

Model checking cu automate

Relații între modele. Raționament compozițional

26 octombrie 2004

Relația între implementare și specificație

Am discutat până acum:

implementarea (modelul): automat cu stări finite

specificația: formulă în logică temporală (LTL, CTL)

○ altă viziune:

– și specificația e un automat

– cu “mai puține detalii” decât implementarea

– model checking pentru LTL: prin traducerea formulei în automat

Mai general: cum definim relația de rafinare între model și specificație?

– incluziunea limbajelor, simulare, bisimulare

Cum se păstrează aceste relații de rafinare la compoziția modulelor ?

Model checking pentru LTL

Ideea de ansamblu:

- verificăm formule de tipul $\mathbf{A}f$ ($f =$ formulă de traiectorie în care singurele subformule de stare sunt propoziții atomice)
- $\mathbf{A}f = \neg\mathbf{E}\neg f \Rightarrow$ suficient să considerăm $\mathbf{E}f$.
- construim un *tableau* T pentru formula $f =$ un automat (structură Kripke) care exprimă *toate* traiectoriile care satisfac f
- se compune modelul M cu tabloul T
- se verifică dacă există o traiectorie în compoziție (cu algoritmi de model checking CTL)

Construirea tabloului. Formule elementare

Fie AP_f mulțimea propozițiilor atomice care apar în f .

$T = (S_T, R_T, L_T)$, cu $L_T : S_T \rightarrow 2^{AP_f}$.

Stările tabloului: mulțimi de *formule elementare* extrase din f .

- $el(p) = \{p\}$ pentru $p \in AP_f$
- $el(\neg g) = el(g)$
- $el(g_1 \vee g_2) = el(g_1) \cup el(g_2)$
- $el(\mathbf{X}g) = \{\mathbf{X}g\} \cup el(g)$
- $el(g_1 \mathbf{U} g_2) = \{\mathbf{X}(g_1 \mathbf{U} g_2)\} \cup el(g_1) \cup el(g_2)$

Mulțimea stărilor tabloului: $S_T = \mathcal{P}(el(f))$

Relația de satisfacere în tablou

Asociem fiecărei subformule din f o mulțime de stări din T (intuitiv: mulțimea de stări care satisfac acea formulă)

- $sat(g) = \{s \mid g \in s\}$ pentru $g \in el(f)$
- $sat(\neg g) = \{s \mid s \notin sat(g)\}$
- $sat(g_1 \vee g_2) = sat(g_1) \cup sat(g_2)$
- $sat(g_1 \mathbf{U} g_2) = sat(g_2) \cup (sat(g_1) \cap sat(\mathbf{X}(g_1 \mathbf{U} g_2)))$

