Asupra relaţiilor şi modelelor comportamentale în teoria sistemelor concurente

Conf.dr. Lucian Luca
Universitatea "Tibiscus" din Timişoara

ABSTRACT. The behaviour models provide a suitable base for explaining the behaviour of the systems in time, being useful for modelling the future behaviour of a system. The study, starting from those remarks, presents the way in which there can be algebraically structured the abstract behavioural models, "event structures" and "Mazurkiewicz traces", endowing them with morphisms.

1 Introducere

Atunci când se lucrează cu programe concurente, trebuie avută grijă la definirea noţiunii de "corectitudine". Programele secvenţiale tradiţionale (deterministe) pot fi văzute ca funcţii (parţiale) de la intrări la ieşiri: într-o astfel de abordare, specificaţiile pot fi date ca o pereche, constând dintr-o precondiţie, descriind intrările "permise" şi o postcondiţie, descriind rezultatele dorite pentru aceste intrări.

Această noţiune de specificaţie rămâne potrivită pentru un program concurent reprezentând o versiune "paralelizată" a unui anume program secvenţial; în acest caz, concurenţa (paralelismul) a fost introdusă numai pentru scopuri de performanţă.

Totuşi, pentru sistemele concurente reactive, posibil fără sfârşit şi potenţial non-deterministe, această abordare este prea limitată. Prezentăm, în continuare, noţiuni alternative care au fost dezvoltate pe marginea acestui subiect şi discutăm abordările înrudite pe care ele le generează pentru a raţiona asupra sistemelor.

Pentru specificarea sistemelor concurente am identificat în literatură două şcoli care folosesc mecanisme diferite.

Una se bazează pe folosirea formulelor logice în codificarea proprietăților care interesează; cealaltă foloseşte sisteme de "ordin-înalt" ca specificații pentru cele de nivel jos.

1.1 Logici pentru concurență

Cerințele pe care proiectanții doresc să le impună sistemelor reactive se pot împărți în două categorii:

- *proprietăți de siguranță*, care spun că "ceva rău nu se va întîmpla niciodată".

  Un sistem care satisface o astfel de proprietate trebuie să nu se angajeze în activitatea proscrisă.
- *proprietăți de trăire*, care spun că "ceva bun, eventual, se va întâmpla".

  Pentru a satisface o astfel de proprietate, un sistem trebuie să se angajeze într-o anumită activitate dorită.

Chiar informală, această clasificare s-a dovedit extrem de utilă şi formalizarea ei a fost o motivație principală pentru o mare parte din munca de cercetare efectuată în specificarea şi verificarea sistemelor concurente. O parte din muncă a avut ca scop caracterizarea semantică a acestor proprietăți [AS85].

O altă parte a dezvoltat logici care permit o formulare precisă a celor două proprietăți de mai sus.

Cele mai studiate sunt *logicile temporale*, introduse prima dată în ştiința calculatoarelor de către Pnuelli, care susțin şi permit formularea proprietăților comportamentului unui sistem în timp. Vom prezenta câteva din cercetările legate de logicile temporale. Dihotomiile pe care le-am amintit legat de modelarea concurenței sunt prezente şi în dezvoltarea logicilor concurente şi ele au dat naştere la două şcoli principale de logică temporală.

- *Logici cu timp liniar,* care permit formularea de proprietăți despre secvențele de execuție pe care le arată un sistem.
- *Logici cu timp ramificat*, care permit utilizatorilor să scrie formule care include sensibilitate la alegerile disponibile unui sistem în timpul execuției lui.

În literatură au fost propuse numeroase variante atât pentru logica cu timp liniar, cât şi pentru logica cu ramificare a timpului, după cum

cercetătorii au investigat diferiţi operatori care să uşureze formularea proprietăţilor în diferitele contexte şi setări [MP91]. Expresivitatea acestor formalisme a fost, de asemenea, comparată [EH86] şi, într-un anume fel, a fost dezvoltată o logică temporală canonică expresivă, μ-calculul modal [Koz83].

