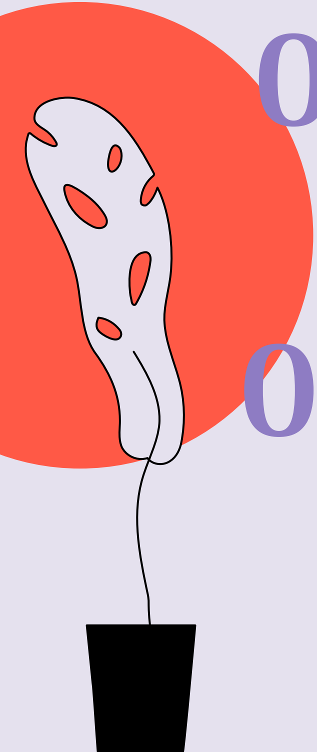
Cyber-Physical Programming

Practical Assignment

Table of contents



01 Introduction

Workflow

02 First Task

Managing shared resources with UPPAL

03 Second Task

Essay on using the right concepts in software development

04 Third Task

Unified program semantics

01

Introduction

Presented By

PG 50240 Beatriz Rodrigues

PG 50667 Gonçalo Rodrigues

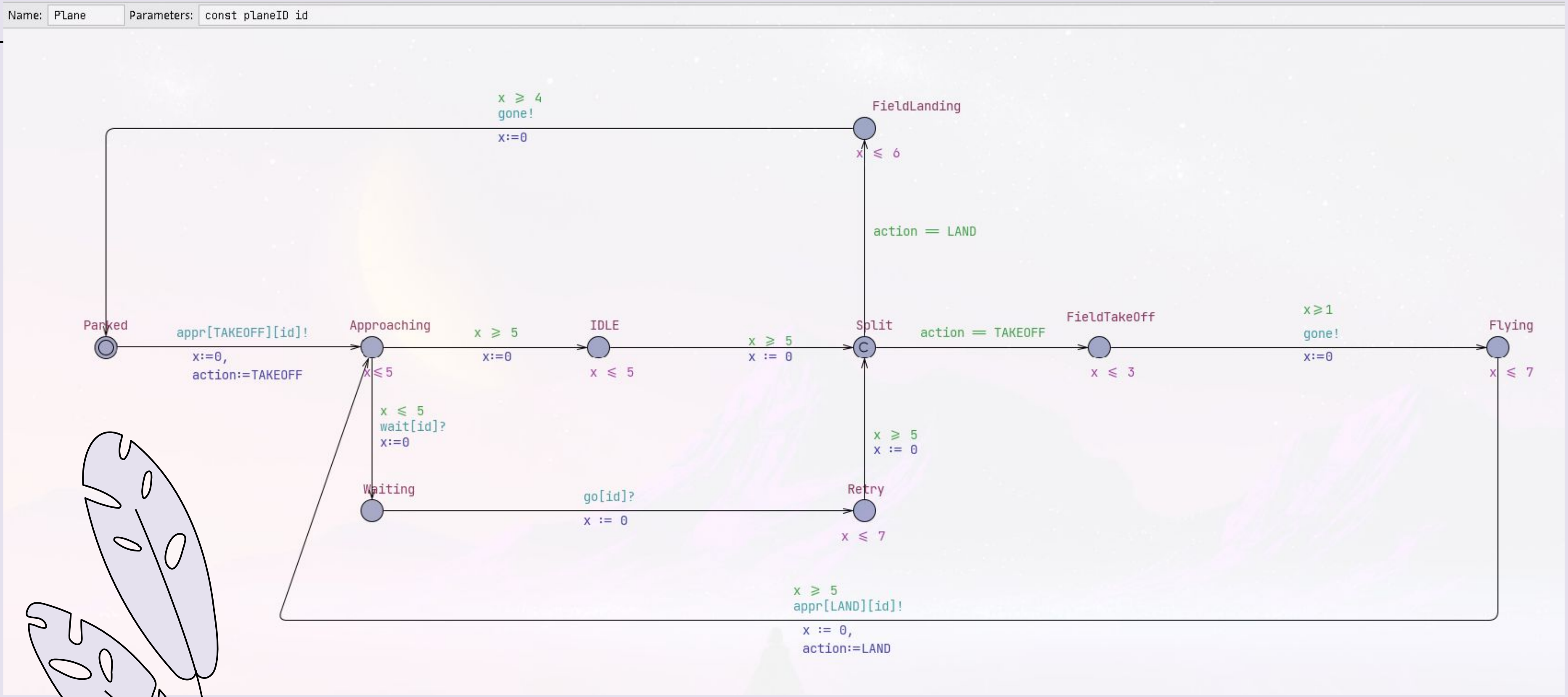
> “Testing can
> only find bugs, not
> prove their absence”

