

Introdução a CSP

Disciplina: Métodos Formais
Bacharelado em Ciência da Computação

Prof. Lucas Albertins
lucas.albertins@ufrpe.br

```
Contador(60) = cuco -> Contador(0)
Contador(min) = tick -> Contador(min + 1)
RELOGIO = Contador(0)
```

1

Roteiro

- Visão geral de CSP
- Comunicação
- Eventos e canais
- Operador de prefixo
- Recursão
- Processos parametrizados

2


Communicating Sequential Processes (CSP)

- Bom para modelar concorrência e distribuição
- Linguagem para especificação e propriedades é a mesma
 - Comparamos um processo que corresponde a propriedade com outro processo corresponde a especificação do sistema.


3

Nomes importantes

- Concebida por Sir Tony Hoare [85] (Microsoft Research) e atualizada por Bill Roscoe (Oxford University) [97]



Hoare
1980 ACM Turing Price Winner

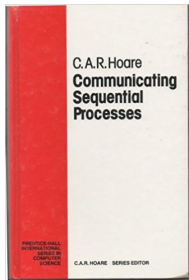


Roscoe

4

Primeiro livro de CSP

- Livro *Communicating Sequential Processes* de Tony Hoare (1985) um dos livros mais citados da Ciência da Computação



5

Principal livro gratuito

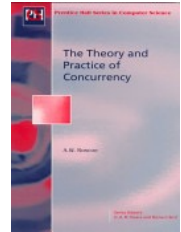
- Theory and Practice of Concurrency

Disponível para download em

<https://www.cs.ox.ac.uk/people/bill.r oscoe/publications/68b.pdf>

Exemplos do livro (arquivos .csp) em

<https://www.cs.ox.ac.uk/publications /books/concurrency/examples/booke xamples/>



6

Ferramentas

- Existem várias ferramentas para animação e verificação de refinamento automática
 - FDR, PAT, Probe, gCSP, Casper, ProB
- Inspirou linguagens de programação como Occam, Go, etc
- Inspirou bibliotecas para programação concorrente/distribuída
 - Java (JCSP, JTI)
 - Scala (CSO)
 - C++ (C++CSP2)

7

Notação para CSP

- Possui uma notação matemática (utilizada nos livros e artigos)
- Possui notação ASCII utilizada por ferramentas
 - Exemplo: CSP_M é a notação usada em FDR
 - Exemplo: CSP# é a notação usada em PAT
- Vamos usar CSP_M e FDR

8

Conceito: processo

- Processo é o principal elemento da notação
 - São componentes do sistema que comunicam **eventos**
 - Eventos são a parte visível dos processos (computações são internas ao processo)
- Um processo pode se **comunicar** com outros através de **sincronização de eventos** (visto mais na frente no curso)

9

Processo RELOGIO

channel cuco, tick

Declaração de eventos

Contador(60) = cuco -> Contador(0)
 Contador(min) = tick -> Contador(min + 1)
 RELOGIO = Contador(0)

Nome do processo

Operador de prefixo

10

Casamento de padrão ou if/else

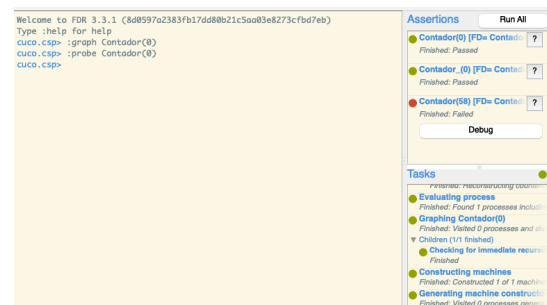
- Processo Contador definido em um estilo funcional (casamento de padrão)


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


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

11

Analisando com FDR



12

- **Sintaxe**
 - Operadores para combinar processos
 - Linguagem funcional para computação de valores
- **Semântica dos processos**
 - Labelled Transition Systems (LTS)
 - Conjunto de eventos, falhas, divergências

13

CSP_M - Resumo da sintaxe

P = STOP	Processo que está em deadlock
SKIP	Processo que termina com sucesso
a -> Q	Operador de prefixo
Q [] R	Escolha externa
Q ; R	Composição sequencial
Q [X] R	Composição paralela generalizada
Q R	Entrelaçamento de processos (interleaving)
Q \ X	Operador de esconder (Hiding)
Q /\ R	Interruption
let .. within	Definições locais