Celelalte două dihotomii - intensional vs. extensional şi întrepătrundere vs. adevărată concurenţă - au rămas relativ neexplorate. Logicile temporale tradiţionale au adoptat, în general, o vedere extensională a comportamentului sistemului şi un model cu întrepătrundere pentru concurenţă, cu toate că lucrări relativ recente au explorat logici pentru adevărata concurenţă [Thi94].

În final, au fost dezvoltate şi alte logici pentru a raţiona asupra sistemelor concurente, cum ar fi diferite logici dinamice şi logici pentru cunoştinţe. Primele permit includerea programelor în formule [Pel87], iar celelalte permit utilizatorilor să exprime înţelegerea pe care agenţi individuali o au asupra stărilor altor agenţi la un punct din timp dat [HM90], [HZ92].

1.2 Relaţii comportamentale

O altă abordare des folosită pentru specificarea sistemelor concurente implică folosirea *echivalenţelor comportamentale* şi a *preordinilor* pentru a lega specificaţiile de implementări. Ea a fost introdusă de Milner [Mil80] şi a fost exploatată extensiv de către cercetătorii din domeniul algebrelor de proces [BW90], [Hoa85], [Mil89]. Specificaţiile şi implementările se dau în aceeaşi notaţie; primele descriu comportamentul de nivel înalt dorit, iar ultimele oferă detalii de nivel jos indicând cum se obţine acest comportament.

În metodologiile bazate pe echivalenţă trebuie să se dea o cantitate corectă de implementare pentru a stabili că ea se comportă "la fel ca" specificaţia sa.

În metodologiile bazate pe preordine se arată că implementarea oferă "cel puţin" comportamentul dictat de către specificaţie.

Pentru a susţine această abordare, au fost propuse mai multe echivalenţe şi preordini, în funcţie de ce aspecte ale comportamentului sistemului interesează. Relaţiile se pot clasifica pe baza gradului în care ele abstractizează mai departe, plecând de la :

- detaliile interne ale descrierilor sistemului (vis-a-vis de dihotomia intensional/extensional)
- mărimea sensibilității pe care ele o afișează la alegerile sistemului făcute în timpul execuției lor (vis-a-vis de dihotomia timp liniar/timp cu ramificații)
- atitudinea adoptată față de întrepătrundere/adevărata concurență.

De exemplu, echivalența bisimulare [Mil80] este o echivalență cu timp ramificat, bazată pe întrepătrundere şi intensională, pe când echivalența observațională [Mil80] este o echivalență cu timp ramificat, bazată pe întrepătrundere şi extensională.

Alte exemple notabile ar fi relațiile eşecuri/testări (timp liniar, întrepătrundere, extensionale) [BHR84], [DNH84] şi echivalența pomset (timp liniar, adevărată concurență, extensională) [Pra86]. Un studiu detaliat asupra relațiilor dintre diferite echivalențe se găseşte în [Gla90].

Aceste relații se pot folosi, de asemenea, pentru a umple golul dintre modelele intensionale şi extensionale ale concurenței, aşa cum le-am descris mai sus. Adică, pentru a defini o relație peste modelele intensionale, trebuie, la început, selectată o informație extensională sau "observabilă", o informație pe care procesele o pot arăta, şi apoi această informație să fie folosită ca şi bază pentru modelele înrudite.

Principala distincție dintre cele două abordări pentru specificații constă în cantitatea de informații pe care trebuie să o specifice utilizatorii.

Abordările bazate pe logică cer puține specificații, fiindcă este permis a se identifica doar şi o singură proprietate pe care ar trebui să o aibe sistemul.

Abordările bazate pe sistem cer specificații cât mai complete despre comportamentul observabil cerut, chiar dacă, în general, preordinile permit mai puține specificații decât echivalențele.

