Verificarea sistemelor în timp real

2 noiembrie 2004

– timp discret, timp continuu
– extensii ale algoritmilor de model checking obişnuiţi
– logici temporale cantitative
– modele în timp continuu: automate temporizate

Sisteme temporizate. Domenii şi aplicaţii

= sisteme a căror corectitudine funcţională depinde de satisfacerea unor proprietăţi (restricţii) temporale
– sisteme critice (aviaţie, militare)
– circuite asincrone de mare viteză
– sisteme de fabricaţie; controlul proceselor industriale (ex. chimice)
– protocoale de comunicaţie
– în creştere: sisteme electronice de larg consum
(protocoale multimedia, controlul automobilelor)
– protocoale de sincronizare a bazei de timp (sisteme distribuite, protocoale de securitate)

Modelare

Orice sistem real evoluează în timp fizic ⇒ modele studiate până acum (fără apariţia explicită a timpului) sunt o *abstracţie*
Ex: logica temporală exprimă proprietăţi *calitative*, nu *cantitative*
Totuşi: majoritatea formalismelor pornesc de la un model fără timp la care se agaugă ulterior o dimensiune temporală.
– *timp discret*: toate evenimentele se petrec la momente care sunt multipli ai unei constante de timp
  modele: ex. automate cu durată întreagă pt. fiecare tranziţie
– *timp continuu*: evenimente la momente arbitrare pe scara reală
  modele: automate temporizate, reţele Petri temporizate, limbaje de programare cu facilitaţi de timp

Puţine formalisme create special, cu timpul ca dimensiune primitivă.
Exemplu: *duration calculus* [Zhou, Hoare, Ravn '91], cu operatorii:
– $\lfloor f \rfloor$: *durata* cât este valabilă f (integrală după timp)
– concatenarea a două intervale de timp

Timp discret şi timp continuu

Care e diferenţa în expresivitate şi eficienţă ?
(Cum se compară sistemul în timp continuu cu cel *discretizat* ?)

[Henzinger, Manna, Pnueli '92]: discută *timed transition systems* (automate cu limite de timp inferioare/superioare pt. tranziţii)
– discretizarea păstrează proprietăţile *calitative*şi unele *cantitative*:
ex. cele de invarianţă (**G**p) şi răspuns (p ⇒ **F**q) cu limite de timp
– pentru alte proprietăţi, se obţin variante mai slabe pt. timp discret

[Asarin, Maler, Pnueli '98] discută circuite combinaţionale, cu intârzieri limitate la ieşirea fiecărei porţi:
– pt. circuite aciclice, există o cuantă de discretizare care păstrează comportarea calitativă (ordonarea evenimentelor)
  ex. $1/n$ pentru un circuit cu n semnale
– există însă circuite *ciclice* a căror comportare calitativă nu e păstrată de nici o discretizare (ex. un inel de 3 invertoare)

Teorii clasice pt. sisteme timp real

Principala problemă: planificarea execuţiei (*schedulability analysis*)
Fiind dat un set de procese cu parametrii lor (ex. perioade, deadlines), există o planificare satisfăcătoare ?

Rate-monotonic scheduling [Lehoczky, Liu, Layland]
– atribuie priorităţi în ordinea crescătoare a perioadelor
(se demonstrează optimalitatea)
– test de satisfiabilitate bazat pe utilizarea totală (%)

– Avantaje: metodă simplă, optimă, analiză rapidă
– Dezavantaje: model restrictiv (procese periodice, cu câteva extensii) metodă incompletă, inaplicabilă la încărcări ridicate ($> ln2 \simeq 70\%$)

În continuare, discutăm metode mai generale.

Analiza cantitativă a proprietăţilor temporale

RTCTL permite exprimarea de relaţii temporale cantitative
(ex. p nu apare mai devreme de 5 unităţi de timp)
dar nu o analiză detaliată (care este intârzierea maximă a lui p)

⇒ Definim algoritmi care pot calcula astfel de parametri
şi au o implementare eficientă, *simbolică* (cu BDD-uri)

– lungimea drumului minim şi maxim dintre două mulţimi de stări
(exprimate prin predicatele care le caracterizează)
ex. timpul maxim de încheierea a execuţiei (*schedulability*)
– numărul minim/maxim de apariţii a unei proprietăţi pe o cale
ex. de câte ori procesul este în starea *wait*

[Courcoubetis & Yannakakis; Campos, Clarke et al.]

