ARx: Reactive Programming for Synchronous Connectors

José Proença, Guillermina Cledou

Coordination @ DisCoTec 2020

INESCTEC

Universidade do Minho

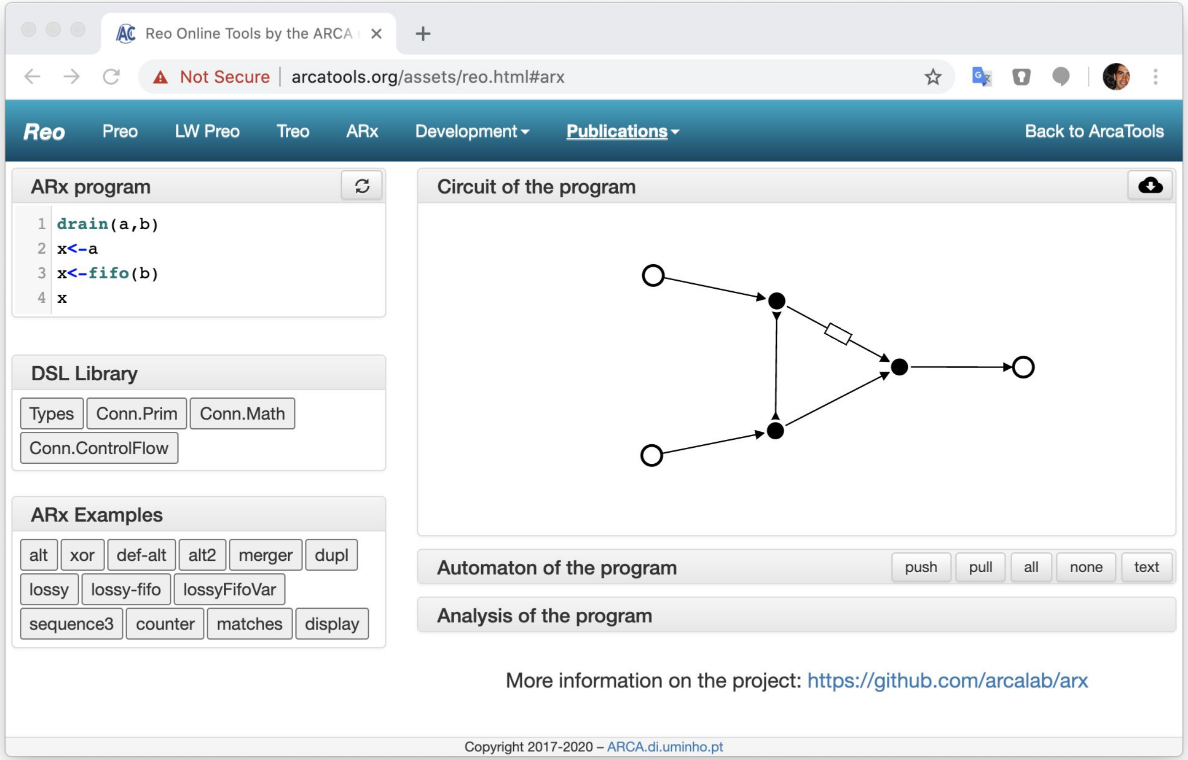
CISTER
Research Center in
Real-Time & Embedded
Computing Systems

isep Instituto Superior de Engenharia do Porto

Tools: http://arcatools.org/#arx

Video of the presentation: https://www.youtube.com/watch?v=74RUzfYneNI&t=2s

arcatools.org/#arx



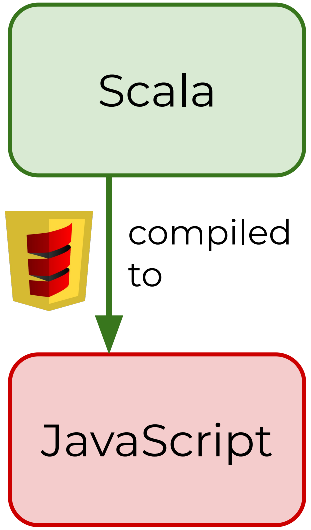
Reo Online Tools by the ARCA

Not Secure | arcatools.org/assets/reo.html#arx

```
drain(a,b)
x<-a
x<-fifo(b)
x
```

