Algoritmos de *model checking* CTL

Aula 15

Algoritmos de *model checking*

A semântica dada para o CTL e o LTL, permite decidir se

$$s_0 \models \varphi$$

Mas é conveniente ter algoritmos de decisão.

Normalmente, dado $\mathcal{M}$ e φ os algoritmos determinam *o conjunto de estados que satisfazem* φ

Depois basta ver se s_0 está nesse conjunto.

Model checking para o CTL

Vamos apenas considerar um conjunto completo de conectivas,

$$\{\mathsf{false}, \neg, \wedge, \mathrm{AF}, \mathrm{EU}, \mathrm{EX}\}$$

$$\begin{array}{rcl}
\mathrm{EG}\varphi & \equiv & \neg\mathrm{AF}\neg\varphi \\
\mathrm{EF}\varphi & \equiv & \mathrm{E}[\neg\mathsf{false}\mathrm{U}\varphi] \\
\mathrm{AG}\varphi & \equiv & \neg\mathrm{EF}\neg\varphi \equiv \neg\mathrm{E}[\neg\mathsf{false}\mathrm{U}\neg\varphi] \\
\mathrm{AX}\varphi & \equiv & \neg\mathrm{EX}\neg\varphi \\
\mathrm{A}[\varphi\mathrm{U}\psi] & \equiv & \neg(\mathrm{E}[\neg\psi\mathrm{U}(\neg\varphi \wedge \neg\psi)]) \wedge \mathrm{AF}\psi
\end{array}$$

sendo que no LTL

$$\varphi\mathrm{U}\psi \quad \equiv \quad \neg(\neg\psi\mathrm{U}(\neg\varphi \wedge \neg\psi)) \wedge \mathrm{F}\psi$$

Model checking para o CTL – algoritmo de etiquetagem

[Clarke and Emerson 1981, 2007 Turing Award]

Dados: um sistema de transições $T = (S, Act, \longrightarrow, AP, I, L)$ e uma fórmula CTL φ

Saída: $Sat(\varphi)$, i.e. conjunto de estados de T que satisfazem φ,

Temos que $T \models \varphi$ sse $I \subseteq Sat(\varphi)$.

O algoritmo etiqueta cada estado de T com as subfórmulas de φ que são satisfeitas nesse estado, começando das mais pequenas até φ. O algoritmo procede por indução na estrutura de φ. Supondo que ψ é uma subfórmula de φ cujas subfórmulas imediatas já etiquetam todos os estados em que são satisfeitas, determina-se por análise de casos os estados que têm ψ na etiqueta.

Etiquetagem

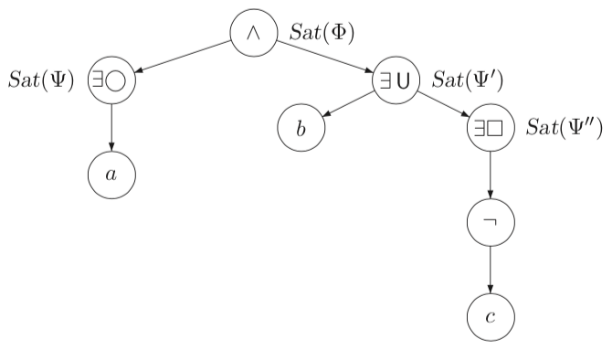
for $i \leq |\varphi|$ **do**
  for $\psi \in Sub(\varphi)$ com $i = |\psi|$ **do**
    Calcular $Sat(\psi)$
if $I \subseteq Sat(\varphi)$ **then return** true
return false

onde $Sub(\varphi)$ é o conjunto de todas as subfórmulas de φ.

Bastará percorrer a árvore sintáctica de φ começando com as folhas. Para cada ψ podemos associar uma nova variável proposicional a_ψ e para cada estado s tal que $s \in Sat(\psi)$ podemos supor que acrescentamos a_ψ a $L(s)$.

