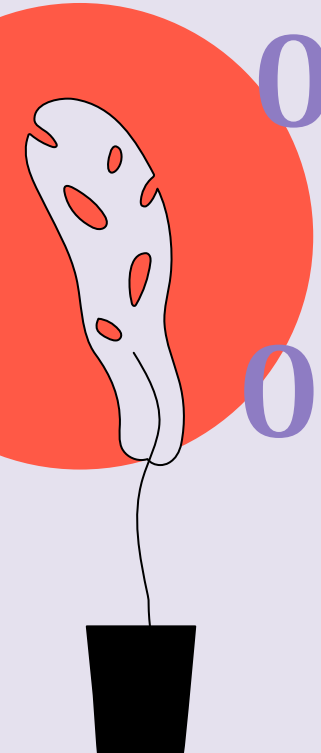
# Cyber-Physical Programming

Practical Assignment

# Table of contents



# 01

# Introduction

# Presented By

PG 50240 Beatriz Rodrigues

PG 50667 Gonçalo Rodrigues

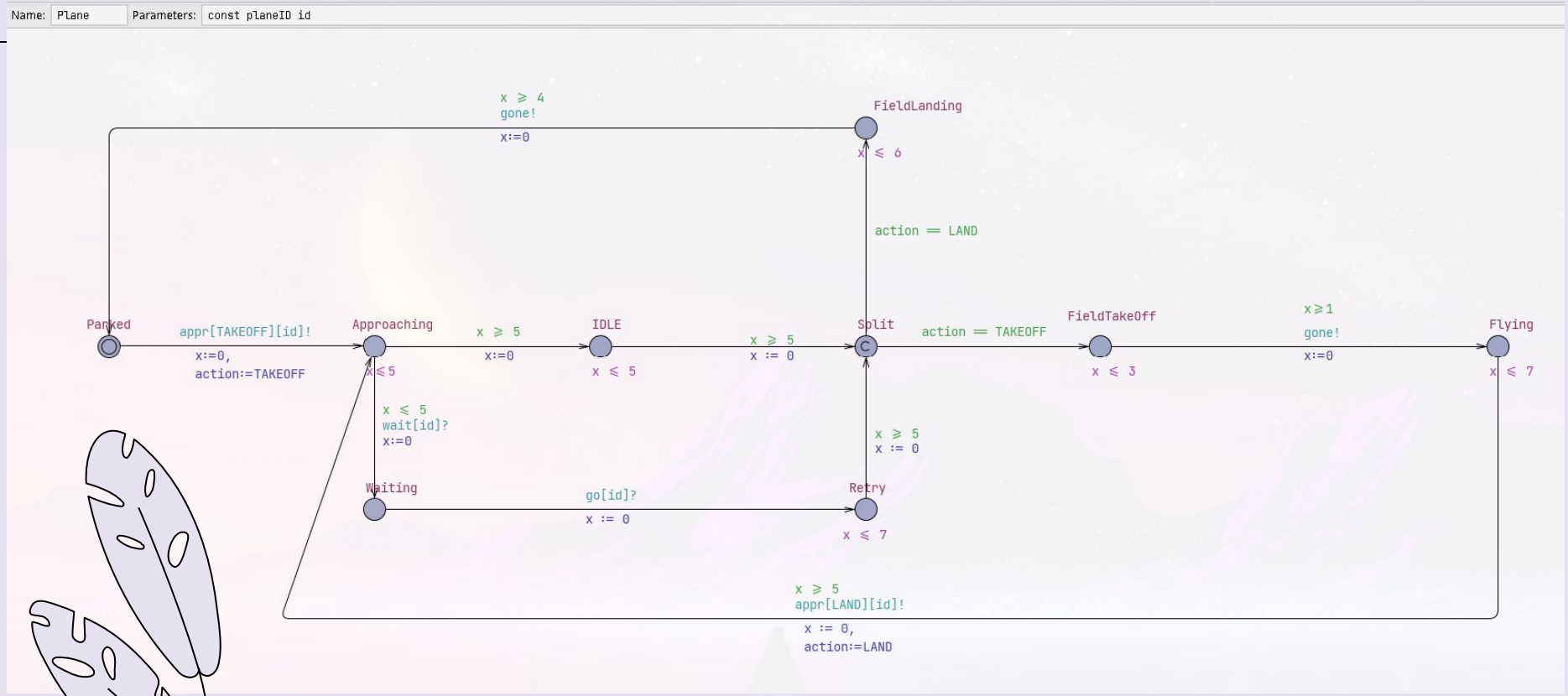
“Testing can
only find bugs, not
prove their absence”