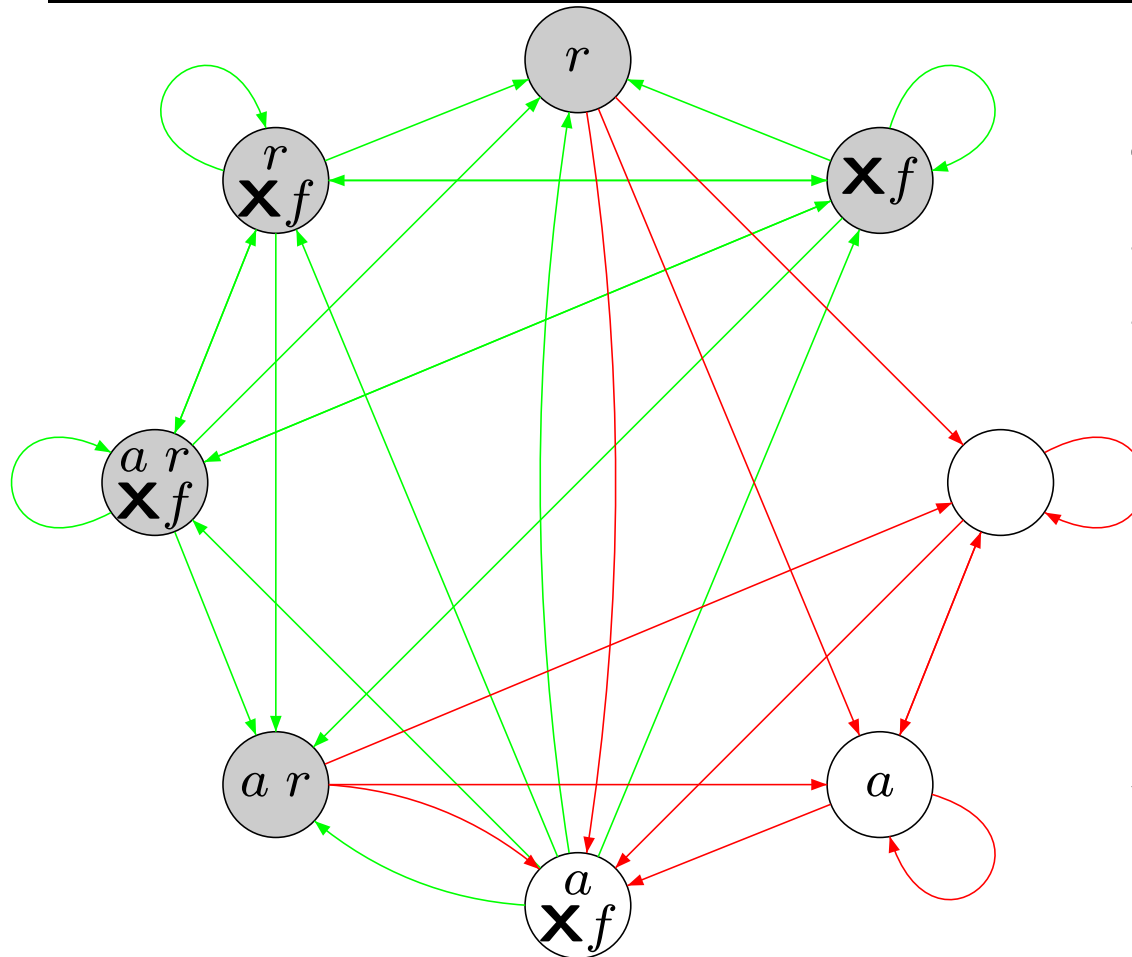
Relația de tranziție: consistentă cu semantica lui \mathbf{X}

$$- \mathbf{X}g \in s \rightarrow \forall s' . R(s, s') \rightarrow g \in s'$$

$$- \mathbf{X}g \notin s \rightarrow \forall s' . R(s, s') \rightarrow g \notin s'$$

$$R_T(s, s') = \bigwedge_{\mathbf{X}g \in el(f)} s \in sat(\mathbf{X}g) \Leftrightarrow s' \in sat(g)$$

Un exemplu: $f = (\neg ack)Urecv$



$$el(f) = \{a, r, \mathbf{X}f\}$$

$$sat(f) = \text{shaded circle} =$$

$$sat(r) \cup (\neg sat(a) \cap sat(\mathbf{X}f))$$

$$R_T = \text{green arrow} \cup \text{red arrow}$$

$$\text{green arrow} = sat(\mathbf{X}f) \times sat(f)$$

$$\text{red arrow} = \neg sat(\mathbf{X}f) \times \neg sat(f)$$

Calculul produsului

Definim $T \times M = (S_T, R_T, L_T) \times (S_M, R_M, L_M) = (S, R, L) = P$

- $S = \{(s_T, s_M) \mid s_T \in S_T, s_M \in S_M, L_T(s_T) = L_M(s_M) \cap AP_f\}$
- $R((s_T, s_M), (s'_T, s'_M)) = R_T(s_T, s'_T) \wedge R_M(s_M, s'_M)$
- $L((s_T, s_M)) = L_T(s_T)$

(tranziții simultane, doar pentru stările etichetate la fel).

Produsul: restrâns la stările din care există cel puțin o tranziție.