Exemplo

$$\Phi = \underbrace{\exists\bigcirc a}_{\Psi} \wedge \underbrace{\exists(b\,\mathsf{U}\,\underbrace{\exists\Box\neg c}_{\Psi''})}_{\Psi'} \quad .$$



Por exemplo Ψ'' pode ser substituido por a_1, tal que $a_1 \in L(s)$ se só se $s \in Sat(\Psi'')$ e depois seria necessário calcular $Sat(\mathrm{E}[b\mathrm{U}a_1])$.

Model checking para o CTL – algoritmo de etiquetagem

Se ψ é

false Não etiqueta nenhum estado

p etiquetar os estados s tal que $p \in L(s)$

$\psi_1 \wedge \psi_2$ etiquetar os estados s que estejam etiquetados com ψ_1 e ψ_2

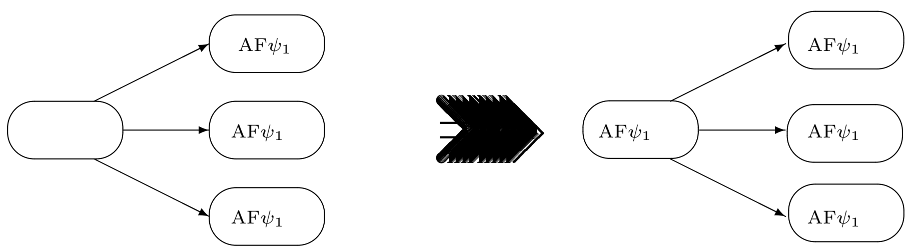
$\neg\psi_1$ etiquetar os estados s que não estejam etiquetados com ψ_1

A etiquetagem é baseada nas seguintes equivalências:

$$
\begin{aligned}
\mathrm{AG}\varphi &\equiv \varphi \ \wedge \ \mathrm{AXAG}\varphi \\
\mathrm{EG}\varphi &\equiv \varphi \ \wedge \ \mathrm{EXEG}\varphi \\
\mathrm{AF}\varphi &\equiv \varphi \ \vee \ \mathrm{AXAF}\varphi \\
\mathrm{EF}\varphi &\equiv \varphi \ \vee \ \mathrm{EXEF}\varphi \\
\mathrm{A}[\varphi \mathrm{U} \psi] &\equiv \psi \vee (\varphi \wedge \mathrm{AXA}[\varphi \mathrm{U} \psi]) \\
\mathrm{E}[\varphi \mathrm{U} \psi] &\equiv \psi \vee (\varphi \wedge \mathrm{EXE}[\varphi \mathrm{U} \psi])
\end{aligned}
$$

Model checking para o CTL – algoritmo de etiquetagem

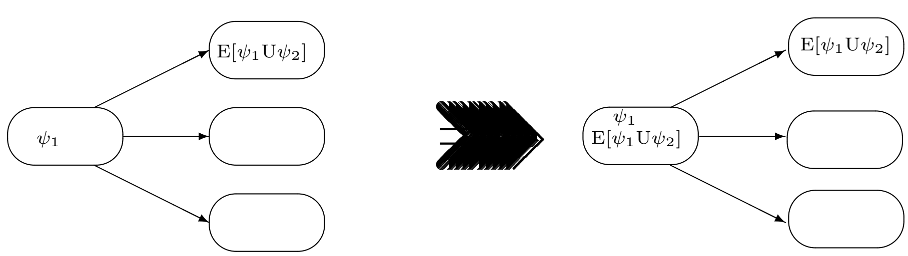
AFψ_1
- Se existe s que está etiquetado com ψ_1 então etiquetá-lo com AFψ_1.
- Repetir: Se todos os sucessores de um estado estiverem etiquetados com AFψ_1, etiquetar esse estado. Até não haver alteração.



$$\mathrm{AF}\varphi \equiv \varphi \ \vee \ \mathrm{AXAF}\varphi$$

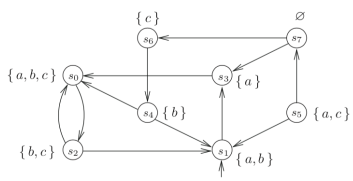
Model checking para o CTL – algoritmo de etiquetagem