Drumul minim între două mulţimi de stări

Parcurgere prin cuprindere pornind din $start$ până se atinge prima dată $final$ sau nu se mai ating stări noi.

La fiecare iteratie: Q = stările atinse în i paşi, R = multimea tuturor stărilor atinse, creşte până la punct fix.

procedure min(start, final)
for $(i \leftarrow 0, R \leftarrow Q \leftarrow start; Q \cap final = \emptyset; i{+}{+})$ **do**
  $Q \leftarrow$ Succ(Q); $R' \leftarrow R \cup Q$; /* Q = frontiera */
  if $(R' = R)$ **then**
    return ∞; /* nu se poate atinge $final$ */
  $R \leftarrow R'$; /* R = toate stările atinse */
return i;

Drumul maxim între două mulţimi de stări

Determină lungimea maximă a unui drum până când atinge prima dată $final$ pornind din $start$.
Presupunem: sistemul redus la stările ce pot fi atinse
Căutăm cel mai lung drum din $start$ care nu atinge $final$, prin traversare inapoi din stările ce nu satisfac $final$.
R = punctele iniţiale ale căilor care pot ramâne i paşi în afara lui $final$; scade până la punct fix.

procedure max(start, final)
for $(i \leftarrow 0, R = S \setminus final; R \cap start \neq \emptyset; i{+}{+})$ **do**
  $R' \leftarrow$ Pred$(R) \setminus final$;
  if $(R' = R)$ **then**
    return ∞; /* exista drum care nu atinge $final$ */
  $R = R'$;
return i;

Logici temporale cu timp explicit

Logicile temporale discutate până în prezent (LTL, CTL) au modalităţi temporale (stare următoare, viitor), dar fără referire la timp explicit

⇒ E nevoie de facilităţi suplimentare pentru specificarea proprietăţilor în timp real (ex. răspuns în timp limitat)

Există o mare varietate de logici cu timp explicit, după diverse decizii:
– liniare sau cu ramificaţie
– cu timp discret sau cu timp continuu
– operatori cu limite de timp, sau variabile explicite pt. timp

În funcţie de alegere, apar mari diferenţe de expresivitate, complexitate algoritmică, sau chiar decidabilitate !

Logica temporală RTCTL

Simplifică exprimarea proprietăţilor temporale în cazul discret prin augmentarea operatorilor temporali cu intervale de timp.

Fie o traiectorie $\pi = s_0 s_1 \cdots$. Definim:
– $\pi \models f \mathbf{U}_{[a,b]} g \Leftrightarrow \exists i \,.\, a \leq i \leq b \wedge s_i \models g \wedge \forall j < i \,.\, s_j \models f$
– $\pi \models \mathbf{G}_{[a,b]} f \Leftrightarrow \forall i \,.\, a \leq i \leq b \Rightarrow s_i \models f$
– $\pi \models \mathbf{F}_{[a,b]} f \Leftrightarrow \exists i \,.\, a \leq i \leq b \wedge s_i \models f$

Exemplu: $\mathbf{AG}(p \rightarrow \mathbf{AF}_{[0,3]} q)$:
p e întotdeauna urmat de q după cel mult trei unităţi de timp.
Semantica CTL se obţine pentru $a = 0, b = \infty$.

Algoritmi: recursivi, prin modificarea celor de punct fix:
– $\mathbf{E}[f \mathbf{U}_{[a,b]} g] = f \wedge \mathbf{EX}\, \mathbf{E}[f \mathbf{U}_{[a-1,b-1]} g]$ $(a, b > 0)$
– $\mathbf{E}[f \mathbf{U}_{[0,b]} g] = g \vee (f \wedge \mathbf{EX}\, \mathbf{E}[f \mathbf{U}_{[0,b-1]} g])$ $(b > 0)$
– $\mathbf{E}[f \mathbf{U}_{[0,0]} g] = g$

TPTL: Timed Propositional Temporal Logic

[Alur, Henzinger 1989]
– o extensie a fragmentului propoziţional al LTL (doar formule de cale)
– timp liniar, discret (interpretat peste secvenţe de stări)
– foloseşte variabile explicite pentru timp, dar cu restricţii:
orice variabilă e legată de timpul într-o anumită stare (printr-un operator de cuantificare)

