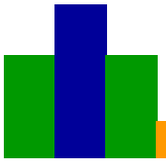


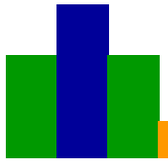
Lecţia 3:

Sisteme complexe: MSC-uri extinse, griduri, RV-scenarii

v1.0

Gheorghe Stefanescu — Universitatea Bucureşti

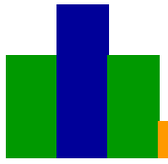
Metode de Dezvoltare Software, Sem.2
Februarie 2007— Iunie 2007



MSC-uri extinse, griduri, RV-scenarii

Cuprins:

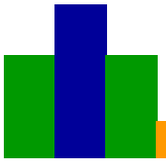
- *Generalităţi*
- MSC-uri şi extensii
- Griduri
- RV-scenarii
- Concluzii, diverse, etc.



Generalităţi

Specificarea sistememlor complexe

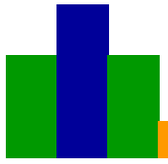
- mai multe procese/obiecte contribuie, prin *interacţie*, la realizarea sistemului
- este *dificil* de specificat *comportamentul complet* al sistemului
- de regulă, se identifică anumite *scenarii*, anume moduri tipice de comportament (execuţie) care *sunt* ori *pot fi* fie *permise*, fie *respinse* de sistemul software



MSC-uri extinse, griduri, RV-scenarii

Cuprins:

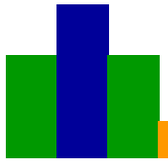
- Generalităţi
- *MSC-uri şi extensii*
  - *MSC-uri (Message Sequence Charts)*
  - HMSC-uri (High-level MSC-uri)
  - CHMSC-uri (Compositional HMSC-uri)
  - LSC-uri (Live Sequence Charts)
- Griduri
- RV-scenarii
- Concluzii, diverse, etc.



MSC-uri (Message Sequence Charts)

MSC-uri (Message Sequence Charts)

- descriu *interacţia* dintre obiecte
- introduse de ITU (International Telecommunication Union) în conjuncţie cu limbajul SDL
- “standard”: ITU-T Recommendation Z.120 (1996)
- curent, foarte populare prin încorporarea lor în UML
- admit atât o *descriere grafică*, cât şi una *textuală*
- cunoscute şi ca “sequence diagrams” ori “timing diagrams”



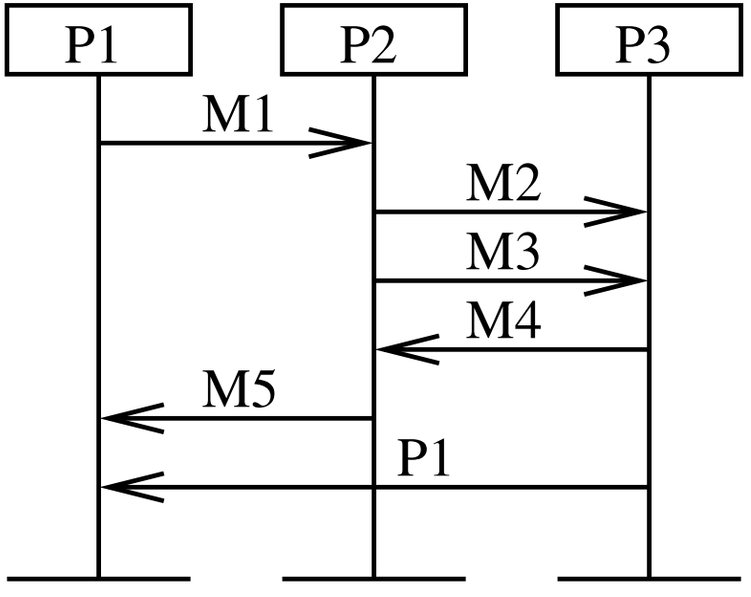
MSC-uri (Message Sequence Charts)

Definiţie

- o MSC descrie un scenariu unde mai multe *procese* comunică între ele prin *mesaje*
- sunt incluse descrieri pentru: *mesaje trimise* (send), *mesaje primite* (receive), *evenimente locale* (calcule), şi ordinea lor în timp
- în versiunea grafică, un *proces* este reprezentat printr-o linie *verticală* (reprezentând evoluţia sa în timp), iar un *mesaj* printr-o linie *orizontală ori oblică* de la transmiţător la destinatar

MSC-uri (Message Sequence Charts)

Exemplu (stânga grafic, dreapta textual)



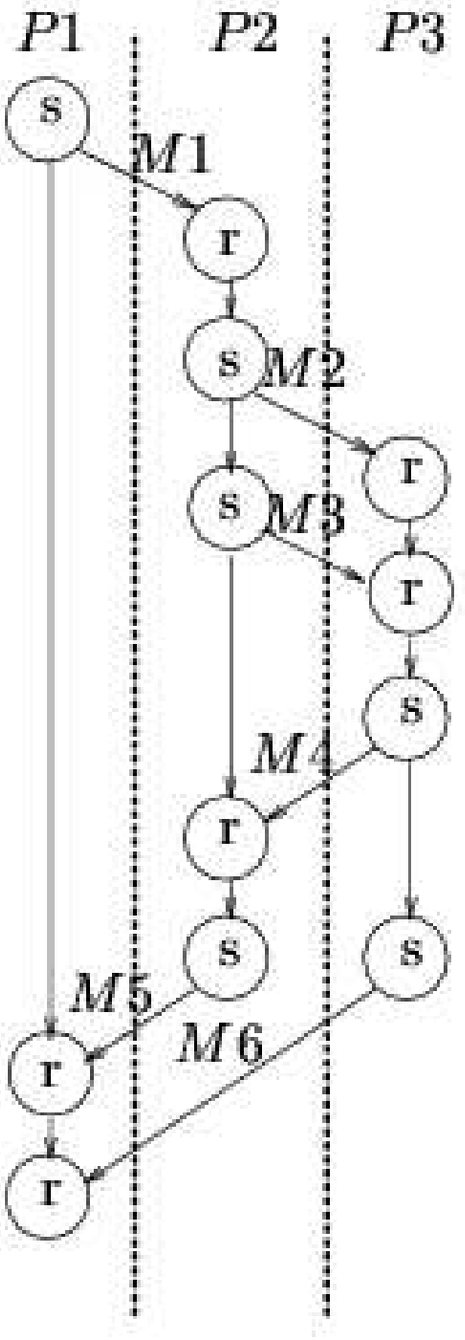

```
msc MSC;
inst
P1:  process Root,
P2:  process Root,
P3:  process Root;
instance P1;
out M1 to P2;
in M5 from P2;
in M6 from P3;
endinstance;
```

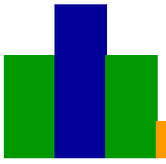
```
instance P2;
in M1 from P1;
out M2 to P3;
out M3 to P3;
in M4 from P3;
out M5 to P1;
endinstance;
instance P3;
in M2 from P2;
in M3 from P2;
out M4 to P2;
out M6 to P1;
endinstance;
endmsc;
```

MSC-uri (Message Sequence Charts)

Exemplu Relaţia de ordine între evenimentele unui MSC

- s (resp. r) înseamnă "send" (resp. "receive")
- relaţia de ordine "$\leq$" este indusă de relaţia de ordine imediată "$<$", definită prin:
  - într-un proces, $a < b$ dacă b urmează imediat după a;
  - pentru un mesaj $s \mapsto r$, am $s < r$
- de notat ca evenimentele send (notate s) pentru $M5, M6$ sunt incomparabile

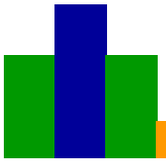




MSC-uri extinse, griduri, RV-scenarii

Cuprins:

- Generalităţi
- *MSC-uri şi extensii*
  - MSC-uri (Message Sequence Charts)
  - *HMSC-uri (High-level MSC-uri)*
  - CHMSC-uri (Compositional HMSC-uri)
  - LSC-uri (Live Sequence Charts)
- Griduri
- RV-scenarii
- Concluzii, diverse, etc.



