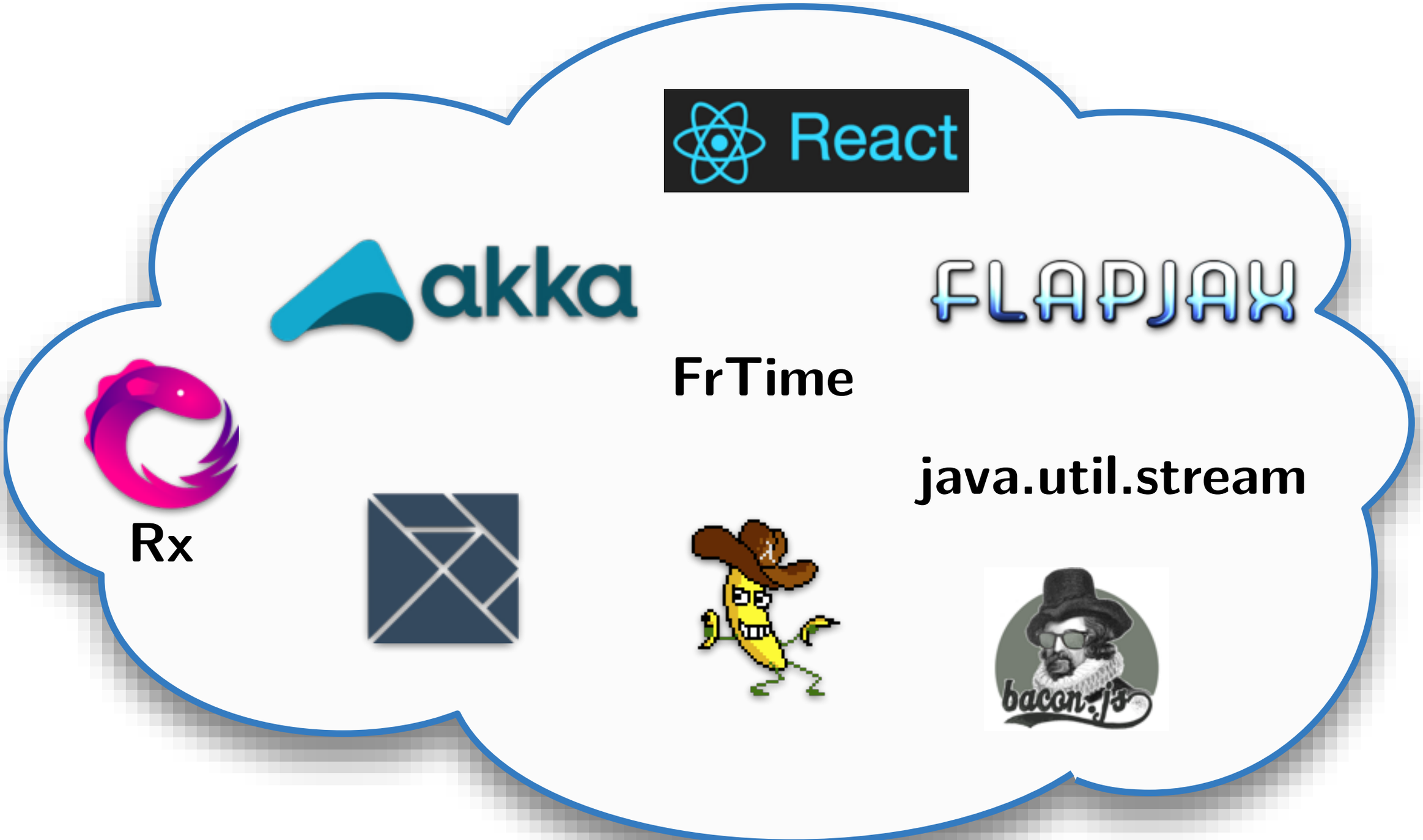
Temporal Correlation Patterns

Intersecting Joins, Streams, Events and Reactive Programming

Oliver Bračevac
Software Technology Group (Mira Mezini)
TU Darmstadt, Germany
bracevac@cs.tu-darmstadt.de

REBLS' 15
10/27/15 - Pittsburgh, PA, USA

Reactive Programming?



Correlate |ˈkɒrəleɪt, -rɪ-|

Correlate |ˈkɒrəleɪt, -rɪ-|

- Have/establish a mutual relationship or connection, in which one thing affects or depends on another.

Correlate |ˈkɒrəleɪt, -rɪ-|

- Have/establish a mutual relationship or connection, in which one thing affects or depends on another.

- "*We should* ***correlate*** *general trends in public opinion* ***with*** *trends in the content of television news*."

(Oxford Dictionary of English)

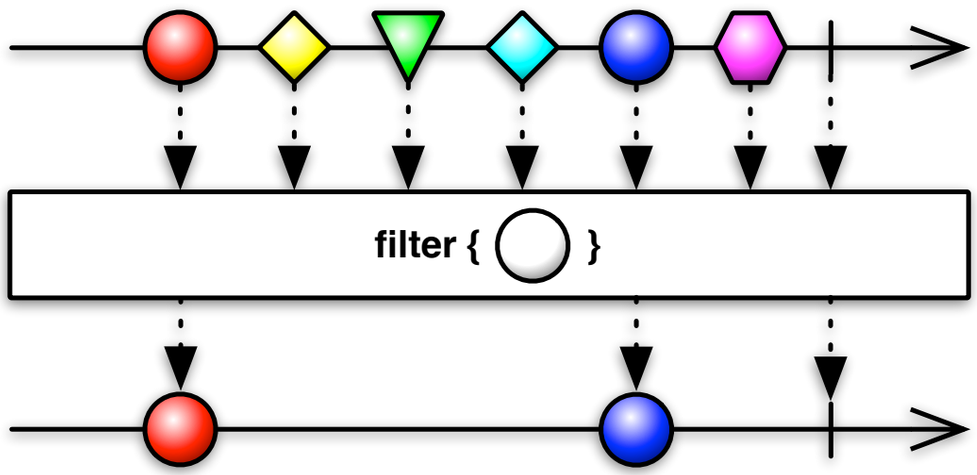
Correlate |ˈkɒrəleɪt, -rɪ-|

- Have/est nnection, in which on ther.
- *"We sho blic opinion* ***with*** *tre ws."*

Reactive languages correlate data!

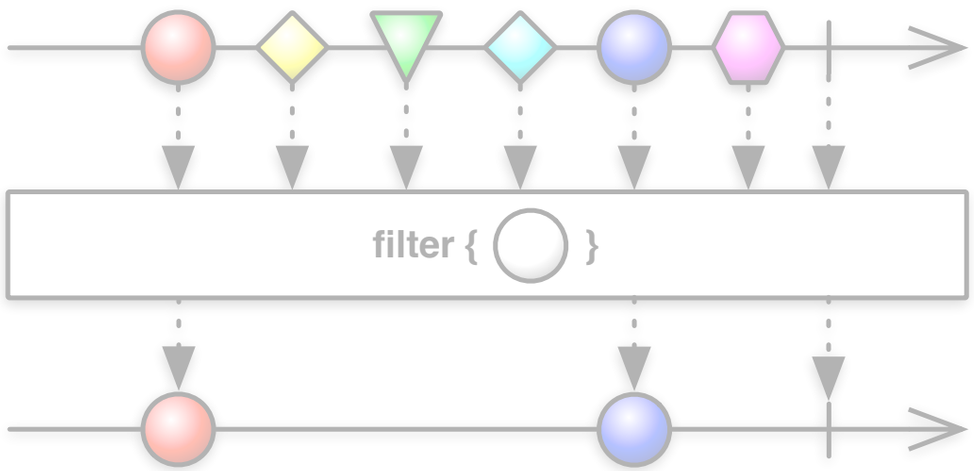
(Oxford Dictionary of English)

