Asupra algebrelor de proces pentru sistemele în timp real

Conf.dr. Lucian Luca, Asist. Adela Ionescu,
Universitatea „Tibiscus" Timişoara
Mat.ec. Sorina-Carmen Luca
Frankfurt-Bukarest Bank AG.

ABSTRACT. There are presented the attempts to define a calculus for parallel processes, in the beginning being described some representative process algebras (CCS, CSP, π-calculus). Then, there are presented and discussed another extensions of classical semantics models for the real-time systems, analysing their advantages and disadvantages.

Există mai multe încercări pentru a defini un calcul pentru procese paralele, care să fie la fel de fundamental ca şi λ-calculul pentru programa-rea funcţională. Într-un anume fel, aceste încercări încă nu au reuşit, dar toate algebrele de proces introduse au dat o înţelegere mai bună asupra comportamentelor de bază pe care le pot arăta sistemele concurente. Ne vom opri asupra câtorva, considerate reprezentative.

1 CCS

Limbajul original ([Mil89]) a avut variabile, valori şi canale. CCS pur este o simplificare, în care comunicaţiile de-a lungul canalelor sunt simplifi-cate în felul următor:

- valorile actuale ce se transmit pe canale se pierd după transmitere;
- de asemenea, se pierd numele variabilelor, păstrându-se numai numele canalului;

- pentru a obține o valoare într-o variabilă ***x*** pe canalul ***a***, adică ***a?x***, se abstractizează prin ***a***;
- pentru a trimite o valoare ***n*** pe canalul ***a***, adică ***a!n***, se abstractizează prin acțiunea complementară, $\overline{\boldsymbol{a}}$ **.**

În acest mod, comunicația este abstractizată prin sincronizarea unei acțiuni şi a acțiunii complementare. Această sincronizare este observabilă prin apariția unei acțiuni invizibile **τ**.

În cele urmează, **α** reprezintă acțiuni, acțiuni complementare şi τ, iar ***f*** este o funcție pe acțiuni care comută cu $\boldsymbol{x} \to \overline{\boldsymbol{x}}$. Sintaxa lui CCS pur este:

t::= ***nil***	(procesul nul)
$\|t_1+t_2$	(operatorde alegere)
$\|t_1 \mid t_2$	(compunere paralelă)
$\|\alpha.t$	(prefixare)
$\|t_1\backslash c$	(operator de restricționare)
$\|p[f]$	(re-etichetare)
$\|$***rec*** $x.t(x)$	(agent recursiv)

Prima semantică [Mil89] a fost dată în termeni de arbori de sincronizare (sisteme cu tranziții aciclice). Adesea ea se dă folosind reguli SOS [Plo81], care descriu sistemele cu tranziții astfel:

$$\frac{}{\alpha.t \xrightarrow{\alpha} t}$$

Această regulă spune că $\alpha.t$ poate declanşa prima dată o acțiune α.

$$\frac{t_i \xrightarrow{\alpha} t'_i}{t_1 + t_2 \xrightarrow{\alpha} t'_i}$$

Această regulă arată că t_1+t_2 se comportă ca t_1 sau ca t_2 odată şi pentru totdeauna.

$$\frac{t_1 \xrightarrow{\alpha} t'_1}{t_1 \mid t_2 \xrightarrow{\alpha} t'_1 \mid t_2}$$

$$\frac{t_2 \xrightarrow{\alpha} t'_2}{t_1 \mid t_2 \xrightarrow{\alpha} t_1 \mid t'_2}$$

Aceste reguli definesc pe $t_1|t_2$, prin întrepătrunderea lor.

$$\frac{t_1 \xrightarrow{\alpha} t'_1 \qquad t_2 \xrightarrow{\overline{\alpha}} t'_2}{t_1 \mid t_2 \xrightarrow{\tau} t'_1 \mid t'_2}$$

Această regulă defineşte *sincronizarea între acțiuni complementare* care, atunci când se "anihilează" între ele, produc o acțiune tăcută τ.

$$\frac{t \xrightarrow{\alpha} t' \quad \alpha \neq c \quad şi \quad \alpha \neq \bar{c}}{t \setminus c \xrightarrow{\alpha} t' \setminus c}$$

Această regulă defineşte *operatorul de restricţionare*, care se aplică atât unei acţiuni, cât şi acţiunii sale complementare.

$$\frac{t \xrightarrow{\alpha} t'}{t[f] \xrightarrow{f(\alpha)} t'[f]}$$

Această regulă defineşte într-un mod evident *operatorul de re-etichetare.*

$$\frac{\textbf{rec}\ x.t[x] \xrightarrow{\alpha} t'}{t[\textbf{rec}\ x.t[f]] \xrightarrow{\alpha} t'}$$

Această regulă spune că ***rec*** *x.t*[*x*] este similar cu "termenul infinit" *t*[*t*[...[***nil***]]].