Problemă: T nu garantează proprietățile de *eventualitate*:

R_T asigură $sat(g\mathbf{U}h)$ continuu până la $sat(h)$, dar nu și $\mathbf{F}sat(h)$
 \Rightarrow model checking cu *fairness*: $\{sat(g\mathbf{U}h) \rightarrow h \mid g\mathbf{U}h \text{ apare în } f\}$

Teoremă: $M, s_M \models \mathbf{E}f \Leftrightarrow \exists s_T \in sat(f) . P, (s_T, s_M) \models_F \mathbf{EG} True$
 cu condițiile de fairness $\{sat(g\mathbf{U}h) \rightarrow h \mid g\mathbf{U}h \text{ apare în } f\}$

Incluziunea între limbaje

= language inclusion, trace inclusion

Fie o structură Kripke M cu o mulțime AP de propoziții atomice

Limbajul lui M = mulțimea execuțiilor văzută ca secvență de etichetări

Formal: $\mathcal{L}(M)$ = mulțimea de cuvinte (șiruri) infinite $\alpha_0\alpha_1\alpha_2\dots$

astfel încât există o cale $s_0s_1s_2\dots$ a lui M cu $L(s_i) = \alpha_i$.

Incluziunea între limbaje păstrează exact proprietățile LTL:

$$\mathcal{L}(M) \subseteq \mathcal{L}(S) \Leftrightarrow \forall \mathbf{A}f \in LTL . S \models \mathbf{A}f \Rightarrow M \models \mathbf{A}f$$

Relația de simulare

Fie două structuri M și M' , cu $AP \supseteq AP'$. O relație $\preceq \subseteq S \times S'$ este o relație de *simulare* între M și M' dacă și numai dacă $\forall s \preceq s'$:

- $L(s) \cap AP' = L'(s')$ (s și s' etichetate la fel în raport cu AP')
- $\forall s_1$ cu $s \rightarrow s_1$ există s'_1 cu $s' \rightarrow s'_1$ și $s_1 \preceq s'_1$

(orice succesori al lui s e simulat de un succesori al lui s')

Structura M' simulează pe M ($M \preceq M'$) dacă există o relație de simulare \preceq a.î. pt. stările inițiale: $\forall s_0 \in S_0 \exists s'_0 \in S'_0 . s_0 \preceq s'_0$

Prop: Relația de simulare este o *preordine* pe mulțimea structurilor. (reflexivă și tranzitivă). Alegem: $s \preceq s'' \Leftrightarrow \exists s' . s \preceq_1 s' \wedge s' \preceq_2 s''$

Teoremă: Dacă $M \preceq M'$, atunci $M' \models f \Rightarrow M \models f$, pentru orice formulă f în ACTL* peste AP' .

Relația de bisimulare

Fie M și M' două structuri cu $AP' = AP$. O relație $\simeq \subseteq S \times S'$ este o relație de *bisimulare* între M și M' dacă și numai dacă $\forall s, s'$ cu $s \simeq s'$:

– $L(s) = L(s')$

– $\forall s_1$ cu $s \rightarrow s_1$ există s'_1 cu $s' \rightarrow s'_1$ și $s_1 \simeq s'_1$

– $\forall s'_1$ cu $s' \rightarrow s'_1$ există s_1 cu $s \rightarrow s_1$ și $s_1 \simeq s'_1$

(sau: \simeq relație de simulare *simetrică*, între M și M' și între M' și M)

