Aula 2

1 Model checking

Model checking

Os sistema reactivos são descritos pelas suas propriedades temporais. Uma especificação é verificada se for satisfeita ao longo da execução do sistema.

Assim, o *Model checking* é baseado em Lógicas temporais.

Um modelo temporal $\mathcal{M}$ é constituído por um conjunto de estados e transições entre eles (sistema de transições).

Uma fórmula ϕ pode ser **V** nalguns estado e **F** noutros.

A noção de verdade não é absoluta, mas sim dinâmica. Dado um estado s, pretende-se saber se:

$$\mathcal{M}, s \models \phi$$

O verificador de modelos (*model checker*) é um algoritmo que decide este problema (responde sim ou não).

Propriedades

***Reachability* (estados atingíveis)** propriedades que garantem que um dado estado pode ser atingido

***Safety* (segurança)** propriedades que têm de se verificar para todas as computações e em todos os estados (*Something bad never happens*)

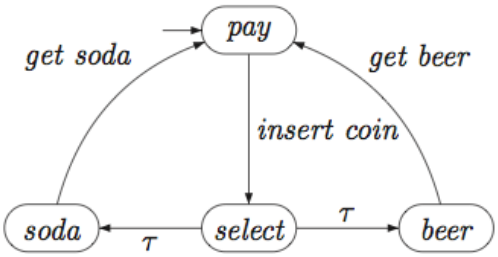
***Liveness* (vivacidade)** indicam que determinados estados do sistema têm de ser atingidos: restringue compotações infinitas mas não finitas (*Something good will eventually happen*)

Persistência indicam que para todas as computações a partir de um dado estado uma certa propriedade tem de se verificar.

***Fairness* (razoabilidade)** indicam que uma dada propriedade tem de se verificar um número não finito de vezes. Exemplo: nenhum processo é esquecido sempre por um temporizador.

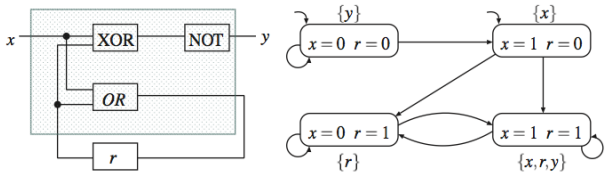
2 Sistemas de Transições

Sistemas de Transições - I



Sistemas de Transições - II

Um circuito sequencial com um registo de um bit r, tal que $f(y) = \neg(x \oplus r)$ e $g(r) = x \vee r$.

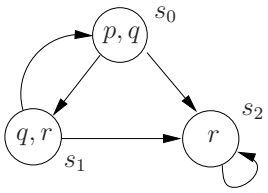


Sistemas de Transições (Modelos de Kripke)

Sistema de transições (Modelo)

Dado um conjunto de variáveis proposicionais (`Atoms`), é um trio $\mathcal{M} = (S, \rightarrow, L)$ onde S é um conjunto de estados, $\rightarrow \subseteq S \times S$, relação binária total (i.e $\forall s \in S, \exists s' \in S,\ s \rightarrow s'$) e $L : S \longrightarrow \mathcal{P}(\texttt{Atoms})$ uma função de etiquetagem que associa a um estado um conjunto de variáveis proposicionais.

Um modelo é representado por um digrafo. Se $\rightarrow$ não for total (se encrava...), basta acrescentar um estado `morto`, que só tenha transições para ele...



Paralelismo e comunicação

Normalmente um sistema informático será modelado por sistemas de transições que funcionam em sequência ou, mais frequentemente, em paralelo.

$$ST_1||ST_2 \cdots ||ST_n$$

Existem muitas maneiras de modelar o paralelismo, aqui apenas enunciamos algumas possibilidades:

- Processos intercalados (*interleaving*) (assíncronos). Ex: processos independentes associados a semáforos de tráfego em ruas distintas.
- Comunicação por variáveis partilhadas
- *Handshaking* (uma ação síncroniza os processos)
- Comunicação por canais (filas de mensagens, etc.)

Algumas exemplos de propriedades

As variáveis proposicionais correspondem a estados de uma sistema reactivo real.

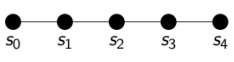
- Não é possível chegar a um estado em que `started` se verifica e `ready` não se verifica.
- Para qualquer estado, se `request` se verifica, então no futuro também se irá verificar `ack`.
- Um processo é activado (`enabled`) um número infinito de vezes em todos os caminhos de computação.
- Um processo irá ficar permanentemente em `deadlock`.
- Se um processo é activado (`enabled`) um número infinito de vezes, então ele executa (`run`) um número infinito de vezes.
- Para todos os estados, existe uma caminho para um estado que satisfaz `restart`

3 Noção de Tempo

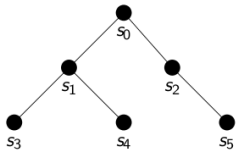
Tempo

Os instantes temporais são denominados estados.

