Lógica Computacional
Verificação com Lógicas Temporais

Cláudia Nalon

Universidade de Brasília

Instituto de Ciências Exatas

Departamento de Ciência da Computação

nalon@cic.unb.br

Motivação

- Lógicas temporais foram introduzidas para formalizar conceitos temporais presentes no discurso (lingüística) [Prior, 1967];
- Mais recentemente, algumas variações de lógicas temporais têm sido utilizadas para formalizar propriedades de sistemas computacionais;
- A idéia é tirar vantagem dos aspectos formais da linguagem lógica para verificar automaticamente se os requisitos dos sistemas são atendidos.

Verificação

- É importante verificar a correção de sistemas computacionais: *hardware*, *software* ou a combinação de ambos;
- Isto é mais óbvio no caso de **safety-critical systems**, sistemas cuja falha pode causar o ferimento ou a morte de seres humanos:
  - sistema de controle de uma aeronave;
  - sistema de controle de uma usina nuclear;
  - sistema de controle de tráfego.
- Mas verificação também é importante no caso de **commercially critical systems**, aqueles cuja falha pode causar imenso prejuízo à indústria e a seus consumidores, como, por exemplo, a produção em massa de componentes eletrônicos;
- Muitos outros exemplos...

Verificação

O processo requer três componentes:

- uma linguagem para a modelagem;
- uma linguagem de especificação;
- um método de verificação.

Os enfoques se caracterizam pelos seguintes critérios:

- método de verificação: apresentação de uma prova ou verificação de um modelo;
- grau de automação;
- verificação completa ou de propriedades;
- domínio de aplica ção;
- período de aplicação.

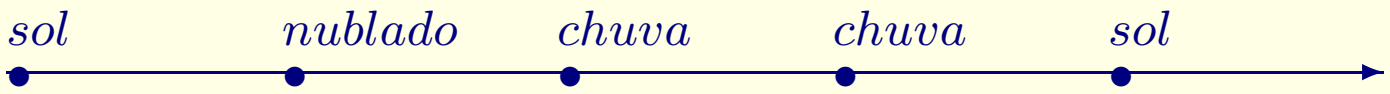
Lógicas Temporais

- linear $\times$ *branching*:
  - um único futuro;
  - vários possíveis futuros.
- contínua $\times$ discreta:
  - índices reais;
  - índices naturais.

Modelo para o Tempo

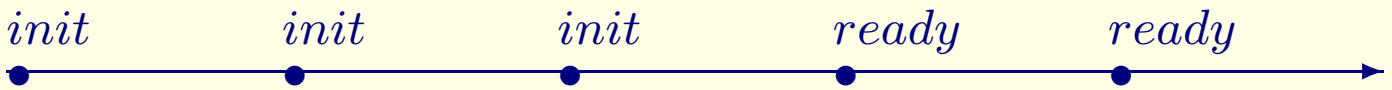
Nós nos concentraremos em lógica temporal proposicional, linear e discreta.

- os modelos são seqüências infinitas de estados;
- cada estado representa a situação em um determinado momento.

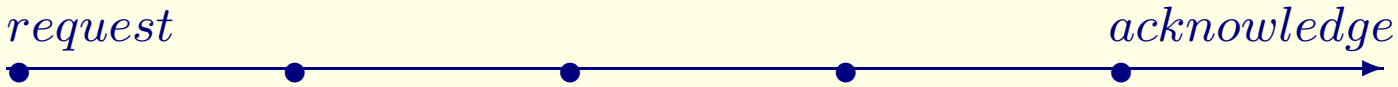


Exemplos - I

- Ao final da inicialização, o serviço está pronto para atender solicitações;



- Se uma requisição ocorrer, ela será em algum momento confirmada;



Exemplos - II

- Um determinado recurso sempre será liberado:

- Uma determinada situação x provoca *deadlock*:

A Linguagem da Lógica Temporal

Símbolos Proposicionais: $\mathcal{P} = \{p, q, r, p', q', r', \ldots\}$

Constantes: $\mathrm{true}, \mathrm{false}$

Classicos: $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$

Temporais: $\Diamond$, $\Box$, $\bigcirc$, $\mathcal{U}$, $\mathcal{W}$

pontuação: $($ e $)$.