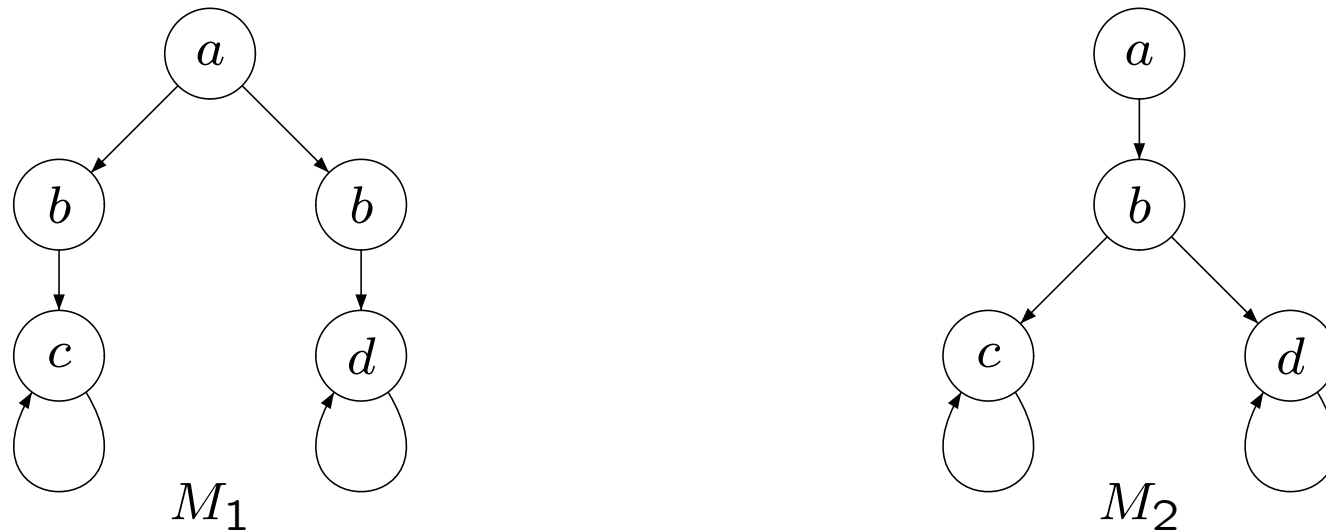
Structurile M și M' sunt *bisimilare* dacă \exists relație de bisimulare \simeq a.î. pt. stările inițiale: $\forall s_0 \in S_0 \exists s'_0 \in S'_0 . s_0 \simeq s'_0$, și $\forall s'_0 \in S'_0 \exists s_0 \in S_0 . s_0 \simeq s'_0$.

Prop: Relația de bisimulare este o relație de echivalență între structuri.

Teoremă: Dacă $M \simeq M'$ atunci $\forall f \in CTL^*$, $M \models f \Leftrightarrow M' \models f$.

Reciproc: Două structuri care satisfac aceleași formule CTL^* (chiar CTL) sunt bisimilare (echivalent: două structuri care nu sunt bisimilare pot fi deosebite printr-o formulă CTL).

Exemplu: incluziunea limbajelor si simulare

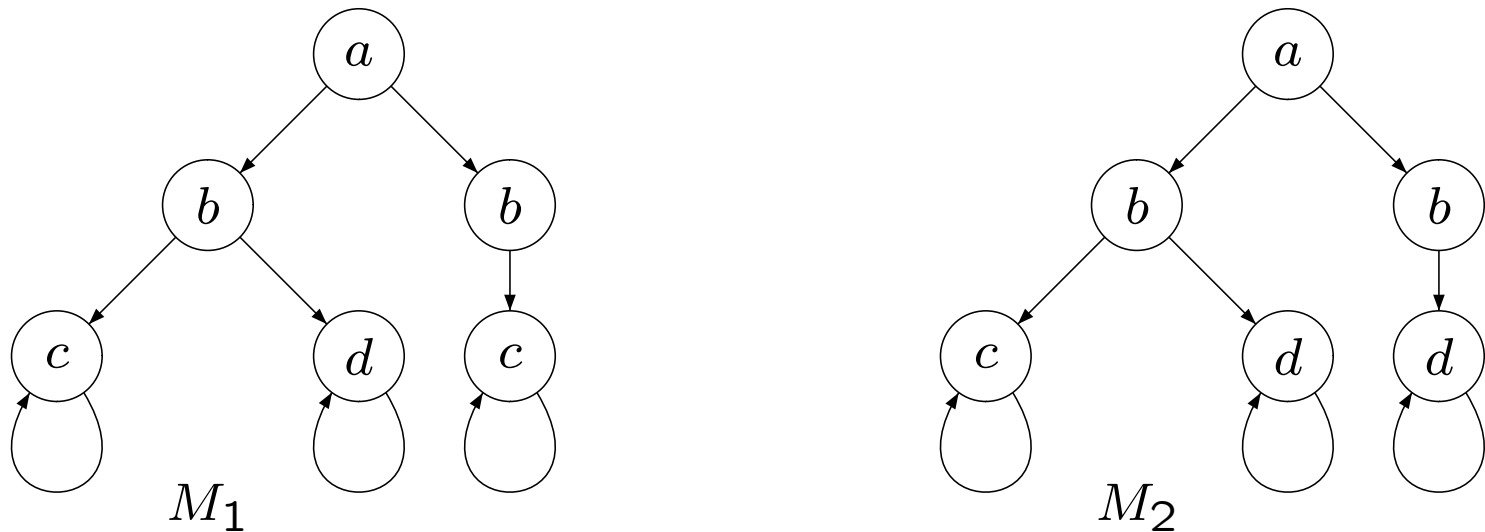


În general: $M \preceq M' \Rightarrow \mathcal{L}(M)|_{AP'} \subseteq \mathcal{L}(M')$

