Verificação de Modelos (Model Checking)

Estes slides são baseados nas notas de aula da Profa. Corina Cîrstea

Agenda

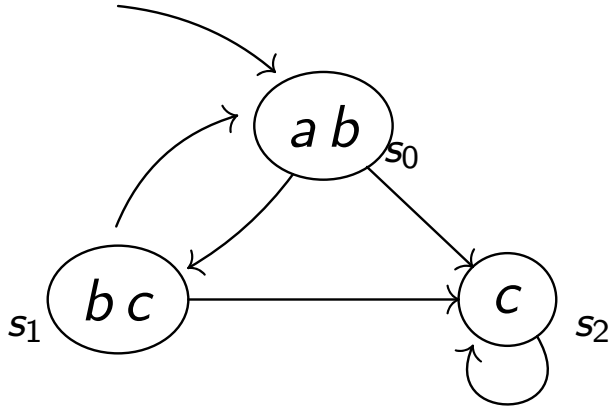
Lógica Temporal

Lógica de Árvore de Computação (CTL)

Verificação de Modelo do CTL

Caminhos de Computação versus Árvores de Computação

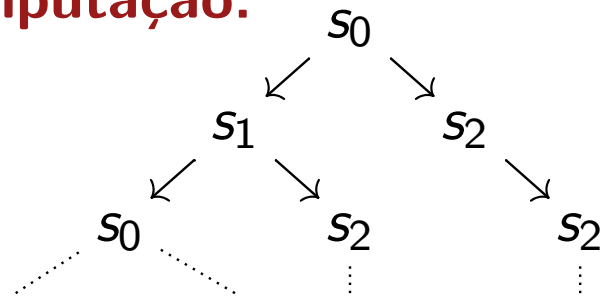
- Estrutura Kripke:



- suposição adicional: cada estado tem no mínimo um sucessor
  $\Rightarrow$ processos infinitos !
- alguns **caminhos de computação:**

$$s_0 \to s_1 \to s_0 \to s_1 \to \ldots$$
$$s_0 \to s_1 \to s_2 \to s_2 \to \ldots$$
$$s_0 \to s_2 \to s_2 \to \ldots$$

- **árvore de computação:**



Lógicas de Tempo Linear versus Tempo de Ramificação (Branching Time)

- modelos de tempo:
  1. tempo linear: conjunto de todos os caminhos de computação
  2. tempo de ramificação: árvore de computação

- lógicas temporais:
  1. lógicas de tempo linear:
     - fórmulas avaliadas em caminhos de computação

       (estrutura de ramificação é abstraída)
     - e.g. "para todos os caminhos, *p* detém *em algum ponto* ao longo do caminho"
  2. lógicas de tempo de ramificação:
     - fórmulas avaliadas em árvores de computação
     - e.g. "*sempre que p* ocorre, é possível alcançar um estado onde *q* ocorre"

Exclusão Mútua: Propriedades de Corretude Revisado

- **exclusão mútua**: no máximo um processo na região crítica *a qualquer momento* (i.e. *em todos os estados alcançáveis*)
- **starvation freedom**: *sempre que* um processo tenta entrar em sua região crítica, ele irá *eventualmente* suceder (*ao longo de cada caminho de computação*)
- **sem bloqueio** (vivacidade): um processo pode sempre, em algum ponto no futuro (*ao longo de algum caminho de computação*), solicitar para entrar em sua região crítica

As *propriedades temporais* acima devem se manter *em todos os estados iniciais*.

Recap em Lógica Proposicional

- assuma um conjunto *Prop* de proposições atômicas
- sintaxe:

  $$\phi, \psi ::= p \mid \mathtt{tt} \mid \neg\phi \mid \phi \wedge \psi \qquad (p \in \mathit{Prop})$$

  Note: todos os outros operadores booleanos são definíveis, por exemplo

  $$\mathtt{ff} ::= \neg\mathtt{tt}$$
  $$\phi \vee \psi ::= \neg(\neg\phi \wedge \neg\psi)$$
  $$\phi \to \psi ::= \neg(\phi \wedge \neg\psi)$$

- modelos são *avaliações*, dizendo quais proposições são verdadeiras e quais não são:

  $$V : \mathit{Prop} \to \{\mathtt{True}, \mathtt{False}\}$$

- significado (*semântica*) das fórmulas:
  - V define o significado das proposições atômicas
  - a tabela da verdade define o significado dos operadores booleanos

Lógica de Árvore de Computação (CTL)