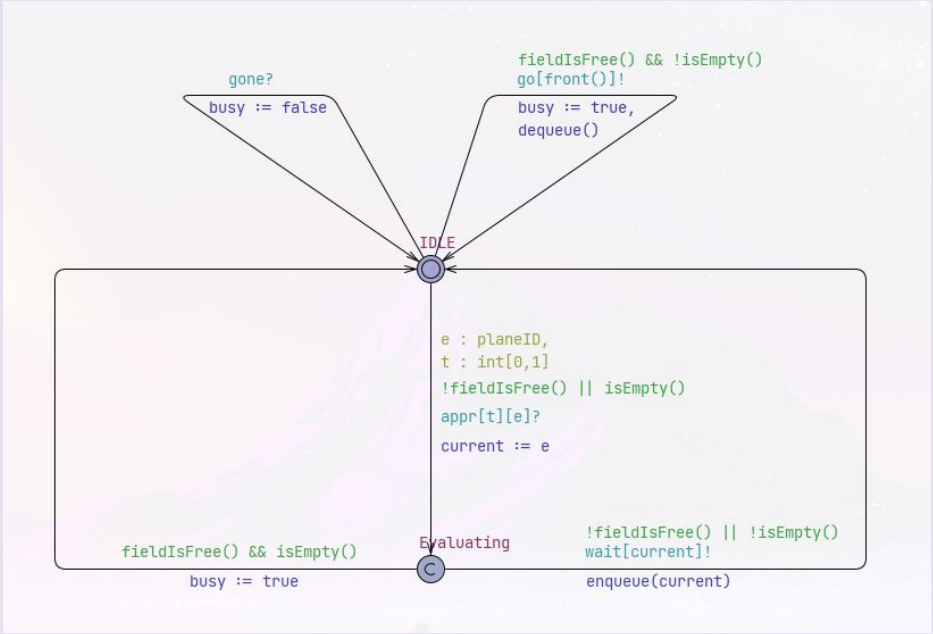
—Edsger Dijkstra

# 02

# First Task

Managing shared resources with UPPAAL


Name: Plane
Parameters: const planeID id
x ⩾ 4
gone!
x:=0
FieldLanding
x ⩽ 6
action == LAND
Parked
appr[TAKEOFF][id]!
x:=0,
action:=TAKEOFF
Approaching
x ⩽ 5
x ⩾ 5
x:=0
IDLE
x ⩽ 5
x ⩾ 5
x := 0
Split
action == TAKEOFF
FieldTakeOff
x ⩽ 3
x⩾1
gone!
x:=0
Flying
x ⩽ 7
x ⩽ 5
wait[id]?
x:=0
x ⩾ 5
x := 0
Waiting
go[id]?
x := 0
Retry
x ⩽ 7
x ⩾ 5
appr[LAND][id]!
x := 0,
action:=LAND


gone?
busy := false
fieldIsFree() && !isEmpty()
go[front()]!
busy := true,
dequeue()
IDLE
e : planeID,
t : int[0,1]
!fieldIsFree() || isEmpty()
appr[t][e]?
current := e
fieldIsFree() && isEmpty()
busy := true
Evaluating
!fieldIsFree() || !isEmpty()
wait[current]!
enqueue(current)

# Verification - Reachability Properties

**There will be a state where plane 1 is using the field and all other planes will be waiting for their turn.**

```
E<> (Plane(1).FieldTakeOff || Plane(1).FieldLanding) && forall (i:planeID) i != 1 imply
Plane(i).Waiting
```

**Plane 1 is able to fly.**

```
E<> Plane(1).Flying:
```

# Verification - Safety Properties

**There can never be N planes in the queue.**

```
A[] Controller.len < N
```

**The model should have no deadlocks.**

```
A[] not deadlock
```

# Verification - Safety Properties

**The field can only be accessed by one plane at a time.**

```
A[] forall (i:planeID) forall (j:planeID) (Plane(i).FieldLanding||Plane(i).FieldTakeOff)
&& (Plane(j).FieldLanding || Plane(j).FieldTakeOff) imply i == j
```