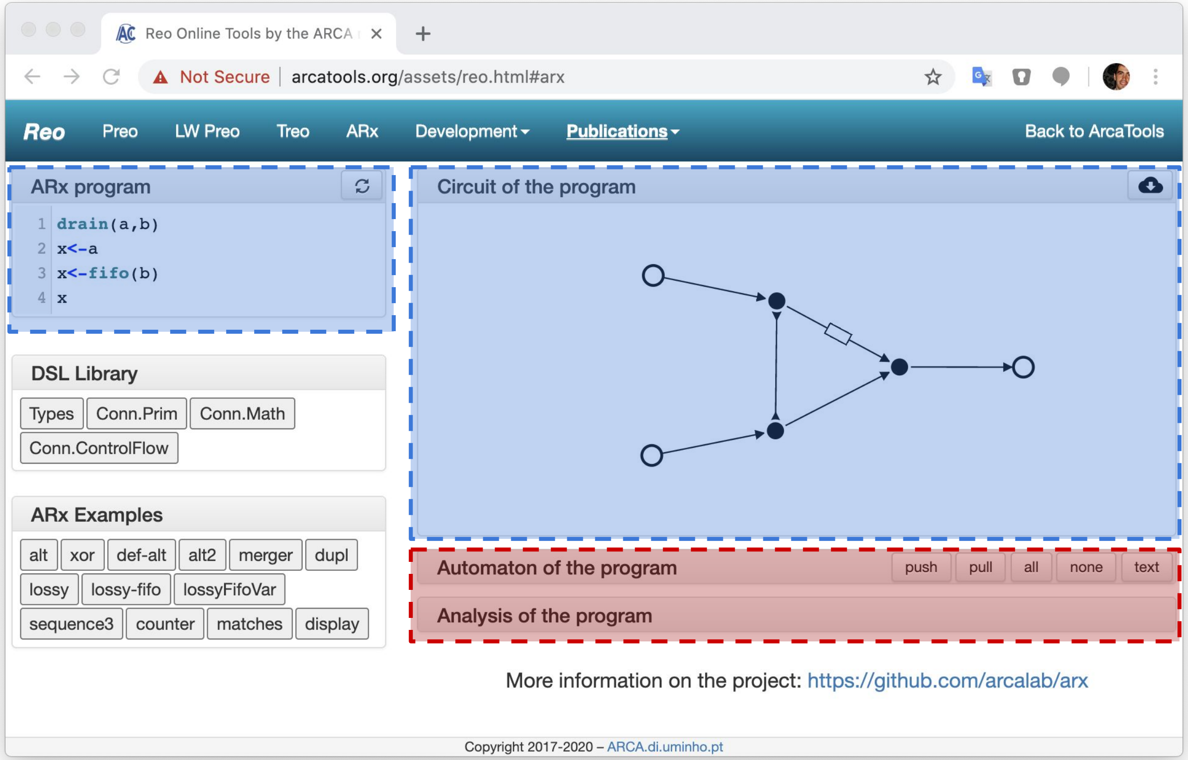
More information on the project: https://github.com/arcalab/arx

Copyright 2017-2020 – ARCA.di.uminho.pt



Tools: http://arcatools.org/#arx

arcatools.org/#arx


Reo Online Tools by the ARCA

Not Secure | arcatools.org/assets/reo.html#arx

Reo Preo LW Preo Treo ARx Development Publications Back to ArcaTools

ARx program

```
drain(a,b)
x<-a
x<-fifo(b)
x
```

Circuit of the program

DSL Library

Types | Conn.Prim | Conn.Math | Conn.ControlFlow

ARx Examples

alt | xor | def-alt | alt2 | merger | dupl | lossy | lossy-fifo | lossyFifoVar | sequence3 | counter | matches | display