În figură: $\mathcal{L}(M_1) = \mathcal{L}(M_2)$, $M_1 \preceq M_2$, $M_2 \not\preceq M_1$

Definiție echivalentă (teoria jocurilor): $M \preceq M'$ dacă orice mutare în M poate fi urmată de o mutare etichetată la fel în M' .

Exemplu: simulare și bisimulare



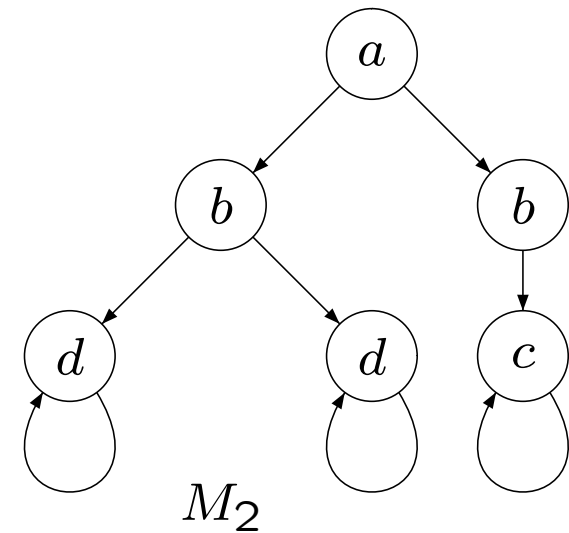
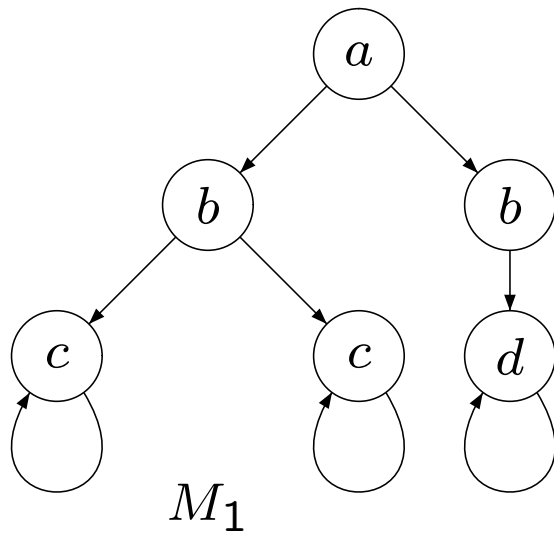
În general: $M \simeq M' \Rightarrow M \preceq M' \wedge M' \preceq M$

În figură: $M_1 \preceq M_2$, $M_2 \preceq M_1$ dar $M_1 \not\approx M_2$

Definiție echivalentă (teoria jocurilor): $M \simeq M'$ dacă orice alegere a unuia din modele și a unei mutări în el poate fi urmată de o mutare etichetată la fel în celălalt model.

(alegerea modelului se face la fiecare pas \Rightarrow simetrie)

Exemplu: bisimulare



$$M_1 \simeq M_2$$

(duplicarea nodurilor nu schimbă proprietățile de ramificare)

Extindere la fairness

Relația $\preceq_F \subseteq S \times S'$ este o relație de simulare *echitabilă* între M și M' (cu $AP' \subseteq AP$) dacă și numai dacă $\forall s \preceq_F s'$:

- $L(s) \cap AP' = L'(s')$
- pentru orice traiectorie *echitabilă* $\pi = ss_1s_2\dots$ în M există o traiectorie *echitabilă* $\pi' = s's'_1s'_2\dots$ în M' a.î. $\forall i > 0 . s_i \preceq s'_i$.

