

Propriedades

Aula para disciplina de Métodos Formais

Gabriela Moreira

Departamento de Ciência da Computação - DCC
Universidade do Estado de Santa Catarina - UDESC

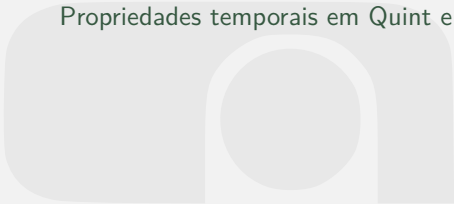
13 de maio de 2024



Conteúdo



Tipos de propriedades



Propriedades temporais em Quint e TLA+



Outline



Tipos de propriedades



Propriedades temporais em Quint e TLA+



Propriedades

Propriedades podem ser propriedades de **segurança** (*safety properties*), **vivacidade** (*liveness properties*) ou uma combinação das duas.



Segurança

*“Algo ruim **não** acontece”*

Descreve algo específico. Basta esse algo acontecer uma única vez para que a propriedade seja violada.

Exemplos:

- “O saque não deve ser autorizado, a menos que uma senha correta tenha sido digitada”
- “Dois processos não devem estar na seção crítica ao mesmo tempo”
- “Ao receber um saque, eu fico com mais dinheiro do que eu tinha antes”

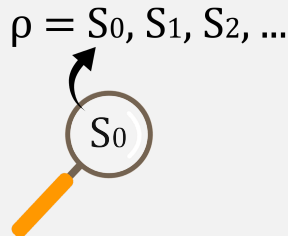
Segurança - Invariantes

Invariantes são um tipo de propriedade de segurança.

Uma invariante é uma propriedade sobre um **estado**, *não sobre uma execução*. Uma invariante não consegue “ver além” de um único estado.

Uma **execução** satisfaz uma invariante sse cada estado da execução satisfaz a invariante.

Uma **estrutura de Kripke** satisfaz uma invariante sse cada estado alcançável satisfaz a invariante.





Vivacidade

“Algo bom eventualmente acontece”

$\Diamond F$



Vivacidade

“Algo bom eventualmente acontece”

$\diamond F$

Exemplos:

- “Se um processo pediu pra entrar na seção crítica, ele eventualmente deve conseguir”
- “Cada sinaleiro deve sempre eventualmente ficar verde”



Vivacidade

“Algo bom eventualmente acontece”

$\diamond F$

Exemplos:

- “Se um processo pediu pra entrar na seção crítica, ele eventualmente deve conseguir”
- “Cada sinaleiro deve sempre eventualmente ficar verde”

Lembrando que em Quint e TLA+ usamos LTL, não CTL. Portanto, as fórmulas devem ser verdadeiras em todas as execuções.



Vivacidade - propriedade de persistência

“Eventualmente, algo é satisfeito pra sempre”

$\diamond \square F$



Vivacidade - propriedade de persistência

“Eventualmente, algo é satisfeito pra sempre”

$\diamond \square F$

Exemplos:

- Ao entrar na faculdade, eventualmente vou ter um diploma
- Eventualmente estaremos mortos
- Eventualmente teremos cabelos brancos ou calvice
- Eventualmente as partes chegam em consenso



Demonstrando vivacidade com runs em Quint

Uma forma alternativa de mostrar que “coisas boas acontecem” em Quint é através de runs.

- Uma **run** pode definir uma ou mais execuções onde algo acontece.
 - Não serve para mostrar que algo acontece em todas as execuções, como a propriedade em si
- Útil quando queremos demonstrar que algo acontece em algumas execuções, mas não necessariamente todas.



Demonstrando vivacidade com runs em Quint

Uma forma alternativa de mostrar que “coisas boas acontecem” em Quint é através de runs.

- Uma **run** pode definir uma ou mais execuções onde algo acontece.
 - Não serve para mostrar que algo acontece em todas as execuções, como a propriedade em si
- Útil quando queremos demonstrar que algo acontece em algumas execuções, mas não necessariamente todas.

