



UFCG – CEEI – DSC – CCC Grupo PET Computação Ciclo de Seminários

Linguagem formal CSP: Uma alternativa ao teste de software.

Natasha Bezerra
natasha.silva@ccc.ufcg.edu.br
Outubro 2012

Agenda

- Surgimento da linguagem
- Principais elementos de CSP
- Operadores
- Semântica
- Refinamentos
- Ferramentas
- Aplicação
- Referências

Surgimento da linguagem

- Hoare (1985) e Roscoe (1998)
- Linguagem formal para modelar aspectos comportamentais
- Operadores Algébricos
- Análise de propriedades (Deadlock, Livelock e Determinismo)

Principais elementos de CSP

- **Processo** - entidades básicas que capturam um comportamento.

Processos primitivos em CSP: *STOP* e *SKIP*.

$P(s) ::=$	$STOP$	
	$SKIP$	
	$a \rightarrow P(s)$	(<i>prefixo</i>)
	$P(s)$	(<i>recursao</i>)
	$g \ \& \ P(s)$	(<i>escolha condicional</i>)
	$P(s) \ \square \ P(s)$	(<i>escolha externa</i>)
	$P(s) \ \sqcap \ P(s)$	(<i>escolha interna</i>)
	$P(s) \setminus C$	(<i>internalizacao</i>)
	$P(s)[R]$	(<i>renomeacao</i>)
	$P(s) \ ; \ P(s)$	(<i>composicao sequencial</i>)
	$P(s) \ \triangle \ P(s)$	(<i>interrupcao</i>)
	$P(s) \ \ P(s)$	(<i>paralelismo</i>)
	C	

Principais elementos de CSP

- Datatype (datatype D)
- Comunicação
 - Evento (tic e tac de um relógio)
 - Canal (canal *debitar* entre os processos CAIXA e SERVIDOR)
- Alfabeto ($\text{Alfabeto} = \bigcup \text{Eventos}$)

Operadores

- Escolha Condicional
- Prefixo
- Recursão
- Escolha Interna e Externa
- Comp. Paralela
- ...

Processo	Operadores
$a \rightarrow P$	Pré-fixo
$P \square Q$	Escolha Externa
$P \sqcap Q$	Escolha Interna
$P \parallel Q$ X	Paralelismo
$P \text{ III } Q$	Interleaving
$P \setminus X$	Hiding
$P \Leftarrow b \Rrightarrow Q$	Se Então Senão

Semântica

Conjunto de leis algébricas que permitem provar equivalências semânticas (por meio de refinamentos) entre processos sintaticamente diferentes.

- Modelo de *Traces*
- Modelo de Falhas
- Modelo de Falhas e Divergências

$$\text{traces}(P \sqcap Q) = \text{traces}(P) \cup \text{traces}(Q)$$

$$\text{failures}(P \sqcap Q) = \text{failures}(P) \cup \text{failures}(Q)$$

$$(\text{failures}^\perp(P), \text{divergences}(P))$$

Refinamentos

Nova forma de implementar ou modelar o sistema sem nenhum impacto.

Relações de refinamento:

- Refinamentos por traces

$(P \sqsubseteq_T Q)$ se e somente se $\text{traces}(Q) \subseteq \text{traces}(P)$

$$P =_T Q \equiv P \sqsubseteq_T Q \wedge Q \sqsubseteq_T P$$

- Refinamentos por falhas

$(P \sqsubseteq_F Q)$ se e somente se $\text{traces}(Q) \subseteq \text{traces}(P) \wedge \text{failures}(Q) \subseteq \text{failures}(P)$

- Refinamentos por falhas e divergências

$$P \sqsubseteq_{FD} Q \equiv \text{failures}_\perp(Q) \subseteq \text{failures}_\perp(P) \wedge \text{divergences}(Q) \subseteq \text{divergences}(P)$$

