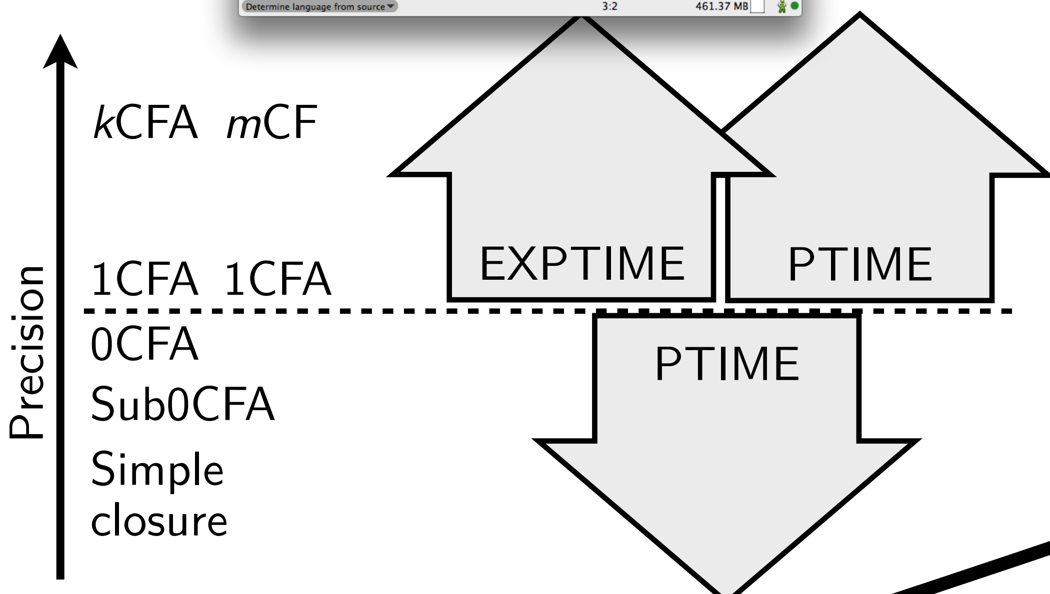




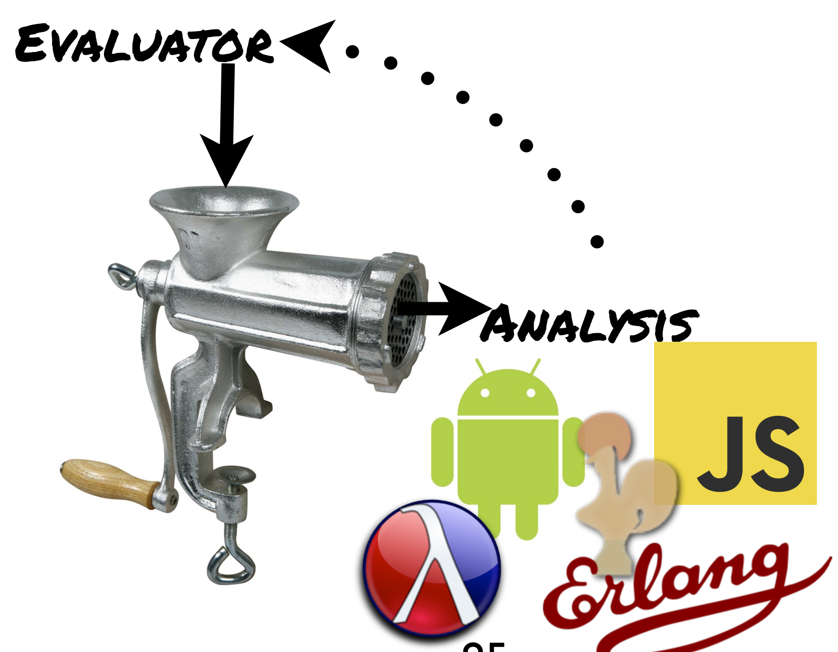
AnaDroid

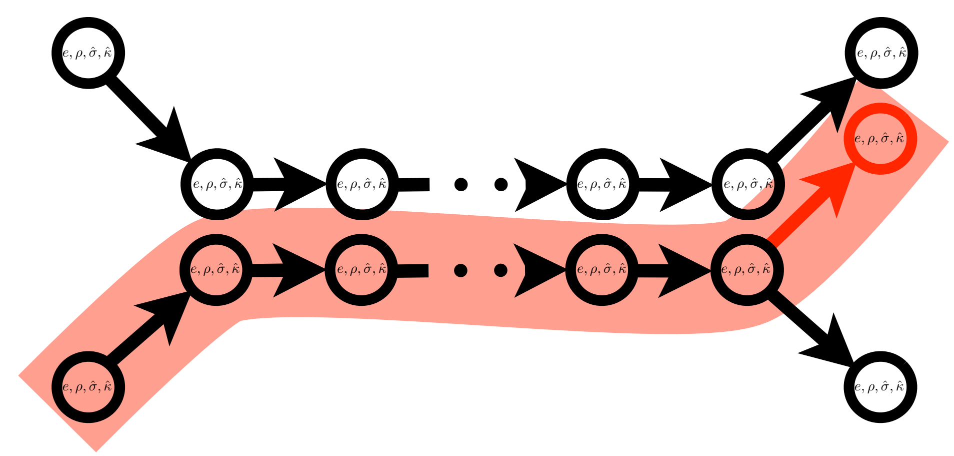




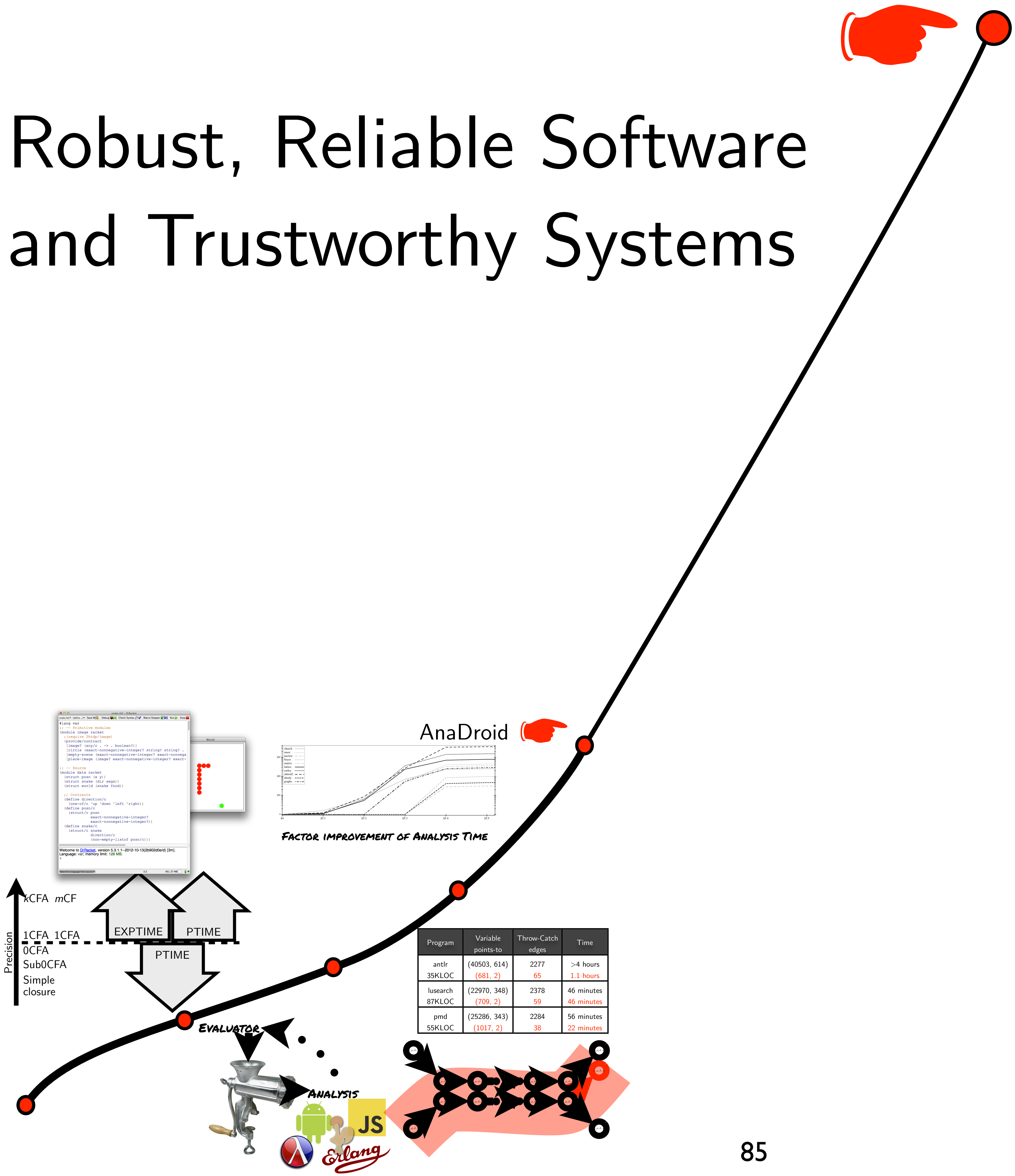


Program	Variable points-to	Throw-Catch edges	Time
antlr 35KLOC	(40503, 614) (681, 2)	2277 65	>4 hours 1.1 hours
lusearch 87KLOC	(22970, 348) (709, 2)	2378 59	46 minutes 46 minutes
pmd 55KLOC	(25286, 343) (1017, 2)	2284 38	56 minutes 22 minutes

