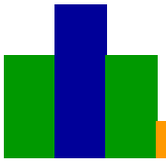
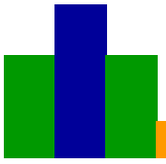


Lecţia 12:

Model checking I: CTL*, sintaxa, semantica, exemple

v1.0 (05.05.07)
Gheorghe Stefanescu — Universitatea Bucureşti

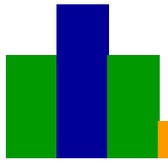
Metode de Dezvoltare Software, Sem.2
Februarie 2007— Iunie 2007



Model checking I

Cuprins:

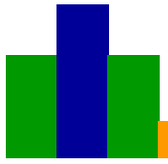
- *Generalităţi despre verificare*
- CTL - computation tree logic
- LTL - logică temporală lineară
- CTL*
- Concluzii, diverse, etc.



Clasificarea metodelor de verificare

Metodele de verificare se pot clasifica după următoarele criterii:

- *demonstraţii vs. modele*:
  —demonstraţii = formule valide = adevărate în toate modelele (dacă avem o teoremă de corectitudine şi completitudine);
  —pe-modele = satisfiabilitatea într-un singur model
- *grad de automatizare* - complet automat, complet manual, ori între ele
- *verificare totală vs. verificare de proprietăţi* - comportament total vs. o singură proprietate
- *domeniul de aplicare* - hardware ori software; programe secvenţiale ori paralele; programe reactive ori care se termină; etc.
- folosite *înainte ori după dezoltatea* aplicaţiei



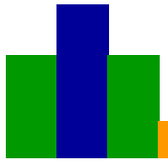
Model checking

Model checking este o metodă de verificare care este:

- bazată pe modele, automată, verifică proprietăţi, folosită mai mult pentru sisteme concurente, reactive (iniţial a fost folosită în faza de post-dezvoltare)

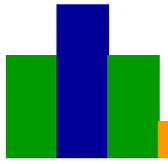
Prin contrast, *verificarea programelor* (prezentată într-un curs ulterior) este:

- bazată pe demonstraţii, asistată de calculator (necesită intervenţia omului), verifică comportamentul total, folosită mai mult pentru programe care se termină şi produc un rezultat



Structuri Kripke

- Logicile clasice propoziţionale şi predicative folosesc un *unic* univers pentru a interpreta formulele.
- In anii ’50 Kripke a introdus nişte modele semantice, numite curent *structuri Kripke*, unde mai multe universuri (locale) sunt posibile.
- Există o relaţie de *accesibilitate* între aceste universuri şi operatori care le conectează permiţând exprimarea diverselor tipuri de *modalităţi*.
- Când se adaugă astfel de operatori se obţn *logici modale*. Dacă ceea ce produce trecerea de la un univers la altul este *timpul*, atunci logicile rezultate se numesc *logici temporale*.



Structuri Kripke pentru programe

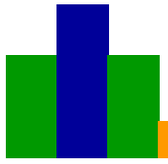
Programele se potrivesc perfect în această filozofie:

- un univers corespunde unei stări
- relaţia de accesibilitate este dată de tranziţia de la o stare la alta datorată efectuării instrucţiunilor
- logica predicativă clasică se foloseşte pentru a specifica relaţii între valorile dintr-o stare a variabilelelor din program;

Ce lipseşte este

- un mecanism care să conecteze universurile stărilor între ele

Aici vom folosi o logică particulară CTL care este un tip de logică temporală.

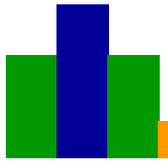


Formalizări ale timpului

Timpul folosit în logicile temporale poate fi

- timp *linear* (un lanţ de instanţe ale timpului) ori timp *ramificat* (mai multe alternative pentru lumile viitoare se folosesc la fiecare pas)
- *discret* ori *continuu*

CTL foloseşte timp *ramificat* şi *discret*.



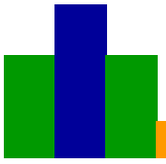
Model checking (in CTL)

Problema la care trebuie dat un răspuns este o întrebare

$$\mathcal{M}, s \models \phi \ ?$$

unde

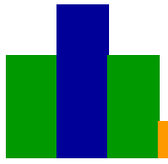
- $\mathcal{M}$ este un model al sistemului analizat şi s este o stare a modelului
- ϕ este o formulă CTL care vrem să fie satisfăcută de sistem



Model checking I

Cuprins:

- Generalităţi despre verificare
- *CTL - computation tree logic*
- LTL - logică temporală lineară
- CTL*
- Concluzii, diverse, etc.

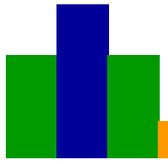


Sintaxa lui CTL

Computation tree logic (pe scurt *CTL*) este o logică definită de următoarea sintaxă BNF:

$$\phi ::= \bot \mid \top \mid p \mid (\neg\phi) \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid (\phi \rightarrow \phi)$$
$$\mid \mathrm{AX}\phi \mid \mathrm{EX}\phi \mid \mathrm{A}[\phi \mathrm{U} \phi] \mid \mathrm{E}[\phi \mathrm{U} \phi] \mid \mathrm{AG}\phi \mid \mathrm{EG}\phi \mid \mathrm{AF}\phi \mid \mathrm{EF}\phi$$

Noii conctori AX, EX, AU, EU, AG, EG, AF, şi EF se numesc *conectori temporali*.



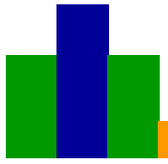
Cuantificare in adancime si in latime

Conectorii temporali folosesc două litere:

- A şi E pentru cuantificarea în lăţime:
  —A: se iau *toate alternativele* din punctul de ramificare (*A*ll)
  —E: *există cel puţin o alternativă* în punctul de ramificare (*E*xists)
- G şi F cuantifică de-a lungul drumurilor:
  —G: *toate stările viitoare* de pe drum (Globally)
  —F: *există cel puţin o stare vitoare* pe drum (*F*uture)

Se mai folosesc doi operatori spre a exprima proprietăţi de-a lungul drumurilor:

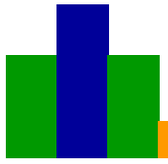
- X se referă la *starea următoare* de pe drum (ne*X*t) - deci avem timp discret
- U - operatorul *Until*



Prioritati

Convenţie:

- conectorii unari (incluzând AX, EX, AG, EG, AF, şi EF) leagă cel mai tare;
- apoi vin $\wedge$ şi $\vee$;
- ultimii vin $\rightarrow$, AU şi EU.



Exemple

Exemple:

1 $\text{EG}\, r$

2 $\text{AG}(q \rightarrow \text{EG}\, r)$

3 $\text{A}[r\ \text{U}\ q]$

4 $\text{EF}\ \text{E}[r\ \text{U}\ q]$

5 $\text{A}[p\ \text{U}\ \text{EF}\ r]$

6 $\text{EF}\ \text{EG}p \rightarrow \text{AF}\ r$

7 $\text{AG}\ \text{AF}\ r$

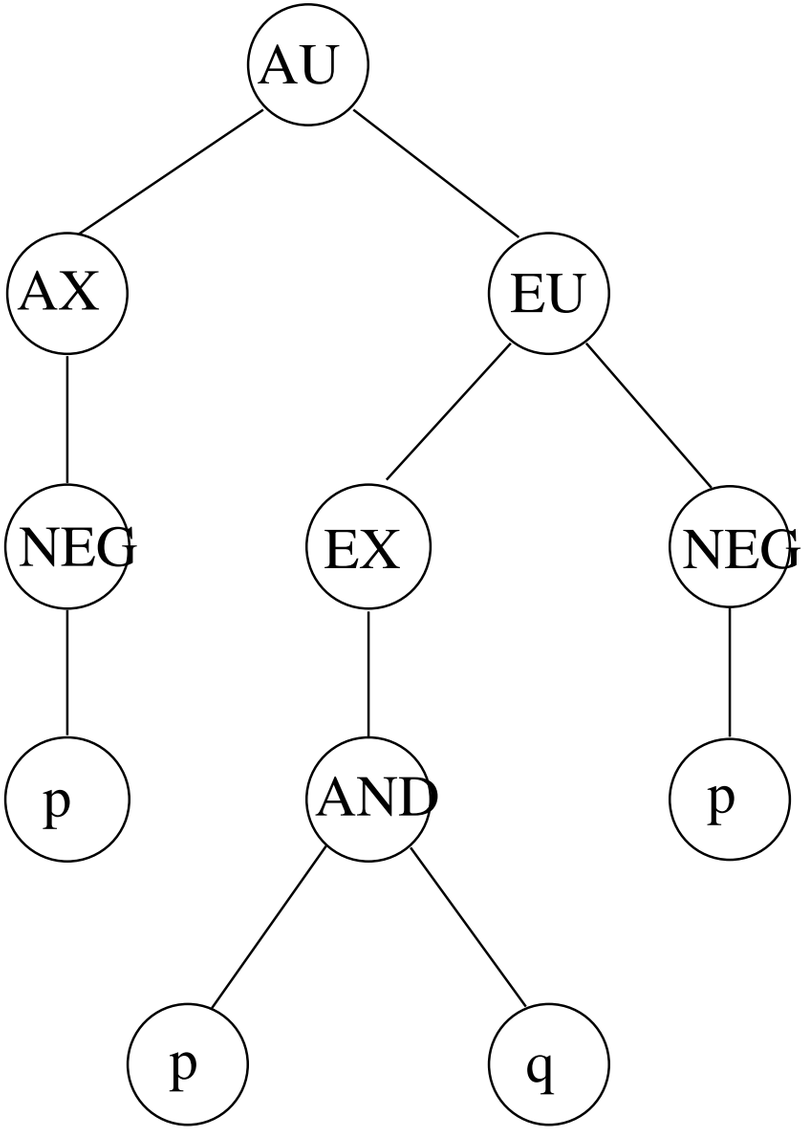
8 $\text{A}[p_1\ \text{U}\ \text{A}[p_2\ \text{U}\ p_3]]$

9 $\text{E}[\text{A}[p_1\ \text{U}\ p_2]\ \text{U}\ p_3]$

10 $\text{AG}(p \rightarrow \text{A}[p\ \text{U}\ (\neg p \wedge \text{A}[\neg p\ \text{U}\ q])])$

Sintaxa este un pic complicată de operatorul binar "until" care foloseşte notaţia 'prefixată' pentru A,E*, şi pe cea 'infixată' pentru* U*. In arborele de derivare folosim notaţiile infixate* AU *şi* EU*. Această sintaxă este folosită în model-checkeri ca SMV.*

Exemple



Arborele de derivare pentru formula

$$\mathrm{A}\left[\mathrm{AX}\ \neg p\ \mathrm{U}\ \mathrm{E}\left[\mathrm{EX}\left(p \wedge q\right)\ \mathrm{U}\ \neg p\right]\right]$$

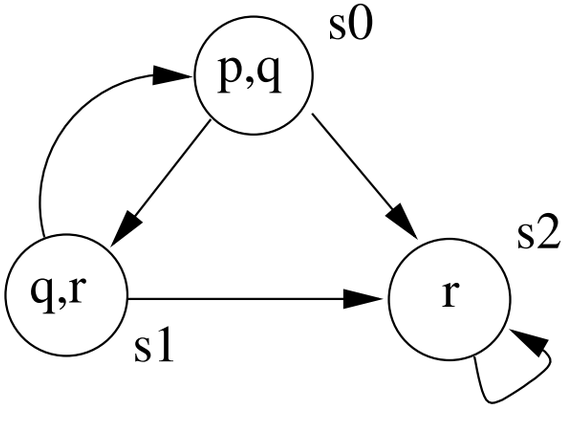
Semantica CTL

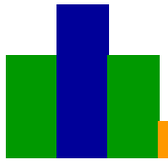
Un *model* $\mathcal{M} = (S, \rightarrow, L)$ pentru CTL constă din

- o mulţime de stări S
- o relaţie binară '$\rightarrow$' pe S astfel ca pentru orice $s \in S$ există $s' \in S$ cu $s \rightarrow s'$ (sistemul de tranziţii *nu se termină*)
- o funcţie de etichetare $L : S \rightarrow \mathcal{P}(Atoms)$

Intuiţia este că L indică atomii care sunt adevăraţi într-o stare şi $\rightarrow$ descrie evoluţia sistemului dintr-o stare în alta

Descriere grafică:
$S = \{s0, s1, s2\}$
$\rightarrow = \{(s0, s1), (s0, s2), (s1, s0), (s1, s2), (s2, s2)\}$
$L(s0) = \{p, q\}, L(s1) = \{q, r\}, L(s2) = \{r\}$





Relaţia de satisfacere

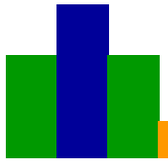
Fie $\mathcal{M} = (S, \rightarrow, L)$ un model pentru CTL, $s \in S$, şi ϕ o formulă CTL.

Relaţia de satisfacere

$$\mathcal{M}, s \models \phi$$

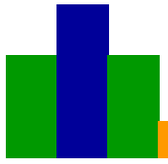
se defineşte inductiv astfel:

1. $\mathcal{M}, s \models \top$ şi $\mathcal{M}, s \not\models \bot$ pentru toţi $s \in S$;
2. $\mathcal{M}, s \models p$ dnd $p \in L(s)$;
3. $\mathcal{M}, s \models \neg\phi$ dnd $\mathcal{M}, s \not\models \phi$;
4. $\mathcal{M}, s \models \phi \wedge \psi$ dnd $\mathcal{M}, s \models \phi$ şi $\mathcal{M}, s \models \psi$;
5. $\mathcal{M}, s \models \phi \vee \psi$ dnd $\mathcal{M}, s \models \phi$ ori $\mathcal{M}, s \models \psi$;
6. $\mathcal{M}, s \models \phi \rightarrow \psi$ dnd $\mathcal{M}, s \not\models \phi$ ori $\mathcal{M}, s \models \psi$;
7. $\mathcal{M}, s \models \mathrm{AX}\,\phi$ dnd pentru toţi s' cu $s \rightarrow s'$ avem $\mathcal{M}, s' \models \phi$;
8. $\mathcal{M}, s \models \mathrm{EX}\,\phi$ dnd pentru un s' cu $s \rightarrow s'$ avem $\mathcal{M}, s' \models \phi$;