- sintaxe:

$$\phi, \psi ::= p \mid \mathtt{tt} \mid \neg\phi \mid \phi \wedge \psi \mid \mathbf{AX}\,\phi \mid \mathbf{EX}\,\phi \mid \mathbf{AF}\,\phi \mid \mathbf{EF}\,\phi \mid \mathbf{AG}\,\phi \mid \mathbf{EG}\,\phi \mid \mathbf{A}\,[\phi\,\mathbf{U}\,\psi] \mid \mathbf{E}\,[\phi\,\mathbf{U}\,\psi] \qquad (p \in \mathit{Prop})$$

- leituras de fórmulas modais:

$\mathbf{AX}\,\phi$	ϕ holds in **A**ll ne**X**t states
$\mathbf{EX}\,\phi$	there **E**xists a ne**X**t state in which ϕ holds
$\mathbf{AF}\,\phi$	along **A**ll paths, ϕ holds in some **F**uture state
$\mathbf{EF}\,\phi$	there **E**xists a path along which ϕ holds in some **F**uture
$\mathbf{AG}\,\phi$	along **A**ll paths, ϕ holds **G**lobally
$\mathbf{EG}\,\phi$	there **E**xists a path along which ϕ holds **G**lobally
$\mathbf{A}\,[\phi\,\mathbf{U}\,\psi]$	along **A**ll paths, ϕ holds **U**ntil ψ holds
$\mathbf{E}\,[\phi\,\mathbf{U}\,\psi]$	there **E**xists a path along which ϕ holds **U**ntil ψ holds

- exemplo de fórmulas: $\mathbf{AG}\ \mathbf{EF}\,\phi$, $\mathbf{E}\,[\mathtt{tt}\ \mathbf{U}\ \mathbf{AG}\,\phi]$

Algumas Observações na Sintaxe da CTL

- os pares $\mathbf{AX}$, $\mathbf{EX}$, . . . , $\mathbf{A}\,[\,\mathbf{U}\,]$, $\mathbf{E}\,[\,\mathbf{U}\,]$ são indivisíveis !
- prioridades de ligação:
  - $\neg$, $\mathbf{AG}$, $\mathbf{EG}$, $\mathbf{AF}$, $\mathbf{EF}$, $\mathbf{AX}$ e $\mathbf{EX}$ ligam mais firmemente
  - $\wedge$ e $\vee$ estão próximos
  - $\rightarrow$, $\mathbf{A}\,[\,\mathbf{U}\,]$, $\mathbf{E}\,[\,\mathbf{U}\,]$ ligam menos firmemente
- e.g. $\mathbf{EF}\ \mathbf{EG}\,p \rightarrow\ \mathbf{AF}\,r$ significa $(\,\mathbf{EF}\ \mathbf{EG}\,p) \rightarrow\ \mathbf{AF}\,r$
- parênteses podem ser usados para substituir as prioridades acima, e.g. $\mathbf{EF}\,(\,\mathbf{EG}\,p \rightarrow\ \mathbf{AF}\,r)$, $\mathbf{EF}\ \mathbf{EG}\,(p \rightarrow\ \mathbf{AF}\,r)$

Semântica da CTL

Assuma a estrutura Kripke $M = (S, R, V)$.

A semântica da CTL define quando uma fórmula ϕ da CTL é verdadeira *em um estado* $s \in S$.

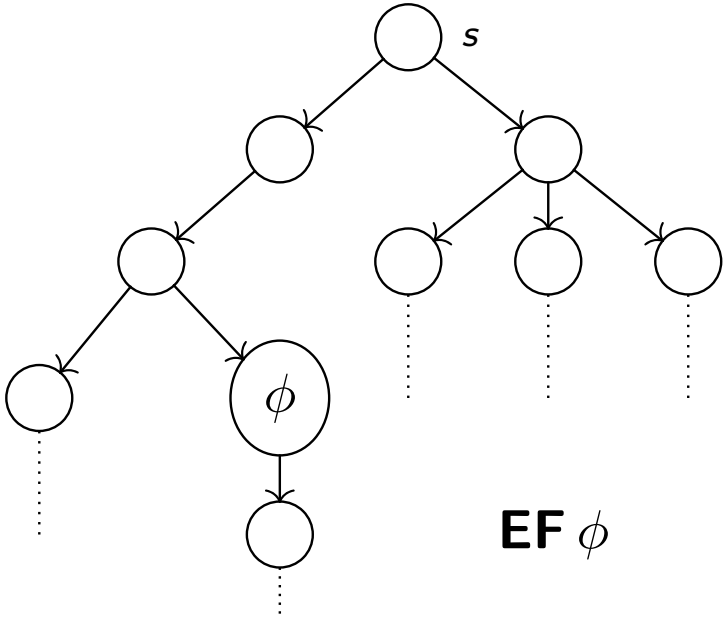
Quando escrevemos $s \models \phi$ (e dizemos que s **satisfaz** ϕ) se ϕ é verdadeiro em s.