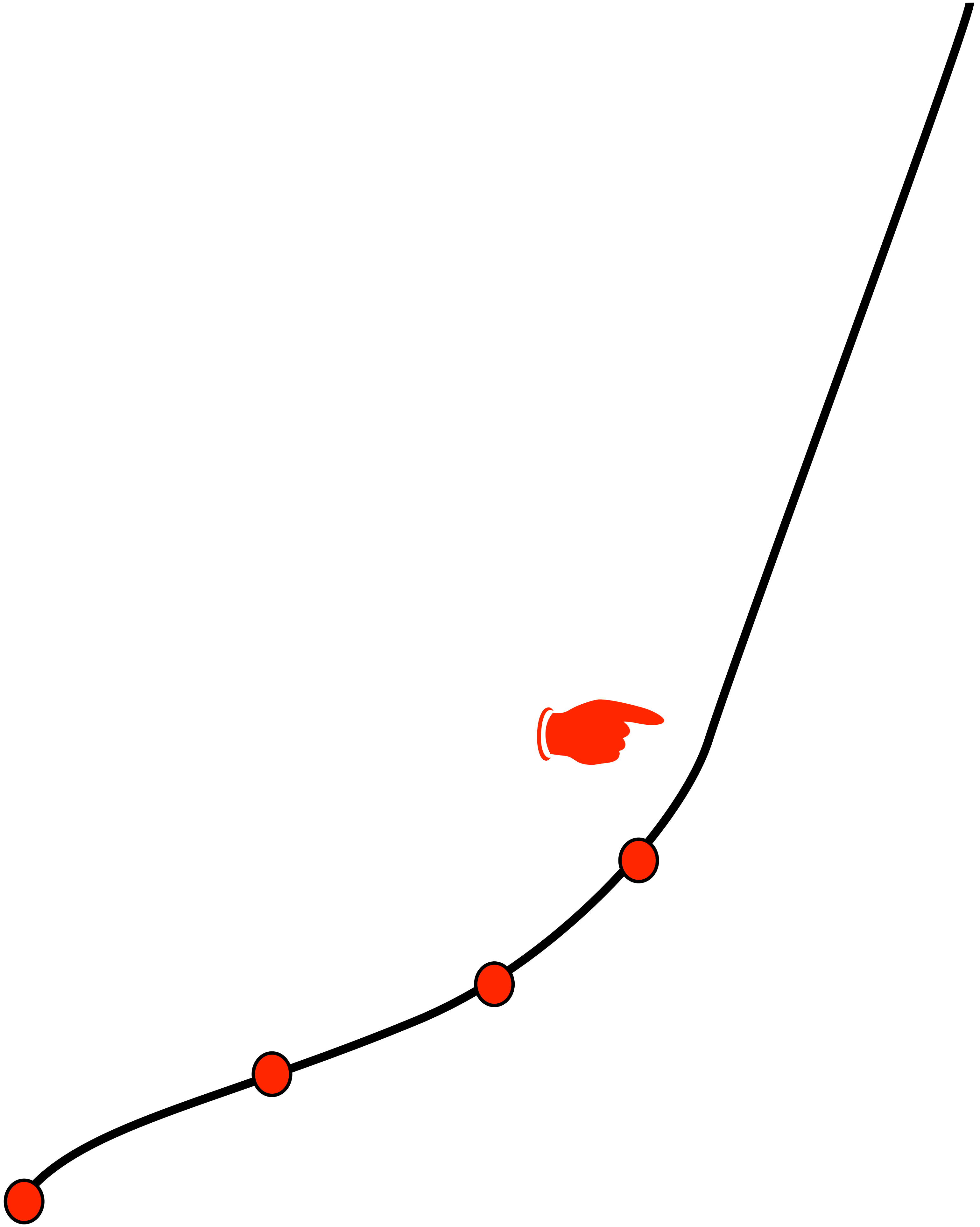


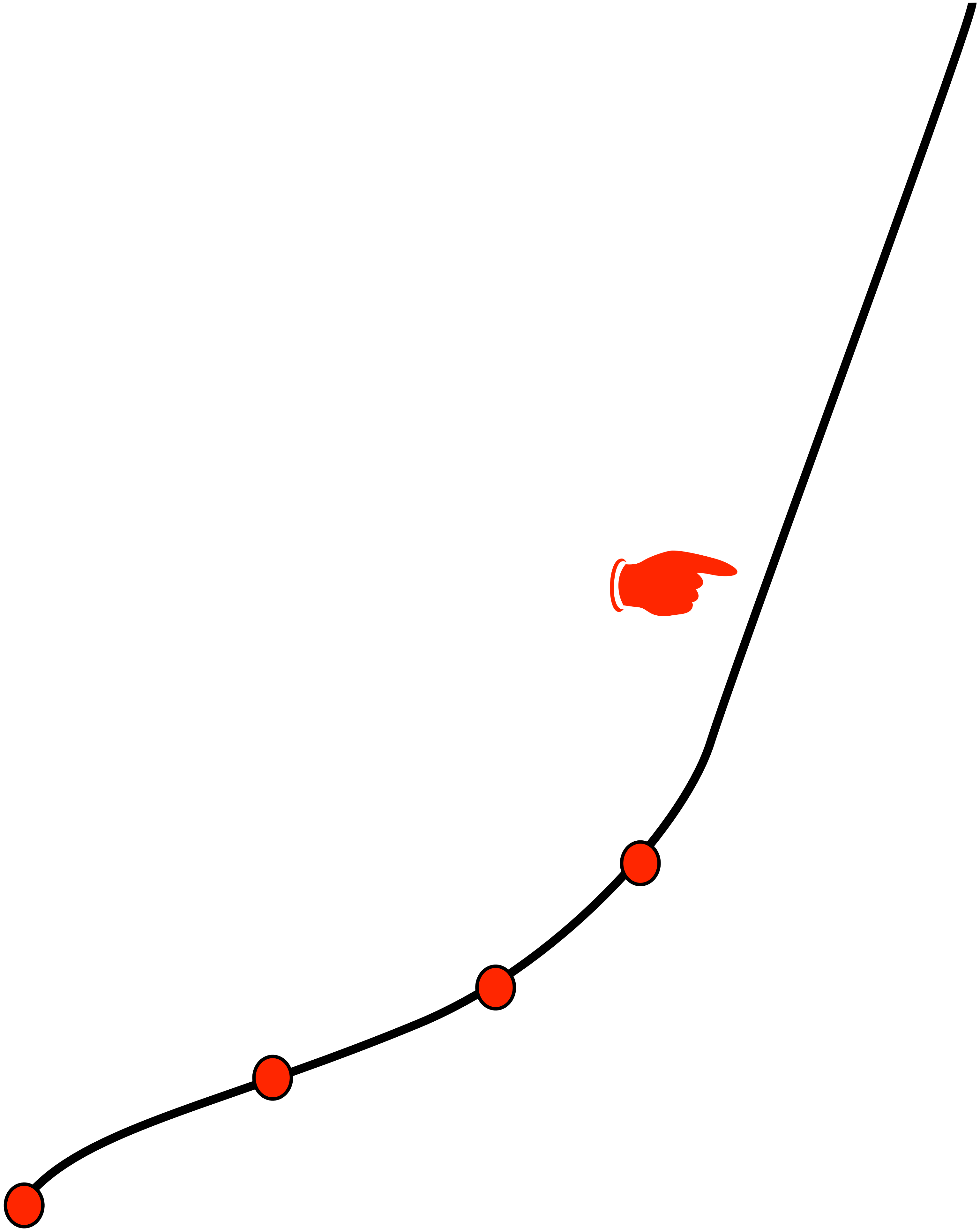


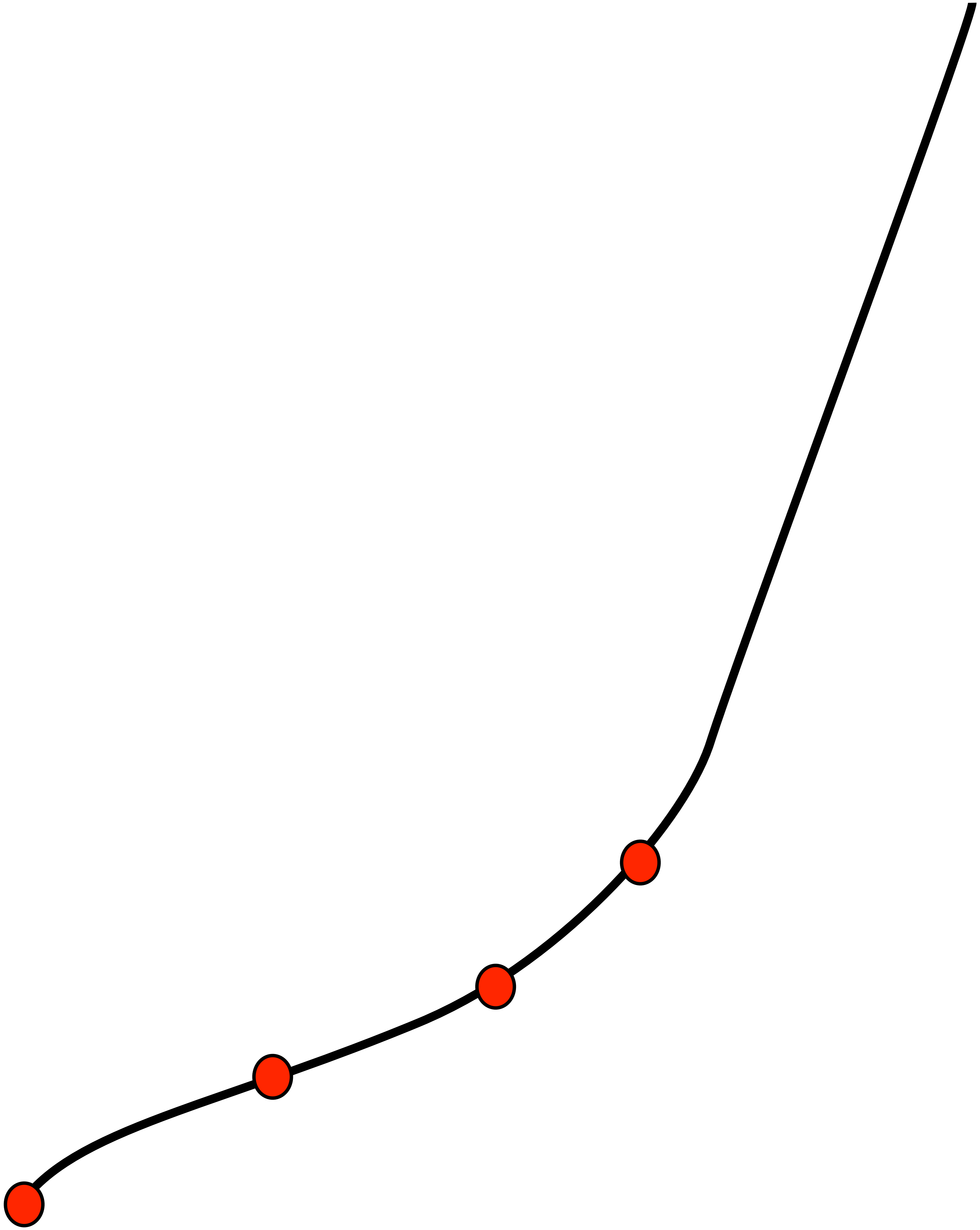
Robust, Reliable Software and Trustworthy Systems

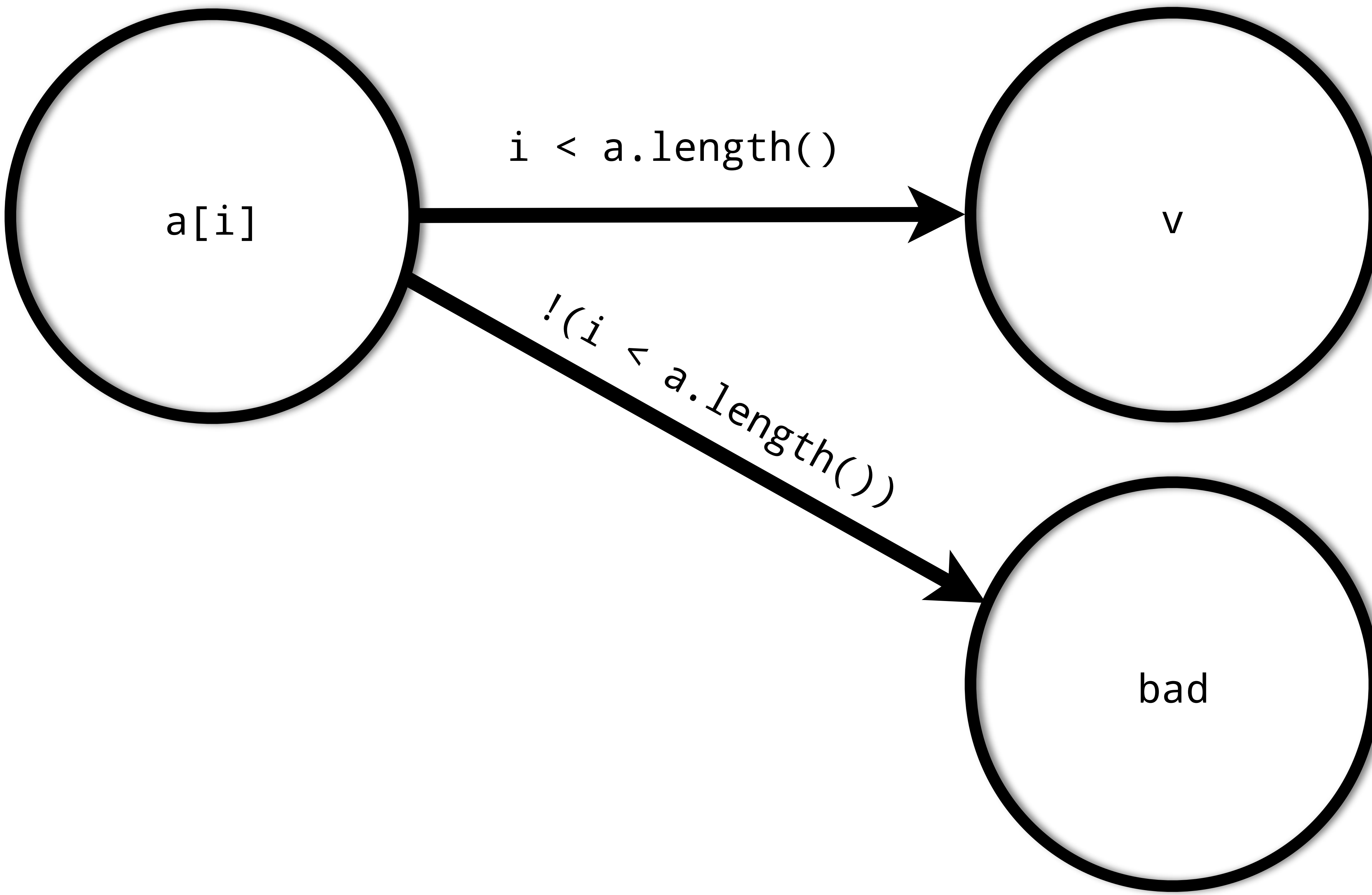


Program	Variable points-to	Throw-Catch edges	Time
antlr 35KLOC	(40503, 614) (681, 2)	2277 65	>4 hours 1.1 hours
lusearch 87KLOC	(22970, 348) (709, 2)	2378 59	46 minutes 46 minutes
pmd 55KLOC	(25286, 343) (1017, 2)	2284 38	56 minutes 22 minutes