..Relaţia de satisfacere

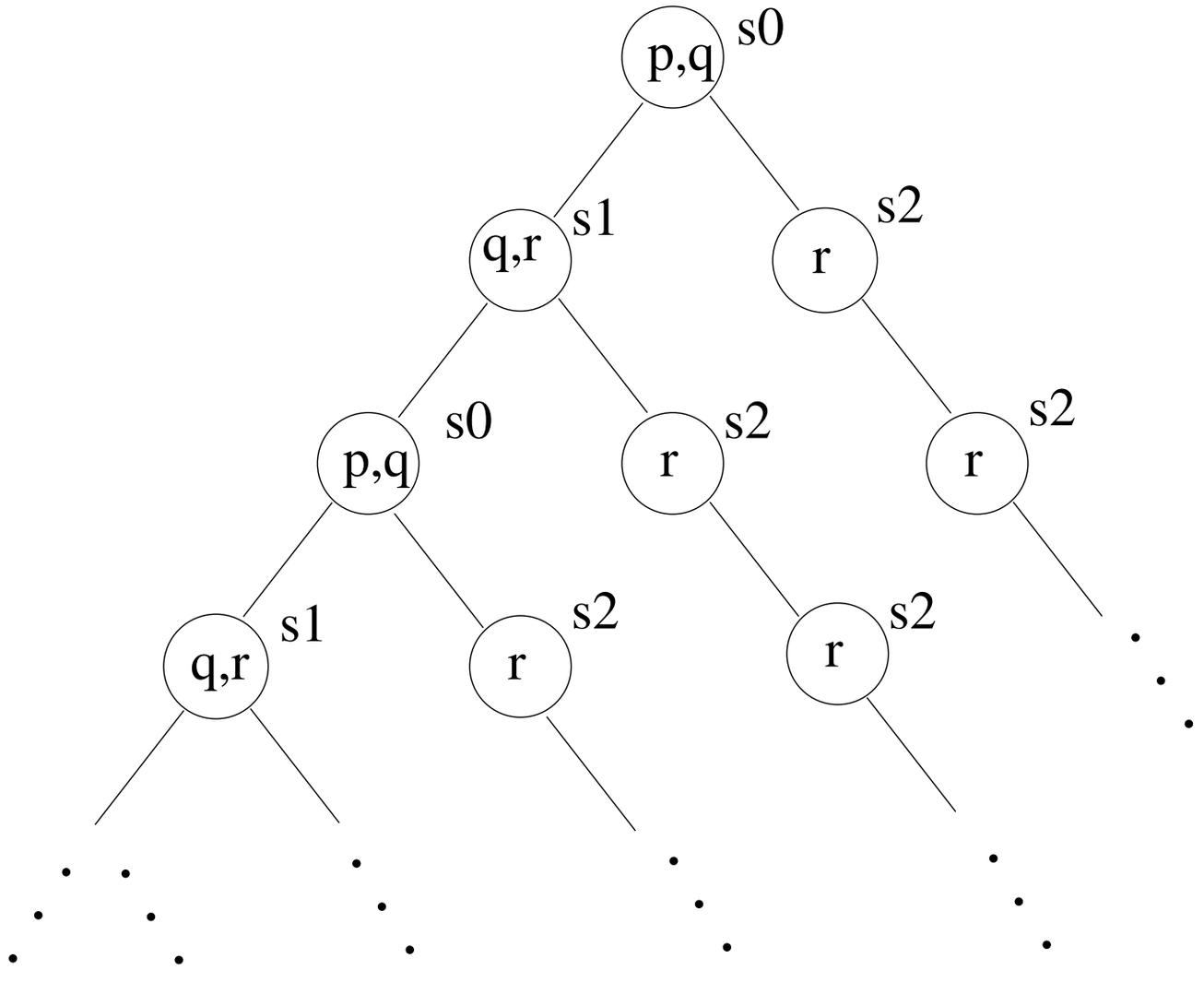
9. $\mathcal{M}, s \models \mathrm{AG}\ \phi$ dnd pentru toate drumurile $s = s_0 \rightarrow s_1 \rightarrow \ldots$ avem $\mathcal{M}, s_i \models \phi$, pentru toţi i;
10. $\mathcal{M}, s \models \mathrm{EG}\ \phi$ dnd există un drum $s = s_0 \rightarrow s_1 \rightarrow \ldots$ cu $\mathcal{M}, s_i \models \phi$, pentru toţi i;
11. $\mathcal{M}, s \models \mathrm{AF}\ \phi$ dnd pentru toate drumurile $s = s_0 \rightarrow s_1 \rightarrow \ldots$ avem $\mathcal{M}, s_i \models \phi$, pentru un i;
12. $\mathcal{M}, s \models \mathrm{EF}\ \phi$ dnd există un drum $s = s_0 \rightarrow s_1 \rightarrow \ldots$ cu $\mathcal{M}, s_i \models \phi$, pentru un i;
13. $\mathcal{M}, s \models \mathrm{A}[\phi\ \mathrm{U}\ \psi]$ dnd pentru toate drumurile $s = s_0 \rightarrow s_1 \rightarrow \ldots$ există un i cu $\mathcal{M}, s_i \models \psi$ şi $\mathcal{M}, s_j \models \phi$ pentru toţi $j < i$;
14. $\mathcal{M}, s \models \mathrm{E}[\phi\ \mathrm{U}\ \psi]$ dnd există un drum $s = s_0 \rightarrow s_1 \rightarrow \ldots$ şi un i cu $\mathcal{M}, s_i \models \psi$ şi $\mathcal{M}, s_j \models \phi$ pentru toţi $j < i$; $\triangle$

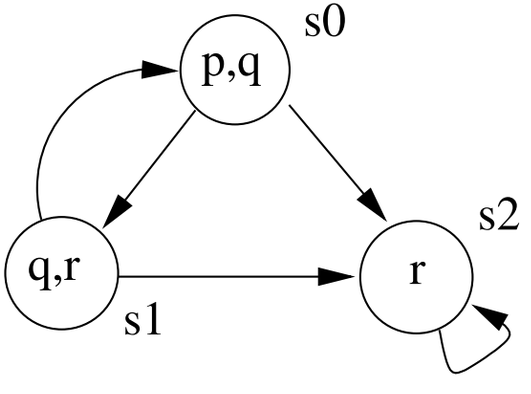


Comentrii

- Să notăm că "viitorul" este închiderea reflexiv-transitivă $\rightarrow^*$ a relaţiei de accesibilitate directă $\rightarrow$.
- Deci:
  — *viitorul conţine prezentul* şi
  — *un viitor al viitorului lui t este un viitor al lui t*.
- Desfăşurând graful unui model CTL obţinem un *arbore infinit*, de unde şi numele *'computation tree logic'*.

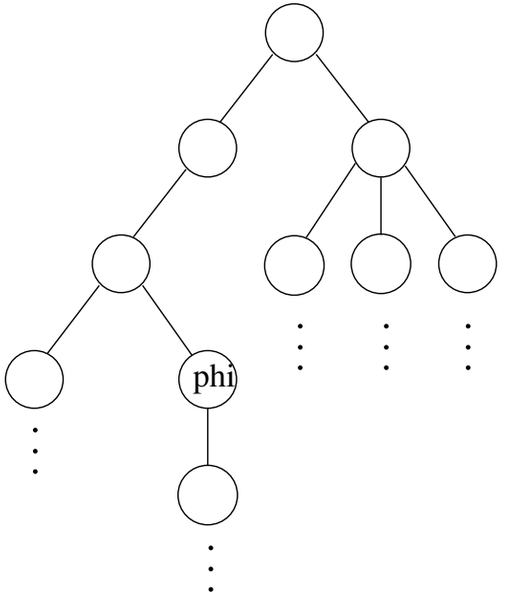
Desfasurare



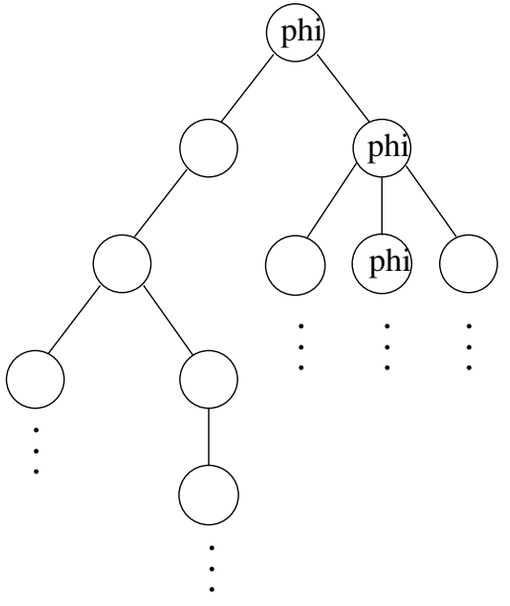


Un graf CTL şi desfăşurarea sa.