**It's possible for plane 1 to never leave the Parked state.**

```
E[] Plane(1).Parked
```

# Verification - Liveness Properties

**If plane 1 is waiting, its turn will eventually arrive.**

```
Plane(1).Waiting --> (Plane(1).Flying || Plane(1).Parked)
```

**If a plane leaves the field to fly, it will eventually attempt to return.**

```
Plane(1).Flying --> Plane(1).Approaching
```

03

# Second Task

Using the right concepts in
software development

# 04

# Third Task

Unified program semantics

# Semantics

$$\frac{\langle p,\sigma\rangle \Downarrow \sum_i^n p_i \cdot \sigma_i \quad \forall i \le n.\langle q,\sigma_i\rangle \Downarrow \mu_i}{\langle p;q,\sigma\rangle \Downarrow \sum_i^n p_i \cdot \mu_i} \quad \text{(seq)}$$

$$\frac{\langle t,\sigma\rangle \Downarrow r}{\langle x := t,\sigma\rangle \Downarrow 1 \cdot \sigma[r/x]} \quad \text{(asg)}$$

$$\frac{\langle p,\sigma\rangle \Downarrow \sum_i^n p_i \cdot \sigma_i \quad \langle q,\sigma\rangle \Downarrow \sum_i^m q_i \cdot \sigma'_i}{\langle p +_p q,\sigma\rangle \Downarrow \sum_i^n (p_i * p) \cdot \sigma_i + \sum_i^m (q_i * (1-p)) \cdot \sigma'_i} \quad \text{(cho)}$$

$$\frac{\langle b,\sigma\rangle \Downarrow tt \quad \langle p,\sigma\rangle \Downarrow \sigma'}{\langle \text{if b then p else q},\sigma\rangle \Downarrow 1 \cdot \sigma'} \quad \text{(if1)}$$

$$\frac{\langle b,\sigma\rangle \Downarrow ff \quad \langle q,\sigma\rangle \Downarrow \sigma'}{\langle \text{if b then p else q},\sigma\rangle \Downarrow 1 \cdot \sigma'} \quad \text{(if2)}$$

$$\frac{\langle b,\sigma\rangle \Downarrow tt \quad \langle p,\sigma\rangle \Downarrow \sum_i^n p_i \cdot \sigma_i \quad \forall i \le n.\langle \text{while b do } \{p\},\sigma_i\rangle \Downarrow \mu_i}{\langle \text{while b do } \{p\},\sigma\rangle \Downarrow \sum_i^n p_i \cdot \mu_i} \quad \text{(wh1)}$$

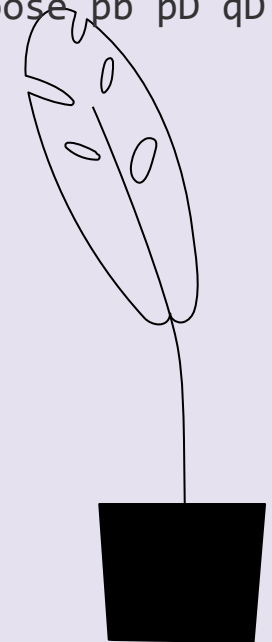
$$\frac{\langle b,\sigma\rangle \Downarrow ff}{\langle \text{while b do } \{p\},\sigma\rangle \Downarrow 1 \cdot \sigma} \quad \text{(wh2)}$$

# Data type

```
data ProgTerm = Asg Vars LTerm
              | Choice ProbRep ProgTerm ProgTerm
              | Seq ProgTerm ProgTerm
              | Ife BTerm ProgTerm ProgTerm
              | Wh BTerm ProgTerm
    deriving Show
```

# Function

```
wsem :: ProgTerm -> (Vars -> Double) -> Dist (Vars -> Double)
wsem (Asg x t) m = return $ chMem x (sem t m) m
wsem (Choice pb p q) m = do pD <- wsem p m ; qD <- wsem q m ; choose pb pD qD
wsem (Seq p q) m = wsem p m >>= wsem q
wsem (Ife b p q) m | bsem b m = wsem p m
                   | otherwise = wsem q m
wsem (Wh b p) m | bsem b m = wsem p m >>= wsem (Wh b p)
                | otherwise = return m
```



# Thanks!