Pe de altă parte, *relațiile comportamentale* oferă sprijin pentru rafinările bazate pe paşi, la fel ca şi abordări compoziționale pentru a analiza comportamentul sistemului, ceea ce, în general, logica temporală nu oferă, deoarece notațiile pentru specificații şi pentru sistem diferă.

În literatură au fost explorate şi conexiunile dintre cele două abordări. în particular, o logică temporală induce o echivalență pe sisteme în felul următor: două sisteme sunt echivalente dacă şi numai dacă ele satisfac aceleaşi formule.

Folosind acest cadru, s-au stabilit relații şi echivalențe între diferite logici cu timp liniar şi cu timp cu ramificații [HM85], [BCG88].

2 Modele comportamentale

2.1 Structuri de evenimente

Structurile de evenimente descriu un sistem concurent prin apariția unor anume evenimente, cum ar fi *a avut loc o anumită acțiune*, *starea mașinii s-a schimbat*, etc. Aceasta se face printr-o ordine parțială, $\leq$, pe mulțimea evenimentelor. Atunci când două evenimente nu sunt legate prin $\leq$, ele sunt candidate pentru a se executa în paralel, deoarece nici un eveniment nu trebuie să îl preceadă pe celălalt. Pentru a putea modela excluderile mutuale dintre evenimente, avem nevoie şi de o relație de conflict între evenimente, $\bowtie$, definită astfel:

$$e_0 \bowtie e_1 \Leftrightarrow e_0 \text{ şi } e_1 \text{ nu pot să aibe loc în acelaşi moment.}$$

Evenimentele e_0 şi e_1 sunt adevărat concurente, $\odot$, când nici unul nu este înaintea celuilalt şi ele nu sunt nici în conflict.

$$(e_0 \odot e_1) \Leftrightarrow \neg((e_0 \leq e_1) \vee (e_1 \leq e_0) \vee (e_0 \bowtie e_1))$$

Definiția 1 *1. O* ***structură primară de evenimente*** *este o structură* $(E,\leq,\bowtie)$ *unde* E *este o mulțime de evenimente parțial ordonată* de $\leq$, *numită relația de dependență cauzală şi* $\bowtie \subseteq E \times E$ *este o relație simetrică şi ireflexivă, numită relația de conflict, satisfăcând:*

- $\{e^1 / e^1 \leq e\}$ *este finită (axioma "cauzelor finite").*
- $e \bowtie e^1$ şi $e^1 \leq e^2$ *implică* $e \bowtie e^2$ *(conflictul este ereditar).*

2. O ***structură de evenimente etichetate*** $(E, \leq, \bowtie, l, L)$ *este compusă dintr-o structură de evenimente* $(E, \leq,\bowtie)$, *o mulțime de etichete* L *şi o funcție de etichetare* $l : E \rightarrow L$.

Definiția 2 *Fie* $S_1 = (E_1, \leq_1, \bowtie_1)$ şi $S_2 = (E_2, \leq_2, \bowtie_2)$ *două structuri de evenimente. Un* ***morfism de structuri de evenimente*** *de la* S_1 *la* S_2 *este o funcție* $f : E_1 \rightarrow E_2$ *astfel încât:*

- $\{e' / e' \leq_2 f(e)\} \subseteq f(\{e'' / e'' \leq_1 e\})$

$f(e^0) \bowtie f(e^1)$ or $f(e^0) = f(e^1)$ *implică* $e^0 \bowtie e^1$ sau $e^0 = e^1$.

Morfismele de structuri de evenimente etichetate sunt perechi

$$(\eta, \lambda) : (E_1, \leq_1, \bowtie_1, l_1, L_1) \rightarrow (E_2, \leq_2, \bowtie_2, l_2, L_2)$$

astfel încât η este un morfism de structuri de evenimente

$$\eta: (E_1, \leq_1, \bowtie_1) \rightarrow (E_2, \leq_2, \bowtie_2)$$

şi

$$\lambda: L_1 \rightarrow L_2$$

este o funcție satisfăcând $\lambda \circ l_1 = l_2 \circ \eta$.

