Κόμπος: A Concurrency-Agnostic Debugger

An Example for Domain-Specific Online Debugging





VRIJE UNIVERSITEIT BRUSSEL

C. Torres Lopez E. Gonzalez Boix S. Marr D. Aumayr H. Mössenböck

JOHANNES KEPLER UNIVERSITY LINZ

Research Foundation Flanders
Opening new horizons

IFIP WG 2.11, July 17-20, 2017, Koblenz

FWF Der Wissenschaftsfonds.

What am I doing?

A simple VM for all concurrency models

actors, CSP, STM, fork/join, data flow, threads+locks, …

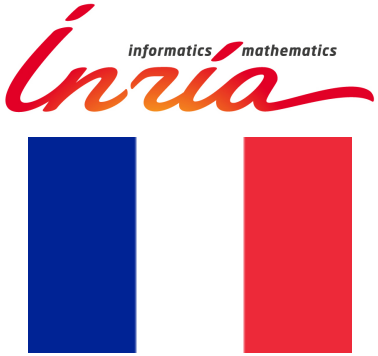
Ownership-based metaobject protocol

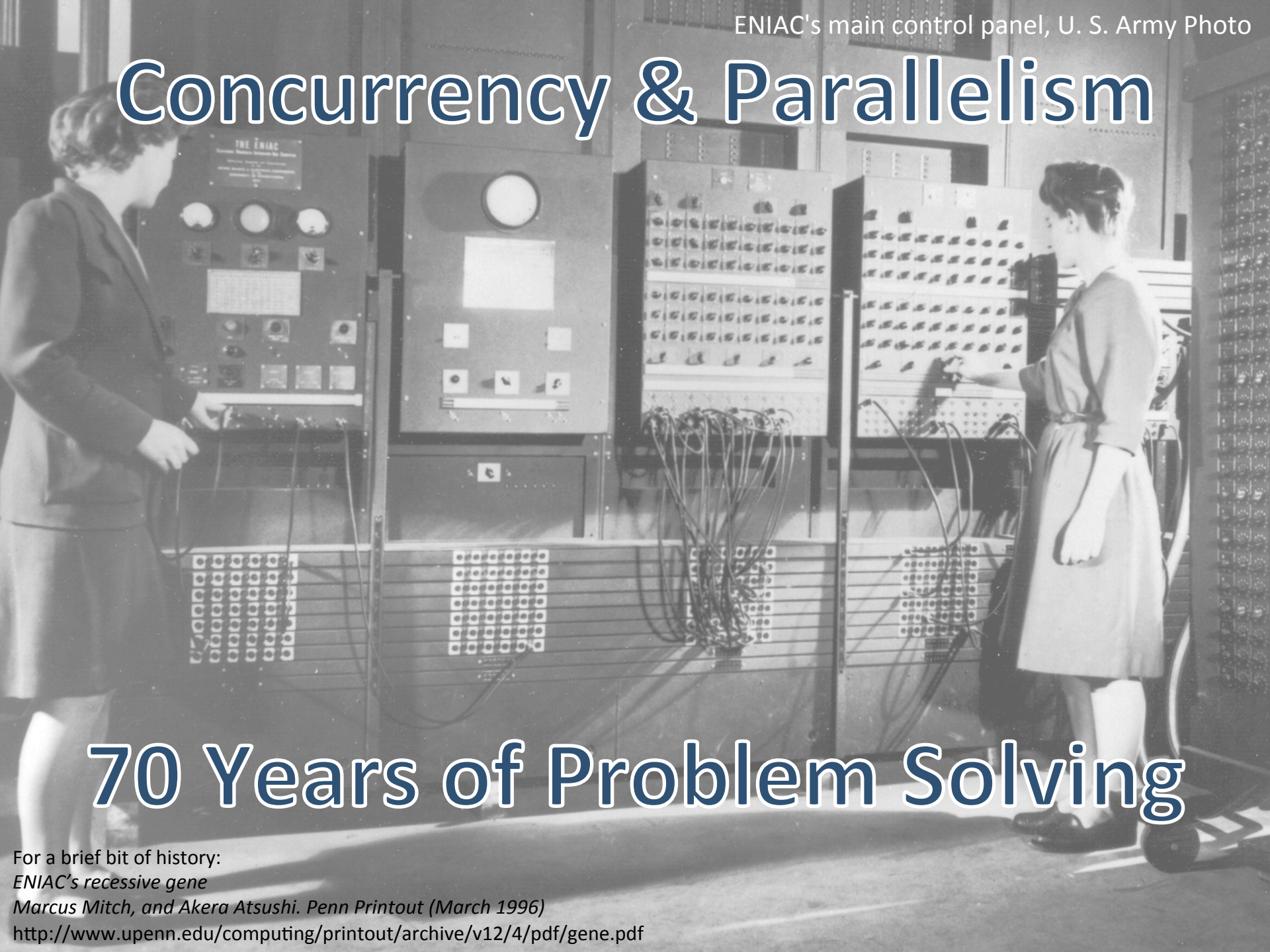
Making dynamic languages fast, easily! (incl. metaobject protocols)

