

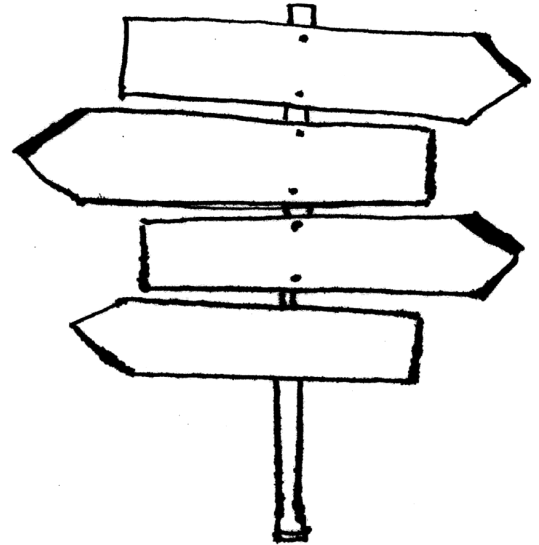


# Polite for EV3

Stefan Borer, 11-932-407  
Theodor Truffer, 11-103-157

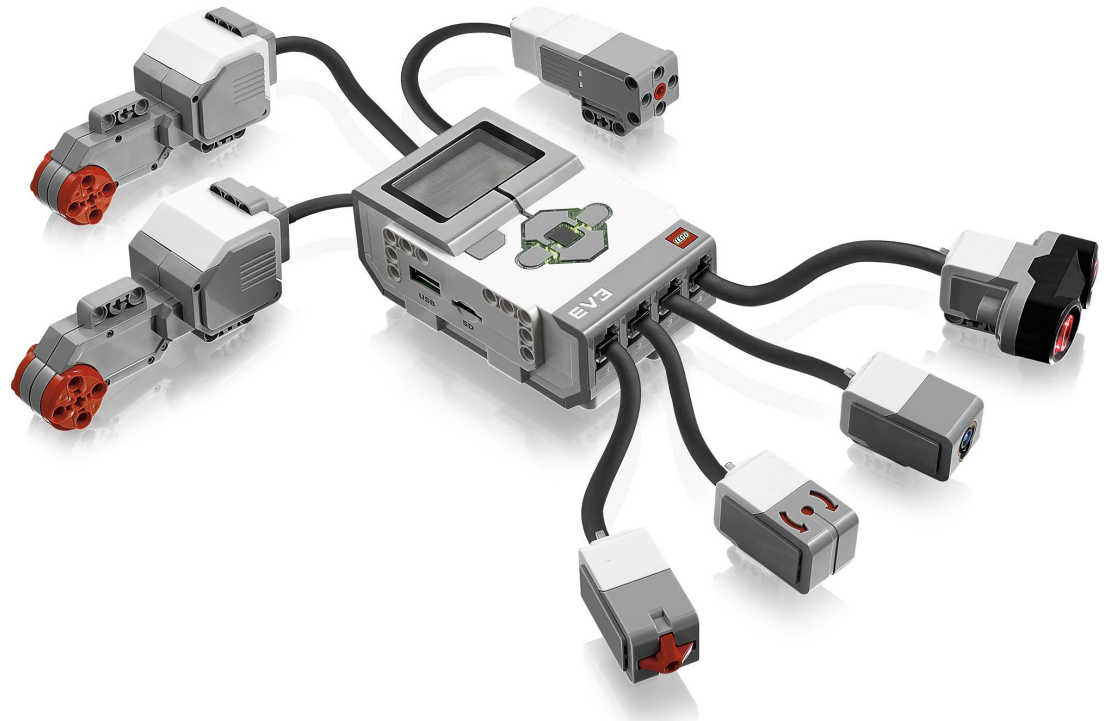
