
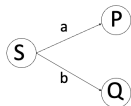

 UNIVERSIDADE  
FEDERAL RURAL  
DE PERNAMBUCO


 DEPARTAMENTO DE COMPUTAÇÃO  
UFRPE

## Operadores de escolha

Disciplina: Métodos Formais  
 Bacharelado em Ciência da Computação

Prof. Lucas Albertins  
[lucas.albertins@ufrpe.br](mailto:lucas.albertins@ufrpe.br)



```

graph LR
    S((S)) -- a --> P((P))
    S -- b --> Q((Q))
  
```

1

## Roteiro

- Operadores de escolha
- Algumas leis algébricas

2

## Operadores de Escolha

- Permitem criar ramificações no comportamento de um processo sequencial
- Escolha condicional  
if cond then P else Q
- Escolha externa  
 $P \parallel Q$
- Escolha interna (não-determinística)  
 $P \mid \sim \mid Q$

3

## Escolha Condicional

- Exemplo: contador de 0 até 60

```

Contador(min) =
  if min == 60 then
    cuco -> Contador(0)
  else
    tick -> Contador(min + 1)
  
```

4

## Escolha externa

- Exemplo: Especificação de um telefone celular

```
NUMEROS = {0..3}
channel liga, atendeu, ocupado, recusou : NUMEROS

T_LIVRE(n) =
  liga?x:diff(NUMEROS,{n}) ->
  (
    ocupado.x -> T_LIVRE(n)
    []
    atendeu.x -> T_OCUPADO(n,x)
    []
    recusou.x -> T_LIVRE(n)
  )

T_OCUPADO(n,n2) = ...
```

5

## Escolha externa

- Especifica um processo cujo comportamento é definido pela decisão (escolha) do ambiente

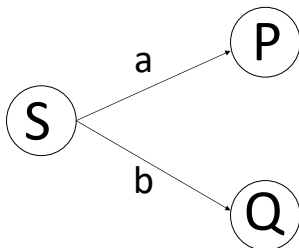
$$P \ [] \ Q$$

- O lado que comunica o primeiro evento progride após a escolha, isto é, processo  $P \ [] \ Q$  comporta-se como  $P'$  caso o lado esquerdo seja escolhido ( $Q'$  caso o lado direito seja escolhido) .

6

## LTS da escolha externa

Seja  $S = (a \rightarrow P) \ [] \ (b \rightarrow Q)$ , então



7

## Processo guardado

- O processo a seguir usa o parâmetro  $n$  para registrar o andar atual. Os eventos `up` e `down` denotam subir um andar e descer um andar. E  $L$  é o andar mais alto do prédio.

channel up, down

```
LCOUNT(L,n) =
  n<L & up -> LCOUNT(L,n+1)
  []
  n>0 & down -> LCOUNT(L,n-1)
```

8

## Processo guardado

- A expressão

`cond & P`

É equivalente a

`if cond then P else STOP`

9

## Especificação de um buffer

`DATA = {0,1}`

`channel in, out: DATA`

`B(s,N) =`

`#s < N & in?x -> B(s^<x>,N)`

`[]`

`#s > 0 & out!head(s) -> B(tail(s),N)`


Primeiro a entrar é o primeiro a sair (FIFO)

10

## Entrada e escolha

Considere que a declaração do canal `saque` é  
`channel saque : {0..10}`

A semântica da expressão `saque?v` corresponde a uma escolha externa

`saque?v -> P`  `saque.0 -> P[0/v]`  
`[]`  
`...`  
`[]`  
`saque.10 -> P[10/v]`

11

## Escolha interna

- $P \mid \sim \mid Q$  processo onde a escolha é realizada pelo processo (evento interno *tau* é comunicado)
  - ambiente não realiza a escolha
- Comporta-se como  $P$  ou como  $Q$ , escolha não determinística. A escolha acontece antes que os processos comuniquem algum evento visível

12

## Escolha interna

- Não determinismo é uma abstração
  - Deixa em aberto motivo da escolha
- Trocar  $| \sim |$  por  $[]$  refina um processo

13

## Escolha interna

```
MAQUINA_abstrata1 =
  umReal ->
  (
    leite -> MAQUINA_abstrata1
    |~|
    umReal -> cafe -> MAQUINA_abstrata1
  )
  |~|
  doisReais -> cafe -> MAQUINA_abstrata1
```