A definição de quando $s \models \phi$ é por *indução na estrutura de* ϕ:

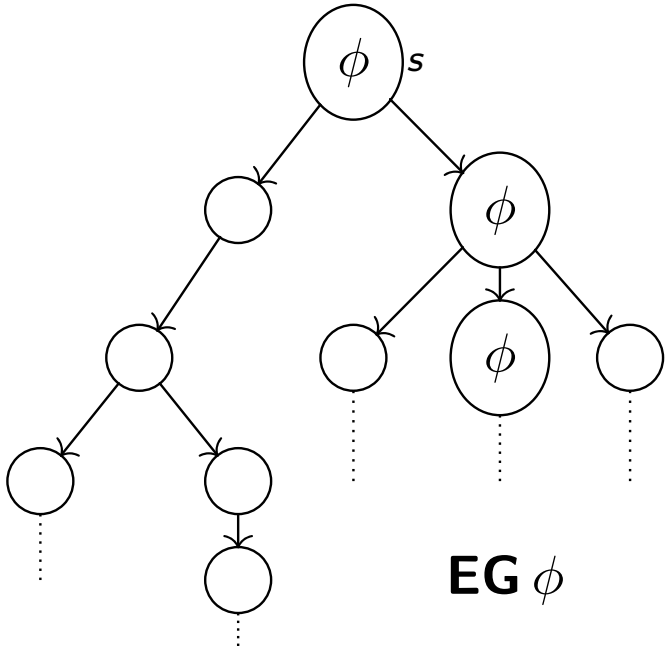
$$
\begin{array}{lll}
s \models p & \text{sse} & V(p, s) = \mathit{true} \\
s \models \neg\phi & \text{sse} & s \models \phi \ \text{ não detém} \\
s \models \phi \wedge \psi & \text{sse} & s \models \phi \ \text{ e } \ s \models \psi
\end{array}
$$

