


UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

DEPARTAMENTO DE COMPUTAÇÃO

Desenvolvimento Formal

Disciplina: Métodos Formais
Bacharelado em Ciência da Computação

Prof. Lucas Albertins
lucas.albertins@ufrpe.br



```

graph TD
    A[Validação] --> B[Verificação de propriedades]
    B --> C[Animação da especificação]
    C --> D[Refinamento da especificação]
    D --> A
  
```

1

Desenvolvimento formal

- Principais atividades:
 - **Escrever** especificação
 - **Verificar** propriedades
 - **Construir** um programa (ou outra especificação mais concreta)


2

Especificação formal

- Consiste na descrição (geralmente abstrata) do comportamento esperado para o sistema
- Ponto de partida para uso de qualquer método formal
- Descreve o **quê** será implementado e não o **como** será implementado

3

Desenvolvimento formal



```

graph TD
    A[Descrição informal] <--> B[Especificação abstrata]
    B --> C[Especificação refinada]
    C --> B
    B --> B
  
```

Validação

Verificação de propriedades

Animação da especificação

Refinamento da especificação

4

Especificação abstrata

- O simples fato de escrever uma especificação formal possui como vantagem
 - Identificar possíveis ambiguidades e falta de clareza dos requisitos
 - Antecipar problemas e acelera a tomada de decisões sobre o projeto do sistema

```
channel umReal, doisReais -- moedas
channel cafe, leite -- bebidas

Maquina = umReal ->
{
  leite -> Maquina
  []
  umReal -> cafe -> Maquina
}
doisReais -> cafe -> Maquina
```

5

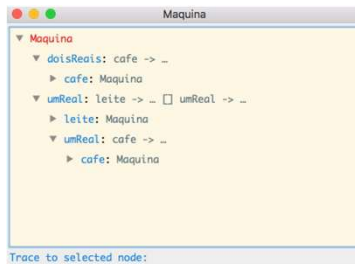
Validação do modelo

- Validação do modelo
 - Como garantir que a especificação captura o comportamento esperado?
- Além das técnicas tradicionais da engenharia de software para validação, existe a possibilidade de **animar** a especificação

6

Animando a especificação

- Animação de uma máquina de café especificada em CSP. Tela capturada da ferramenta FDR.



7

Animando a especificação

- Animação da especificação
 - Permite que o usuário forneça valores/estímulos de entrada para a especificação e observe o comportamento de um "passo" da especificação
 - Suportada por ferramentas
- Ajuda a entender o sistema (não verifica propriedades)

8

Desenvolvimento formal



9

Verificação

- Calcula se propriedades são válidas nas possíveis entradas do programa
- Pode ser manual ou com suporte de ferramentas

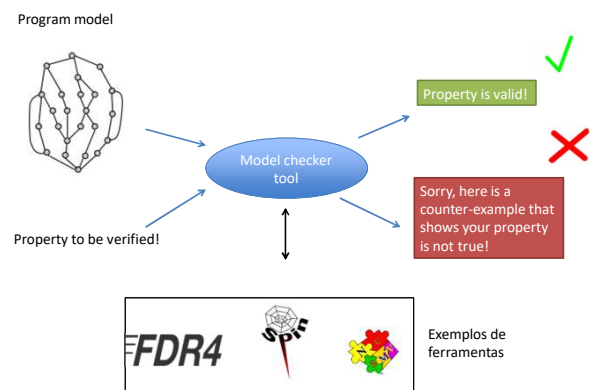
10

Ferramentas para verificação

- Principais ferramentas de apoio para verificação formal
 - Verificador de modelos (model checkers)
 - Assistente de prova
 - Provador de teoremas automático
 - Analisadores estáticos
 - SAT/SMT Solvers