Tornando a especificação  
mais concreta

```
MAQUINA_abstrata2 =
  umReal ->
  (
    leite -> MAQUINA_abstrata2
    []
    umReal -> cafe -> MAQUINA_abstrata2
  )
  |~|
  doisReais -> cafe -> MAQUINA_abstrata2
```

```
MAQUINA = umReal ->
  (
    leite -> MAQUINA
    []
    umReal -> cafe -> MAQUINA
  )
  []
  doisReais -> cafe -> MAQUINA
```

14

## Escolha externa vs. escolha interna

```
(esquerda -> STOP) [] (direita -> STOP)
```

Externa: ambiente escolhe direita ou esquerda. Ambiente não falha quando tenta sincronizar com esquerda/direita.

```
(esquerda -> STOP) |~| (direita -> STOP)
```

Interna: processo escolhe direita ou esquerda. Ambiente pode falhar ao tentar sincronizar com esquerda (direita) se internamente o processo escolher pelo lado direito (esquerdo)

15

## Especificação da máquina de ATM

- 1ª versão: sem escolhas (sempre dá o dinheiro, independente do saldo)

```
ATM1 = incard?c -> pin.fpin(c) ->
  req?n -> dispense!n ->
  outcard.c -> ATM1
```

16

## Especificação da máquina de ATM2

- 2ª versão: escolha interna abstrai o motivo da máquina para dar o dinheiro ou negar o saque.

```
channel refuse
ATM2 =
  incard?c -> pin.fpin(c) -> req?n ->
    ( dispense!n -> outcard.c -> ATM2
      | ~ |
      refuse -> outcard.c -> ATM2
    )
```

17

## Especificação da máquina de ATM

- 3ª versão: possível refinamento

```
channel refuse
ATM3(saldo) =
  incard?c -> pin.fpin(c) -> req?n ->
    if(saldo-n >= 0) then (
      dispense!n -> outcard.c ->
        ATM3(saldo-n)
    ) else (
      refuse -> outcard.c -> ATM3(saldo)
    )
```

```
dispense!n -> outcard.c -> ATM2
|~|
Refuse -> outcard.c -> ATM2
```

Substituto  
para

18

## Verificação de determinismo

- Esta asserção de FDR verifica se um processo  $P$  é determinístico

```
assert P :[ deterministic ]
```

- Se resultado é **true**: o processo  $P$  é determinístico
- Se resultado é **false**: possui ao menos um ponto de não determinismo (FDR mostra o ponto mais próximo do estado inicial - busca em profundidade)

19

## Verificação de determinismo

| Processo  | Determinístico? |
|-----------|-----------------|
| ATM1      | true            |
| ATM2      | false           |
| ATM3(100) | true            |

20

## Não determinismo com outros operadores

- **Atenção:** mesmo quando o operador  $| \sim |$  não é utilizado um processo pode haver não determinismo
- Exemplo: o operador  $[]$  pode causar não determinismo:
  - Quando os processos na escolha oferecem eventos iniciais em comum

21

## Não determinismo com escolha externa

```
espera -> esquerda -> STOP
[]
espera -> direita -> STOP
```

← Não determinístico

é igual a

```
espera -> ( esquerda -> STOP
           |~|
           direita -> STOP
           )
```

22

## Leis algébricas

$$x + y = y + x$$

$$(x + y) + z = x + (y + z)$$

- Determinam a igualdade entre duas descrições de processos
- Forma de definir semântica
- Úteis para: entendimento, simplificação, comparação, prova

23

## Leis algébricas de CSP

|   |   |
|---|---|
| $P \square P = P$                                   | <b>Identidade (idempot.)</b>                    |
| $P \square Q = Q \square P$                         | <b>Simetria (comut.)</b>                        |
| $P \square (Q \square R) = (P \square Q) \square R$ | <b>Associatividade</b>                          |
| $STOP \square P = P$                                | <b>STOP é unidade para <math>\square</math></b> |

24

## Leis algébricas de CSP

$P \mid\sim P = P$  **Identidade (idempot.)**

$P \mid\sim Q = Q \mid\sim P$  **Simetria (comutat.)**

$P \mid\sim (Q \mid\sim R) = (P \mid\sim Q) \mid\sim R$  **Associatividade**

25

## Leitura e exercícios

- Livro: Theory and Practice of Concurrency
  - Leitura:
    - 1.1.5, 1.2
  - Exercícios:
    - Essenciais: 1.1.6, 1.1.8,
    - Opcionais: 1.1.9, 1.1.7, 1.2.6

26



27