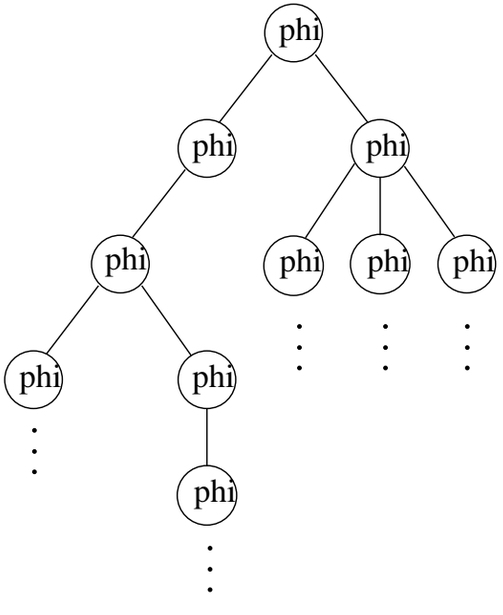
Semnificatia operatorilor EF,EG,AG, AF



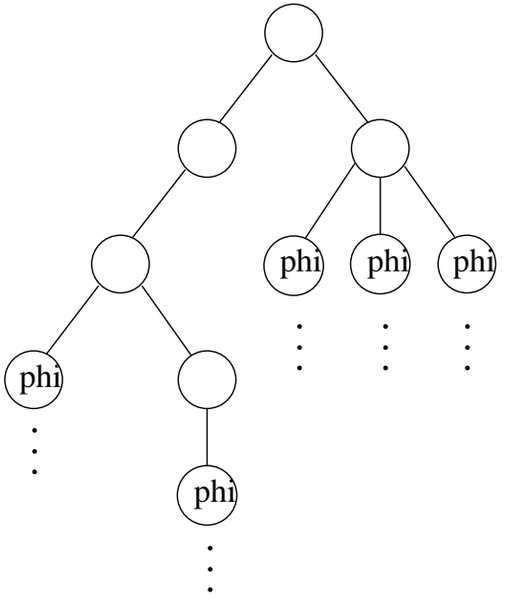
EF ϕ:



EG ϕ:

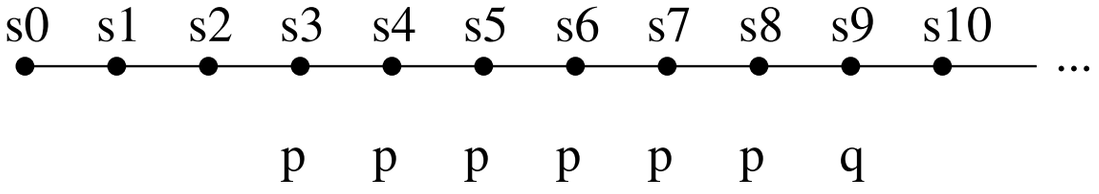


AG ϕ:



AF ϕ:

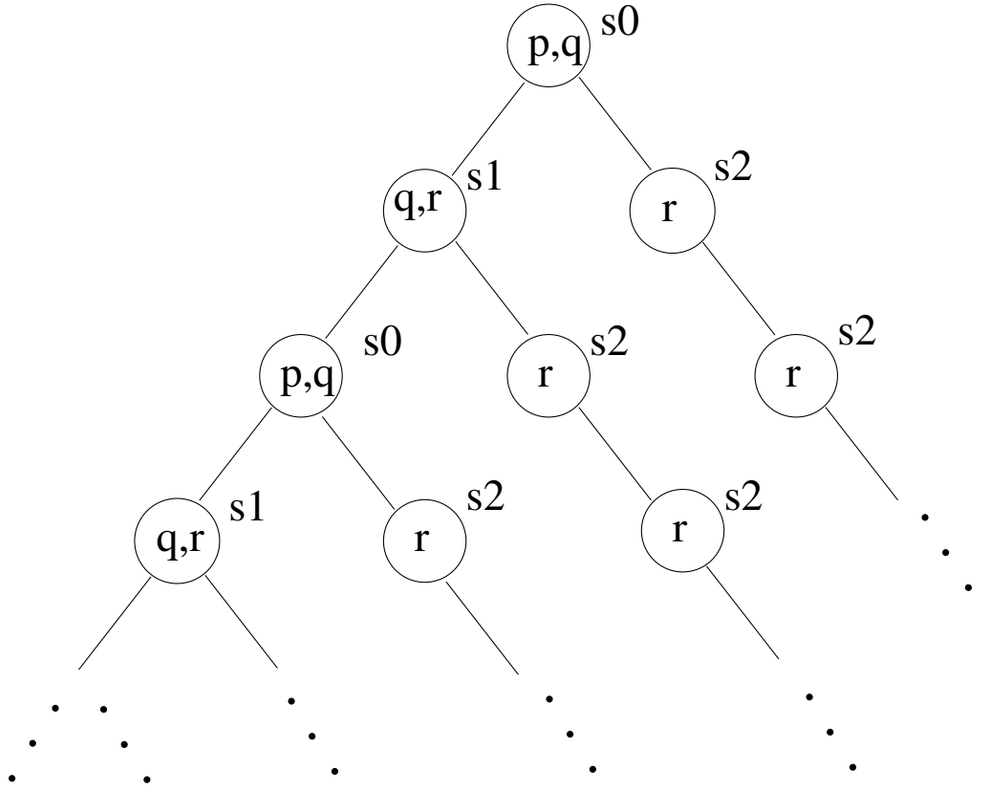
Until



Until p U q într-un model cu timp linear [ori pe un drum din CTL].

Formula p U q este adevărată în $s3$, dar nu în $s0$ (p este validă doar în stările unde este afişată)

Exemple



$\mathcal{M}, s0 \models p \wedge q$ - Da

$\mathcal{M}, s0 \models \neg r$ - Da

$\mathcal{M}, s0 \models \top$ - Da

$\mathcal{M}, s0 \models \mathrm{EX}\,(q \wedge r)$ - Da

$\mathcal{M}, s0 \models \neg\mathrm{AX}\,(q \wedge r)$ - Da

$\mathcal{M}, s0 \models \neg\mathrm{EF}\,(p \wedge r)$ - Da

$\mathcal{M}, s0 \models \mathrm{EG}\ r$ - Nu

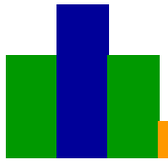
$\mathcal{M}, s2 \models \mathrm{EG}\ r$ - Da

$\mathcal{M}, s2 \models \mathrm{AG}\ r$ - Da

$\mathcal{M}, s0 \models \mathrm{AF}\ r$ - Da

$\mathcal{M}, s0 \models \mathrm{E}[(p \wedge q)\ \mathrm{U}\ r]$ - Da

$\mathcal{M}, s0 \models \mathrm{A}[p\ \mathrm{U}\ r]$ - Da



Specificatii practice

Exemple de proprietăţi descrie informal şi prin formule CTL:

- este posibil să ajungem într-o stare unde un proces a început (*started*), dar nu este încă gata (*ready*):

$$\mathrm{EF}(started \wedge \neg ready)$$

- pentru orice stare, dacă a apărut o cerere (*request*), atunci ea va fi ulterior confirmată (*acknowledged*):

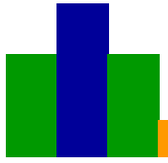
$$\mathrm{AG}(request \rightarrow \mathrm{AF}\ acknowledged)$$

- un proces este disponibil (*enabled*) de o infinitate de ori pe orice drum:

$$\mathrm{AG}(\mathrm{AF}\ enabled)$$

- orice s-ar întâmpla, procesul va fi permanent blocat (*deadlocked*):

$$\mathrm{AF}(\mathrm{AG}\ deadlocked)$$



..Specificatii practice

- din orice stare este posibil să revenim într-o stare dată *restart*:

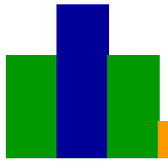
$$\mathrm{AG}(\mathrm{EF}\ restart)$$

- un lift care se deplasează în sus la etajul 2 nu-şi schimbă direcţia dacă pasagerii merg la etajul 5:

$$\mathrm{AG}(floor = 2 \wedge direction = up \wedge ButtonPressed5 \\ \rightarrow \mathrm{A}[direction = up\ \mathrm{U}\ floor = 5])$$

- un lift poate rămâne inactiv la etajul 3 cu uşile închise

$$\mathrm{AG}(floor = 3 \wedge idle \wedge door = closed \\ \rightarrow \mathrm{EG}(floor = 3 \wedge idle \wedge door = closed))$$



Echivalente utile

Definiţii: Două formule CTL ϕ şi ψ sunt *semantic echivalente*, notat $\phi \equiv \psi$, dacă orice stare din orice model care satisface o formulă o satisface şi pe cealaltă.

