

# Checando propriedades com Lógica Temporal

Aula para disciplina de Métodos Formais

Gabriela Moreira

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

19 de março de 2025

# Conteúdo

*Model checking*

Lógica Temporal

LTL

CTL

Verificando propriedades

# Outline

*Model checking*

Lógica Temporal

LTL




CTL

Verificando propriedades

# Contexto

- **Problema:** É difícil pensar em todos os cenários, principalmente em sistemas concorrentes/distribuídos

# Contexto

- **Problema:** É difícil pensar em todos os cenários, principalmente em sistemas concorrentes/distribuídos
- **Solução:** Diferentes níveis de verificação
  - ①  *Fuzzing* e simulação
  - ②  *Model checking*
  - ③  Provas (com assistente de provas)

# Interface de um *model checker*



Como um *model checker* é usado?



# Interface de um *model checker*

Como um *model checker* é usado?



## Duas entradas:

- ①  Um modelo - uma máquina de estado finita\*
- ②  Uma propriedade - uma fórmula de alguma lógica temporal




# Interface de um *model checker*

Como um *model checker* é usado?

## Duas entradas:

- 1  Um modelo - uma máquina de estado finita\*
- 2  Uma propriedade - uma fórmula de alguma lógica temporal

## Três possíveis saídas

- 1  Sucesso
- 2  Contra-exemplo: Uma sequência de estados que viola a propriedade
- 3  Não há memória suficiente





Opcionalmente, também pode detectar *deadlocks*.

- *Deadlock*: o modelo não obedece a restrição de uma estrutura de Kripke, ou seja, existe um estado sem transição alguma saindo dele.







# Vantagens e Desvantagens

## Vantagens




- 1  Contra-exemplos
- 2  Verificação parcial
- 3  Processo automatizado
- 4  Sequências infinitas de estados

# Vantagens e Desvantagens

## Vantagens





- 1  Contra-exemplos
- 2  Verificação parcial
- 3  Processo automatizado
- 4  Sequências infinitas de estados

## Desvantagens




- 1  Explosão de estados
- 2  Não permite generalização (i.e.  $N$  processos)
- 3  *Model checkers* em si não são verificados

# Vantagens e Desvantagens

## Vantagens

- 1  Contra-exemplos
- 2  Verificação parcial
- 3  Processo automatizado
- 4  Sequências infinitas de estados

## Desvantagens

- 1  Explosão de estados
- 2  Não permite generalização (i.e.  $N$  processos)
- 3  *Model checkers* em si não são verificados

No geral, é um método bom para encontrar falhas em software, que permite a verificação com certas restrições.

Comparado a testes e simulações aleatórias, pode ser mais eficiente em encontrar casos de borda onde falhas ocorrem.



# Modelos

Linguagens de especificação fornecem diferentes abstrações para como definir uma máquina de estados. Exemplos: Redes de Petri, TLA+ (*Temporal Logic of Actions+*), CSP (*Communicating Sequential Processes*), Alloy, entre outras.

# ≠ Propriedades - ⌚ Lógica temporal

Dois operadores temporais principais:

- Eventualmente ( $\Diamond$ ) ou Finalmente ( $F$ ): Cada semáforo deve eventualmente ficar verde
- Sempre ( $\Box$ ) ou ( $A$ ): O número do próximo estado é sempre maior que o número no estado anterior

Fórmulas de lógica temporal são sobre um comportamento (execução) do sistema modelado.

# ≠ Propriedades - ○ Invariantes

Invariantes são predicados sobre estados individuais do sistema. Um invariante é satisfeito se e somente se ele é verdadeiro para todos os estados do sistema.

# ≠ Propriedades - ○ Invariantes



Invariantes são predicados sobre estados individuais do sistema. Um invariante é satisfeito se e somente se ele é verdadeiro para todos os estados do sistema.

**Invariantes Indutivos** são tipos especiais de invariantes que podem ser provados com indução matemática, isso é, sem necessidade de explorar todos os estados.

