




Visão geral dos modelos semânticos

Disciplina: Métodos Formais
Bacharelado em Ciência da Computação

Prof. Lucas Albertins
lucas.albertins@ufrpe.br

$$\begin{array}{c}
 P \approx_M Q \\
 \updownarrow \\
 P \models M=Q \text{ e } Q \models M=P
 \end{array}$$

1

Roteiro

- Modelos semânticos
- Igualdade vs refinamento
- Propriedades de um refinamento
- Modelo de traces

2

Modelos Semânticos

- Um modelo semântico é um conjunto de observações para um processo
- Existem vários modelos semânticos em CSP, por exemplo:
 - Traces (T)
 - Falhas (F)
 - Falhas e Divergências (FD)
 - Refusal Testing (RT)
 - etc

3

Modelos Semânticos

- Exemplo: $\text{traces}(P)$ denota o modelo de traces de um processo P , considere

$$P = a \rightarrow b \rightarrow c \rightarrow \text{STOP} \quad [] \quad d \rightarrow e \rightarrow \text{STOP}$$

$$\begin{array}{l}
 \text{traces}(P) = \{ \\
 \quad \langle \rangle, \langle a \rangle, \langle a, b \rangle, \langle a, b, c \rangle, \langle d \rangle, \langle d, e \rangle \\
 \}
 \end{array}$$

4

Modelos Semânticos

- Sintaxe *versus* Semântica
 - Um processo tem uma (única) semântica
 - Processos sintaticamente diferentes podem ter a mesma semântica
- Ex: P e Q tem a mesma semântica

```

P = a -> b -> P      traces(P) = traces(Q) =
Q = a -> Q2           { <>, <a>, <a,b>, <a,b,a>, <a,b,a,b>,
Q2 = b -> Q           <a,b,a,b,a>, ...
                      }

```

5

Principais modelos semânticos de CSP

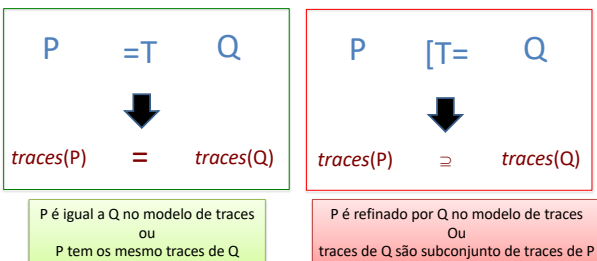
- Traces (T)
 - Registra as sequências de eventos que PODEM ser observadas
Propriedades safety (nada ruim acontece)
- Failures (F)
 - Registra traces e também falhas (o que o processo pode recusar fazer). Não-determinismo e deadlock são registrados.
Propriedades liveness (o que deve acontecer)
- Failures-divergences (FD)
 - Registra traces/falhas e também traces onde acontece livelock (divergência)

As leis algébricas de CSP consideram o modelo FD

6

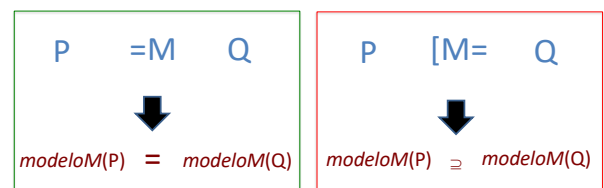
Modelos Semânticos

- Estabelecem igualdade e refinamento entre processos
- Exemplo: igualdade e refinamento de traces



7

Igualdade e refinamento



Refinamento/igualdade é representado substituindo M por um dos modelos semânticos T, F, FD

8

Especificação vs. implementação

Especificação [M= Implementação

O lado esquerdo, pode ser mais abstrato, é chamado de especificação

O lado direito, pode ser mais concreto, é chamado de implementação

9

Refinamento em FDR

- Noções de refinamento:

$P \sqsubseteq T = Q$

$P \sqsubseteq F = Q$

$P \sqsubseteq FD = Q$

10

Verificando refinamento em FDR

- O comando `assert` de FDR é usado (entre outras coisas) para verificar refinamento entre processos
- Exemplo: o primeiro refinamento a seguir não é válido, FDR retorna `<a>` como contraexemplo

```
assert STOP [T= a -> STOP
assert a -> STOP [T= STOP
```

```
traces(STOP) = {<>}
traces(a -> STOP) = {<>, <a>}
```

11

Contraexemplo

- Se $\text{not}(P \sqsubseteq M = Q)$ então existe (ao menos) um comportamento que pertence ao modelo de Q e não pertence ao modelo de P
 - Este exemplo é chamado de contraexemplo
 - FDR retorna o menor contraexemplo de um refinamento que não é válido

12

Refinamento vs. equivalência

$$P =_M Q$$



$$P [M= Q \text{ e } Q [M= P$$

P é igual a Q no modelo M se, e somente se, Q refina P e P refina Q

13

Refinamento vs. equivalência

$P = a \rightarrow b \rightarrow P$
 $Q = a \rightarrow Q2$
 $Q2 = b \rightarrow Q$

Temos que

$P [T= Q$
 $Q [T= P$

$\text{traces}(P) = \text{traces}(Q) =$
 $\{ \langle \rangle, \langle a \rangle, \langle a, b \rangle, \langle a, b, a \rangle, \langle a, b, a, b \rangle, \dots \}$

Isto implica que

$P T= Q$

14

Relação entre modelos

- $P [FD= Q \Rightarrow P [F= Q \Rightarrow P [T= Q$

(quando P e Q são livres de divergência)

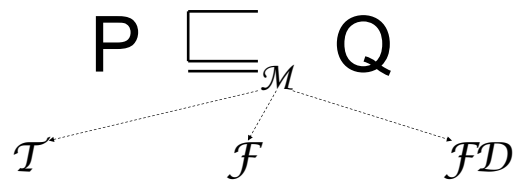
- A implicação acima não é verdade na direção oposta
 - Exemplo, pode refinar em T mas não em F como a seguir

$a \rightarrow \text{STOP} [T= \text{STOP}$
 $\text{not } (a \rightarrow \text{STOP} [F= \text{STOP})$

15

Símbolos matemáticos para modelos

- A seguir, símbolos que representam os refinamentos nos diversos modelos na notação matemática utilizada no livro de CSP




16

Complexidade da análise dos modelos

- Modelos mais ricos são mais custosos de serem analisados

Traces (\mathcal{T}) Failures (\mathcal{F}) Failures-divergences (\mathcal{FD})

Grau crescente de precisão/custo de análise 

17

Traces

- $\text{traces}(P)$ é o conjunto de todas as histórias (traces) do processo P :

```
traces(a -> b -> STOP) =
  {<>, <a>, <a, b>}
traces(a -> STOP [] b -> STOP) =
  {<>, <a>, <b>}
traces(μX.a -> X) =
  {<>, <a>, <a, a>, <a, a, a>, ...}
```

21

Calculando Traces

```
traces(STOP) = {<>}

traces(a -> P) = {<>} ∪
  {<a>^s | s ∈ traces(P)}

traces(c?x:A -> P) =
  {<>} ∪
  {<c.a>^s | a ∈ A, s ∈ traces(P[a/x])}

traces(P [] Q) = traces(P) ∪ traces(Q)

traces(P |~| Q) = traces(P) ∪ traces(Q)
```

22

Refinamento de traces

- Permite que a implementação tenha menos traces do que a especificação (inclusive nenhum traces)

- Exemplos:

```
P [T= STOP
up -> down -> STOP [T= up -> STOP
not (up -> STOP [T=
up -> STOP [] down -> STOP)
```