Por exemplo, se queremos saber se é possível, em algum cenário, chegar em consenso. Podemos definir uma run semelhante a:

- 1 Estado inicial
- 2 Processo 1 propõe “A”
- 3 Processo 2 propõe “A”
- 4 Trocas de mensagens seguindo protocolo de consenso
- 5 Processos 1 e 2 decidem “A”



Fairness (razoabilidade)

Propriedades de razoabilidade (*Fairness properties*)

- Razoabilidade incondicional (*Unconditional fairness*): “Algo acontece com frequência infinita”



Fairness (razoabilidade)

Propriedades de razoabilidade (*Fairness properties*)

- Razoabilidade incondicional (*Unconditional fairness*): “Algo acontece com frequência infinita”
 - Razoabilidade **forte** (*Strong fairness*): “Algo acontece com frequência infinita **se é habilitado com frequência infinita**”



Fairness (razoabilidade)

Propriedades de razoabilidade (*Fairness properties*)

- Razoabilidade incondicional (*Unconditional fairness*): “Algo acontece com frequência infinita”
 - Razoabilidade **forte** (*Strong fairness*): “Algo acontece com frequência infinita **se é habilitado com frequência infinita**”
 - Razoabilidade **fraca** (*Weak fairness*): “Algo acontece com frequência infinita **se é continuamente habilitado a partir de um certo momento**”



Fairness (razoabilidade)

Propriedades de razoabilidade (*Fairness properties*)

- Razoabilidade incondicional (*Unconditional fairness*): “Algo acontece com frequência infinita”
 - Razoabilidade **forte** (*Strong fairness*): “Algo acontece com frequência infinita **se é habilitado com frequência infinita**”
 - Razoabilidade **fraca** (*Weak fairness*): “Algo acontece com frequência infinita **se é continuamente habilitado a partir de um certo momento**”

Usamos essas propriedades como **pré-condições** para descartar execuções não realistas.

- $WF(...) \rightarrow F$
- $SF(...) \rightarrow F$



Fairness - definições precisas

Primeiramente, precisamos definir **passos balbuciantes** (*stuttering steps*): são aqueles em que o valor de uma variável ou de um conjunto de variáveis não se altera.

- Por exemplo, $x' = x$ é um passo balbuciante para a variável x .



Fairness - definições precisas

Primeiramente, precisamos definir **passos balbuciantes** (*stuttering steps*): são aqueles em que o valor de uma variável ou de um conjunto de variáveis não se altera.

- Por exemplo, $x' = x$ é um passo balbuciante para a variável x .

Operador **enabled** (ativado):

- $\text{ENABLED } A$ (ou $\text{enabled}(A)$ em Quint) para uma ação A é verdadeiro em um estado s sse é possível fazer um passo A a partir de s .
- Ou seja, se existe um estado t tal que o passo $s \rightarrow t$ satisfaz A .



Fairness - definições precisas

Primeiramente, precisamos definir **passos balbuciantes** (*stuttering steps*): são aqueles em que o valor de uma variável ou de um conjunto de variáveis não se altera.

- Por exemplo, $x' = x$ é um passo balbuciante para a variável x .

Operador **enabled** (ativado):

- $\text{ENABLED } A$ (ou $\text{enabled}(A)$ em Quint) para uma ação A é verdadeiro em um estado s sse é possível fazer um passo A a partir de s .
- Ou seja, se existe um estado t tal que o passo $s \rightarrow t$ satisfaz A .

Seguem definições precisas copiadas do meu TCC (traduzidas do livro do Lamport (LAMPORT, 2002)).

- Infelizmente não tem como simplificar essas definições, mas tenham em mente que elas estão aqui por questões de completude.



Weak fairness - definição precisa

A razoabilidade fraca para uma fórmula de estado f e uma ação A é escrita como $WF_f(A)$.

- É satisfeita por um comportamento sse $A \wedge (f' \neq f)$ é infinitamente não ativável (ENABLED) ou infinitos passos $A \wedge (f' \neq f)$ ocorrem.
- Garante que A não possa permanecer continuamente ativável para sempre sem que um passo A ocorra. Essa condição pode ser escrita de forma equivalente como
 - $\Box(\text{ENABLED } A \implies \Diamond \langle A \rangle_f)$



Weak fairness - definição precisa

A razoabilidade fraca para uma fórmula de estado f e uma ação A é escrita como $WF_f(A)$.