Automaton of the program — push | pull | all | none | text

Analysis of the program

More information on the project: https://github.com/arcalab/arx

Copyright 2017-2020 – ARCA.di.uminho.pt

Syntax
- Constructs
- Layout

Motivation
- Reactive Languages
- Synchronous Languages

Semantics
- Stream Builders
- Reactive Interpretation

Tools: http://arcatools.org/#arx

Syntax

Reo-based constructs

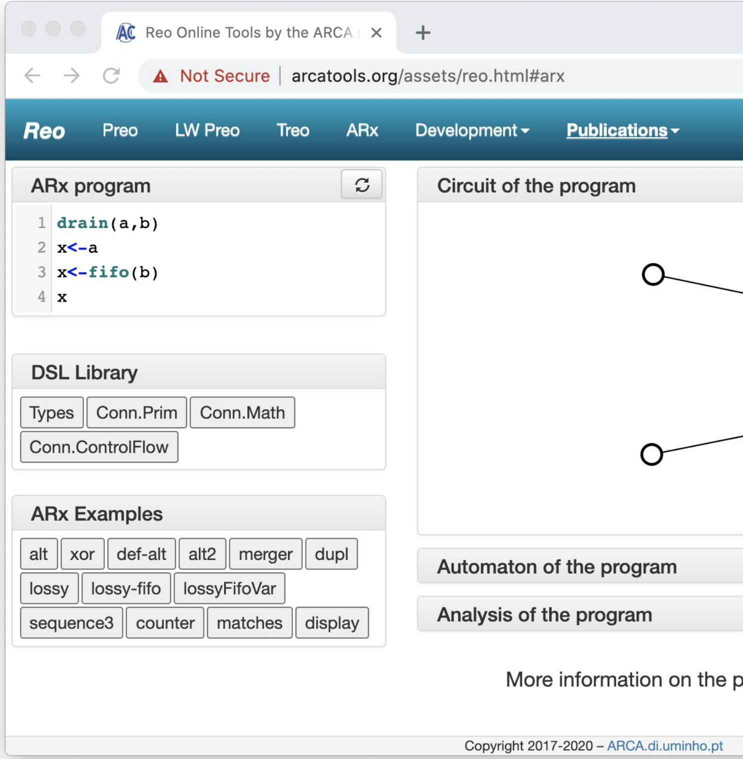
```
lossy(a)
fifo(a)
drain(a,b)

c <- a
c <- b

b <- a
c <- a
```

Reactive variables

```
a <~ b
```

Algebraic data types

```
Data Bool =
  True | False
```

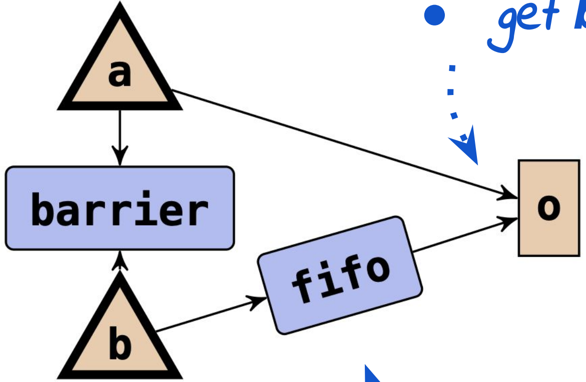


```
drain(a,b)
x<-a
x<-fifo(b)
x
```

Tools: http://arcatools.org/#arx

Synchronous connectors

(E.g., **Reo**, Lustre, Esterel)



Either
- get **a**
- get **b**

a and **b** at the same time

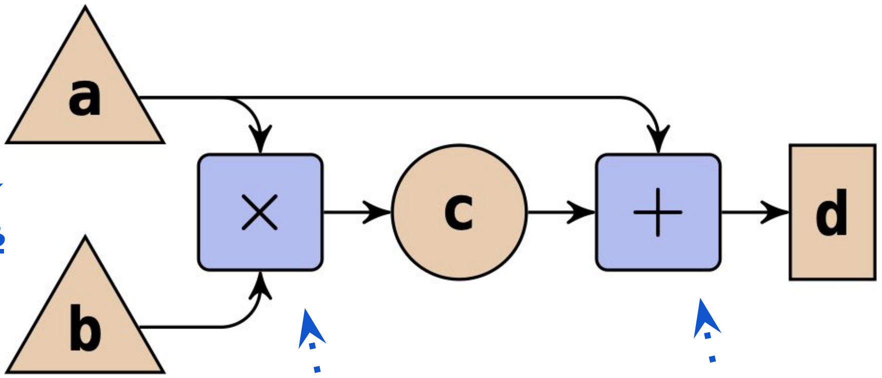
1. get **b**
2. block **b** until empty
3. sent to **o**

```
drain(a,b)
o <- a
o <- fifo(b)
o
```

Tools: http://arcatools.org/#arx

Reactive Programs (1)

(E.g., Angular, Yampa, ReScala)



Reactive variables

Triggered if *b* is updated

Triggered if both *a* and *c* are updated

```
...
c <- Times(a,b)
Plus(a,c)
```

From: E. Bainomugisha, A.L. Carreton, T. van Cutsem, S. Mostinckx, and W. de Meuter. **A survey on reactive programming**. ACM Comput. Surv., 2013.