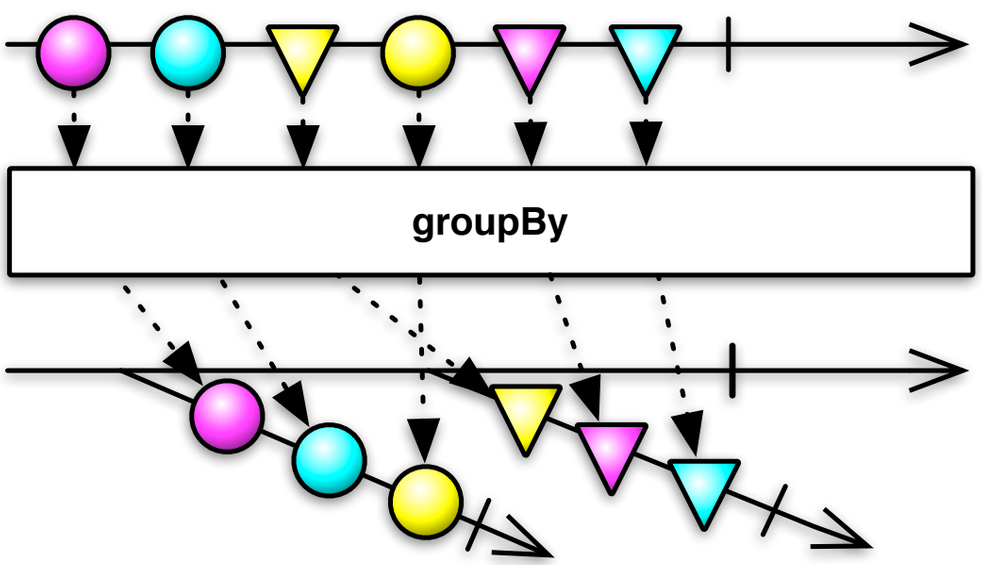
Data Correlation in RP



http://reactivex.io

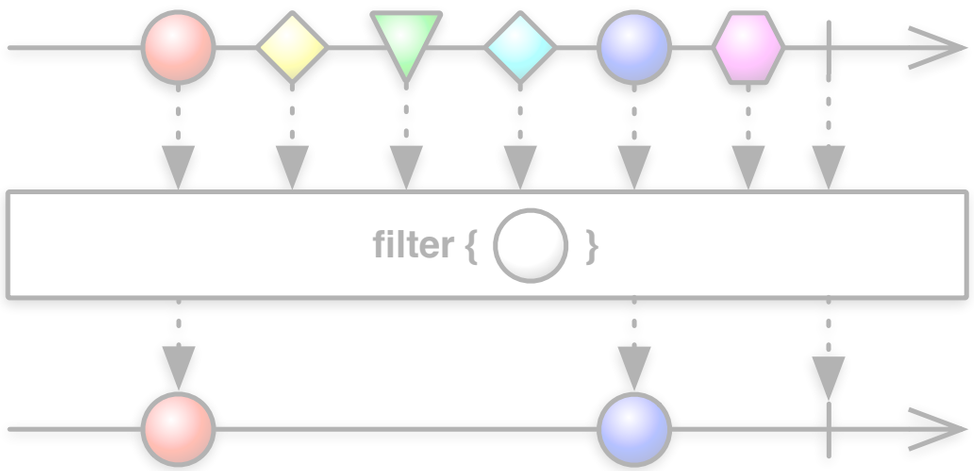
Data Correlation in RP

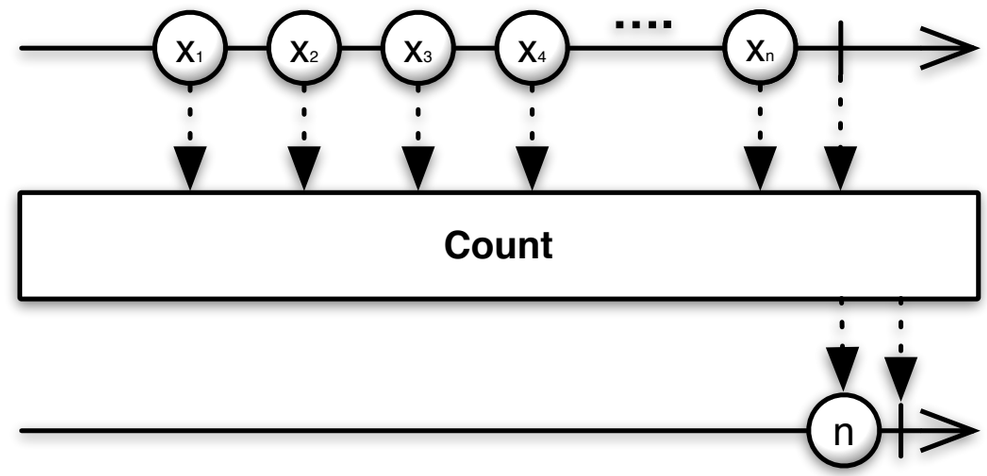


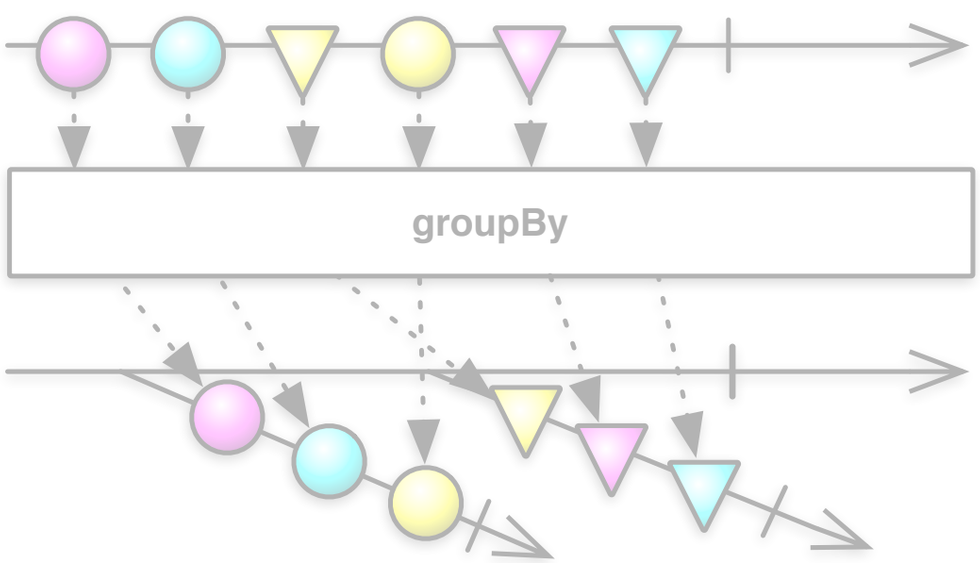


http://reactivex.io

Data Correlation in RP

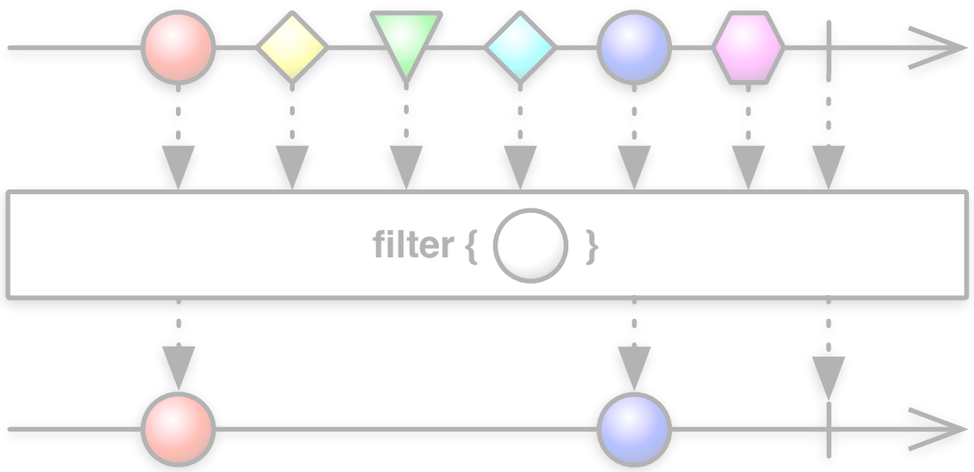


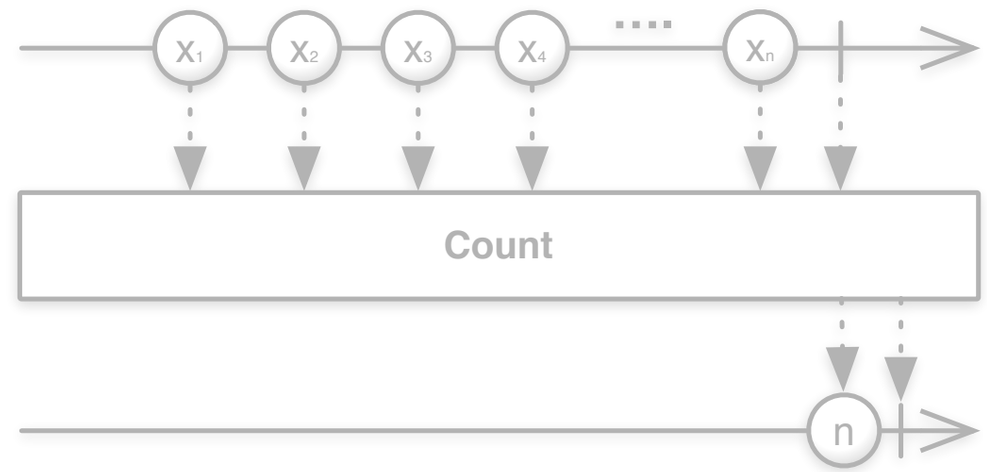


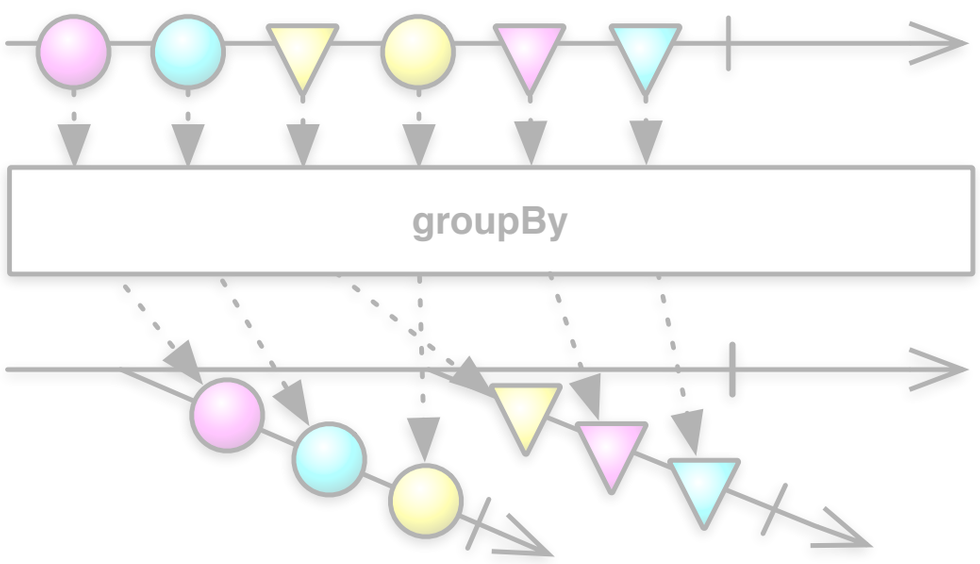


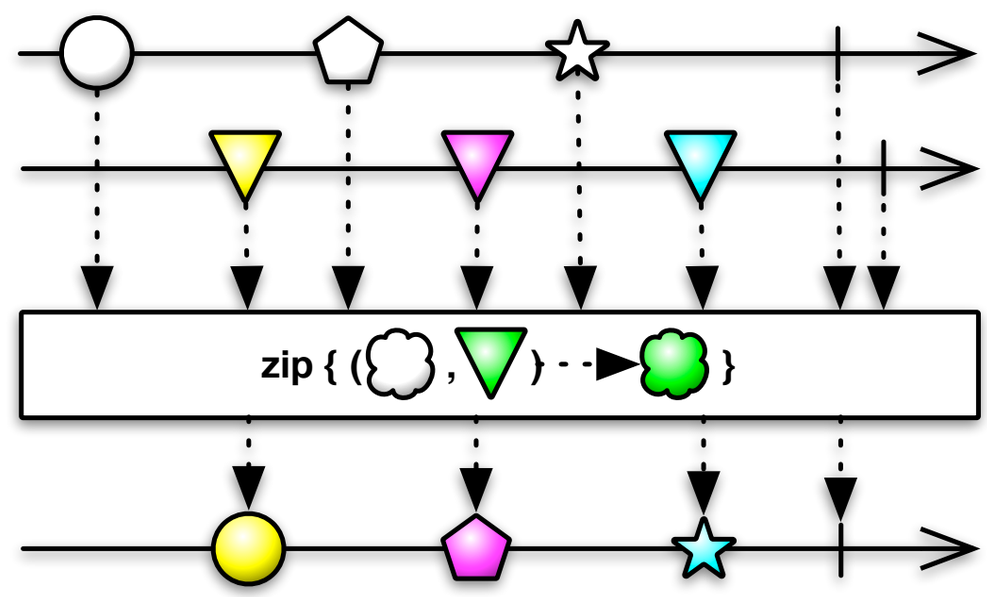
http://reactivex.io

Data Correlation in RP



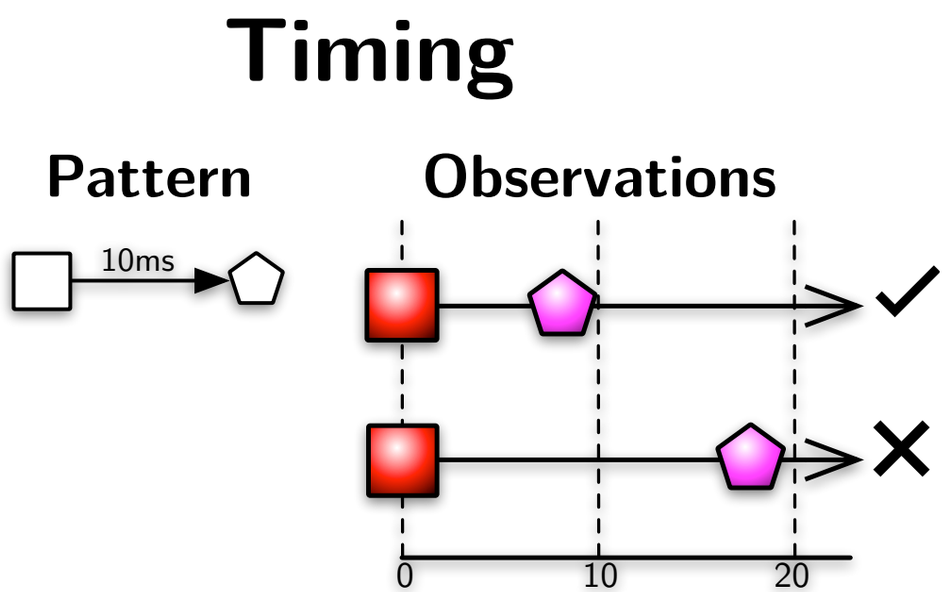






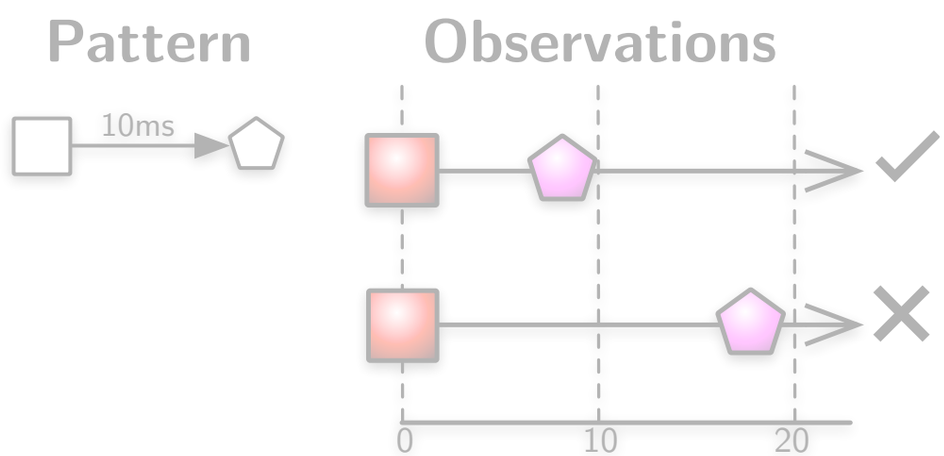
http://reactivex.io

Where RP Fails Fast:

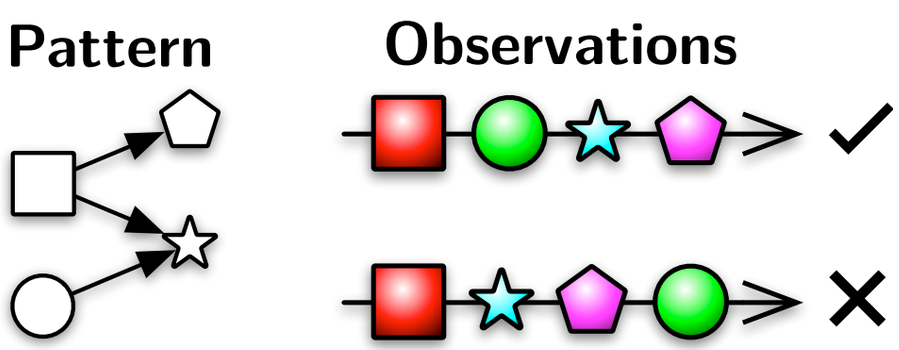


Where RP Fails Fast:

Timing



Partial orders

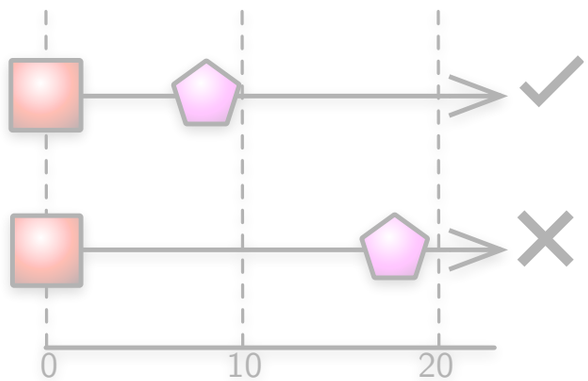


Where RP Fails Fast:

Timing

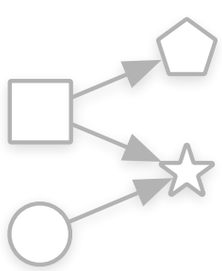
Pattern

Observations

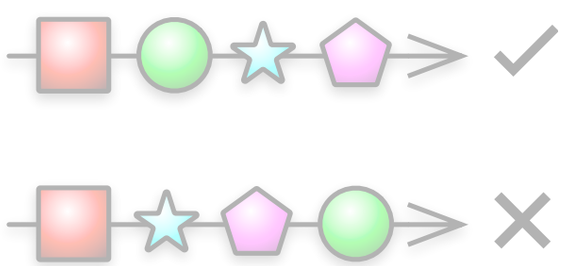


Partial orders

Pattern

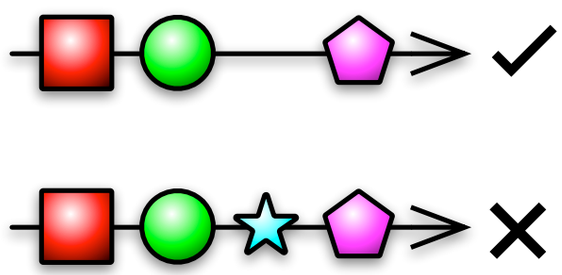


Observations



Absence/negation

Observations

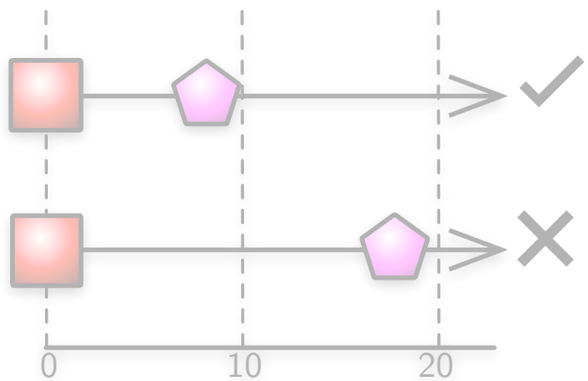


Where RP Fails Fast:

Timing

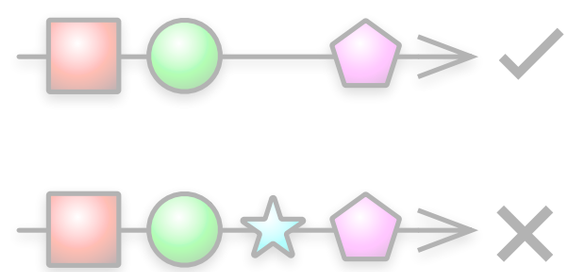
Pattern

Observations



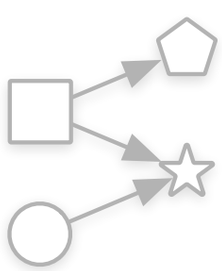
Absence/negation

Observations

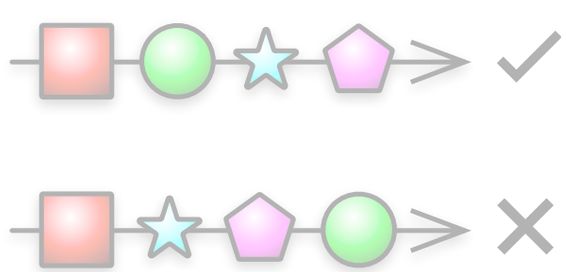


Partial orders

Pattern

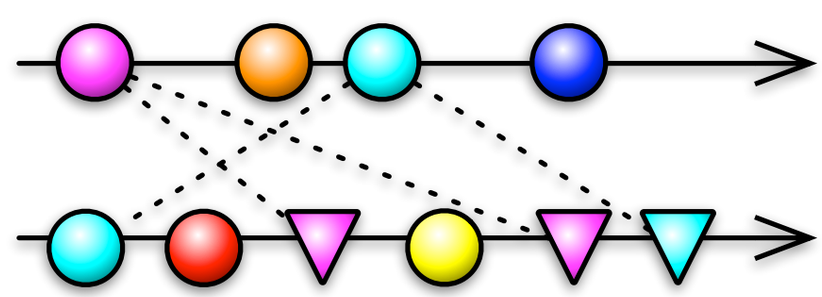


Observations



"Criss-crossing"