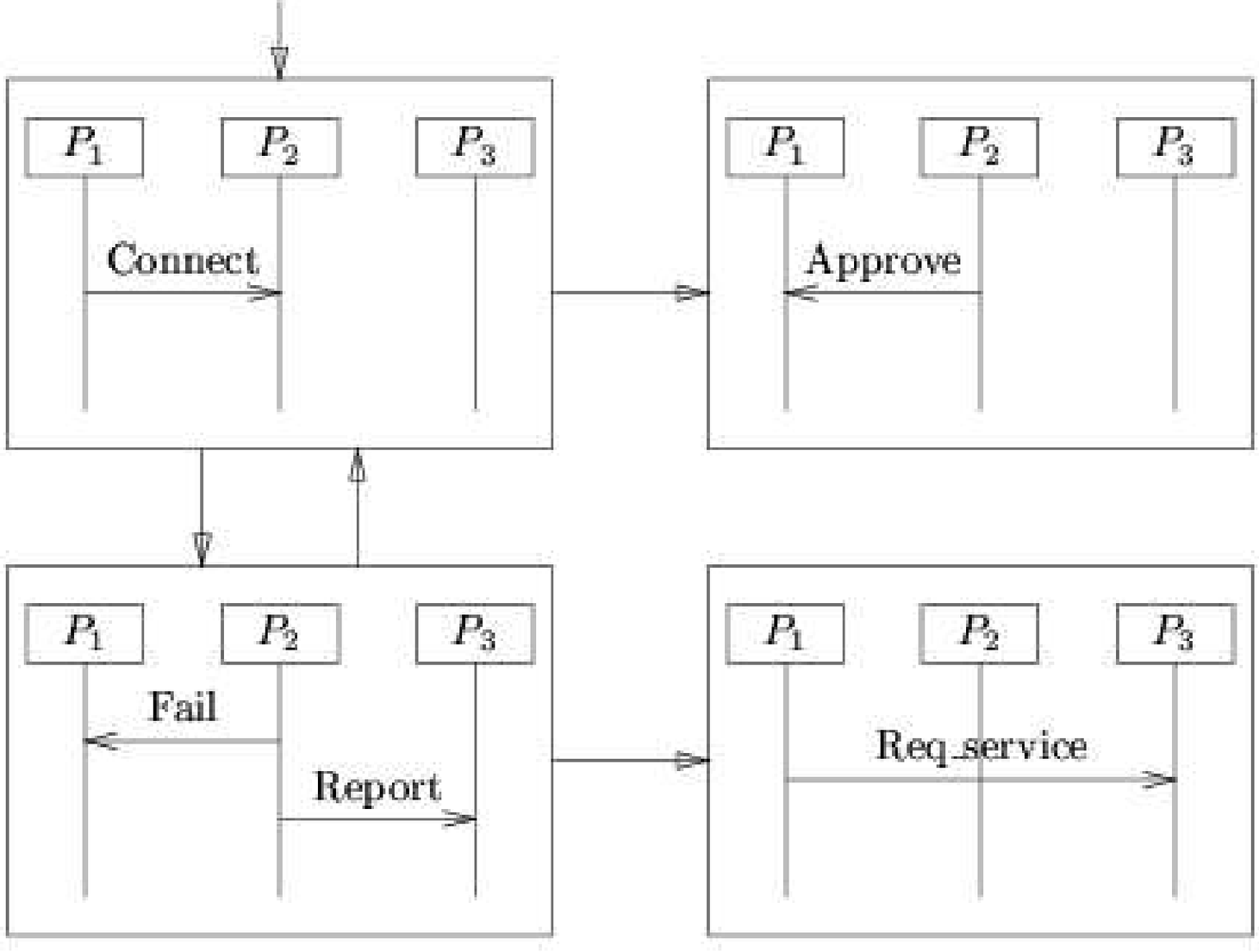
HMSC-uri (High-level MSC-uri)

Generalităţi

- în genere, pentru specificarea unui sistem se folosesc *mulţimi de scenarii*
- *HMSC-uri (High-level MSC-uri)* permit definirea de mulţimi de scenarii folosind un fel de scheme logice *MSG-uri* (Message Sequence Graph) în care nodurile sunt MSC-uri
- scenariile asociate unui MSG se obţin astfel:
  - luăm un drum maximal din graf, ori, dacă are intrări-ieşiri, un drum de la intrare la ieşire
  - "compunem" MSC-urile din nodurile grafului prin care se trece (compunerea revine la un fel de concatenare a MSC-urilor "lipind" MSC-urile unele după altele prin procese)

HMSC-uri (High-level MSC-uri)

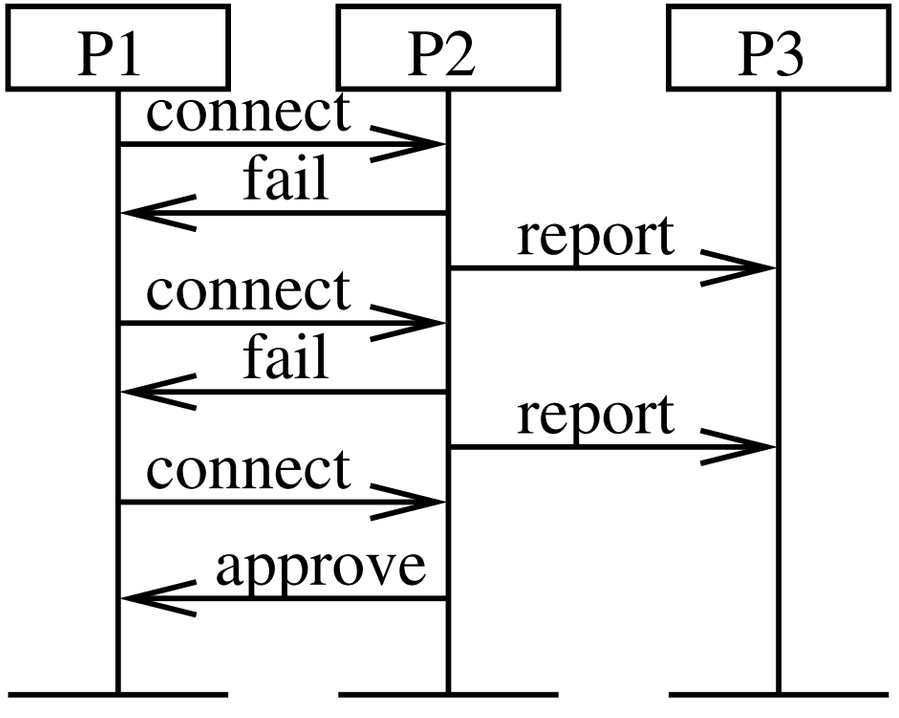
Exemplu (încercare de conectare la internet)