Echivalenţe utile:

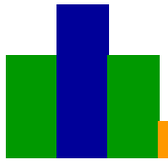
1 $\neg \mathrm{AF}\ \phi \equiv \mathrm{EG}\ \neg\phi$

2 $\neg \mathrm{EF}\ \phi \equiv \mathrm{AG}\ \neg\phi$

3 $\neg \mathrm{AX}\ \phi \equiv \mathrm{EX}\ \neg\phi$

4 $\mathrm{AF}\ \phi \equiv \mathrm{A}[\top\ \mathrm{U}\ \phi]$

5 $\mathrm{EF}\ \phi \equiv\ \mathrm{E}[\top\ \mathrm{U}\ \phi]$

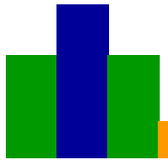


Mulţimi adecvate de operatori temporali

Corolar (mulţimi adecvate de operatori temporali):

Următoarele submulţimi de operatori sunt mulţimi adecvate pentru CTL (anume, orice formula CTL poate fi transformată într-una echivalentă care foloseşte doar acei operatori):

1. AU, EU *şi* EX;
2. EG, EU *şi* EX;
   (indicaţie de demonstraţie: $\mathrm{A}[\phi \mathrm{U} \psi] \equiv \neg(\mathrm{E}[\neg\psi \mathrm{U} (\neg\phi \wedge \neg\psi)] \vee \mathrm{EG}\ \neg\psi))$
3. AG, AU *and* AX;
4. AF, EU *and* AX



..Echivalente utile (definiţii cu puncte-fixe)

Alte echivalenţe utile, folosite şi ca definiţii prin puncte-fixe

$$6 \quad \mathrm{AG}\ \phi \equiv \phi \wedge \mathrm{AX}\ \mathrm{AG}\ \phi$$

$$7 \quad \mathrm{EG}\ \phi \equiv \phi \wedge \mathrm{EX}\ \mathrm{EG}\ \phi$$

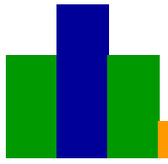
$$8 \quad \mathrm{AF}\ \phi \equiv \phi \vee \mathrm{AX}\ \mathrm{AF}\ \phi$$

$$9 \quad \mathrm{EF}\ \phi \equiv \phi \vee \mathrm{EX}\ \mathrm{EF}\ \phi$$

$$10 \quad \mathrm{A}[\phi\ \mathrm{U}\ \psi] \equiv \psi \vee (\phi \wedge \mathrm{AX}\ \mathrm{A}[\phi\ \mathrm{U}\ \psi])$$

$$11 \quad \mathrm{E}[\phi\ \mathrm{U}\ \psi] \equiv \psi \vee (\phi \wedge \mathrm{EX}\ \mathrm{E}[\phi\ \mathrm{U}\ \psi])$$

Un mecanism pentru rezolvarea de ecuaţii cu "puncte-fixe" ca $Y = \phi \wedge \mathrm{AX}\ Y$ *şi operatorii next (*AX *şi* EX*) sunt suficiente pentru a defini toţi operatorii temporali.*



Excluderea mutuala

Scop: *de a dezvolta protocoale pentru a accesa secţiuni critice astfel ca cel mult un proces să poate fi în secţiunea critică în orice moment de timp.*

O colecţie de proprietăţi pe care ar trebui să le satisfacă un astfel de protocol sunt:

Safety: protocolul permite numai unui proces să fie în secţiunea lui crtică în orice moment de timp

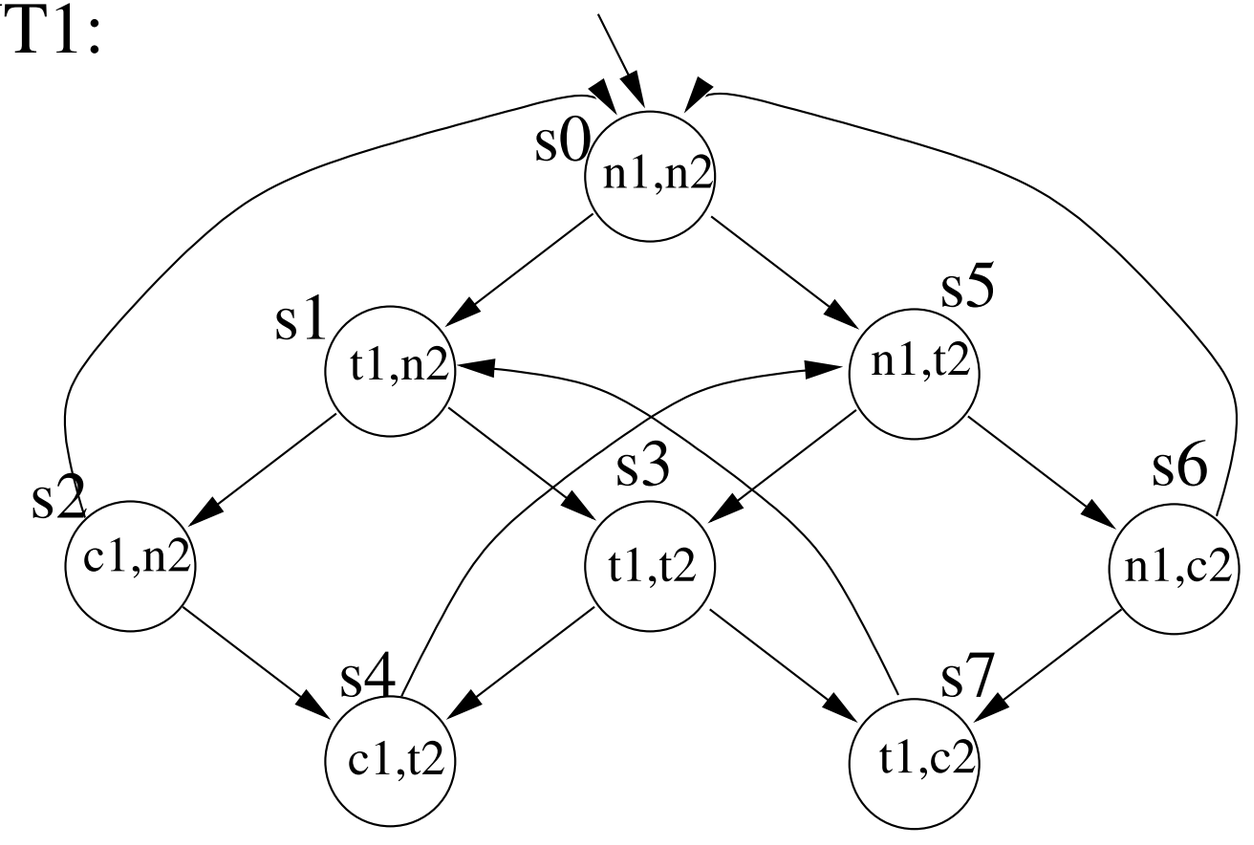
Liveness: de fiecare dată când un proces vrea să intre în secţiunea sa critică i se va permite să intre cândva

Non-blocking: un proces poate cere oricând să intre în secţiunea sa critică

No strict sequencing: procesele nu trebuie să intre alternant în secţiunile lor critice

..Excluderea mutuala

Un model simplu MUT1:



Sistemul are două procese $P1$ şi $P2$, fiecare făcând o buclă $n \rightarrow t \rightarrow c \rightarrow \ldots$ (necritic $\rightarrow$ doreşte-critic $\rightarrow$ critic $\rightarrow \ldots$).
Comportamentul sistemului este produsul prin *întrepătrundere* al comportamentelor lui $P1$ şi $P2$, dar cu starea $(c1, c2)$ eliminată.