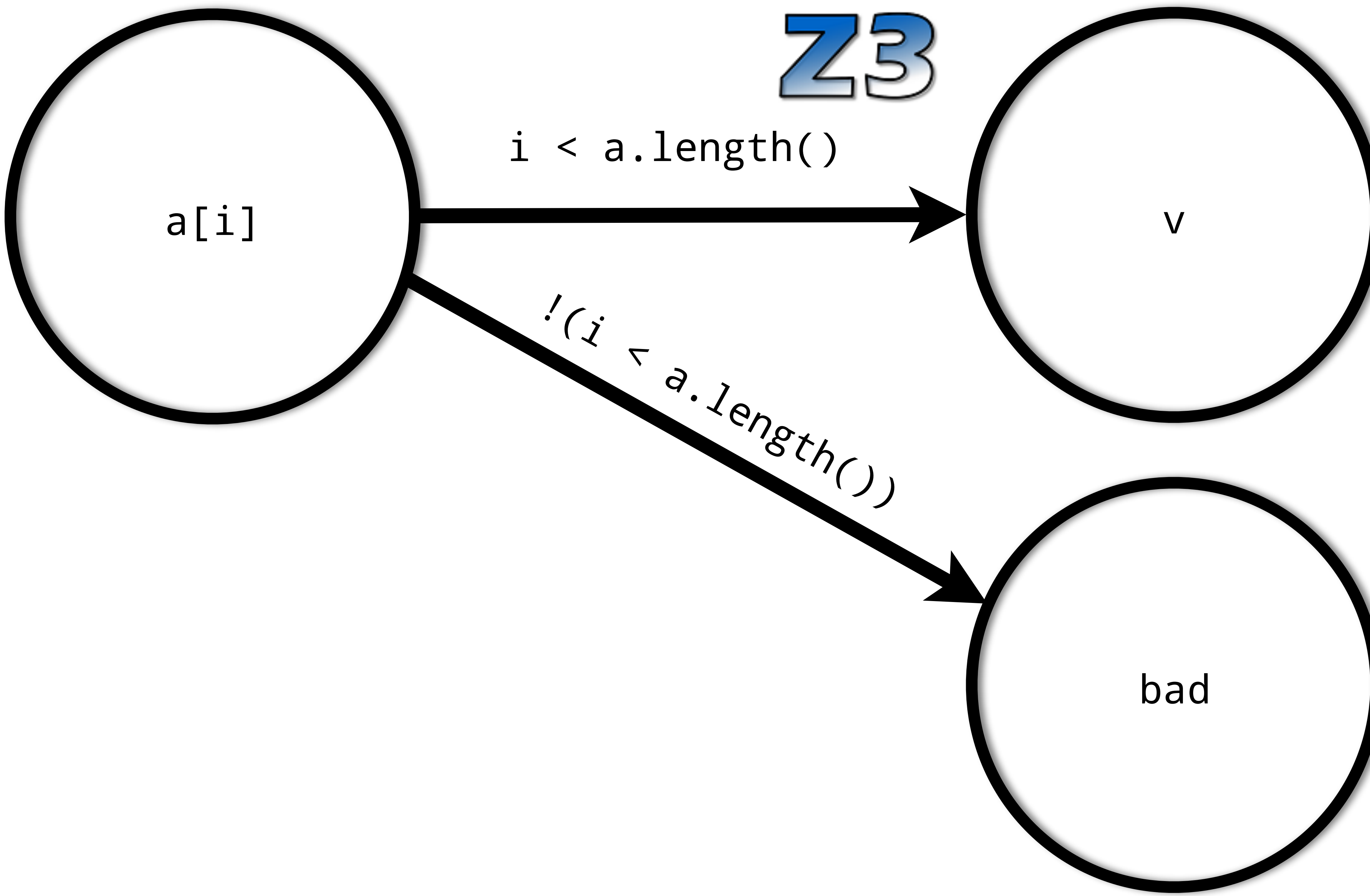


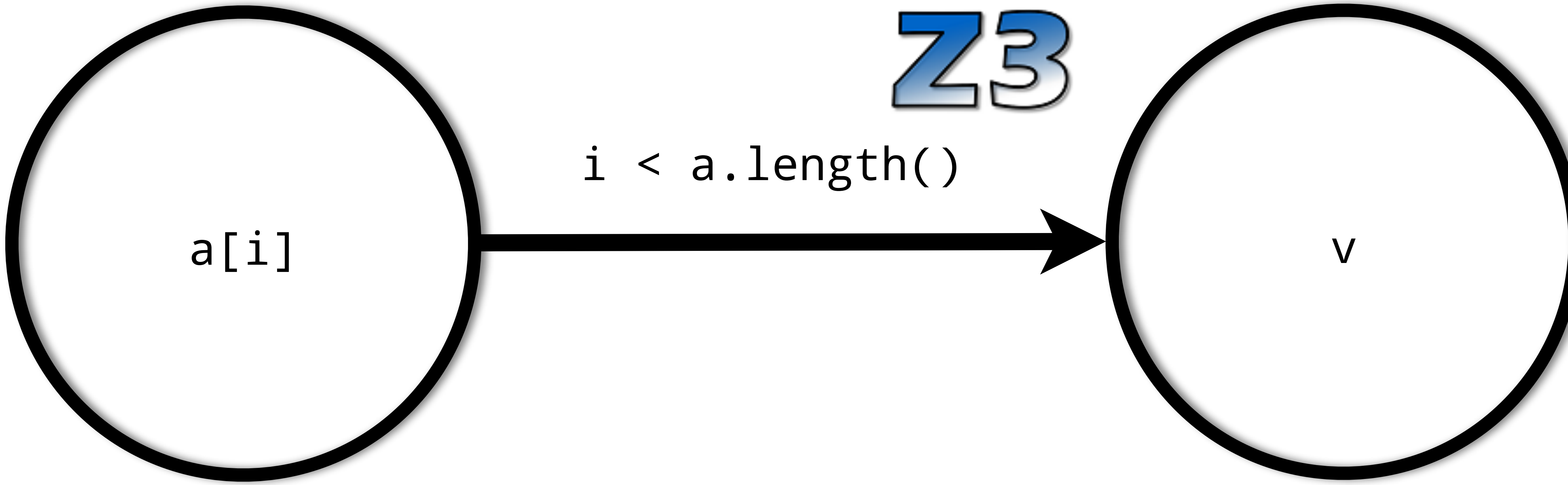


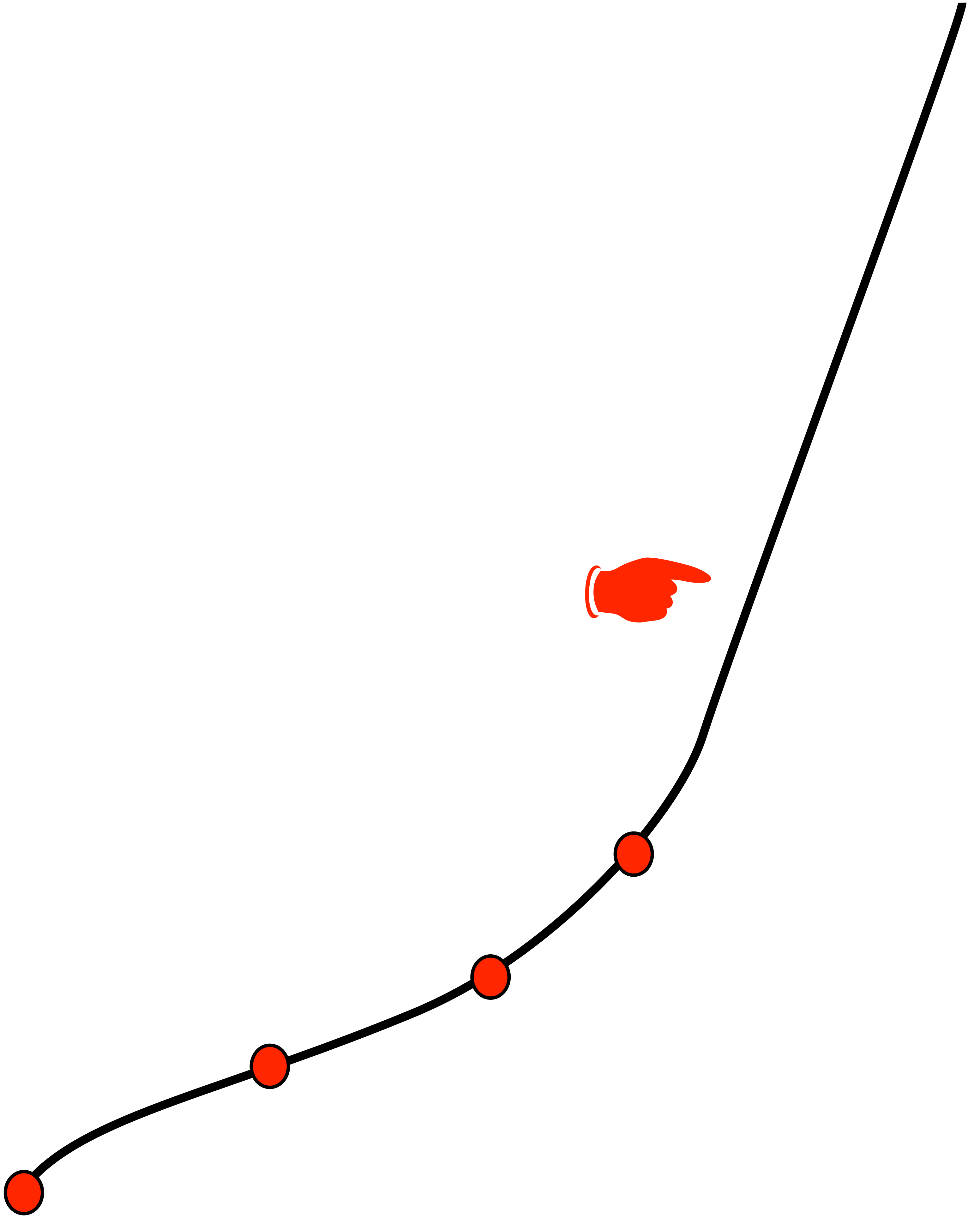


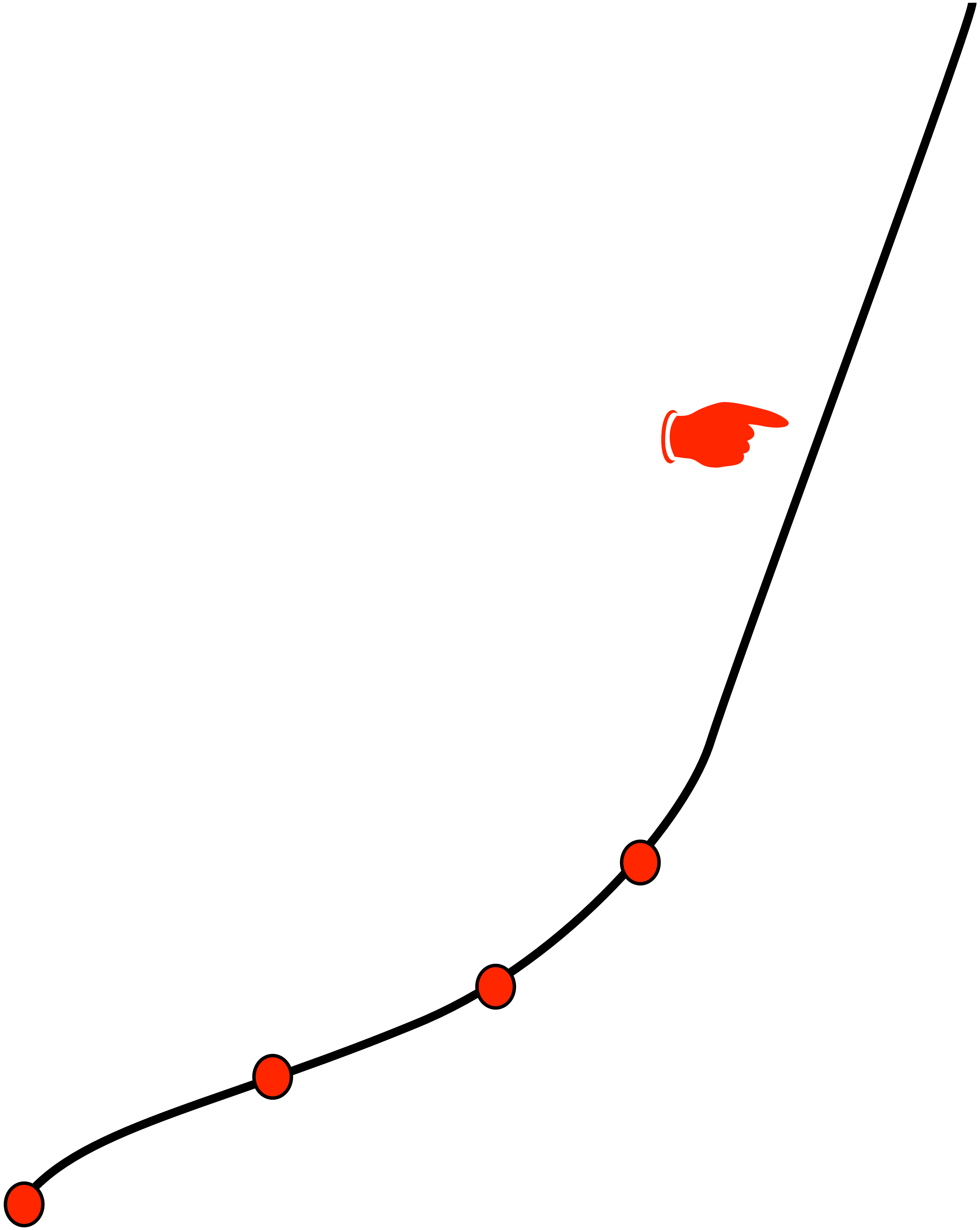


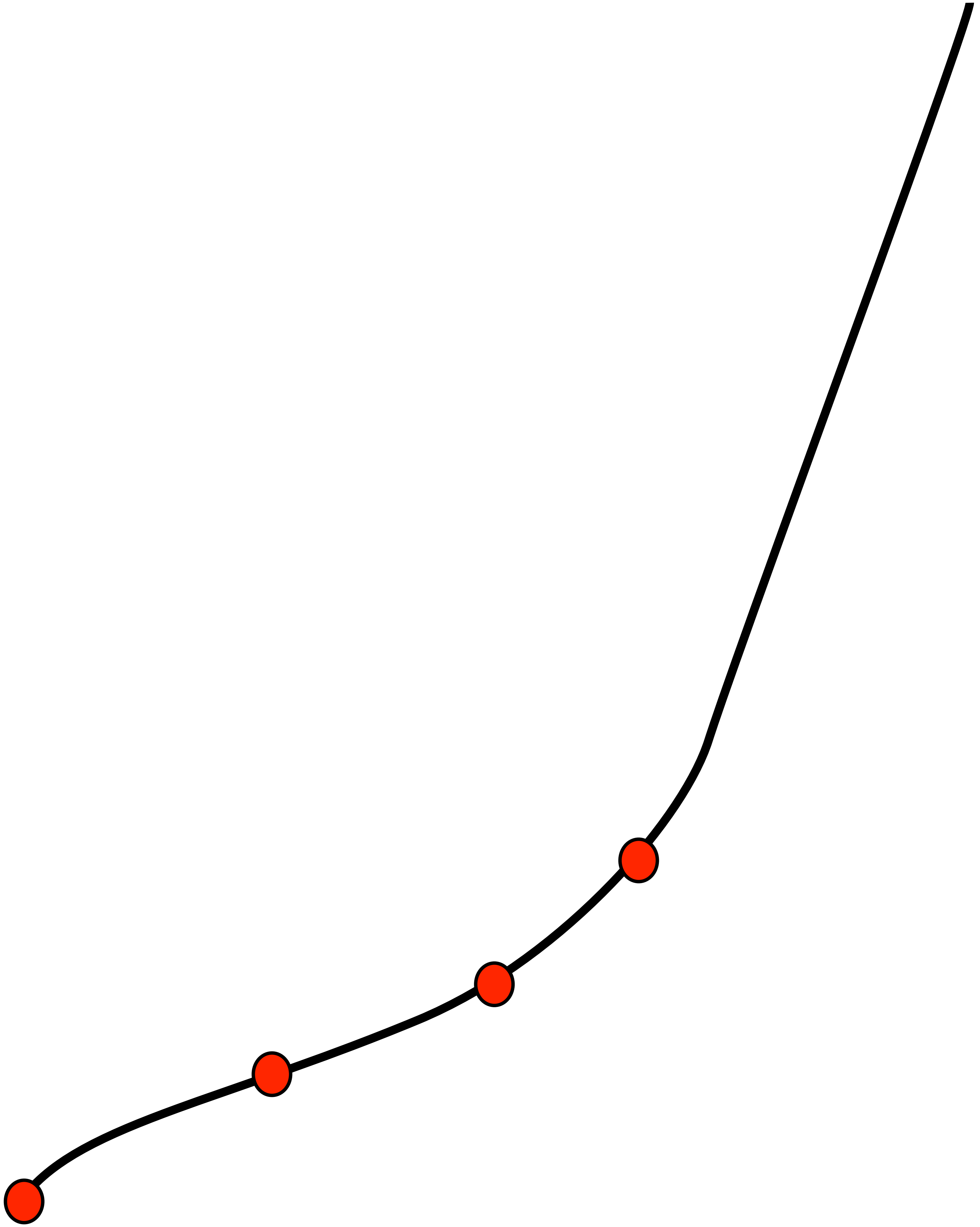
a[i]
i < a.length()
v
!(i < a.length())
bad