Dacă $M \preceq_F M'$, atunci $\forall f \in ACTL^*$, $M' \models_F f \Rightarrow M \models_F f$

Relația $\simeq_F \subseteq S \times S'$ este o relație de bisimulare *echitabilă* între M și M' (cu $AP' = AP$) dacă și numai dacă $\forall s \simeq_F s'$:

- $L(s) = L(s')$
- pentru orice traiectorie *echitabilă* $\pi = ss_1s_2\dots$ în M există o traiectorie *echitabilă* $\pi' = s's'_1s'_2\dots$ în M' a.î. $\forall i > 0 . s_i \simeq s'_i$.
- pentru orice traiectorie *echitabilă* $\pi' = s's'_1s'_2\dots$ în M' există o traiectorie *echitabilă* $\pi = ss_1s_2\dots$ în M a.î. $\forall i > 0 . s_i \simeq s'_i$.

Dacă $M \simeq_F M'$, atunci $\forall f \in CTL^*$, $M' \models_F f \Leftrightarrow M \models_F f$

Algoritmi de verificare a (bi)simulării

Sistem determinist: o singură stare inițială; orice doi succesori etichetați diferit: $s \rightarrow s_1 \wedge s \rightarrow s_2 \wedge s_1 \neq s_2 \Rightarrow L(s_1) \neq L(s_2)$

Simulare:

M, M' deterministe: $M \preceq M' \Leftrightarrow \mathcal{L}(M) \subseteq \mathcal{L}(M')$

În general: definim recursiv: $s \preceq_0 s' \Leftrightarrow L(s) \cap AP' = L(s')$

$s \preceq_{n+1} s' \Leftrightarrow s \preceq_n s' \wedge \forall s_1 . s \rightarrow s_1 \Rightarrow \exists s'_1 . s' \rightarrow s'_1 \wedge s_1 \preceq_n s'_1$

Avem $\preceq_{i+1} \subseteq \preceq_i \Rightarrow \exists n . \preceq_n = \preceq_{n+1} = \preceq$ (modele finite)

Bisimulare:

M, M' deterministe: $M \simeq M' \Leftrightarrow \mathcal{L}(M) = \mathcal{L}(M')$

În general: definim recursiv: $s \simeq_0 s' \Leftrightarrow L(s) = L(s')$

$s \simeq_{n+1} s' \Leftrightarrow s \simeq_n s' \wedge \forall s_1 [s \rightarrow s_1 \Rightarrow \exists s'_1 . s' \rightarrow s'_1 \wedge s_1 \simeq_n s'_1]$
 $\wedge \forall s'_1 [s' \rightarrow s'_1 \Rightarrow \exists s_1 . s \rightarrow s_1 \wedge s_1 \simeq_n s'_1]$

Avem $\simeq_{i+1} \subseteq \simeq_i \Rightarrow \exists n . \simeq_n = \simeq_{n+1} = \simeq$ (modele finite)

Raționament compozițional

O aplicație a principiului general “divide and conquer” pentru verificarea unui sistem structurat în componente:

- verificarea de proprietăți locale ale componentelor
- obținerea proprietăților globale din proprietățile locale
- fără a construi un model al întregului sistem (impracticabil)

Raționament compozițional: termen generic pentru reguli de tipul:

$$- M_1 \models f_1 \wedge M_2 \models f_2 \Rightarrow Compose(M_1, M_2) \models LogicOp(f_1, f_2)$$

ex. compoziție paralelă și $LogicOp = \wedge$

$$- M_1 \prec M_2 \Rightarrow CompOp(M_1) \prec CompOp(M_2)$$

ex. $\prec =$ implementare, rafinare; $CompOp(\cdot) = \cdot || M$

$$- M_1 \prec S_1 \wedge M_2 \prec S_2 \Rightarrow Compose(M_1, M_2) \prec Compose(S_1, S_2)$$

Compoziție sincronă, simulare, și fair ACTL

Fie $M = (S, S_0, AP, L, R, F)$ și $M' = (S', S'_0, AP', L', R', F')$.