- Muito poderosos mas também difíceis de se definir

## Contra-exemplo



### Entradas:

-  Modelo: 2 semáforos sem controle de revezamento
-  Propriedade: para cada semáforo, ele deve eventualmente ficar verde



# Contra-exemplo



## Entradas:

-  Modelo: 2 semáforos sem controle de revezamento
-  Propriedade: para cada semáforo, ele deve eventualmente ficar verde

Execução do *model checker*

## Contra-exemplo

### Entradas:

-  Modelo: 2 semáforos sem controle de revezamento
-  Propriedade: para cada semáforo, ele deve eventualmente ficar verde



Execução do *model checker*

### Saída - Contra-exemplo:

- 1 Semáforo 1 inicia vermelho, semáforo 2 inicia vermelho
- 2 Semáforo 1 fica verde (e semáforo 2 permanece vermelho)
- 3 Semáforo 1 fica amarelo (e semáforo 2 permanece vermelho)
- 4 Retorna ao estado (1)

## Contra-exemplo

### Entradas:

-  Modelo: 2 semáforos sem controle de revezamento
-  Propriedade: para cada semáforo, ele deve eventualmente ficar verde

Execução do *model checker*

### Saída - Contra-exemplo:

- 1 Semáforo 1 inicia vermelho, semáforo 2 inicia vermelho
- 2 Semáforo 1 fica verde (e semáforo 2 permanece vermelho)
- 3 Semáforo 1 fica amarelo (e semáforo 2 permanece vermelho)
- 4 Retorna ao estado (1)

Ótimo artefato para reprodução de bugs e geração de testes automatizados.



## Exercício: Qual dessas fórmulas pode ser um invariante?

- 1 Ao fazer uma transferência bancária, eu acabo com menos dinheiro do que tinha antes
- 2 Ao ligar uma chaleira elétrica, ela eventualmente chegará a 100 graus
- 3 Em um jogo da velha, a diferença entre o número de X e O não é maior do que 1



## Exercício: Qual dessas fórmulas pode ser um invariante?

- 1 Ao fazer uma transferência bancária, eu acabo com menos dinheiro do que tinha antes
- 2 Ao ligar uma chaleira elétrica, ela eventualmente chegará a 100 graus
- 3 Em um jogo da velha, a diferença entre o número de X e O não é maior do que 1

💡 Dica: Um invariante é uma fórmula a ser avaliada em cada estado do sistema.



## Exercício: Qual dessas fórmulas pode ser um invariante?

- 1 Ao fazer uma transferência bancária, eu acabo com menos dinheiro do que tinha antes
- 2 Ao ligar uma chaleira elétrica, ela eventualmente chegará a 100 graus
- 3 Em um jogo da velha, a diferença entre o número de X e O não é maior do que 1

💡Dica: Um invariante é uma fórmula a ser avaliada em cada estado do sistema.

Resposta: 3

# Outline

*Model checking*

Lógica Temporal

LTL

CTL

Verificando propriedades

# Operadores Temporais - Unários

- $\Box$  ou **G**: *Globally*, sempre.
  - **G**  $\phi$ :  $\phi$  deve ser verdadeiro por toda a execução a partir de agora.
  - Exemplo: Comida sacia a fome



# Operadores Temporais - Unários

- $\Box$  ou **G**: *Globally*, sempre.
  - **G**  $\phi$ :  $\phi$  deve ser verdadeiro por toda a execução a partir de agora.
  - Exemplo: Comida sacia a fome
- $\Diamond$  ou **F**: *Finally*, eventualmente, no Futuro.
  - **F**  $\phi$ : eventualmente (na execução a partir de agora),  $\phi$  deve ser verdadeiro.
  - Exemplo: Eventualmente, terei fome