11

Verificador de modelos



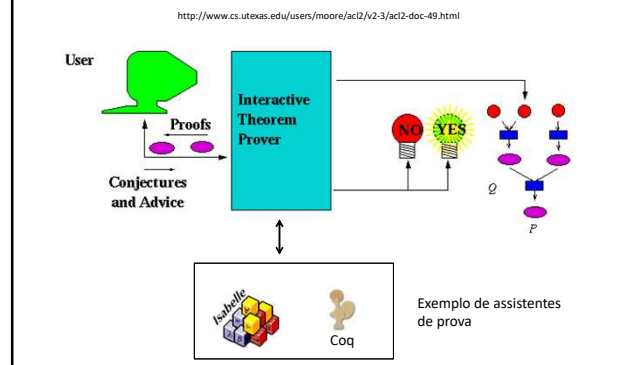
12

Verificador de modelos

- **Model checker em inglês**
- Geralmente usados para verificar sistemas concorrentes
- Exemplos de model checkers: Spin, UPPAL, JavaPathFinder
- Propriedade a ser verificada geralmente é uma expressão em lógica temporal
- Se a quantidade de estados for muito grande (explosão de estados) a ferramenta pode não ser capaz de analisar o modelo

13

Assistente de prova



14

Assistente de prova

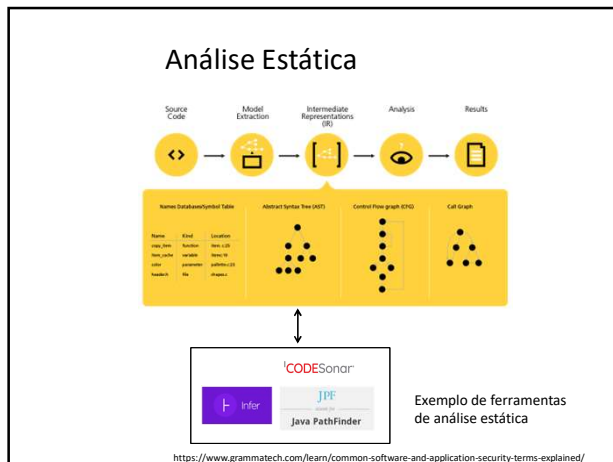
- **Assistente de prova** ajuda no processo (iterativo) de prova indicando o ponto atual da prova e quais os próximos passos
- Proof assistant é o nome para ferramentas deste tipo
 - Isabelle é um exemplo de assistente de prova
- Necessita de muito esforço do usuário porém lida com problemas complexos/indecidíveis

15

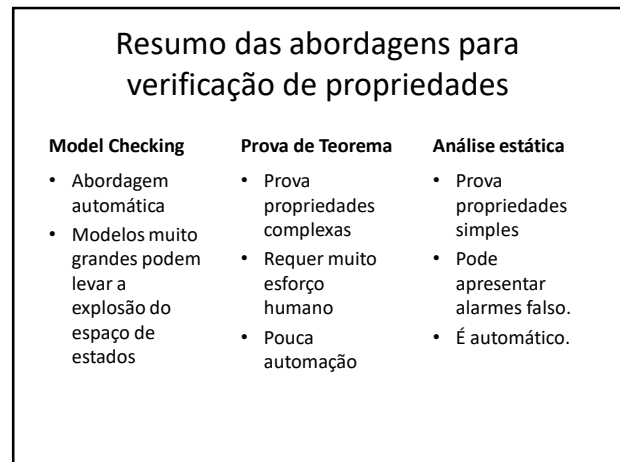
Provedor automático de teoremas

- **Provedor de teorema automático** é um tipo de ferramenta que busca soluções para uma dada expressão em lógica
 - Se existe uma solução retorna um exemplo, se não existe solução diz que a expressão não pode ser satisfeita
- SMT Solver é o tipo de ferramenta que busca soluções de forma automática
 - Exemplo: Z3 que encontra soluções para problemas de aritmética (não) linear, arrays, etc
- É totalmente automático, porém lida com um subconjunto de problemas decidíveis