Observations

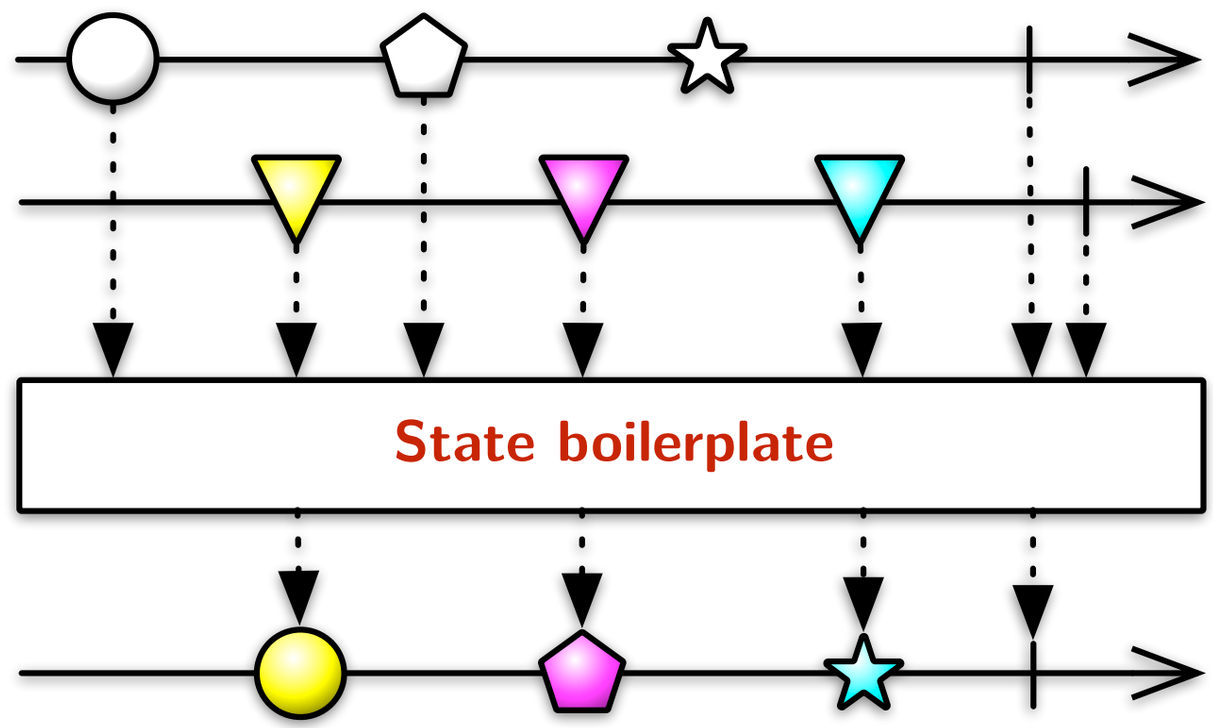


Where RP Fails Fast:

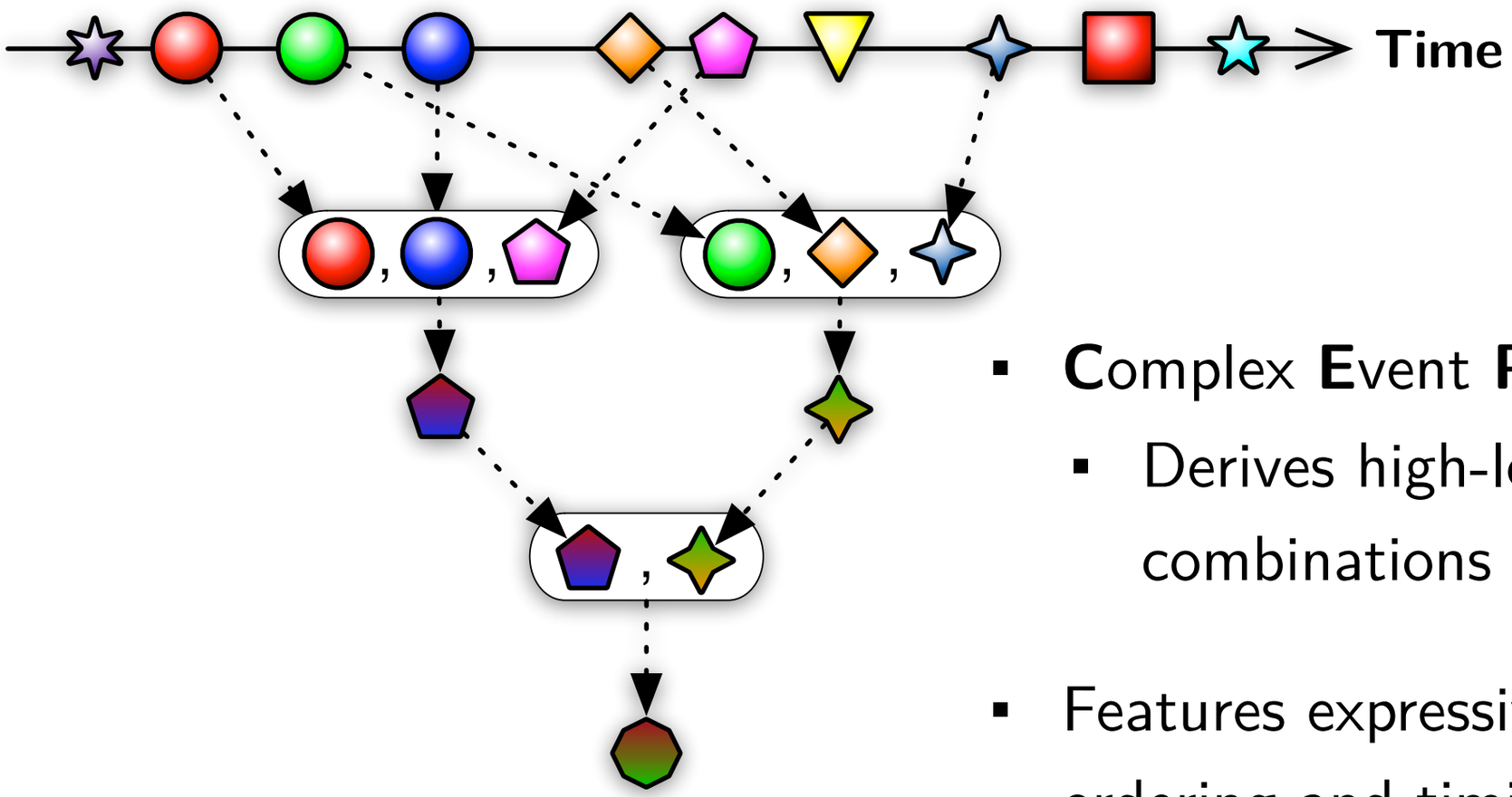
Timing

Absence/negation

- No exposure of arrival orders or timings!
- We need a pattern language!

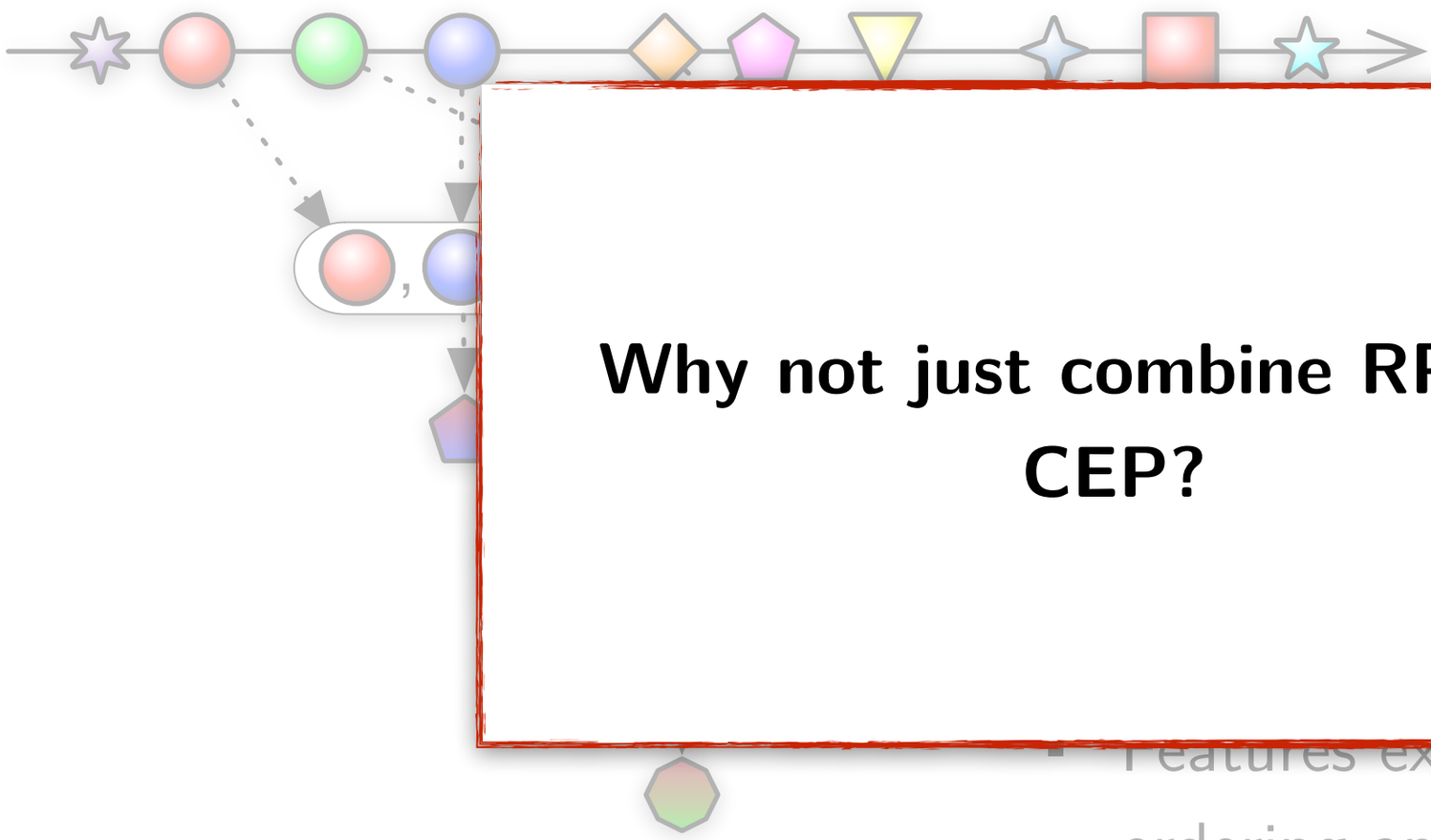


CEP to the Rescue!



- **C**omplex **E**vent **P**rocessing
  - Derives high-level events from combinations of input events
- Features expressive pattern languages for ordering and timing constraints

CEP to the Rescue!



Why not just combine RP with CEP?

ssing

vents from

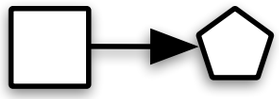
ut events

- Features expressive pattern languages for ordering and timing constraints

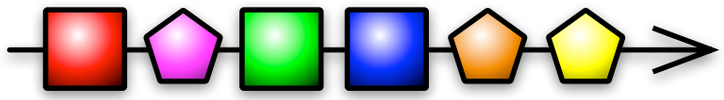
Variability in CEP

Selection

Pattern



Observations

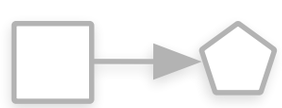


which ones to select?

Variability in CEP

Selection

Pattern

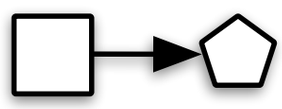


Observations

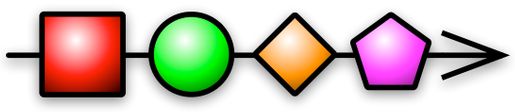
which ones to select?

Contiguity

Pattern



Observation

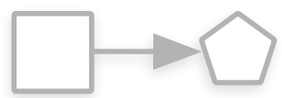


does it match?