# Roadmap

1. What is EV3?
2. Existing Projects
3. Polite for EV3
4. Demo



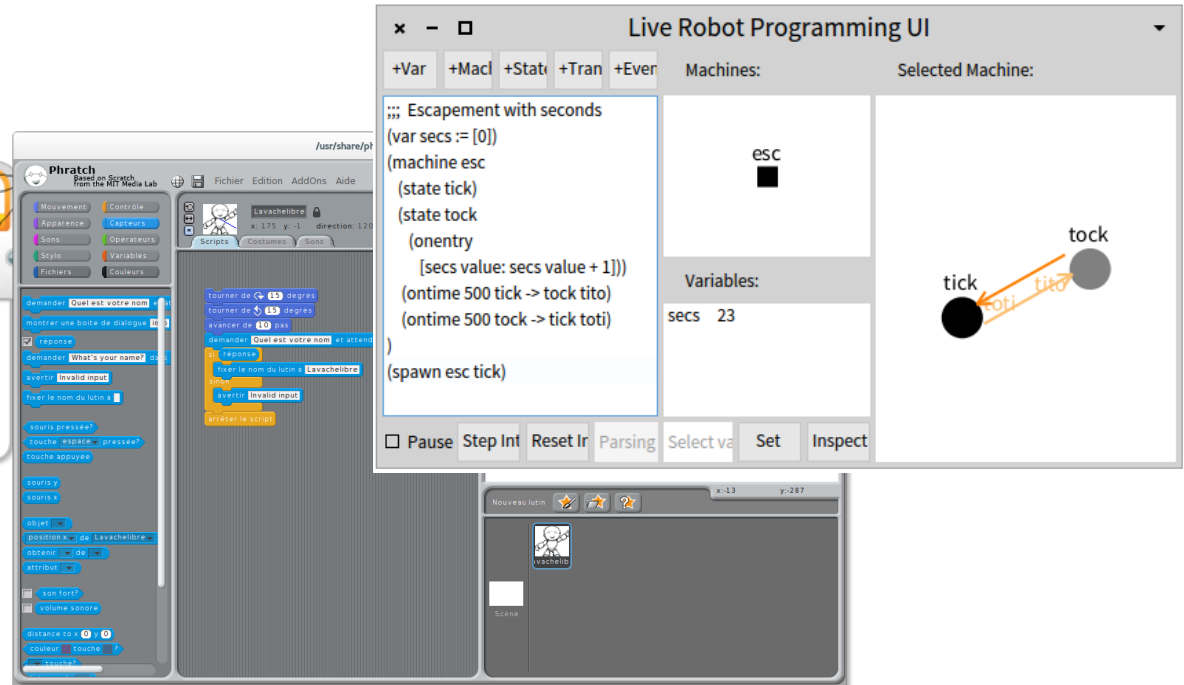
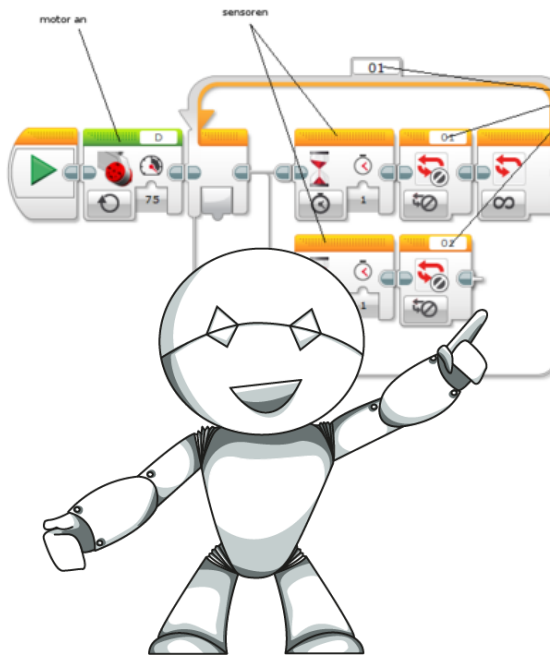
# EV3 Brick