# Operadores Temporais - Unários

- $\square$  ou **G**: *Globally*, sempre.
  - **G**  $\phi$ :  $\phi$  deve ser verdadeiro por toda a execução a partir de agora.
  - Exemplo: Comida sacia a fome
- $\diamond$  ou **F**: *Finally*, eventualmente, no Futuro.
  - **F**  $\phi$ : eventualmente (na execução a partir de agora),  $\phi$  deve ser verdadeiro.
  - Exemplo: Eventualmente, terei fome
- $\bigcirc$  ou **X**: *Ne(x)t*, próximo.
  - **X**  $\phi$ :  $\phi$  deve ser verdadeiro no próximo estado.
  - Exemplo: Logo após comer, tenho sede

# Operadores Temporais - Binários

- **U**: *Until*, até.
  - $\psi \text{ U } \phi$ :  $\psi$  deve ser verdade até que  $\phi$  seja verdade, sendo que  $\phi$  deve ser verdade no presente ou no futuro.
  - Exemplo: Eu tenho fome até eu comer alguma coisa

# Operadores Temporais - Binários

- **U**: *Until*, até.
  - $\psi \text{ U } \phi$ :  $\psi$  deve ser verdade até que  $\phi$  seja verdade, sendo que  $\phi$  deve ser verdade no presente ou no futuro.
  - Exemplo: Eu tenho fome até eu comer alguma coisa
- **R**: *Release*, libera.
  - $\psi \text{ R } \phi$ :  $\phi$  deve ser verdade até e incluindo o momento que  $\psi$  se torna verdadeiro. Se  $\psi$  nunca ficar verdadeiro,  $\phi$  deve permanecer verdadeiro para sempre.
  - Exemplo: Ao comer chocolate, deixo de ter vontade de comer doce. Detalhe: Eu posso continuar com vontade de comer e acabar nunca comendo chocolate.

# LTL e CTL

- **LTL** - *Linear Temporal Logic* (Lógica Temporal Linear). Em LTL, as fórmulas são implicitamente universalmente quantificadas. Propriedades que falam sobre a existência de uma execução não podem ser expressadas.
- **CTL** - *Computational Tree Logic* (Lógica de Árvore Computacional). CTL é uma lógica sobre a ramificação do tempo.

# LTL e CTL - Visualização

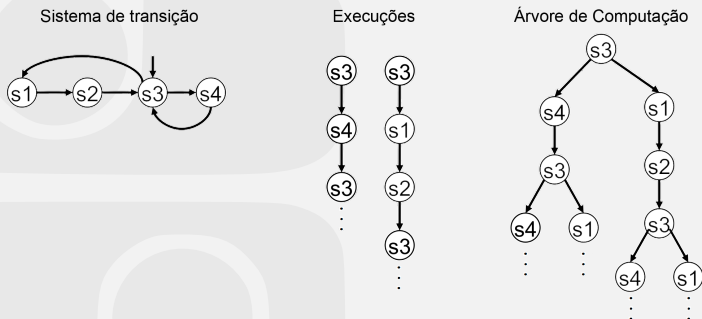


Figure 1: Fonte (BULTAN, 2023)



# Outline

*Model checking*

Lógica Temporal

**LTL**

CTL

Verificando propriedades

# LTL - *Linear Temporal Logic*

Na lógica temporal linear (LTL), temos operadores para descrever eventos ao longo de uma única execução.

Seja  $AP$  um conjunto finito de proposições atômicas (i.e.  $\{p_0, p_1, \dots, p_n\}$ ), o conjunto de fórmulas LTL sobre  $AP$  é definido indutivamente por:

- se  $p \in AP$  então  $p$  é uma fórmula LTL;
- se  $\psi$  e  $\phi$  são fórmulas LTL, então  $\neg\psi$ ,  $\phi \vee \psi$ ,  $\mathbf{X}\psi$ , e  $\phi \mathbf{U}\psi$  são fórmulas LTL.

$$\phi ::= \perp \mid \top \mid p \mid \neg\phi \mid \phi \vee \psi \mid \mathbf{X}\psi \mid \phi \mathbf{U}\psi$$



# Equivalências

Os operadores **G**, **F** e **R** podem ser definidos usando somente **X** e **U**.