23

Conceito: alfabetos

- αP denota o alfabeto de um processo P
- O alfabeto é o conjunto dos eventos que podem ser comunicados pelo processo
 - Exemplo: $\alpha \text{ATM1} = \{\text{incard.0}, \dots, \text{incard.9}, \text{pin.PIN.0}, \dots, \text{pin.PIN.9}, \text{req.10}, \dots, \text{req.50}, \text{dispense.10}, \dots, \text{dispense.50}, \text{outcard.0}, \dots, \text{outcard.9}\}$
- Σ denota o alfabeto de uma especificação: a união dos alfabetos de todos os processos
 - `Events` é uma constante em FDR que representa Σ

24

Processo RUN

- Comunica todas as sequência possíveis de eventos (traces) a partir do conjunto $X = \{e_1, e_2, \dots, e_n\}$ (alfabeto)

$$\text{RUN}(X) = [] \ x : X \ @ \ x \rightarrow \text{RUN}(X)$$

O processo é equivalente a

$$\begin{aligned} \text{RUN}(\{e_1, e_2, \dots, e_n\}) = \\ e_1 \rightarrow \text{RUN}(X) \\ [] \ e_2 \rightarrow \text{RUN}(\{e_1, e_2, \dots, e_n\}) \\ \dots \\ [] \ e_n \rightarrow \text{RUN}(\{e_1, e_2, \dots, e_n\}) \end{aligned}$$

25

Processo RUN

- $\text{traces}(\text{RUN}(X)) = X^*$
- Em FDR $\text{RUN}(\text{Events})$ comunica todas as sequências possíveis de eventos declarados em um arquivo `.csp`

26

STOP vs. RUN

- No modelo de trace, `STOP` é uma implementação de qualquer processo

$$P \ [T= \text{STOP}]$$

- O processo $\text{RUN}(\text{Events})$ é uma especificação para qualquer processo

$$\text{RUN}(\text{Events}) \ [T= P]$$

28

Traces verifica propriedades safety

- **Verifica**
 - Se trace (não) **pode** acontecer
- **Não verifica**
 - Se trace **sempre** acontece (liveness)

29

Verificando se evento (não) pode acontecer

```
RUN(diff(Events, {falha}))
```

- `RUN(diff(Events, {falha}))` faz todos os traces considerando os alfabetos dos processos no arquivo .csp, menos traces que tem o evento falha

31

Verificando se evento (não) pode acontecer

```
assert RUN(diff(Events, {falha})) [T= P
```

- **falha** é um evento que queremos saber se acontece ou não
- **P** é o processo onde queremos saber se falha acontece ou não

32

Verificando se evento (não) pode acontecer

```
assert RUN(diff(Events, {falha})) [T= P
```

- **Se a verificação for falsa**, então o evento **falha** pode acontecer em P
 - o contraexemplo mostra um trace onde o evento acontece
- **Se a verificação for verdadeira**, então o evento **falha** não pode acontecer em P

33

Verificando se evento (não) pode acontecer

```
assert RUN(diff(Events,{falha})) [T= P
```

Atenção:

Certifique-se que `Events` é finito.

Caso contrário FDR não vai poder analisar.

34

Verificando se evento (não) pode acontecer

- Exemplo:

$P = a \rightarrow b \rightarrow \text{falha} \rightarrow P$

$Q = a \rightarrow b \rightarrow Q$

<code>assert RUN(diff(Events,{falha})) [T= P</code>	False
<code>assert RUN(diff(Events,{falha})) [T= Q</code>	True

35

Verificando se trace (não) pode acontecer

- A expressão

```
assert P :[has trace [T]]: <e1,..., eN>
```

é verdadeira se, e somente se, P **pode** comunicar o trace $\langle e1, \dots, eN \rangle$

36

Verificando se trace (não) pode acontecer

- Exemplo:

$P = \text{up} \rightarrow \text{down} \rightarrow P$

<code>assert P :[has trace [T]] <up, down, down></code>	False
<code>assert P :[has trace [T]] <up, down, up></code>	True

37

Leitura e exercícios

- Livro: Theory and Practice of Concurrency
 - Leitura:
 - 1.3, 1.3.1, 1.3.2 (até pag 54)
 - Exercícios:
 - 1.3.6 (página 62)

43



44