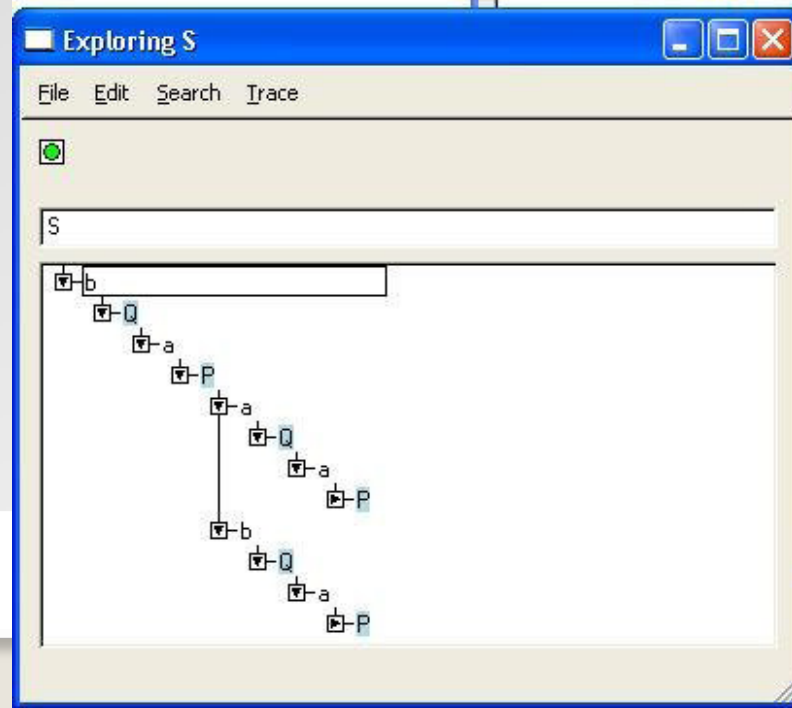
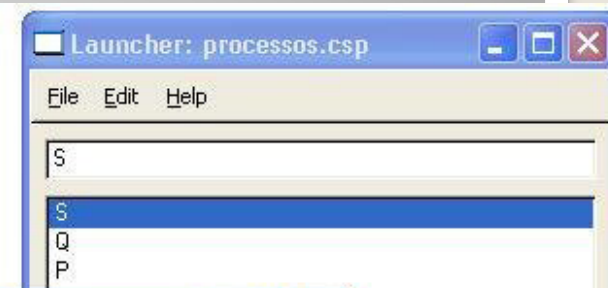
Ferramentas

- Análise automática
- Existem duas abordagens para estudar um processo:
 - Propriedades padrão (deadlock, livelock, não-determinismo)
 - Refinamento entre processos