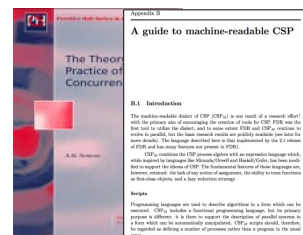
14

Principais funções pré-definidas de CSP_M

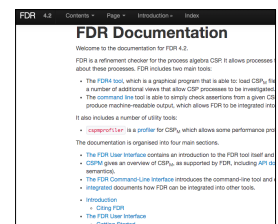
Sequences	Sets	Booleans
<i>Syntax</i>	<i>Syntax</i>	<i>Syntax</i>
$\langle \rangle, \{1,2,3\}$	$\{1,2,3\}$	true, false
$\langle m..n \rangle$	$\{m..n\}$	b_1 and b_2
$\langle m.. \rangle$	$\{m.. \}$	b_1 or b_2
s^*t	$\text{union}(a_1, a_2)$	not b
$\#s, \text{length}(s)$	$\text{inter}(a_1, a_2)$	$x_1 == x_2, x_1 != x_2$
$\text{null}(s)$	$\text{diff}(a_1, a_2)$	$x_1 < x_2, x_1 > x_2, x_1 \leq x_2, x_1 \geq x_2$
$\text{head}(s)$	$\text{Union}(A)$	if b then x_1 else x_2
$\text{tail}(s)$	$\text{Inter}(A)$	
$\text{concat}(s)$	$\text{member}(x, a)$	
$\text{elem}(x, s)$	$\text{card}(a)$	
$\langle x_1, \dots, x_n \mid x < s, b \rangle$	$\text{empty}(a)$	
	$\text{set}(s)$	
	$\text{Set}(s)$	
	$\text{Seq}(a)$	
	$\{x_1, \dots, x_n \mid x < a, b\}$	

15

Resumo da sintaxe



Apêndice B do livro



Manual de FDR no Site

16

Eventos

- Representam acontecimentos relevantes para a especificação
- Exemplos:
 - Fatos/ações perceptíveis (alarme tocou, serviu café, colocou moeda, pressionou uma tecla, etc)
 - Entrada/saída de valores (saque.150, cedula.10, resultado.-45, etc)
 - Sinais de sincronização (espere, continue, etc)

17

Comunicação de um evento

- Quando um evento acontece dizemos que o mesmo foi comunicado
- Eventos são:
 - Instantâneos (comunicação acontece sem ter duração)
 - Atômicos (nada acontece durante o evento, apenas antes ou depois)

18

Operador de Prefixo (->)

- Usado para modelar comportamento sequencial
- Exemplo:


```
ANDAR_INFINITO = esquerda -> direita ->
ANDAR_INFINITO

ANDAR = esquerda -> direita ->
STOP
```

21

Regras para uso de prefixo

- Antes de -> : deve haver um evento
- Após -> : deve haver um processo
- Formas inválidas:


```
P = Q -> esquerda ->
P = esquerda -> P -> direita
```

22

Processo STOP

- Processo pré-definido
- Não comunica eventos e não progride
- Representa deadlock
 - Um estado terminal de onde não é possível sair

23

Deadlock

- Em CSP, o deadlock pode ocorrer em duas situações
 1. Propositamente, com o uso do processo STOP
 2. Situação de impasse
 - Quando todos os processos entram em estado de espera



24

Livre de deadlock

- Quando um processo nunca chega em um estado terminal ele é livre de deadlock
- Com FDR, vemos que Contador(60) está livre de deadlock verificando a asserção

```
assert Contador(60) :[deadlock free]
```

25

Comportamento do processo

- O comportamento muda a medida que o processo progride.
- Exemplo:

```
P0 = up -> down -> up -> down -> STOP
```

```
após comunicar up se comporta como
```

```
down -> up -> down -> STOP
```

```
após comunicar down se comporta como
```

```
up -> down -> STOP
```

```
após comunicar up e down se comporta como
```

```
STOP
```

26

Canal

- Usado para comunicar valores
 - Define um conjunto de eventos com valores associados
- Exemplo: canal `opcao` comunica os valores 0,1,2 e 3


```
channel opcao : {0..3}
```
- O canal `opcao` define o conjunto de eventos


```
{opcao.0, opcao.1, opcao.2, opcao.3}
```
- Em FDR, a expressão `{|opcao|}` retorna todos os eventos do canal `opcao`

