

Verificarea sistemelor în timp real

2 noiembrie 2004

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

Sisteme temporizate. Domenii și aplicații

- = sisteme a căror corectitudine funcțională depinde de satisfacerea unor proprietăți (restricții) temporale
- sisteme critice (aviație, militare)
- circuite asincrone de mare viteză
- sisteme de fabricație; controlul proceselor industriale (ex. chimice)
- protocoale de comunicație
- în creștere: 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 \Rightarrow modele studiate până acum (fără apariția explicită a timpului) sunt o *abstracție*

Ex: logica temporală exprimă proprietăți *calitative*, nu *cantitative*

Totuși: majoritatea formalismelor pornesc de la un model fără timp la care se adaugă 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 tranziție

– *timp continuu*: evenimente la momente arbitrare pe scara reală

 modele: automate temporizate, rețele Petri temporizate, limbaje de programare cu facilități de timp

Puține formalisme create special, cu timpul ca dimensiune primitivă.

Exemplu: *duration calculus* [Zhou, Hoare, Ravn '91], cu operatorii:

– $[f]$: *durata* cât este valabilă f (integrală după timp)

– concatenarea a două intervale de timp

Timp discret și timp continuu

Care e diferența în expresivitate și eficiență ?

(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. tranziții)

– discretizarea păstrează proprietățile *calitative* și unele *cantitative*:

ex. cele de invarianță ($\mathbf{G}p$) și răspuns ($p \Rightarrow \mathbf{F}q$) cu limite de timp

– pentru alte proprietăți, se obțin variante mai slabe pt. timp discret

[Asarin, Maler, Pnueli '98] discută circuite combinaționale, cu întârzieri limitate la ieșirea fiecărei porți:

– 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 execuției (*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 priorități în ordinea crescătoare a perioadelor (se demonstrează optimalitatea)
- test de satisfiabilitate bazat pe utilizarea totală (%)
- Avantaje: metodă simplă, optimă, analiză rapidă
- Dezavantaje: model restrictiv (processe periodice, cu câteva extensii) metodă incompletă, inaplicabilă la încărcări ridicate ($> \ln 2 \simeq 70\%$)

În continuare, discutăm metode mai generale.

Analiza cantitativă a proprietăților temporale

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

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

– lungimea drumului minim și maxim dintre două mulțimi de stări (exprimate prin predicatele care le caracterizează)

ex. timpul maxim de încheierea a execuției (*schedulability*)

– numărul minim/maxim de apariții a unei proprietăți pe o cale
ex. de câte ori procesul este în starea *wait*

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

Drumul minim între două mulțimi 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 pași, R = multimea tuturor stărilor atinse, crește 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 $\infty; /* nu se poate atinge final */$

$R \leftarrow R'; /* R = toate stările atinse */$

return $i;$

Drumul maxim între două mulțimi 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 inițiale ale căilor care pot rămâne i pași 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**

$R' \leftarrow \text{Pred}(R) \setminus final;$

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 modalități temporale (stare următoare, viitor), dar fără referire la timp explicit

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

Există o mare varietate de logici cu timp explicit, după diverse decizii:

- liniare sau cu ramificație
- cu timp discret sau cu timp continuu
- operatori cu limite de timp, sau variabile explicite pt. timp

În funcție de alegere, apar mari diferențe de expresivitate, complexitate algoritmică, sau chiar decidabilitate !

Logica temporală RTCTL

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

Fie o traiectorie $\pi = s_0s_1 \dots$. 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 unități de timp.

Semantica CTL se obține pentru $a = 0, b = \infty$.