Variability in CEP

Selection

Pattern



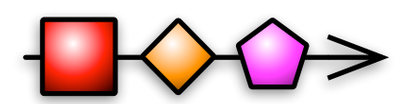
Observations

which ones to select?

Consumption

Pattern

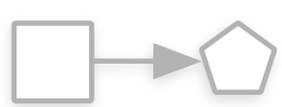
Observations



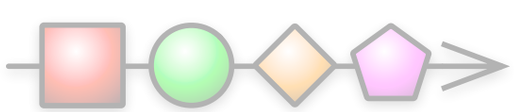
which pattern matches?

Contiguity

Pattern



Observation

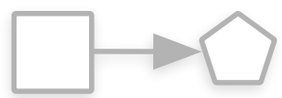


does it match?

Variability in CEP

Selection

Pattern



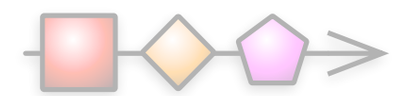
Observations

which ones to select?

Consumption

Pattern

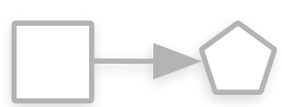
Observations



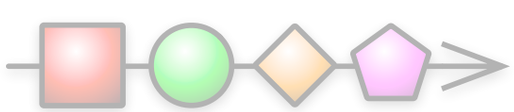
which pattern matches?

Contiguity

Pattern



Observation



does it match?

Time Model

$$t, [t_1, t_2), [t_1, t_2],$$
$$\langle t_1, \ldots, t_n \rangle$$

what is the occurrence "time" of complex events?

Variability in CEP

Selection

Pattern

Obse

which o

Consumption

Pattern

or

s

tches?

- Options are hard-wired in CEP languages
- Unclear semantics

Con

del

P

$t_2]$,

$\langle t_1, \ldots, t_n \rangle$

Observation

does it match?

what is the occurrence "time" of complex events?

Summary

- **Reactive Programming (RP)**
  - Weak support for correlations by timing and order
  - Strong semantic foundations

- **CEP-style patterns complement RP**
  - Huge variability of features
  - Ambiguous and diverse semantics

CorrL

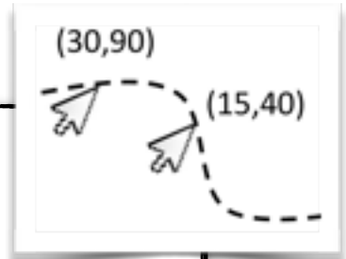
The **Corr**elation **L**anguage

CorrL

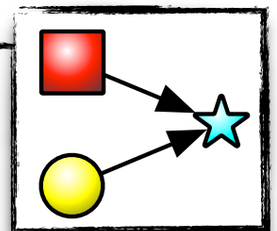
The **Corr**elation **L**anguage

- **CEP-style patterns + Reactive Programming**
  - Embedded DSL (Scala, WIP)
- **Formally specified semantics**
  - Aim for maximal expressivity
  - Support the semantic variability
  - State-machine-based

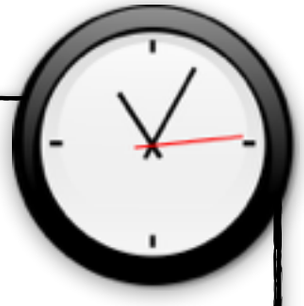
WIP Language Design



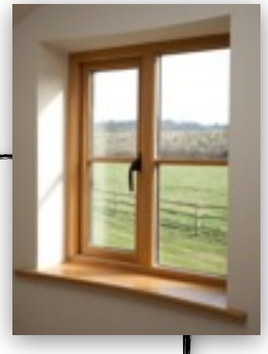

```
await
  down:MouseDown
  on down ->
    move:MouseMove
unless evt:MouseUp
yield Drag(move.x, move.y)
```



```
await
  down:MouseDown
& key:KeyDown.
    key.code == SHIFT
  on down & key ->
    move:MouseMove
unless evt:MouseUp | evt:KeyUp
yield Drag(move.x, move.y)
```



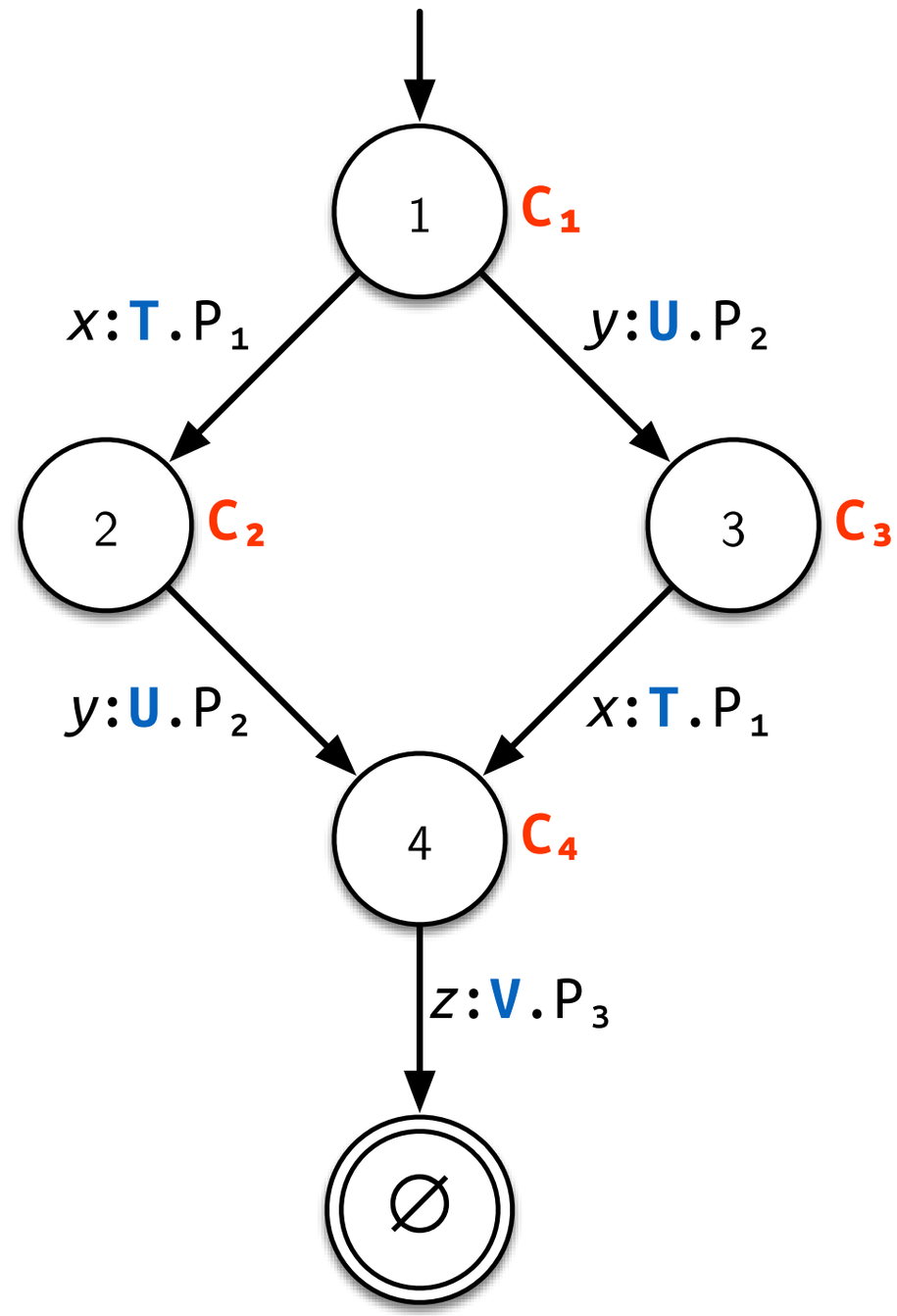
```
await
  down:MouseDown
  on down ->
    last dr:Drag.
    on dr -> move:MouseMove
unless evt:MouseUp
      | time(move)-time(dr) > 0.2s
yield Drag(move.x, move.y)
```



```
await
  down:MouseDown
  on down ->
    move:MouseMove.
      move.y == max(MouseMove.y, 1ms)
unless evt:MouseUp
yield Spike(move.x, move.y)
```

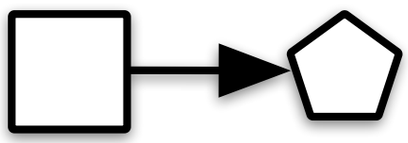
Semantics

```
await
  x:T. P1(x) & y:U. P2(y)
  on x & y ->
    z:V. P3(x,y,z) unless K3(x,y,z)
unless K0 | K1(x) | K2(y)
yield  W(e(x,y,z))
```