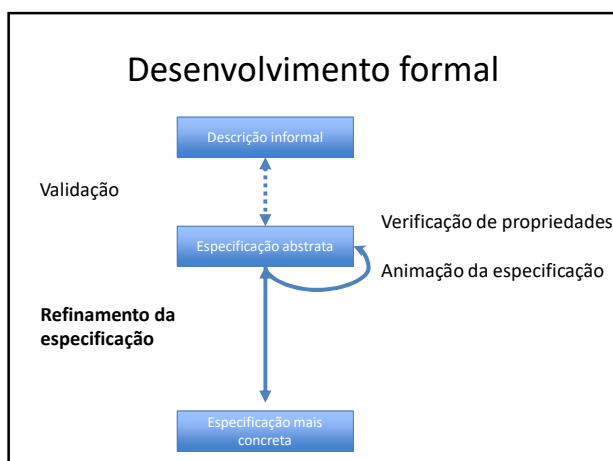
16



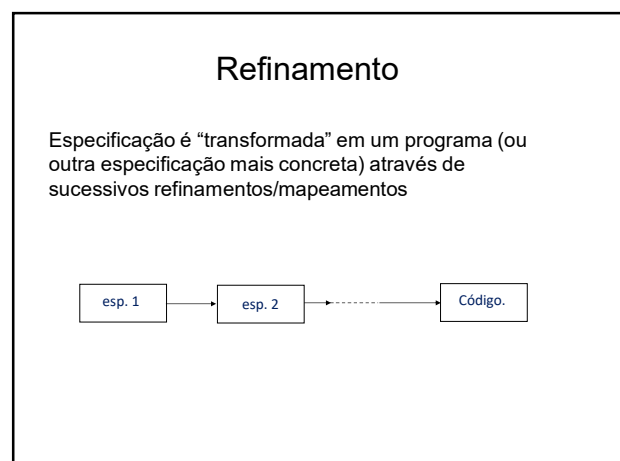
17



18



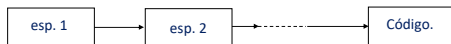
19



20

Refinamento deve ser verificado

- A cada etapa da transformação
- Para garantir que transformações preservam propriedades
 - Se versão posterior é equivalente (ou um melhoramento) da versão anterior
- FDR é um verificador de refinamentos entre processos CSP



21

Correto por construção

- Uma implementação é **correta por construção** se é um refinamento da especificação
 - Neste caso, precisa de menos verificações adicionais

22

Desenvolvimento formal

- ▣ Possibilidade de gerar programas que são corretos por construção
 - O próprio processo de desenvolvimento deve garantir que o programa faz exatamente o que foi especificado
- ▣ Este modelo, geralmente, é aplicado ao desenvolvimento de sistemas críticos, especialmente naqueles onde a segurança é um fator crítico (ex: sistema de controle de ferrovias)

23

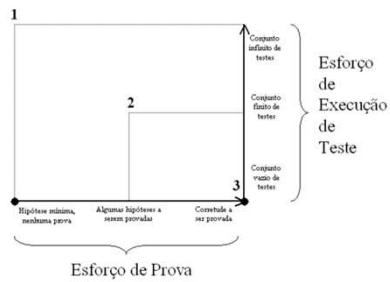
Desenvolvimento formal

- Qual a relação entre teste e prova?
- Paper de Marie Claude Gaudel (1995) mostra um framework matemático que estabelece esta relação.
Em resumo:
 1. Tudo foi provado, nenhum teste é necessário
 2. Alguma coisa é provada/assumida, conjunto finito de testes a executar
 3. Nada foi provado e nenhuma hipótese é assumida, infinitos testes devem ser executados

24

Desenvolvimento formal

- Relação entre teste e prova: Marie Claude Gaudel (1995)



25

Estilos de Especificação

- Alguns exemplos de estilo são:
 - Orientados a Propriedades (Algébricas)
 - Baseados em Modelos
 - Concorrente
- Vamos ilustrar dois deles a seguir

26

26

Especificação Algébrica

Uma Especificação consiste de

- Um conjunto de nomes de tipos (sorts)
- Um conjunto de funções
- Um conjunto de axiomas (semântica)