Algoritmi: recursivi, prin modificarea celor de punct fix:

- $\mathbf{E}[f\mathbf{U}_{[a,b]}g] = f \wedge \mathbf{EXE}[f\mathbf{U}_{[a-1,b-1]}g] \quad (a, b > 0)$
- $\mathbf{E}[f\mathbf{U}_{[0,b]}g] = g \vee (f \wedge \mathbf{EXE}[f\mathbf{U}_{[0,b-1]}g]) \quad (b > 0)$
- $\mathbf{E}[f\mathbf{U}_{[0,0]}g] = g$

TPTL: Timed Propositional Temporal Logic

[Alur, Henzinger 1989]

- o extensie a fragmentului propozițional al LTL (doar formule de cale)
- timp liniar, discret (interpretat peste secvențe de stări)
- folosește variabile explicite pentru timp, dar cu restricții:
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 unități de timp”

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

TPTL: comparație cu alte formalisme

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

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

Comparație 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 potriviți (susceptibil la erori)

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

Comparație 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 $\rho : \mathbb{R} \rightarrow S$ de pe numere reale (timp) la stări

Sintaxa: $\phi ::= p \mid false \mid \phi_1 \rightarrow \phi_2 \mid \exists \phi_1 \mathbf{U}_{\sim c} \phi_2 \mid \forall \phi_1 \mathbf{U}_{\sim c} \phi_2$

(cu \sim unul din $<, >, \leq, \geq, =$), $p \in AP$, $c \in \mathbb{N}$

Semantica:

$s \models \exists \phi_1 \mathbf{U}_{\sim c} \phi_2 \Leftrightarrow \exists \rho, \exists t \sim c \text{ a.î. } \rho(t) \models \phi_2$

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

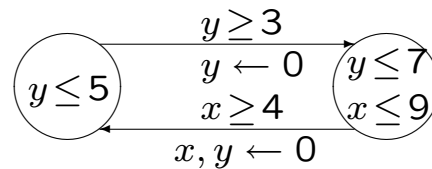
Unul din cele mai des folosite formalisme pentru descrierea sistemelor in timp continuu [Alur & Dill '90]

= 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 locații inițiale
- $\Sigma =$ alfabet de etichete pentru tranziții
- $C =$ mulțime de variabile ceas (*clocks*)
- $I : S \rightarrow \mathcal{B}(C)$ asociază fiecărei 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 în 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 $s \in S = \text{locație}$ și $v : C \rightarrow \mathbb{R}$ 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: $(s, v) \xrightarrow{a} (s', v')$ dacă 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 $I(s)(v + \epsilon)$ adevărată $\forall \epsilon \in [0, d]$ (invariantul e menținut)

\Rightarrow sistem de tranziții cu număr infinit de stări

Traietorii de forma $(s_0, v_0) \xrightarrow{d_1} (s_0, v_1) \xrightarrow{a_1} (s_1, v'_1) \xrightarrow{d_2} (s_1, v_2) \xrightarrow{a_2} \dots$

Compoziția paralelă a automatelor

Execută tranziții sincrone dacă etichetele coincid, și tranziții separate în 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 $\forall s_2 \in S_2$,
 $\langle (s_1, s_2), a_1, g_1, R_1, (s'_1, s_2) \rangle \in T$