Exemplu: "fiecare p e urmat de un q în cel mult 10 unităţi de timp"

$$\Box x.(p \rightarrow \Diamond y.(q \wedge y \leq x + 10))$$

TPTL: comparaţie cu alte formalisme

Exemplu: răspuns în cel mult 10 unităţi de timp, la request continuu:

$$\Box x.(p \rightarrow p \mathbf{U} y.(q \wedge y \leq x + 10))$$

Comparaţie cu logică temporală de ordinul I

Variabila now reprezintă timpul curent în fiecare stare

$$\Box((p \wedge now = x) \rightarrow p \mathbf{U}(q \wedge now = y \wedge y \leq x + 10))$$

apoi completăm cuantificatorii potriviţi (susceptibil la erori)

$$\Box \forall x.((p \wedge now = x) \rightarrow p \mathbf{U} \exists y.(q \wedge now = y \wedge y \leq x + 10))$$

Comparaţie cu operatori temporali cu limite

– TPTL poate exprima astfel de operatori, e.g. $\Diamond_{\leq 5} \phi$ exprimat ca:

$$x. \Diamond\, y.(y \leq x + 5 \wedge \phi)$$

– operatorii cu limită compară timpul a evenimente succesive
TPTL poate compara timpul a două evenimente arbitrare:

$$\Box x.(p \rightarrow \Diamond(q \wedge \Diamond y.(r \wedge y \leq x + 5)))$$

TCTL: Timed CTL

[Alur, Courcoubetis, Dill, '90]
– cu ramificație, timp continuu, operatori cu limite
– interpretată pe arbori de computație în timp continuu
o *cale* e o funcție $\rho : \mathbb{R} \to S$ de pe numere reale (timp) la stări

Sintaxa: $\phi ::= p \mid \mathit{false} \mid \phi_1 \to \phi_2 \mid \exists \phi_1 \mathbf{U}_{\sim c} \phi_2 \mid \forall \phi_1 \mathbf{U}_{\sim c} \phi_2$
(cu $\sim$ unul din $<, >, \leq, \geq, =$), $p \in AP$, $c \in \mathbb{N}$

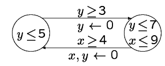
Semantica:
$s \models \exists \phi_1 \mathbf{U}_{\sim c} \phi_2 \ \Leftrightarrow\ \exists \rho, \exists t \sim c$ a.î. $\rho(t) \models \phi_2$
și pentru orice $0 \leq t' < t$, $\rho(t') \models \phi_1$

Satisfiabilitatea: nedecidabilă (!)

Model checking (cu automate temporizate ca model): decidabil
bazat pe construirea unei relații finite de echivalență între stări

Automate temporizate

Unul din cele mai des folosite formalisme pentru descrierea sistemelor in timp continuu [Alur & Dill '90]
= automat cu stări finite, augmentat cu un set de *ceasuri* (clocks)
cu valori reale care avansează sincron
– stări/tranziții etichetate cu condiții de ceas
– un ceas poate fi resetat la executarea unei tranzitii
⇒ măsoară timpul trecut de la producerea unui eveniment



Automat temporizat: Definiție

Multimea condițiilor de ceas $\mathcal{B}(C)$ = conjuncții de termeni de forma $x \prec c$, $c \prec x$, $x - y \prec c$, cu $x, y \in C$, $\prec \in \{<, \leq\}$, $c \in \mathbb{Z}$

$A = (S, S_0, \Sigma, C, I, T)$, unde
– S = mulțime finită de locații (stări, noduri)
– S_0 = mulțime de locatii inițiale
– Σ = alfabet de etichete pentru tranziții
– C = mulțime de variabile ceas (*clocks*)
– $I : S \to \mathcal{B}(C)$ asociază fiecarei stări un invariant
(care limitează trecerea timpului în acea stare)
– $T \subseteq S \times \Sigma \times \mathcal{B}(C) \times 2^C \times S$ = mulțime de tranziții.
Tranziția $\langle s, a, g, R, s' \rangle$ din s in s' etichetată cu a se execută doar dacă condiția g este adevărată, și resetează ceasurile din $R \subseteq C$

Semantica automatelor temporizate

Mulțimea stărilor: perechi (s, v), unde $s \in S$ = locație
și $v : C \to \mathbb{R}$ e o atribuire pentru ceasuri.