Meta-Tracing vs. Partial Evaluation

Combining Concurrency Models

Tooling/Debugging





ENIAC's main control panel, U. S. Army Photo

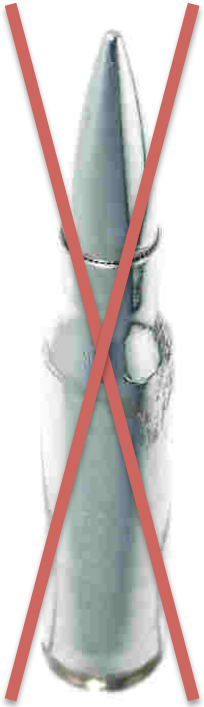
Concurrency & Parallelism

70 Years of Problem Solving

For a brief bit of history:
ENIAC's recessive gene
Marcus Mitch, and Akera Atsushi. Penn Printout (March 1996)
http://www.upenn.edu/computing/printout/archive/v12/4/pdf/gene.pdf

Decades of Research and Solutions for "*Everything*"

But no Silver Bullet



- **Actors**
- **CSP**
- **Locks, Monitors, …**
- **Fork/Join**
- **Transactional Memory**
- **Data Flow**
- …

In Practice: Concurrency Abstractions are Combined

- Uses Locks and Atomic*
- Multiple async. future/ task abstractions
- Multiple 'transaction' systems

> 4500 "deadlock" bugs

> 530 "race condition" bugs

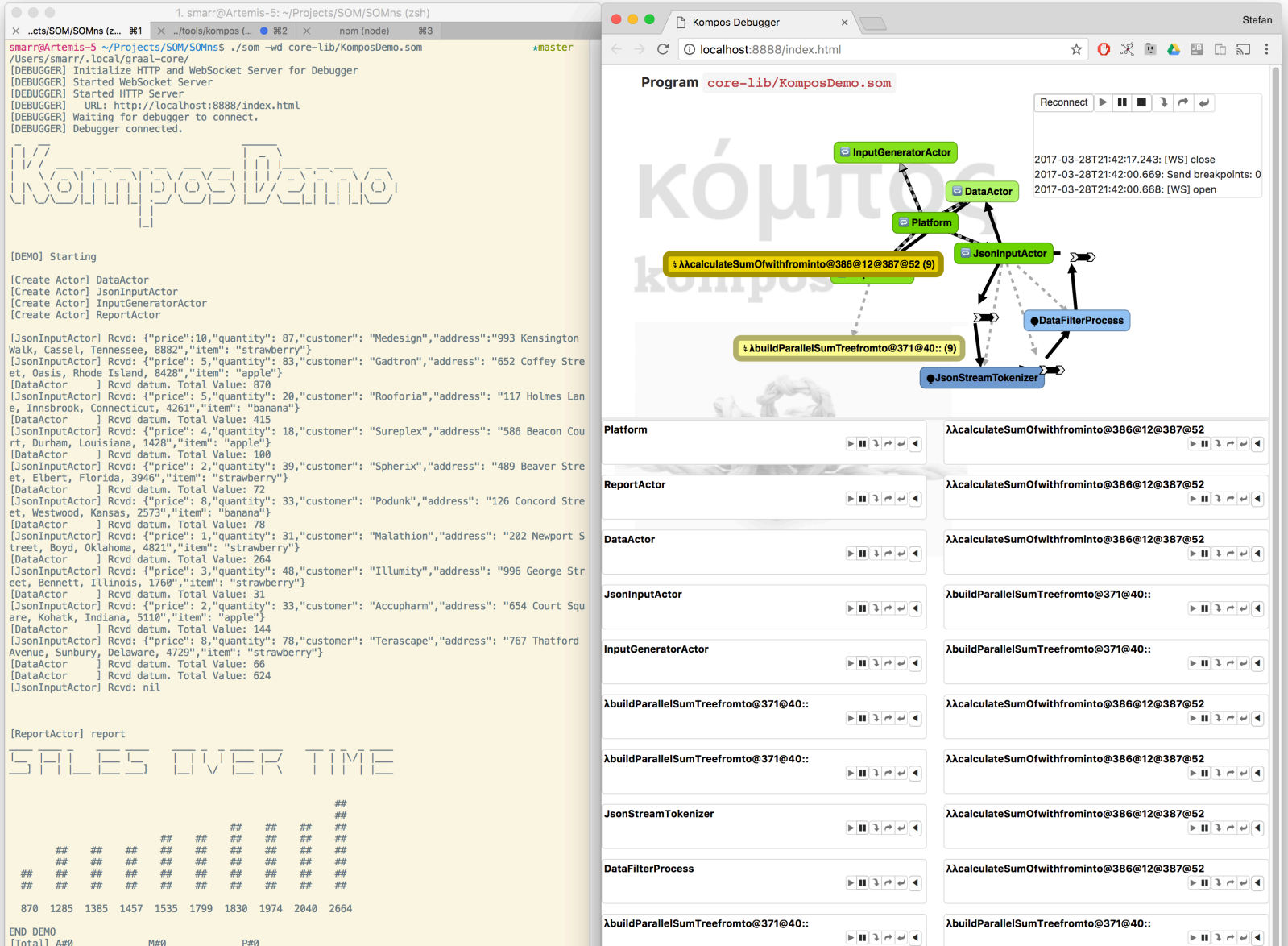
Study on Scala:
S. Tasharofi, P. Dinges, and R. E. Johnson. Why Do Scala Developers Mix the Actor Model with other Concurrency Models? In Proc. of ECOOP, volume 7920 of LNCS, pages 302–326. Springer, 2013.