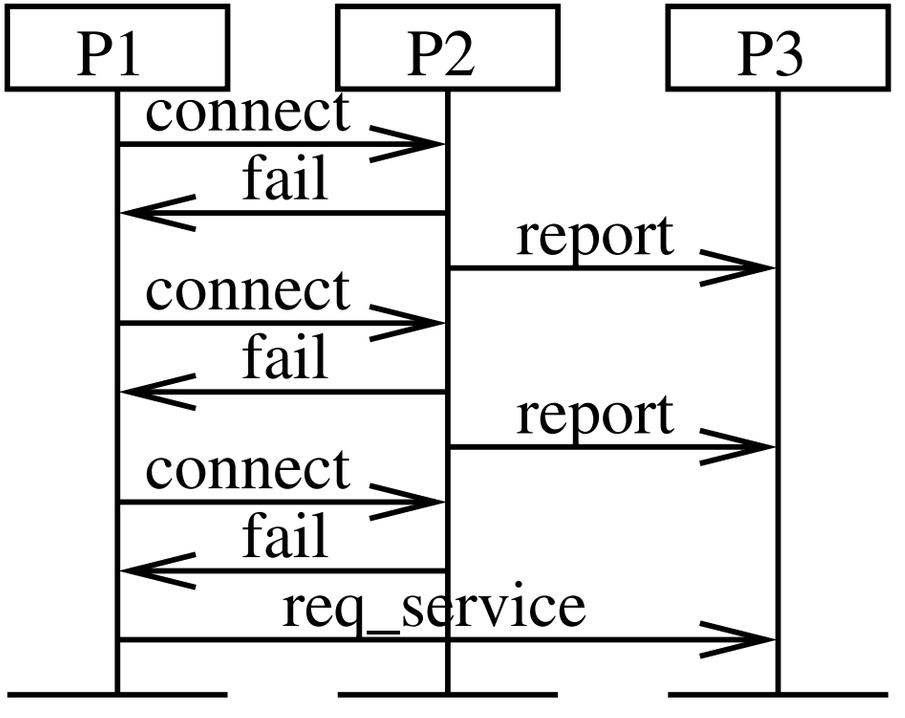


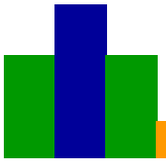
HMSC-uri (High-level MSC-uri)

Exemple de MSC-uri generate de MSG-ul de mai sus:

- în stânga avem un caz cu două eşecuri, urmate de succes
- în dreapta avem un caz cu trei eşecuri urmate de cerere de asistenţă







MSC-uri extinse, griduri, RV-scenarii

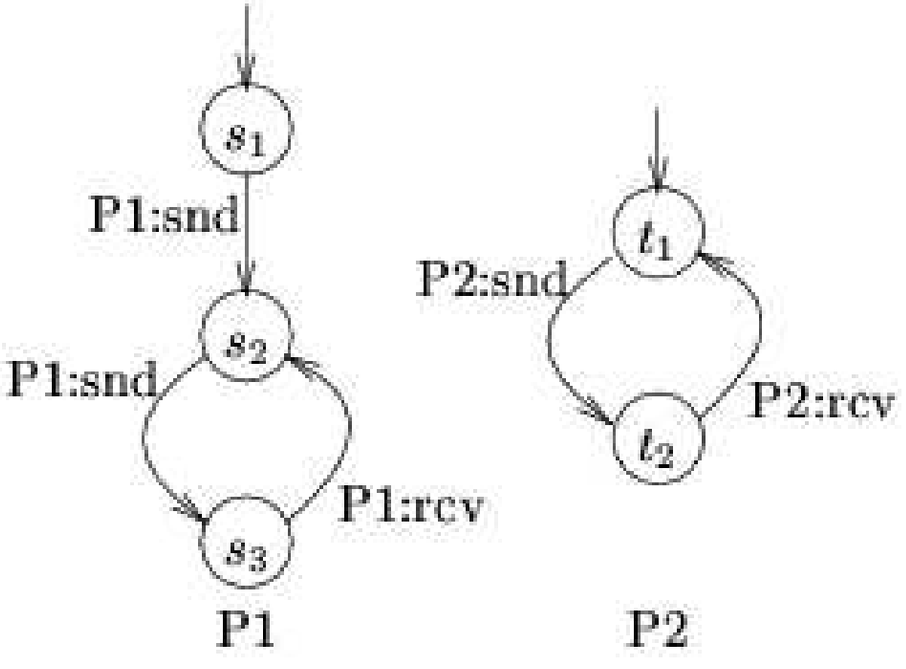
Cuprins:

- Generalităţi
- *MSC-uri şi extensii*
  - MSC-uri (Message Sequence Charts)
  - HMSC-uri (High-level MSC-uri)
  - *CHMSC-uri (Compositional HMSC-uri)*
  - LSC-uri (Live Sequence Charts)
- Griduri
- RV-scenarii
- Concluzii, diverse, etc.

CHMSC-uri (Compositional HMSC-uri)

CFM (communicating finite state machines)

- folosite pentru descrierea protocoalelor de comunicare
- constă din automate finite în care procesele comunică prin mesaje via canale FIFO

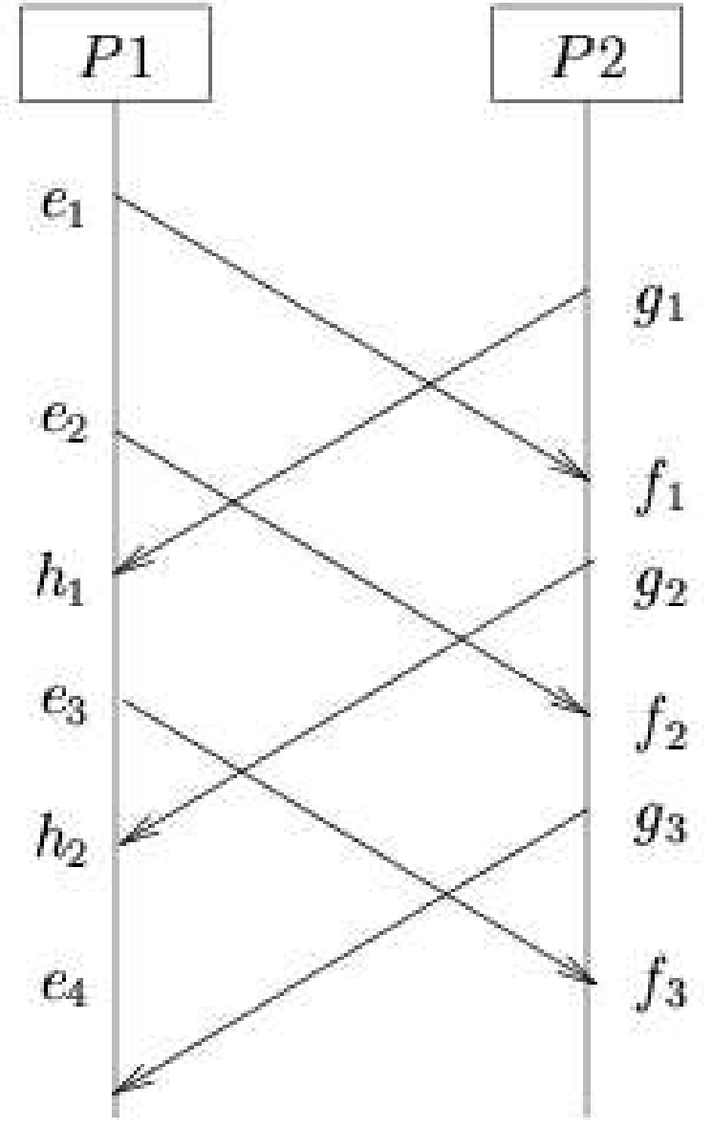


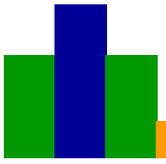
In exemplu, avem o variantă a lui ABP (*alternating bit protocol*); send-urile din $P1$ comunică cu receive-urile din $P2$ şi reciproc.