Automatul poate rămâne într-o stare atât timp cât invariantul stării e satisfăcut, sau poate executa tranziții instantanee între stări, când condiția asociată cu tranziția e adevărată.

Tranziții: de doua feluri:
– acțiune: $(s, v) \xrightarrow{a} (s', v')$ daca există o tranziție $\langle s, a, g, R, s' \rangle \in T$, condiția $g(v)$ e adevărată pentru atribuirea v, iar v' se obține din v resetând ceasurile din R: $v' = v[R \leftarrow 0]$.
– trecerea timpului: $(s, v) \xrightarrow{d} (s, v')$ dacă $v' = v + d$ ($v'(x) = v(x) + d$, $\forall x \in C$), și $I(s)(v + \epsilon)$ adevarată $\forall \epsilon \in [0, d]$ (invariantul e menținut)

⇒ sistem de tranziții cu număr infinit de stări

Traiectorii de forma $(s_0, v_0) \xrightarrow{d_1} (s_0, v_1) \xrightarrow{a_1} (s_1, v_1') \xrightarrow{d_2} (s_1, v_2) \xrightarrow{a_2} \ldots$

Compoziția paralelă a automatelor

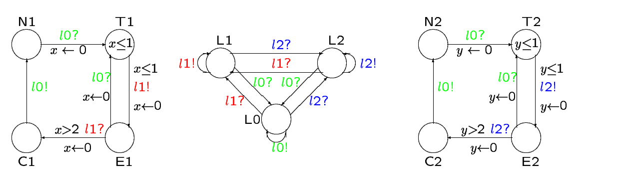
Execută tranziții sincrone dacă etichetele coincid, și tranziții separate in caz contrar.
Fie $A_1 = (S_1, S_{01}, \Sigma_1, C_1, I_1, T_1)$ și $A_2 = (S_2, S_{02}, \Sigma_2, C_2, I_2, T_2)$,
cu $C_1 \cap C_2 = \emptyset$.
Definim $A = A_1 || A_2 = (S_1 \times S_2, S_{01} \times S_{02}, \Sigma_1 \cup \Sigma_2, C_1 \cup C_2, I, T)$, unde:
– $I((s_1, s_2)) = I_1(s_1) \wedge I_2(s_2)$
– dacă $\langle s_1, a, g_1, R_1, s_1' \rangle \in T_1$ și $\langle s_2, a, g_2, R_2, s_2' \rangle \in T_2$, cu $a \in \Sigma_1 \cap \Sigma_2$ atunci $\langle (s_1, s_2), a, g_1 \wedge g_2, R_1 \cup R_2, (s_1', s_2') \rangle \in T$ (sincronizare)
– dacă $\langle s_1, a_1, g_1, R_1, s_1' \rangle \in T_1$, cu $a \in \Sigma_1 \setminus \Sigma_2$, atunci $\forall s_2 \in S_2$,
$\langle (s_1, s_2), a_1, g_1, R_1, (s_1', s_2) \rangle \in T$
– dacă $\langle s_2, a_2, g_2, R_2, s_2' \rangle \in T_2$, cu $a \in \Sigma_2 \setminus \Sigma_1$, atunci $\forall s_1 \in S_1$,
$\langle (s_2, s_1), a_2, g_2, R_2, (s_2', s_1) \rangle \in T$

Mai general: funcție de sincronizare pe etichetele tranzițiilor

Exemplu: excluziune mutuală

Protocolul de excluziune mutuală al lui Fischer
corectitudine bazată pe respectarea restricțiilor de timp

Sincronizare: perechi de tranziții $a?$ și $a!$

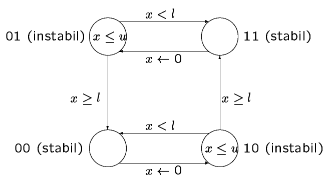


Se demonstrează: corect dacă constantele de timp (aici 1 și 2)
sunt în aceeași relație de ordonare ca și în exemplu

Exemplu: Modelarea circuitelor asincrone

Model în timp continuu ⇒ mai precis decât în timp discret, potrivit pentru modelarea efectelor asincrone și tranzitorii

Ex. element de întârziere: propagă intrarea la ieșire
– dacă pulsul de la intrare nu e mai scurt de l
– cu o întârziere cel mult egală cu u



Reprezentări finite pentru automate temporizate

Stare = pereche (s, v) de locație și atribuire pentru ceasuri
⇒ spațiul stărilor e infinit (mai mult: nenumărabil)

Dar: nu putem observa comportamentul cu precizie arbitrară
– constrângerile din automat au limite întregi de timp
– formulele logicii temporale au de asemenea constante întregi

Întrebări:
– când sunt echivalente două stări (s, v) and (s, v') cu aceeași locație, dar diferite atribuiri pentru ceasuri ?
– și există un număr *finit* de clase de echivalență ?

