Zurück in die Zukunft:
Back-in-time-Debugging mit dem TraceDebugger

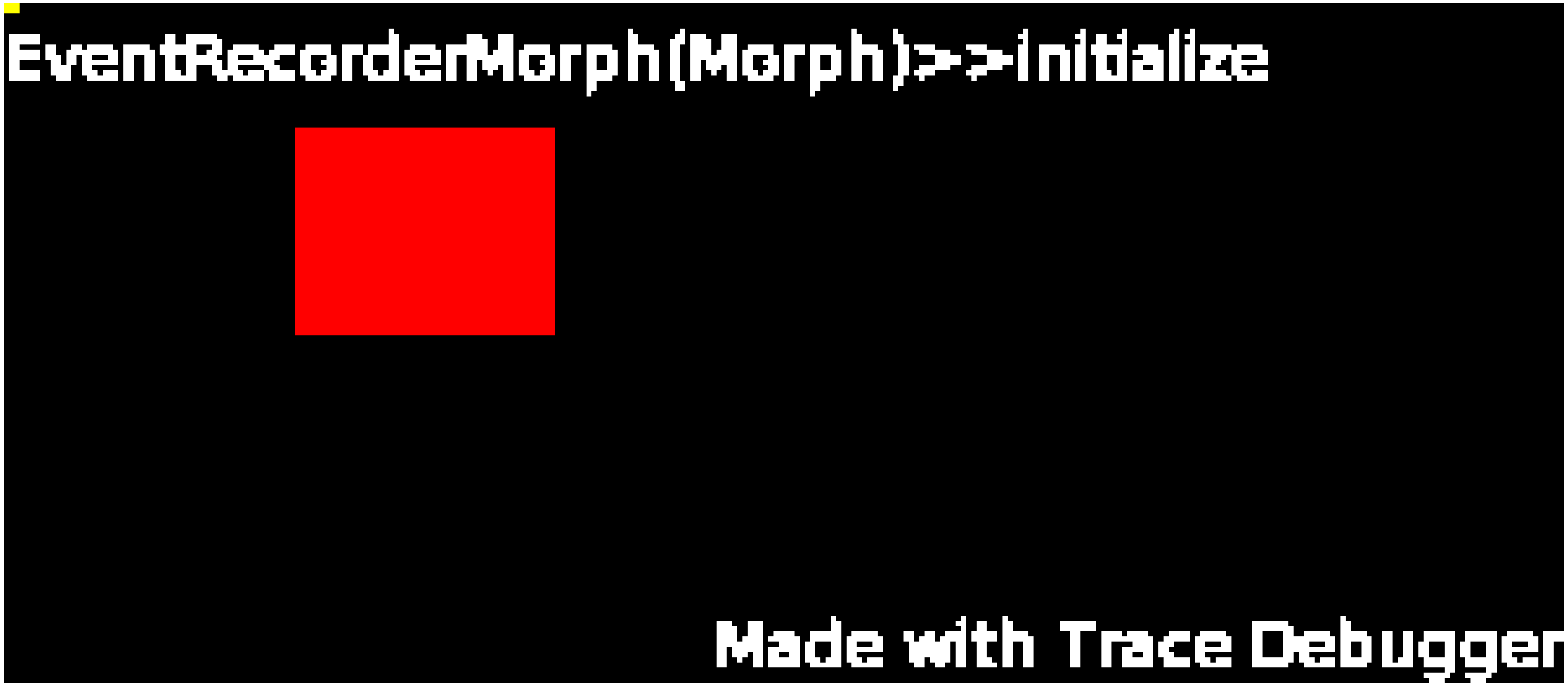
Christoph Thiede

Squeak Demos ’22

2022-11-19

Einige Object Traces …

- Morphic Layout



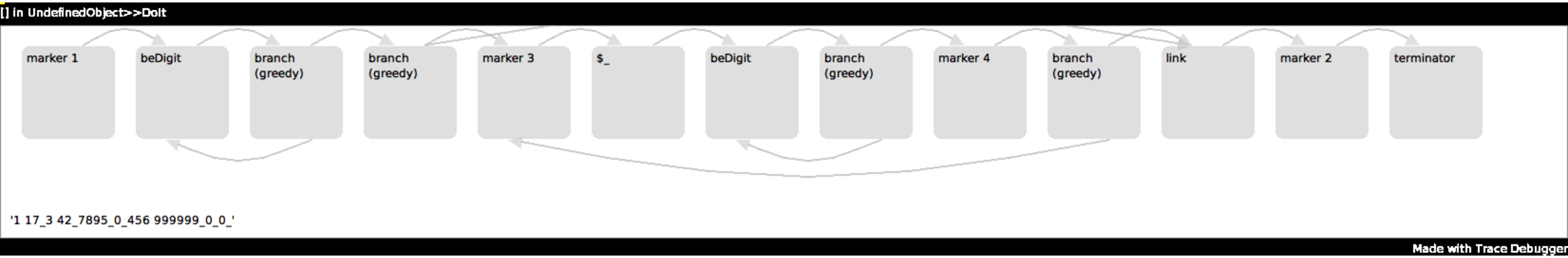
EventRecorderMorph(Morph)>>initialize

Einige Object Traces …

- Morphic Rendering

Einige Object Traces …

- Matching regulärer Ausdrücke visualisieren



'1 17_3 42_7895_0_456 999999_0_0_'