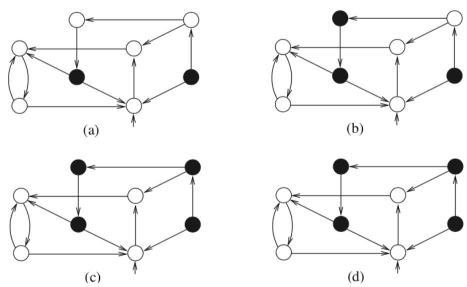
E$(\psi_1$**U**$\psi_2)$
- Se existe s que está etiquetado com ψ_2 então etiquetá-lo com E$(\psi_1 \mathrm{U} \psi_2)$.
- Repetir: Etiquetar um estado com E$(\psi_1 \mathrm{U} \psi_2)$ se está etiquetado com ψ_1 e se pelo menos um dos seus sucessores está etiquetado com E$(\psi_1 \mathrm{U} \psi_2)$. Até não haver alteração.



$$\mathrm{E}[\psi_1 \mathrm{U} \psi_2] \equiv \psi_2 \ \vee \ (\psi_1 \ \wedge \ \mathrm{EXE}[\psi_1 \mathrm{U} \psi_2])$$

$$\mathrm{E}(\mathsf{true}\mathrm{U}(a \leftrightarrow c) \wedge \neg(a \leftrightarrow b))$$





Model checking para o CTL – algoritmo de etiquetagem

EXψ_1 etiquetar com EXψ_1 um estado se pelo menos um dos seus sucessores estiver etiquetado com ψ_1.

Model checking para o CTL – algoritmo de etiquetagem

Exercício 15.1. *Considere o modelo* $T = (\{q_0, q_1, q_2, q_3\}, \{q_0 \to q_1, q_0 \to q_3, q_1 \to q_1, q_1 \to q_2, q_2 \to q_0, q_2 \to q_3, q_3 \to q_0\}, L(q_0) = \{p, q\}, L(q_1) = \{r\}, L(q_2) = \{p, t\}, L(q_3) = \{q, r\})$.

Determine os estados s tal que $q_0 \models \varphi$ *para*

a) $\varphi = AFq$,

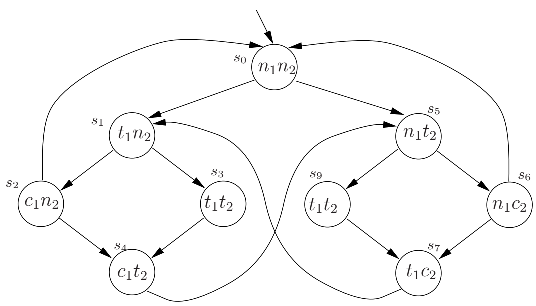
b) $\varphi = EXEXr$ *e*

c) $\varphi = AG(EF(p \vee r))$.

◇

Model checking para o CTL – algoritmo de etiquetagem

Exemplo2: $s_0 \models E(\neg c_2 U c_1)$?



Model checking para o CTL – algoritmo de etiquetagem

O algoritmo tem complexidade $O(f \cdot V \cdot (V+E))$, onde f é o número de conectivas da fórmula, V o número de estados e E o número de transições.

Pode ser mais eficiente se se considerarem mais conectivas, p.e. EG.