$\ldots$

Semântica da CTL: Exemplo

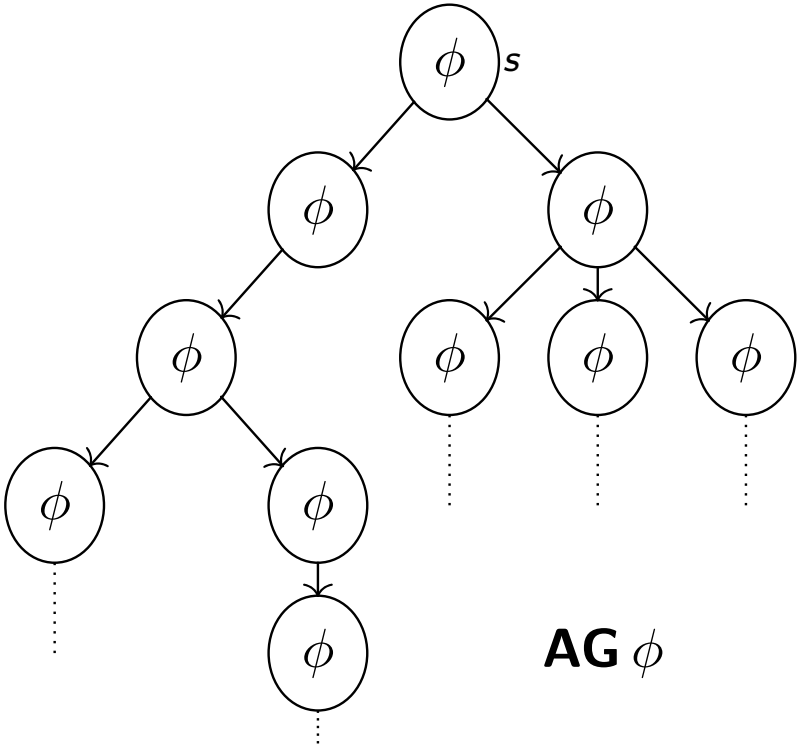


EF ϕ

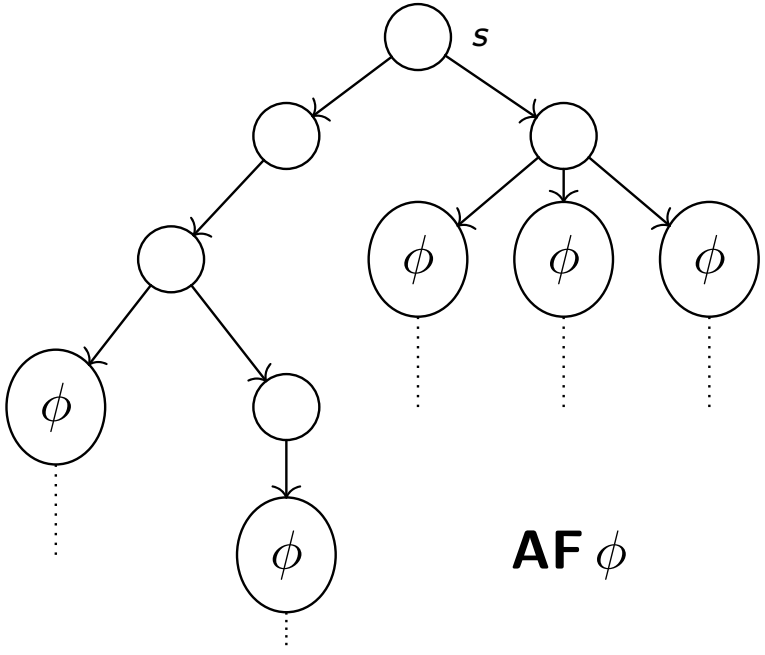


EG ϕ

Semântica da CTL: Exemplo



AG ϕ



AF ϕ

Semântica da CTL: Exemplo (Continuação)

Assuma uma estrutura Kripke $M = (S, R, V)$ e $s \in S$.

$s \models \mathbf{AX}\,\phi$ iff for all s_1 s.t. $s \to s_1$, $s_1 \models \phi$

$s \models \mathbf{EX}\,\phi$ iff there exists s_1 s.t. $s \to s_1$ and $s_1 \models \phi$

$s \models \mathbf{AF}\,\phi$ iff for all paths $s = s_0 \to s_1 \to s_2 \to \ldots$
there exists $i \in \{0, 1, \ldots\}$ s.t. $s_i \models \phi$

$s \models \mathbf{EF}\,\phi$ iff there exist a path $s = s_0 \to s_1 \to s_2 \to \ldots$
and an $i \in \{0, 1, \ldots\}$ s.t. $s_i \models \phi$

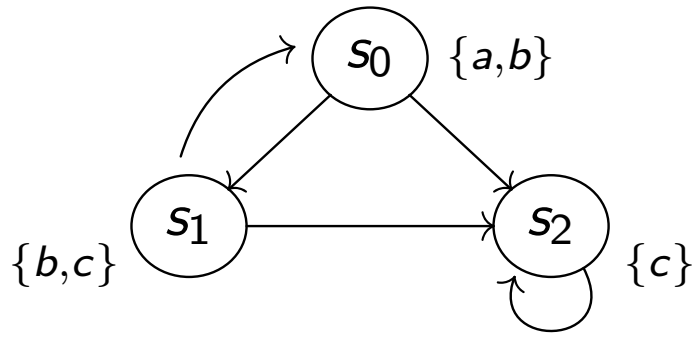
$s \models \mathbf{AG}\,\phi$ iff for all paths $s = s_0 \to s_1 \to s_2 \to \ldots$
for all $i \in \{0, 1, \ldots\}$, $s_i \models \phi$

$s \models \mathbf{EG}\,\phi$ iff there exists a path $s = s_0 \to s_1 \to s_2 \to \ldots$ s.t.
for all $i \in \{0, 1, \ldots\}$, $s_i \models \phi$

$s \models \mathbf{A}\,[\phi\,\mathbf{U}\,\psi]$ iff for all paths $s = s_0 \to s_1 \to s_2 \to \ldots$
there exists $i \in \{0, 1, \ldots\}$ s.t. $s_0 \models \phi, \ldots, s_{i-1} \models \phi, s_i \models \psi$

$s \models \mathbf{E}\,[\phi\,\mathbf{U}\,\psi]$ iff there exists a path $s = s_0 \to s_1 \to s_2 \to \ldots$
and $i \in \{0, 1, \ldots\}$ s.t. $s_0 \models \phi, \ldots, s_{i-1} \models \phi, s_i \models \psi$