Acestea formează categoria LES a structurilor de evenimente etichetate. Acesta este primul model "non-operațional" al concurenței. Putem trece uşor peste intuiția operațională din modelele bazate pe evenimente: trebuie doar să colectăm evenimentele "compatibile", ordonate liniar de către timpul la care ele se pot întâmpla şi dăm peste următoarea

Definiția 3 *Fie* $(E,\leq,\bowtie)$ *o structură de evenimente.* ***Mulțimea sa de configurații,*** $D(E,\leq,\bowtie)$, *este mulțimea acelor submulțimi* $x \in E$ *care sunt:*

1. *fără conflicte: pentru toți* $e, e' \in x$, e *nu este în conflict cu* e'.
2. *închisă în jos: pentru toți* e, e', $e \in x$ *şi* $e' \leq e$ *implică* $e' \leq x$.

Mulțimea configurațiilor finite se notează $D^0(E,\leq,\bowtie)$. Noțiunea de dependență cauzală într-o execuție a unei structuri de evenimente este dată prin relația de "permisibilitate".

Definiția 4 *Fie* $e \in E$ *şi* $c \in D(E,\leq,\bowtie)$. *Spunem că* e ***este permis*** *la o configurație* c, *şi scriem* $c \vdash e$ *dacă:*

1. $e \notin c$
2. $\{e' / e' \leq e \wedge e' \bowtie e\} \subseteq c$
3. $e' \in E$ *şi* $e' \bowtie e$ *implică* $e' \notin c$

Configurațiile finite sunt urme atunci când ordonăm linear elementele lor după dependența cauzală.

$\{e_1 < e_2 < \ldots < e_n\}$ este o ***garantare*** pentru c dacă şi numai dacă

$$\{e_1,\ldots, e_{i-1\backslash}\} \vdash e_i, \text{ pentru } i = 1,2,\ldots,n.$$

Scriem, de asemenea, garantările ca şiruri $e_1,\ldots,e_n$.

Evenimentele sunt doar o parte a unei dualități pentru care automatele sunt cealaltă parte: vezi şi cadrul general pentru automate şi planificări (modele bazate pe evenimente) numit ***spații Chu*** ([Pra94a], [Pra94b]).

Structurile de evenimente pe care tocmai le-am definit nu sunt cele mai generale structuri de evenimente din literatură şi discuția despre configurații ne dă tocmai generalizarea de care avem nevoie: în loc de a raționa numai asupra ordinilor parțiale ale evenimentelor, vom considera în mod direct istorii parțiale, adică mulțimi finite şi consistente de evenimente, descrise de către mulțimea *Con* în definiția următoare [Win88]. *Con* ține cont, de asemenea, şi de informația despre conflicte, aşa că nu mai avem nevoie de relația $\bowtie$. Dinamica este descrisă de către relația de permisibilitate, $\models$.

Definiția 5 *O **structură de evenimente** este un triplet* $(E, Con, \models)$, *unde:*

- E *este o mulțime de evenimente*
- Con *este o mulțime nevidă de submulțimi finite ale lui* E, numită ***predicatul de consistență*** *şi care satisface*

$$X \in Con \wedge Y \subseteq X \Rightarrow Y \in Con$$

- $\models \subseteq Con \times E$ *este relația de **permisiune** care satisface*:

$$X \models e \wedge X \subseteq Y \wedge Y \in Con \Rightarrow Y \models e.$$

În figurile 1 şi 2 vom exemplifica o configurație de evenimente plecând de la un model simplu: un circuit electric cu un comutator paralel pentru un bec.