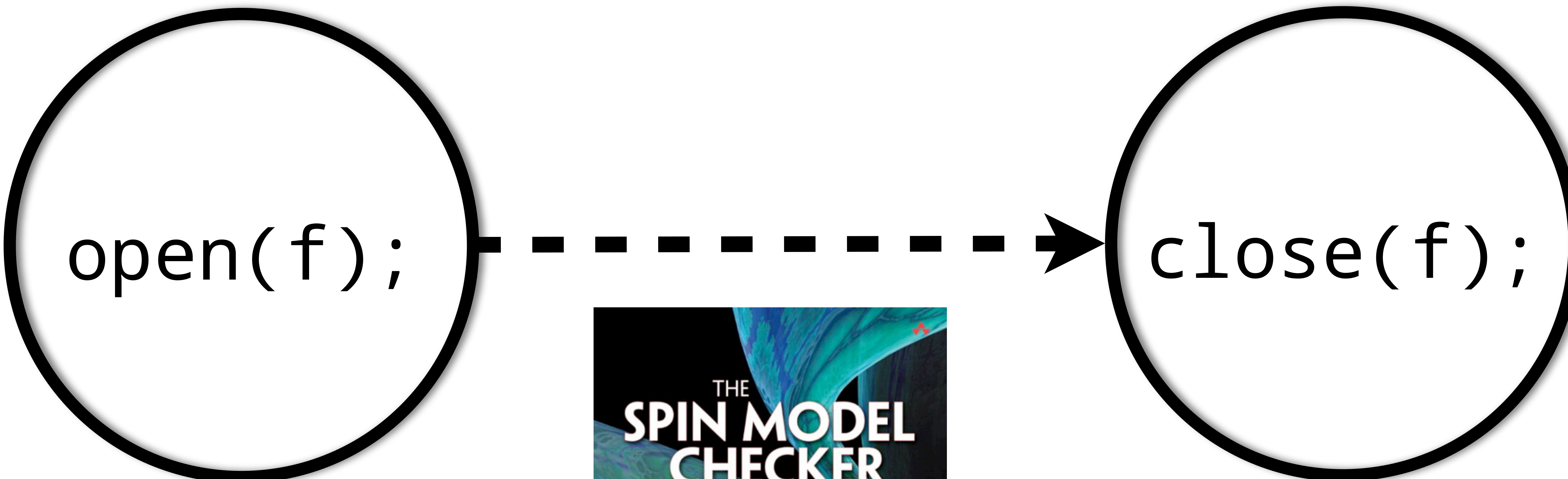

Z3
a[i]
i < a.length()
v
!(i < a.length())
bad

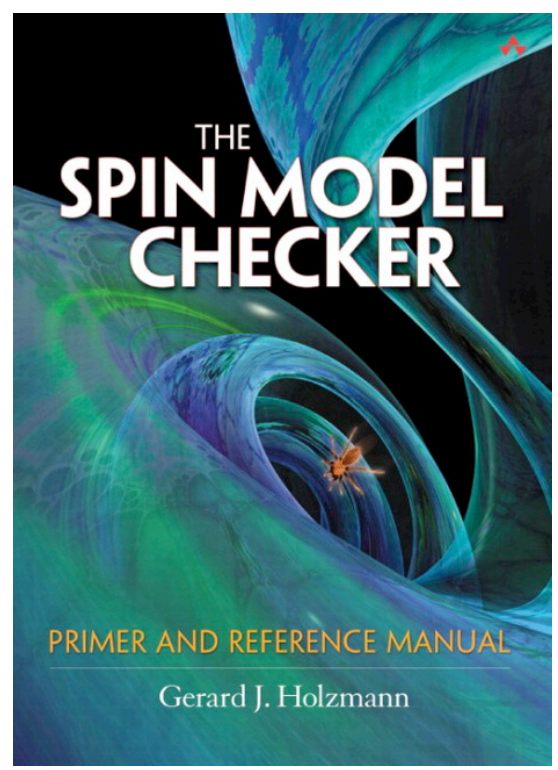

Z3
i < a.length()
a[i]
v

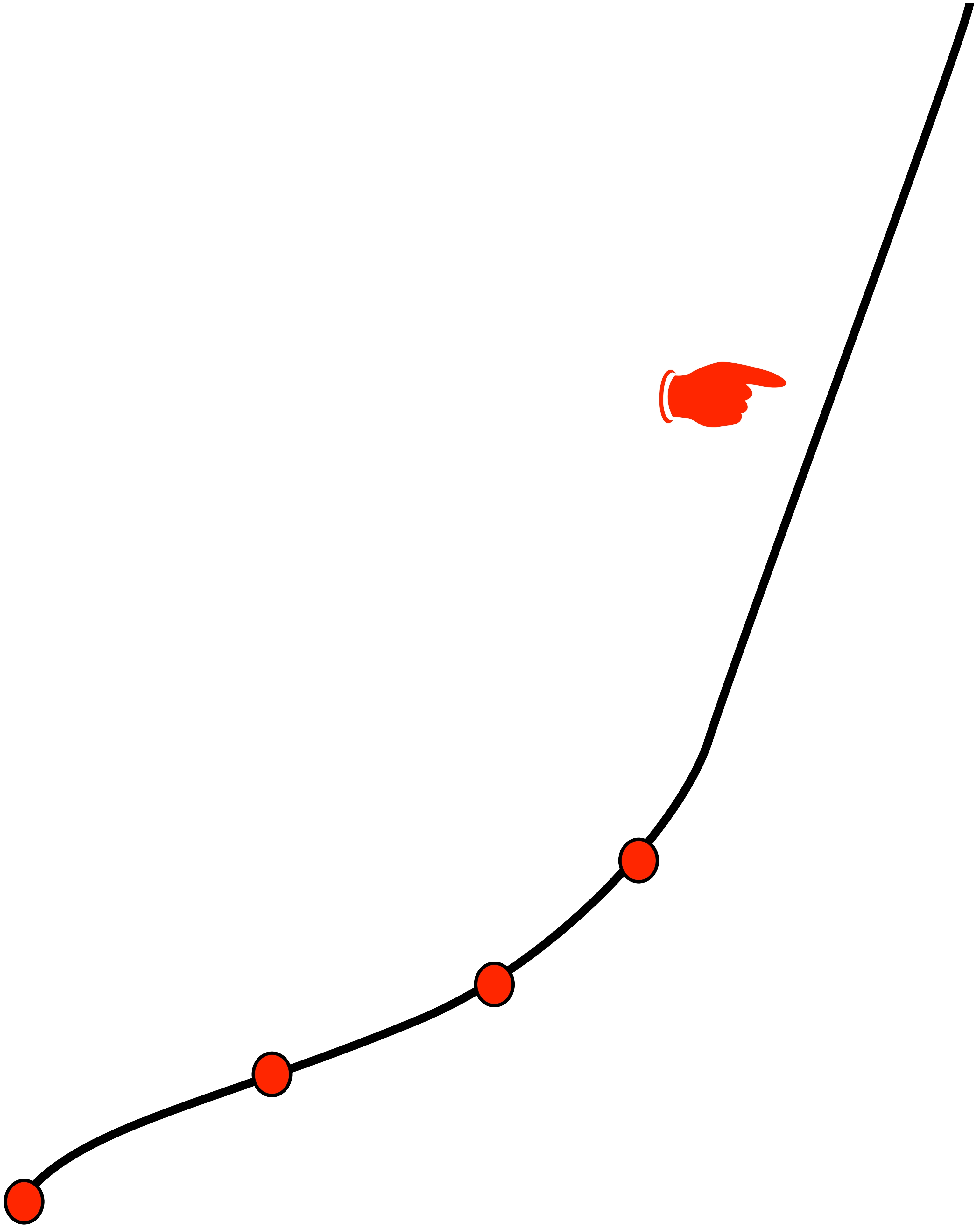


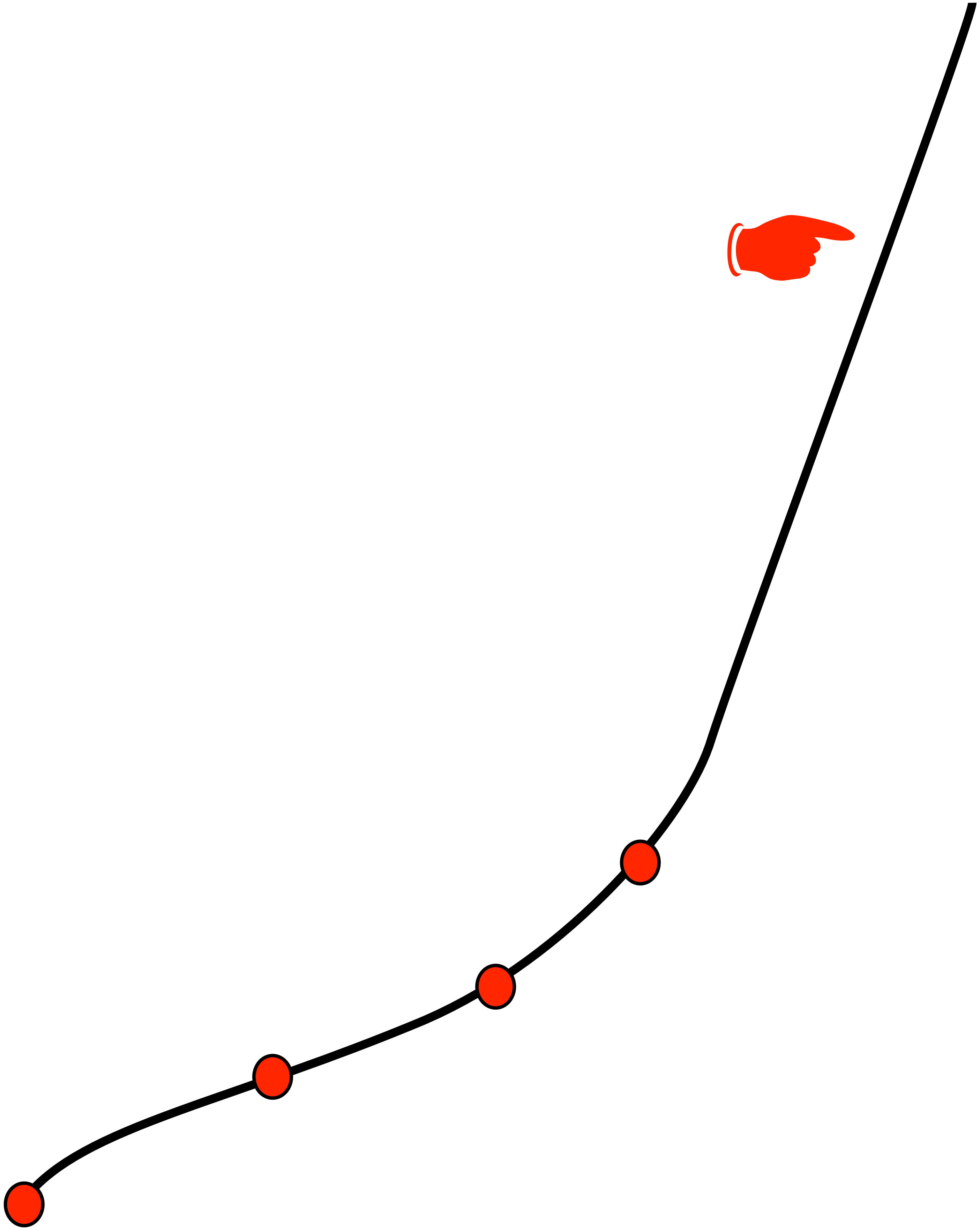


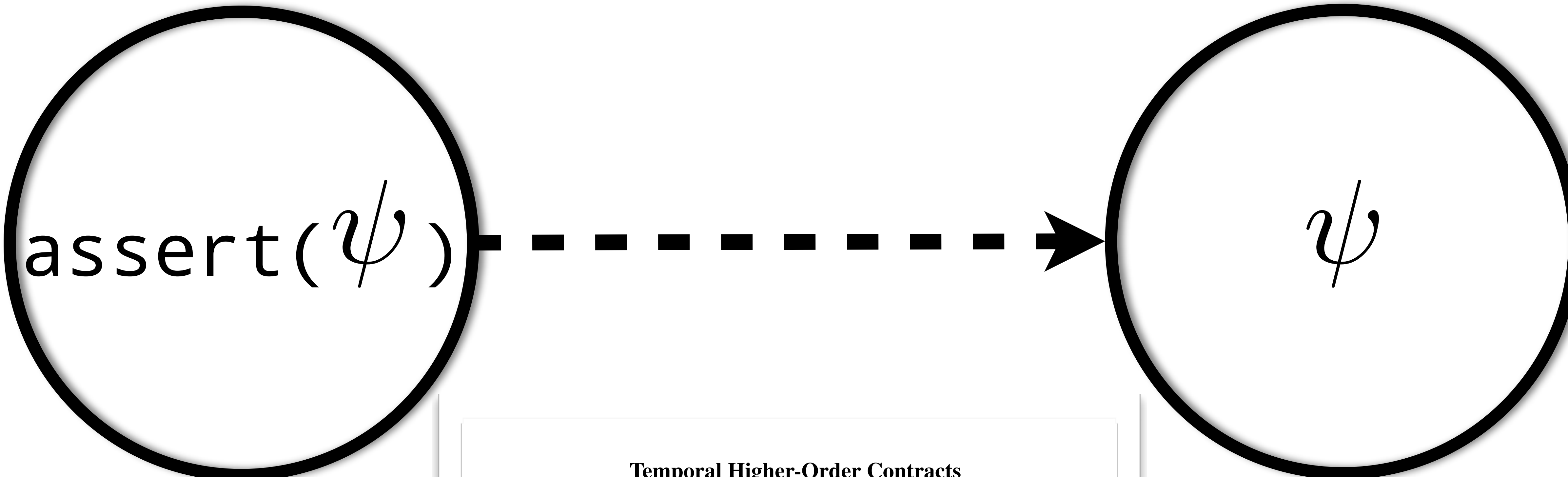


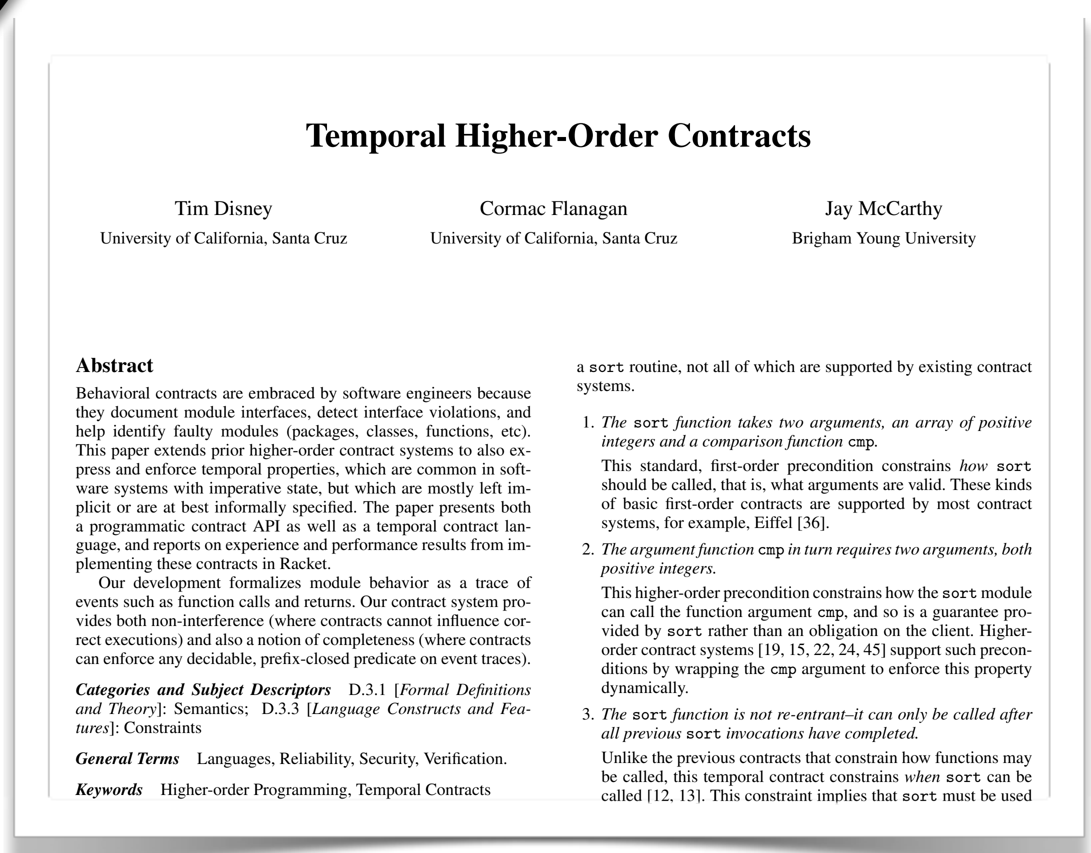

open(f);
close(f);


THE
SPIN MODEL
CHECKER
PRIMER AND REFERENCE MANUAL
Gerard J. Holzmann









Temporal Higher-Order Contracts

Tim Disney
University of California, Santa Cruz

Cormac Flanagan
University of California, Santa Cruz

Jay McCarthy
Brigham Young University

Abstract

Behavioral contracts are embraced by software engineers because they document module interfaces, detect interface violations, and help identify faulty modules (packages, classes, functions, etc). This paper extends prior higher-order contract systems to also express and enforce temporal properties, which are common in software systems with imperative state, but which are mostly left implicit or are at best informally specified. The paper presents both a programmatic contract API as well as a temporal contract language, and reports on experience and performance results from implementing these contracts in Racket.

Our development formalizes module behavior as a trace of events such as function calls and returns. Our contract system provides both non-interference (where contracts cannot influence correct executions) and also a notion of completeness (where contracts can enforce any decidable, prefix-closed predicate on event traces).

Categories and Subject Descriptors D.3.1 [*Formal Definitions and Theory*]: Semantics; D.3.3 [*Language Constructs and Features*]: Constraints

General Terms Languages, Reliability, Security, Verification.

Keywords Higher-order Programming, Temporal Contracts

a `sort` routine, not all of which are supported by existing contract systems.