..Excluderea mutuala

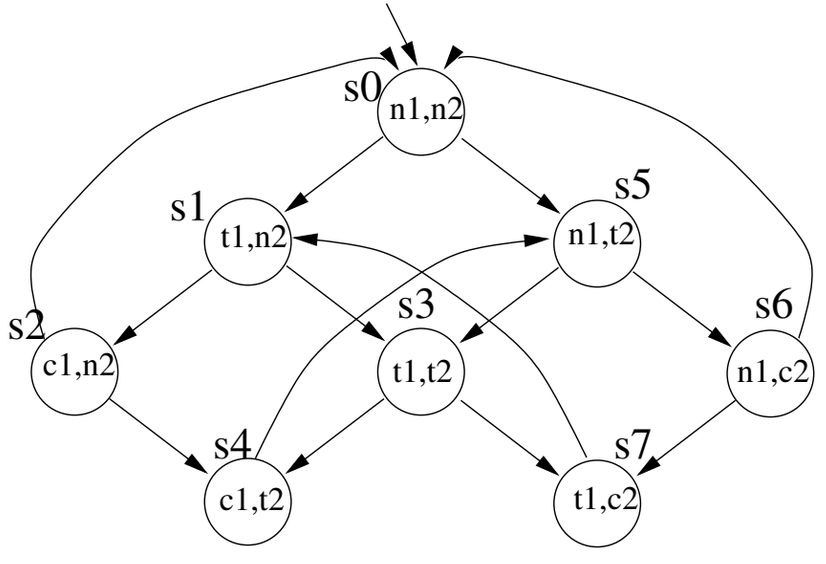
Safety: $\phi_1 =_{def} \text{AG}\,\neg(c1 \wedge c2)$ - satisfăcută în orice stare

Liveness: $\phi_2 =_{def} \text{AG}(t1 \rightarrow \text{AF}\,c1)$ - nesatisfăcută în starea inițială $s0$; e.g., $s1$ este accesibilă, $t1$ este adevărată, dar există un drum $s1 \rightarrow s3 \rightarrow s7 \rightarrow s1 \rightarrow \ldots$ unde $c1$ este mereu falsă

Non-blocking: $\phi_3 =_{def} \text{AG}(n1 \rightarrow \text{EX}\,t1)$ - satisfăcută în orice stare

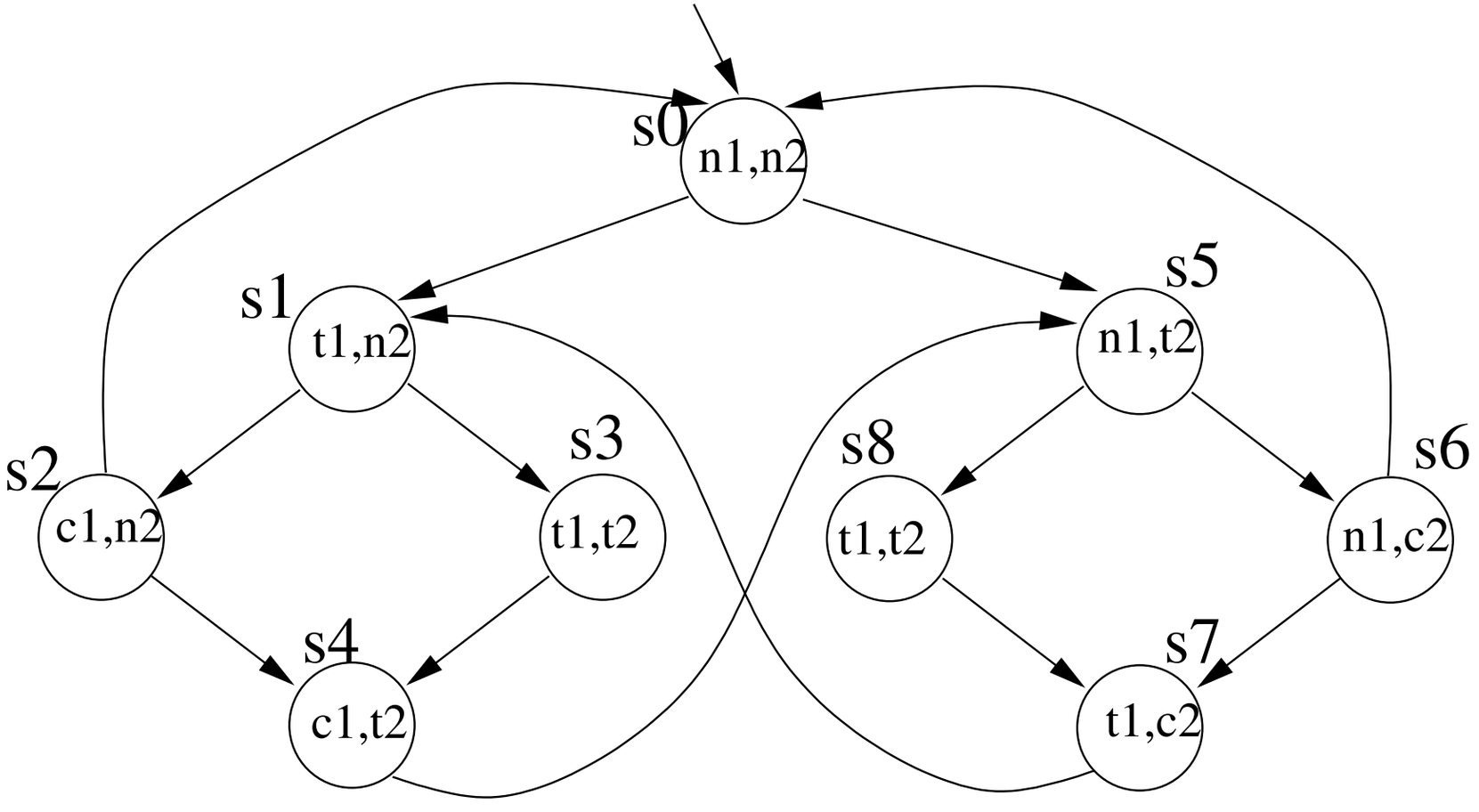
No strict sequencing:

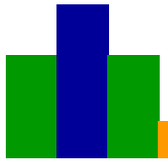
$\phi_4 =_{def} \text{EF}(c1 \wedge \text{E}[c1\ \text{U}\ (\neg c1 \wedge \text{E}[\neg c2\ \text{U}\ c1])])$ - satisfăcută în orice stare



Mutual exclusion (4)

Al doilea model MUT2:

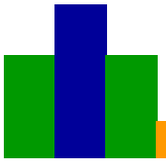




Mutual exclusion (5)

- Modelul se obţine despicând starea $s3$ din MUT1 în două stări diferite $s3$ şi $s9$.
- Prin această procedură, putem identifica procesul care a cerut primul să intre în secţiunea sa critică: dacă $P1$ a fost primul, rezultatul este starea $s3$, altfel $s9$.

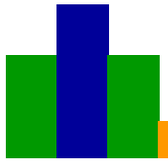
Fapt: *Toate cele patru proprietăţi* $\phi_1 - \phi_4$ *sunt valide în MUT2.*



Model checking I

Cuprins:

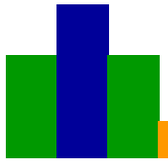
- Generalităţi despre verificare
- CTL - computation tree logic
- *LTL - logică temporală lineară*
- CTL*
- Concluzii, diverse, etc.



Logici pentru sisteme reactive

Există multe limbaje de specificare pentru sisteme reactive (anume, sisteme care merg non-stop, reacţionând la cereri externe), e.g.:

- expresii regulate
- “duration calculus”
- “modal mu-calculus”
- LTL (logică temporală lineară)
- CTL (“computation tree logic”)
- CTL*
- …



LTL

Logica temporală lineară LTL este strâns legată de CTL.
Sintaxa ei este următoarea:

$$\phi ::= \top \mid p \mid (\neg\phi) \mid (\phi \wedge \phi) \mid (\phi \ \mathrm{U} \ \phi) \mid (\mathrm{G}\ \phi) \mid (\mathrm{F}\ \phi) \mid (\mathrm{X}\ \phi)$$

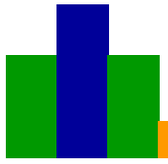
Exemple:

$\mathrm{GF}\ p$

$\mathrm{FG}\ p$

$\mathrm{G}(p \vee \mathrm{X}\ p)$

$\mathrm{G}\ p \rightarrow \mathrm{F}\ q$

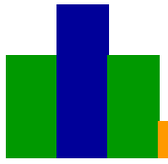


LTL vs. CTL

Comentarii:

- o formulă LTL este evaluată pe un drum, ori pe o mulţime de drumuri; de aceea quantificările din CTL E (există o ramură) şi A (toate ramurile) dispar aici [din acest punct de vedere, LTL apare mai slabă dacât CTL]
- totuşi, LTL permite amestecarea operatorilor modali într-un mod care nu este posibil în CTL, e.g., GFϕ [din acest punct de vedere, LTL apare mai expresivă decât CTL]

Aparent, LTL este mai permisivă şi pentru că permite combinaţii booleene de drumuri, dar astfel de combinaţii pot fi făcute şi în CTL.