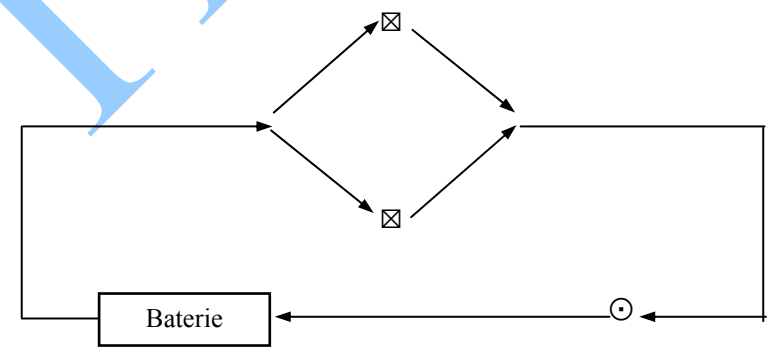

Figura 1: Un comutator paralel pentru un bec

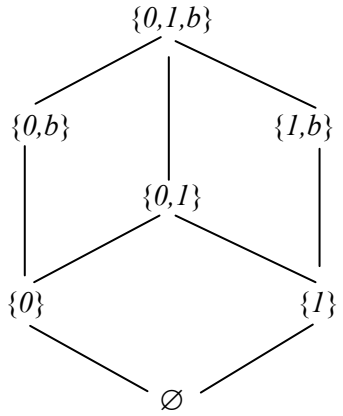

Figura 2: Configuraţii de evenimente pentru exemplul precedent

Aceste structuri de evenimente, [Win88], sunt mai generale în sensul că un eveniment poate fi acum permis în moduri diferite. Un eveniment poate fi cauzat de mai mult de o configuraţie. Spunem că structurile de evenimente pot să arate cauzalitate SAU (OR), pe când structurile primare de evenimente nu o pot arăta. Majoritatea modelelor pentru concurenţă bazate pe ordini parţiale (ex.: pomsets) pot să arate numai cauzalitate AND: un eveniment poate numai să apară dacă şi numai dacă anumite alte evenimente au apărut înainte. Această distincţie între cauzalităţi OR şi AND a fost folosită în [Gun92] pentru a analiza noţiunea lui Milner de confluenţă = determinism + {AND,OR} cauzalitate.

2.2 Urme Mazurkiewitz

În această subsecţiune generalizăm puţin sistemele cu tranziţii asincrone pentru a permite relaţiei de "independenţă" să varieze în concordanţă cu starea locală a maşinii. în literatură, acest lucru nu a fost dezvoltat pentru sistemele cu tranziţii, ci pentru partea de limbaje a teoriei automatelor, adică teoria urmelor Mazurkiewicz. Ca referinţă, folosim [SNW94].

Definiţia 6 *Un* ***limbaj generalizat pentru urme*** *este un triplet* (M,I,L) *unde:*

- L *este o mulţime de simboluri*
- $M \subseteq L^*$

- *$I : M \to 2^{L \times L}$ este o funcție care asociază la fiecare $s \in M$ o relație $I_s \subseteq L \times L$*

astfel încât, dacă definim $\cong$ ca fiind cea mai mică relație de echivalență pe L^* astfel încât sabu $\cong$ sbau dacă a I_s b și:

- *pentru toți $s \in M$, I_s este simetrică și ireflexivă*
- *M este I-închisă: a I_s b $\Rightarrow$ sab $\in M$*
- *I este consistentă: $s \cong s' \Rightarrow I_s = I_{s'}$*
- *I este coerentă*:
  - **(i)** *a I_s b și a I_{sb} c și c I_{sa} b $\Rightarrow$ a I_{sc} b*
  - **(ii)** *a I_s* c și c I_s *b $\Rightarrow$ (a I_s b* dacă și numai dacă *a I_{sc}* b)

Definiția 7 *Fie (M^1,I^1,L^1) și (M^2,I^2,L^2) două limbaje generalizate pentru urme. O funcție $\lambda : L^1 \to L^2$ definește un **morfism** de la (M^1,I^1,L^1) la (M^2,I^2,L^2) dacă și numai dacă:*