I want to reason about actors, fork/join tasks, transactions…

And not some implementation-level shared memory

HOW TO DEBUG SUCH SYSTEMS?

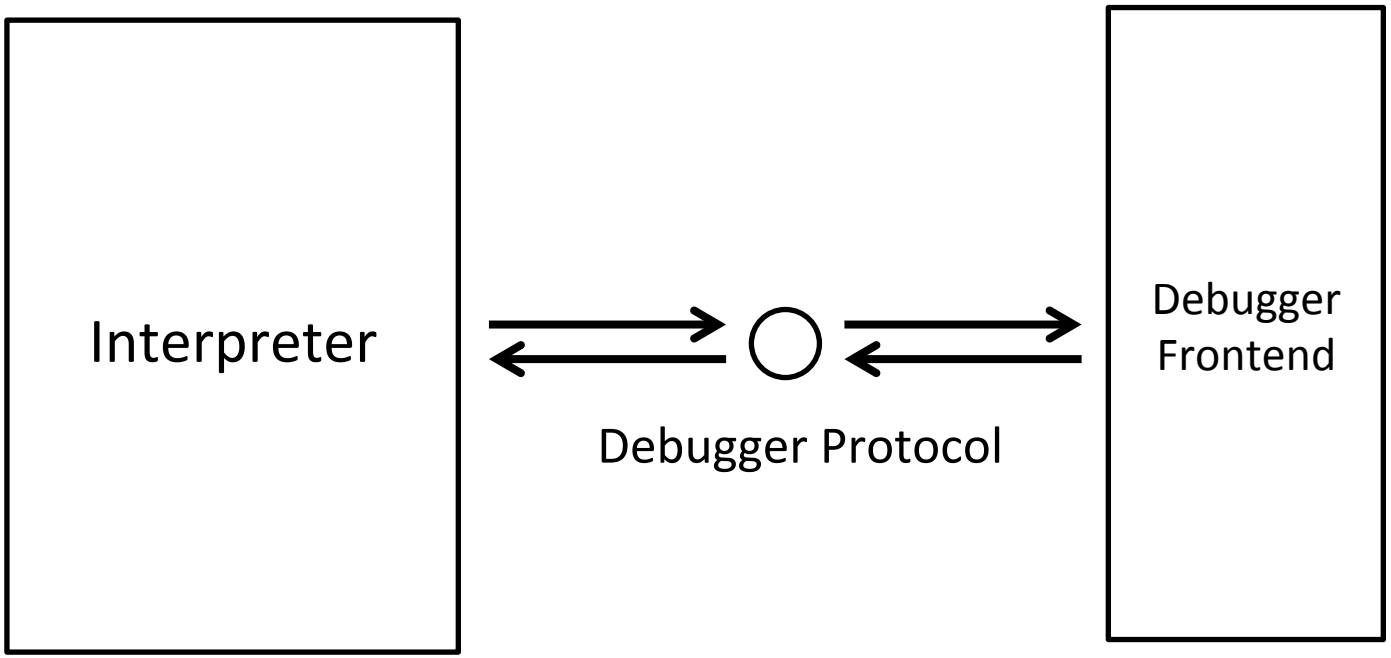
Demo of Kómpos



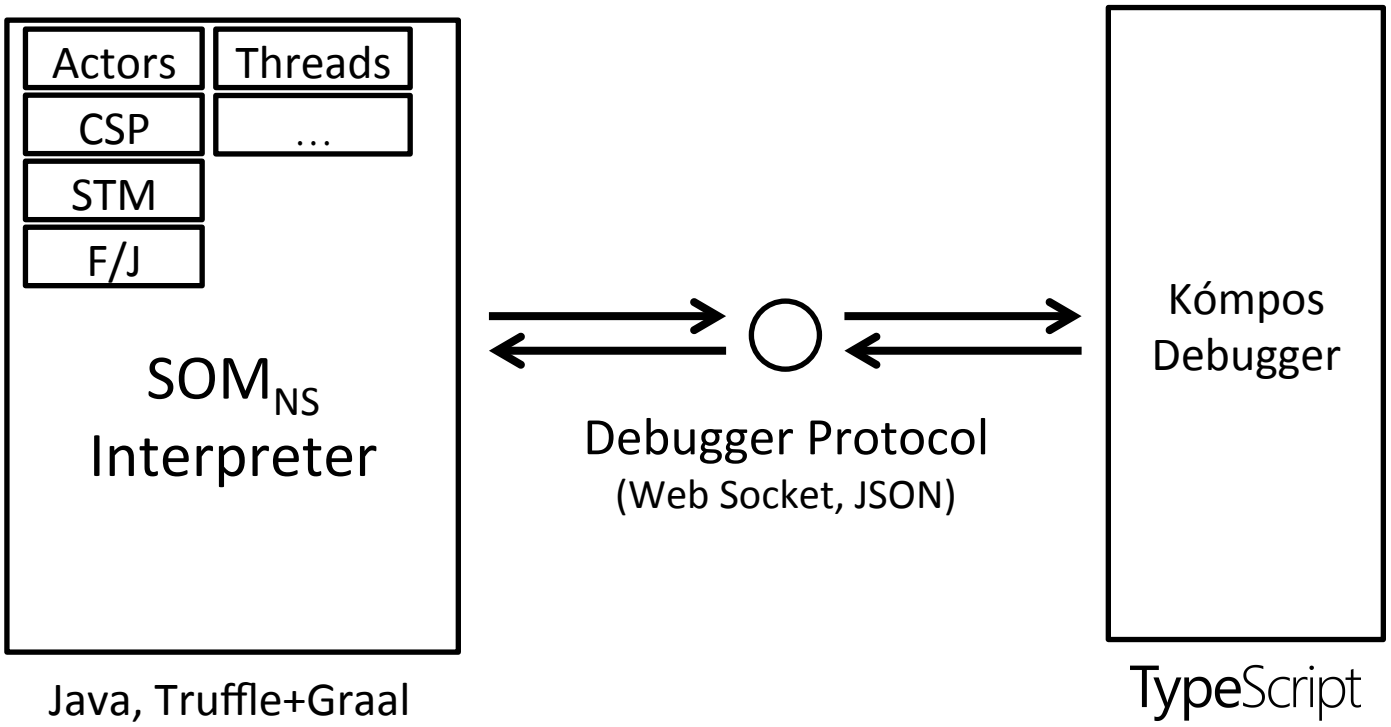
Custom Debugger Features

	Breakpoints	Stepping Operations
Sequential	1	5
Generic	4	2
Actors	6	5
CSP	4	2
STM	3	3
Threads&Locks	4	2
(not intended to be complete)	**22**	**19**