1. *The* `sort` *function takes two arguments, an array of positive integers and a comparison function* `cmp`*.*

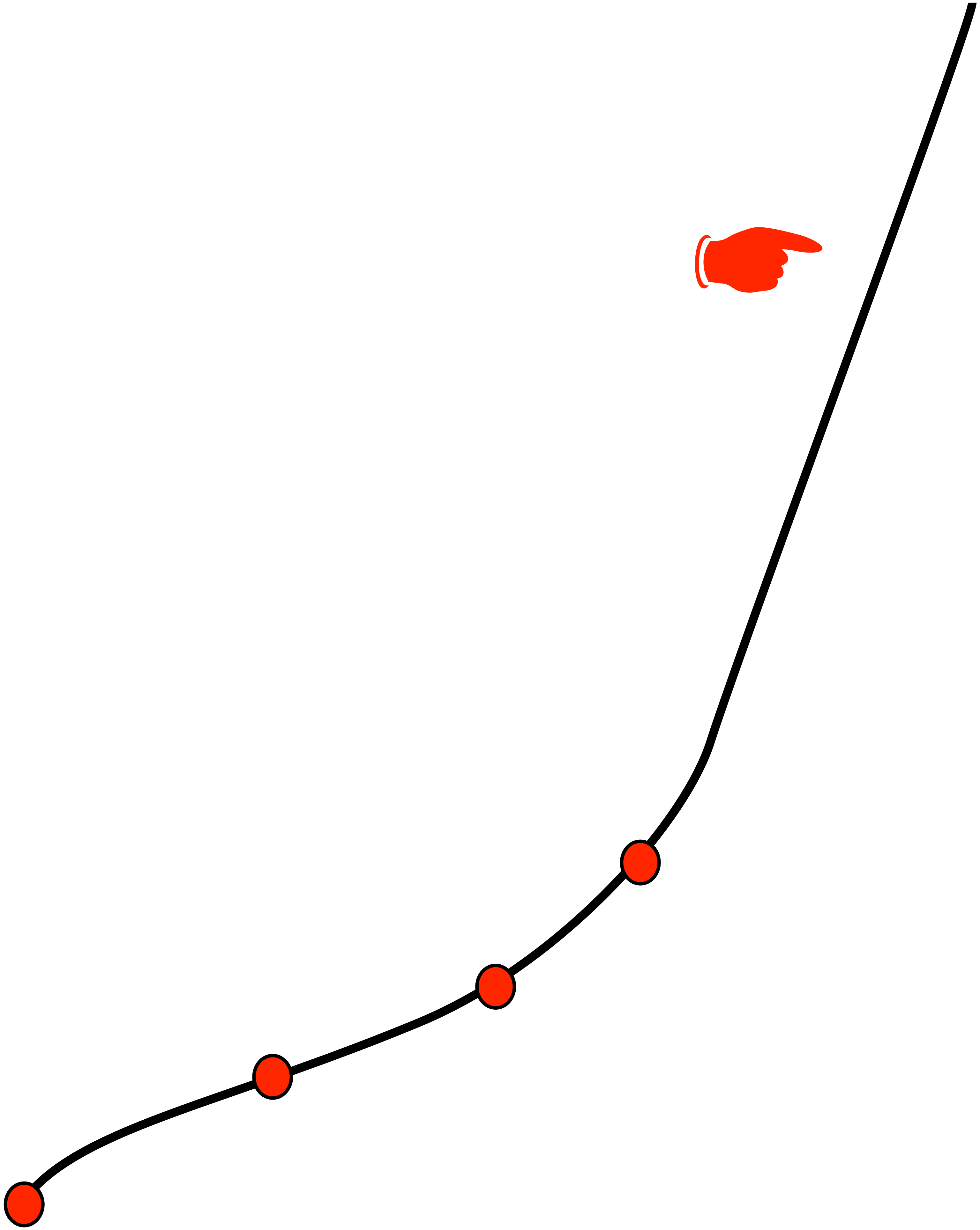
   This standard, first-order precondition constrains *how* `sort` should be called, that is, what arguments are valid. These kinds of basic first-order contracts are supported by most contract systems, for example, Eiffel [36].

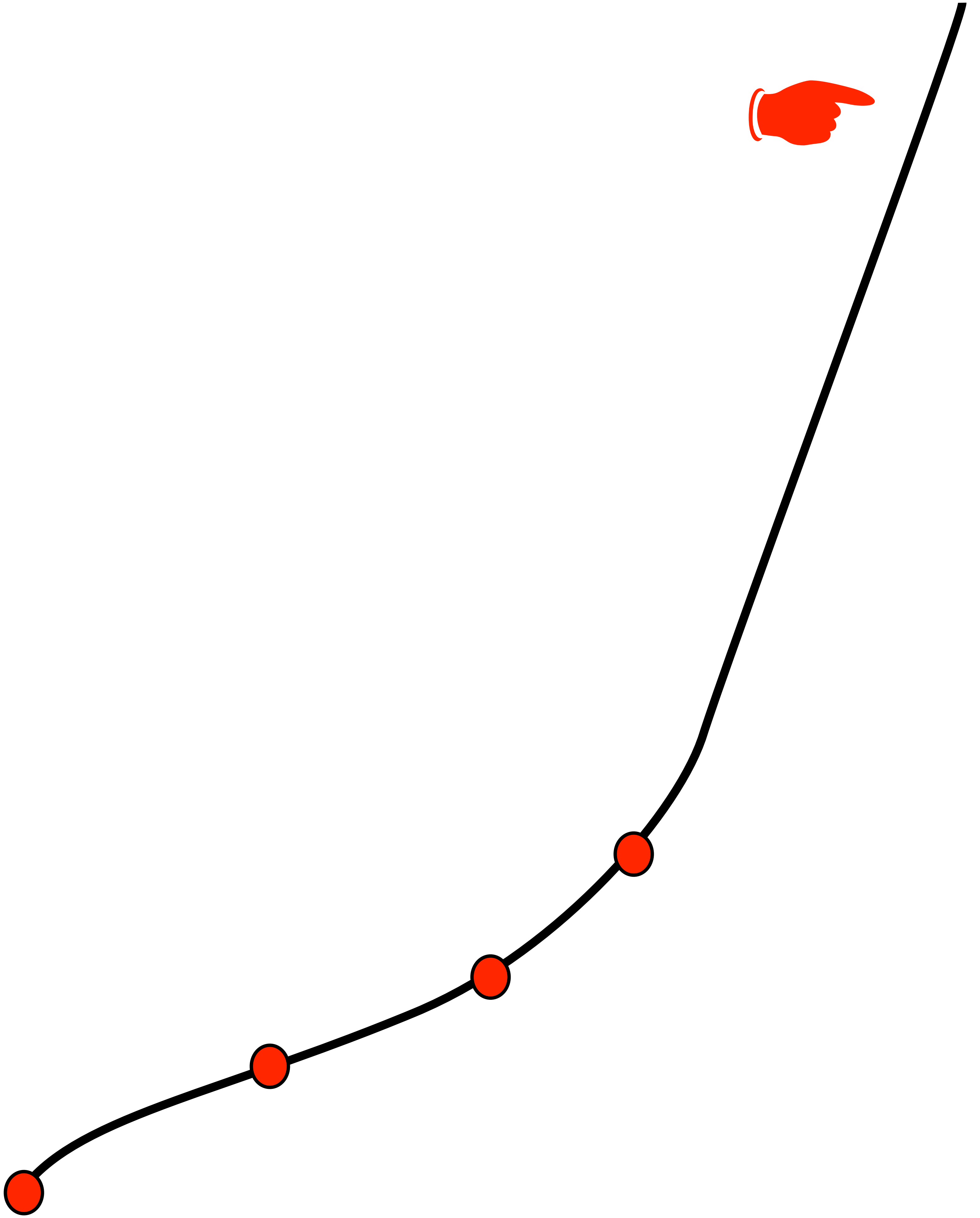
2. *The argument function* `cmp` *in turn requires two arguments, both positive integers.*

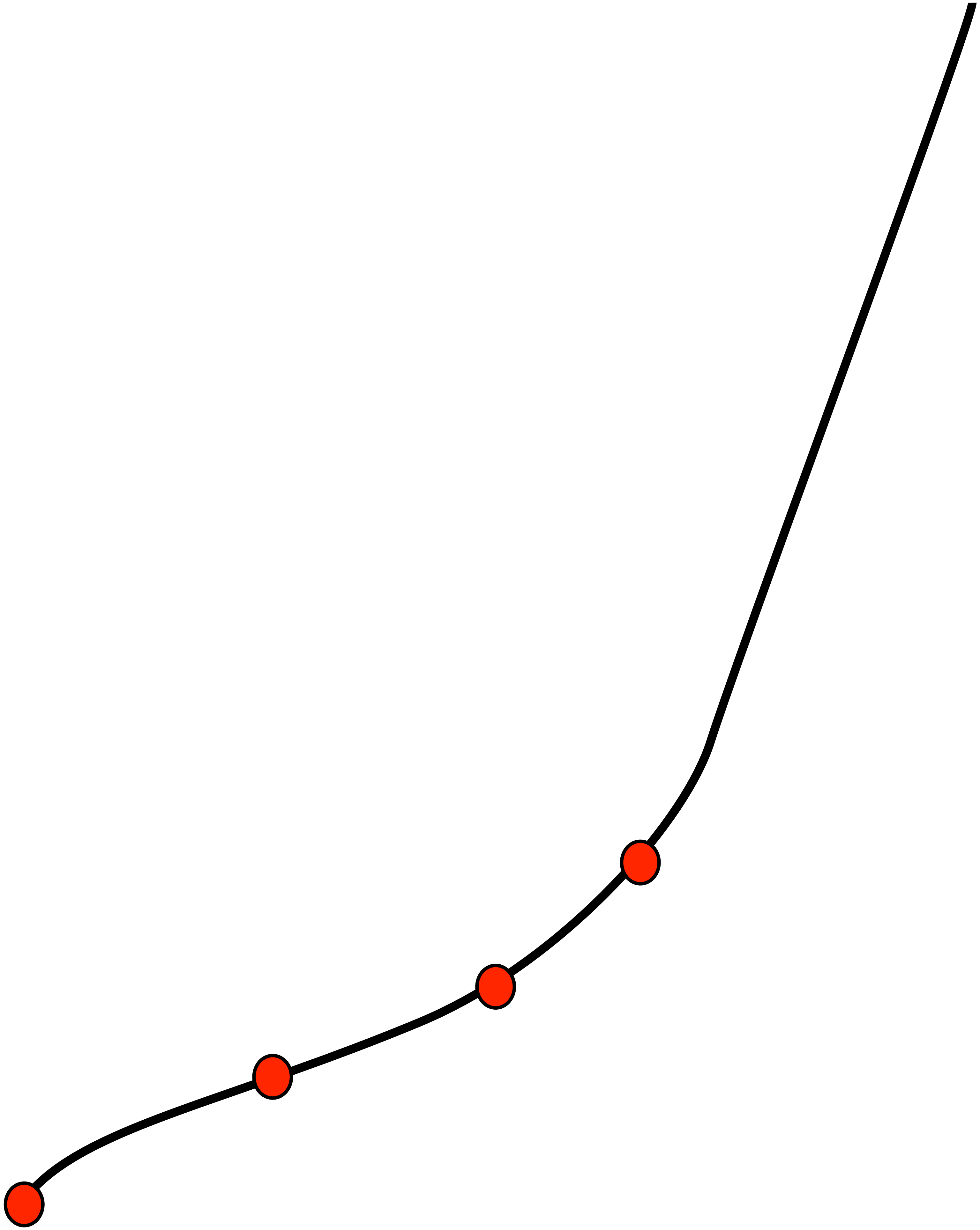
   This higher-order precondition constrains how the `sort` module can call the function argument `cmp`, and so is a guarantee provided by `sort` rather than an obligation on the client. Higher-order contract systems [19, 15, 22, 24, 45] support such preconditions by wrapping the `cmp` argument to enforce this property dynamically.