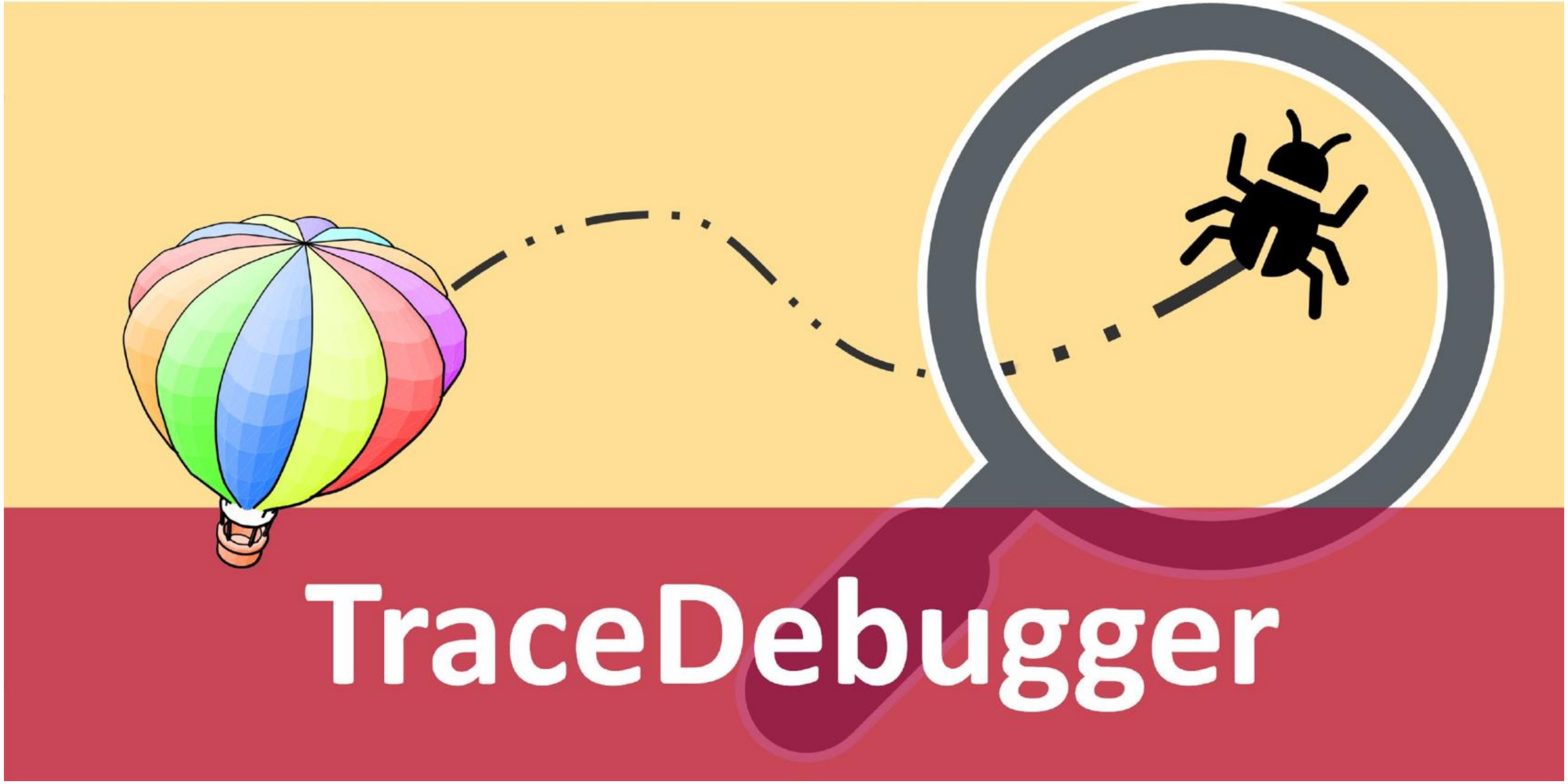
Implementierungsdetails

- Program Tracing mit Code Simulation
  - Methodenaufrufe mitschreiben
  - Seiteneffekte erkennen und alte Werte sichern
- Navigation durch Context Tree mittels Cursor
- Proxies für Zugriff auf historische Objekte
  - Code Simulation zur Emulation früherer Zustände
  - Vektorisierte Code Simulation für Object Traces im History Explorer

Einschränkungen

- Performance
  - Compiler/Decompiler-Aufruf: <1s
  - HTTPS-Anfrage: <10s
  - Toolbuilding: <5m
  - Komplexes Rendering: Minuten bis Stunden
- FFI/VM-Plugins/…
- Retracing von Objektidentitäten und Write Barriers (notYetImplemented)