- É satisfeita por um comportamento sse $A \wedge (f' \neq f)$ é infinitamente não ativável (ENABLED) ou infinitos passos $A \wedge (f' \neq f)$ ocorrem.
- Garante que A não possa permanecer continuamente ativável para sempre sem que um passo A ocorra. Essa condição pode ser escrita de forma equivalente como
 - $\Box(\text{ENABLED } A \implies \Diamond \langle A \rangle_f)$

A conjunção com $(f' \neq f)$, expressada com a notação $\langle A \rangle_f$, se deve ao fato de não ser desejável exigir que passos balbuciantes eventualmente ocorram.

- $A \wedge (f' \neq f)$ pode ser lido como “todos os passos não balbuciantes que satisfazem A ”.



Strong fairness - definição precisa

A razoabilidade fraca recebe a denominação “fraca” porque exige que uma ação permaneça continuamente ativável para garantir a ocorrência de um passo que a satisfaça.

- Se um comportamento repetidamente tornar a ação ativável e em seguida não ativável, a razoabilidade fraca não garante nada sobre a ocorrência da ação neste comportamento.
- Para tal, é necessário garantir a propriedade de razoabilidade forte (*strong fairness*).



Strong fairness - definição precisa

A razoabilidade fraca recebe a denominação “fraca” porque exige que uma ação permaneça continuamente ativável para garantir a ocorrência de um passo que a satisfaça.

- Se um comportamento repetidamente tornar a ação ativável e em seguida não ativável, a razoabilidade fraca não garante nada sobre a ocorrência da ação neste comportamento.
- Para tal, é necessário garantir a propriedade de razoabilidade forte (*strong fairness*).

A razoabilidade forte para uma fórmula de estado f e uma ação A é escrita como $SF_f(A)$.

- É satisfeita por um comportamento sse $A \wedge (f' \neq f)$ ocorre finitas vezes ou infinitos passos $A \wedge (f' \neq f)$ ocorrem.
- Garante que A não possa ser repetidamente ativável para sempre sem que um passo A ocorra.



Fairness na prática

Usamos fairness para “excluir” cenários que não são realistas mas podem causar loops no modelo.

- “loops irrealistas não ocorrem” implica em “coisa boa eventualmente acontece”



Outline



Tipos de propriedades



Propriedades temporais em Quint e TLA+



Propriedades temporais em Quint e TLA+

O Apache atualmente tem algumas limitações para fórmulas temporais, então vamos usar o TLC.



Propriedades temporais em Quint e TLA+

O Apalache atualmente tem algumas limitações para fórmulas temporais, então vamos usar o TLC.

O Quint ainda não está completamente integrado ao TLC. Para usar Quint com TLC, temos que:

- 1 Usar o subcomando `quint compile` para produzir uma especificação em TLA+
- 2 Alterar a definição de `init` na especificação em TLA+ para que seja um predicado (e não uma ação)



Propriedades temporais em Quint e TLA+

O Apalache atualmente tem algumas limitações para fórmulas temporais, então vamos usar o TLC.

O Quint ainda não está completamente integrado ao TLC. Para usar Quint com TLC, temos que:

- 1 Usar o subcomando `quint compile` para produzir uma especificação em TLA+
- 2 Alterar a definição de `init` na especificação em TLA+ para que seja um predicado (e não uma ação)

Como esse processo ainda não está legal, vamos usar somente TLA+ nos testes da aula de hoje. De qualquer forma, veremos as sintaxes nas duas linguagens.



Sintaxe

□F, Sempre, *Always*:

- $\Box F$ (TLA+)
- `always(F)` (Quint)



Sintaxe

$\square F$, Sempre, *Always*:

- $\square F$ (TLA+)
- `always(F)` (Quint)

$\diamond F$, Eventualmente, Finalmente:

- $\langle \rangle F$ (TLA+)
- `eventually(F)` (Quint)

Sintaxe

$\Box F$, Sempre, *Always*:

- $\Box F$ (TLA+)
- `always(F)` (Quint)

$\Diamond F$, Eventualmente, Finalmente:

- $\Diamond F$ (TLA+)
- `eventually(F)` (Quint)

Razoabilidade forte e fraca (*weak fairness* e *strong fairness*) de uma ação A exigindo mudanças nas variáveis `vars`

- $WF_vars(A)$ e $SF_vars(A)$ (TLA+)
- `weakFair(A, vars)` e `strongFair(A, vars)` (Quint*)



Operador *leads to* (leva a)

TLA+ também define o operador temporal $\sim>$ lido com *leads to*.