Semantica "denotaţională" a sistemelor ordinare cu tranziţii se poate da şi folosind proprietăţile lor categoriale[WN94]:

- produsul cartezian este o formă de produs sincronizat, în plus cu întrepătrundere (aceleaşi tranziţii, cu o tranziţie * drept una din componente);
- coprodusul fibrat este alegerea non-determinstă din CCS;
- operatorul de restricţie din CCS este obţinut printr-un lifting cartezian tare.
- operatorul de re-etichetare din CCS corespunde lifting-ului cocartezian tare;
- operatorii paralel şi de prefixare din CCS nu corespund nici unui combinator categorial în categoria sistemelor cu tranziţii.

Semantica "denotaţională" a sistemelor ordinare cu tranziţii se poate defini şi folosind reţelele Petri, chiar dacă construcţiile categoriale în reţele Petri nu sunt uşor de găsit. În [WN94] se arată că cel puţin următoarele construcţii sunt disponibile:

- Coprodusul a două reţele este suma non-determinstă a reţelelor. Comportamentele sunt, într-adevăr, cele ale alegerii non-deterministe numai pentru aşa-zisele *reţele sigure*.
- Produsul a două reţele corespunde unei sincronizări a celor două reţele.

O abordare diferită, folosind reţelele Petri, se găseşte în [DNM88], unde ele sunt considerate a fi sisteme cu tranziţii distribuite. Termenii CSS sunt descompuşi în procese locale şi apoi li se dă o semantică operaţională standard.

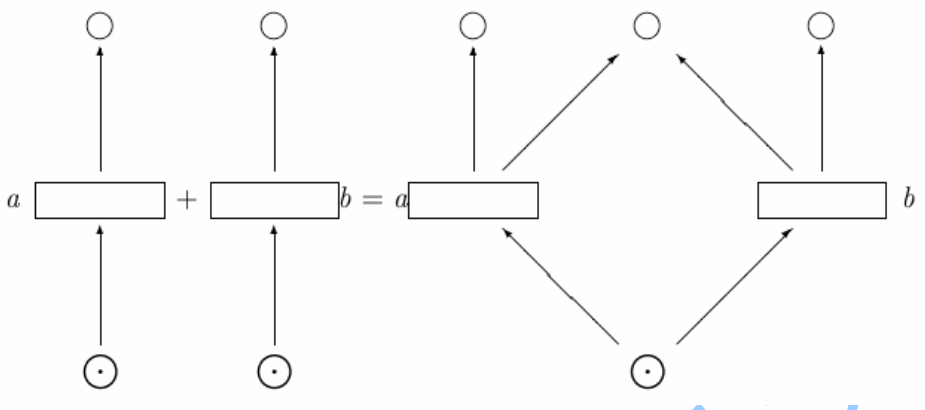

Figura 1.1. Coprodusul a două rețele Petri

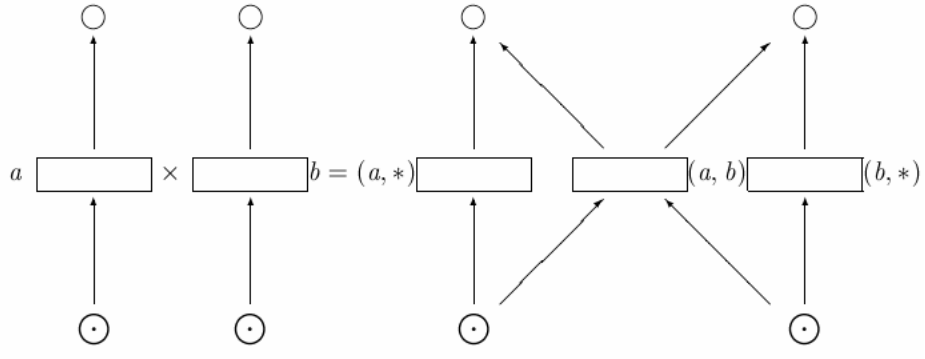

Figura 1.2. Produsul a două rețele Petri

Alte abordări cu rețele Petri țin cont numai de submulțimile lui CCS, aşa cum se întâmplă în [GM84] şi [dC+83].