- 4 Sensors
- up to 4 Motors
- Wifi key
- HTTP-Server

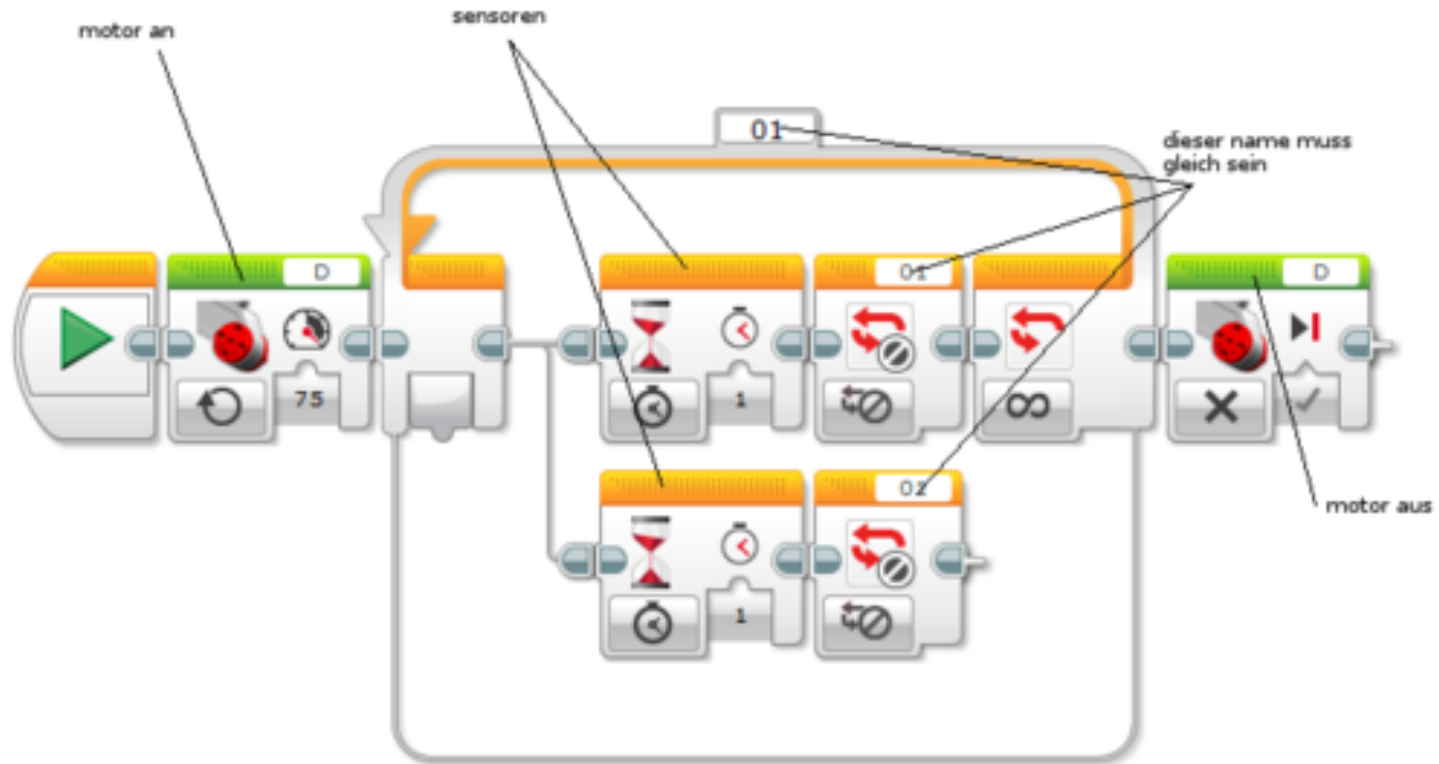


# Existing Projects

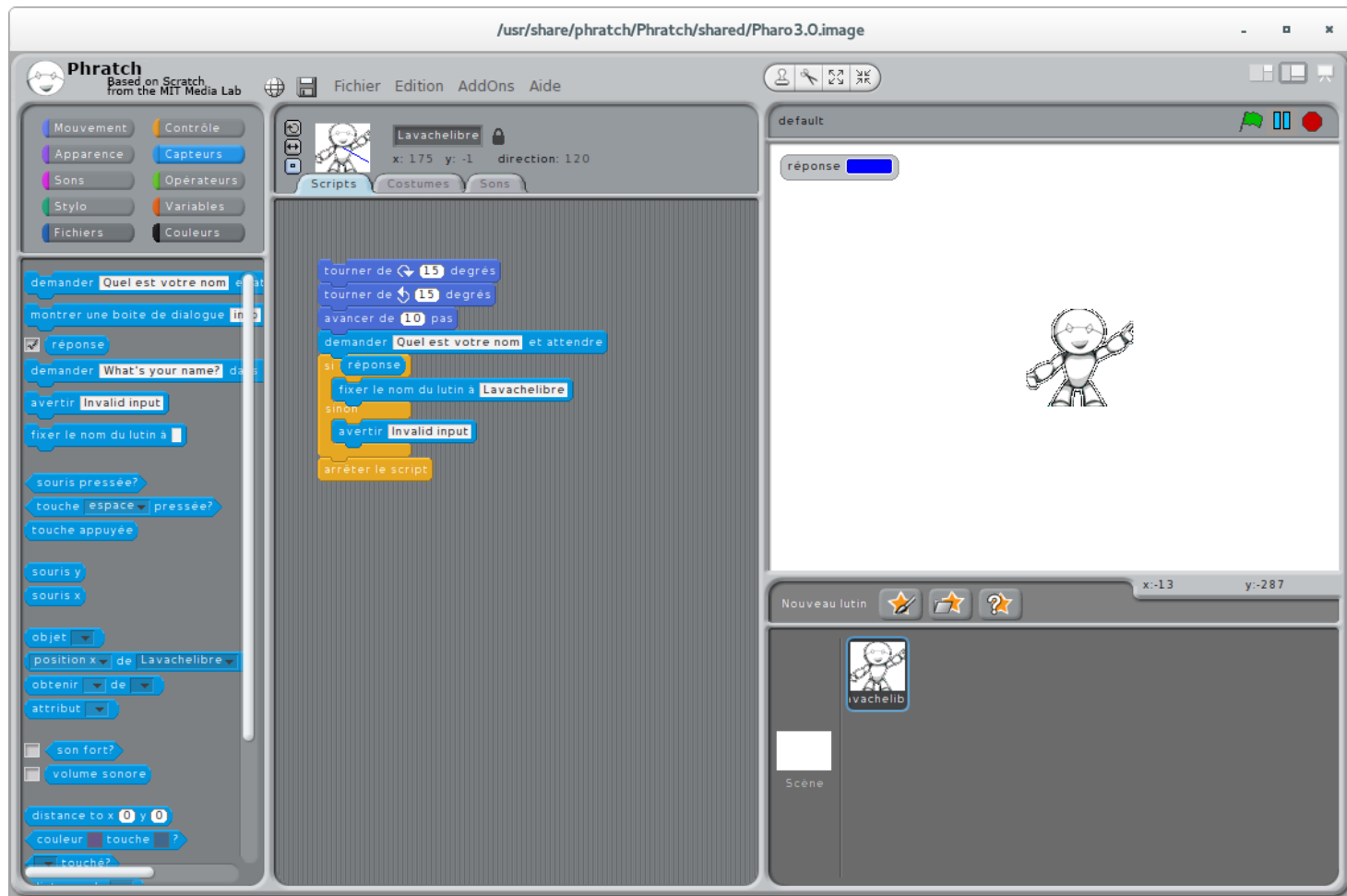
**Pleiad:**



# Lego Mindstorms



# Phratch with Jetstorm



# Live Robot Programming

The screenshot displays the 'Live Robot Programming UI' interface, which is divided into several functional areas:

- Code Editor:** Contains a script for an escapement mechanism. The code defines a variable `secs` and a machine `esc` with states `tick` and `tock`. Transitions are triggered by `onentry`, `onetime 500`, and `onetime 500` events.
- Machines:** A list of active machines, currently showing `esc` represented by a small black square.
- Variables:** A section showing the current value of `secs` as 23.