CHMSC-uri (Compositional HMSC-uri)

De la CFM-uri la MSC-uri

- unui CFM (automat cu comunicare) i se poate asocia o mulţime de MSC-uri care descriu evoluţia sa
- evoluţia CFM-ului din exemplul anterior este dată de un MSC-ul infinit, al cărui început este ilustrat în desen
- perechile (e, f) reprezintă comunicări send-receive de la $P1$ la $P2$, iar (g, h) similar de la $P2$ la $P1$





CHMSC-uri (Compositional HMSC-uri)

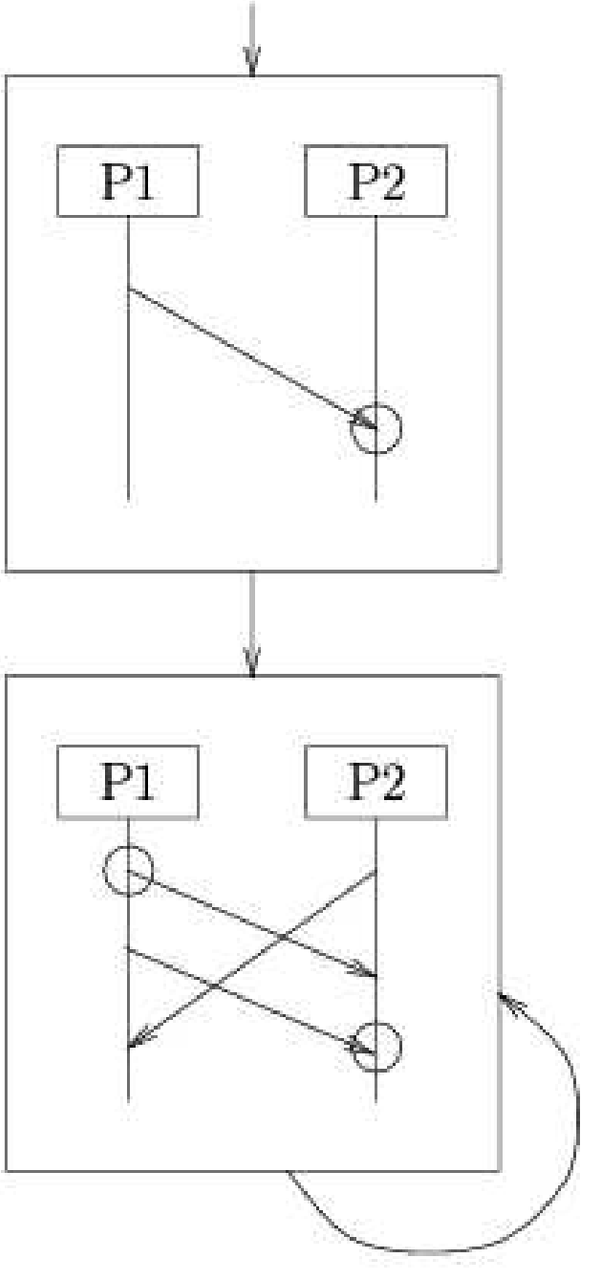
Limitare HMSC

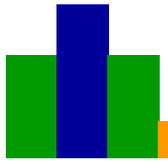
- nu există un HMSC care să definească scenariul ABP obţinut mai sus
- explicaţie: în fiecare nod MSC-urile sunt finite; apoi, orice tăietură a scenariului lasă un mesaj de comunicare cu send-ul într-o componentă şi receive-ul în alta
- ori, în MSG/HMSC se insistă să avem MSC-uri complete în fiecare nod, anume orice send să aibă un receive asociat în acel nod

CHMSC-uri (Compositional HMSC-uri)

CHMSC-uri (Compositional HMSC-uri)

- această extensie permite MSC-uri incomplete în nodurile unui HMSC
- exemplul din dreapta descrie un CHMSC pentru ABP
- send-urile incomplete dintr-un nod se cuplează, în ordine, cu receive-uri incomplete din nodurile care urmează





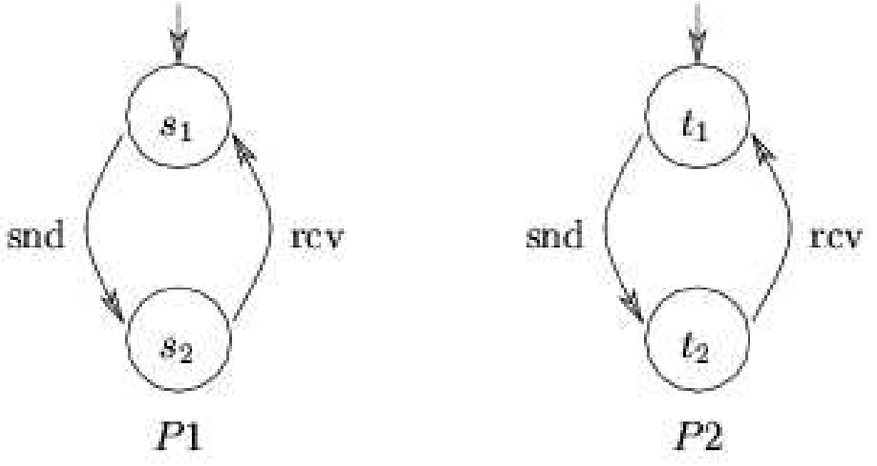
CHMSC-uri (Compositional HMSCs)

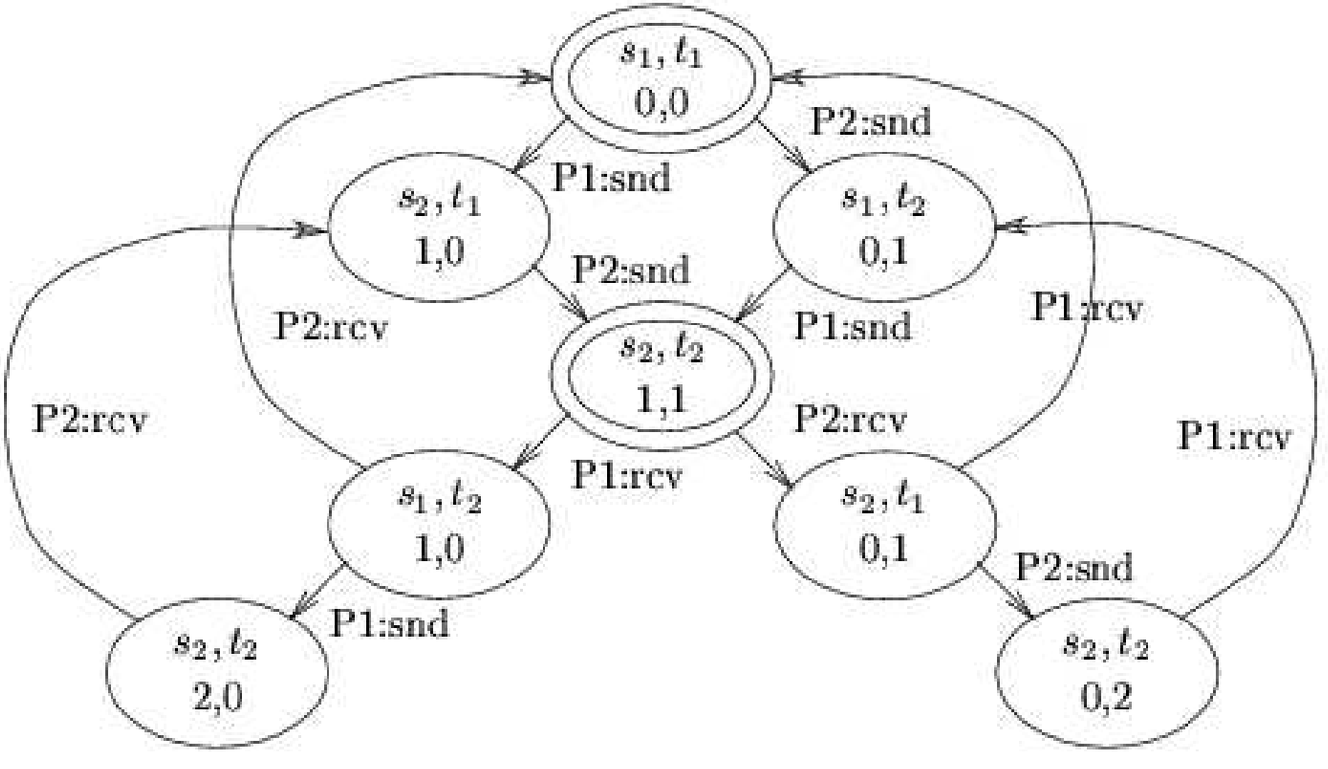
Probleme tehnice cu CHMSC-urile

- pot apare send-uri fară receive (ok), dar şi receive-uri fără send (!?)
- unele proprietăţi simple pentru HMSC devin complicate aici; de exemplu, proprietatea că “pentru un mesaj send există un MSC în care există şi un receive asociat” este banal decidabilă pentru HMSC-uri, dar nedecidabilă pentru CHMSC
- există o subclasă de *CHMSC-uri relizabile* care evită problema de mai sus şi sunt suficiente în descrierea protocoalelor de comunicare