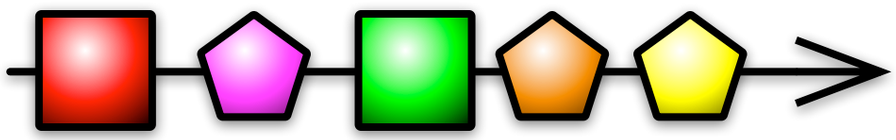


Handling Variability

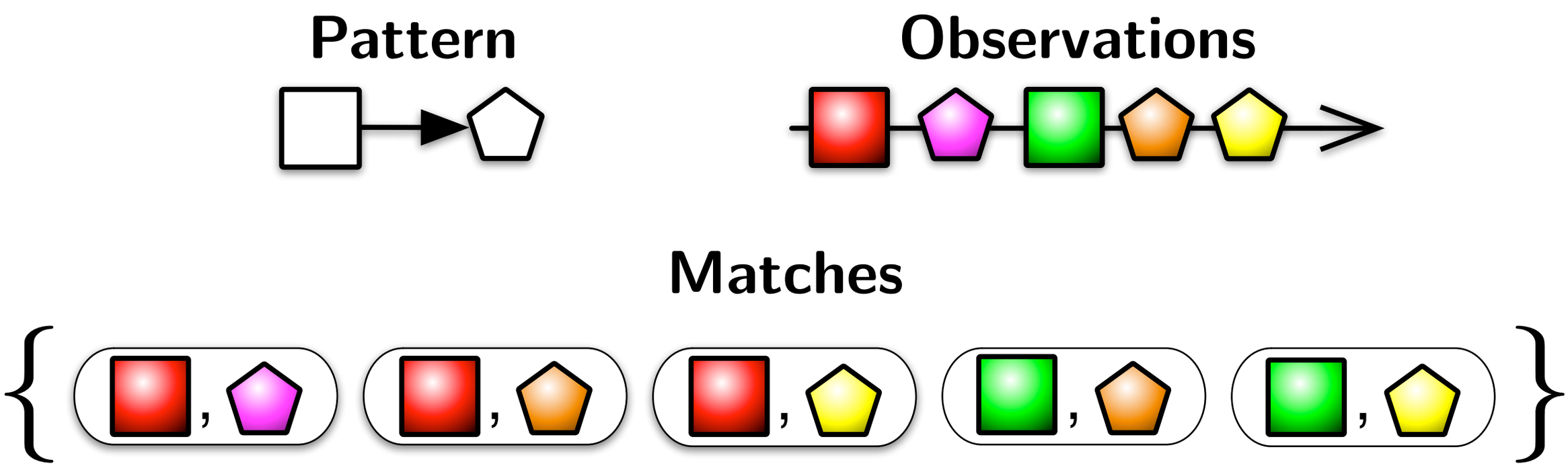
Pattern



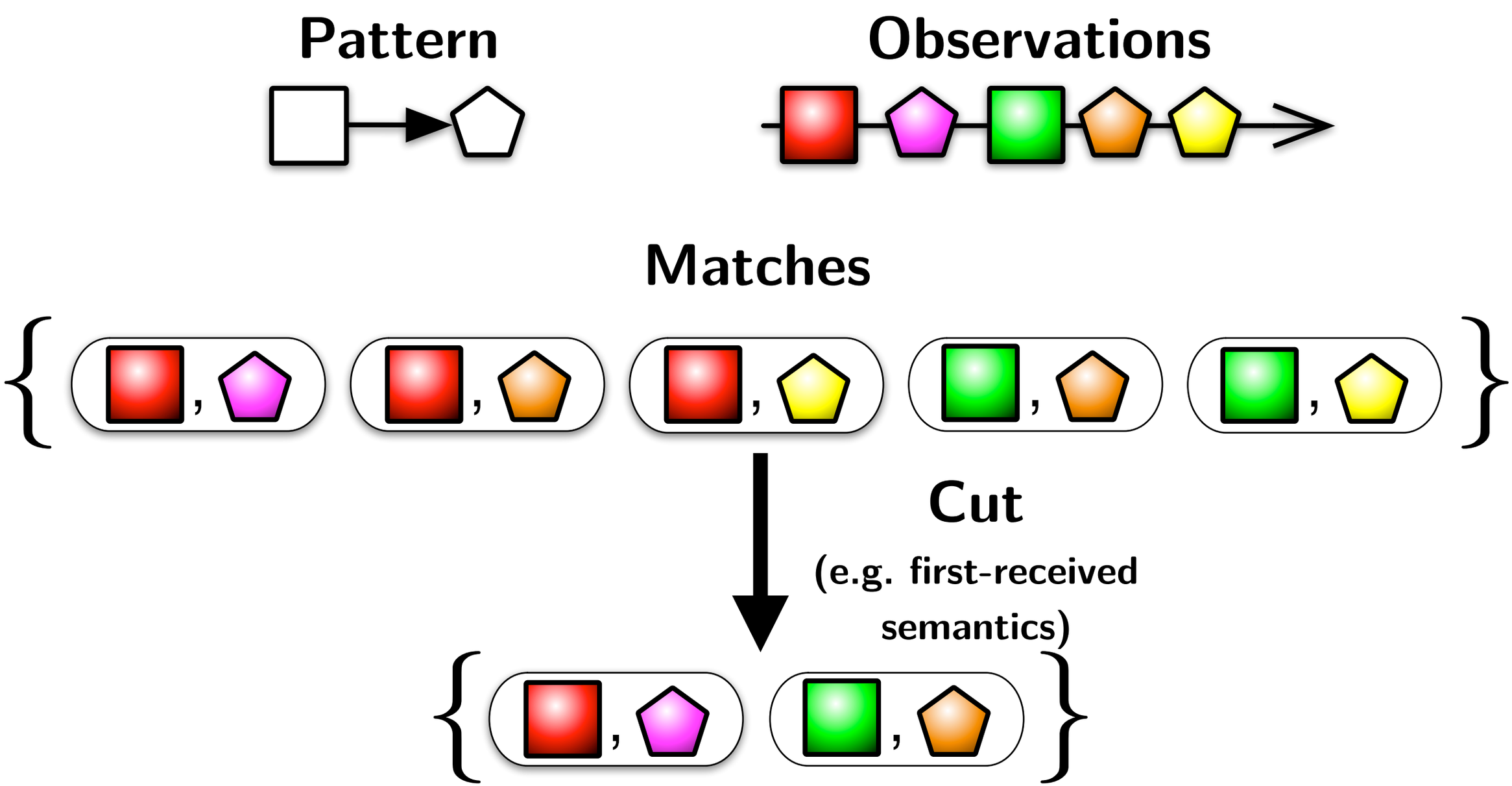
Observations



Handling Variability



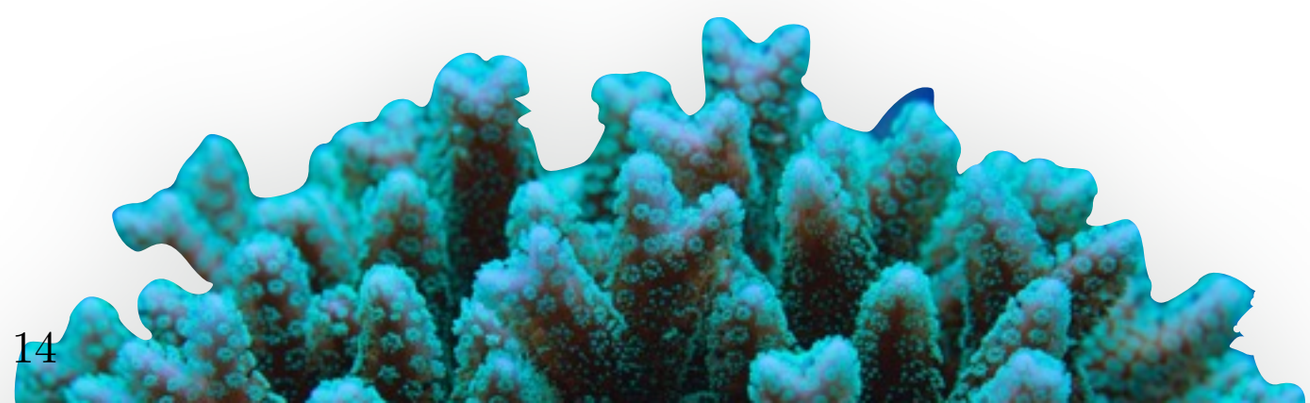
Handling Variability



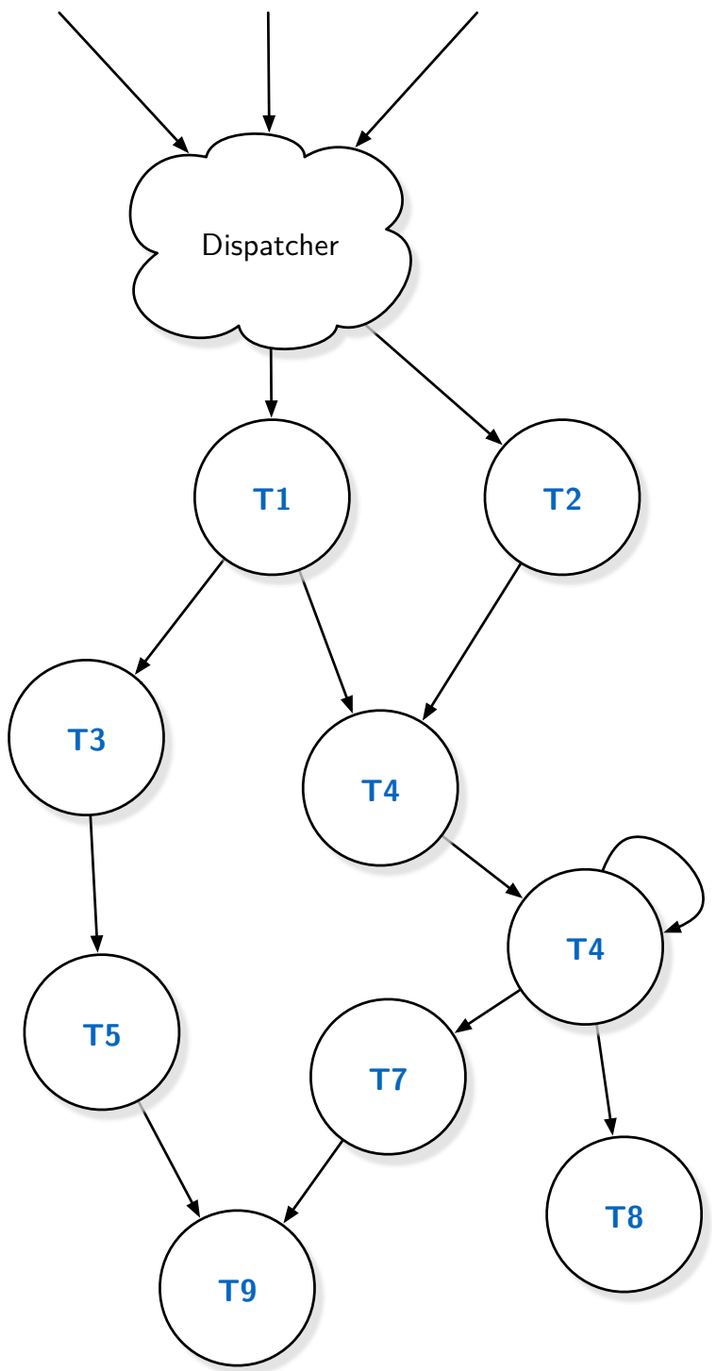
CorrL

Summary and Outlook

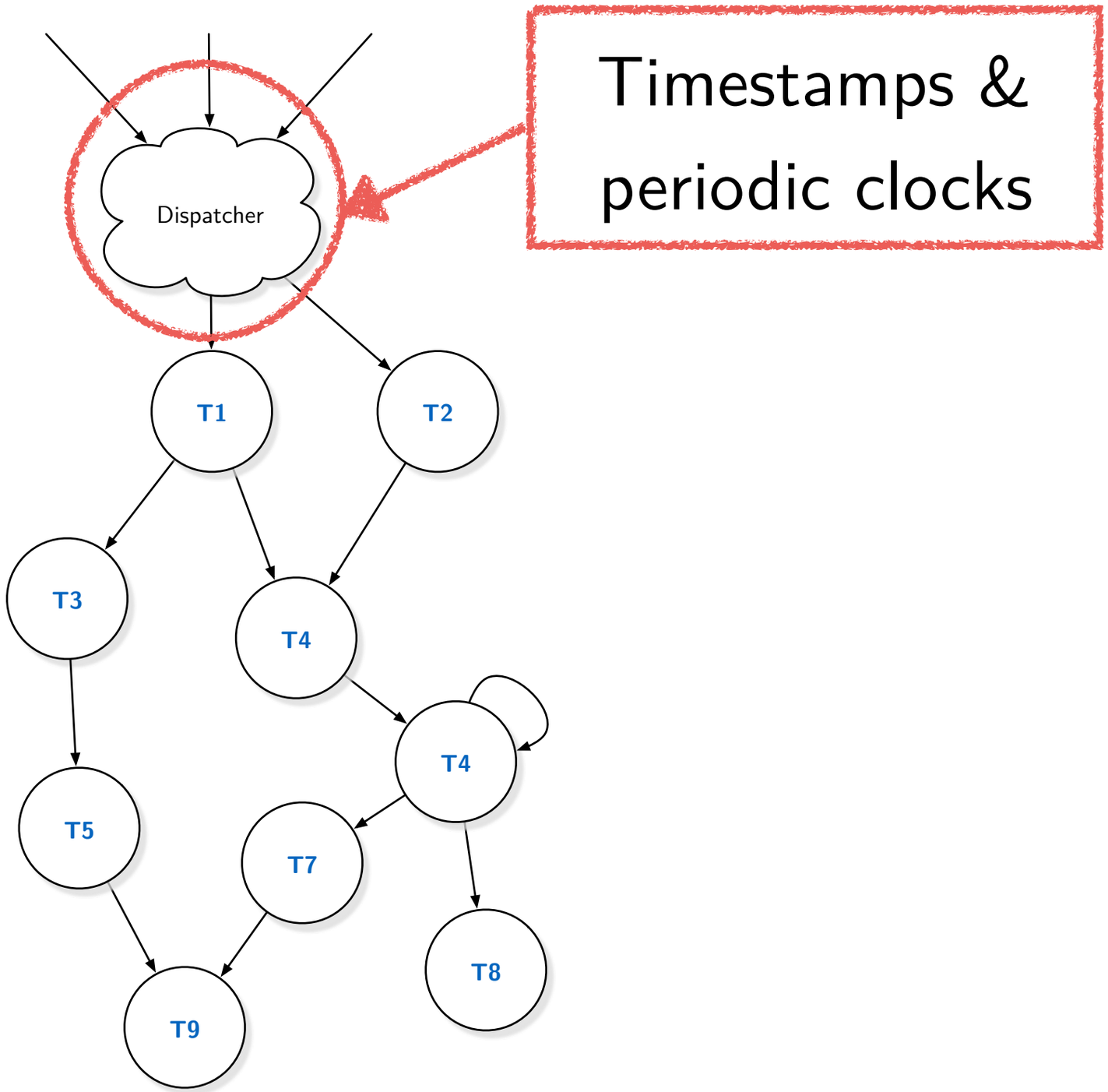
- **Reactive language with expressive correlation patterns**
  - Transformations, Aggregations, Windows, Timing, Partial Orders
- **Which cut functions are efficiently computable?**
  - "No regrets" vs. retraction
- **Explore connection to coordination/ synchronization**