- Selected Machine:** A detailed view of the selected machine's state. It shows a state transition diagram with two states, `tick` (black circle) and `tock` (gray circle). Transitions are labeled `toti` (from `tock` to `tick`) and `tito` (from `tick` to `tock`).
- Controls:** A bottom bar with buttons for `Pause`, `Step Int`, `Reset Ir`, `Parsing`, `Select va`, `Set`, and `Inspect`.

```
;;; Escapement with seconds
(var secs := [0])
(machine esc
  (state tick)
  (state tock
    (onentry
      [secs value: secs value + 1]))
  (onetime 500 tick -> tock titi)
  (onetime 500 tock -> tick toti)
)
(spawn esc tick)
```

Variables:

secs 23

tick tock

titi titi

**What is Polite for EV3?**



# Polite Smalltalk



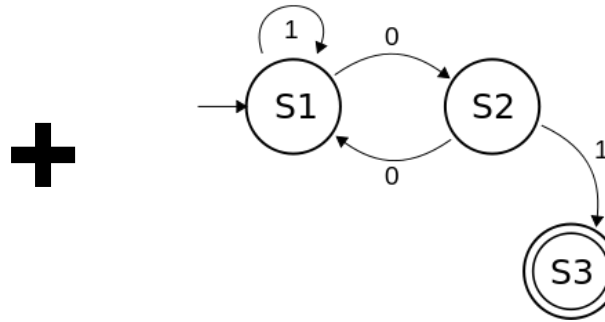
"Have some Polite Code:"

```
protagonist := Polite Hero, named: 'The Red Rider'.
```

```
protagonist, beat the hell out of: (the bad guys) and save: (the lady).
```