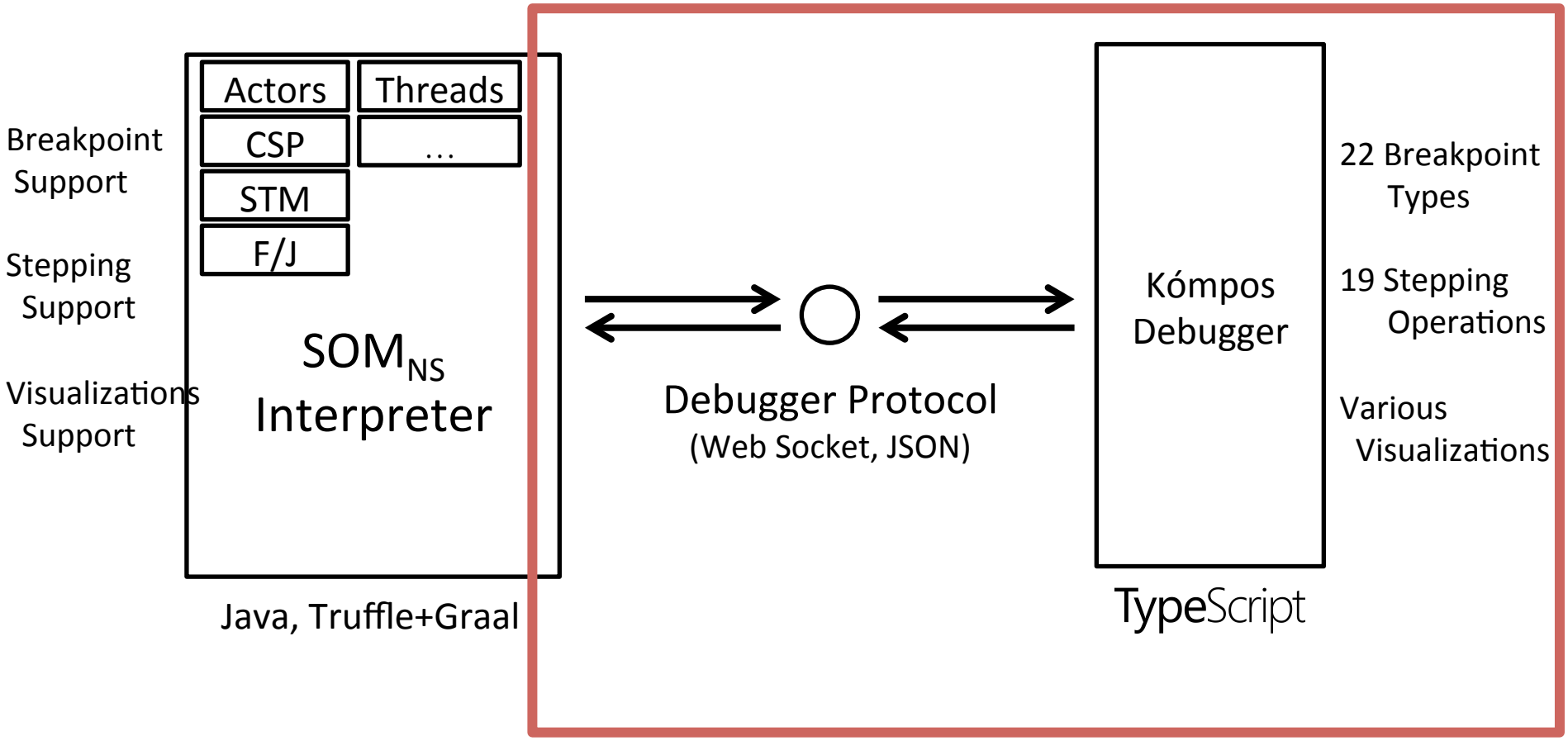
Debugger Architecture



Kómpos Architecture



Kómpos Architecture



How to Abstract from Concurrency Models?