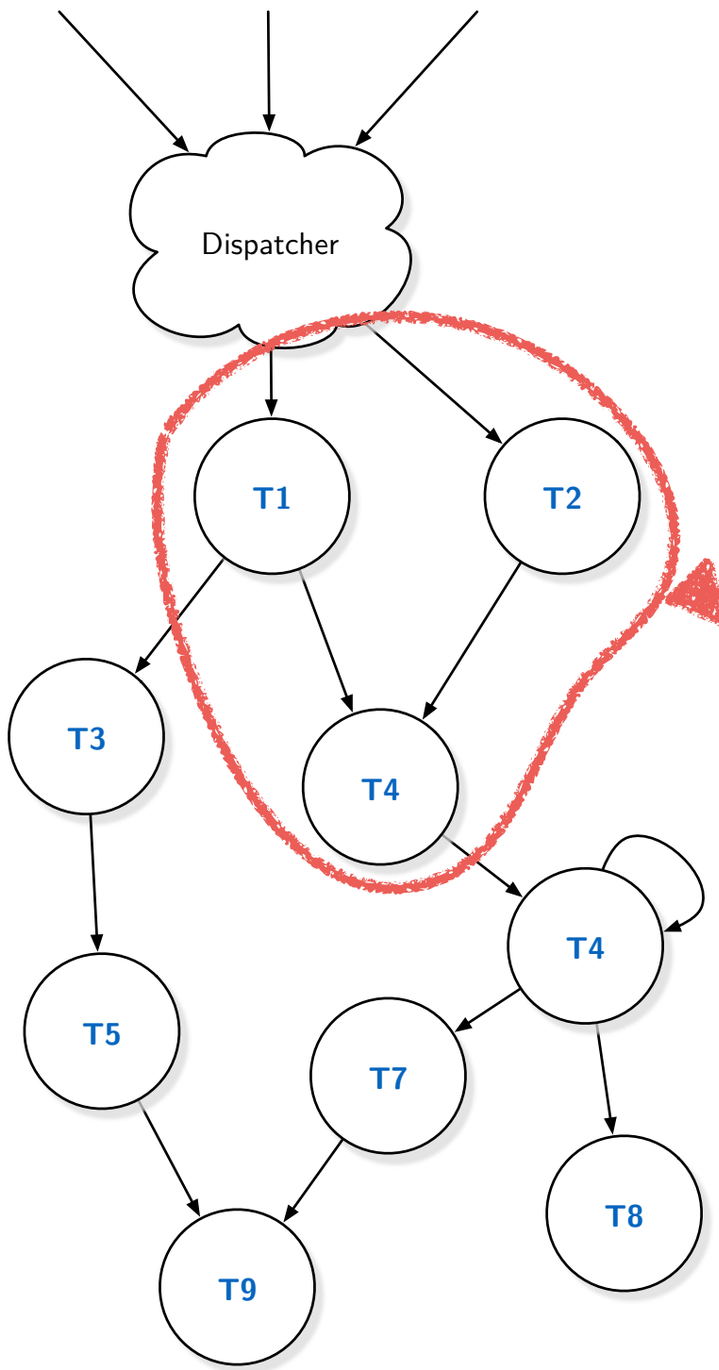
Automata Network



Automata Network



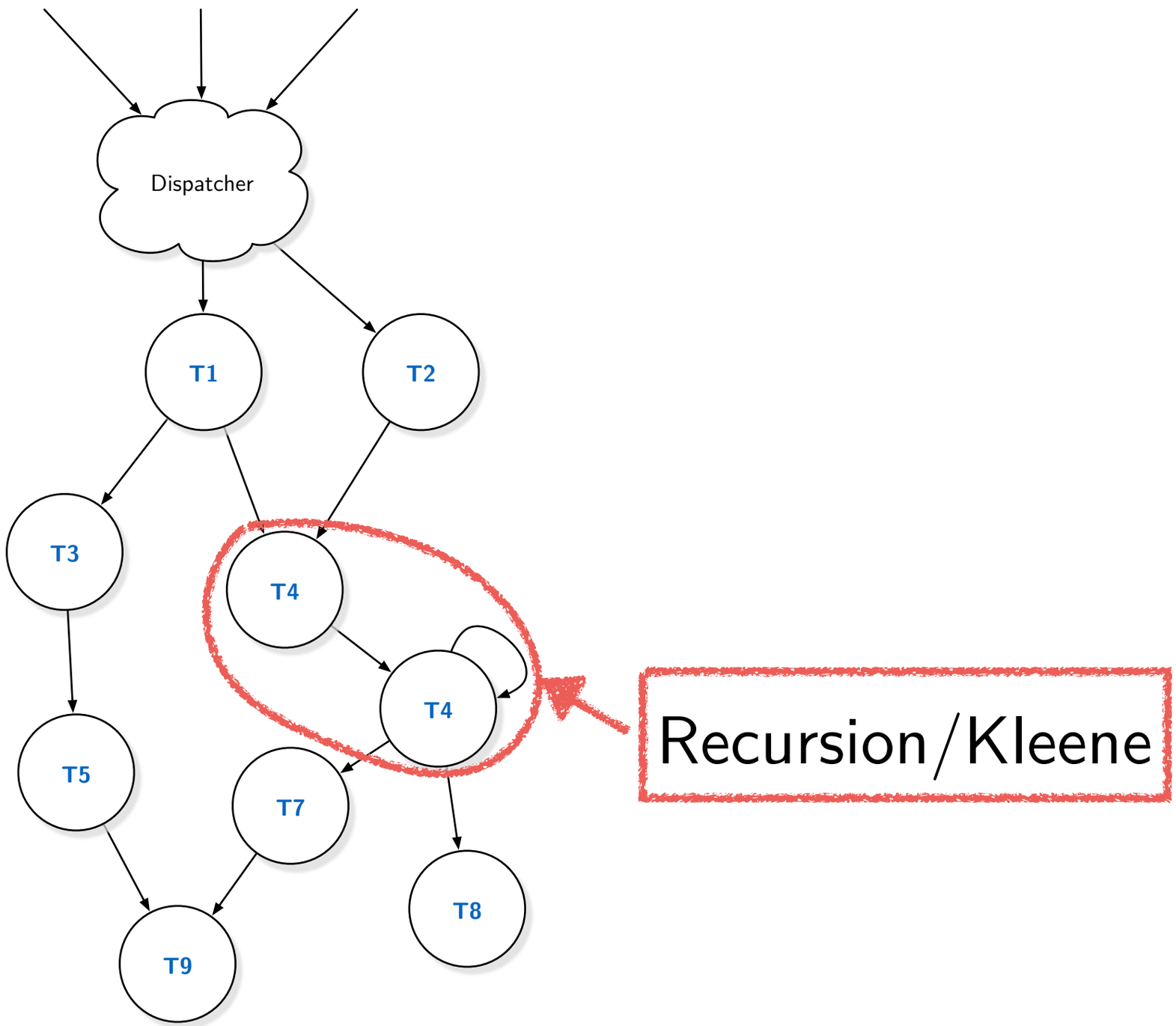
Automata Network



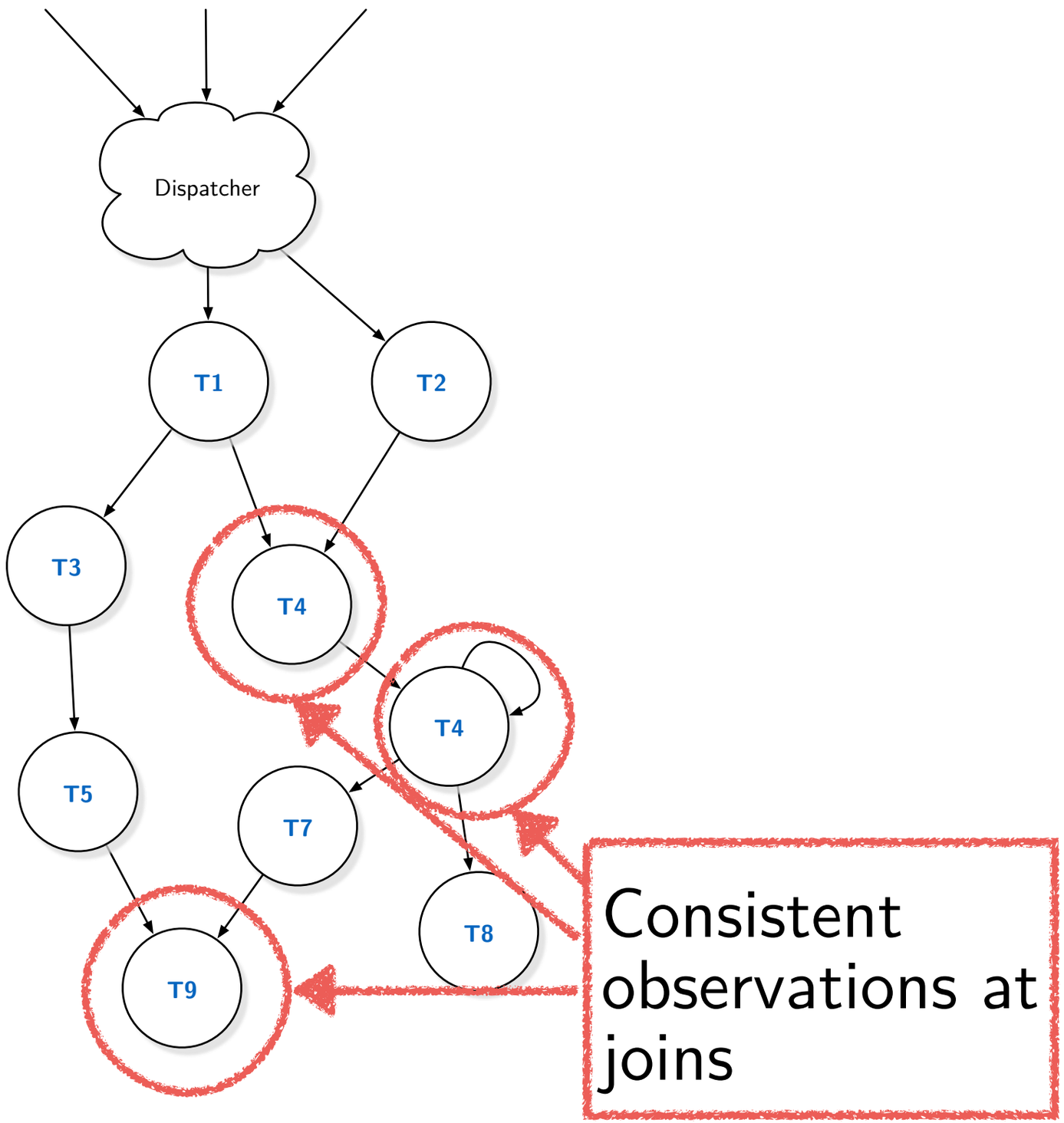
Complex events:

C1(x) & C2(y) => C4(x+y)

Automata Network



Automata Network



C5(t_1)	C7(t_2)
e_1	e_1'
e_2	e_2'
e_3	

Semantics

```
await
  x:T. P₁(x) & y:U. P₂(y)
  on x & y ->
    z:V. P₃(x,y,z) unless K₃(x,y,z)
unless K₀ | K₁(x) | K₂(y)
yield  W(e(x,y,z))
```

Semantics

```
await
  x:T. P1(x) & y:U. P2(y)
  on x & y ->
    z:V. P3(x,y,z) unless K3(x,y,z)
unless K0 | K1(x) | K2(y)
yield  W(e(x,y,z))
```

$\mathbf{v_1}$ = **T**(…)

fork!

```
await
  y:U. P2(y)
  on y ->
    z:V. P3(v1,y,z) unless K3(v1,y,z)
unless K0 | K1(v1) | K2(y)
yield  W(e(v1,y,z))
```

Semantics

```
await
  x:T. P1(x) & y:U. P2(y)
  on x & y ->
    z:V. P3(x,y,z) unless K3(x,y,z)
unless K0 | K1(x) | K2(y)
yield  W(e(x,y,z))
```

$\mathbf{v_1}$ = **T**(…)

fork!

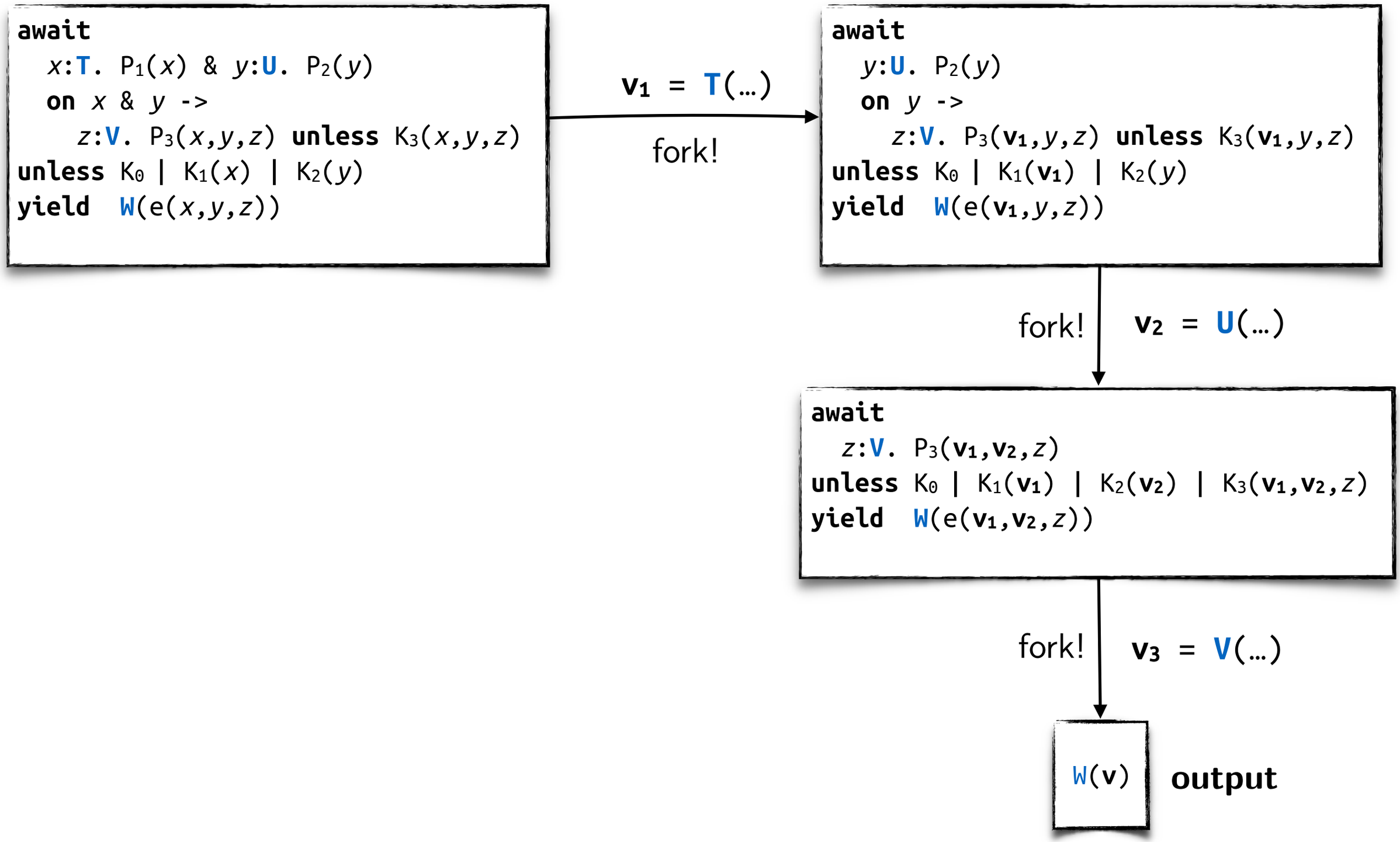
```
await
  y:U. P2(y)
  on y ->
    z:V. P3(v1,y,z) unless K3(v1,y,z)
unless K0 | K1(v1) | K2(y)
yield  W(e(v1,y,z))
```

fork!

$\mathbf{v_2}$ = **U**(…)

```
await
  z:V. P3(v1,v2,z)
unless K0 | K1(v1) | K2(v2) | K3(v1,v2,z)
yield  W(e(v1,v2,z))
```

Semantics


await
x:T. P1(x) & y:U. P2(y)
on x & y ->