Ferramentas

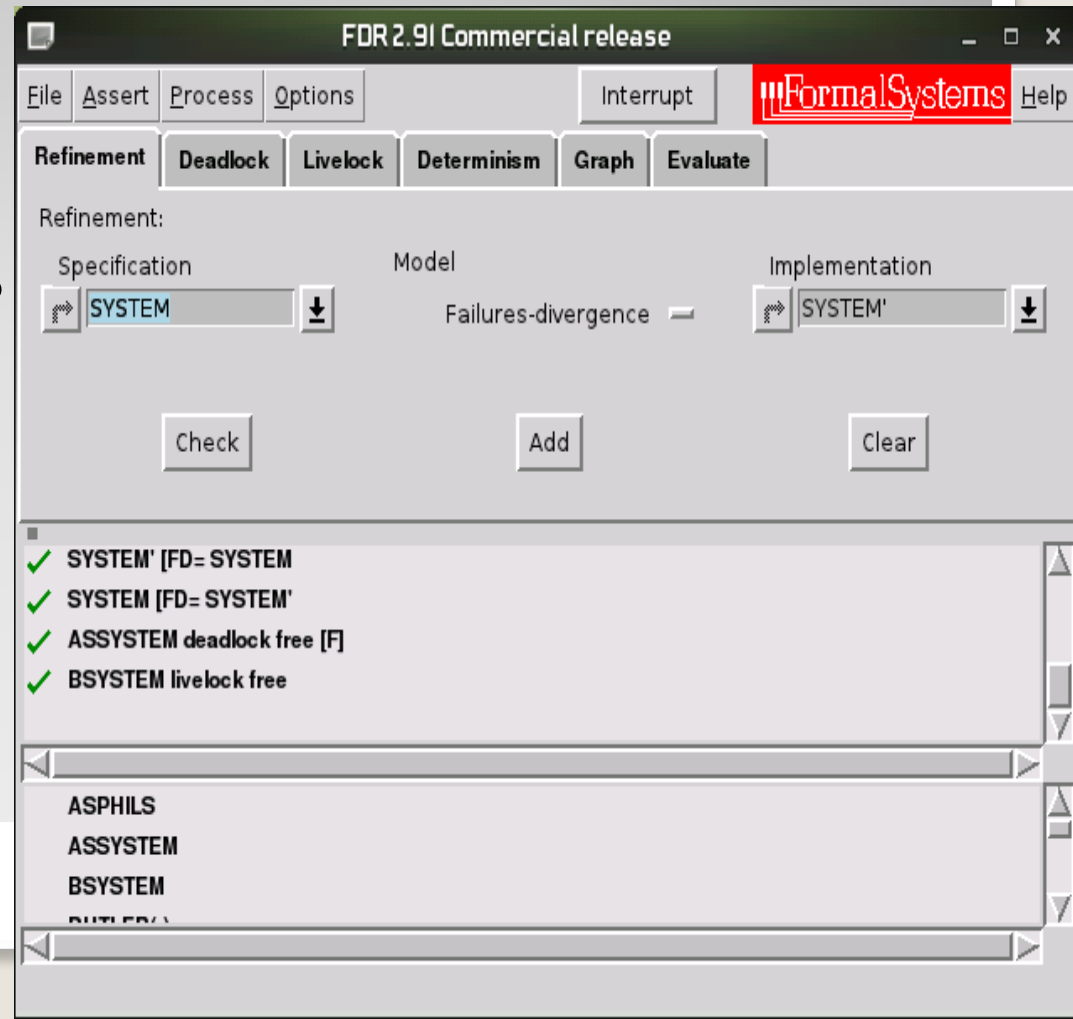
- ProBE:
animar os modelos.

```
channel a, b  
  
P = (a -> Q) [] (b -> Q)  
Q = a -> P  
S = b -> Q
```



