Model Checking. Noţiuni de bază

19 octombrie 2004

- Sisteme cu stări finite
- Logici temporale: LTL, CTL*, CTL
- Model checking cu reprezentarea explicită a stărilor

Ce fel de sisteme putem verifica ?

– sisteme al căror comportament poate fi descris în formă matematică
– ne interesează interacţiunea sistemului cu mediul în care e plasat
– *starea* sistemului = totalitatea mărimilor care determină comportamentul său ulterior în timp
– definirea stării depinde de nivelul de *abstracţie* în reprezentare
Exemplu: un procesor (nivelul ISA, de organizare internă (ex. pipeline), nivelele RTL, logic, al tranzistorilor . . .)
– sisteme *discrete*, *continue* sau *hibride*
– sisteme *finite* (necesar discrete) sau *infinite* (sisteme continue, programe recursive, sau cu structuri de date dinamice)

Modelarea sistemelor cu stări finite

În practică: descriere cu mulţime de variabile $V = \{v_1, v_2, \cdots, v_n\}$
– stare: o *atribuire* $s : V \to D$ de valori dintr-un *domeniu* D
pentru fiecare variabilă $v \in V$.
– Unei stări (atribuiri) i se asociază o *formulă* adevărată doar pentru acea stare (atribuire:

$\langle v_1 \leftarrow 7, v_2 \leftarrow 4, v_3 \leftarrow 2\rangle$ $\qquad$ $(v_1 = 7) \land (v_2 = 4) \land (v_3 = 2)$

– O formulă $\leftrightarrow$ mulţimea *tuturor* atribuirilor care o fac adevărată.
(pot fi şi mai multe stări, ex. $v_1 \leq 5 \land v_2 > 3$
$\Rightarrow$ *mulţimi* de stări: reprezentate prin formule logice

– tranziţie $s \to s'$: o formulă peste $V \cup V'$
V' = copie a lui V (variabilele stării următoare)
ex. $(semaphore = red) \land (semaphore' = green)$
– mulţimea tuturor tranziţiilor: *relaţia de tranziţie*: formulă $\mathcal{R}(V, V')$

Modelarea cu structuri Kripke

Structură Kripke: automat cu stări finite, etichetat:

$$M = (S, S_0, R, L)$$

– S: mulţime finită de stări
– $S_0 \subseteq S$: mulţimea stărilor iniţiale
– $R \subseteq S \times S$: relaţie de tranziţie *totală*: $\forall s \in S\ \exists s' \in S\ .\ (s, s') \in R$
(din orice stare există cel puţin o tranziţie)
– $L : S \to 2^{AP}$: funcţie de etichetare a stărilor

AP = mulţime de propoziţii atomice (observaţii care apar în formule/proprietăţi/specificaţii). Exemple:
– o stare are atributul *stabil* sau nu
– definim propoziţia $bad ::= red\_recvd > 1$

Traiectorie pornind din starea s_0: secvenţă *infinită* de stări

$$\pi = s_0 s_1 s_2 \ldots, \text{ cu } R(s_i, s_{i+1}) \text{ pentru orice } i \geq 0$$

Modelare: circuite și programe

- Circuite *secvențiale*: o variabilă pentru fiecare element de stare (registru), și pentru intrările primare.
  se presupune: propagare combinațională instantanee
- Circuite *asincrone*: o variabilă pentru fiecare semnal
  (în modele mai sofisticate: timp fizic explicit)
- Programe: variabile declarate + contorul de program

Sincronie și asincronie

Tipuri de compoziţie:
(obţinerea comportamentului sistemului din cel al componentelor)

- sincronă: conjuncţie (tranziţii simultane)

$$R(V, V') = R_1(V_1, V_1') \wedge R_2(V_2, V_2') \qquad V = V_1 \cup V_2$$

- asincronă: disjuncţie (tranziţii individuale)

$$R(V, V') = R_1(V_1, V_1') \wedge \mathit{Eq}(V \setminus V_1) \vee R_2(V_2, V_2') \wedge \mathit{Eq}(V \setminus V_2)$$
$$\mathit{Eq}(U) = \bigwedge_{v \in U}(v = v')$$

– alternanţă arbitrară între tranziţiile componentelor
– o tranziţie modifică doar variabilele *unei* componente
– tranziţii simultane se consideră imposibile

Modelarea programelor: de regulă compoziţie asincronă (nu există sincronizare fizică între instrucţiunile a două programe concurente)

Modelarea comportamentului

Sisteme reactive

– interacţionează cu mediul (*reacţie* la un anumit *stimul*)