EGψ

- Etiquetar todos os estados com EGψ.
- Se um estado s não tiver a etiqueta ψ, apagar a etiqueta EGψ.
- Repetir: apagar a etiqueta EGψ se nenhum dos sucessores do estado tiver a etiqueta EGψ. Até não haver mudança.
- EG$\psi \equiv \psi \wedge$ EXEGψ

Equivale ter $\neg$AF$\neg\psi$

Pseudo-código para o algoritmo dado φ e T

function SAT (φ)
begin
  case
    φ is true : **return** S
    φ is false : **return** $\emptyset$
    φ is atomic: **return** $\{s \in S \mid \varphi \in L(s)\}$
    φ is $\neg\varphi_1$: **return** $S -$ SAT (φ_1)
    φ is $\varphi_1 \wedge \varphi_2$: **return** SAT (φ_1) $\cap$ SAT (φ_2)
    φ is $\varphi_1 \vee \varphi_2$: **return** SAT (φ_1) $\cup$ SAT (φ_2)
    φ is $\varphi_1 \rightarrow \varphi_2$: **return** SAT ($\neg\varphi \vee \varphi_2$)
    φ is AXφ_1 : **return** SAT ($\neg$EX$\neg\varphi_1$)
    φ is EXφ_1 : **return** $\mathtt{SAT}_{\mathtt{EX}}(\varphi_1)$
    φ is A[φ_1Uφ_2] : **return** SAT($\neg$(E[$\neg\varphi_2$U($\neg\varphi_1 \wedge \neg\varphi_2$)] $\vee$ EG$\neg\varphi_2$))
    φ is E[φ_1Uφ_2] : **return** $\mathtt{SAT}_{\mathtt{EU}}(\varphi_1, \varphi_2)$

φ is EFφ_1 : **return** SAT (E(trueUφ_1))
φ is EGφ_1 : **return** SAT($\neg$AF$\neg\varphi_1$)
φ is AFφ_1 : **return** $\mathtt{SAT_{AF}}$ (φ_1)
φ is AGφ_1 : **return** SAT ($\neg$EF$\neg\varphi_1$)
end case
end function

Pseudo-código para o algoritmo

Dado um conjunto de estados Y, a função $\mathtt{pre}_\exists(Y)$ ($\mathtt{pre}_\forall(Y)$) determina os estados de que se pode (ou só se pode) chegar a Y:

$$\begin{array}{rcl} \mathtt{pre}_\exists(Y) & = & \{s \in S \mid \exists s'\, s \longrightarrow s' \wedge s' \in Y\} \\ & = & \{s \in S \mid Post(s) \cap Y \neq \emptyset\} \\ \mathtt{pre}_\forall(Y) & = & \{s \in S \mid \forall s'(s \longrightarrow s') \Rightarrow s' \in Y)\} \\ & = & \{s \in S \mid Post(s) \subseteq Y\} \end{array}$$

function $\mathtt{SAT_{EX}}$ (φ)
local var X, Y
begin
$X :=$ SAT (φ);
$Y := \mathtt{pre}_\exists(X)$;
return Y
end

Pseudo-código para o algoritmo

function $\mathtt{SAT_{AF}}$ (φ)
local var X, Y
begin
$X := S$;
$Y :=$ SAT (φ);
repeat until $X = Y$
begin
$X := Y$;
$Y := Y \cup \mathtt{pre}_\forall(Y)$
end
return Y
end

Pseudo-código para o algoritmo

```
function SAT_EU (φ, ψ)
/* determina os estados que satisfazem E[φ U ψ] */
local var W, X, Y
begin
    W := SAT (φ);
    X := S;
    Y := SAT (ψ);
    repeat until X = Y
    begin
        X := Y;
        Y := Y ∪ (W ∩ pre_∃(Y))
    end
    return Y
end
```

Pseudo-código para o algoritmo

```
function SAT_EG (φ)
/* determines the set of states satisfying EG φ */
local var X, Y
begin
    Y := SAT (φ);
    X := ∅;
    repeat until X = Y
    begin
        X := Y;
        Y := Y ∩ pre_∃(Y)
    end
    return Y
end
```

Funções monótonas

Pontos Fixos

Seja S um conjunto de estados e $F : \mathcal{P}(S) \to \mathcal{P}(S)$

1. F é monótona sse $X \subseteq Y$ então $F(X) \subseteq F(Y)$
2. $X \subseteq S$ é *ponto fixo* sse $F(X) = X$

Exercício 15.2. *Sendo* $S = \{s_0, s_1\}$ *e* $F(Y) = Y \cup \{s_0\}$ *mostrar que* F *é monótona e determinar os pontos fixos.*

Correção e Terminação do Algoritmo

Para provar que o algoritmo de etiquetagem é correcto e termina:

- Provar que cada função recursiva é monótona em $(\mathcal{P}(S), \subseteq)$.
- O ponto fixo máximo (ou mínimo) é atingido e é a semântica pretendida.