3. *The* `sort` *function is not re-entrant–it can only be called after all previous* `sort` *invocations have completed.*

   Unlike the previous contracts that constrain how functions may be called, this temporal contract constrains *when* `sort` can be called [12, 13]. This constraint implies that `sort` must be used

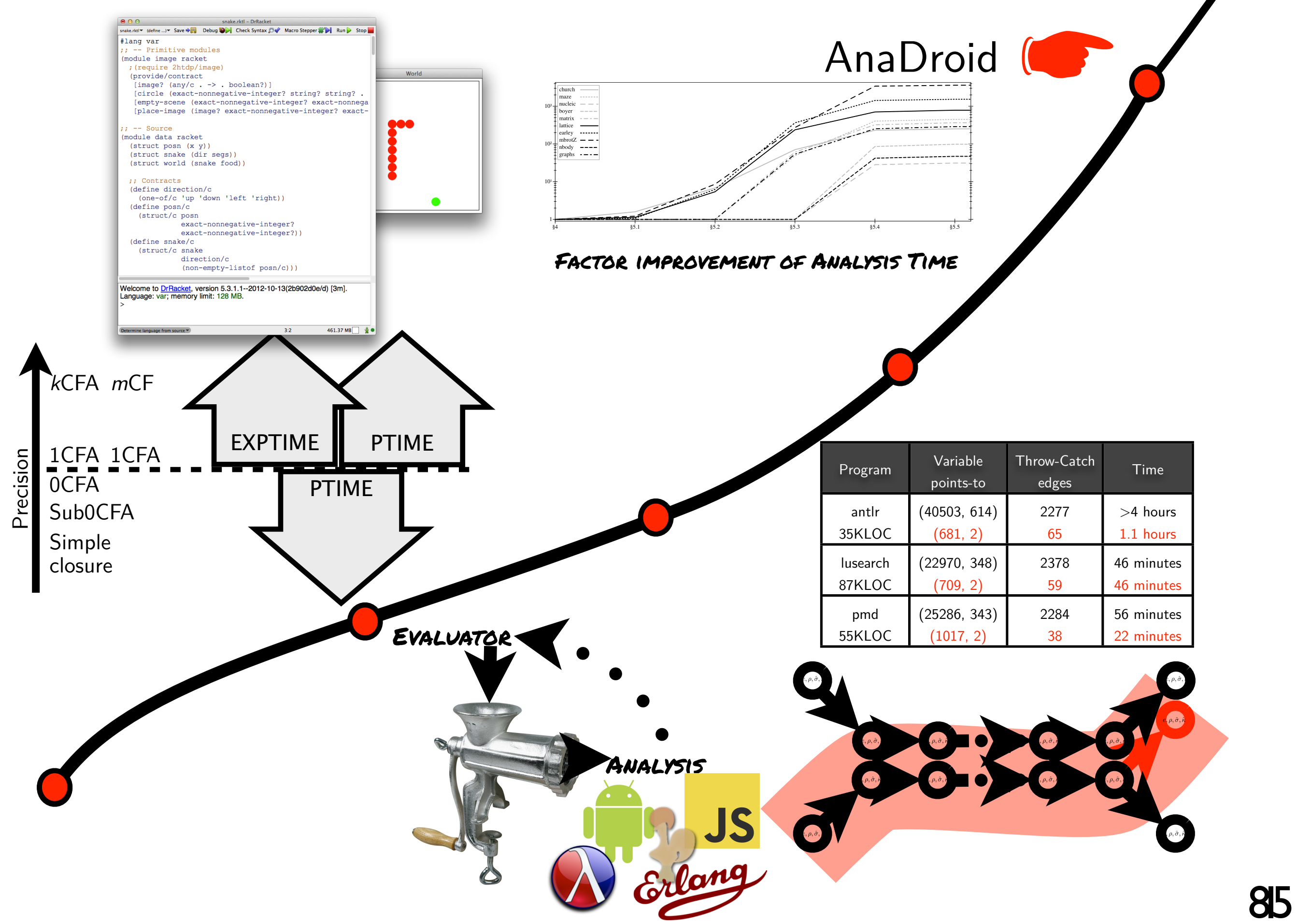






Robust, Reliable Software and Trustworthy Systems

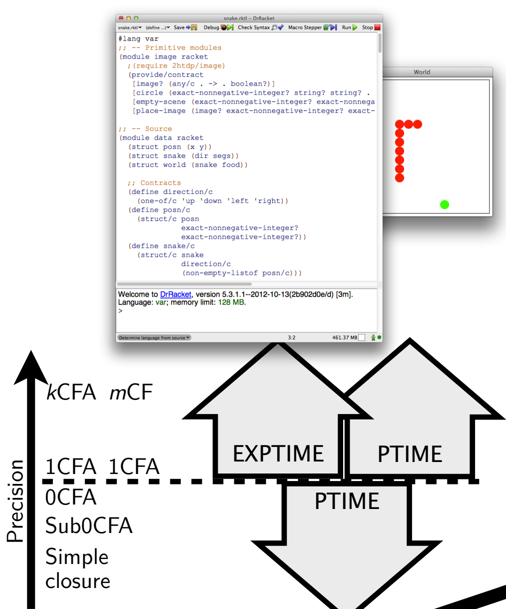
Thank you



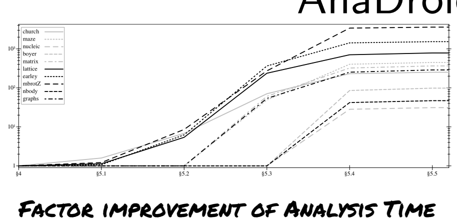
Program	Variable points-to	Throw-Catch edges	Time
antlr 35KLOC	(40503, 614) (681, 2)	2277 65	>4 hours 1.1 hours
lusearch 87KLOC	(22970, 348) (709, 2)	2378 59	46 minutes 46 minutes
pmd 55KLOC	(25286, 343) (1017, 2)	2284 38	56 minutes 22 minutes

Robust, Reliable Software and Trustworthy Systems

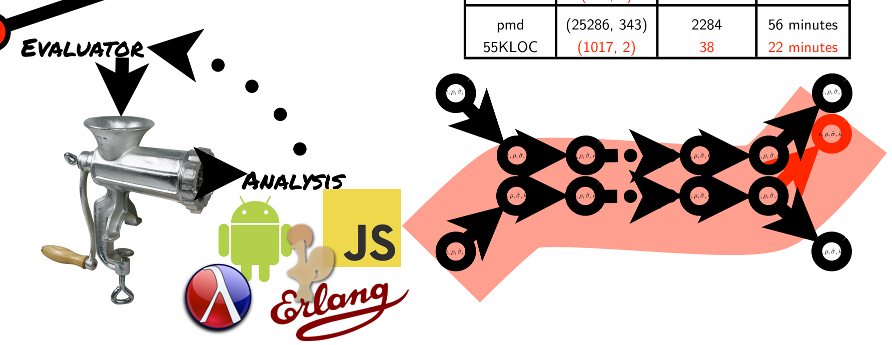
Thank you



AnaDroid



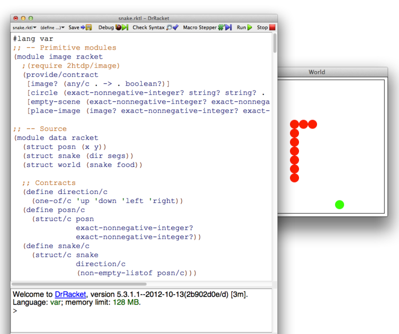
Program	Variable points-to	Throw-Catch edges	Time
antlr 35KLOC	(40503, 614) (681, 2)	2277 65	>4 hours 1.1 hours
lusearch 87KLOC	(22970, 348) (709, 2)	2378 59	46 minutes 46 minutes
pmd 55KLOC	(25286, 343) (1017, 2)	2284 38	56 minutes 22 minutes

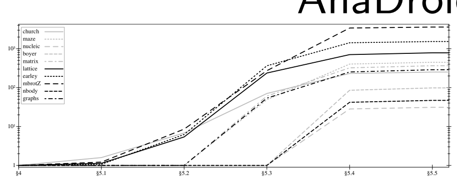


Robust, Reliable Software and Trustworthy Systems

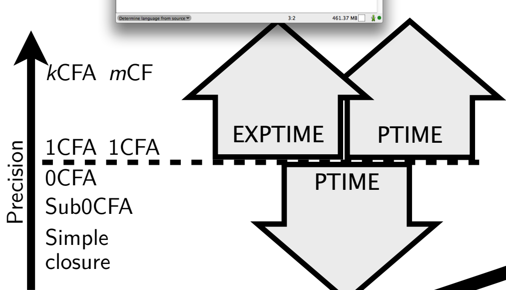
Thank you

AnaDroid



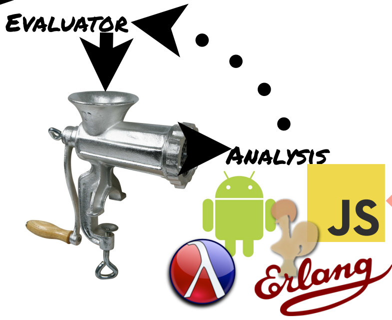


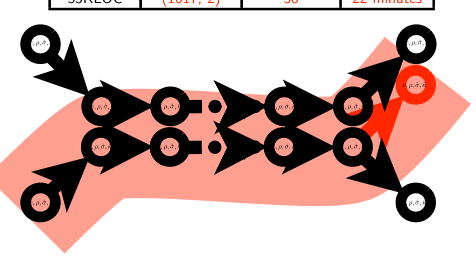
FACTOR IMPROVEMENT OF ANALYSIS TIME



Program	Variable points-to	Throw-Catch edges	Time
antlr 35KLOC	(40503, 614) (681, 2)	2277 65	>4 hours 1.1 hours
lusearch 87KLOC	(22970, 348) (709, 2)	2378 59	46 minutes 46 minutes
pmd 55KLOC	(25286, 343) (1017, 2)	2284 38	56 minutes 22 minutes