MSC-uri (Message Sequence Charts)

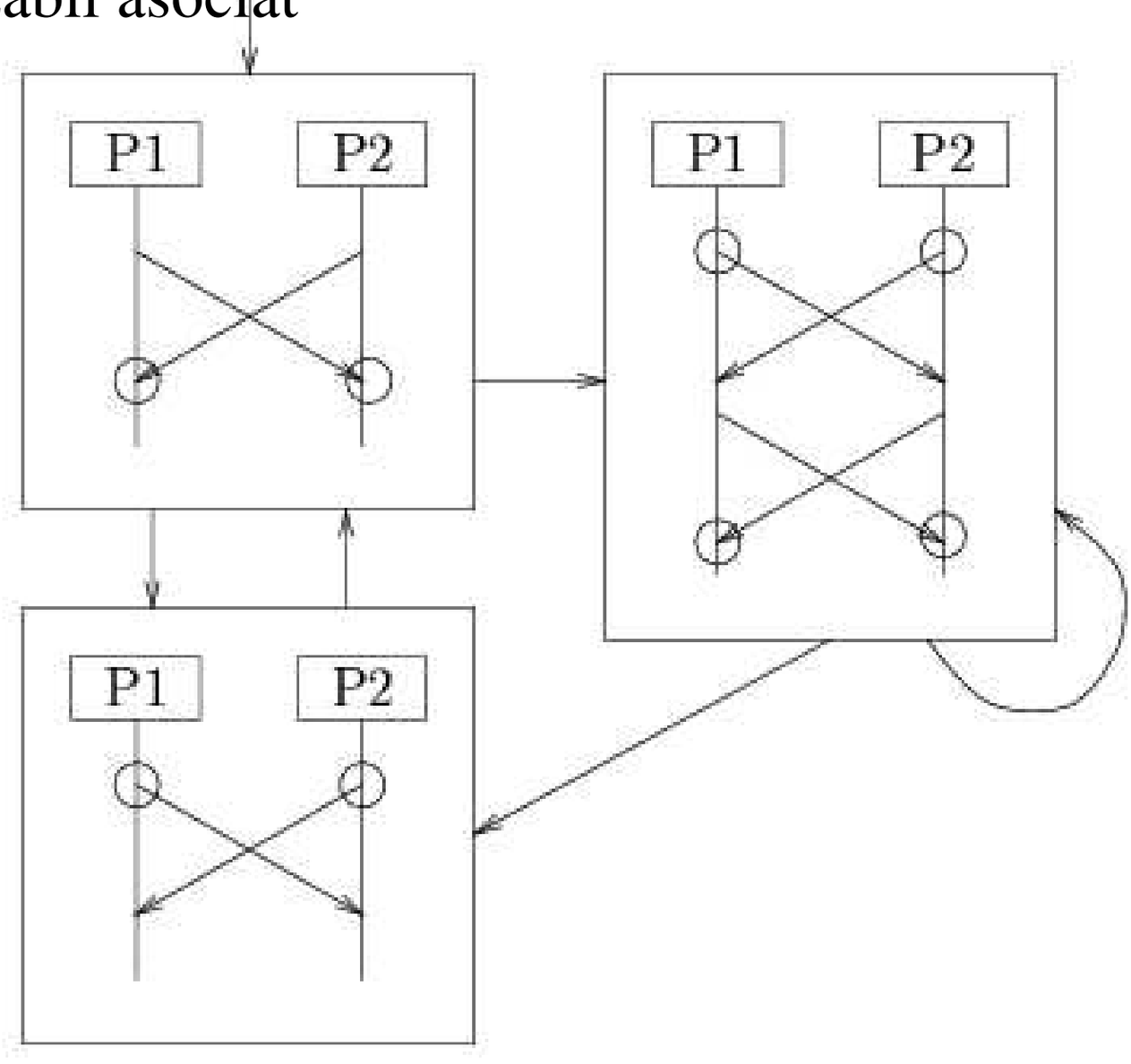
Un protocol...

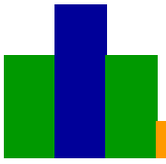




MSC-uri (Message Sequence Charts)

...şi CHMSC-ul relizabil asociat

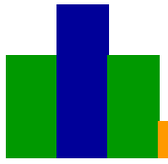




MSC-uri extinse, griduri, RV-scenarii

Cuprins:

- Generalităţi
- *MSC-uri şi extensii*
  - MSC-uri (Message Sequence Charts)
  - HMSC-uri (High-level MSC-uri)
  - CHMSC-uri (Compositional HMSC-uri)
  - *LSC-uri (Live Sequence Charts)*
- Griduri
- RV-scenarii
- Concluzii, diverse, etc.



MSC-uri (Live Sequence Charts)