Tools: http://arcatools.org/#arx

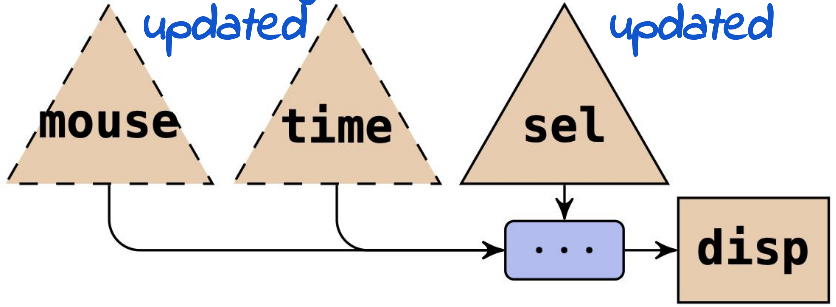
Reactive Programs (2)

Domain: Graphical users interfaces

(E.g., Angular, Yampa, ReScala)

constantly updated

sometimes updated



Dynamic dependencies

May request updated values

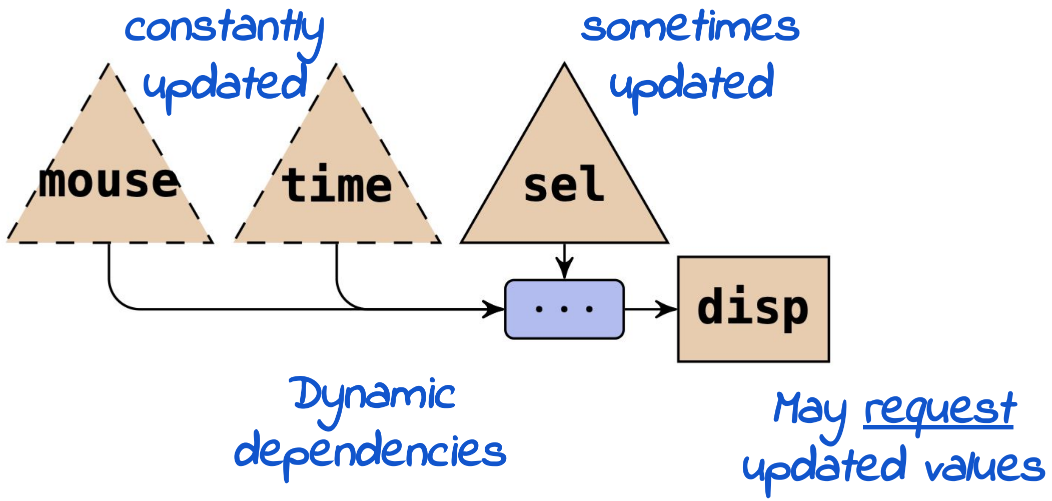
```
if sel then mouse else time
```

Tools: http://arcatools.org/#arx

Reactive Programs (2)

Domain: Graphical users interfaces

(E.g., Angular, Yampa, ReScala)



```
selRx <~ sel
true,false <-
    match(selRx)
...
```

```
if sel then mouse else time
```

Tools: http://arcatools.org/#arx

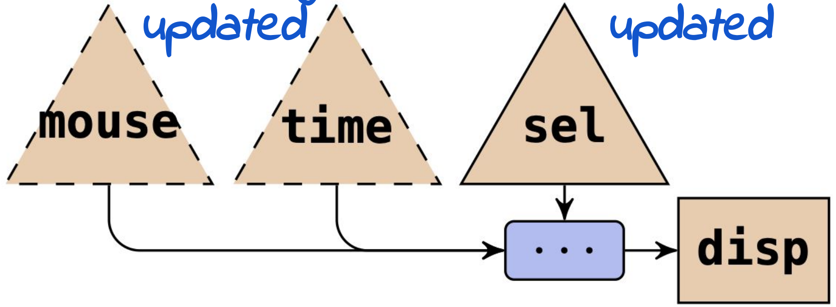
Reactive Programs (2)