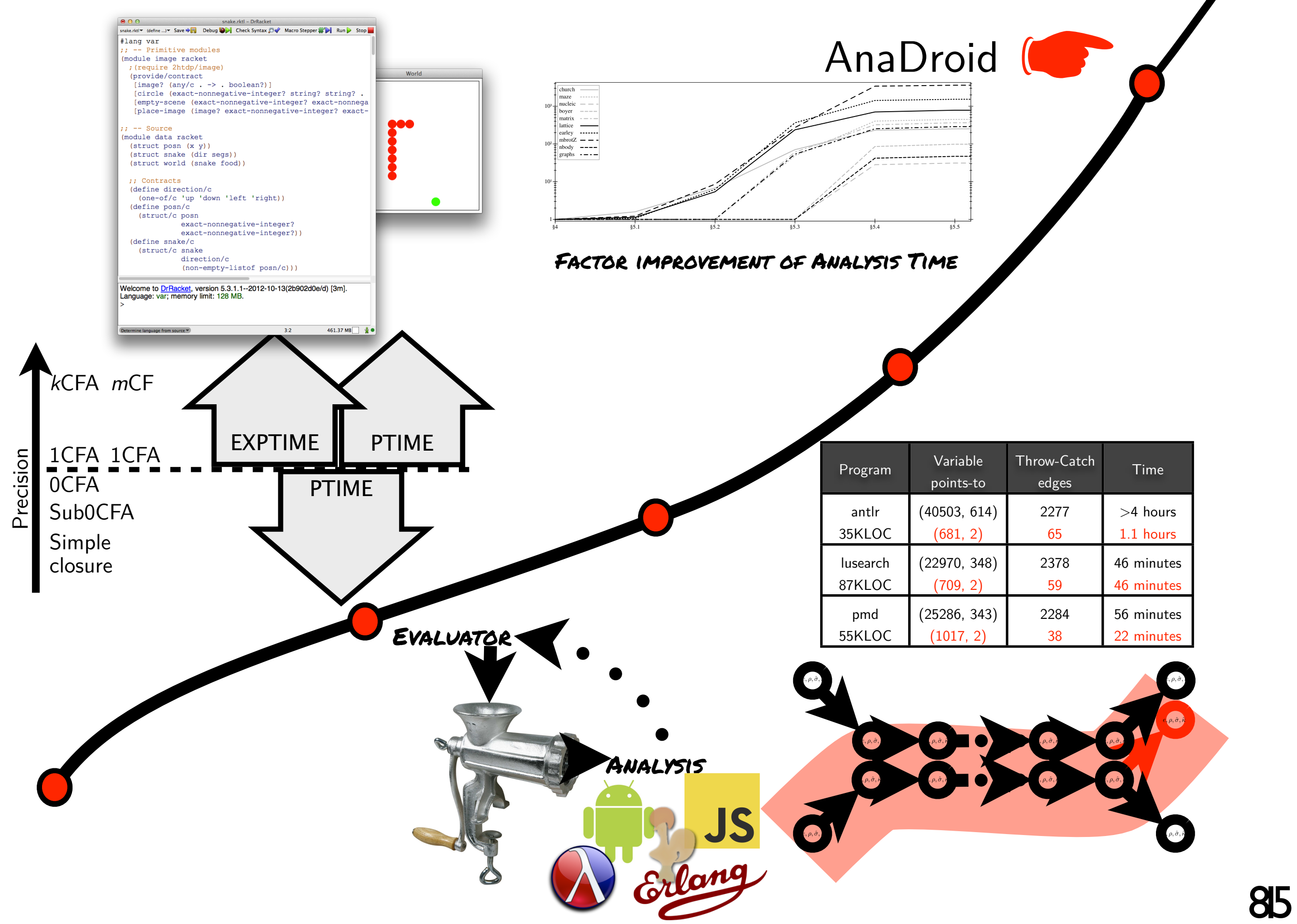
EVALUATOR





Robust, Reliable Software and Trustworthy Systems

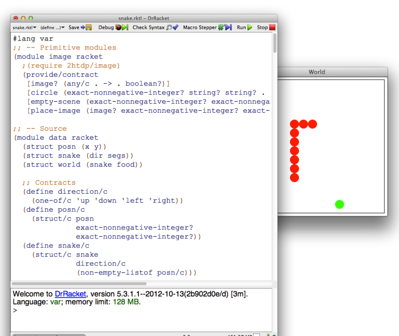
Thank you

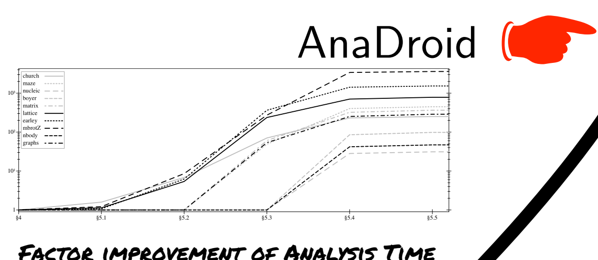


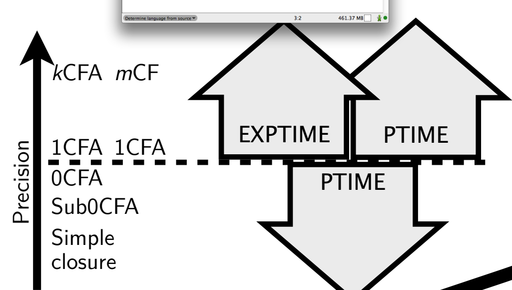
Program	Variable points-to	Throw-Catch edges	Time
antlr 35KLOC	(40503, 614) (681, 2)	2277 65	>4 hours 1.1 hours
lusearch 87KLOC	(22970, 348) (709, 2)	2378 59	46 minutes 46 minutes
pmd 55KLOC	(25286, 343) (1017, 2)	2284 38	56 minutes 22 minutes

Robust, Reliable Software and Trustworthy Systems

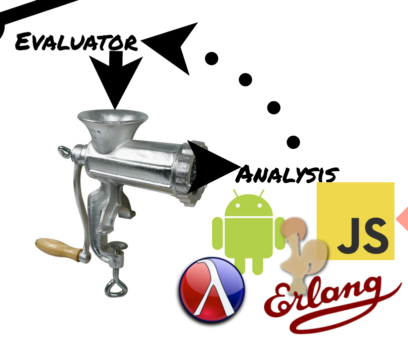
Thank you

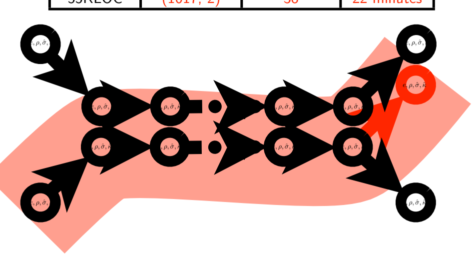




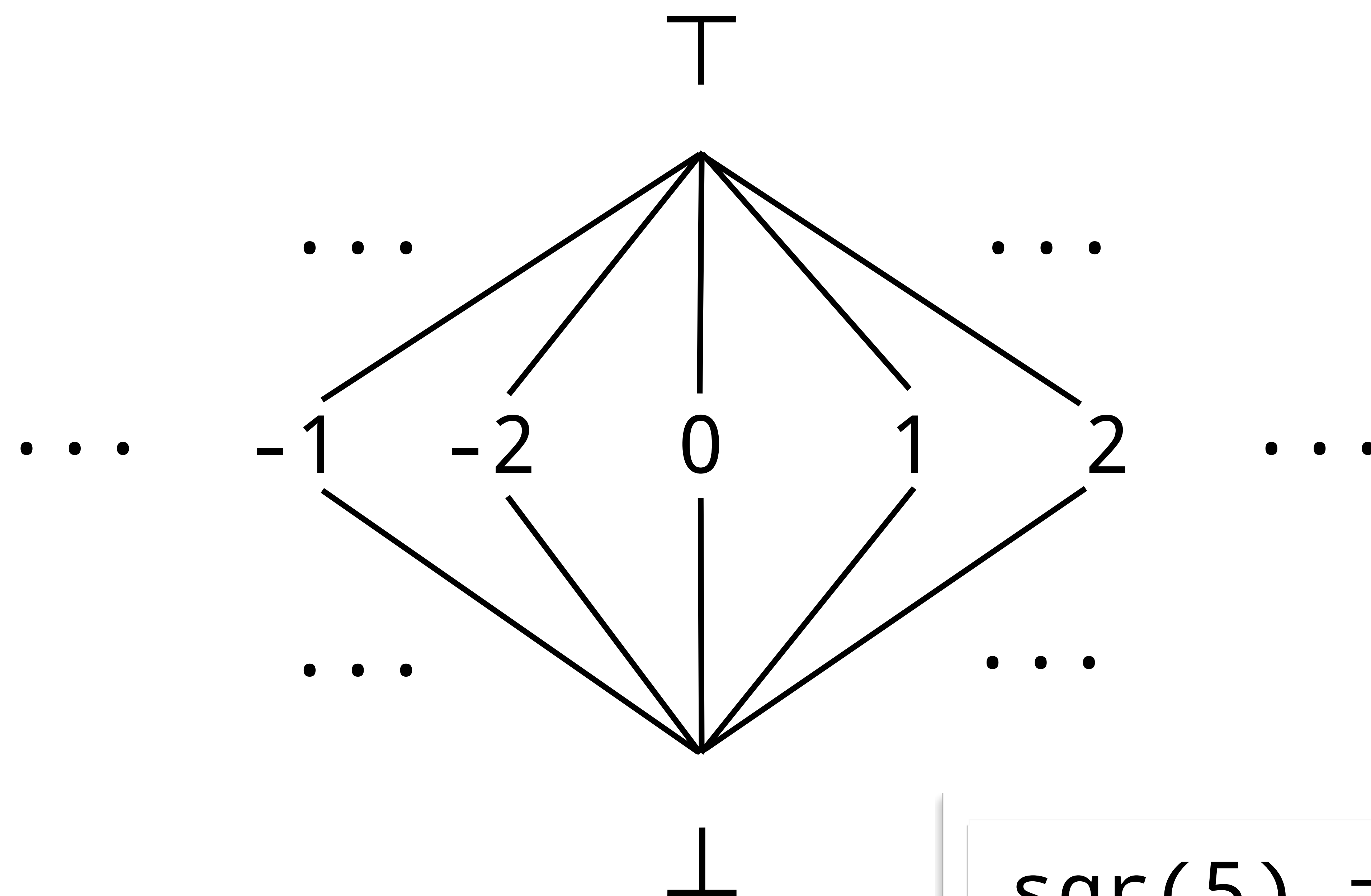


Program	Variable points-to	Throw-Catch edges	Time
antlr 35KLOC	(40503, 614) (681, 2)	2277 65	>4 hours 1.1 hours
lusearch 87KLOC	(22970, 348) (709, 2)	2378 59	46 minutes 46 minutes
pmd 55KLOC	(25286, 343) (1017, 2)	2284 38	56 minutes 22 minutes



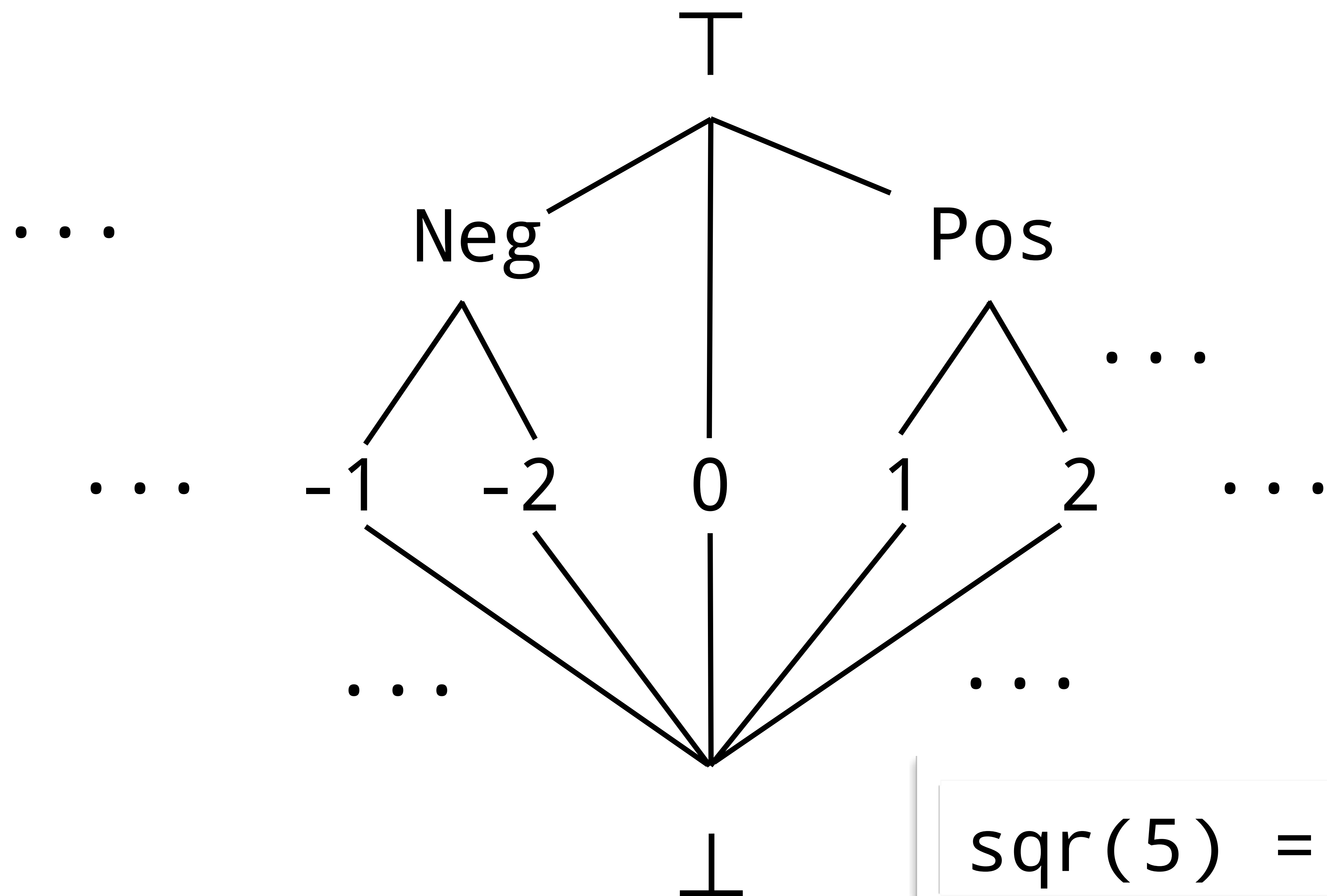


What about numbers, strings, arrays, etc.?




```
sqr(5) = ⊤
```

What about numbers, strings, arrays, etc.?




```
sqr(5) = Pos
```