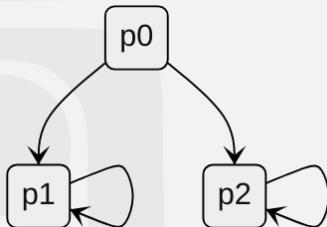
- $\mathbf{G}\psi \equiv \perp \mathbf{R}\psi \equiv \neg \mathbf{F}\neg\psi$
- $\mathbf{F}\psi \equiv \top \mathbf{U}\psi$
- $\phi \mathbf{R}\psi \equiv \neg(\neg\phi \mathbf{U}\neg\psi)$ 
  - Até o momento que  $\psi$  fica falso,  $\phi$  não pode ser falso
- $\neg \mathbf{G}\psi \equiv \mathbf{F}\neg\psi$

# Negações de fórmulas

Uma formula ser falsa não significa que sua negação é verdadeira. Por exemplo, a fórmula a seguir não é necessariamente verdadeira:

$$\mathbf{F}p_1 \vee \neg \mathbf{F}p_1$$

Exemplo:



# Exercícios

Qual dos operadores temporais G (*Globally*), F (*Finally*), X (*Next*), U (*Until*) e R (*Release*) pode ser representado pelo diagrama a seguir?



# Exercícios

Qual dos operadores temporais  $G$  (*Globally*),  $F$  (*Finally*),  $X$  (*Next*),  $U$  (*Until*) e  $R$  (*Release*) pode ser representado pelo diagrama a seguir?



**Resposta: X, Next**

## Exercícios II

Qual dos operadores temporais  $G$  (*Globally*),  $F$  (*Finally*),  $X$  (*Next*),  $U$  (*Until*) e  $R$  (*Release*) pode ser representado pelo diagrama a seguir?



## Exercícios II

Qual dos operadores temporais  $G$  (*Globally*),  $F$  (*Finally*),  $X$  (*Next*),  $U$  (*Until*) e  $R$  (*Release*) pode ser representado pelo diagrama a seguir?



**Resposta:**  $G$ , Globally

## Exercícios III

Qual dos operadores temporais  $G$  (*Globally*),  $F$  (*Finally*),  $X$  (*Next*),  $U$  (*Until*) e  $R$  (*Release*) pode ser representado pelo diagrama a seguir?



## Exercícios III

Qual dos operadores temporais  $G$  (*Globally*),  $F$  (*Finally*),  $X$  (*Next*),  $U$  (*Until*) e  $R$  (*Release*) pode ser representado pelo diagrama a seguir?



**Resposta: U, Until**



## Exercícios IV

Qual dos operadores temporais  $G$  (*Globally*),  $F$  (*Finally*),  $X$  (*Next*),  $U$  (*Until*) e  $R$  (*Release*) pode ser representado pelo diagrama a seguir?



## Exercícios IV

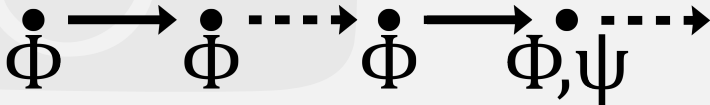
Qual dos operadores temporais  $G$  (*Globally*),  $F$  (*Finally*),  $X$  (*Next*),  $U$  (*Until*) e  $R$  (*Release*) pode ser representado pelo diagrama a seguir?



**Resposta: F, Finally**

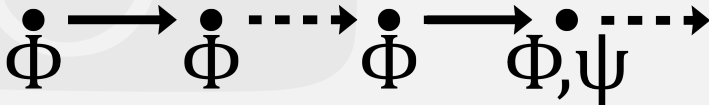
## Exercícios IV

Qual dos operadores temporais  $G$  (*Globally*),  $F$  (*Finally*),  $X$  (*Next*),  $U$  (*Until*) e  $R$  (*Release*) pode ser representado pelo diagrama a seguir?