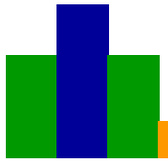


Semantica LTL

Fie $\mathcal{M} = (S, \rightarrow, L)$ un model (tip CTL) şi $\pi = s_0 \rightarrow \ldots$ un drum; notăm cu π^i drumul $s_i \rightarrow s_{i+1} \rightarrow \ldots$

Relaţia de satisfacere $\pi \models \phi$ se defineşte inductiv astfel:

1. $\pi \models \top$
2. $\pi \models p$ dnd $p \in L(s_0)$
3. $\pi \models \neg\phi$ dnd $\pi \not\models \phi$
4. $\pi \models \phi_1 \wedge \phi_2$ dnd $\pi \models \phi_1$ şi $\pi \models \phi_2$
5. $\pi \models \mathrm{X}\ \phi_1$ dnd $\pi^1 \models \phi_1$
6. $\pi \models \mathrm{G}\ \phi_1$ dnd pentru toţi i, $\pi^i \models \phi_1$
7. $\pi \models \mathrm{F}\ \phi_1$ dnd pentru un i, $\pi^i \models \phi_1$
8. $\pi \models \phi_1\ \mathrm{U}\ \phi_2$ dnd există i cu $\pi^i \models \phi_2$ şi pentru toţi $j = 0, \ldots, i-1$ avem $\pi^j \models \phi_1$



Echivalenţe semantice

- Două formule LTL ϕ şi ψ sunt *semantic echivalente*, scris $\phi \equiv \psi$, dacă, pentru orice model, ele sunt adevărate pentru aceeaşi mulţime de drumuri.
- O formulă LTL ϕ este *satisfăcută în starea s* a unui model $\mathcal{M}$ dacă ϕ este satisfăcută în toate stările care încep cu *s*.

Exemple

$$\mathrm{G}\,\phi \equiv \neg \mathrm{F}\,\neg\phi$$
$$\mathrm{F}(\phi \vee \psi) \equiv \mathrm{F}\,\phi \vee \mathrm{F}\,\psi$$
$$\mathrm{G}(\phi \wedge \psi) \equiv \mathrm{G}\,\phi \wedge \mathrm{G}\,\psi$$

Notă: In formalismul CTL, o formulă LTL ϕ se identifică cu $\mathrm{A}[\phi]$ (pentru satisfiabilitate, se verifică toate drumurile)

Identitatea Until

Pentru orice formule LTL ϕ şi ψ

$$\neg(\phi \, \mathrm{U} \, \psi) \equiv \neg\psi \, \mathrm{U}(\neg\phi \wedge \neg\psi) \vee \mathrm{G} \, \neg\psi$$

Dem:

$\neg(\phi \, \mathrm{U} \, \psi)$ este adevărată

dnd

(1) ori ψ este mereu falsă

(2) ori ϕ este falsă înainte ca ψ să devină adevărată

dnd

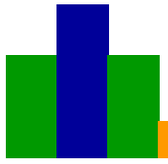
(1) ori $\mathrm{G} \, \neg\psi$ este adevărată

(2) ori $\neg\psi \, \mathrm{U}(\neg\phi \wedge \neg\psi)$ este adevărată

dnd

$\neg\psi \, \mathrm{U}(\neg\phi \wedge \neg\psi) \vee \mathrm{G} \, \neg\psi$ este adevărată

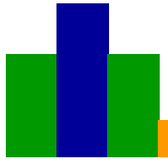
O formă echivalentă este: $\phi \, \mathrm{U} \, \psi \equiv \neg(\neg\psi \, \mathrm{U}(\neg\phi \wedge \neg\psi)) \wedge \mathrm{F} \, \psi$



Model checking I

Cuprins:

- Generalităţi despre verificare
- CTL - computation tree logic
- LTL - logică temporală lineară
- *CTL**
- Concluzii, diverse, etc.



CTL*

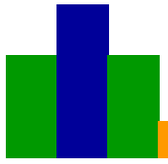
Sintaxa lui CTL* defineşte *formule de stări* şi *formule de drumuri* utilizând următoarele definiţii recursive:

- formule de stări (se evaluează în stări)

$$\phi ::= \top \mid p \mid (\neg\phi) \mid (\phi \wedge \phi) \mid \mathrm{A}[\alpha] \mid \mathrm{E}[\alpha]$$

- formule de drumuri (se evaluează pe drumuri)

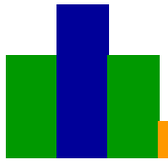
$$\alpha ::= \phi \mid (\neg\alpha) \mid (\alpha \wedge \alpha) \mid (\mathrm{X}\ \alpha) \mid (\mathrm{G}\ \alpha) \mid (\mathrm{F}\ \alpha) \mid (\alpha\ \mathrm{U}\ \alpha)$$



CTL* - exemple

Exemple:

- $\mathrm{A}[(p \ \mathrm{U} \ r) \vee (q \ \mathrm{U} \ r)]$: pe toate drumurile, sau p este true până avem r, ori q este true până avem r
  - nu este echivalentă cu $\mathrm{A}[(p \vee q) \ \mathrm{U} \ r]$
- $\mathrm{A}[\mathrm{X} \ p \vee \mathrm{XX} \ p]$: p este true în starea următoare, ori în a doua stare următoare
  - nu este echivalentă cu $\mathrm{AX} \ p \ \vee \mathrm{AX} \ \mathrm{AX} \ p$
- $\mathrm{E}[\mathrm{GF} \ p]$: există un drum de-a lungul căruia p este true de o infinitate de ori
  - nu este echivalentă $\mathrm{EG} \ \mathrm{EF} \ p$



Semantica CTL*

Fie $\mathcal{M} = (S, \rightarrow, L)$ un model.

- Dacă ϕ este o formulă de stări, atunci

$$\mathcal{M}, s \models \phi$$

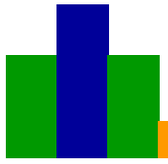
înseamnă că ϕ este true în starea s.

- Dacă α este o formulă de drumuri, atunci

$$\mathcal{M}, \pi \models \alpha$$

înseamnă că α este true pe drumul π.

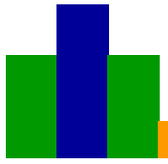
Aceste relaţii se definesc inductiv astfel:



..Semantica CTL*

1. $\mathcal{M}, s \models p$ dnd $p \in L(s)$
2. $\mathcal{M}, s \models \neg\phi$ dnd $\mathcal{M}, s \not\models \phi$
3. $\mathcal{M}, s \models \phi_1 \wedge \phi_2$ dnd $\mathcal{M}, s \models \phi_1$ şi $\mathcal{M}, s \models \phi_2$
4. $\mathcal{M}, s \models \mathrm{A}[\alpha]$ dnd pentru orice drum π care începe din s, $\mathcal{M}, \pi \models \alpha$
5. $\mathcal{M}, s \models \mathrm{E}[\alpha]$ dnd există un drum π care începe din s astfel încât $\mathcal{M}, \pi \models \alpha$

6. $\mathcal{M}, \pi \models \phi$ dnd s este prima stare din π şi $\mathcal{M}, s \models \phi$
7. $\mathcal{M}, \pi \models \alpha_1 \wedge \alpha_2$ dnd $\mathcal{M}, \pi \models \alpha_1$ şi $\mathcal{M}, \pi \models \alpha_2$
8. $\mathcal{M}, \pi \models \mathrm{X}\alpha$ dnd $\mathcal{M}, \pi^1 \models \alpha$
9. $\mathcal{M}, \pi \models \mathrm{G}\alpha$ dnd pentru toţi $k \geq 0$, $\mathcal{M}, \pi^k \models \alpha$
10. $\mathcal{M}, \pi \models \mathrm{F}\alpha$ dnd există un $k \geq 0$ cu $\mathcal{M}, \pi^k \models \alpha$
11. $\mathcal{M}, \pi \models \alpha_1 \ \mathrm{U} \ \alpha_2$ dnd există un $k \geq 0$ cu $\mathcal{M}, \pi^k \models \alpha_2$ şi pentru toţi $0 \leq j < k$, $\mathcal{M}, \pi^j \models \alpha_1$



LTL şi CTL ca subseturi din CTL*

- CTL este cazul particular al lui CTL* în care formulele de drumuri se restrâng la

$$\alpha ::= (\mathrm{X}\ \phi) \mid (\mathrm{G}\ \phi) \mid (\mathrm{F}\ \phi) \mid (\phi\ \mathrm{U}\ \phi)$$

  unde ϕ este o formulă de stare.