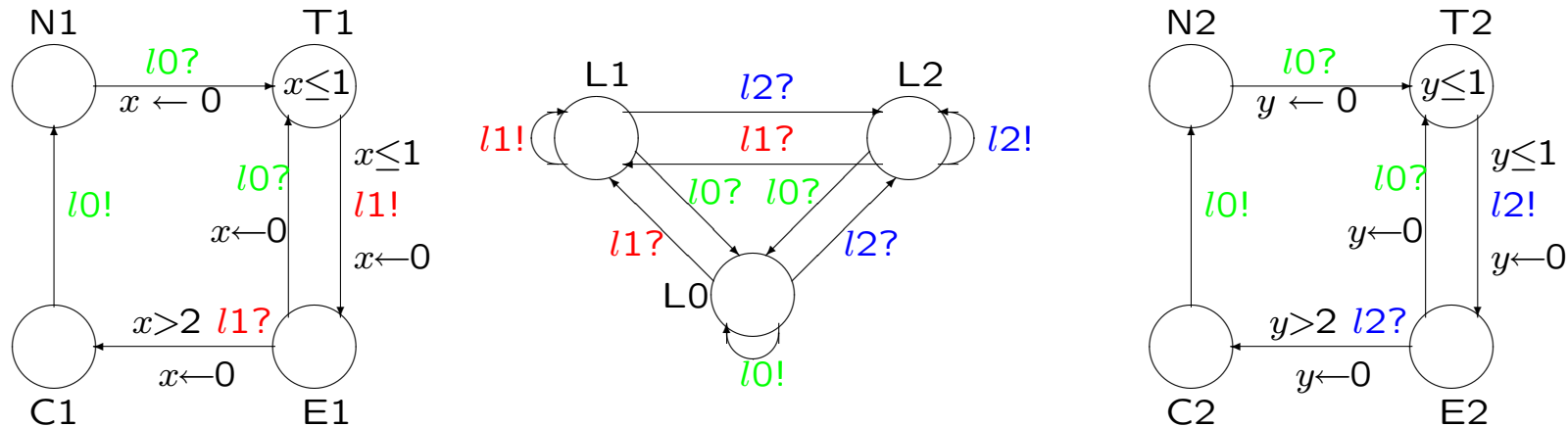
– dacă $\langle s_2, a_2, g_2, R_2, s'_2 \rangle \in T_2$, cu $a \in \Sigma_2 \setminus \Sigma_1$, atunci $\forall s_1 \in S_1$,
 $\langle (s_2, s_1), a_2, g_2, R_2, (s'_2, s_1) \rangle \in T$

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

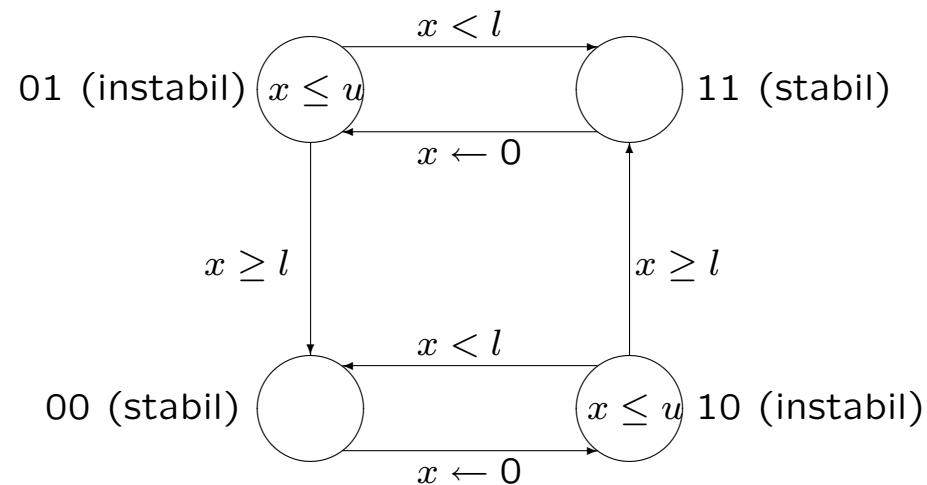
Exemplu: Modelarea circuitelor asincrone

Model în timp continuu \Rightarrow 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 $x \geq 2$ și b cu $y \geq 3$.

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:

– $\forall x \in C . \lfloor v(x) \rfloor = \lfloor v'(x) \rfloor \vee (\lfloor v(x) \rfloor \geq c_x \wedge \lfloor v'(x) \rfloor \geq c_x)$ 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 în ambele atribuiri, fie ambele mai mari sau egale cu constanta maximă)