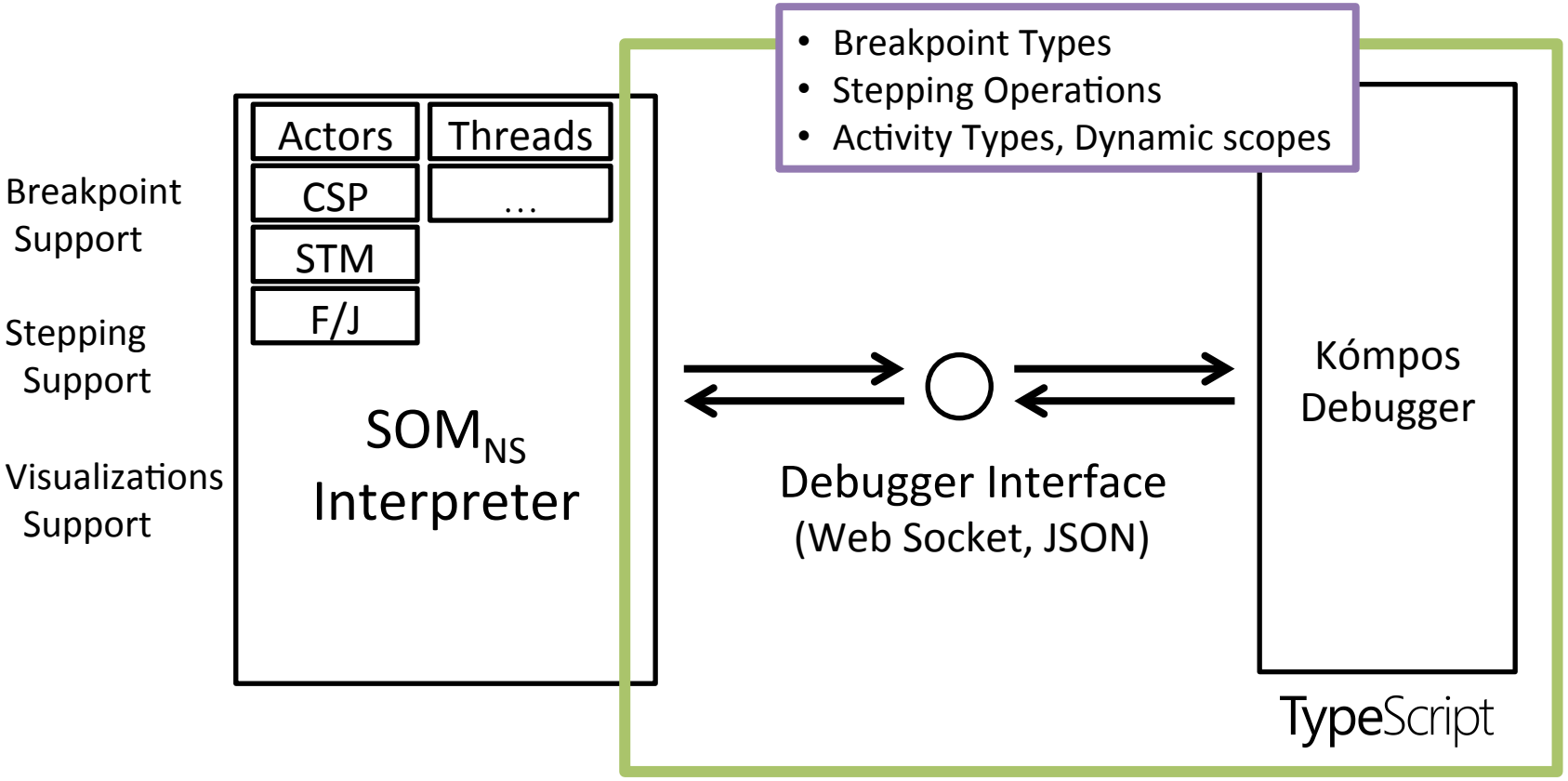
κόμπος

kompos

A Concurrency-Agnostic Debugger Protocol

THE KÓMPOS PROTOCOL

Abstract from Concurrency Models



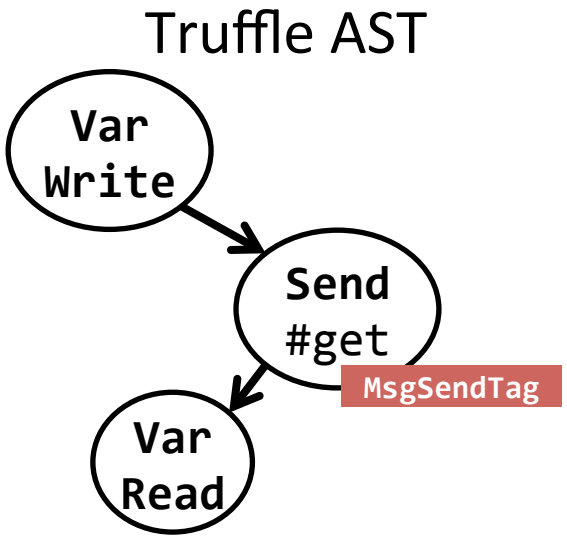
- Interactions modeled via data
- Debugger agnostic of concepts

What's a Breakpoint?