– adesea au execuţie infinită

$\Rightarrow$ o *computaţie* = secvenţă infinită de stări

$\Rightarrow$ nu e suficientă reprezentarea comportamentului de intrare-ieşire

– Exemple simple:

nu se atinge o anumită stare (de eroare)

sistemul nu se blochează (deadlock)

Mai general: proprietăţi descrise în logică temporală

– logică *modală* (noţiune de adevăr cu modalităţi temporale)

– utilizată din antichitate în raţionamente despre timp

– [Pnueli'77] - aplicare la programe concurente

Logica temporală LTL

Linear Temporal Logic [Pnueli 1977]
– ne interesează descrierea evenimentelor de-a lungul unei traiectorii
$\Rightarrow$ structură *liniară*
în viitor apare un eveniment; o proprietate e invariantă de la un moment dat; un eveniment apare după alt eveniment

Operatori temporali (modalităţi de adevăr pe o traiectorie):

- **X** (*next*): în următoarea stare $\bigcirc$
- **F** (*future*): cândva în viitor $\Diamond$
- **G** (*globally*): în orice stare viitoare $\Box$
- **U** (*until*): $prop_1$ obligatorie până când apare $prop_2$

uneori se mai defineşte şi următorul operator:

- **R** (*release*): apariţia $prop_1$ elimină obligativitatea $prop_2$

Formulele logicii LTL

– dorim ca o proprietate să fie adevărată pentru *toate* traiectoriile
$\Rightarrow$ folosim *cuantificatorul universal* **A**
– formulele sunt de tipul **A** f, unde f este o formulă de traiectorie
– sintaxa formulelor de traiectorie:

$$\begin{aligned} f \ ::= \ & p \qquad \text{(unde } p \in AP\text{)} \\ & \mid \neg f_1 \mid f_1 \vee f_2 \mid f_1 \wedge f_2 \\ & \mid \mathbf{X}\, f_1 \mid \mathbf{F}\, f_1 \mid \mathbf{G}\, f_1 \mid f_1 \,\mathbf{U}\, f_2 \mid f_1 \,\mathbf{R}\, f_2 \end{aligned}$$

Semantica logicii LTL

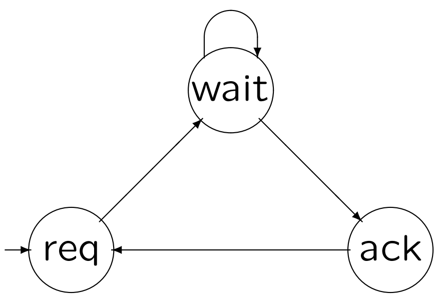
Notăm: $M, s \models f$: în modelul M, starea s satisface f

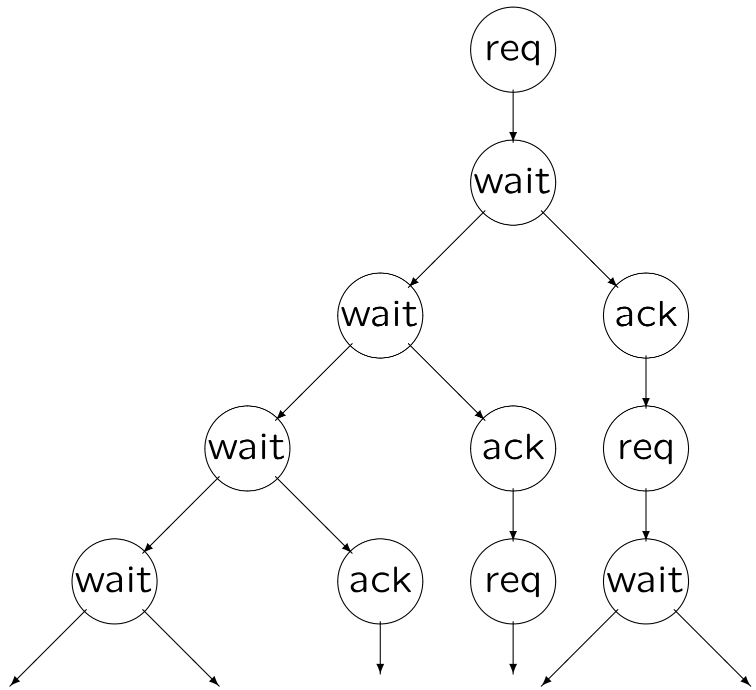
π^i = sufixul traiectoriei $\pi = s_0 s_1 s_2 \ldots$ începând cu s_i