Installation

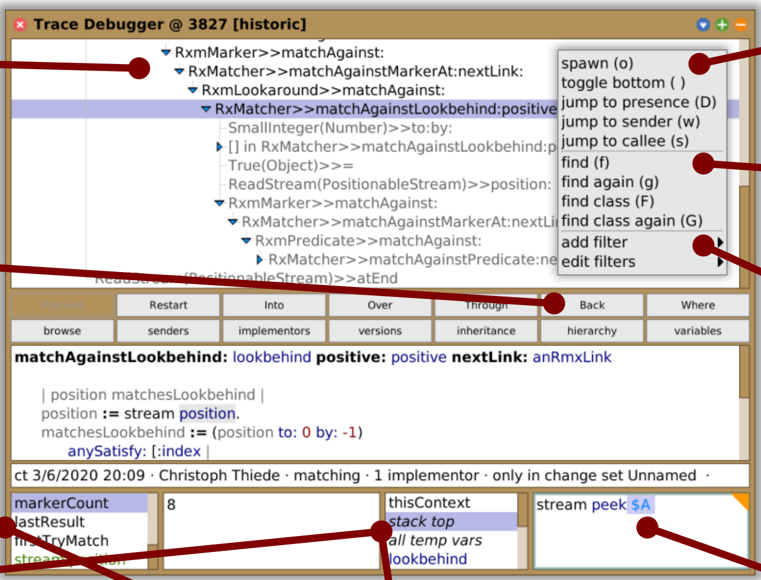


https://github.com/hpi-swa-lab/squeak-tracedebugger

mit fertigem Image zum Download

TraceDebugger

Record prior state and behavior (tracing)



```
RxmMarker>>matchAgainst:
  RxMatcher>>matchAgainstMarkerAt:nextLink:
    RxmLookaround>>matchAgainst:
      RxMatcher>>matchAgainstLookbehind:positive
        SmallInteger(Number)>>to:by:
        [] in RxMatcher>>matchAgainstLookbehind:p
        True(Object)>>=
        ReadStream(PositionableStream)>>position:
        RxmMarker>>matchAgainst:
          RxMatcher>>matchAgainstMarkerAt:nextLi
            RxmPredicate>>matchAgainst:
              RxMatcher>>matchAgainstPredicate:ne
ReadStream(PositionableStream)>>atEnd
```

```
matchAgainstLookbehind: lookbehind positive: positive nextLink: anRmxLink

    | position matchesLookbehind |
    position := stream position.
    matchesLookbehind := (position to: 0 by: -1)
        anySatisfy: [:index |
```

ct 3/6/2020 20:09 · Christoph Thiede · matching · 1 implementor · only in change set Unnamed ·

8

stream peek $A

Explore program trace

Replay program trace

Segmentable call tree (direct manipulation)

Queryable trace

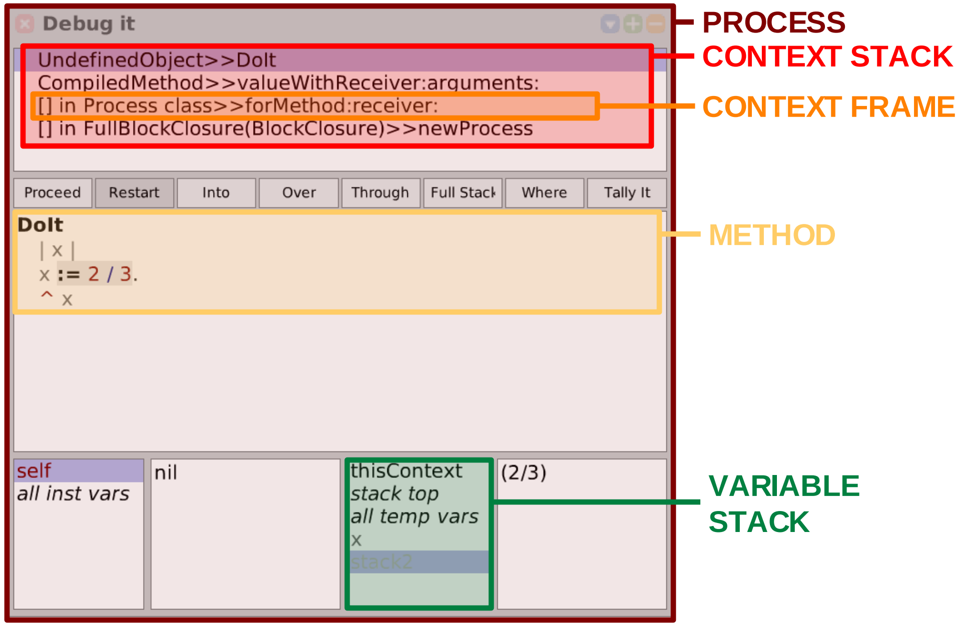
Interactive trace filters

Inspect historic state

Interact with historic state

Explore state changes

Code Simulation 101



Debug it

```
UndefinedObject>>DoIt
CompiledMethod>>valueWithReceiver:arguments:
[] in Process class>>forMethod:receiver:
[] in FullBlockClosure(BlockClosure)>>newProcess
```

```
DoIt
	| x |
	x := 2 / 3.
	^ x
```

self
all inst vars

nil

thisContext
stack top
all temp vars
x
stack2

(2/3)

PROCESS

CONTEXT STACK

CONTEXT FRAME

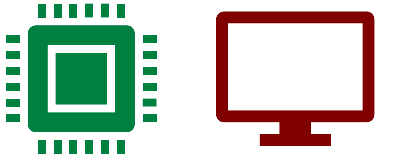
METHOD

VARIABLE STACK

BYTECODE

PC →

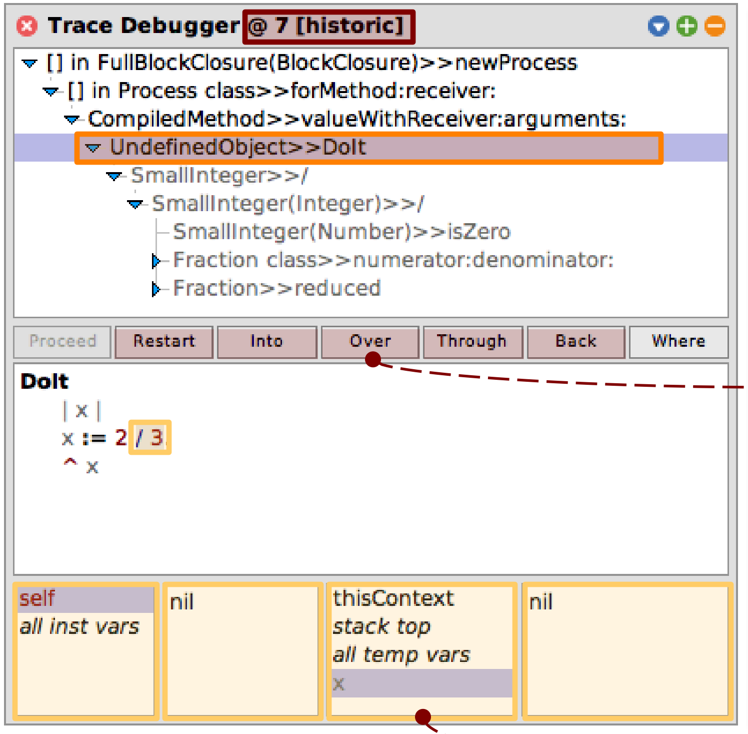
```
33 <E8 02>  pushConstant: 2
35 <E8 03>  pushConstant: 3
37 <60>     send: /
38 <D0>     popIntoTemp: 0
39 <40>     pushTemp: 0
40 <5C>     returnTop
```