Domain: Graphical users interfaces

(E.g., Angular, Yampa, ReScala)

constantly updated

sometimes updated



Dynamic dependencies

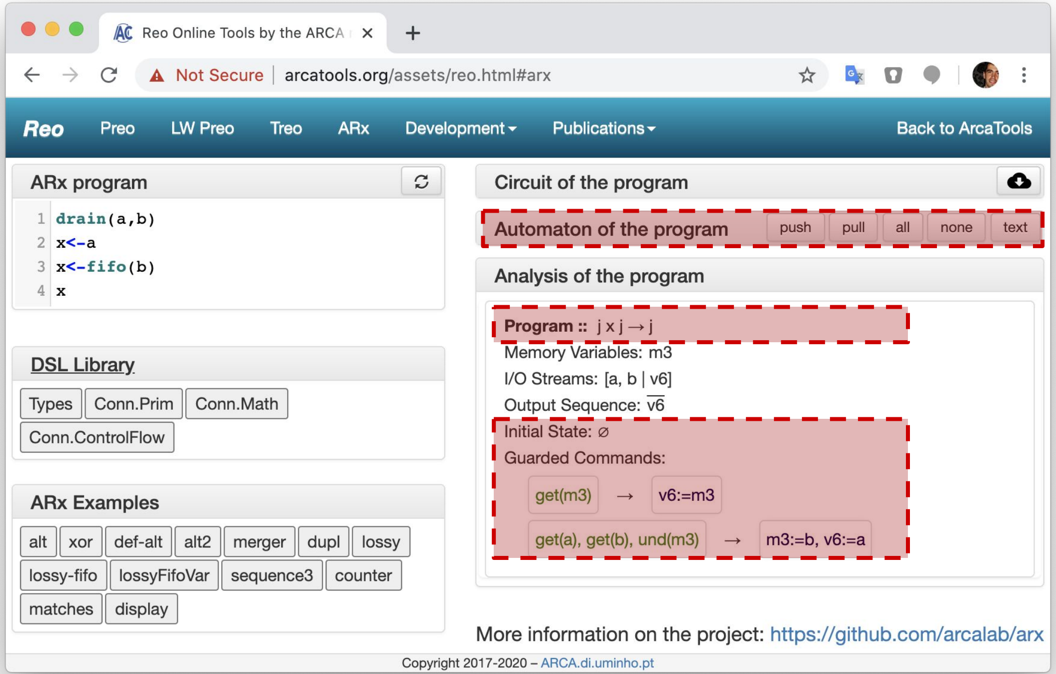
May request updated values

```
data Bool =
    True | False

selRx <~ sel
true,false <-
     match(selRx)
...
```

```
if sel then mouse else time
```

Tools: http://arcatools.org/#arx

Semantics



Reo Online Tools by the ARCA

Not Secure | arcatools.org/assets/reo.html#arx

Reo | Preo | LW Preo | Treo | ARx | Development | Publications | Back to ArcaTools

ARx program

```
drain(a,b)
x<-a
x<-fifo(b)
x
```

DSL Library

Types | Conn.Prim | Conn.Math | Conn.ControlFlow

ARx Examples

alt | xor | def-alt | alt2 | merger | dupl | lossy | lossy-fifo | lossyFifoVar | sequence3 | counter | matches | display

Circuit of the program

Automaton of the program | push | pull | all | none | text

Analysis of the program

Program :: j x j → j
Memory Variables: m3
I/O Streams: [a, b | v6]
Output Sequence: $\overline{\text{v6}}$
Initial State: ∅
Guarded Commands:
get(m3) → v6:=m3
get(a), get(b), und(m3) → m3:=b, v6:=a

More information on the project: https://github.com/arcalab/arx

Copyright 2017-2020 – ARCA.di.uminho.pt

Automata semantics of stream builders

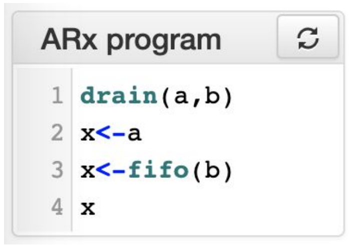
Types

ARx Semantics via **stream builders**

Tools: *http://arcatools.org/#arx*

Stream Builder

Semantics with **composition**



ARx program

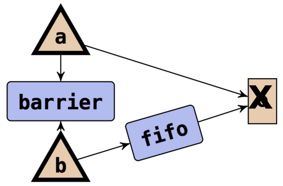
```
drain(a,b)
x<-a
x<-fifo(b)
x
```

Initial State: ∅

Guarded Commands:

get(m3)	→	v6:=m3
get(a), get(b), und(m3)	→	m3:=b, v6:=a

Set of exclusive guarded commands

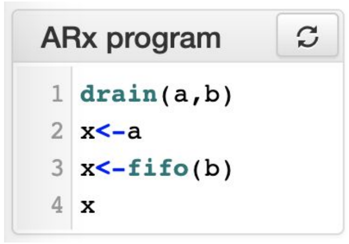


guards how to consume input/memory streams

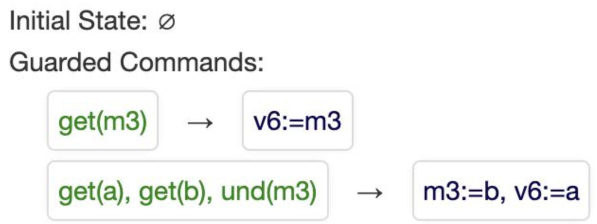
updates how to write output/memory streams

Tools: http://arcatools.org/#arx

Stream Builder



```
drain(a,b)
x<-a
x<-fifo(b)
x
```



Initial State: ∅

Guarded Commands:

get(m3) → v6:=m3

get(a), get(b), und(m3) → m3:=b, v6:=a

get destructive read

und undefined value

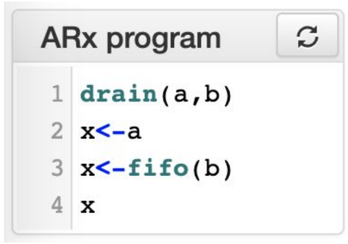
…

Tools: http://arcatools.org/#arx

Stream Builder

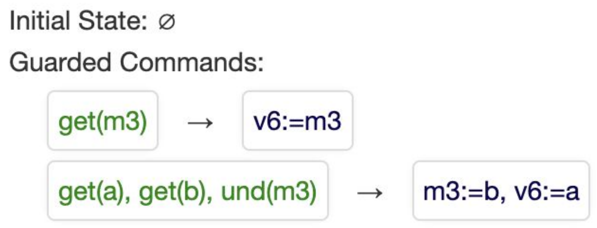
Compact representation

(no state explosion)



ARx program

```
drain(a,b)
x<-a
x<-fifo(b)
x
```



Initial State: ∅

Guarded Commands:

get(m3) → v6:=m3

get(a), get(b), und(m3) → m3:=b, v6:=a

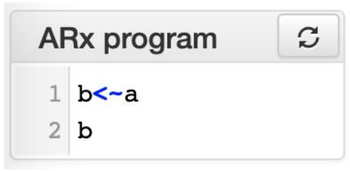
Based on stream constraints

[Dokter and Arbab '18]

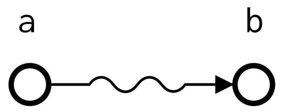
Stream Builder

Reactiveness

latest value **always available**


ARx program

```
b<~a
b
```



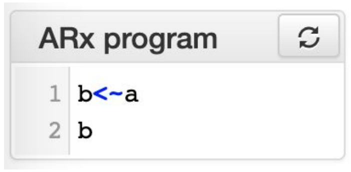
get(a) → m0:=a

ask(m0) → v1:=m0

ask non-destructive read

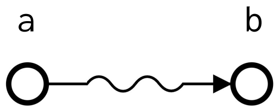
Tools: http://arcatools.org/#arx

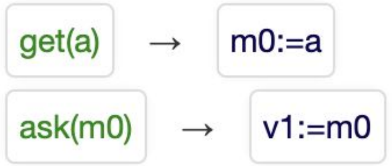
Examples



ARx program

```
b<~a
b
```

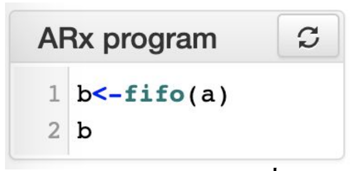




get(a) → m0:=a

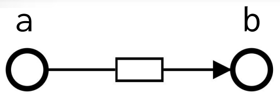
ask(m0) → v1:=m0

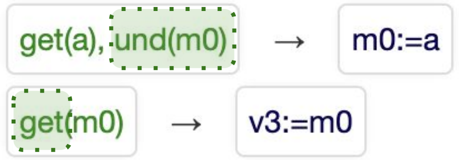
b always available
m0 overwritten



ARx program

```
b<-fifo(a)
b
```





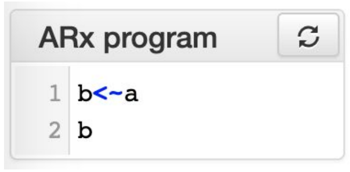
get(a), und(m0) → m0:=a

get(m0) → v3:=m0

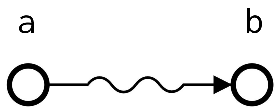
b only once
m0 NOT overwritten

Tools: http://arcatools.org/#arx

Reactive Semantics


ARx program

```
b<~a
b
```



~~Triggered if~~
~~*a* is updated~~

Push-pull interpretation

Triggered if
a is **updated**
or if
a is **active**
and the **environment** wants to read ***b***