# Polite for EV3

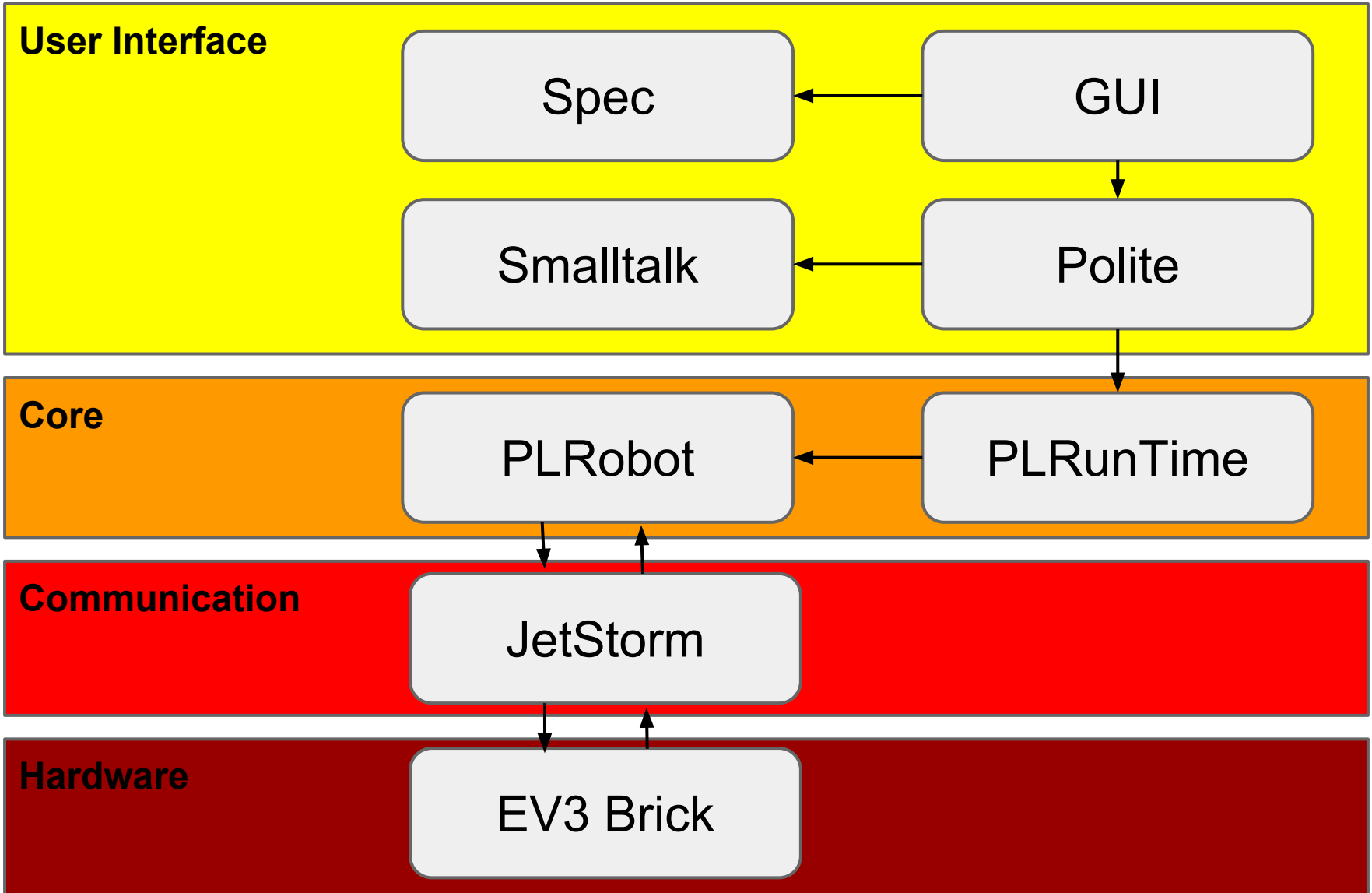
## JetStorm



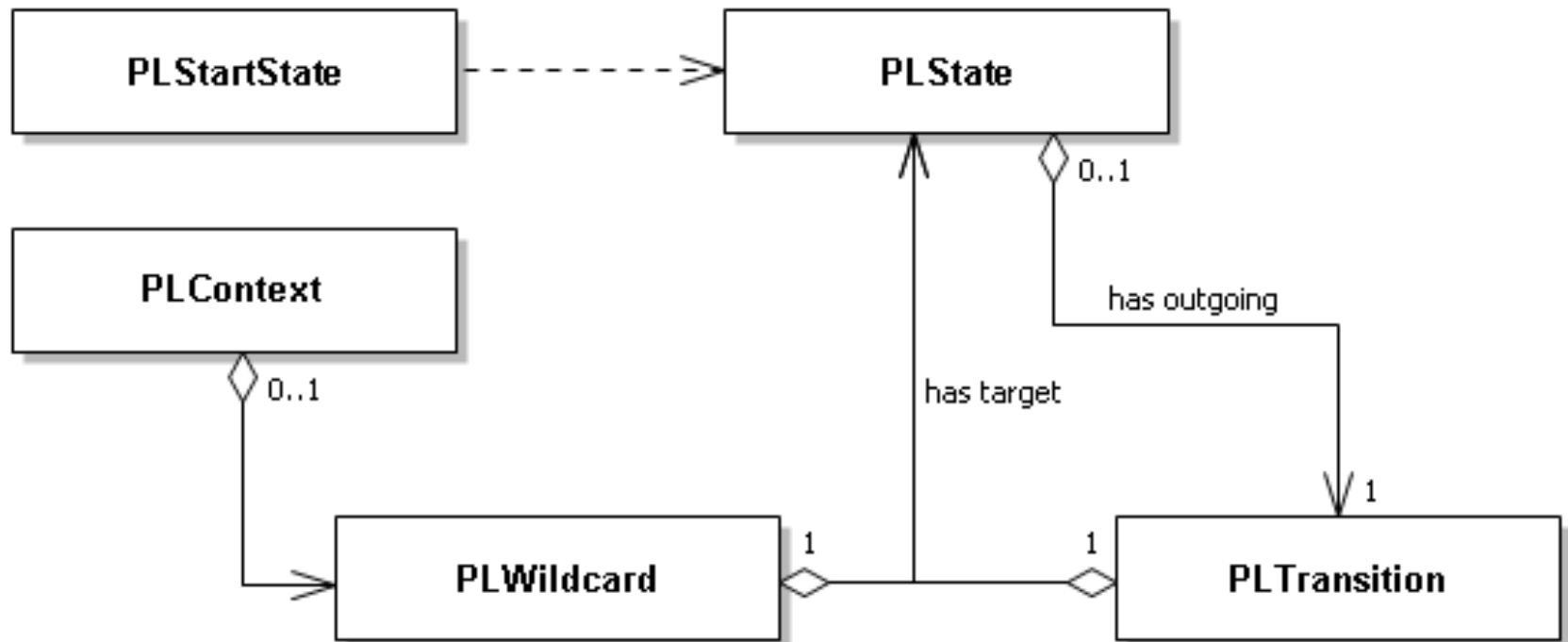
+



# Architectural overview

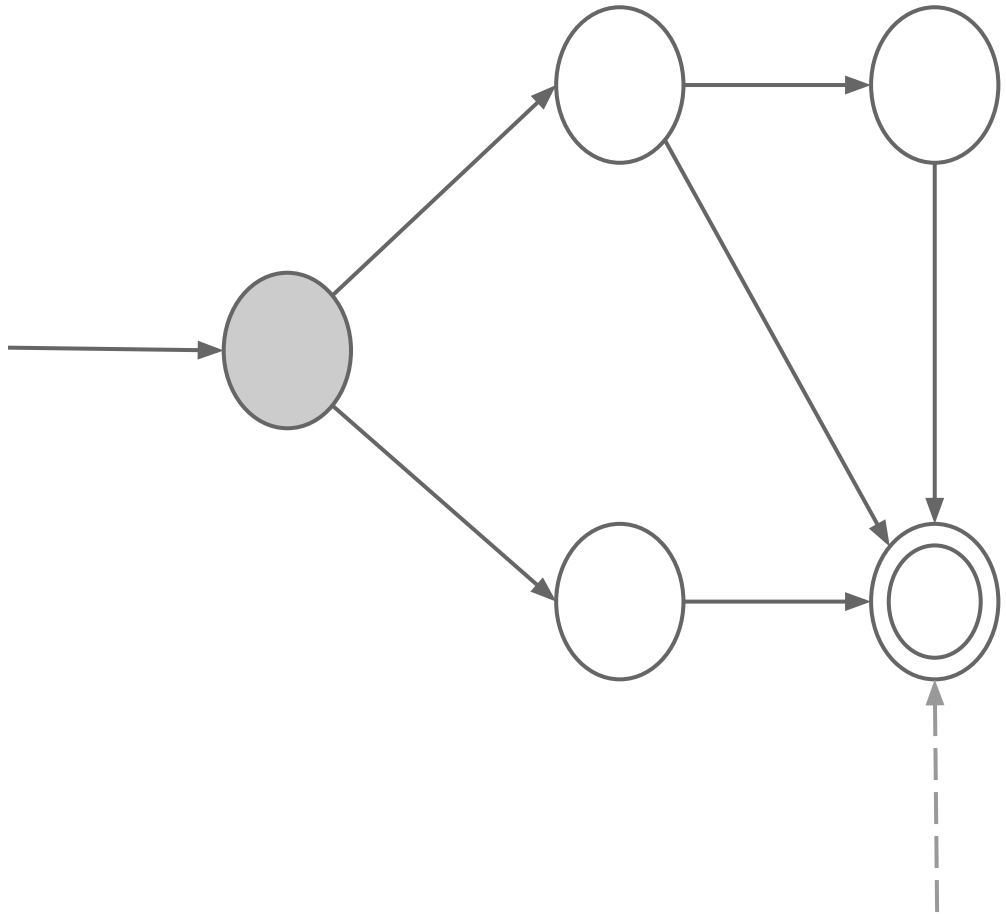