Interpreter (VM-side)

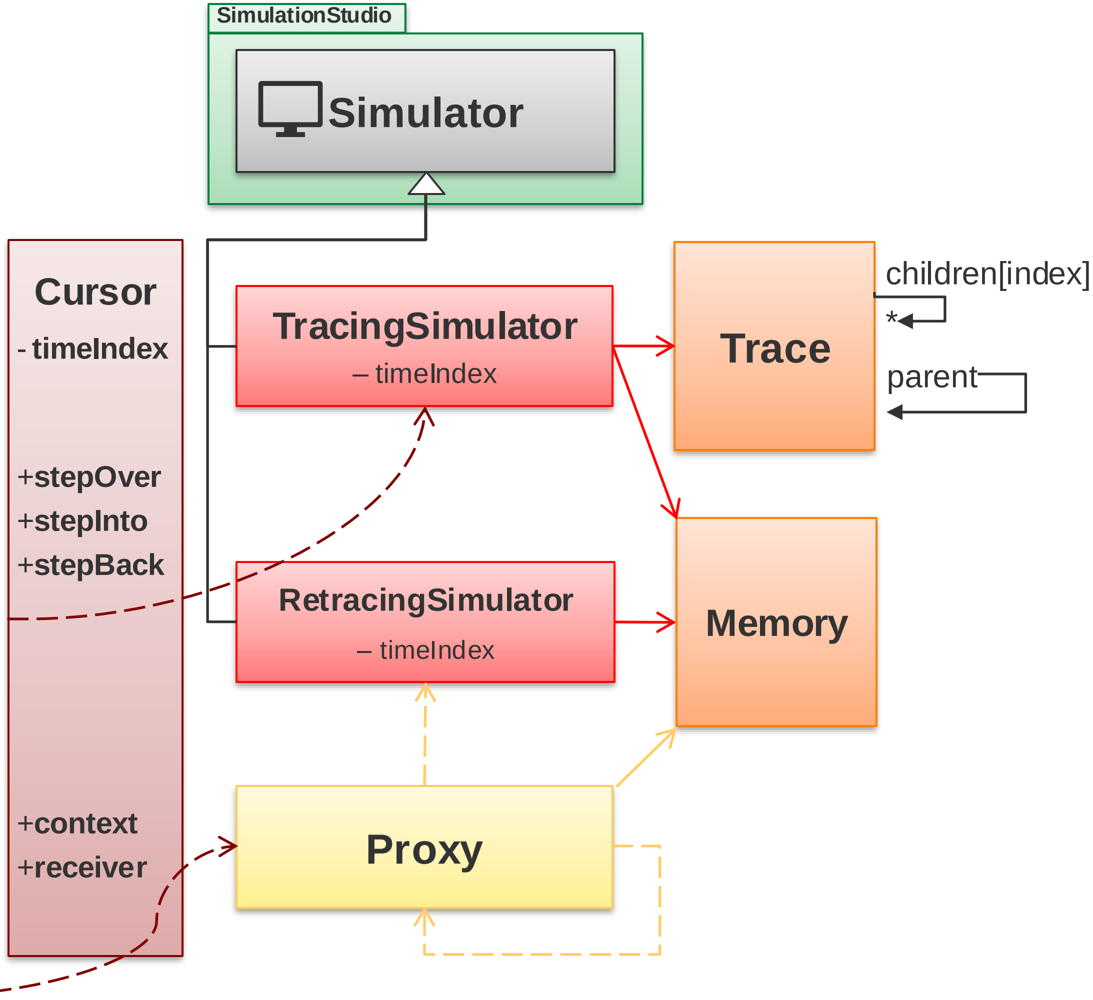
Simulator (image-side)

Architektur



```
[] in FullBlockClosure(BlockClosure)>>newProcess
  [] in Process class>>forMethod:receiver:
    CompiledMethod>>valueWithReceiver:arguments:
      UndefinedObject>>DoIt
        SmallInteger>>/
          SmallInteger(Integer)>>/
            SmallInteger(Number)>>isZero
            Fraction class>>numerator:denominator:
            Fraction>>reduced
```

```
DoIt
	| x |
	x := 2 / 3
	^ x
```