Linear: O tempo é um conjunto de caminhos, onde um caminho é uma sequência de estados.



Ramificado: O tempo é representado em árvore com raíz no momento presente.



4 Lógica Temporal Linear, LTL

Lógica Temporal Linear, LTL

`Atoms`, conjunto de variáveis proposicionais $p, q, r, s, \ldots$

Sintaxe

$$\begin{aligned}\phi \quad ::= \quad & \bot \mid \top \mid p \mid (\neg\phi) \mid (\phi \ \wedge \ \phi) \mid (\phi \ \vee \ \phi) \mid (\phi \rightarrow \phi) \mid \\ & (\mathrm{X}\phi) \mid (\mathrm{F}\phi) \mid (\mathrm{G}\phi) \mid (\phi \mathrm{U} \phi) \mid (\phi \mathrm{W} \phi) \mid (\phi \mathrm{R} \phi)\end{aligned}$$

Conectivas Temporais

$\mathbf{X}\phi$ ϕ verifica-se no estado seguinte (*neXt*)

$\mathbf{F}\phi$ ϕ verifica-se nalgum estado futuros

$\mathbf{G}\phi$ ϕ verifica-se em todos os estados futuros (*Global*)

$\phi\mathbf{U}\psi$ ϕ verifica-se até ψ se verificar (*Until*)

$\phi\mathbf{W}\psi$ ϕ verifica-se até ψ se verificar ou sempre (*Weak until*)

$\phi\mathbf{R}\psi$ ψ verifica-se até ϕ se verificar (inclusivé!) ou sempre (*Release*)

Lógica Temporal Linear, LTL

Exemplos

$$\begin{aligned}&(((\mathrm{F}p) \ \wedge \ (\mathrm{G}p)) \rightarrow (p \mathrm{W} r)) \\ &((\mathrm{G}(\mathrm{F}p)) \rightarrow (\mathrm{F}(q \ \vee \ p))) \\ &(p \mathrm{W} (q \mathrm{W} r))\end{aligned}$$

Convensão de prioridades (omissão de parêntesis)

- As conectivas unárias ($\neg$, X, F,G) têm prioridade mais alta

- Depois as conectivas U, W e R.
- Depois as conectivas $\wedge$ e $\vee$.
- Depois a conectiva $\rightarrow$.

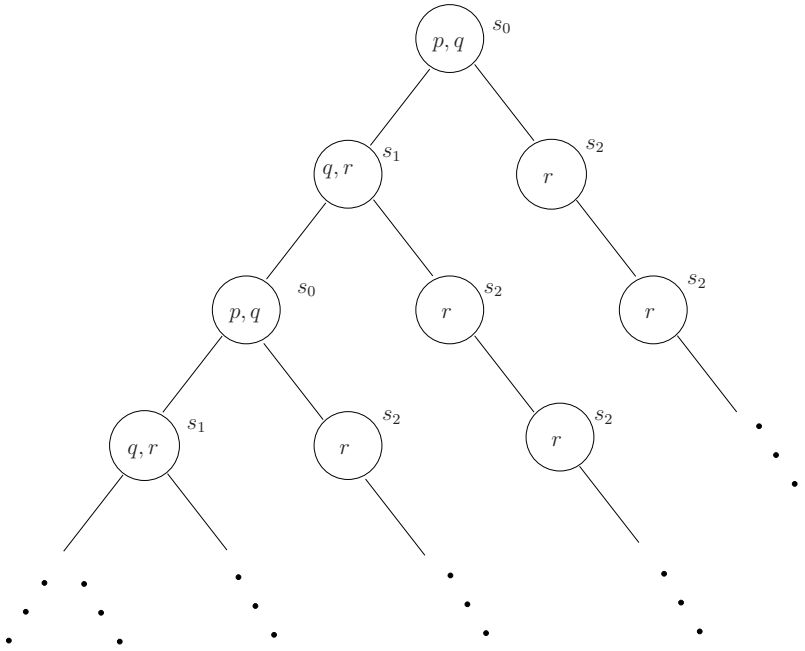
$$\begin{array}{l} \mathrm{F}p \ \wedge \ \mathrm{G}p \rightarrow p\mathrm{W}r \\ \mathrm{GF}p \rightarrow \mathrm{F}(q \ \vee \ p) \\ p\mathrm{W}(q\mathrm{W}r) \end{array}$$

Árvore Infinita de Computação

Caminho (de Computação)

Um caminho num modelo $\mathcal{M} = (S, \rightarrow, L)$ é uma sequência infinita de estados $s_1, s_2, s_3, \ldots$ em S, tal que para todo $i \geq 1$, $s_i \rightarrow s_{i+1}$.Um caminho π é representado por $\pi = s_1 \rightarrow s_2 \rightarrow \ldots$. Dado um caminho π, π^i é o sufixo que começa em s_i.

O conjunto de caminhos a partir dum estado s pode ser visto como uma árvore infinita de computação.