- *păstrează cuvintele, adică $s \in M^1$* implică *$\lambda^*(s) \in M^2$*
- *λ respectă independența: a I^1_s b implică $\lambda(a)\ I^2_{\lambda^*(s)}\ \lambda(b)$*

unde λ^* este o extensie pe cuvinte definită astfel:

$\lambda^*(\varepsilon) = \varepsilon$

$\lambda^*(sa) = \lambda^*(s)\,\lambda(a)$

Notăm cu **GTL** categoria limbajelor generalizate pentru urmă și definim limbajul pentru urme Mazurkiewicz:

Definiția 8: *Un **limbaj pentru urme Mazurkiewicz** este un triplet* (M,I,L), unde:

- *L este o mulțime de simboluri*
- *$M \subseteq L^*$*
- *I este o relație simetrică și ireflexivă pe L, astfel încât:*
  1. *I este **închisă față de prefix**: sa $\in$ M implică s $\in$ M pentru toți s $\in L^*$ și a $\in$ L*
  2. *M este **I-închisă**: sabt $\in$ M și a I b implică sbat $\in$ M pentru toți s,t $\in L^*$ și toți a,b $\in$ L*
  3. *M este **coerentă**: sa $\in$ M și sb $\in$ M și a* I *b implică sab $\in$ M pentru toți s $\in L^*$ și toți a,b$\in$* L

Urmele Mazurkiewicz sunt un caz special al limbajelor generalizate pentru urme, căci I defineşte o funcţie constantă de la M la $2^{L\times L}$, satisfăcând axiomele din *definiţia 6*.

3 Discuţie

Punctul de plecare pentru diferenţierea între diferitele teorii ale concurentei îl constituie definiţia şablonului (modelului). Mai exact, modelele care au fost dezvoltate se pot clasifica pe baza a ceea ce acceptă ele din următoarele trei dihotomii:

- intensionalitate ↔ extensionalitate;
- întrepătrundere ↔ adevărată concurentă;
- timp cu ramificări ↔ timp liniar.

Prima dihotomie apare şi în semantica secvenţializării, iar ultimele două sunt specifice doar concurenţei.

Intensionalitate ↔ extensionalitate

Modelele intensionale descriu ce fac sistemele, iar modelele extensionale se bazează pe ce vede un observator exterior. În consecinţă, teoriile intensionale se mai zic şi operaţionale, iar cele extensionale se mai zic denotaţionale, denotaţia unui proces fiind multimea observaţiilor pe care el le produce.

Teoriile intensionale modelează sistemul în termeni de stări şi de tranziţii între stări. Astfel:

- în *sistemele cu tranziţii etichetate*, tranziţiile sunt decorate cu acţiuni atomice reprezentând interacţiuni cu mediul; acestea au fost studiate pe larg în contextul algebrelor de proces, cum ar fi ACP, CCS şi CSP [Hoa85].
- *automatele I/O* urmează şi ele această abordare, dar permit distincţii între diferitele tipuri de acţiuni (intrări şi ieşiri).
- *reţelele Petri* extind paradigma stare/tranziţie pemitând stărilor să fie "distribuite" în diferite locaţii.
- *UNITY* adoptă un stil imperativ, cu tranziţii corespunzând execuţiilor (atomice) de instrucţiuni de asignare condiţională.