  (cu alte cuvinte, orice operator temporal este precedat direct de un cuantificator de drumuri A ori E conducând la operatorii CTL formaţi din ‘două litere’: AG, etc.)

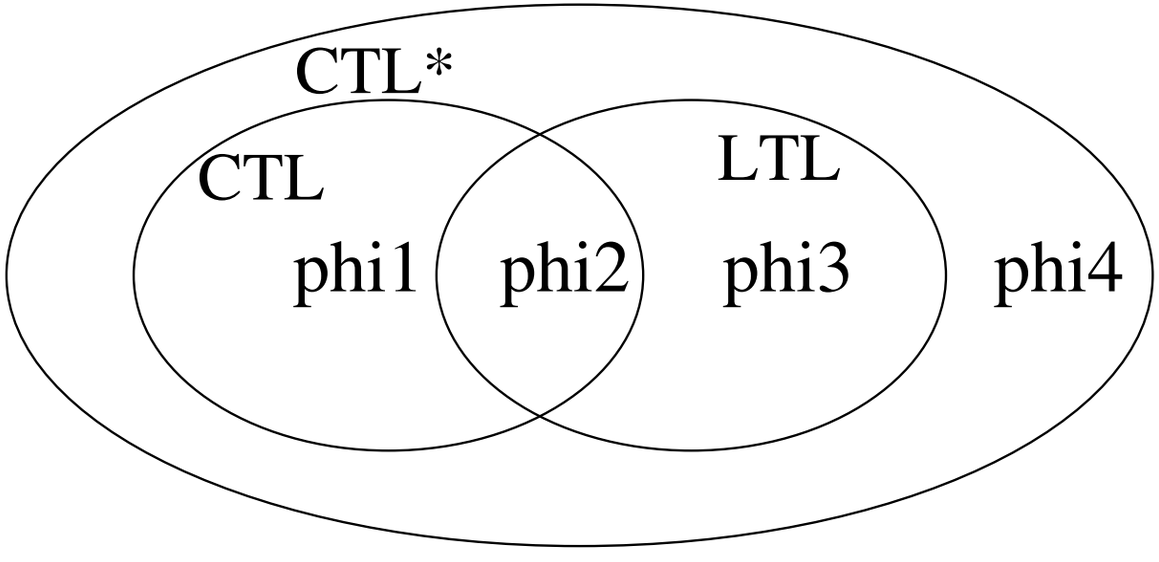
- o formulă LTL α se identifică cu formula CTL*

$$\mathrm{A}[\alpha]$$

  (semantic, se consideră toate drumurile pentru a testa formula)

CTL, LTL, şi CTL*

- LTL şi CTL sunt incomparabile relativ la puterea de expresie
- extensia comună CTL* a fost extensiv studiată



Exemple:

$\phi_1 = \mathrm{AG}(\mathrm{EF}\ p)$

$\phi_3 = \mathrm{A}[\mathrm{FG}\ p]$

$\phi_4 = \phi_1 \vee \phi_3$

Ori:

$\phi_3 = \mathrm{A}[\mathrm{GF}\ p \rightarrow \mathrm{F}\ q]$

$\phi_4 = \mathrm{E}[\mathrm{GF}\ p]$

..CTL, LTL, şi CTL*

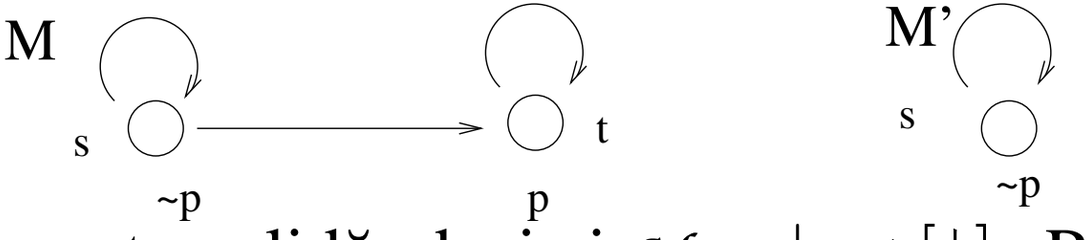
Formula CTL

$$\texttt{phi1} = \text{AG EF } p$$

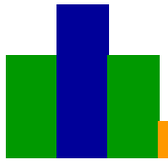
descrie proprietatea

"orunde am merge, putem reveni la o stare în care p este true"

Această proprietate *nu se poate exprima în LTL*: Intr-adevăr, dacă există o formulă LTL ϕ cu $\text{A}[\phi] \equiv \text{AG EF } p$, atunci relativ la diagramele



$\mathcal{M}, s \models \text{AG EF } p$ este validă, deci şi $\mathcal{M}, s \models \text{A}[\phi]$. Pe de altă parte, drumurile din diagrama $\mathcal{M}'$ sunt un subset al drumurilor din diagrama $\mathcal{M}$, deci $\mathcal{M}', s \models \text{A}[\phi]$, dar acest lucru nu este adevărat.



...(CTL, LTL, şi CTL*)

Formula LTL

$$\texttt{phi3} = \mathrm{A}[\mathrm{GF}\ p \rightarrow \mathrm{F}\ q]$$

descrie proprietatea

"dacă există o infinitate de p pe un drum, atunci există şi o prezenţă a lui q"

Această proprietate *nu poate fi exprimată în CTL*.

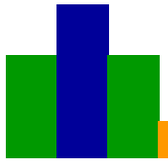
Formula CTL*

$$\texttt{phi4} = \mathrm{E}[\mathrm{GF}\ p]$$

descrie proprietatea

"există un drum cu o infinitate de p"

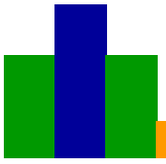
care *nu poate fi exprimată nici în CTL nici în LTL*.



Combinatii booleene de drumuri in CTL

- Combinatii booleene de drumuri in CTL:
  - $\mathrm{E}[\mathrm{F}p \wedge \mathrm{F}q] \equiv \mathrm{EF}[p \wedge \mathrm{EF}q] \vee \mathrm{EF}[q \wedge \mathrm{EF}p]$
  - $\mathrm{E}[(p_1 \mathrm{U} q_1) \wedge (p_2 \mathrm{U} q_2)] \equiv \mathrm{E}[(p_1 \wedge p_2) \mathrm{U} (q_1 \wedge \mathrm{E}[p_2 \mathrm{U} q_2])] \vee \mathrm{E}[(p_1 \vee p_2) \mathrm{U} (q_2 \wedge \mathrm{E}[p_1 \mathrm{U} q_1])]$
  - $\mathrm{E}[\neg(p \mathrm{U} q)] \equiv \mathrm{E}[\neg q \mathrm{U} (\neg p \wedge \neg q)] \vee \mathrm{EG} \neg q$
- Operatorul *until slab* W se defineşte în LTL ori CTL* prin
  - $p \mathrm{W} q \equiv (p \mathrm{U} q) \vee \mathrm{G} p$

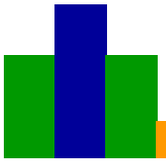
  Această formulă nu se poate folosi în CTL, dar putem folosi identităţile
  - $\mathrm{E}[p \mathrm{W} q] \equiv \mathrm{E}[p \mathrm{U} q] \vee \mathrm{EG} p$
  - $\mathrm{A}[p \mathrm{W} q] \equiv \neg \mathrm{E}[\neg q \mathrm{U} \neg(p \vee q)]$



Model checking I

Cuprins:

- Generalităţi despre verificare
- CTL - computation tree logic
- LTL - logică temporală lineară
- CTL*
- *Concluzii, diverse, etc.*



Concluzii, diverse, etc.

a se insera...