Definim compoziția paralelă sincronă $M'' = M || M'$:

- $S'' = \{(s, s') \in S \times S' \mid L(s) \cap AP' = L'(s') \cap AP\}$
- $S''_0 = (S_0 \times S'_0) \cap S''$
- $AP'' = AP \cup AP'$
- $L''(s, s') = L(s) \cup L'(s')$
- $R''((s, s')(t, t')) = R(s, t) \wedge R'(s', t')$
- $F'' = \{(P \times S') \cap S'' \mid P \in F\} \cup \{(S \times P') \cap S'' \mid P' \in F'\}$

Folosim logica ACTL cu fairness: pentru orice formulă ACTL f se poate construi un tablou \mathcal{T}_f , și avem $M \models_F f \Leftrightarrow M \preceq_F \mathcal{T}_f$
 \Rightarrow putem raționa uniform cu formule și modele (tablouri)

- (a) Pentru orice M și M' , $M || M' \preceq_F M$.
- (b) Pentru orice M , M' și M'' , $M \preceq_F M' \Rightarrow M || M'' \preceq_F M' || M''$
- (c) Pentru orice M , $M \preceq_F M || M$

Assume-guarantee necircular

Folosim notația $\langle f \rangle M \langle g \rangle$:

Orice sistem care satisface prezumția f și conține M garantează g .
(f, g sunt fie formule, fie modele)

O structură tipică de raționament:

$$\langle true \rangle M \langle A \rangle \wedge \langle A \rangle M' \langle g \rangle \wedge \langle g \rangle M \langle f \rangle \Rightarrow \langle true \rangle M || M' \langle f \rangle$$

Instanțiere în termeni concreți:

M = un transmițător complex

A = un model simplu de transmițător periodic

$\langle true \rangle M \langle A \rangle$: M funcționează la fel ca și A

M' = un receptor

g = “mesajele sunt preluate la timp”

$\langle A \rangle M' \langle g \rangle$ = M' compus cu A preia mesajele la timp

f = “nu avem buffer overflow”

$\langle g \rangle M \langle f \rangle$ = dacă M e într-un sistem care preia mesajele la timp,
nu avem buffer overflow.

\Rightarrow în sistemul $M || M'$ nu apare buffer overflow.

Justificarea raționamentului

- | | |
|---|--|
| (1) $M \preceq_F A$ | ipoteză |
| (2) $M M' \preceq_F A M'$ | (1) și compoziționalitate (a) |
| (3) $A M' \models_F g$ | ipoteză |
| (4) $A M' \preceq_F \mathcal{T}_g$ | (3) și prop. tabloului ACTL |
| (5) $M M' \preceq_F \mathcal{T}_g$ | (2), (4) și tranzitivitatea \preceq_F |
| (6) $M M M' \preceq_F \mathcal{T}_g M$ | (5) și compoziționalitate (b) |
| (7) $\mathcal{T}_g M \models_F f$ | ipoteză |
| (8) $M M M' \models_F f$ | (6), (7) și $\preceq_F \Rightarrow \models_F$ |
| (9) $M \preceq_F M M$ | compoziționalitate (c) |
| (10) $M M' \preceq_F M M M'$ | (9) și compoziționalitate (b) |
| (11) $M M' \models_F f$ | (8), (10) și $\preceq_F \Rightarrow \models_F$ |

Demonstratoare de teoreme pot mecaniza descompunerea în raționamente pe componente și asigura validitatea deducției.

Assume-guaranteee circular

Adeseori, regulile compoziționale sunt insuficient de puternice. Spre exemplu, avem implementări M_i și specificări S_i , $i = 1, 2$. Pentru ca $M_1 || M_2 \prec S_1 || S_2$ ar fi suficient ca $M_1 \prec S_1$ și $M_2 \prec S_2$. Frecvent, relațiile individuale nu sunt însă satisfăcute:

- componentele M_1 și M_2 nu sunt proiectate independent
- fiecare se bazează că e executată în mediul reprezentat de cealaltă

Exemplu de dependențe

Modelăm algoritmul obișnuit de împărțire a două numere, $n \div d$, în baza b , cu două componente:

$M_Q(in : r, d; out : q)$ calculează următoarea cifră din cât: $q = \lfloor r/d \rfloor$

$M_R(in : n, d, q; out : r)$ actualizează restul: $r' = (r - q*d)*b + next_digit(n)$

Dorim ca $M_Q || M_R$ să satisfacă împreună următorii invarianți:

- $S_Q: 0 \leq q < b \wedge q * d \leq r < (q + 1) * d$
- $S_R: 0 \leq r < b * d$

Totuși, individual nu avem nici $M_Q \models S_Q$ și nici $M_R \models S_R$:

funcționarea corectă a fiecărui modul depinde de celălalt

Dar avem $S_Q \Rightarrow M_R \models S_R$ și $S_R \Rightarrow M_Q \models S_Q$.