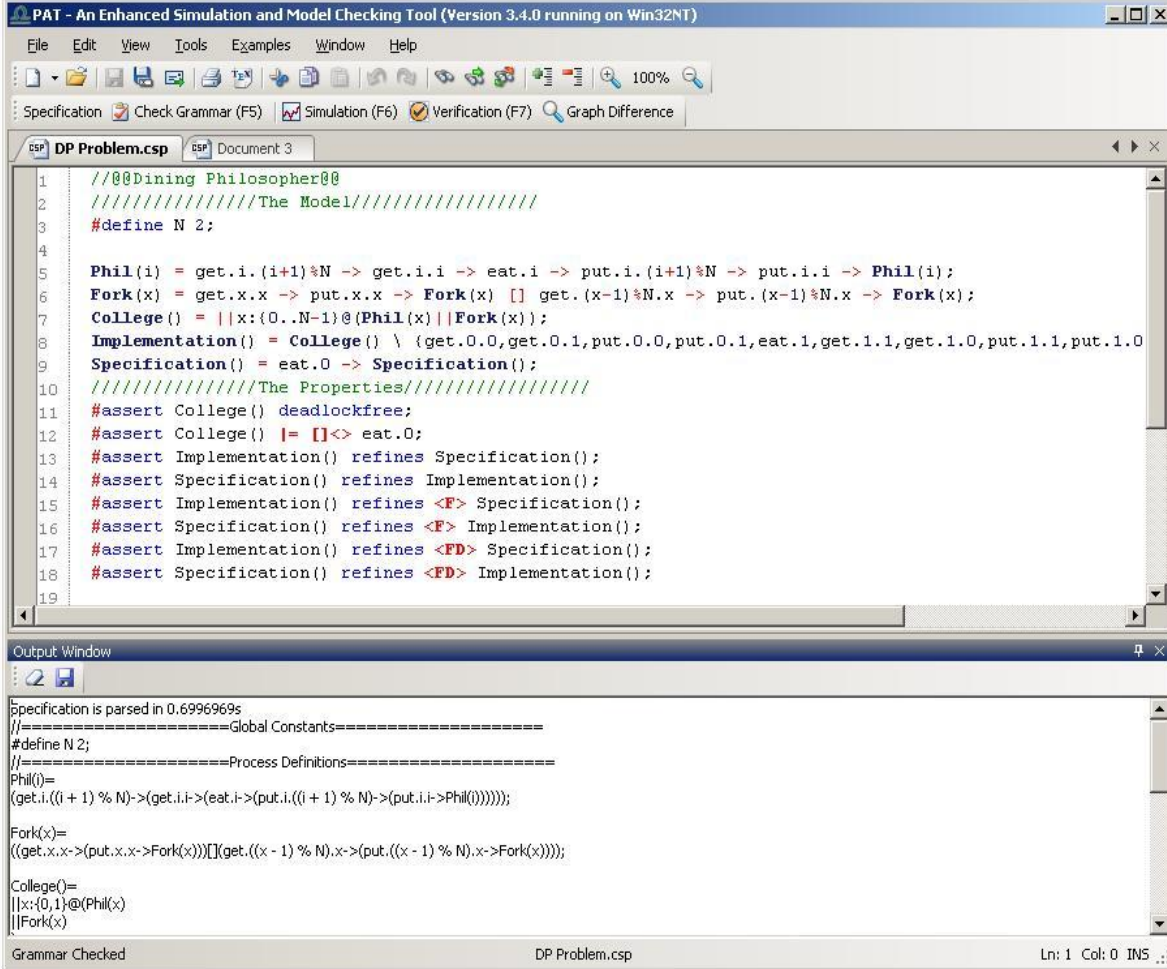
Ferramentas

- FDR:
testar afirmações
sobre processos,
como equivalências
entre processos e
existência de
deadlocks,
livelocks ou
não-determinismo.



Ferramentas

- PAT:
ferramenta
de simulação,
verificação de
modelo e
refinamento
para sistemas
concorrentes
e de tempo real.



The screenshot displays the PAT (Process Analysis Tool) interface, version 3.4.0, running on Win32NT. The main window shows a CSP specification for a Dining Philosopher problem. The specification includes process definitions for Phil(i), Fork(x), and College(), along with an Implementation() and Specification() function. The bottom panel shows the output window with the parsed specification and the results of the verification process.

```
//@@Dining Philosopher@@
//=====The Model=====
#define N 2;

Phil(i) = get.i.(i+1)*N -> get.i.i -> eat.i -> put.i.(i+1)*N -> put.i.i -> Phil(i);
Fork(x) = get.x.x -> put.x.x -> Fork(x) [] get.(x-1)*N.x -> put.(x-1)*N.x -> Fork(x);
College() = ||x:{0..N-1}@ (Phil(x) || Fork(x));
Implementation() = College() \ {get.0.0, get.0.1, put.0.0, put.0.1, eat.1, get.1.1, get.1.0, put.1.1, put.1.0};
Specification() = eat.0 -> Specification();

//=====The Properties=====
#assert College() deadlockfree;
#assert College() != [] <> eat.0;
#assert Implementation() refines Specification();
#assert Specification() refines Implementation();
#assert Implementation() refines <F> Specification();
#assert Specification() refines <F> Implementation();
#assert Implementation() refines <FD> Specification();
#assert Specification() refines <FD> Implementation();
```