message send
message receive
promise resolver
promise resolution

```
prom := aResult <-: get.
```

- Name/Type
- Source Locations
  - Identified by Source Tags



Debugger Requirements

- Annotated AST

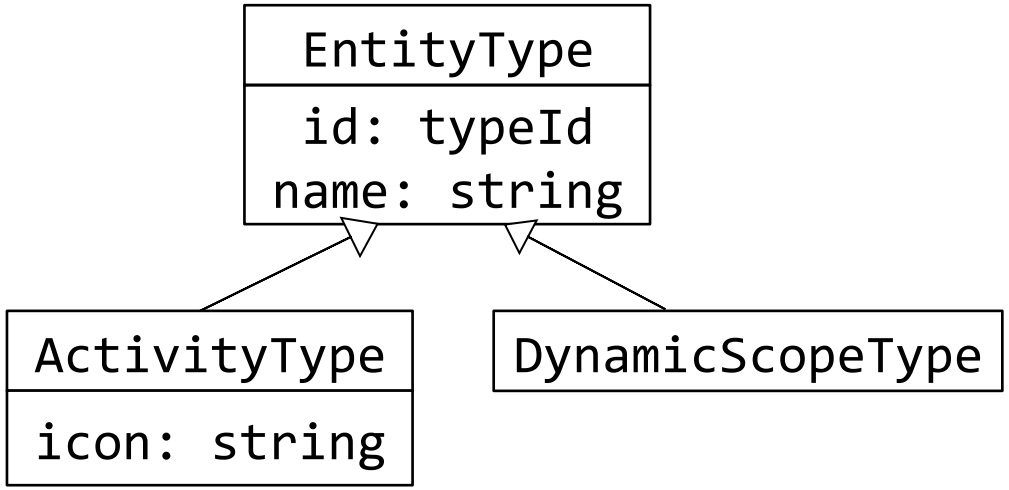
What's a Stepping Operation?