Două abordări:
– *regiuni temporale* ⇒ graf al regiunilor: automat finit
– *zone temporale* (geometrice) ⇒ explorare simbolică

Graful regiunilor: Motivație

Când sunt echivalente două stări (s, v) și (s, v') ?

1. dacă aceleași tranziții pot fi executate din ambele stări
– condițiile pe tranziții pot avea limite întregi arbitrare
ex. pot exista tranziții a cu $x > 4$ și b cu $x < 5$
$(s, x = 4.2)$ și $(s, x = 4.7)$ pot executa fie a fie b
$(s, x = 4.2)$ și $(s, x = 5.1)$ *nu* sunt ecivalente
⇒ trebuie să aibă aceeași parte întreagă pentru fiecare ceas

$(s, x = 4)$ nu poate executa a, dar $(s, x = 4.1)$ poate.
⇒ părțile fracționare trebuie să fie ambele nule sau ambele nenule

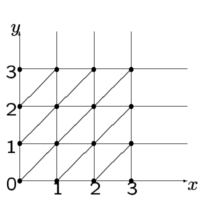
2. trebuie să execute tranzițiile *în aceeași ordine*
considerăm tranzițiile a cu $x \ge 2$ și b cu $y \ge 3$.
din starea $(s, x = 1.5, y = 2.7)$ se poate executa b înainte de a
din starea $(s, x = 1.4, y = 2.3)$ se poate executa a înainte de b
⇒ stările nu sunt echivalente
⇒ ceasurile trebuie să aibă aceeași ordonare pt. părțile fracționare

Graful regiunilor. Definiție

[Alur & Dill '90]: definim $v \simeq v'$ dacă:
– $\forall x \in C \,.\, \lfloor v(x) \rfloor = \lfloor v'(x) \rfloor \lor (\lfloor v(x) \rfloor \ge c_x \land \lfloor v'(x) \rfloor \ge c_x)$ unde $c_x \in \mathbb{Z}$ e cea mai mare constantă cu care e comparat x în automat
(părțile întregi ale valorilor ceasurilor sunt fie egale in ambele atribuiri, fie ambele mai mari sau egale cu constanta maximă)
– $\forall x, y \in C$, $\lfloor v(x) \rfloor < c_x$, $\lfloor v(y) \rfloor < c_y$, $\{v(x)\} \le \{v(y)\} \Leftrightarrow \{v'(x)\} \le \{v'(y)\}$
(părțile fracționare ale ceasurilor au aceeași ordine în ambele atribuiri)
– $\forall x \in C$ cu $\lfloor v(x) \rfloor \le c$, $\{v(x)\} = 0 \Leftrightarrow \{v'(x)\} = 0$

⇒ *regiunea* asociata cu starea (s, v) = multimea stărilor (s, v') cu $v \simeq v'$.
⇒ reprezentare cu număr finit de clase de echivalență

Graful regiunilor (cont.)



Ex.: graf al regiunilor pt. două ceasuri și constantă maximă $c = 3$

Regiunile sunt:
– 0-dimensionale: puncte de coordonate întregi, $x, y \in \{0, 1, 2, 3\}$
– unidimensionale: segmente/diagonale; segmente nelimitate (≥ 3)
– bidimensionale: limitate (triunghiuri) sau nu (benzi rectangulare)

Din două stări (puncte) din aceeași regiune:
– se pot face aceleași tranziții
– prin trecerea timpului, se parcurg aceleași regiuni