# State machine - UML

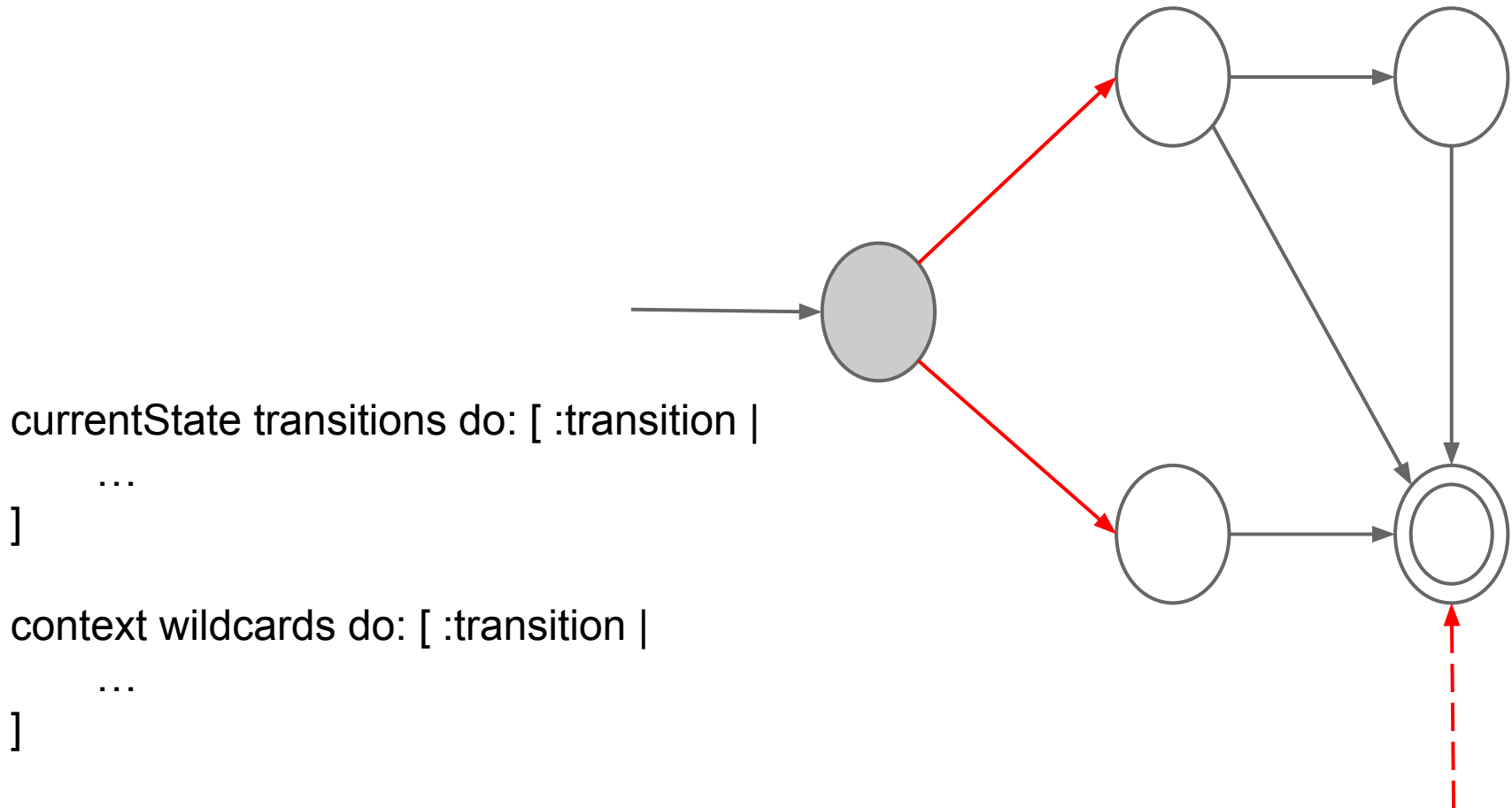


# State machine - Processing

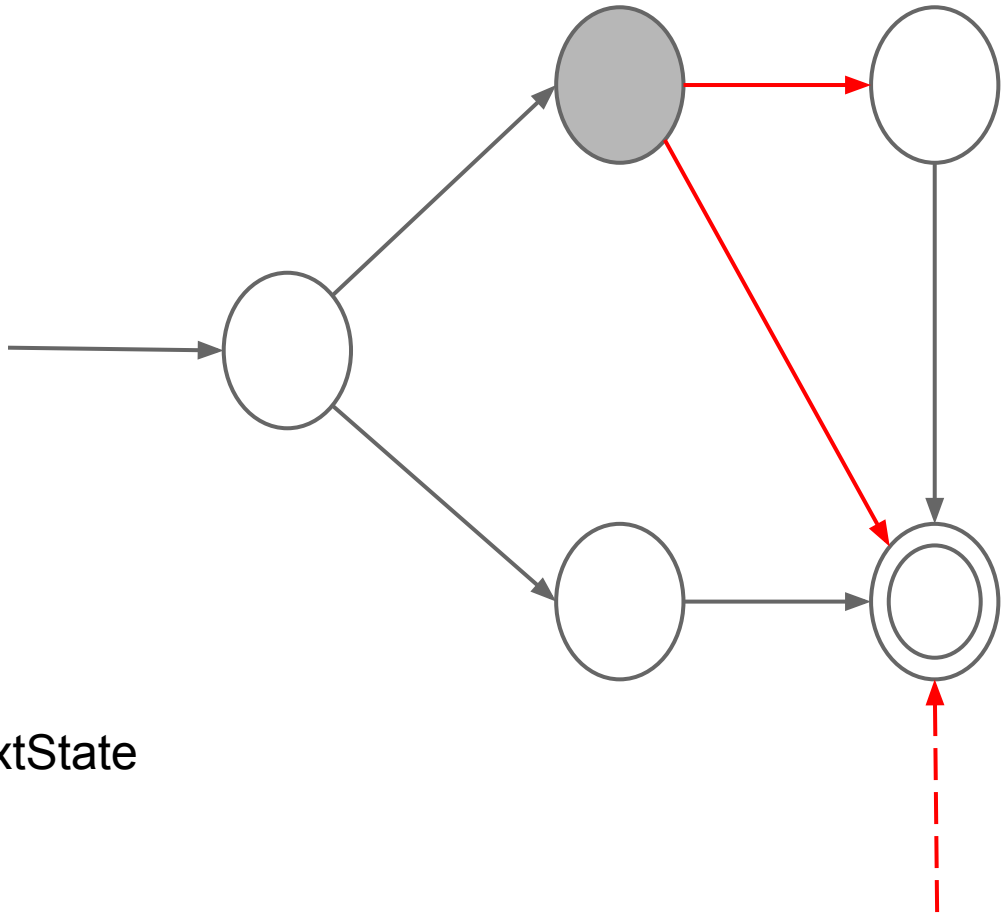
...  
processor, start: (start state).  
...



# State machine - Processing



# State machine - Processing



```
(transition condition) ifTrue: [  
    transition action.  
    currentState := transition nextState  
]
```

**Demo**



# Outlook

- Add beginner-friendly API-Layer
- Support for variables
- More responsive UI
- Allow rendering of nested machine

**Thank you!**