O semantică folosind structuri de evenimente a fost dată în [Win89], iar semantici folosind ordini parțiale au fost date în [DDNM85] şi [DNM88]. În [Gup94] se dă o semantică în termeni de spații Chu.

2 Procese secvențiale comunicante (CSP)

CSP (*Communicating Sequential Processes*) au fost introduse în [Hoa85]. Ele au aproximativ aceleaşi primitive ca şi CCS, însă mecanismul de sincronizare este puțin diferit. Sintaxa unui CSP "cu valori" este dată de următoarele reguli:

$t ::= \alpha\,?\,X \rightarrow t$	(input pe canalul α şi memorează în X, apoi execută t)
$\mid \alpha!a \rightarrow t$	(output valoarea a pe canalul α şi apoi execută t)
$\mid t_1 \parallel t_2$	(compunere paralelă)
$\mid t_1 \square t_2$	(operator de alegere)
$\mid$ l : t	(etichetare)
$\mu\, x.t(x)$	(agent recursiv, pentru t o expresie cu gardă

O semantică în termenii sistemelor cu tranziţii etichetate se poate găsi în [Win93]. În acest calcul, am ales să reprezentăm valorile şi variabilele, aşa că stările sistemului cu tranziţii vor fi de forma $\langle c,\sigma \rangle$, unde c este un program CSP şi σ este o memorare (adică o funcţie de la variabile la valori). Deoarece semantica acestui limbaj CSP este extrem de similară cu cea a limbajului CCS, o vom specifica doar pentru operatorul paralel:

$$\frac{\langle c_0,\sigma\rangle \xrightarrow{\lambda} \langle c'_0,\sigma'\rangle}{\langle c_0 \| c_1,\sigma\rangle \xrightarrow{\lambda} \langle c'_0 \| c_1,\sigma'\rangle}$$

$$\frac{\langle c_1,\sigma\rangle \xrightarrow{\lambda} \langle c'_1,\sigma'\rangle}{\langle c_0 \| c_1,\sigma\rangle \xrightarrow{\lambda} \langle c_0 \| c'_1,\sigma'\rangle}$$

CSP fără valori are sintaxa următoare:

$$\begin{array}{rccc} P ::= & SKIP & STOP & a \rightarrow P \\ & P;\, Q & P \cap Q & P \parallel Q \\ & P \,|||\, Q & P \setminus a & \begin{array}{c} repeat \\ P \end{array} \end{array}$$

$P|||Q$ denotă întrepătrunderea lui P cu Q. $P\|Q$ este sincronizarea cu pas de încuiere a lui P cu Q. Aceasta înseamnă că procesele trebuie să sincronizeze fiecare acţiune a lui P şi Q cu acelaşi nume.

O semantică denotaţională (adevărata-concurenţă) a lui CSP fără valori, folosind structuri de evenimente, a fost propusă în [Win89].

3 π-calcul

Fie X o mulţime de nume cu elementele tipice x, y, etc. şi fie p, q procese din mulţimea P de procese. Ele se construiesc folosind sintaxa următoare:

p::= **0** (procesul nul)
| $\sum_{i \in I} \pi_i \cdot p_i$ (sumă finită de oricâte procese)
| p|q (compunere paralelă)
|!*p* (replicare)
| *vxp* (operator de restricționare)}

Prefixele π_i din suma de mai sus reprezintă acțiuni atomice care pot fi de forma următoare:

- *x(y)* înseamnă: *input un nume, să-l numim* ***y****, de-a lungul legăturii numite* ***x****;*
- $\bar{x}y$ înseamnă: *output numele* ***y****, de-a lungul legăturii numite* ***x****.*

Înainte de a defini o semantică operațională (bazată pe sisteme ordinare cu tranziții), definim o congruență structurală ≅ pe mulțimea termenilor:

- procesele care diferă numai printr-o schimbare a numelor de legătură sunt identificate;
- + şi | sunt comutative pe clasele de echivalență modulo ≅
- $p + \mathbf{0} \cong p$ şi $p \mid \mathbf{0} \cong p$
- $!p \cong p \mid !p$
- $vx\,\mathbf{0} \cong \mathbf{0}$ şi $vxvyp \cong vyvxp$
- dacă *x* nu este un nume liber în *p* atunci $vx(p|q) \cong p \mid vxq$