Idee

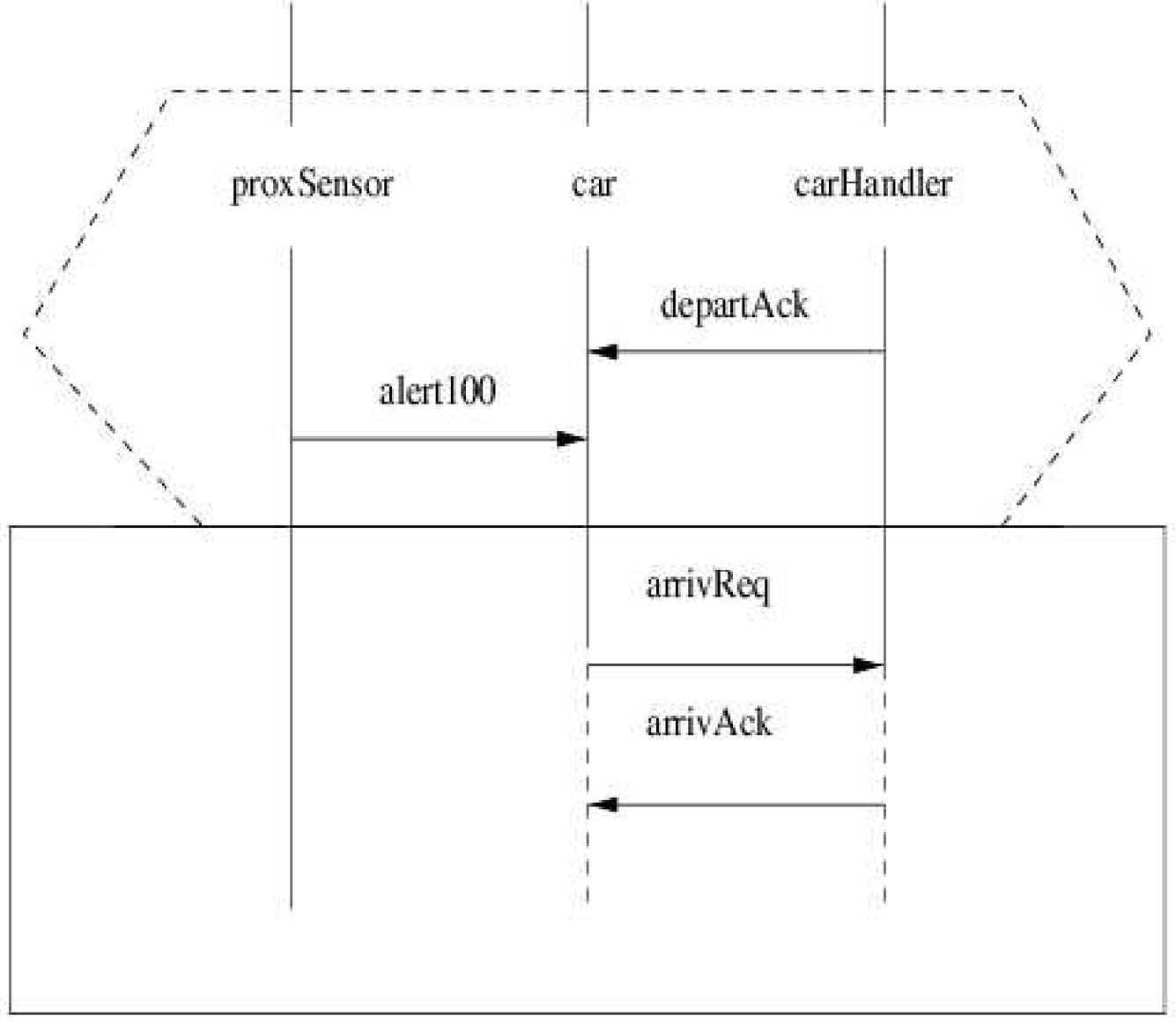
- LSC-urile aduc în discuţie distincţia dintre *trebuie* şi *este posibil*
- există adnotări specifice care să indică că un fragment de MSC trebuie să fie executat (în genere, desenat cu *linii continue*) ori numai că este posibil (cu *linii întrerupte*)
- se folosesc cuvintele *hot* pentru linii continue şi *cold* pentru punctate
- se folosesc şi *pre-chart*-uri cu sens condiţional (dacă un pre-chart are loc, atunci se execută ce urmează), notate cu romburi

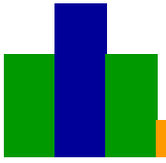
MSC-uri (..Live Sequence Charts)

Exemplu de LSC

— dacă condiţia specificată în romb are loc, se face ce urmează

— liniile întrerupte de jos indică situaţia când un mesaj trimis poate să nu fie recepţionat





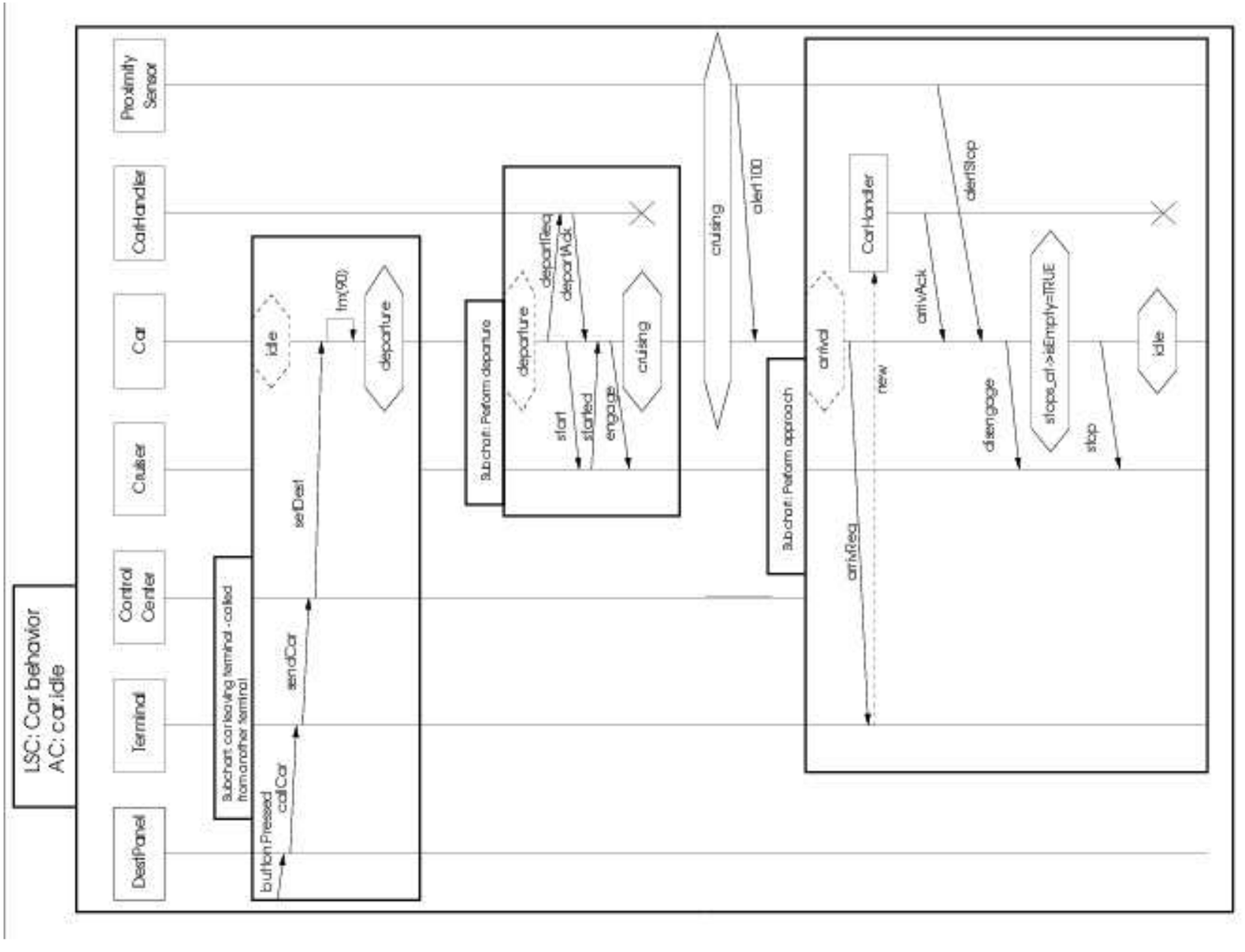
MSC-uri (..Live Sequence Charts)

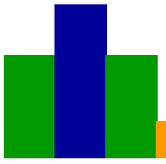
Distincţii

		Obligatoriu	Posibil
chart	mod	universal	existenţial
	semantică	toate execuţiie satisfac char-ul	există o execuţie care satisface char-ul
locaţie	mod	hot	cold
	semantică	instanţa trece sigur de locaţie	instanţa nu trebuie neapărat să treacă de locaţie
mesaj	mod	hot	cold
	semantică	mesajul trebuie să ajungă la destinaţie	mesajul ne trebuie să ajungă neapărat la destinaţie
condiţie	mod	hot	cold
	semantică	condiţia trebuie satisfăcută, altfel abort	dacă condiţia nu-i satisfăcută se iese din sub-chart

MSC-uri (..Live Sequence Charts)

Exemplu de LSC (mai complex)

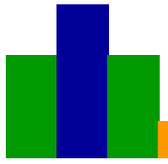




MSC-uri extinse, griduri, RV-scenarii

Cuprins:

- Generalităţi
- MSC-uri şi extensii
- *Griduri*
  - *Dualitatea spatiu-timp (pe scurt)*
  - Griduri şi expresii regulate 2-dimensionale
  - Operatorul de platizare
  - Modelarea *inter*-actiunii prin griduri
  - Griduri vs. MSC-uri
- RV-scenarii
- Concluzii, diverse, etc.