Model checking pentru graful regiunilor

[Alur, Courcoubetis, Dill '90]
Pentru automatul temporizat A, definim automatul finit $R(A)$:
– stările lui $R(A)$ sunt *regiuni*
– există tranziții între regiunile r și r' dacă și numai dacă
r' e regiunea *succesor* a lui r în raport cu trecerea timpului
există o tranziție (acțiune) $(s, v) \xrightarrow{a} (s', v')$ între doi reprezentanți (stări temporizate) $(s, v) \in r$ și $(s', v') \in r'$

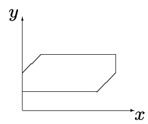
Se demonstrează: model checking TCTL pt. un automat temporizat se reduce la model checking CTL pentru graful regiunilor
(cu ceasuri suplimentare pentru a măsura durata operatorilor)

Dimensiunea grafului regiunilor: cel mult $|C|! \cdot 2^{|C|} \prod_{x \in C} (2c_x + 2)$
– exponențială în numărul de ceasuri
– exponentială în valoarea constantei maxime (problematic în practică)

Reprezentări finite. Zone temporale

Graful regiunilor: exponenţial in c si numărul de ceasuri
$\Rightarrow$ adeseori foarte costisitor de construit si analizat
$\Rightarrow$ reprezentare alternativă: prin inegalităţi temporale

zonă temporală = condiţie din $\mathcal{B}(C)$
ex. $x \leq 5 \wedge 1 \leq y \leq 3 \wedge -2 \leq x - y \leq 3$
(reprezintă un poliedru convex în hiperspaţiul $\mathbb{R}^{|C|}$)



$\Rightarrow$ o zonă = o reuniune convexă de regiuni

Graful zonelor temporale

Considerăm zone maximale relativ la evoluţia posibilă a timpului într-o locaţie (până la limita impusă de invariant)
$\Rightarrow$ zone iniţiale: $\langle s_0, I(s_0) \wedge \bigwedge_{i \neq j}(x_i = x_j)\rangle$ cu $s_0 \in S_0$, $x_i, x_j \in C$

Succesorii unei zone ϕ printr-o tranziţie combinată acţiune + timp:
– se face conjuncţia cu condiţia g a tranziţiei: $\phi \wedge g$
– se resetează ceasurile asociate cu tranziţia: $\phi[x \leftarrow 0] = \exists x \phi \wedge (x = 0)$
(cuantificare existenţială după $x \in R$, şi apoi conjuncţie cu $x = 0$)
– se consideră trecerea timpului: $\phi\Uparrow = \exists t > 0 \,.\, \phi(v - t)$
(se elimină inegalităţile $x \prec d$)
– se impune invariantul stării destinaţie (conjuncţie cu $I(s')$)

Pe ansamblu:
$$\phi' = (\phi \wedge g)[R \leftarrow 0] \Uparrow \wedge I(s')$$

Reprezentarea zonelor: difference bound matrices

O zonă = conjuncţie de inegalităţi $x - y \prec c$, $x \prec c$ sau $c \prec x$
$\Rightarrow$ se poate reprezenta cu matrice pătrată de dimensiune $|C| + 1$
(o linie pentru fiecare ceas, si una pentru comparaţia cu zero)
Elementele sunt intregi din intervalul $[-c, c]$:
valoarea d pentru (x, y) $(x, y \in C)$ inseamna $x - y \leq d$
(plus eventual un bit pentru inegalitate strictă sau nu)
prima linie si coloană: pentru comparaţii cu valoarea 0

Pentru a obţine nr. finit de zone: aceeaşi observaţie ca la regiuni
$x \prec d$ pentru $d > c_{max}$ devine $x \prec \infty$
$x \prec d$ pentru $d < -c_{max}$ devine $x \prec -c_{max}$

Lucrul cu matrici de diferenţe

Exemplu: $x \leq 5 \wedge 1 \leq y \wedge -2 \leq x - y \leq 3$

	0	x	y
0	≤ 0	≤ 0	≤ -1
x	≤ 5	≤ 0	≤ 3
y	$\leq \infty$	≤ 2	≤ 0

– conjuncţie dintre două zone: elementele minime din cele 2 matrici
+ relaxare pentru propagarea constrângerilor (drumuri minime în graf)
– resetarea unui ceas: copiem linia/coloana 0 în linia/coloana pt. ceas
– trecerea timpului: setarea limitelor $x \prec c$ la $x \prec \infty$

În practică, verificatoarele (ex. UPPAAL, KRONOS) folosesc această reprezentare (sau variante optimizate)