Outro Exemplo



$$s_0 \models \mathbf{EX}\,(b \wedge c)$$
$$s_0 \models \neg\,\mathbf{AX}\,(b \wedge c)$$
$$s_0 \not\models \mathbf{EG}\,c$$
$$s_0 \models \mathbf{AG}\ \mathbf{AF}\,c$$
$$s_0 \models \mathbf{A}\,[a\,\mathbf{U}\,c]$$
$$s_0 \models \mathbf{E}\,[(a \wedge b)\,\mathbf{U}\,c]$$
$$s_0 \not\models \mathbf{AG}\,(b \rightarrow\ \mathbf{EG}\,c)$$
$$s_0 \models \mathbf{AG}\ \mathbf{EF}\ \mathbf{EG}\,c$$

Note: $s \models \mathbf{AG}\ \mathbf{AF}\,\phi$ sse ϕ é verdadeiro infinitas vezes ao longo de cada caminho de s.

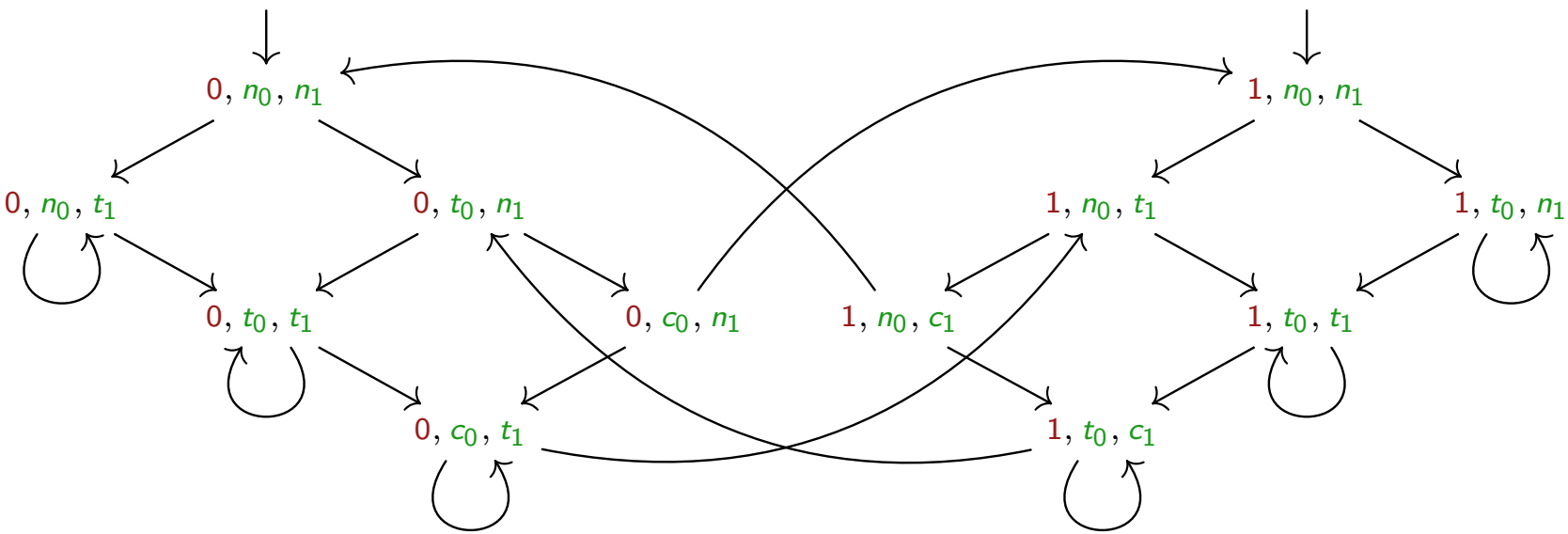
Exclusão Mútua: um Modelo

int $turn$;

$P = m$: **cobegin** $P_0 \parallel P_1$ **coend** m'

$P_0 = n_0$: **while** $True$ **do**
  t_0 : **wait** $(turn = 0)$;
  c_0 : `use resource;` $turn := 1$;
endwhile; n_0'

$P_1 = n_1$: **while** $True$ **do**
  t_1 : **wait** $(turn = 1)$;
  c_1 : `use resource;` $turn := 0$;
endwhile; n_1'



Exclusão Mútua: Propriedades de Corretude

Proposições atômicas:

c_0 , c_1	(estado crítico)
n_0 , n_1	(estado não crítico)
t_0 , t_1	(tentando entrar no estado crítico)