Step into
Step over
Return
Step to Receiver
Step to Promise Resolver
Step to Promise Resolution

```
prom :=
    aResult <-: get.
```

- Name/Type
- Source Locations
  - Identified by Source Tags
- Current Type of Activity
  - Actor, Process, Fork/Join Task, Thread
- Current Dynamic Scope
  - Held Monitor, Transaction

Kómpos Protocol Metadata



BreakpointType
name: string label: string applicableTo: Tag[]

SteppingType
name: string label: string applicableTo: Tag[] activities: ActivityType[] scopes: DynamicScopeType[]

Kómpos Protocol Messages

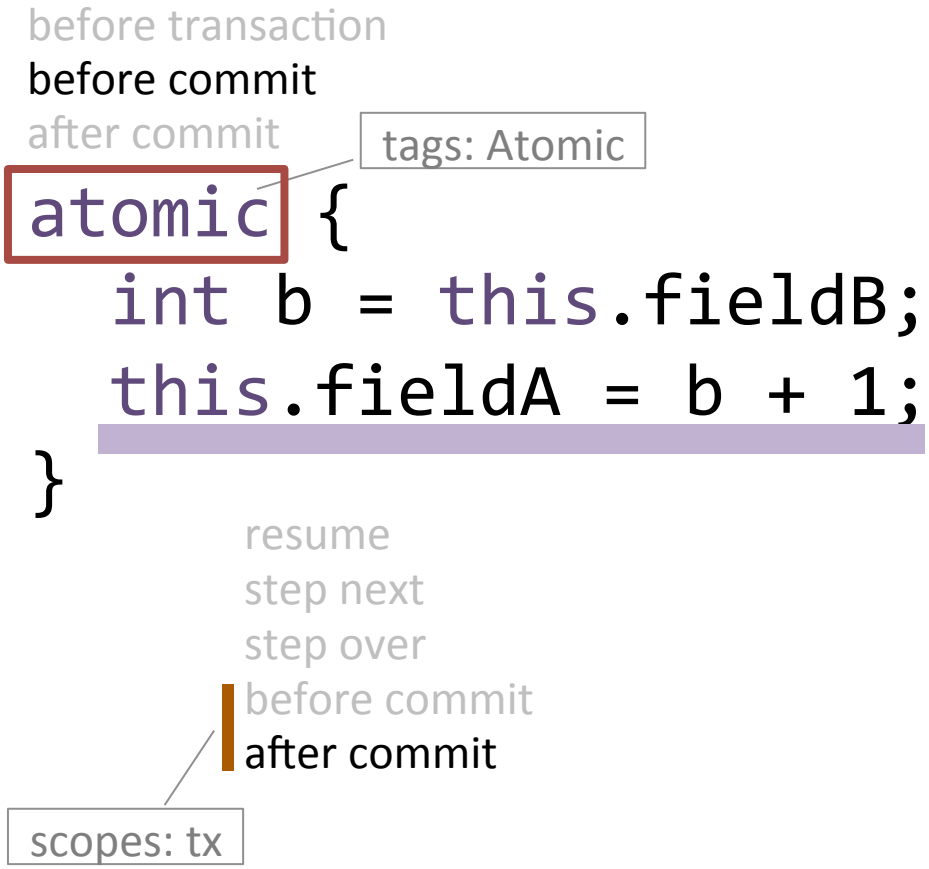
Source
URI: URI sourceText: string locations: TaggedCoord[]

BreakpointUpdate
location: Coord type: BreakpointType

Stopped
activityId: id location: Coord actType: ActivityType scopes: DynamicScopeType[]

Step
activityId: id type: SteppingType

Example for Transactions




```
to int: {BreakpointUpdate,
         loc: 1:1:6,
         type: beforeCommit}

from int: {Stopped,
           actId: 1,
           loc: 3:3:20,
           scopes: [tx]}

to int: {Step,
         actId: 1,
         type: afterCommit}
```

How concurrency-agnostic is the protocol design?