27

27

Um Exemplo Clássico: pilhas

- Tipo: pilha-int
- Funções


```
vazia: -> pilha-int
push: int pilha-int -> pilha-int
pop: pilha-int -> pilha-int
top: pilha-int -> int
e_vazia: pilha-int -> bool
```
- Axiomas


```
pop (push (i, p)) = p
top (push (i, p)) = i
e_vazia (p) = (p = vazia)
```

28

28

Métodos Baseados em Modelos

- Componentes de uma Especificação

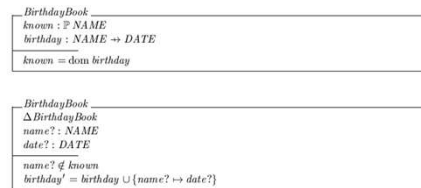


- A semântica é baseada em conjuntos, máquina de estado que descrevem o estado do sistema
- Exemplos de métodos: B, Z e VDM

29

29

Métodos Baseados em Modelos



Modelo para uma agenda de contatos na notação Z
com a operação de adicionar contato

30

30

Um Modelo Matemático para Pilhas

```

var
  pilha, pilha' : seq[Int]
operações (procedimentos, métodos)
  vazia      = (pilha' = [])
  push(i? : Int) = pilha' = [i?] ^ pilha
  pop()      = (pilha ≠ []) => pilha' = tail pilha
  top(i! : Int) = (pilha ≠ []) => i! = head pilha
  e_vazia (b! : Bool) = b! <=> (pilha = [])

```

31

31

Especificação de sistemas concorrentes

- Modelos - Exemplos
 - CSP (Communicating Sequential Processes)
 - CCS (Calculus of Communicating Systems)
- Um sistema é uma rede de processos independentes e comunicantes
- Existem vários

32

32

Um Exemplo Simples

- Relógio de parede com cuco
 - Modelado por dois componentes:
 - Contador de 1 a 60
 - Cuco: aparece a cada 60 minutos
 - A interação entre esses componentes se dá a cada 60 minutos



Cuco

33

33

Um Exemplo Simples

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

Passaro = cuco -> Passaro

Relogio = Contador(0) [|{cuco}|] Passaro

34

34

Modelos dos métodos

- Alguns exemplos de métodos e os respectivos modelos matemáticos:
 - B : baseado em ASM (Abstract State Machines)
 - Z, VDM : baseados em teoria dos conjuntos
 - CSP, CCS : baseados em automatos

35

35

Dicas para o uso de métodos formais

- Escolha a notação apropriada
- Estime os custos/benefícios
- Não abandone métodos tradicionais (use métodos semi-formais em conjunto com métodos formais).
Ex. UML + OCL
- Documente
- Teste

36

36

Desenvolvimento semiformal

- Usa ferramentas ou linguagens que tem algum nível de formalismo matemático
- Exemplos:
 - Programação declarativa
 - Uso de anotações

37

37

Programação declarativa

- Programas escritos em linguagens de programação (puramente) declarativas funcionam como especificações que podem ser executadas
- **Programação lógica** : semântica é baseada em lógica
 - Ex: Prolog
- **Programação funcional**: semântica é lambda calculus
 - Ex: Haskell

38

Uso de anotações

- **Anotações** é um método semi formal de desenvolvimento que consiste em especificar o comportamento esperado do programa
- As anotações que especificam invariantes, pré e pós condições são chamadas de **contrato**
- Durante a execução do programa (ou de forma estática) os contratos são verificados
 - Violações aos contratos são informados
- JML é um exemplo de sistema para anotação em Java
- Linguagens de programação como Eiffel possuem contrato como parte da sintaxe

39

Considerações

- Um aspecto unificador de estilos: evitar tradução entre linguagens sempre que possível.
 - Cálculo de Refinamentos [Morgan]
- Na prática, nem sempre é possível usar uma única linguagem. Mas deve-se buscar uma unificação semântica.
 - Exemplo: De CSP para Java (JCSP) ou GO ou Occam

40

40

Referências

- [1] Overview of Formal Methods
- [2] Wikipedia – Formal Methods
- [3] An Overview of Formal Methods Tools and Techniques

41



42