- exclusão mútua: no máximo um processo na região crítica *a qualquer momento*

  $$\mathbf{AG}\,\neg(c_0 \wedge c_1)$$

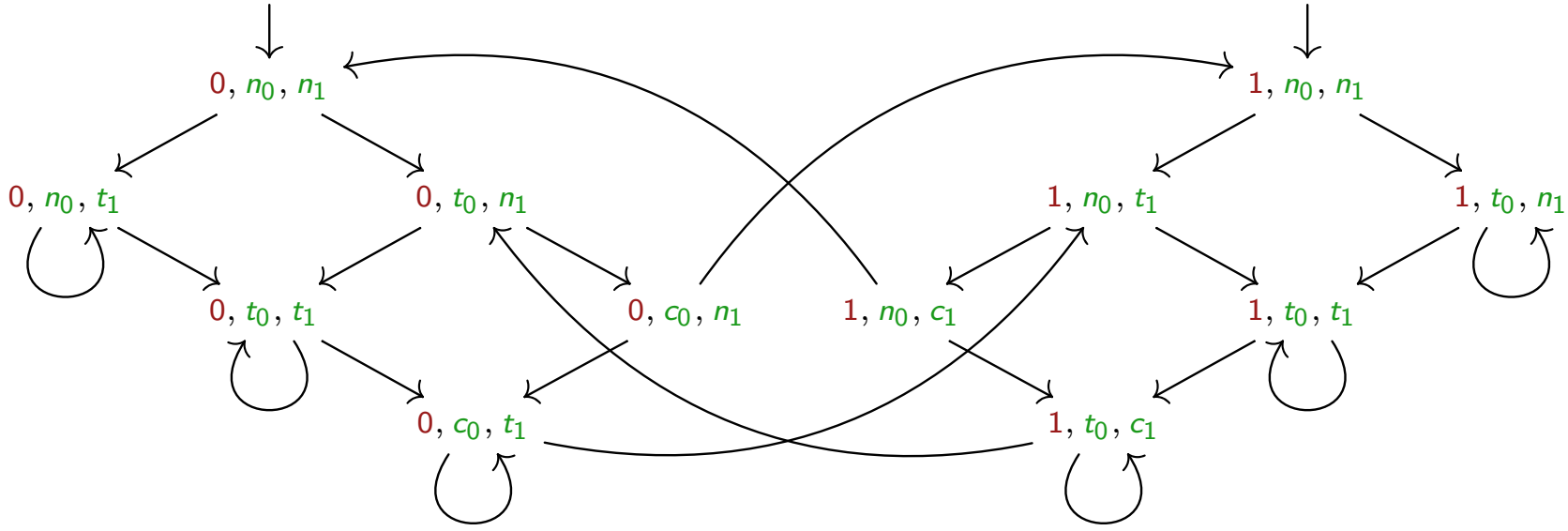
- starvation freedom: *sempre que* um processo tenta entrar na sua região crítica, ele *eventualmente* entrará

  $$\mathbf{AG}\,((t_0 \rightarrow \mathbf{AF}\,c_0) \wedge (t_1 \rightarrow \mathbf{AF}\,c_1))$$

- sem bloqueio: um processo pode sempre, em algum ponto no futuro, pedir para entrar em sua região crítica

  $$\mathbf{AG}\,(\,\mathbf{EF}\,t_0 \wedge \mathbf{EF}\,t_1\,)$$

Exclusão Mútua: Checagem da Corretude



$\mathbf{AG}\,\neg(c_0 \wedge c_1)$ ✓

$\mathbf{AG}\,((t_0 \rightarrow \mathbf{AF}\,c_0) \wedge (t_1 \rightarrow \mathbf{AF}\,c_1))$ × ✓

(precisa do conceito de weak fairness para assegurar a propriedade!)

$\mathbf{AG}\,(\mathbf{EF}\,t_0 \wedge \mathbf{EF}\,t_1)$ ✓

$\mathbf{EF}\,(c_0 \wedge \mathbf{E}\,[c_0\,\mathbf{U}\,(\neg c_0 \wedge \mathbf{E}\,[\neg c_1\,\mathbf{U}\,c_0])]) \vee \ldots$ ×

(difícil verificar a mão!)

Exercício

Assuma as seguintes proposições atômicas: *start*, *ready*, *requested*, *acknowledged*, *enabled*, *running*, *deadlock*.

Especifique as seguintes propriedades de corretude em CTL:

1. É possível/impossível de alcançar um estado onde *start* é verdadeiro mas *ready* não é.
2. Sempre que ocorre uma solicitação (*request*), ela irá eventualmente ser reconhecida (*acknowledged*).
3. Se um processo é habilitado infinitas vezes ao longo de um caminho, então ele executa muitas vezes infinitamente ao longo desse caminho.
4. Aconteça o que acontecer, *deadlock* irá eventualmente ocorrer.
5. A partir de qualquer estado, é possível alcançar um estado *ready*.