SimulationStudio

Simulator

Cursor

- timeIndex

+stepOver

+stepInto

+stepBack

+context

+receiver

TracingSimulator

– timeIndex

RetracingSimulator

– timeIndex

Proxy

Trace

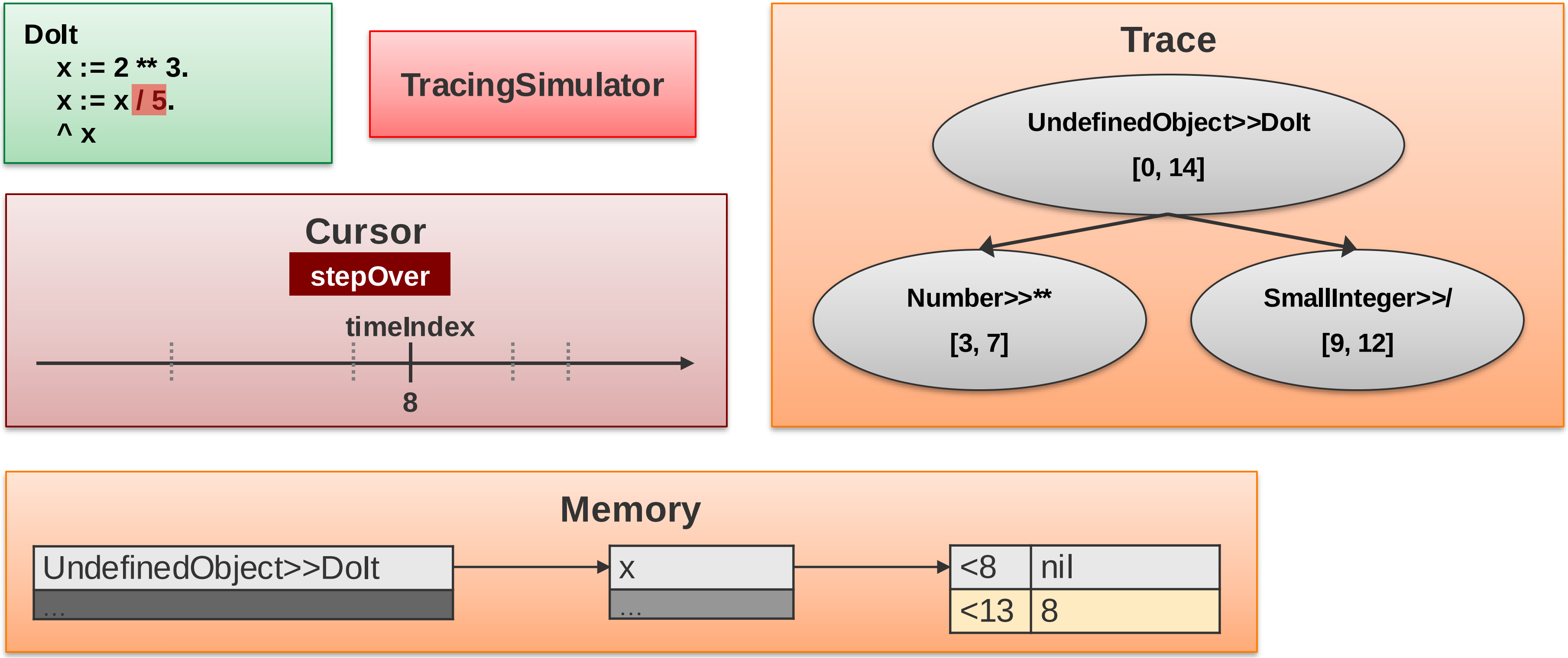
children[index]

*

parent

Memory

Architektur: Cursor-Navigation



```
DoIt
   x := 2 ** 3.
   x := x / 5.
   ^ x
```

TracingSimulator

Cursor

stepOver

timeIndex

8

Trace

UndefinedObject>>DoIt
[0, 14]

Number>>**
[3, 7]

SmallInteger>>/
[9, 12]

Memory

UndefinedObject>>DoIt
…

x
…

<8	nil
<13	8

Evaluation of Range Queries

- How can we access historic states?
  - Point retracing: Evaluate a query against a historic state

```
(self object: basket atTime: 14)
    numberOfItems
```

```
numberOfItems
| x |
x := oranges.
x := x + apples.
^ x
```

