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:
$\langle v_1 \leftarrow 7, v_2 \leftarrow 4, v_3 \leftarrow 2\rangle$ $\qquad (v_1 = 7) \wedge (v_2 = 4) \wedge (v_3 = 2)$
– O formulă $\leftrightarrow$ mulţimea *tuturor* atribuirilor care o fac adevărată.
(pot fi şi mai multe stări, ex. $v_1 \le 5 \wedge v_2 > 3$
$\Rightarrow$ *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. $(semaphore = red) \wedge (semaphore' = green)$
– 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
$\pi = s_0 s_1 s_2 \ldots$, cu $R(s_i, s_{i+1})$ pentru orice $i \ge 0$

Modelare: circuite şi programe

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

Sincronie şi 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
– interacționează cu mediul (*reacție* la un anumit *stimul*)
– adesea au execuție infinită
$\Rightarrow$ o *computație* = secvență infinită de stări
$\Rightarrow$ nu e suficientă reprezentarea comportamentului de intrare-ieșire

– Exemple simple:
nu se atinge o anumită stare (de eroare)
sistemul nu se blochează (deadlock)

Mai general: proprietăți descrise în logică temporală
– logică *modală* (noțiune de adevăr cu modalități temporale)
– utilizată din antichitate în raționamente 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 (modalități 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 definește și următorul operator:
- **R** (*release*): apariția $prop_1$ 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 $\mathbf{A}\, f$, unde f este o formulă de traiectorie
– sintaxa formulelor de traiectorie:

$f ::= p$ (unde $p \in AP$)
$\quad | \neg f_1 \mid f_1 \vee f_2 \mid f_1 \wedge f_2$
$\quad | \mathbf{X}\, f_1 \mid \mathbf{F}\, f_1 \mid \mathbf{G}\, f_1 \mid f_1 \mathbf{U} f_2 \mid f_1 \mathbf{R} f_2$

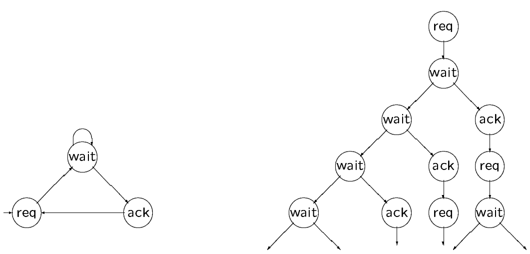
Semantica logicii LTL

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)
$\Rightarrow$ alt model: arbori de computație (*computation trees*):
desfășurare infinită a grafului de stări-tranziții
pornind de la o stare inițială



Structura formulelor CTL*

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

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

$f ::= p$ (unde $p \in AP$)
$\quad | \neg f_1 \mid f_1 \vee f_2 \mid f_1 \wedge f_2$
$\quad | \mathbf{E}\, g \mid \mathbf{A}\, g$ (unde g = formulă de traiectorie)

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

$g ::= f$ (unde f = formulă de stare)
$\quad | \neg g_1 \mid g_1 \vee g_2 \mid g_1 \wedge g_2$
$\quad | \mathbf{X}\, g_1 \mid \mathbf{F}\, g_1 \mid \mathbf{G}\, g_1 \mid g_1 \mathbf{U} g_2 \mid g_1 \mathbf{R} g_2$

Semantica: la fel ca LTL, în plus:
$M, s \models \mathbf{E}\, g \Leftrightarrow \exists$ traiectorie π din s cu $M, \pi \models g$

Relaţii între operatori

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

⇒ Operatorii ¬, ∨, **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 ⇒ algoritmi mai eficienţi
– structură *ramificată* (*branching*), ca şi 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 şi derivaţi

10 operatori de bază, exprimabili folosind **EX**, **EG** şi **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

- **EF** *finish*
  Este posibil să se ajungă într-o stare în care *finish* = *true*.
- **AG** (*send* → **AF** *ack*)
  Orice *send* este urmat în cele din urmă de un *ack*.
- **AF AG** *stable*
  În orice execuţie, de la un moment dat, *stable* este invariant.
- **AG** (*req* → **A** [*req* **U** *grant*])
  Întotdeauna, un *req* ramîne activ până se obţine un *grant*.
- **AG AF** *ready*
  Pe orice traiectorie, *ready* e satisfăcut de un număr infinit de ori.
- **AG EF** *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 îşi 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 = mulțime de submulțimi de stări, $\{P_1, \cdots, P_n\}, P_i \subseteq S$)

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

(mulțimea 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 mulțime din F)

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

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

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

```
procedure CheckEU(f1, f2)
  T := {s | f2 ∈ l(s)}
  forall s ∈ T do l(s) := l(s) ∪ {E [f1 U f2]};
  while T ≠ ∅ do
    choose s ∈ T;
    T := T \ {s};
    forall s1 . R(s1, s) do
      if E [f1 U f2] ∉ l(s1) ∧ f1 ∈ l(s1) then
        l(s1) := l(s1) ∪ {E [f1 U f2]};
        T := T ∪ {s1};
```


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 | f ∈ l(s)};
  SCC := {C | C e o SCC netrivială în S'};
  T := ∪_{C∈SCC} {s | s ∈ C};
  forall s ∈ T do l(s) := l(s) ∪ {EG f};
  while T ≠ ∅ do
    choose s ∈ T;
    T := T \ {s};
    forall s1 . s1 ∈ S' ∧ R(s1, s) do
      if EG f ∉ l(s1) then
        l(s1) := l(s1) ∪ {EG f};
        T := T ∪ {s1};
```


Model checking cu fairness

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

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

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