—Edsger Dijkstra

02

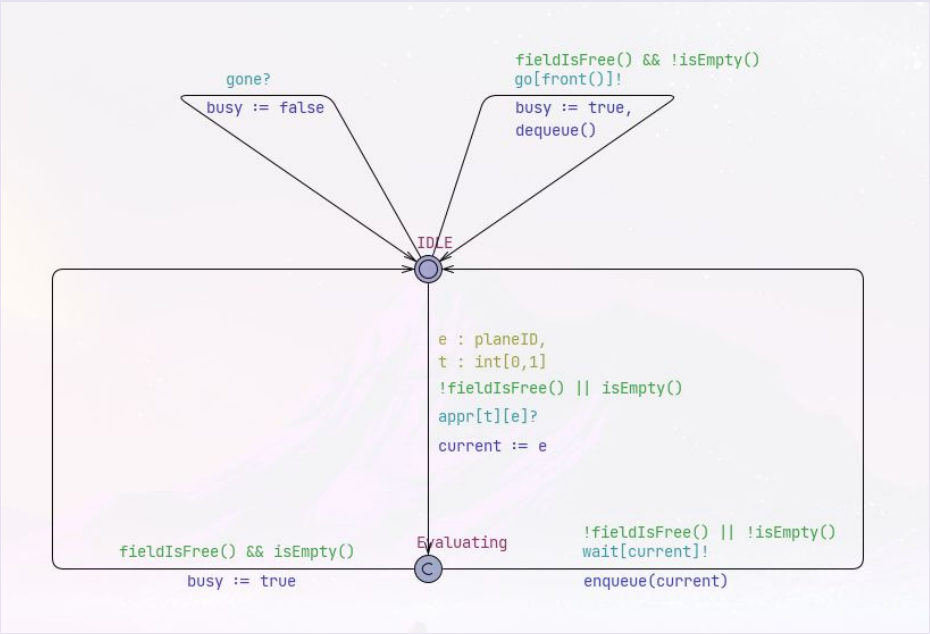
First Task

Managing shared resources with UPPAAL



Name: Plane

Parameters: const planeID id



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
C
!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

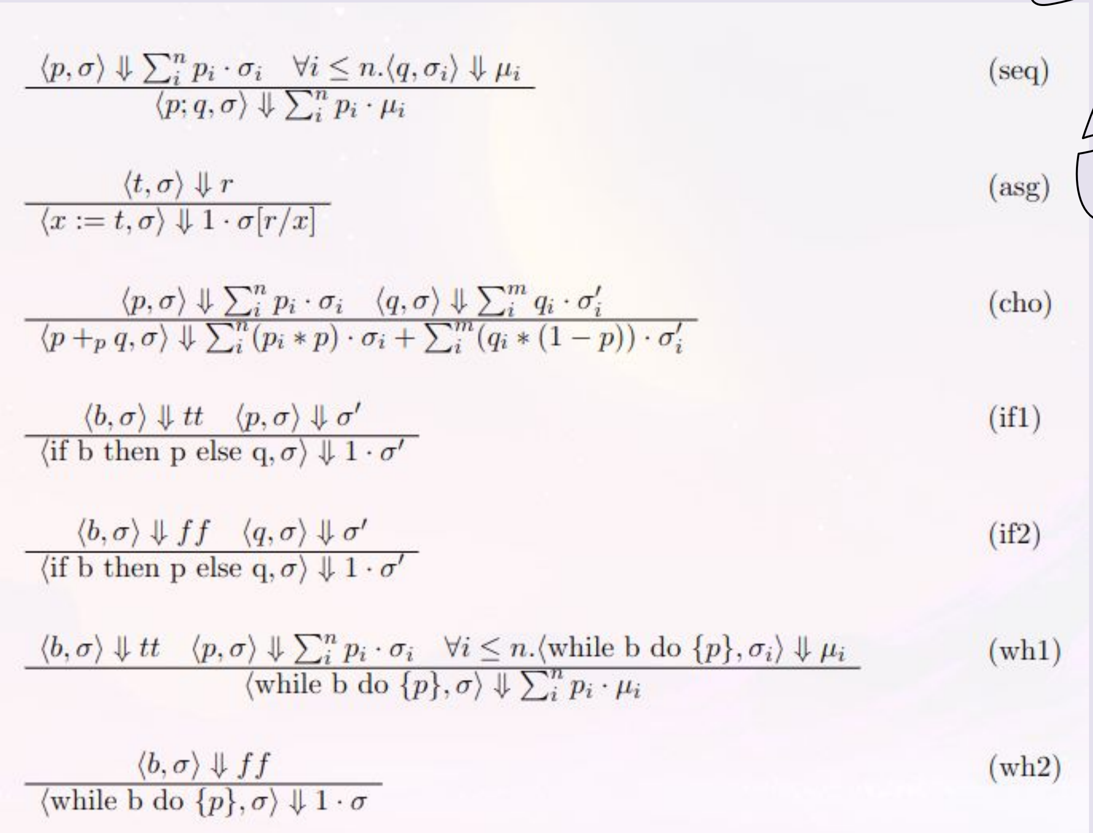
Using the right concepts in software development

04

Third Task

Unified program semantics

Semantics



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

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

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

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

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

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

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

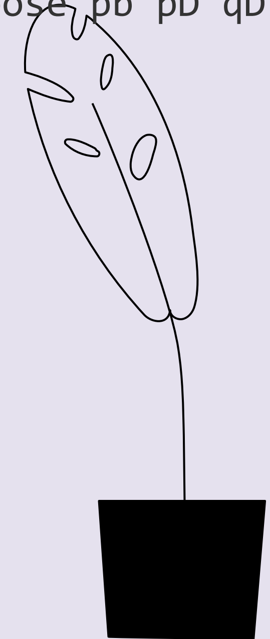
Data type

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

Main 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!