z:V. P3(x,y,z) unless K3(x,y,z)
unless K0 | K1(x) | K2(y)
yield W(e(x,y,z))
v1 = T(…)
fork!
await
y:U. P2(y)
on y ->
z:V. P3(v1,y,z) unless K3(v1,y,z)
unless K0 | K1(v1) | K2(y)
yield W(e(v1,y,z))
fork!
v2 = U(…)
await
z:V. P3(v1,v2,z)
unless K0 | K1(v1) | K2(v2) | K3(v1,v2,z)
yield W(e(v1,v2,z))
fork!
v3 = V(…)
W(v)
output

Semantics

```
await
  x:T. P1(x) & y:U. P2(y)
  on x & y ->
    z:V. P3(x,y,z) unless K3(x,y,z)
unless K0 | K1(x) | K2(y)
yield  W(e(x,y,z))
```

$\mathbf{v_1} = \mathbf{T}(\ldots)$

fork!

```
await
  y:U. P2(y)
  on y ->
    z:V. P3(v1,y,z) unless K3(v1,y,z)
unless K0 | K1(v1) | K2(y)
yield  W(e(v1,y,z))
```

fork! $\mathbf{v_2} = \mathbf{U}(\ldots)$

```
await
  z:V. P3(v1,v2,z)
unless K0 | K1(v1) | K2(v2) | K3(v1,v2,z)
yield  W(e(v1,v2,z))
```

fork! $\mathbf{v_3} = \mathbf{V}(\ldots)$

$\mathbf{W}(\mathbf{v})$ **output**

Fork on every reduction step, maximal selectivity

Join Patterns

```
//initialize buffer with the first frame
srcs⟨frame⟩ if frame.id == 0 ▷
  buffer⟨frame :: Nil⟩ ‖ last⟨0⟩

//append to buffer in the order of the id
srcs⟨frame⟩ & buffer⟨xs⟩ & last⟨n⟩
  if frame.id == n + 1 ▷
     buffer⟨xs.frame⟩ ‖ last⟨frame.id⟩

//output buffer in chunks
buffer⟨fs⟩ if fs.length ≥ N ▷
  buffer⟨fs.drop(N)⟩ ‖ out⟨fs.take(N)⟩
```

Synchronization = Correlation!