## Exercícios IV

Qual dos operadores temporais G (*Globally*), F (*Finally*), X (*Next*), U (*Until*) e R (*Release*) pode ser representado pelo diagrama a seguir?



**Resposta:** R, Release

# LTL em Estruturas de Kripke

Uma fórmula LTL é verdadeira para uma **estrutura de Kripke** se ela é verdadeira no(s) **estado(s) inicial(is)**.

Uma fórmula LTL é verdadeira em um **estado** se ela é verdadeira para **todas as execuções** iniciando naquele estado.

Ou seja, a fórmula deve ser verdadeira para todos as execuções (comportamentos) da estrutura.

# Outline

*Model checking*

Lógica Temporal

LTL

CTL

Verificando propriedades

# CTL - *Computational Tree Logic*

A gramática a seguir define fórmulas em CTL (sendo  $p \in AP$ ):

$$\phi ::= \perp \mid \top \mid p \mid \neg\phi \mid \phi \vee \psi \mid A[\phi \mathbf{U} \psi] \mid E[\phi \mathbf{U} \psi] \mid A \mid E$$

Todos os operadores temporais devem ser precedidos de **A** (*All*, *Todo*) ou **E** (*Exists*, *Existe*).

- **A**  $\phi$ :  $\phi$  deve ser verdadeiro em todas as execuções a partir do estado atual;
- **E**  $\phi$ : Existe ao menos um caminho a partir do estado atual onde  $\phi$  é verdade.

# Visualização

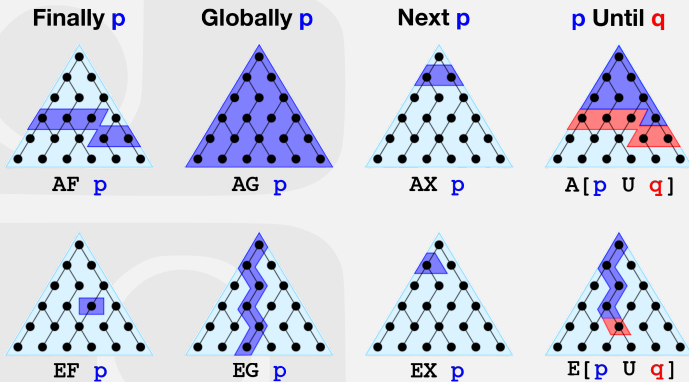


Figure 2: Fonte (RAJU, 2014)



# Outline

*Model checking*

Lógica Temporal

LTL

CTL

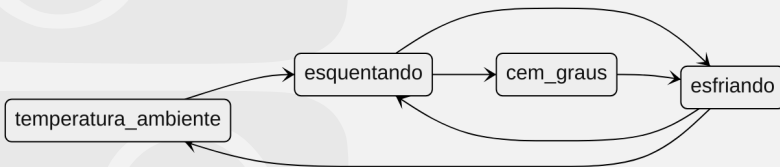
Verificando propriedades

# LTL vs CTL

**Atenção:**  $LTL \not\subseteq CTL$  and  $CTL \not\subseteq LTL$

- $F(Gp)$  é uma fórmula LTL que não pode ser expressa em CTL.
- $EXp$  é uma fórmula CTL que não pode ser expressa em LTL.

## Exemplo chaleiras - modelo



# Exemplo chaleiras

PS: Release é V nesse sistema

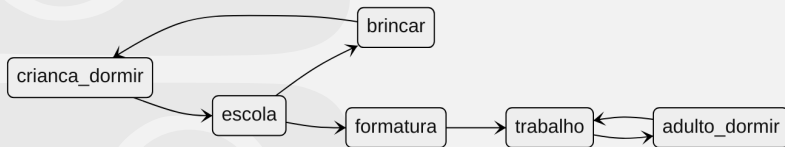
## CTL:

- $EF(\text{cem\_graus})$ : true
- $\text{esquentando} \rightarrow AF \text{ cem\_graus}$ : false
- $\text{esquentando} \rightarrow EF \text{ temperatura\_ambiente}$ : true
- $EF(EG(\neg \text{cem\_graus}))$ : true