$M, s \models p$	$\Leftrightarrow$	$p \in L(s)$
$M, s \models \mathbf{A}\, f$	$\Leftrightarrow$	$\forall$ traiectorie π din s, $M, \pi \models f$
$M, \pi \models p$	$\Leftrightarrow$	$M, s \models p$, unde $p \in AP$ și s e prima stare din π
$M, \pi \models \neg f$	$\Leftrightarrow$	$M, \pi \not\models f$
$M, \pi \models f_1 \vee f_2$	$\Leftrightarrow$	$M, \pi \models f_1 \vee M, \pi \models f_2$
$M, \pi \models f_1 \wedge f_2$	$\Leftrightarrow$	$M, \pi \models f_1 \wedge M, \pi \models f_2$
$M, \pi \models \mathbf{X}\, f$	$\Leftrightarrow$	$M, \pi^1 \models f$
$M, \pi \models \mathbf{F}\, f$	$\Leftrightarrow$	$\exists k \geq 0 \,.\, M, \pi^k \models f$
$M, \pi \models \mathbf{G}\, f$	$\Leftrightarrow$	$\forall k \geq 0 \,.\, M, \pi^k \models f$
$M, \pi \models f_1 \,\mathbf{U}\, f_2$	$\Leftrightarrow$	$\exists k \geq 0 \,.\, M, \pi^k \models f_2 \wedge \forall j < k \,.\, M, \pi^j \models f_1$
$M, \pi \models f_1 \,\mathbf{R}\, f_2$	$\Leftrightarrow$	$\forall k \geq 0 \,.\, (\forall j < k \,.\, M, \pi^j \not\models f_1) \rightarrow M, \pi^k \models f_2$

Logica temporală CTL*

Uneori: modelul liniar insuficient (ex. *e posibil* să se atingă o stare)
⇒ alt model: arbori de computaţie (*computation trees*):
desfăşurare infinită a grafului de stări-tranziţii
pornind de la o stare iniţială





Structura formulelor CTL*

în plus: cuantificatorul existenţial **E** (există o traiectorie) $\exists$

Două tipuri de formule:

– formule de stare (*state formula*), evaluate într-o stare

f ::= p (unde $p \in AP$)
$\mid \neg f_1 \mid f_1 \vee f_2 \mid f_1 \wedge f_2$
$\mid$ **E** g $\mid$ **A** g (unde $g =$ formulă de traiectorie)

– formule de traiectorie (*path formula*), evaluate pe o traiectorie

g ::= f (unde $f =$ formulă de stare)
$\mid \neg g_1 \mid g_1 \vee g_2 \mid g_1 \wedge g_2$
$\mid$ **X** g_1 $\mid$ **F** g_1 $\mid$ **G** g_1 $\mid$ g_1 **U** g_2 $\mid$ g_1 **R** g_2

Semantica: la fel ca LTL, în plus:

$M, s \models$ **E** $g \Leftrightarrow \exists$ traiectorie π din s cu $M, \pi \models g$

Relaţii între operatori

- $f \wedge g \equiv \neg(\neg f \vee \neg g)$
- $f\,\mathbf{R}\,g \equiv \neg(\neg f\,\mathbf{U}\,\neg g)$
- $\mathbf{F}\,f \equiv \mathit{true}\,\mathbf{U}\,f$
- $\mathbf{G}\,f \equiv \neg\mathbf{F}\,\neg f$
- $\mathbf{A}\,f \equiv \neg\mathbf{E}\,\neg f$

$\Rightarrow$ Operatorii $\neg$, $\vee$, **X** , **U** şi **E** sunt suficienţi pentru a exprima orice formulă în CTL*.

Sublogica: CTL

CTL (Computation Tree Logic) [Clarke, Emerson 1981]

– suficientă în multe cazuri, dar mai simplă $\Rightarrow$ algoritmi mai eficienţi

– structură *ramificată* (*branching*), ca și CTL*

– cuantificare asupra traiectoriilor posibile dintr-o stare

– operatorii **X** , **F** , **G** , **U** , **R** precedaţi imediat de **A** sau **E**

– sintaxa formulelor de traiectorie:

$$g ::= \mathbf{X}\, f \mid \mathbf{F}\, f \mid \mathbf{G}\, f \mid f_1\, \mathbf{U}\, f_2 \mid f_1\, \mathbf{R}\, f_2$$

CTL: Operatori de bază și derivați

10 operatori de bază, exprimabili folosind **EX**, **EG** și **EU**:

- $\mathbf{AX}\, f \equiv \neg \mathbf{EX}\, \neg f$
- $\mathbf{EF}\, f \equiv \mathbf{E}\, [\mathit{true}\, \mathbf{U}\, f]$
- $\mathbf{AF}\, f \equiv \neg \mathbf{EG}\, \neg f$
- $\mathbf{AG}\, f \equiv \neg \mathbf{EF}\, \neg f$
- $\mathbf{A}\, [f\, \mathbf{U}\, g] \equiv \neg \mathbf{EG}\, \neg g \wedge \neg \mathbf{E}\, [\neg g\, \mathbf{U}\, (\neg f \wedge \neg g)]$
- $\mathbf{E}\, [f\, \mathbf{R}\, g] \equiv \neg \mathbf{A}\, [\neg f\, \mathbf{U}\, \neg g]$
- $\mathbf{A}\, [f\, \mathbf{R}\, g] \equiv \neg \mathbf{E}\, [\neg f\, \mathbf{U}\, \neg g]$

Exemple de formule în CTL

- $\mathbf{EF}\,\textit{finish}$
  Este posibil să se ajungă într-o stare în care $\textit{finish} = \textit{true}$.
- $\mathbf{AG}\,(\textit{send} \rightarrow \mathbf{AF}\,\textit{ack})$
  Orice *send* este urmat în cele din urmă de un *ack*.
- $\mathbf{AF}\,\mathbf{AG}\,\textit{stable}$
  În orice execuţie, de la un moment dat, *stable* este invariant.
- $\mathbf{AG}\,(\textit{req} \rightarrow \mathbf{A}\,[\textit{reg}\,\mathbf{U}\,\textit{grant}])$
  Întotdeauna, un *req* ramîne activ până se obţine un *grant*.
- $\mathbf{AG}\,\mathbf{AF}\,\textit{ready}$
  Pe orice traiectorie, *ready* e satisfăcut de un număr infinit de ori.
- $\mathbf{AG}\,\mathbf{EF}\,\textit{restart}$
  Din orice stare e posibil să se ajungă în starea *restart*.

Relaţii între diferitele logici

CTL şi LTL sunt incomparabile:
– **A F G** p e în LTL, nu are echivalent CTL
– **AG EF** p e în CTL, nu are echivalent LTL
– disjuncţia lor e în CTL*, dar nu în CTL, nici în LTL

Unele tehnici (compoziţionalitate, abstracţie) necesită restricţii:
în mod tipic, e permis doar cuantificatorul universal **A** .
– ACTL (inclusă în CTL, incomparabilă cu LTL)
– ACTL* (inclusă în CTL*, mai expresivă decât LTL)

Noţiunea de fairness

În practică: presupuneri rezonabile de tipul:
– un arbitru nu ignoră la infinit una din cereri
– un mesaj retransmis continuu îsi atinge destinaţia
= proprietăţi exprimabile în CTL*, dar nu şi în CTL.
⇒ se defineşte nouă semantică pentru CTL cu *fairness*

O restricţie de *fairness* e o formulă în logică temporală.

O traiectorie e *echitabilă* dacă fiecare restricţie e adevărată infinit de multe ori de-a lungul traiectoriei.

In particular: restricţie exprimată ca mulţime de stări:
o traiectorie echitabilă trece infinit de multe ori prin acea mulţime.

CTL cu fairness

Augmentăm structura Kripke, $M = (S, S_0, R, L, F)$, cu $F \subseteq 2^S$
(F = mulţime de submulţimi de stări, $\{P_1, \cdots, P_n\}, P_i \subseteq S$)

$$\text{inf}(\pi) \stackrel{\text{def}}{=} \{s \mid s = s_i \text{ pt. infinit de mulţi } i\}$$

(mulţimea stărilor care apar infinit de multe ori pe π)

π este echitabilă $\Leftrightarrow \forall P \in F \,.\, \text{inf}(\pi) \cap P \neq \emptyset$.
(π trece infinit de multe ori prin orice mulţime din F)

Notăm $\models_F$ relaţia de satisfacere cu fairness.
Clauze modificate în semantica CTL:

$M, s \models_F p$	$\Leftrightarrow$	există o traiectorie echitabilă plecând din s și $p \in L(s)$
$M, s \models_F \mathbf{E}\, g$	$\Leftrightarrow$	$\exists$ traiectorie echitabilă π din s cu $M, \pi \models_F g$
$M, s \models_F \mathbf{A}\, g$	$\Leftrightarrow$	$\forall$ traiectoriile echitabile π din s, $M, \pi \models_F g$