Solução do Exercício

Fórmulas atômicas:

$start$, $ready$, $requested$, $acknowledged$, $enabled$, $running$, $deadlock$.

1. É (im)possível de alcançar um estado onde $start$ é verdadeiro mas $ready$ não é.

   $\mathbf{AG}\,\neg(start \wedge \neg ready)$ $\qquad$ $\mathbf{EF}\,(start \wedge \neg ready)$

2. Sempre que ocorre uma solicitação ($request$), ela irá eventualmente ser reconhecida ($acknowledged$).

   $\mathbf{AG}\,(requested \rightarrow \mathbf{AF}\,acknowledged)$

3. Se um processo é habilitado infinitas vezes ao longo de um caminho, então ele executa muitas vezes infinitamente ao longo desse caminho.

   Não é expresso em CTL!

4. Aconteça o que acontecer, $deadlock$ eventualmente ocorrerá.

   $\mathbf{AF}\,deadlock$

5. A partir de qualquer estado, é possível alcançar um estado $ready$.

   $\mathbf{AG}\,(\,\mathbf{EF}\,ready)$

Algumas Equivalências entre Fórmulas CTL

- equivalência semântica: $\phi \equiv \psi$ se qualquer estado de qualquer modelo que satisfaz ϕ também satisfaz ψ, e reciprocamente.

- algumas equivalências:

$$\mathbf{AX}\,\phi \equiv \neg\,\mathbf{EX}\,\neg\phi$$
$$\mathbf{AG}\,\phi \equiv \neg\,\mathbf{EF}\,\neg\phi$$
$$\mathbf{AF}\,\phi \equiv \mathbf{A}\,[\mathtt{tt}\,\mathbf{U}\,\phi]$$
$$\mathbf{EG}\,\phi \equiv \neg\,\mathbf{AF}\,\neg\phi$$
$$\mathbf{EF}\,\phi \equiv \mathbf{E}\,[\mathtt{tt}\,\mathbf{U}\,\phi]$$

$\Longrightarrow$ $\mathbf{A}\,[\_\,\mathbf{U}\,\_]$, $\mathbf{E}\,[\_\,\mathbf{U}\,\_]$, $\mathbf{EX}$ suficiente para expressar todos os outros operadores CTL (conjunto adequado de conectivos CTL).

Sub-conjunto Suficiente de Operadores CTL

Teorema. Um conjunto de conectivos temporais em CTL é adequado, se e somente se ele contiver:

- pelo menos um dos **AX** , **EX**
- pelo menos um dos **EG** , **AF** , **A** [_ **U** _]
- **E** [_ **U** _]

Corolário. Os operadores **EX** , **AF** e **E** [_ **U** _] são suficientes para expressar todas os outros.

Algoritmo de Verificação de Modelo para CTL

Entradas: estrutura Kripke finita M, fórmula CTL ϕ contendo apenas **EX**, **AF** e **E** $[\_\,\mathbf{U}\,\_]$ como operadores temporais.

Saídas: os estados de M que satisfazem ϕ

Algoritmo:

- Gerar todas as subfórmulas de ϕ e ordená-las por complexidade (proposições atômicas primeiro, ϕ por último).
- Repetir: pegar uma sub-fórmula da lista e *anotar* os estados de M que sejam satisfeitos por esta fórmula.
- Quando nós alcançarmos o fim da lista, sabemos quais estados satisfazem ϕ.

Como Anotar?

- A avaliação de V nos diz como anotar os estados com proposições atômicas
- Todos os estados são anotados com `tt`.
- Quando uma sub-fórmula atual é $\neg\phi$, nós anotamos com isto os estados que não estão com ϕ (note que ϕ precede $\neg\phi$ na lista de sub-fórmulas, assim nós já anotamos os estados com ϕ).
- Da mesma forma para $\phi_1 \wedge \phi_2$ ...

Como Anotar? (Continuação)

- Para $\mathbf{EX}\,\phi$, anotar antecessores de qualquer estado rotulado ϕ por $\mathbf{EX}\,\phi$.
- Para $\mathbf{E}\,[\phi_1\,\mathbf{U}\,\phi_2]$, primeiro encontrar todos os estados anotados por ϕ_2 e anotá-los por $\mathbf{E}\,[\phi_1\,\mathbf{U}\,\phi_2]$. Então trabalhe para trás a partir daqueles estados e todo tempo que nós encontrarmos os estados ϕ_1, anotá-los com $\mathbf{E}\,[\phi_1\,\mathbf{U}\,\phi_2]$.
- Para $\mathbf{AF}\,\phi$, primeiro anote todos os estados anotados com ϕ por $\mathbf{AF}\,\phi$. Então anote um estado com $\mathbf{AF}\,\phi$ se todos os seus sucessores estiverem anotados com $\mathbf{AF}\,\phi$. Repita até que não exista nenhuma mudança.