EVALUATION

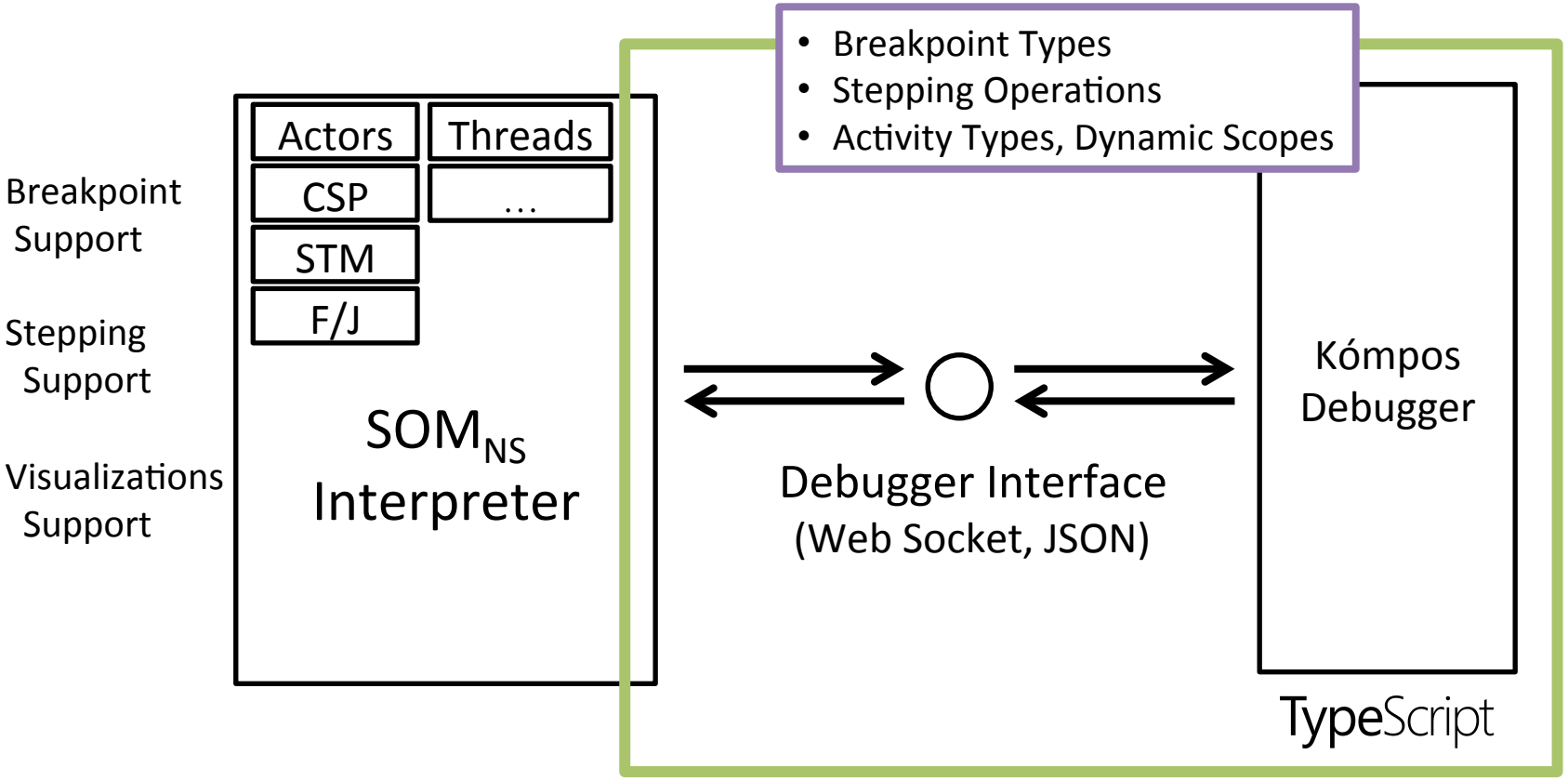
Implemented Interactions
No changes to frontend required!

	Breakpoints	Stepping Operations
Sequential	1	5
Generic	4	2
Actors	6	5
CSP	4	2
STM	3	3
Threads&Locks	4	2
(not intended to be complete)	**22**	**19**

CONCLUSION

Too many Concepts, Too many Variations

Solution:
Model Domain with Metadata



META

Workshop on Meta-Programming Techniques and Reflection

Paper Submission: 14 Aug 2017
Late WIP Submission: 7 Sep 2017

Co-located with

SPLASH VANCOUVER 2017 OCTOBER 22–27

Paper Deadline: 14 August 2017

Workshop Proposals: 1 October 2017

Kómpos: Platform for Debugging Complex Concurrent Applications

- Concurrent Debugger
  - Concept-agnostic
- Visualization, Breakpoints, Stepping, Tracing
- Replay, Assertions,…
- For Actors, CSP, Fork/Join, STM, Locks&Threads,…