5 Semântica do LTL

Semântica do LTL

Satisfazibilidade

Dado um modelo $\mathcal{M} = (S, \rightarrow, L)$ e um caminho $\pi = s_1 \rightarrow \ldots$, define-se a relação de satisfabilidade $\models$ indutivamente por:

1. $\pi \models \top$
2. $\pi \not\models \bot$
3. $\pi \models p$ sse $p \in L(s_1)$
4. $\pi \models \neg\phi$ sse $\pi \not\models \phi$
5. $\pi \models \phi \wedge \psi$ sse $\pi \models \phi$ e $\pi \models \psi$
6. $\pi \models \phi \vee \psi$ sse $\pi \models \phi$ ou $\pi \models \psi$
7. $\pi \models \phi \rightarrow \psi$ sse sempre que $\pi \models \phi$ então $\pi \models \psi$
8. $\pi \models \mathrm{X}\phi$ sse $\pi^2 \models \phi$
9. $\pi \models \mathrm{G}\phi$ sse $\forall i \geq 1, \pi^i \models \phi$
10. $\pi \models \mathrm{F}\phi$ sse $\exists i \geq 1, \pi^i \models \phi$
11. $\pi \models \phi \mathrm{U} \psi$ sse $\exists i \geq 1, \pi^i \models \psi$ e $\forall\, 1 \leq j < i$, $\pi^j \models \phi$
12. $\pi \models \phi \mathrm{W} \psi$ sse ou $\exists i \geq 1, \pi^i \models \psi$ e $\forall\, 1 \leq j < i$, $\pi^j \models \phi$ ou $\forall k \geq 1$, $\pi^k \models \phi$
13. $\pi \models \phi \mathrm{R} \psi$ sse ou $\exists i \geq 1, \pi^i \models \phi$ e $\forall\, 1 \leq j \leq i$, $\pi^j \models \psi$ ou $\forall k \geq 1$, $\pi^k \models \psi$

Semântica do LTL

Satisfazibilidade num estado

Seja $\mathcal{M} = (S, \rightarrow, L)$, $s \in S$ e ϕ uma fórmula do LTL. Diz-se que $\mathcal{M}, s \models \phi$, se para todos os caminhos de computação π começando em s, se tem $\pi \models \phi$.

Exercício 2.1. *Considera o modelo* $\mathcal{M} = (S = \{s_0, s_1, s_2\}, \{s_0 \rightarrow s_1, s_0 \rightarrow s_2, s_1 \rightarrow s_2, s_1 \rightarrow s_0, s_2 \rightarrow s_2\}, L(s_0) = \{p, q\}, L(s_1) = \{q, r\}, L(s_2) = \{r\})$. *Determina quais destas relações são verdadeiras:*

1. $\mathcal{M}, s_0 \models p \wedge q$
2. $\mathcal{M}, s_0 \models Xr$
3. $\mathcal{M}, s_0 \models X(q \wedge r)$

4. $\mathcal{M}, s_0 \models G\neg(p \wedge r)$
5. $\mathcal{M}, s_0 \models GFp$
6. $\mathcal{M}, s_0 \models GFp \rightarrow GFr$

$\diamond$

Algumas especificações práticas (padrões)

As variáveis proposicionais correspondem a estados de uma sistema reactivo real.

- Não é possível chegar a um estado em que `started` se verifica e `ready` não se verifica: G$\neg$(`started` $\wedge$ $\neg$`ready`)
- Para qualquer estado, se `request` se verifica, então no futuro também se irá verificar `ack`: G(`request` $\rightarrow$ F`ack`)
- Um processo é activado (`enabled`) um número infinito de vezes em todos os caminhos de computação: GF`enabled`
- Um processo irá ficar permanentemente em `deadlock`: FG`deadlock`
- Se um processo é activado (`enabled`) um número infinito de vezes, então ele executa (`run`) um número infinito de vezes: GF`enabled` $\rightarrow$ GF`run`
- Para todos os estados, existe uma caminho para um estado que satisfaz `restart`: Não se pode exprimir em LTL!

Em LTL não se pode exprimir a existência de um caminho!

Exercício 2.2. *Considera o modelo* $\mathcal{M} = (S = \{q_1, q_2, q_3, q_4\}, \{q_1 \rightarrow q_2, q_2 \rightarrow q_2, q_3 \rightarrow q_1, q_3 \rightarrow q_2, q_3 \rightarrow q_4, q_4 \rightarrow q_3\}, L(q_1) = \{\}, L(q_2) = \{b\}, L(q_3) = \{a\}, L(q_4) = \{a, b\})$.
Para cada uma das fórmulas seguintes:

a) Ga;

b) aUb;

c) $aUX(a \wedge \neg b)$;

d) $X\neg b \wedge G(\neg a \vee \neg b)$;

e) $X(a \wedge b) \wedge F(\neg a \wedge \neg b)$;

- *encontra um caminho a partir do estado* q_3 *que satisfaz* φ;
- *determina se* $\mathcal{M}, q_3 \models \varphi$.

$\diamond$