Notas:

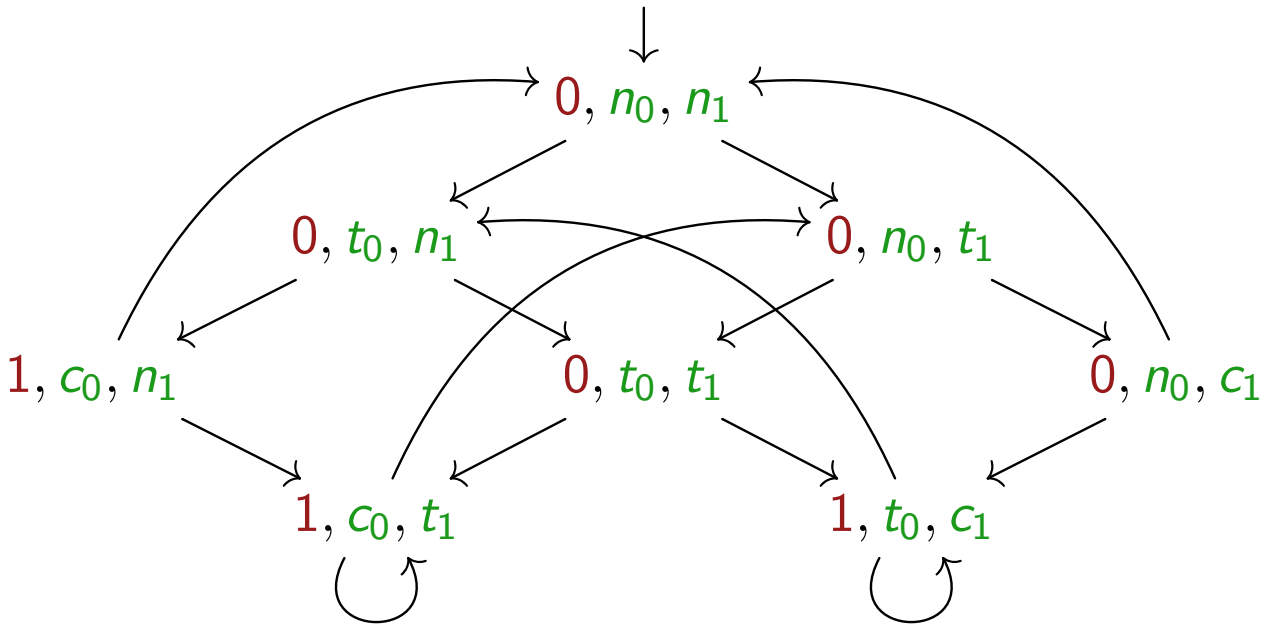
- os operadores acim são suficientes para expressar todos os outros operadores do CTL,
- as equivalências semânticas dadas anteriormente precisam ser usadas quando a fórmula a ser verificada contém os operadores temporais que não $\mathbf{EX}$, $\mathbf{AF}$ e $\mathbf{E}\,[\_\,\mathbf{U}\,\_]$.

Observações sobre o Algoritmo

- Existe uma variante do algoritmo que considera apenas os caminhos de computação que são justos.
- A complexidade de tempo do algoritmo é quadrático no tamanho do modelo (número de estados + número de transições) e linear no tamanho da fórmula (número de conectivos).
- Existe uma melhoria do algoritmo (substituindo **AF** por **EG**) cuja complexidade de tempo é linear no tamanho do modelo e no tamanho da fórmula.
- Isto ainda *não é bom o suficiente*, tendo em vista que o tamanho do modelo é exponencial no número de variáveis e no número de componentes paralelos ...
- O verificador NuSMV usa estruturas de dados chamada de **OBDDs** (**ordered binary decision diagrams**) para representar *conjunto de estados* em vez de *estados individuais*.

Exercício

Considere o seguinte modelo de exclusão mútua:



Verifique se o estado inicial deste modelo satisfaz as fórmulas $\mathbf{E}\,[\neg c_1\ \mathbf{U}\ c_0]$ e $\mathbf{AF}\ c_0$.