Teorema 15.1 (Knaster-Tarski)**.** *Se S é um conjunto com $n+1$ elementos e $F : \mathcal{P}(S) \to \mathcal{P}(S)$ é monótona, então $F^{n+1}(\emptyset)$ é o ponto fixo mínimo de F e $F^{n+1}(S)$ é o ponto fixo máximo de F.*

Teorema de Knaster-Tarski

Demonstração. Como $\emptyset \subseteq F(\emptyset)$ então também $F(\emptyset) \subseteq F(F(\emptyset))$ e por indução podemos mostrar que

$$\emptyset \subseteq F(\emptyset) \subseteq F(F(\emptyset)) \subseteq \cdots \subseteq F^i(\emptyset))$$

para qualquer $i \geq 1$. Para $i = n+1$ vamos mostrar que existe $0 \leq k \leq n+1$ tal que $F^k(\emptyset)$ é um ponto fixo de F. Por contradição suponhamos que não. Então, $F(\emptyset)$ tem pelo menos 1 elemento (senão seria igual a $\emptyset$), $F^2(\emptyset)$ 2 elementos, etc, e $F^{n+2}(\emptyset)$ teria $n+2$ elementos o que é absurdo. Logo concluímos que em particular $F^{n+1}(\emptyset)$ é um ponto fixo. Agora suponhamos que $F(X) = X$ temos de mostrar que $F^{n+1}(\emptyset) \subset X$. Como $\emptyset \subseteq X$ temos $F(\emptyset) \subseteq F(X) = X$. Mas repetindo temos $F^2(\emptyset) \subseteq F(X) = X$, $\ldots$, $F^{n+1}(\emptyset) \subseteq X$. □

Correção e Terminação do Algoritmo

Para demonstrar a correção de `SAT` temos:

1. A semântica de EG, AF e EU pode ser expressa em termos de pontos fixos máximos ou mínimos de funções em $P(S)$,
2. Estes pontos fixos são calculáveis
3. $\mathtt{SAT}_{\mathtt{EG}}$, $\mathtt{SAT}_{\mathtt{AF}}$ e $\mathtt{SAT}_{\mathtt{EU}}$ codificam o cálculo destes pontos fixos (o primeiro máximo, e os outros mínimos).

Correção e Terminação do Algoritmo

Seja $Sat(\varphi) \subseteq S$ o conjunto de estados que satisfaz φ.

Os caso base $(\mathsf{false}, \mathsf{true}, p)$ são calculados directamente.

Para a negação e disjunção é calculado o conjunto usando o complemento e a união respectivamente:

$Sat(\neg\varphi) = S \setminus Sat(\varphi)$

$Sat(\varphi_1 \vee \varphi_2) = Sat(\varphi_1) \cup Sat(\varphi_2)$.

E para EX vem:

$Sat(\mathrm{EX}\varphi) = pre_\exists(Sat(\varphi))$

Correção e Terminação do Algoritmo

Para as restantes conectivas temporais temos:

- $Sat(\mathrm{EG}\varphi) = Sat(\varphi) \cap pre_\exists(Sat(\mathrm{EG}\varphi))$
- $Sat(\mathrm{E}[\varphi \mathrm{U} \psi]) = Sat(\psi) \cup (Sat(\varphi) \cap pre_\exists Sat(\mathrm{E}[\varphi \mathrm{U} \psi]))$
- $Sat(\mathrm{AF}\varphi) = Sat(\varphi) \cup pre_\forall(Sat(\mathrm{AF}\varphi))$

Para $\texttt{SAT}_{\mathrm{EG}}$, $\texttt{SAT}_{\mathrm{EU}}$, e $\texttt{SAT}_{\mathrm{AF}}$ basta associar a respectiva função, provar a sua monotonia e mostrar que o ponto fixo máximo (mínimos) é a semântica pretendida.