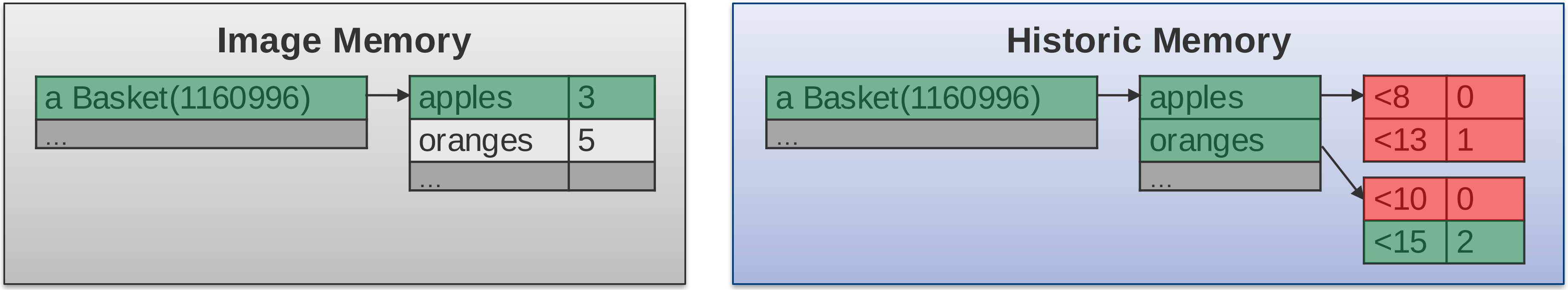


Image Memory

a Basket(1160996)
…

apples	3
oranges	5
…	

Historic Memory

a Basket(1160996)
…

apples
oranges
…

<8	0
<13	1

<10	0
<15	2

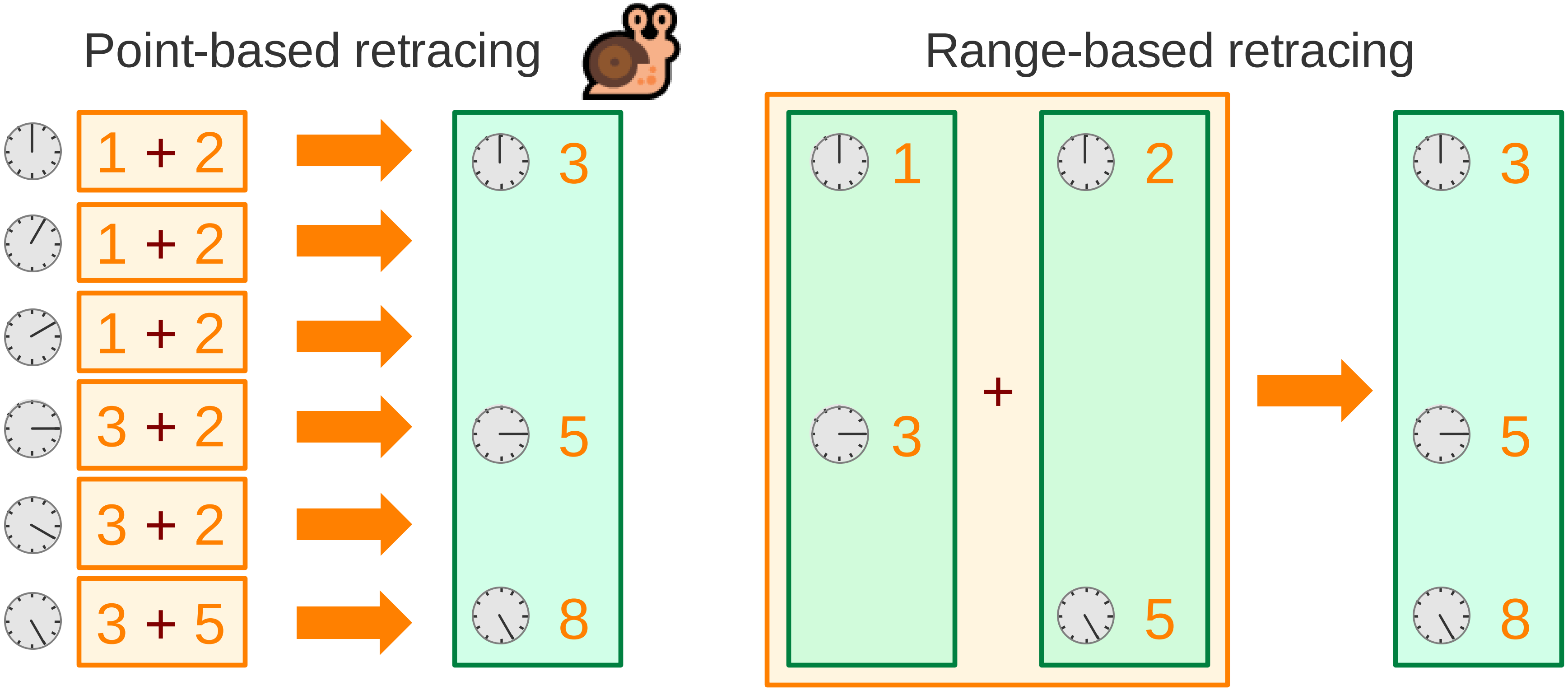
Evaluation of Range Queries

Program

```
basket := Basket new.
basket apples: 1.
basket oranges: 2.
basket apples: 3.
basket oranges: 5.
```