Fórmulas são seqüências de símbolos lógicos.

Sintaxe – $\mathrm{FBF}_{\mathrm{PLTL}}$

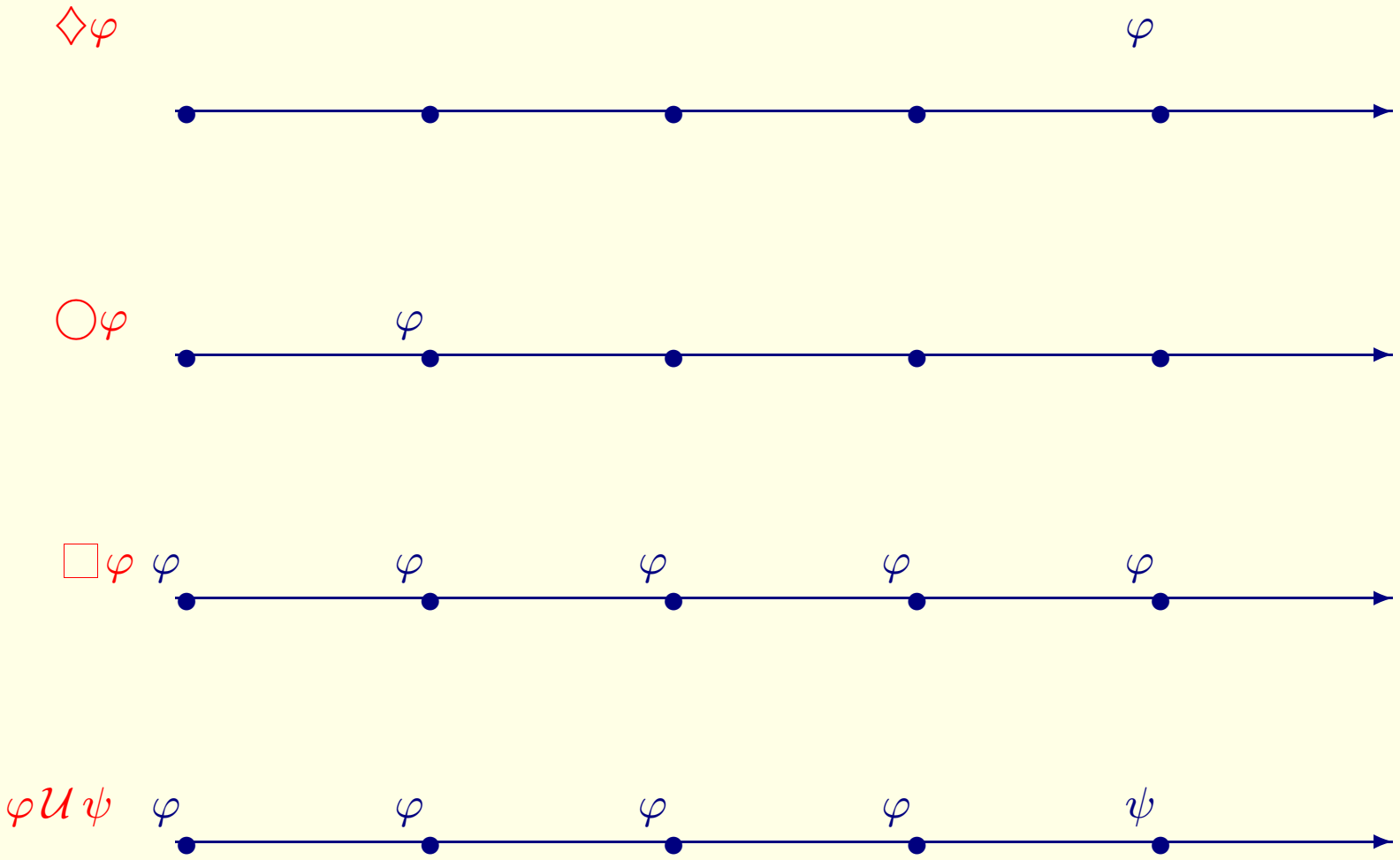
Fórmulas bem-formadas ($\mathrm{FBF}_{\mathrm{PLTL}}$) são construídas indutivamente:

- $p \in \mathcal{P}$ é uma fórmula bem-formada;
- se φ e ψ são fórmulas bem formadas, então:

  $$\neg(\varphi), (\varphi \vee \psi), (\varphi \wedge \psi), (\varphi \rightarrow \psi), \Diamond\varphi, \Box\varphi, \bigcirc\varphi, (\varphi \,\mathcal{U}\, \phi), (\varphi \,\mathcal{W}\, \phi)$$

  são fórmulas bem-formadas.

Semântica dos Operadores



Exemplo – Sintaxe

- Ao final da inicialização, o serviço está pronto para atender solicitações;

$$(\texttt{init}\,\mathcal{U}\,\texttt{end-init}) \wedge (\texttt{end-init} \rightarrow \bigcirc \Box\, \texttt{ready})$$

- Se uma requisição ocorrer, ela será em algum momento confirmada;

$$\Box(\texttt{request} \rightarrow \Diamond \texttt{acknowledge})$$

- Um determinado recurso sempre será liberado:

$$\Box \Diamond \texttt{ready}$$

- Uma determinada situação x provoca *deadlock*:

$$\Box(x \rightarrow \Box\, \texttt{deadlock})$$

- Um elevador não muda a direção:

$$\Box((\texttt{floor\_2} \wedge \texttt{up} \wedge \texttt{pressed\_5}) \rightarrow (\texttt{up}\,\mathcal{U}\,\texttt{floor\_5}))$$

Semântica – Estrutura de Kripke

A semântica é definida em termos de **Estruturas de Kripke**.

Uma estrutura de Kripke M sobre $\mathcal{P}$ é uma tupla:

$$M = \langle \mathcal{S}, \mathcal{R}, \pi \rangle$$

onde

- $\mathcal{S}$ é um conjunto (de mundos ou estados possíveis);
- $\mathcal{R}$ é uma relação serial em $\mathcal{S}$;
- π é uma interpretação que associa, para cada estado de S, uma valoração para os símbolos proposicionais, ou seja, $\pi(s) : \mathcal{P} \longrightarrow \{V, F\}$; e

Um modelo σ é, portanto, uma seqüência infinita de estados:

$$\sigma = s_0, s_1, s_2, \ldots$$

Semântica – Interpretação

Sejam $p \in \mathcal{P}$ e $\varphi, \psi \in \mathsf{FBF}_{\mathsf{PLTL}}$ e σ um modelo:

- $(\sigma, i) \models p$ se, e somente se, $\pi(s_i)(p) = V$;
- $(\sigma, i) \models \varphi \wedge \psi$ se, e somente se, $(\sigma, i) \models \varphi$ e $(M, s) \models \psi$;
- $(\sigma, i) \models \neg\varphi$ se, e somente se, $(\sigma, i) \not\models \varphi$;
- $(\sigma, i) \models \bigcirc\varphi$ if, and only if, $(\sigma, i+1) \models \varphi$
- $(\sigma, i) \models \Diamond\varphi$ if, and only if, $\exists k, k \in \mathbb{N}, k \geq i, (\sigma, k) \models \varphi$
- $(\sigma, i) \models \Box\varphi$ if, and only if, $\forall k, k \in \mathbb{N}$, if $k \geq i$, then $(\sigma, k) \models \varphi$
- $(\sigma, i) \models \varphi\,\mathcal{U}\,\psi$ if, and only if, $\exists k, k \in \mathbb{N}, k \geq i, (\sigma, k) \models \psi$ and $\forall j, j \in \mathbb{N}$, if $i \leq j < k$, then $(\sigma, j) \models \varphi$;
- $(\sigma, i) \models \varphi\,\mathcal{W}\,\psi$ if, and only if, either $(\sigma, i) \models \varphi\,\mathcal{U}\,\psi$ or $(\sigma, i) \models \Box\varphi$.