Dualitatea spatiu-timp

Dualitatea spatiu-timp:

- bazată pe *teza* că mediul de *"calculabilitate globală"* (anume calculabilitate în reţea) este *invariant relativ la dualitatea spaţiu-timp*
- prin dualitatea spaţiu-timp, paradigma clasică de *calculabilitate în timp* se transformă în paradigma de *calculabilitate în spaţiu* şi vice-versa
- interesul este de *combina ambele paradigme*

Figura pentru ST-Dualitate

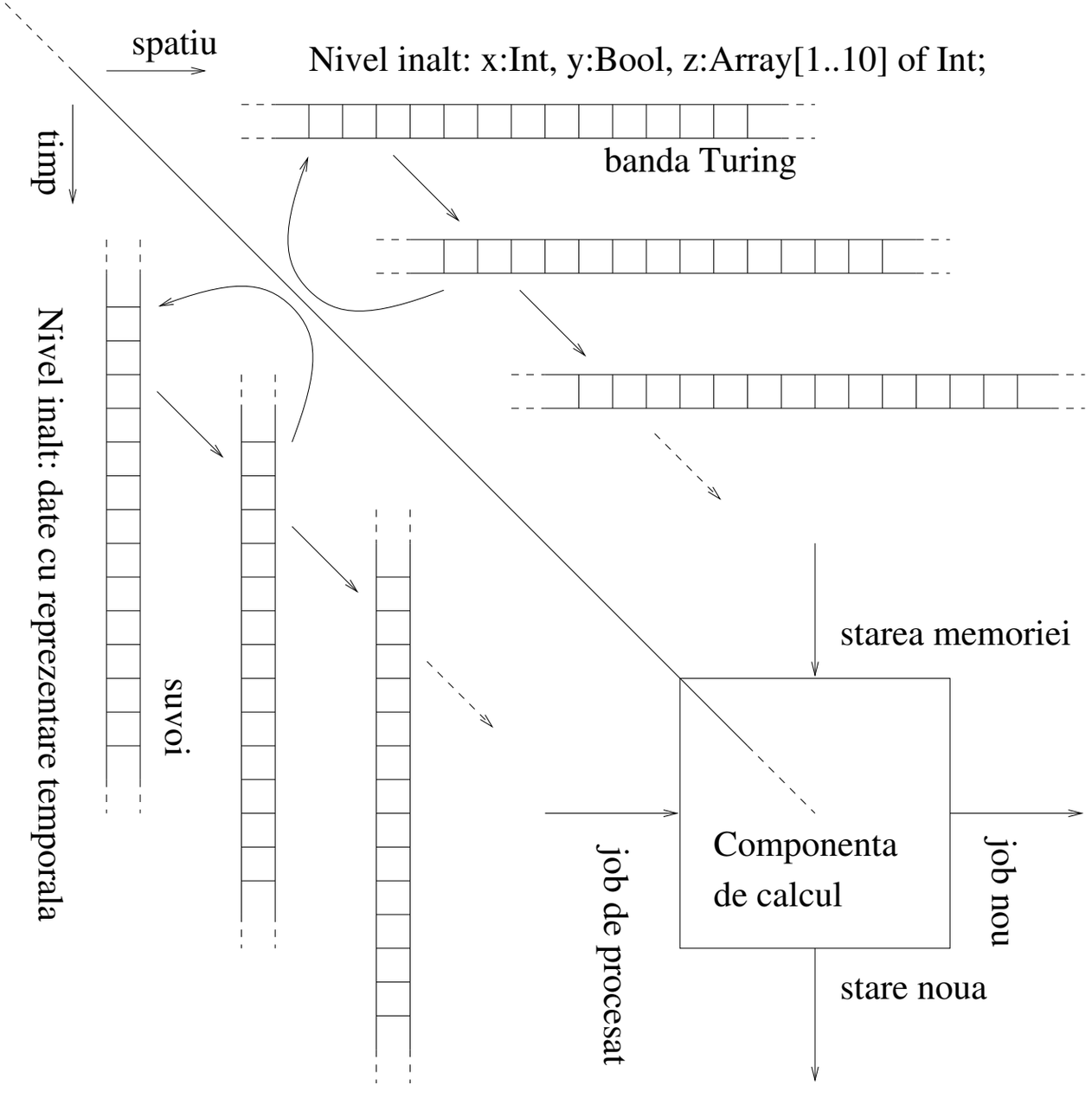


Figura pentru
ST-Dualitate

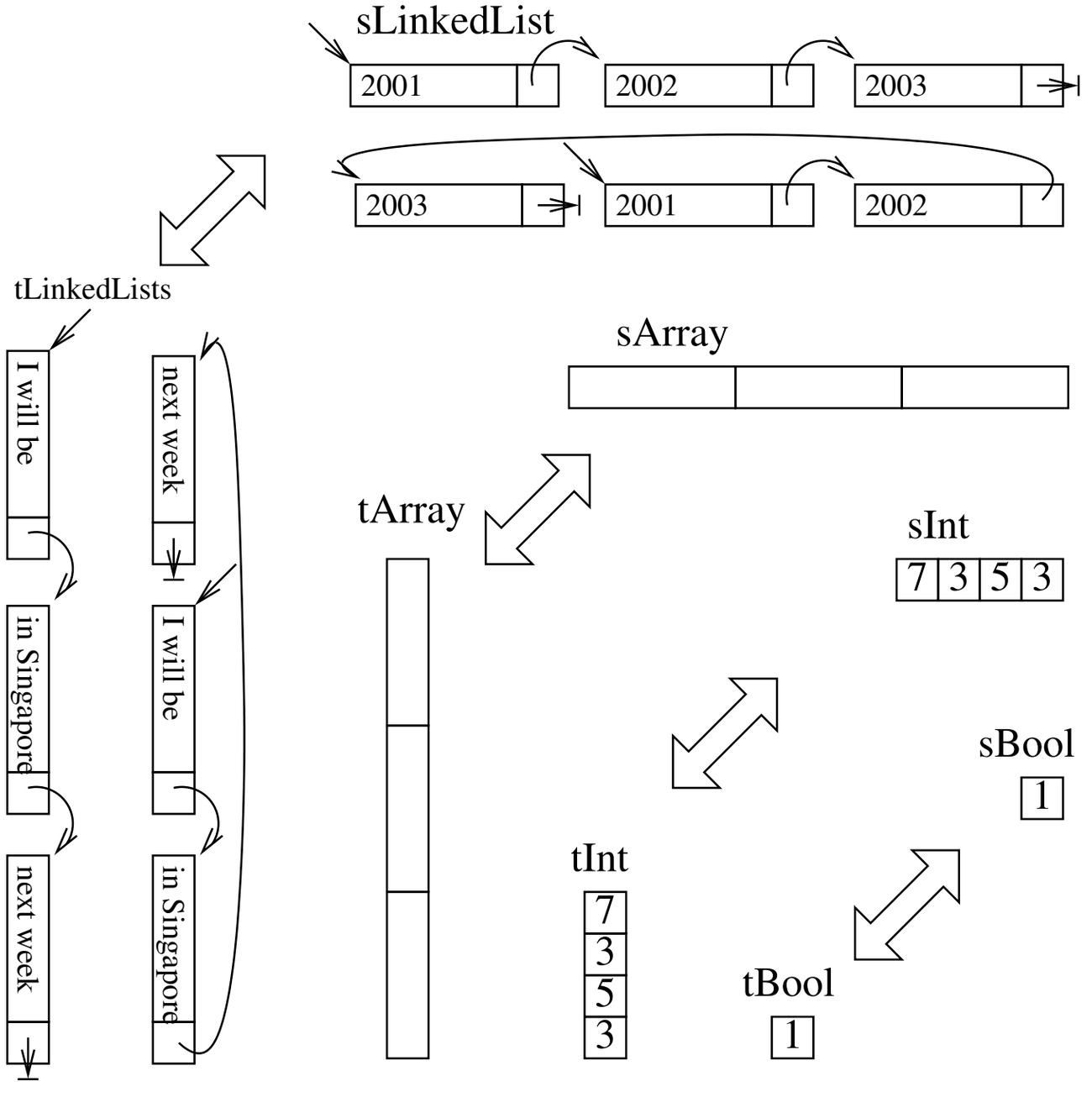
Structuri temporale de nivel înalt

date cu reprezentarea lor uzuală (*spaţială*):

sBool, sInt, sArray, sLinkedList, etc.