Din contră, modelele extensionale definesc prima dată o noţiune de observaţie şi apoi reprezintă sistemele în funcţie de observaţiile ce se pot face asupra lor:

- una din observaţiile de bază despre un sistem este ***urma*** (trace), adică secvenţa de acţiuni atomice executate de către un sistem [Hoa85]. Elaborări ulterioare ale acestui model includ *arbori de acceptare* şi *mulţimi de eşec*. Ambele abordări adaugă la informaţia de bază pentru urmă informaţii despre stimulii la care poate răspunde un sistem după ce a executat o secvenţă.
- *arborii de sincronizare* [Mil89] codifică comportamentul sistemului ca şi un arbore cu arcele etichetate de acţiuni.
- ***urmele Mazurkievicz*** includ o relaţie de independenţă între acţiuni pentru a captura posibila concurenţă.
- *reţelele Kahn* definesc comportamentul sistemelor flux de date, în care componentele sunt date de către ecuaţii de I/O, prin folosirea teoriei punctelor fixe.

Alte modele, cum ar fi *pomset*, codifică informaţia despre concurenţă prin ordini parţiale pe evenimente atomice.

Ele se pot îmbogăţi cu o relaţie de conflict pentru a codifica informaţia despre alegerile făcute în timpul execuţiei, dând naştere la ***structuri de evenimente***.

Întrepătrundere ↔ adevărată concurenţă

Modelele cu întrepătrundere reduc concurenţa la non-determinism prin tratarea execuţiei paralele a acţiunilor ca alegere între secvenţializările lor. Această abordare poate fi numită şi abordarea "uniprocesor" a concurenţei. Pentru a evita situaţii anormale, se impun uneori restricţii de corectitudine (fairness constraints) proceselor individuale pentru ca acestea, într-adevăr să "facă progrese" dacă sunt capabile de aşa ceva. Teorii ca *ACP*, *CCS* şi *CSP* folosesc întrepătrunderea, la fel ca şi *automatele I/O* şi *UNITY*. Ultimele două includ, de asemenea, mecanisme pentru a forţa restricţiile de corectitudine. Concurenţa este modelată prin întrepătrundere şi în modelele bazate pe urmă şi în modelele extensionale cu arbore de sincronizare.

Din contră, modelele cu adevărată concurenţă tratează concurenţa ca pe o noţiune primitivă; comportamentul unui sistem este reprezentat în termeni de relaţii cauzale între evenimentele ce au loc la diferite "locaţii" dintr-un sistem. Concurenţa este prezentată în această formă de către *reţelele Petri*, *reţelele Kahn*, ***urmele Mazurkiewicz***, *pomset-uri* şi ***structuri de evenimente***.

Timp cu ramificări ↔ timp liniar

Diferenţa dintre modelele care folosesc timpul cu ramificări şi cele

care folosesc timpul liniar constă în modalitatea în care ele tratează alegerile pe care sistemele trebuie să le facă în timpul execuţiei lor.

Modelele cu timp liniar descriu sistemele concurente în termeni de mulţimi de execuţii (parţiale) posibile, în timp ce în modelele cu ramificare a timpului se înregistrează punctele la care diferite calcule diverg între ele.

Urmele şi *pomset-urile* sunt modele cu timp liniar, iar *arborii de sincronizare* şi ***structurile de evenimente*** sunt modele cu timp cu ramificări.

Deciziile privitoare la fiecare din cele trei dihotomii sunt consecinţele unor scopuri diferite [LD00c].

Modelele extensionale oferă o bază potrivită pentru explicarea comportamentului sistemelor, în timp ce cele intensionale sunt adesea mai potrivite pentru analiza automatizată, căci ele dau naştere, de obicei, la maşini cu stări finite.

Semantica întrepătrunderii este utilă pentru specificarea sistemelor, iar semantica adevăratei concurenţe ar putea fi baza pentru descrierea posibilelor implementări, acolo unde, de exemplu, performanţa contează.

O semantică cu ramificare a timpului este utilă pentru a modela comportamentul viitor al unui sistem, iar timpul liniar este suficient pentru a descrie istoria execuţiilor.

Bibliografie

[BCG88] **M. C. Browne, E. M. Clarke, and O. Grümberg** – *Characterizing finite Kripke structures in propositional temporal logic*, Theoretical Computer Science, 59(1,2):115-131, 1988.