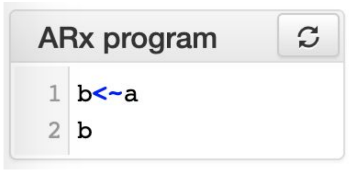
The **environment** controls when to read (pull) / write (push)

Tools: http://arcatools.org/#arx

Reactive Semantics

Push-pull interpretation

via stream builder automata

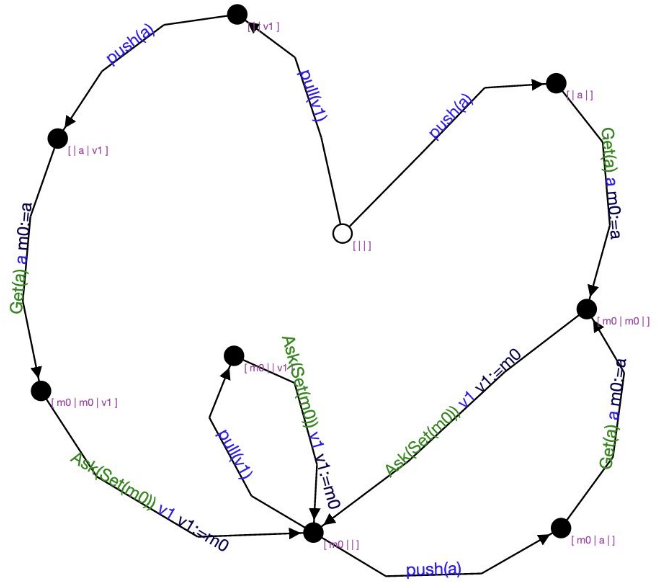


ARx program

```
b<~a
b
```

get(a) → m0:=a

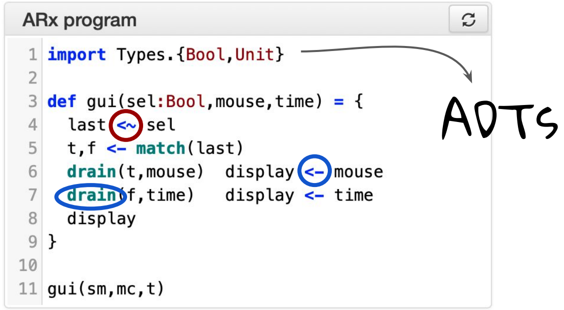
ask(m0) → v1:=m0



Tools: http://arcatools.org/#arx

Wrap up

Synchronous +
Reactive DSL



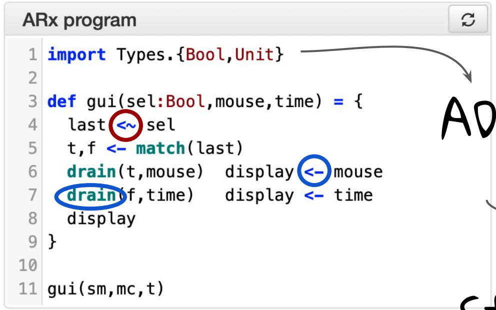
```
import Types.{Bool,Unit}

def gui(sel:Bool,mouse,time) = {
  last <~ sel
  t,f <- match(last)
  drain(t,mouse)  display <- mouse
  drain(f,time)   display <- time
  display
}

gui(sm,mc,t)
```

Scala +
Javascript

Tools: http://arcatools.org/#arx

Wrap up

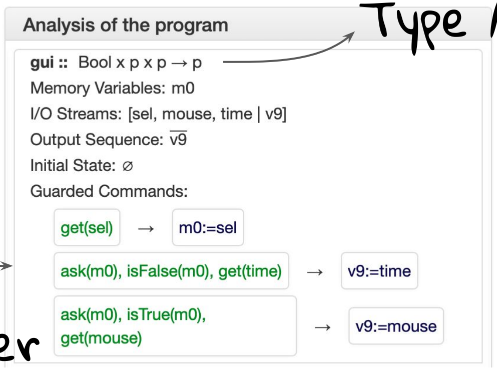
Synchronous +
Reactive DSL


ARx program

```
import Types.{Bool,Unit}

def gui(sel:Bool,mouse,time) = {
  last <~ sel
  t,f <- match(last)
  drain(t,mouse)  display <- mouse
  drain(f,time)   display <- time
  display
}

gui(sm,mc,t)
```