- $F \sim> G$ determina que, sempre que F é verdade, G deve ser verdade eventualmente
- Equivalente a $\Box(F \rightarrow \Diamond G)$



Operador *leads to* (leva a)

TLA+ também define o operador temporal $\sim>$ lido com *leads to*.

- $F \sim> G$ determina que, sempre que F é verdade, G deve ser verdade eventualmente
- Equivalente a $\Box(F \rightarrow \Diamond G)$

Não existe *leads to* em Quint, mas podemos definir a versão equivalente:

- `always(F implies eventually(Q))`

Operador *leads to* (leva a)

TLA+ também define o operador temporal $\sim>$ lido com *leads to*.

- $F \sim> G$ determina que, sempre que F é verdade, G deve ser verdade eventualmente
- Equivalente a $\Box(F \rightarrow \Diamond G)$

Não existe *leads to* em Quint, mas podemos definir a versão equivalente:

- `always(F implies eventually(Q))`

PS: Não confundir com until ou release da lógica temporal.



Verificando propriedades temporais

Em Quint (instável):

```
quint verify --temporal minha_propriedade arquivo.qnt
```

- PS: as formulas em Quint devem ser escritas em definições do modo `temporal`
 - i.e. `temporal minha_propriedade = eventually(true)`

Verificando propriedades temporais

Em Quint (instável):

```
quint verify --temporal minha_propriedade arquivo.qnt
```

- PS: as formulas em Quint devem ser escritas em definições do modo `temporal`
 - i.e. `temporal minha_propriedade = eventually(true)`

Em TLA+ (com TLC):

- No arquivo `.cfg`, adicionar:

```
PROPERTY  
MinhaPropriedade
```

- Depois, só rodar o model checker normalmente.



Especificação dos semáforos

Vamos verificar duas propriedades temporais para a especificação dos semáforos.

```
EventualmenteAbre == WF_<<cores>>(Next) =>  
  \A s \in SEMAFOROS : <>(cores[s] = "verde")  
  
SeAbriuVaiFechar == WF_<<cores>>(Next) =>  
  \A s \in SEMAFOROS : (cores[s] = "verde" ~> cores[s]  
    = "vermelho")
```

Especificação da chaleira

```
MODULE Chaleira
EXTENDS Integers
VARIABLE chaleira

Init  $\triangleq$  chaleira = "temperatura_ambiente"

Ligar  $\triangleq$  chaleira = "temperatura_ambiente"  $\wedge$  chaleira' = "esquentando"

Desligar  $\triangleq$  chaleira = "esquentando"  $\wedge$  chaleira' = "esfriando"

DesligarPorSensor  $\triangleq$  chaleira = "esquentando"  $\wedge$  chaleira' = "cem_graus"

Estabilizar  $\triangleq$ 
   $\vee$  chaleira = "cem_graus"  $\wedge$  chaleira' = "esfriando"
   $\vee$  chaleira = "esfriando"  $\wedge$  chaleira' = "temperatura_ambiente"

Next  $\triangleq$  Ligar  $\vee$  Desligar  $\vee$  DesligarPorSensor  $\vee$  Estabilizar

EventualmenteCemGaus  $\triangleq$  (WFchaleira(Next)  $\wedge$  SFchaleira(DesligarPorSensor))  $\Rightarrow$ 
   $\Diamond$ (chaleira = "cem_graus")
```



Referências

LAMPORT, L. **Specifying systems: The tla+ language and tools for hardware and software engineers.** Boston: Addison-Wesley, 2002.



Propriedades

Aula para disciplina de Métodos Formais

Gabriela Moreira

Departamento de Ciência da Computação - DCC
Universidade do Estado de Santa Catarina - UDESC

13 de maio de 2024