Ca şi pentru CCS, termenii sintactici sunt stări ale sistemului cu tranziții, dar luați modulo congruența ≅. Tranzițiile se exprimă acum în forma SOS [ACS96] astfel:

$$\frac{}{(\ldots x(y).p) \mid (\ldots \bar{x}z.q) \rightarrow p[z/y] \mid q}$$

$$\frac{p \rightarrow p'}{p|q \rightarrow p'|q} \qquad \frac{p \rightarrow p'}{vxp| \rightarrow vxp'}$$

$$\frac{q \cong p \qquad p \rightarrow p' \qquad p' \cong q'}{q \rightarrow q'}$$

Vom prezenta în continuare exemplul [Mil93] pentru tipul de mobilitate care poate fi atins folosind π-calculul:

- Să considerăm termenii:
  $P = \bar{x}y\,.\mathbf{0}$ $Q = x(u).\bar{u}\,v.\mathbf{0}$ $R = \bar{x}\,z.\mathbf{0}$
- şi procesele

$X = P \mid Q \mid R$

- P poate trimite pe y la Q şi R poate trimite pe z la Q, dar nu ambele
- Astfel, cele două alternative pentru rezultat sunt:
  $\mathbf{0} \mid \bar{y}\, v.\mathbf{0} \mid \bar{x}\ \mathrm{y}.\mathbf{0}$ sau
  $\bar{x}\ \mathrm{y}.\mathbf{0} \mid \bar{z}\, v.\mathbf{0} \mid \mathbf{0}$
- R a devenit astfel $\bar{y}\, v.\mathbf{0}$ sau $\bar{z}\, v.\mathbf{0}$
- Comunicația a determinat care canal R se poate folosi imediat în continuare pentru ieşire, y sau z.

4 Sisteme în timp real

Prin ***sistem în timp real*** se înțelege o maşină secvențială sau concurentă ale cărei stări depind de unul sau mai multe ceasuri (orologii), adică a cărei evoluție este constrânsă de timp. Timpii de execuție pentru acțiuni sunt considerați a fi insignifianți comparativ cu constantele de timp ale semnalelor externe (de fapt, intrările în program). Exemple de astfel de sistem sunt la tot pasul în lumea reală: ceasuri cu alarmă, maşini automate pentru cafea, limbaje în timp real, calculatoare, etc. Modelele pentru timp real sunt considerate bune dacă ele sunt rafinabile, digitizabile şi operaționale.

4.1 Sisteme de tranziții cu timp

Ele presupun existența unui ceas global fictiv cu valori reale [HMP93] şi se bazează pe sistemele cu tranziții ordinare la care adaugă restricții asupra întârzierilor minime şi maxime dintre acțiunile pe care le pot autoriza să se declanşeze. Tranzițiile nu au durată, adică timpul lor de execuție e nul.

Definiția 4.1. *Un **sistem de tranziții cu timp** este notat TTS=(S,i,E,Tran,l,u) unde:*

- *(S, i, E, Tran) este un sistem de tranziții;*
- *l este o colecție $l_\tau \in \mathbb{N}$ de întârzieri minime pentru acțiunile $\tau \in E$;*
- *u este o colecție $u_\tau \in \mathbb{N} \cup \{\infty\}$ de întârzieri maxime pentru acțiunile $\tau \in E$.* ■

Urmele, numite *secvențe de execuții cu timp*, sunt secvențe $(\sigma_i, T_i)_{i\in\mathbb{N}}$ $\sigma_i \in S$ şi $T_i \in R$ astfel încât:

- $(\sigma_i)_{i\in\mathbb{N}}$ este o urmă a sistemului de tranziții de fundal;
- "timpul nu descreşte niciodată", adică

$$\forall\ i \in \mathbb{N},\ (T_{i+1} = T_i) \vee ((T_{i+1} > T_i) \wedge (\sigma_{i+1} = \sigma_i))$$

- "timpul diverge", adică
$$\forall\ t \in \mathbb{R},\ \exists\ i \geq 0,\ T_i \geq t$$
- "o tranziție τ trebuie să fie autorizată cel puțin l_τ unități de timp pentru a putea fi declanşată", adică pentru orice tranziție $\tau \in E$ şi $i \geq 0, j \geq i$ cu $T_j < T_i + l_\tau$, dacă τ este luată la poziția j a lui σ atunci τ este autorizată pe σ_i.
- "o tranziție τ nu poate fi autorizată pentru mai mult de u_τ unități de timp, fără a fi luată", adică pentru orice tranziție $\tau \in E$ şi $i \geq 0$, există $j \geq i$ cu $T_j \leq T_i + u_\tau$ astfel încât fie τ nu este autorizată pe σ_j sau τ este luată la poziția j a lui σ.