## LTL:

- $F(\text{temperatura\_ambiente} \mid \text{cem\_graus})$ : false
- $F(\text{esquentando} \mid \text{esfriando})$ : true
  - A não ser que haja uma auto-transição em  $\text{temperatura\_ambiente}$

## Exemplo Adultos e Crianças - modelo



# Exemplo Adultos e Crianças

## CTL:

- $EF \text{ trabalho}$ : true
- $AF \text{ trabalho}$ : false

## LTL:

- $F \text{ trabalho}$ : false
  - Não é possível expressar nada como  $EF \text{ trabalho}$
- $\text{formatura} \rightarrow X(G(\text{trabalho} \mid \text{adulto\_dormir}))$ : true
- $\text{formatura} \rightarrow X(\text{trabalho} \text{ U } \text{adulto\_dormir})$ : true
- $(F \text{ brincar}) \text{ U } \text{formatura}$ : false

## *Workaround* para falta do existencial com invariantes

Na prática, quando queremos verificar o equivalente a  $EF\ p$  onde  $p$  é uma proposição (não uma fórmula temporal), fazemos o seguinte:

## Workaround para falta do existencial com invariantes

Na prática, quando queremos verificar o equivalente a  $EF\ p$  onde  $p$  é uma proposição (não uma fórmula temporal), fazemos o seguinte:

- 1 Definimos  $p$  como uma invariante (isso é,  $p$  deve ser verdade em todos os estados)



## Workaround para falta do existencial com invariantes

Na prática, quando queremos verificar o equivalente a  $EF\ p$  onde  $p$  é uma proposição (não uma fórmula temporal), fazemos o seguinte:

- 1 Definimos  $p$  como uma invariante (isso é,  $p$  deve ser verdade em todos os estados)
- 2 Rodamos o *model checker*

## Workaround para falta do existencial com invariantes

Na prática, quando queremos verificar o equivalente a  $EF\ p$  onde  $p$  é uma proposição (não uma fórmula temporal), fazemos o seguinte:

- 1 Definimos  $p$  como uma invariante (isso é,  $p$  deve ser verdade em todos os estados)
- 2 Rodamos o *model checker*
- 3 Invertemos o resultado:

## Workaround para falta do existencial com invariantes

Na prática, quando queremos verificar o equivalente a  $EF\ p$  onde  $p$  é uma proposição (não uma fórmula temporal), fazemos o seguinte:

- 1 Definimos  $p$  como uma invariante (isso é,  $p$  deve ser verdade em todos os estados)
- 2 Rodamos o *model checker*
- 3 Invertemos o resultado:
  - Se for “ok”, é porque a propriedade  $EF\ p$  é falsa

## Workaround para falta do existencial com invariantes

Na prática, quando queremos verificar o equivalente a  $EF\ p$  onde  $p$  é uma proposição (não uma fórmula temporal), fazemos o seguinte:

- 1 Definimos  $p$  como uma invariante (isto é,  $p$  deve ser verdade em todos os estados)
- 2 Rodamos o *model checker*
- 3 Invertemos o resultado:
  - Se for “ok”, é porque a propriedade  $EF\ p$  é falsa
  - Se for uma violação, é porque a propriedade  $EF\ p$  é verdadeira (e o contra-exemplo é um exemplo de execução onde  $F\ p$  é verdade).

# Referências

BULTAN, T. **Cs 267: Automated verification - lecture 2**. Disponível em: <<https://sites.cs.ucsb.edu/~bultan/courses/267/lectures/12.pdf>>.

RAJU, D. **Ltl and ctl - lecture notes by dhananjay raju**. Disponível em: <<https://www.cs.utexas.edu/~draju/Verification/class2.pdf>>.

# Checando propriedades com Lógica Temporal

Aula para disciplina de Métodos Formais

Gabriela Moreira

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

19 de março de 2025