Query

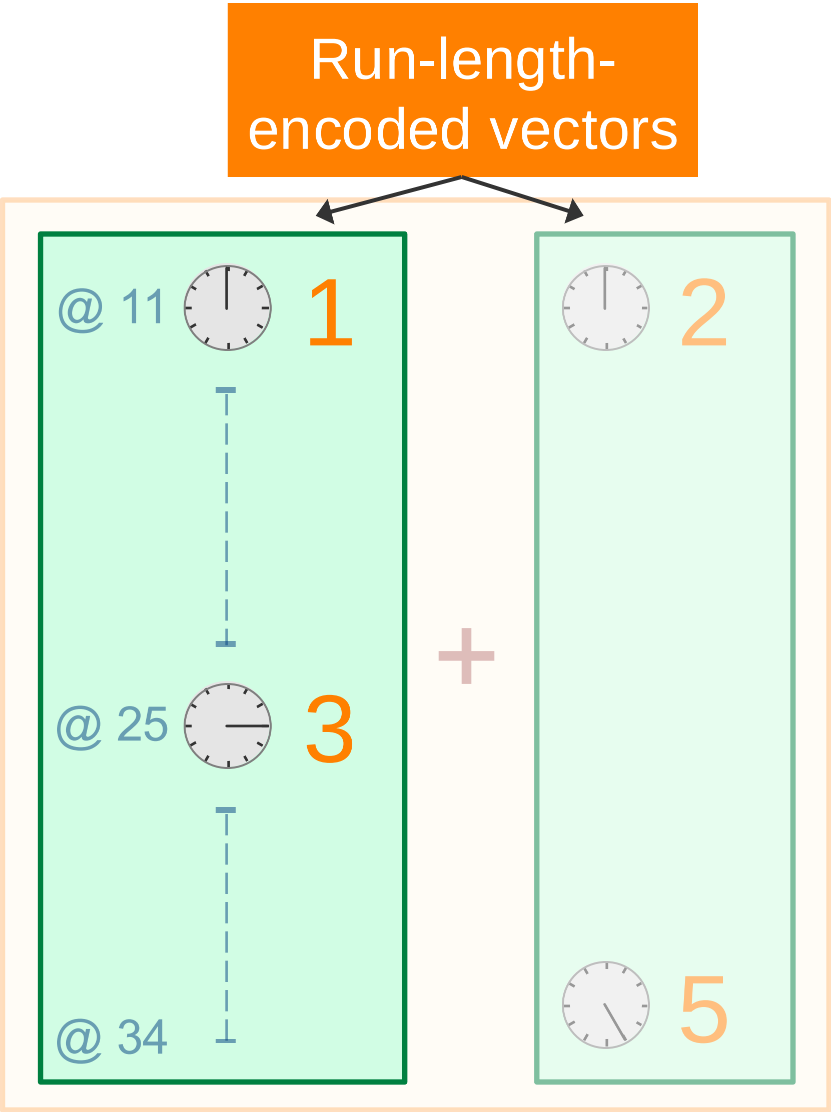
```
basket apples + basket oranges
```

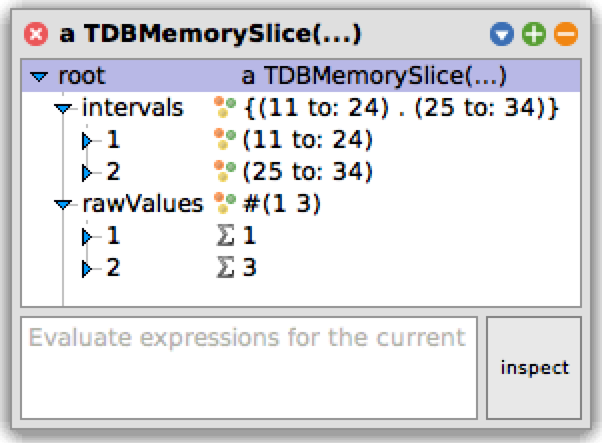


Point-based retracing

Range-based retracing

Evaluation of Range Queries


Run-length-encoded vectors


```
a TDBMemorySlice(...)
root            a TDBMemorySlice(...)
  intervals     {(11 to: 24) . (25 to: 34)}
    1           (11 to: 24)
    2           (25 to: 34)
  rawValues     #(1 3)
    1           Σ 1
    2           Σ 3
```

Evaluate expressions for the current

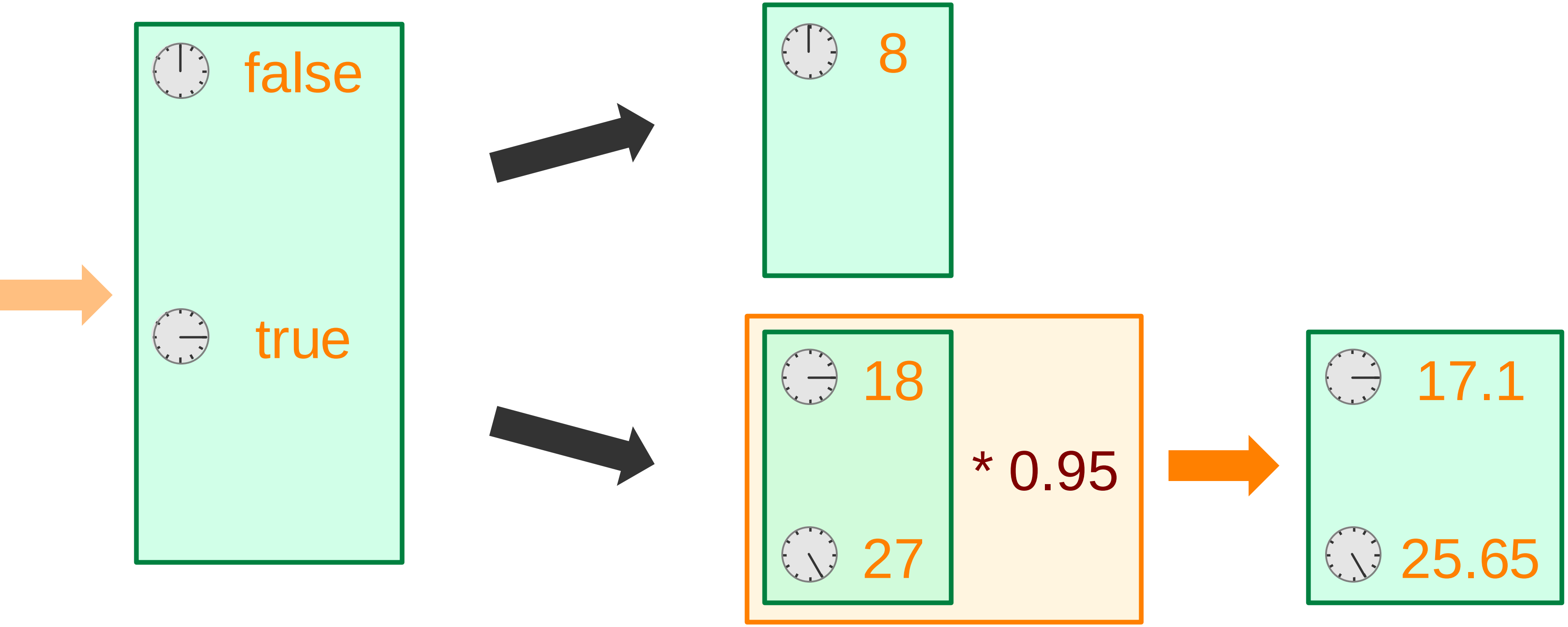
Range Retracing: Divergent Control Flow

Program

```
basket := Basket new.
basket apples: 1.
basket oranges: 2.
basket apples: 6.
basket oranges: 5.
```

Query

```
totalPrice
| price |
price := self apples * 2 + (self oranges * 3).
self apples > 3 ifTrue:
        [price := price * 0.95].
^ price
```