4.2 Automate cu timp

Maşinile cu stări finite se pot generaliza peste şiruri infinite [AD94]. Ele constau, în principal, din mulțimi finite de locații şi mulțimi finite de ceasuri cu valori reale. Tranzițiile nu au lungime în timp, iar stările aşteaptă perioade pentru ca restricțiile ceasurilor să fie satisfăcute.

Definiția 4.2 *Dacă X este o mulțime de ceasuri, mulțimea $\Phi(X)$ de* ***restricții de ceas*** *δ este definită inductiv astfel:*

$$\delta := x \leq c \mid c \leq x \mid \neg \delta \mid \delta_1 \wedge \delta_2$$

unde x este un ceas în X şi c este o constantă din domeniul timpului. ■

Definiția 4.3 ***O tabelă de tranziții cu timp*** *este un tuplu $\langle \Sigma, S, S_0, C, E \rangle$ unde:*

- *Σ este un alfabet finit;*
- *S este o mulțime finită de stări;*
- *$S_0 \subseteq S$ este o mulțime de stări de start;*
- *C este o mulțime finită de ceasuri;*
- *$E \subseteq S \times S \times \sigma \times 2^C \times \Phi(C)$ dă mulțimea tranzițiilor.* ■

Muchiile (tranzițiile) sunt $(s,s',a,\lambda,\delta) \in E$, plecând din starea s în starea s', pe simbolul de intrare a. $\lambda \in C$ dă ceasul care trebuie resetat cu această tranziție, iar δ este o restricție de ceas peste C.

Se pot da mai multe tipuri de criterii de acceptare: Büchi, Müller, etc. pentru aceste automate şi apoi ele acceptă limbaje cu timp (un limbaj al automatului de fundal fără timp şi cu constrângeri de timp).

5 Discuție

Majoritatea modelelor semantice pentru sistemele cu timp real presupun că tranzițiile nu au lungime în timp şi că stările suportă modificări de timp. Această abordare a influențat profund numeroasele algebre de proces care au apărut în ultimii ani [LL03].

Toate algebrele de proces se bazează pe o "schemă de funcționare în două faze": execuțiile lor alternează de la o parte sincronă, în care toate componentele sunt de acord pentru ca timpul să progreseze, la o parte asincronă, în care progresul timpului este blocat.

Această perspectivă (puternic legată de comportamentul real al unor sisteme hibride) conduce la dificultăți tehnice care ascund adevăratele probleme ale modelării sistemelor în timp real. În particular, ***deadlock-freeness***, faptul că nici un proces nu poate bloca progresul timpului, este o dificultate tehnică a modelului semantic şi, sigur, nu este o proprietate interesantă pentru sistemele în timp real.

Unele dintre algebrele de proces cu timp real sunt simple extensii ale unor bine-cunoscute algebre de proces fără timp:

1. **TCSP** este o extensie simplă a lui CSP cu:
   - un operator de întârziere $t.P$ (se comportă ca P după exact t unități de timp);
   - un operator de time-out slab - se comportă ca P şi apoi execută Q după d unități de timp.
2. **TeCCS**, **TiCCS** sunt extensii ale lui CCS cu:
   - încuiere de timp 0 (pentru TeCCS);
   - un operator de întârziere pe care îl scriem $(t).P$ (pentru ambele) - el se comportă ca P după exact t unități de timp;
   - inactivitate nemărginită - δP poate acționa ca şi P după orice durată de timp (pentru TeCCS);
   - un operator de prefixare $a@vP$ care execută pe a şi se comportă apoi ca şi P cu variabila de timp v la orice moment (TiCCS).
3. **ACP$_\rho$** bazat pe ACP cu:
   - $\delta(d)$ o încuiere de timp la momentul d;
   - acțiuni ştampilate cu timp:
     - ştampile absolute, ca $a(d)$, care execută acțiunea a la momentul d, sau
     - ştampile relative, ca $a[d]$ $(d \in \mathbb{R})$ care execută acțiunea a la d unități de timp după ce acțiunea precedentă a fost executată;

- un operator integral $\int_{v \in V} P(v)$, care se comportă ca şi P cu variabila de timp v înlocuită prin orice valoare a lui V.