Exemplo – Exclusão Mútua

Quando processos concorrentes compartilham determinado recurso, pode ser necessário ter que assegurar que o acesso a este recurso não seja feito ao mesmo tempo.

Nós identificamos determinadas **regiões críticas** nos códigos dos processos e fazemos com que apenas um único processo possa estar em sua região crítica.

O problema é encontrar um **protocolo** para determinar qual processo terá o direito de entrar em sua região crítica. Uma vez que este protocolo seja determinado, nós verificamos nossa solução através da checagem de determinadas propriedades.

Exemplo – Exclusão Mútua – Propriedades

- ***Safety***: em qualquer momento, apenas um processo está em sua região crítica;
- ***Liveness*** se um processo solicitar a entrada em sua região crítica, em algum momento ele terá permissão para fazê-lo;
- ***Non-blocking***: todo processo tem o direito de requisitar a entrada em sua região crítica;
- ***No strict sequencing***: não é necessário que a entrada na região crítica se dê em seqüência.

Exemplo – Exclusão Mútua – Modelagem

Possíveis situações:

- processo não está na sua região crítica: n_i;
- processo está tentando entrar na sua região crítica: t_i;
- processo está na sua região crítica: c_i.

Cada processo passa pelo seguinte ciclo:

$$n_i \Longrightarrow t_i \Longrightarrow c_i \Longrightarrow n_i \Longrightarrow \ldots$$

Os processos podem ter os seus ciclos intercalados.

Exemplo – Exclusão Mútua – Modelagem

No começo, nenhum dos dois processos está em sua região crítica:

$$\mathbf{start} \rightarrow (n_1 \wedge n_2)$$

Mas algum dos dois processos irá tentar entrar em sua região crítica:

$$\Box((n_1 \wedge n_2) \rightarrow \bigcirc((t_1 \wedge n_2) \vee (n_1 \wedge t_2)))$$

Se um dos processos está tentando entrar em sua região crítica, ou ele obtém permissão ou o outro processo pode tentar entrar também:

$$\Box((t_1 \wedge n_2) \rightarrow \bigcirc((c_1 \wedge n_2) \vee (t_1 \wedge t_2)))$$

$$\Box((n_1 \wedge t_2) \rightarrow \bigcirc((n_1 \wedge c_2) \vee (t_1 \wedge t_2)))$$

Exemplo – Exclusão Mútua – Modelagem

Se apenas um dos processos está na sua região crítica, ou ele sai ou o outro tenta entrar:

$$\Box((c_1 \wedge n_2) \rightarrow \bigcirc((n_1 \wedge n_2) \vee (c_1 \wedge t_2)))$$

$$\Box((n_1 \wedge c_2) \rightarrow \bigcirc((n_1 \wedge n_2) \vee (t_1 \wedge c_2)))$$

Se os dois processos estão tentando entrar na região crítica, um (e apenas um) irá obter permissão:

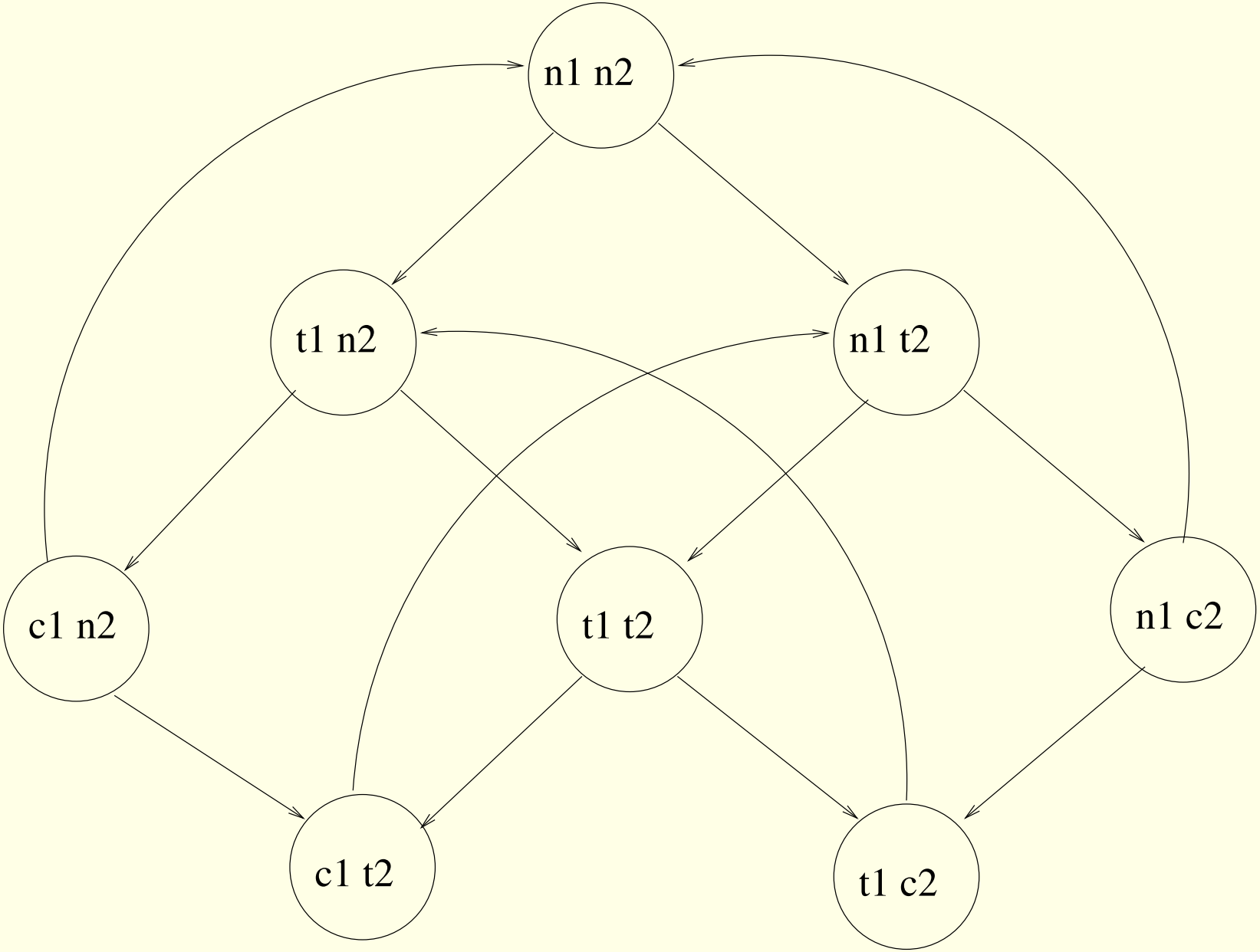
$$\Box((t_1 \wedge t_2) \rightarrow \bigcirc((c_1 \wedge t_2) \vee (t_1 \wedge c_2)))$$

Se um um processos está na região crítica, ele sai:

$$\Box((c_1 \wedge t_2) \rightarrow \bigcirc(n_1 \wedge t_2))$$

$$\Box((t_1 \wedge c_2) \rightarrow \bigcirc(t_1 \wedge n_2))$$

Exemplo – Exclusão Mútua



Exemplo – Exclusão Mútua – Verificação

Safety: É satisfeita pela especificação

$$\Box \neg (c_1 \wedge c_2)$$

Liveness: É satisfeita pela especificação

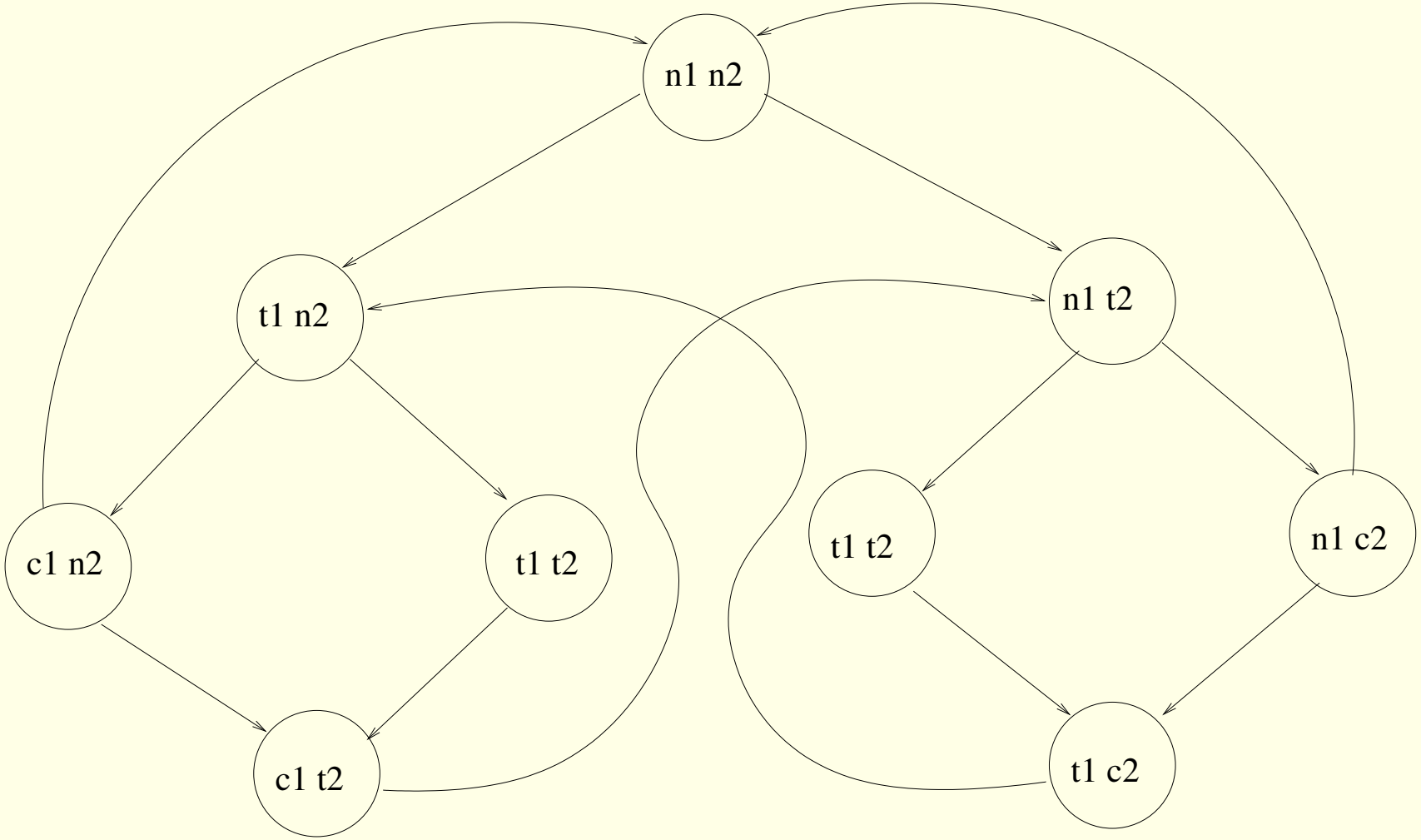
$$\Box (t_1 \rightarrow \Diamond c_1)$$

Non-blocking: É satisfeita, mas não pode ser expressa.

No strict sequencing: Não é satisfeita pela especificação

$$\Box (c_1 \rightarrow c_1 \,\mathcal{W}\, (\neg c_1 \wedge \neg c_1 \,\mathcal{W}\, c_2))$$

Exemplo – Exclusão Mútua – Segunda Modelagem



Conclusões

- Lógicas temporais formalizam de modo natural problemas computacionais;
- A verificação das especificações pode ser feita automaticamente, através de **provadores de teorema** ou **checagem de modelos**;
- O problema da validação é PSPACE para o fragmento proposicional e indecidível para o fragmento de primeira ordem;
- A combinação de lógicas temporais com outras lógicas modais produz uma linguagem de especificação que vem sendo utilizada em várias aplicações, tais como verificação de protocolos e de sistemas reativos.
- Muito a fazer :)