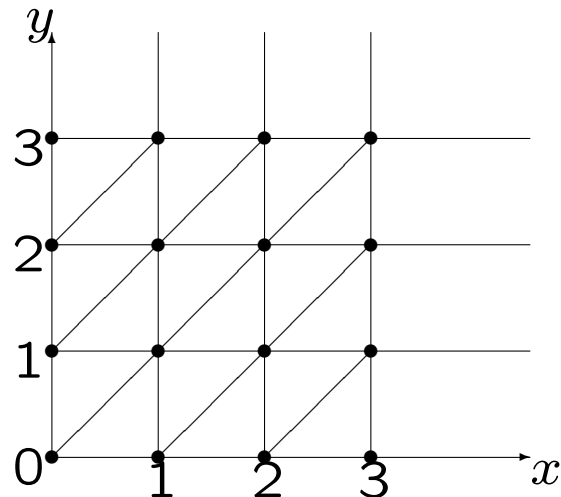
– $\forall x, y \in C, \lfloor v(x) \rfloor < c_x, \lfloor v(y) \rfloor < c_y, \{v(x)\} \leq \{v(y)\} \Leftrightarrow \{v'(x)\} \leq \{v'(y)\}$
(părțile fracționare ale ceasurilor au aceeași ordine în ambele atribuiri)

– $\forall x \in C$ cu $\lfloor v(x) \rfloor \leq c, \{v(x)\} = 0 \Leftrightarrow \{v'(x)\} = 0$

\Rightarrow regiunea asociată cu starea $(s, v) =$ mulțimea stărilor (s, v') cu $v \simeq v'$.

\Rightarrow 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: exponențial în c și numărul de ceasuri

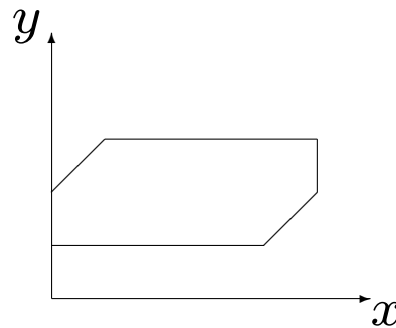
⇒ adeseori foarte costisitor de construit și analizat

⇒ reprezentare alternativă: prin inegalități temporale

zonă temporală = condiție 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 spațiul $\mathbb{R}^{|C|}$)



⇒ o zonă = o reuniune convexă de regiuni

Graful zonelor temporale

Considerăm zone maximale relativ la evoluția posibilă a timpului într-o locație (până la limita impusă de invariant)

⇒ zone inițiale: $\langle s_0, I(s_0) \wedge \bigwedge_{i \neq j} (x_i = x_j) \rangle$ cu $s_0 \in S_0$, $x_i, x_j \in C$

Succesorii unei zone ϕ printr-o tranziție combinată acțiune + timp:

- se face conjuncția cu condiția g a tranziției: $\phi \wedge g$
- se resetează ceasurile asociate cu tranziția: $\phi[x \leftarrow 0] = \exists x \phi \wedge (x = 0)$
(cuantificare existențială după $x \in R$, și apoi conjuncție cu $x = 0$)
- se consideră trecerea timpului: $\phi \uparrow = \exists t > 0 . \phi(v - t)$
(se elimină inegalitățile $x < d$)
- se impune invariantul stării destinație (conjuncție cu $I(s')$)

Pe ansamblu:

$$\phi' = (\phi \wedge g)[R \leftarrow 0] \uparrow \wedge I(s')$$

Reprezentarea zonelor: difference bound matrices

O zonă = conjuncție de inegalități $x - y \prec c$, $x \prec c$ sau $c \prec x$
 \Rightarrow se poate reprezenta cu matrice pătrată de dimensiune $|C| + 1$
 (o linie pentru fiecare ceas, și una pentru comparația cu zero)

Elementele sunt întregi din intervalul $[-c, c]$:

valoarea d pentru (x, y) ($x, y \in C$) înseamnă $x - y \leq d$

(plus eventual un bit pentru inegalitate strictă sau nu)

prima linie și coloană: pentru comparații cu valoarea 0

Pentru a obține nr. finit de zone: aceeași observație 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 diferențe

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

- conjuncție 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)