4. **ATP** cu:
   - inactivitate nemărginită, în sensul că $\lfloor P \rfloor^{\omega}$ poate executa acţiuni din P sau poate fi inactiv cât de mult timp doreşte;
   - un operator de start-întârziere, $\lfloor P \rfloor^{d}$ (Q) care se comportă ca şi P pentru cel mult d unităţi de timp şi apoi execută Q.
5. **TPL** cu:
   - un operator de întârziere $(t).P$ care se comportă ca şi P după exact t unităţi de timp.
6. **U-LOTUS** cu:
   - un operator de întârziere $(t).P$, care se comportă ca şi P după t unităţi de timp;
   - un operator *asap* (eng.: *as soon as possible*), ce forţează urgenţa unei mulţimi de acţiuni în întreaga execuţie a unui proces.

Toate sunt ***fără interblocări*** (*deadlock-free*), în sensul că nici un proces nu poate bloca progresul timpului, cu excepţia ACP_{ρ}, TeCCS şi U-LOTUS în care se folosesc ***încuieri de timp*** (*time locks*) pentru a detecta inconsistenţele din specificaţii.

Toate au proprietatea de ***urgenţă a acţiunii***, adică anumite acţiuni trebuie declanşate fără întârziere, cu observaţia că în TCSP, TiCCS şi TPL, acestea sunt numai acţiunile invizibile.

TCSP, TiCCS şi U-LOTUS au proprietatea de ***persistenţă contra-intuitivă*** (conterintuitive persistence), prin faptul că progresul timpului nu poate suprima abilitatea de a executa o acţiune.

Bibliografie

[Abr96] **S. Abramsky** - *Retracing some paths in process algebra*, Lecture Notes in Computer Science, 1119:1-17, Springer-Verlag, 1996.

[ACS96] **R. Amadio, I. Castellani, and D. Sangiorgi** - *On bisimulations for the asynchronous π-calculus*, Lecture Notes in Computer Science, 1119:147-162, Springer-Verlag, 1996.

[AD94] **R. Alur and D. Dill** - *The theory of timed automata*, TCS, 126(2), 1994.

[dC+83] **F. De Cindio, G. De Michelis, L. Pomello, and C. Simone** - *Milner's communicating systems and Petri nets*, The 3rd European Workshop on Applications and Theory of Petri nets, Informatik-Fachberichte 66, Springer-Verlag, 1983.

[DNM88] **P. Degano, R. de Nicola and U. Montanari** - *A distributed operational semantics for CCS based on condition/event systems*, Acta Informatica, 26(1/2):59:91, 1988.

[GM84] **U. Goltz and A. Mycroft** - *On the relationship of CCS and Petri nets*, Lecture Notes in Computer Science 172, Springer-Verlag, 1984.

[Gup94] **V. Gupta** - *Chu spaces*, PhD thesis, Stanford University, 1994.

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

[HMP93] **T. A. Henzinger, Z. Manna and A. Pnueli** - *Timed transition systems*, Lecture Notes in Computer Science, Springer-Verlag, 1993.

[Mil93] **R. Milner** - *Action calculi, or syntactic action structures*, Lecture Notes in Computer Science, 711:105-121, Springer-Verlag, 1993.

[LL03] **L. Luca** – *Spații de procese fuzzy*, Ed. Mirton, Timişoara, 2003.

[Plo81] **G. D. Plotkin** - *A structural approach to operational semantics*, Technical Report DAIMI FN-19, Computer Science Department, Aarhus University, 1981.

[Win89] **G. Winskel** - *An introduction to event structures*, Lecture Notes in Computer Science, 354:364-397, Springer-Verlag, 1989.

[Win93] **G. Winskel** - *The formal semantics of programming languages*, The MIT Press, 1993.

[WN95] **G. Winskel and M. Nielsen** - *Models for concurrency*, Handbook of Logic in Computer Science, 4:1-148, Oxford University Press, 1995.