(un modul funcționează corect în mediul dat de *specificarea* celuilalt)

\Rightarrow Putem deduce de aici că $M_Q || M_R \models S_Q \wedge S_R$?

Reguli circulare de assume-guarantee

Studiate în diverse contexte [Chandi & Misra'81, Abadi & Lamport'93]

Ne referim concret la Reactive Modules [Alur & Henzinger '95]:

– module cu variabile de intrare, de ieșire, relație de tranziție

– relație de dependență $\prec \subseteq (V_{in} \cup V_{out}) \times V_{out}$

– $x \prec y$: y depinde *combi-național* de x ;

altfel, doar valoarea următoare a lui y poate depinde (secvențial) de x

– compoziția paralelă sincronă $M_1 || M_2$ e posibilă

dacă $V_{out}(M_1) \cap V_{out}(M_2) = \emptyset$ și $\prec_{M_1} \cup \prec_{M_2}$ e o relație aciclică.

Definim relația de *rafinare* (implementare) $M \leq M'$ dacă

$V(M') \subseteq V(M)$, $V_{out}(M') \subseteq V_{out}(M)$, $\prec_M \supseteq \prec'_{M'}$, $\mathcal{L}(M)|_{V(M')} \subseteq \mathcal{L}(M')$

(primele 3: dacă P poate funcționa într-un context, atunci și Q)

Reguli circulare de assume-guarantee (cont.)

Pentru module reactive:

$$\frac{M_1 \parallel S_2 \leq S_1 \parallel S_2 \quad S_1 \parallel M_2 \leq S_1 \parallel S_2}{M_1 \parallel M_2 \leq S_1 \parallel S_2}$$

(presupunând că toate compozițiile sunt bine definite)

Avantajul: deși avem de demonstrat două relații, fiecare din ele este mai simplă decât cea originală

- descrierea specificării S_i e mult mai simplă decât implementarea M_i
- nu trebuie compuse cele două implementări (adesea imposibil)

Regulă cu inducție temporală [McMillan'97]

valabilă pentru *invarianti* (safety properties)

- dacă $P_1 \wedge Q_1$ valabile la $0, 1, \dots, t \Rightarrow Q_2$ valabil la $t + 1$
- dacă $P_2 \wedge Q_2$ valabile la $0, 1, \dots, t \Rightarrow Q_1$ valabil la $t + 1$
- atunci pentru orice t , $P_1 \wedge P_2 \Rightarrow Q_1 \wedge Q_2$

Relația dintre compoziționalitate și rafinare

[Henzinger'01] - studiu al teoriei interfețelor

Pt. o relație de rafinare \leq și una de compoziție \parallel , dorim:

Dacă $M_1 \leq S_1$ și $M_2 \leq S_2$, atunci $M_1 \parallel M_2 \leq S_1 \parallel S_2$

În general, insuficient – posibilă incompatibilitate la compoziție
 \Rightarrow două variante:

- Dacă $M_1 \leq S_1$ și $M_2 \leq S_2$, și $M_1 \parallel M_2$ e definit, atunci $S_1 \parallel S_2$ e definit și $M_1 \parallel M_2 \leq S_1 \parallel S_2$
 - formalism axat pe *componente*
 - permite verificarea independentă a componentelor (bottom-up)
- Dacă $M_1 \leq S_1$ și $M_2 \leq S_2$, și $S_1 \parallel S_2$ e definit, atunci $M_1 \parallel M_2$ e definit și $M_1 \parallel M_2 \leq S_1 \parallel S_2$
 - formalism axat pe *interfețe*
 - permite implementarea independentă a interfețelor (top-down)