Model checking. Enunțul problemei

Fiind date o structură Kripke $M = (S, S_0, R, L)$ și o formulă f în logică temporală, să se găsească stările din S care satisfac f:

$$\{s \in S \mid M, s \models f\}$$

Specificația e satisfăcută dacă toate stările inițiale satisfac f:

$$\forall s_0 \in S_0 \,.\, M, s_0 \models f$$

Istoric

– independent, Clarke & Emerson, resp. Quielle & Sifakis (1981).
– inițial: $10^4 - 10^5$ stări. Actualmente, simbolic: cca 10^{100} stări

Model checking pentru CTL

– Descompunere după structura formulei f. Pentru fiecare $s \in S$, calculează $l(s)$ = mulțimea subformulelor lui f valabile în s.
– Inițial $l(s) = L(s)$. Trivial pentru conectorii logici $\neg, \vee, \wedge$
– **EX** f: se etichetează orice stare cu un succesor etichetat cu f.
– Ceilalți operatori de bază: **EU** și **EG**

Model checking pentru CTL. Operatorul EU

$\mathbf{E}\,[f_1\,\mathbf{U}\,f_2]$: traversare înapoi pornind de la f_2, cât timp se satisface f_1.

procedure *CheckEU*(f_1, f_2)
  $T := \{s \mid f_2 \in l(s)\}$
  forall $s \in T$ **do** $l(s) := l(s) \cup \{\mathbf{E}\,[f_1\,\mathbf{U}\,f_2]\}$;
  while $T \neq \emptyset$ **do**
    choose $s \in T$;
    $T := T \setminus \{s\}$;
    forall $s_1 \,.\, R(s_1, s)$ **do**
      if $\mathbf{E}\,[f_1\,\mathbf{U}\,f_2] \notin l(s_1) \wedge f_1 \in l(s_1)$ **then**
        $l(s_1) := l(s_1) \cup \{\mathbf{E}\,[f_1\,\mathbf{U}\,f_2]\}$;
        $T := T \cup \{s_1\}$;

Model checking pentru CTL. Operatorul EG

EG f: se consideră doar statele care satisfac f. Se traversează înapoi pornind de la componentele puternic conectate (SCC)

procedure *CheckEG*(f)
  $S' := \{s \mid f \in l(s)\}$;
  $SCC := \{C \mid C$ e o SCC netrivială în $S'\}$;
  $T := \cup_{C \in SCC}\{s \mid s \in C\}$;
  forall $s \in T$ **do** $l(s) := l(s) \cup \{\mathbf{EG}\, f\}$;
  while $T \neq \emptyset$ **do**
    choose $s \in T$;
    $T := T \setminus \{s\}$;
    forall $s_1 \,.\, s_1 \in S' \wedge R(s_1, s)$ **do**
      if EG $f \notin l(s_1)$ **then**
        $l(s_1) := l(s_1) \cup \{\mathbf{EG}\, f\}$;
        $T := T \cup \{s_1\}$;

Model checking cu fairness

Considerăm restricţia de fairness $F = \{P_1, \cdots, P_k\}$, unde $P_i \subseteq S$

Fie *fair* o nouă propoziţie atomică, valabilă în starea s dacă există o traiectorie echitabilă care porneşte din s.
Deci *fair* $\in L(s) \;\Leftrightarrow\; M, s \models_F \mathbf{EG}\,$ *true*.
Pentru ceilalţi operatori, reducem la model checking obişnuit:
$M, s \models_F p \Leftrightarrow M, s \models p \wedge \mathit{fair}$
$M, s \models_F \mathbf{EX}\, f \Leftrightarrow M, s \models \mathbf{EX}\,(f \wedge \mathit{fair})$
$M, s \models_F \mathbf{E}\,[f_1\, \mathbf{U}\, f_2] \Leftrightarrow M, s \models \mathbf{E}\,[f_1\, \mathbf{U}\,(f_2 \wedge \mathit{fair})]$

Pentru $M, s \models_F \mathbf{EG}\, f$ modificăm algoritmul anterior, considerând doar SCC-urile cu $\forall i \,.\, C \cap P_i \neq \emptyset$ (care conţin cel putin o stare din fiecare componentă a restricţiei de fairness).

Complexitatea algoritmilor de model checking

– model checking CTL: $O(|f| \cdot (|S| + |R|))$
(liniar în dimensiunea modelului și a formulei)
– CTL cu fairness F: $O(|f| \cdot (|S| + |R|) \cdot |F|)$
– LTL: PSPACE-complet $|M| \cdot 2^{O(|f|)}$
(algoritm de alt tip, bazat pe o construcție de tablou)
– CTL*: la fel ca LTL $|M| \cdot 2^{O(|f|)}$

CTL: adesea preferat, datorită algoritmului polinomial
dar și în LTL, exponențiala e doar în dimensiunea formulei (mică)