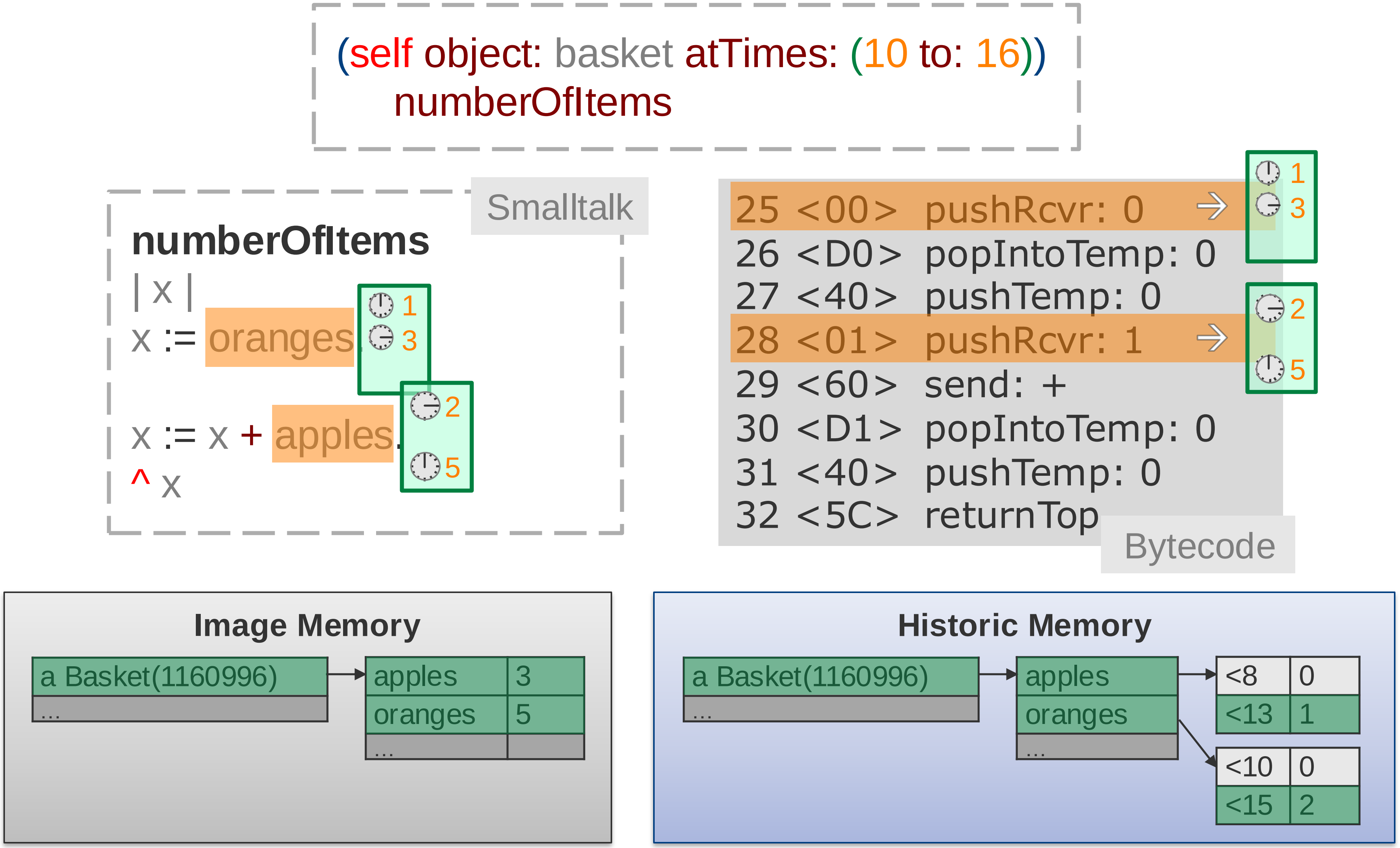


Range Retracing

- The idea:
  - Evaluate a query against a range of historic states simultaneously
  - Use data-parallel[1] execution to operate on sparse memory slices
  - Add vectorization/SIMD semantics to the interpreter of range queries
  - Fork process upon diverged control flow
  - Isolate side effects from different forks (stored in virtual memory per process)

[1] Not parallel on hardware level, just concurrent on interpreter level.

Range Retracing: Implementation in Squeak



```
(self object: basket atTimes: (10 to: 16))
    numberOfItems
```

Smalltalk

```
numberOfItems
| x |
x := oranges.
x := x + apples.
^ x
```

Bytecode

```
25 <00> pushRcvr: 0
26 <D0> popIntoTemp: 0
27 <40> pushTemp: 0
28 <01> pushRcvr: 1
29 <60> send: +
30 <D1> popIntoTemp: 0
31 <40> pushTemp: 0
32 <5C> returnTop
```

Image Memory

a Basket(1160996)	
…	

apples	3
oranges	5
…	

Historic Memory

a Basket(1160996)	
…	

apples	
oranges	
…	

<8	0
<13	1

<10	0
<15	2

Range Retracing: Implementation in Squeak

- Modified interpreter via code simulation
  - **Read instructions/primitives:** fetch and return vector from memory instead of current value
  - **Arithmetic/subscript/store management primitives:** apply SIMD semantics to transform vector receiver/arguments; if not implemented, fallback to fork for each vector combination
  - **Jump instructions:** fork for condition vector
  - **Message sends:** fork for lookup class vector
- https://github.com/LinqLover/SimulationStudio

Performance

Navigation in TraceDebugger*:

Workload	Steps	**Speed [ms/step into]** (smaller is better)					**RAM [kB]** (smaller is better)
		Tracing		**Retracing**		**Baseline**	
Small ^ 2 / 3	147	18.7	(–3.2%)	19.3	(+0%)	19.3	37.5
Medium ‘\w+’ asRegex	2205	27.8	(+28.1%)	26.2	(+20.7%)	21.7	509.0
Large ActiveWorld doOneCycleNow	63,072	45.4	(+102.7%)	39.1	(+74.6%)	22.4	14,574.8

Simulation in TraceDebugger*:

Workload		**Speed [ms]** (smaller is better)					
Tracing							
		Tracing		**Retracing**		**Baseline**	**Cog VM (JIT)**
	Factorial ^ 20000 factorial	2,459	(+392%)	654	(+31%)	500	105
	Regex ‘\w+’ asRegex	283	(+388%)	92	(+59%)	58	0
	Word cycles ActiveWorld doOneCycleNow	411.52	(+206%)	–		7.47	1.98
Retracing							
		Retracing (proxy)		**Retracing (always)**		**Baseline**	**Cog VM (JIT)**
	#bench[Proxy]ImageForm	3651	(+3350%)	22,368	(+38%)	16,229	109

*System: OSVM 202112201228/Win 21H1 | Intel i7-8550U CPU @ 1.80GHz