Output Window:

```
Specification is parsed in 0.6996969s
//=====Global Constants=====
#define N 2;
//=====Process Definitions=====
Phil(i)=
((get.i.(i+1)*N)->((get.i.i->(eat.i->(put.i.(i+1)*N)->(put.i.i->Phil(i))))));
Fork(x)=
((get.x.x->(put.x.x->Fork(x))[]((get.((x-1)*N).x->(put.((x-1)*N).x->Fork(x))));
College()=
||x:{0..1}@ (Phil(x) || Fork(x))
```

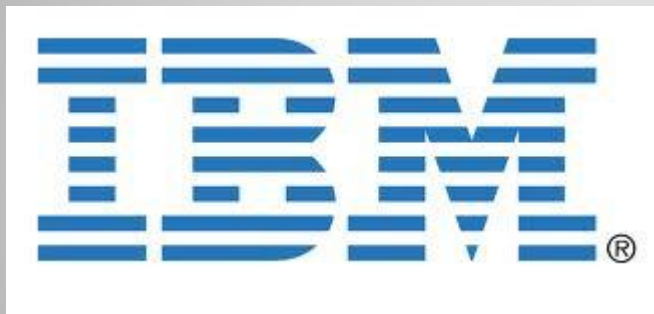
Grammar Checked DP Problem.csp Ln: 1 Col: 0 INS

Ferramentas

É importante salientar que existem diferenças entre os dialetos de CSP aceitos pelo FDR e pelo PAT.

Estrutura	CSPFDR	CSPPAT
Valor bolleano	true false	true false
Processos primitivos	STOPSKIP	Stop Skip
Conjunção lógica	b1 and b2	b1 && b2
Disjunção lógica	b1 or b2	b1 b2
Negação	not b	! b
Comparação	x1==x2	x1==x2
Diferença (booleana)	x1!=x2	x1!=x2
Menor que	x1<x2	x1<x2
Menor ou igual	x1<=x2	x1<=x2
Maior que	x1>x2	x1>x2
Maior ou igual	x1>=x2	x1>=x2
Expressão condicional	if b then x1 else x2	if (b) {x1} else {x2}
Canais	channel b	
Prefixo	c -> p	c -> p
Escolha externa	P[]Q	P [*] Q
Escolha interna	P ~ Q	P <> Q
Paralelismo	P Q	P Q
Interleaving	P Q	P Q
Hiding	P \ A	P \ A
Processos com parametros	P(x)	P(x)
Operadores	* / % + -	* / % + -
Interrupção	P /\ Q	
Composição Sequencial	P;Q	P;Q
Conjunto literal	{}	
Conjunto com intervalo fechado	{m..n}	{m..n}
Conjunto com intervalo aberto	{m..}	{m..}

Aplicação



Referência

- A. W. Roscoe. *The theory and Practice of Concurrency*. Prentice Hall, 1998.
- C.A.R Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.
- Formal Systems (Europe) Ltd. *FDR: User Manual and Tutorial*, version 2.80, 2003.
- PAT. PAT: The Process Analysis Toolkit, 2009. Disponível em: <http://www.comp.nus.edu.sg/pat>
- http://cgibin.erols.com/ziring/cgi-bin/cep/cep.pl?_key=CSP
- <http://www.cin.ufpe.br/~tg/2007-1/fmmf.pdf>
- <http://home.comcast.net/~refilman/text/dpl/csp.pdf>
- http://www.iist.unu.edu/~cbertolini/papers/6TLA3_09Bertolini.pdf

Dúvidas?





UFCG – CEEI – DSC – CCC

Grupo PET Computação

Ciclo de Seminários

**Linguagem formal CSP:
Uma alternativa ao teste de software.**

Natasha Bezerra
natasha.silva@ccc.ufcg.edu.br
Outubro 2012