27

Canal - Exemplo

- Exemplo: a seguir, o valor `b` na expressão `escolhe?b` é um tipo de bebida escolhido pelo ambiente.

```
datatype BEBIDA = CAFE | CHA | REFRIGERANTE

channel escolhe, serve : BEBIDA

MAQUINA_BEBIDA = escolhe?b -> serve!b ->
MAQUINA_BEBIDA
```

28

Canal - Exemplo

- Exemplo: o canal `valor` pode comunicar valores entre `MIN` e `MAX` enquanto a soma é menor do que 30

```
MIN = 0 -- constante inteira
MAX = 5 -- constante inteira

channel valor : {MIN..MAX}

VALORES(sum) = if sum <= 30 then
                valor?v -> VALORES(sum+v)
              else
                STOP

INIT = VALORES(0)
```

29

Canal - Sintaxe

- Considere $T = \{v_1, \dots, v_n\}$ é um tipo e declaramos o canal `ch` a seguir

```
channel ch : T
```

- Três expressões podem ser usadas com `ch`

`ch?var` lê um valor `var` do tipo `T` do ambiente ("entrada" de dados)

`ch!exp` comunica uma valor `exp` do tipo `T` ("envio" ou "saída")

`ch.exp` o mesmo que `ch!exp`

30

Cardinalidade de tipos causa explosão de estados

- Exemplo de tipo que causa explosão de estados

```
channel ch : Int
P = ch?n -> STOP
```

- Outro exemplo cuja cardinalidade é menor

```
channel ch: {1..10}
P = ch?n -> STOP
```

31

Cardinalidade de tipos causa explosão de estados

- Quanto maior a cardinalidade mais custosa será a análise da expressão $ch?v$
- Recomendação:
 - Procure utilizar conjuntos pequenos (tamanho suficiente)

32

Especificação ATM

```
CARD = {0..9}
datatype pinnumbers = PIN.Int
fpin(c) = PIN.c
PINs = {fpin(c) | c <- CARD}
WA = {10,20,30,40,50}
```

```
channel incard, outcard: CARD
channel pin: PINs
channel req, dispense: WA
```

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

Definição de
constante. Conjunto
de 0 até 9.

Definição de tipo

Definição de função

Compreensão de
conjunto

34

Não esqueça de dar um limite para a recursão

- Recursões podem gerar processos com infinitos estados
- FDR só consegue analisar quantidade finita de estados
- Exemplo: No processo a seguir valor de n não tem limite superior, portanto o comando `:graph Contador(0)` nunca termina

```
Contador(min) = tick -> Contador(min+1)
```

Se colocar a expressão `Contador(60) = cuco -> Contador(0)` antes da definição acima, limitamos o valor de min em 60.

35

Não esqueça de dar um limite para a recursão

- O que acontece com o processo INIT se no lugar de `sum <= 30` colocarmos `true`?

```
MIN = 0 -- constante inteira
MAX = 5 -- constante inteira

channel valor : {MIN..MAX}

VALORES(sum) = if sum <= 30 then
                valor?v -> VALORES(sum+v)
            else
                STOP

INIT = VALORES(0)
```

36

Máquinas de estados e CSP

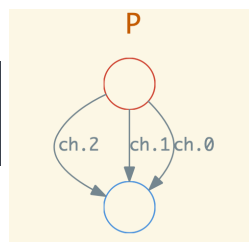
- Todo processo em CSP pode ser transformado em um Sistema Rotulado de Transições (LTS - Labelled Transition Systems)
- FDR verifica processos cujo LTS tenha um número finito de estados (a quantidade de caminhos pode ser infinita)
- A opção `:graph` é usada em FDR3 para desenhar o LTS do processo CSP

37

Máquinas de estados e CSP

- Resultado do comando `:graph P` no lado direito

```
channel ch: Int
P = ch?n:{0,1,2} -> STOP
```



38

Prática

- Especifique processos que detalham o que você faz em três dias da semana
 - Lembre-se que você pode usar processos dentro de outros
- Crie um processo que representa o uso de uma máquina de café que recebe moedas e o usuário pode escolher entre café, cappuccino e chocolate quente.

39

Referência teórica

- Livro: Theory and Practice of Concurrency
 - Leitura:
 - Cap 0
 - Cap 1 até 1.1.2

42



43