ADTs

Stream builder
Semantics

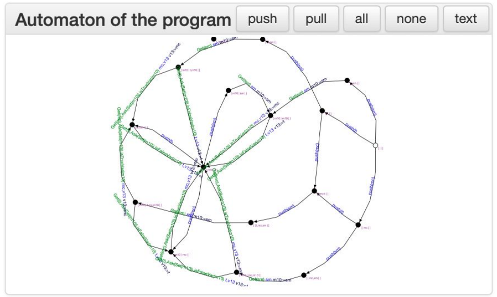

Analysis of the program

gui :: Bool x p x p → p
Memory Variables: m0
I/O Streams: [sel, mouse, time | v9]
Output Sequence: $\overline{v9}$
Initial State: ∅
Guarded Commands:

get(sel) → m0:=sel
ask(m0), isFalse(m0), get(time) → v9:=time
ask(m0), isTrue(m0), get(mouse) → v9:=mouse

Type Analysis

Scala +
Javascript

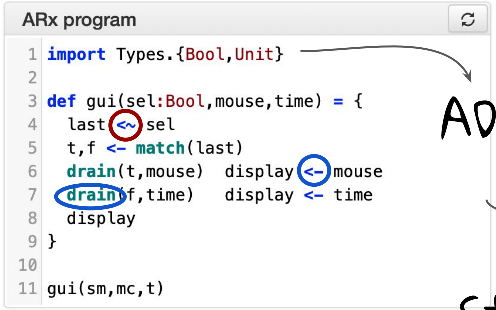

Automaton of the program

Reactive Semantics
for SB
(with push-pull
interpretation)

Tools: http://arcatools.org/#arx

Wrap up

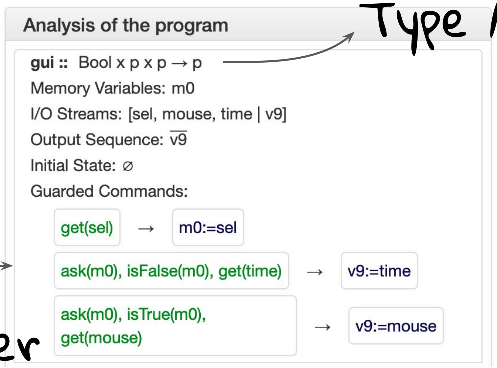
Synchronous + Reactive DSL


ARx program

```
import Types.{Bool,Unit}

def gui(sel:Bool,mouse,time) = {
  last <~ sel
  t,f <- match(last)
  drain(t,mouse)  display <- mouse
  drain(f,time)   display <- time
  display
}

gui(sm,mc,t)
```

ADTs

Stream builder Semantics

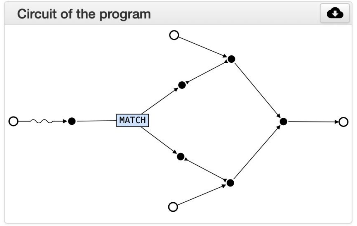

Analysis of the program

gui :: Bool x p x p → p
Memory Variables: m0
I/O Streams: [sel, mouse, time | v9]
Output Sequence: $\overline{\text{v9}}$
Initial State: ∅
Guarded Commands:

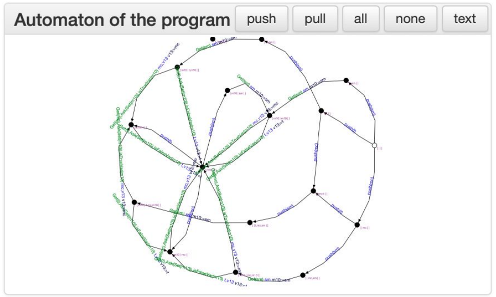
- get(sel) → m0:=sel
- ask(m0), isFalse(m0), get(time) → v9:=time
- ask(m0), isTrue(m0), get(mouse) → v9:=mouse

Type Analysis

Scala + Javascript


Circuit of the program

Architectural view


Automaton of the program

Reactive Semantics for SB (with push-pull interpretation)

Tools: http://arcatools.org/#arx