Correção de EGφ

Teorema 15.2. *Seja $F(X) = Sat(\varphi) \cap pre_\exists(X)$ e S com $n+1$ elementos. Então $F^{n+1}(S) = Sat(EG\varphi)$.*

1. F é monótona: se $X \subseteq Y$ então $pre_\exists(X) \subseteq pre_\exists(Y)$
2. $Sat(\mathrm{EG}\varphi)$ é um ponto fixo de F (por construção)
3. $Sat(\mathrm{EG}\varphi)$ ponto fixo máximo de F
   - Provar que se $F(X) = X$ então $X \subseteq Sat(\mathrm{EG}\varphi)$
   - Se $s_0 \in X = F(X) = Sat(\varphi) \cap pre_\exists(X)$ então existe $s_1 \in Post(s_0)$ e $s_1 \in X$
   - mas então $s_1 \in X = F(X) = Sat(\varphi) \cap pre_\exists(X)$ e existe $s_2 \in Post(s_1)$ e $s_2 \in X$
   - iterando temos $\pi = s_0 \longrightarrow s_1 \longrightarrow s_2 \longrightarrow \cdots$ tal que $s_i \in Sat(\varphi)$. Logo $s_0 \in Sat(\mathrm{EG}\varphi)$
4. $F^{n+1}(S) = Sat(\mathrm{EG}\varphi)$ pelo teorema de Knaster-Tarski.

Correção de EUφ

$Sat(\mathrm{E}[\varphi \mathrm{U} \psi]) = Sat(\psi) \cup (Sat(\varphi) \cap pre_\exists Sat(\mathrm{E}[\varphi \mathrm{U} \psi]))$

Teorema 15.3. *Seja $G(X) = Sat(\psi) \cup (Sat(\varphi) \cap pre_\exists(X))$ e S com $n+1$ elementos. Então $G^{n+1}(\emptyset) = Sat(E[\varphi U \psi])$.*

1. G é monótona: se $X \subseteq Y$ então $pre_\exists(X) \subseteq pre_\exists(Y)$
2. $Sat(\mathrm{E}[\varphi \mathrm{U} \psi])$ ponto fixo mínimo de G
    - se $s_0 \in Sat(\mathrm{E}[\varphi \mathrm{U} \psi])$ então existe $\pi \in Paths(s_0)$, existe $i \geq 0$ $\pi[i] \models \psi$, e para $0 \leq j < i$ $\pi[j] \models \varphi$.
    - Temos que $G(\emptyset) = Sat(\psi)$. Logo $G(\emptyset) \subseteq Sat(\mathrm{E}[\varphi \mathrm{U} \psi])$ (considerando os s_0 com $i = 0$).
    - $G^2(\emptyset) = Sat(\psi) \cup (Sat(\varphi) \cap pre_\exists(Sat(\psi))) \subseteq Sat(\mathrm{E}[\varphi \mathrm{U} \psi])$ considerando $i \leq 1$.
    - por indução em k, temos $G^{k+1}(\emptyset) \subseteq Sat(\mathrm{E}[\varphi \mathrm{U} \psi])$, considerando os s_0 com $i \leq k$.
    - Logo $Sat(\mathrm{E}[\varphi \mathrm{U} \psi]) = \cup_{k \geq 0} G^k(\emptyset)$, mas como $G^{n+1}(\emptyset)$ é um ponto fixo temos o resultado.

Exercício 15.3. *Determina a função associada a $AF\varphi$ e mostra a sua correção.* $\diamond$

Exercício 15.4. *Concluí que* $\mathtt{SAT}_{EG}$, $\mathtt{SAT}_{EU}$, *e* $\mathtt{SAT}_{AF}$ *estão correctas.* $\diamond$