[BHR84] **S. D. Brookes, C. A. R. Hoare, and A. W. Roscoe** - *A theory of communicating sequential processes*, Journal of the ACM, 31(3):560-599, July 1984.

[BW90] **J. C. M. Baeten and W. P. Weijland** - *Process Algebra*, Cambridge Tracts in Theoretical Computer Science, 18, Cambridge University Press, 1990.

[DL96] **I. Despi, L. Luca** – *Baze de date orientate pe obiecte*, Editura Mirton, Timişoara, 1996

[dNH84] **R. de Nicola and M. Hennessy** - *Testing equivalences for processes*, Theoretical Computer Science, 34:83-133, 1984.

[EH86] E**. A. Emerson and J. Y. Halpern** - *"Sometime" and "not never" revisited: On branching versus linear time temporal logic*, Journal of the ACM, 33(1):151-178, 1986.

[Gla90] **R. J. van Glabbeek** - *Comparative Concurrency Semantics and Refinement of Actions*, PhD thesis, Free University of Amsterdam, 1990.

[Gun92] **J. Gunawardena** – *Causal Automata*, Theoretical Computer Science, (101):265-288, 1992

[HM85] **M. Hennessy and R. Milner** - *Algebraic laws for nondeterminism and concurrency*, Journal of the ACM, 32(1):137-161, 1985.

[HM90] **Halpern and Y. Moses** - *Knowledge and common knowledge in a distributed environment*, Journal of the ACM, 37(3):549-587, July 1990.

[Hoa85] **C. A. R. Hoare** - *Communicating Sequential Processes*, Prentice-Hall, London, 1985.

[HZ92] **J. Halpern and L. Zuck** - *A little knowledge goes a long way: Knowledge-based derivations and correctness proofs for a family of protocols*, Journal of the ACM, 39(3):449-478, July 1992.

[Koz83] **D. Kozen** - *Results on the propositional μ-calculus*, Theoretical Computer Science, 27:333-354, 1983.

[LD00c] **L. Luca şi I. Despi** – *Modele matematice pentru concurență*, Analele Universității "Tibiscus", vol. X: 101-106, Timişoara, 2000

[Mil80] **R. Milner** - *A Calculus of Communicating Systems*, Lecture Notes in Computer Science, 92, Springer-Verlag, 1980.

[Mil89] **R. Milner** - *Communication and Concurrency*, International Series in Computer Science, Prentice Hall, 1989.

[MP91] **Z. Manna and A. Pnueli** - *The Temporal Logic of Reactive and Concurrent Systems: Specification*, Springer-Verlag, 1991.

[Pel87] **D. Peled** - *Concurrent dynamic logic*, Journal of the ACM, 34(2):450-479, April 1987.

[Pra86] **V. R. Pratt** - *Modeling concurrency with partial orders*, International Journal of Parallel Programming, 15(1):33-71, 1986.

[Pra94a] **V. R. Pratt** - *Chu spaces: Automata with quantum aspects*, Technical report, Stanford University, 1994

[Pra94b] **V. R. Pratt** - *Chu Spaces: Complementarity and uncertainty in rational mechanics*, Technical report, Stanford University, 1994.

[SNW94] **V.Sassone, M. Nielsen and G. Winskel** – *Relationships between models of concurrency*, in Proceedings of the REX 3 school and symposium, 1994.

[Thi94] **P. S. Thiagarajan** - *A trace based extension of linear time temporal logic*, in Ninth Annual Symposium on Logic in Computer Science (LICS '94), pp. 438-447, Versailles, France, July 1994, Computer Society Press.

[Win88] **G. Winskel** - *An introduction to event structures*, in J.W. de Bakker, W.P. de Roever, and G. Rozenberg, editors, REX School and Workshop on Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, Noordwijkerhout, The Netherlands, May/June 1988, Lecture Notes in Computer Science, 354:364-397, Springer-Verlag, 1989.