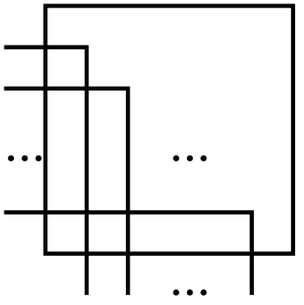
şi dualul lor *temporal* (i.e., date cu *reprezentare temporală*):

tBool, tInt, tArray, tLinkedList, etc.

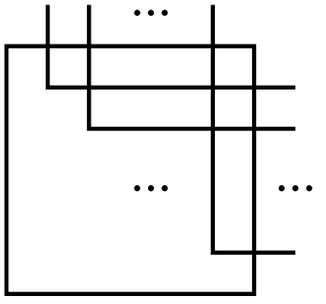


Convertori spatiu-timp; calculabilitate în spatiu

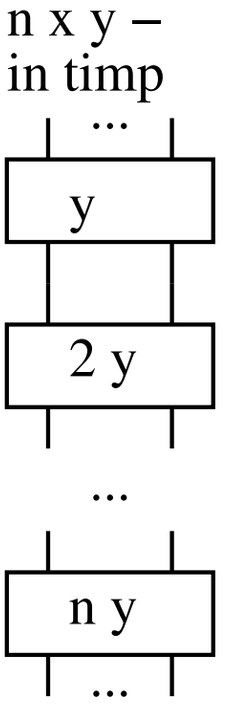
convertori
spatiu-timp
şi
timp-spatiu
folosiţi pentru
a transforma
calculabilitatea
în timp
în
calculabilitate
în spaţiu



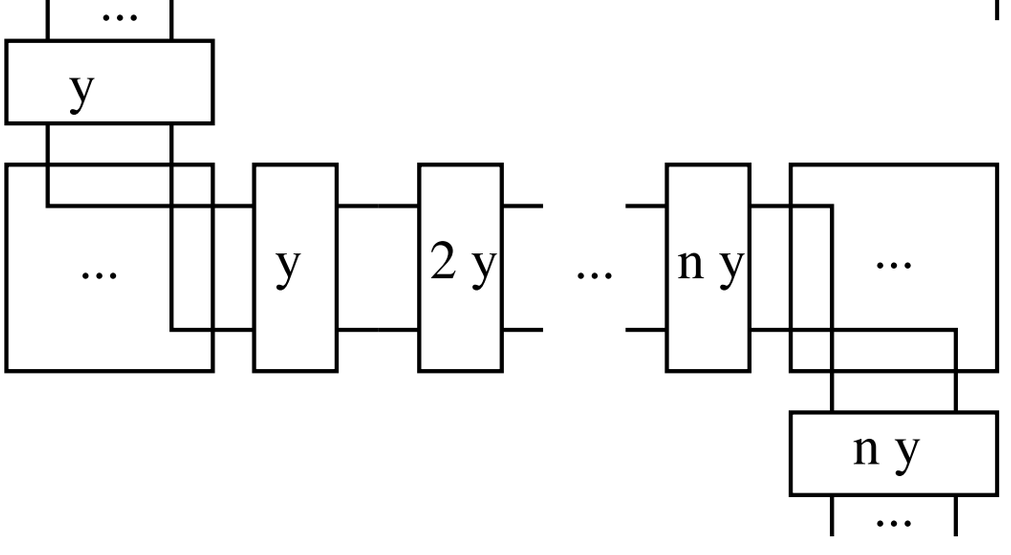
timp–spatiu

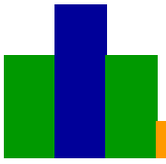


spatiu–timp



n x y – calculat in spatiu

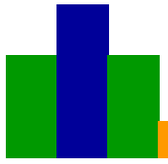




MSC-uri extinse, griduri, RV-scenarii

Cuprins:

- Generalităţi
- MSC-uri şi extensii
- *Griduri*
  - Dualitatea spatiu-timp (pe scurt)
  - *Griduri şi expresii regulate 2-dimensionale*
  - Operatorul de platizare
  - Modelarea *inter*-actiunii prin griduri
  - Griduri vs. MSC-uri
- RV-scenarii
- Concluzii, diverse, etc.



Griduri (ori cuvinte planare)

Un *grid* (ori *cuvânt planar*) este

- o *arie 2-dimensională* dreptunghiulară
- umplută cu *litere* dintr-un alfabet dat

Exemple:

```
aabbabb
abbcdbb
bbabbca
ccccaaa
```

(neutilizate aici:

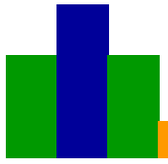
```
aabb...
..bc..b
bbabbca
..c...a
```

)

Un grid p are o bordură de *nord* (resp. *sud, vest, est*) notată cu

$$n(p) \quad \text{(resp.} \quad s(p), w(p), e(p))$$

Notă: Cerinţa de a avea o arie dreptunghiulară poate fi slăbită; de exemplu, se poate cere o arie conexă, dar nu neapărat dreptunghiulară.



Operatorul de platizare

Operatorul de platizare

$$\flat : \text{LangGrids}(V) \quad \rightarrow \quad \text{LangWords}(V)$$

duce

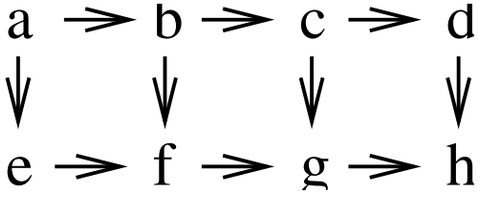
mulţimi de *griduri*

în

mulţimi de cuvinte reprezentând sortarile lor topologice.

..Operatorul de platizare

- un grid dă un *graf direcţionat aciclic* desenând:
  – arcuri orizontale de la o literă la vecina din dreapta şi
  – arcuri verticale de la o literă la vecina de jos.
- astfel de grafuri (şi subgrafurile lor) au *noduri minimale* (noduri în care nu intră arce).



- procedura de *sortarea topologică* constă în:
  (1) selecţia unui nod minimal, apoi
  (2) ştergerea lui şi a arcelor care pleacă din el, şi
  (3) repetarea paşilor 1-2 cât timp este posibil;
  se obţine un cuvânt uzual (secvenţă).
- $\flat$ *pe un grid w*: $\flat(w)$ constă din toate cuvintele care se obţin cu procedura de mai sus variind nodul selecat la fiecare pas
- $\flat$ *pe un limbaj L*: $\flat(L) = \{\flat(w), w \in L\}$.

..Operatorul de platizare

Exemple:

– plecăm cu $\begin{matrix}\texttt{abcd}\\ \texttt{efgh}\end{matrix}$;

– există un singur element minimal `a`;

după ştergerea lui obţinem $\begin{matrix}\texttt{bcd}\\ \texttt{efgh}\end{matrix}$;

– elementele minimale sunt `b` şi `e`; să zicem că alegem `b`;

rămâne $\begin{matrix}\texttt{cd}\\ \texttt{efgh}\end{matrix}$;

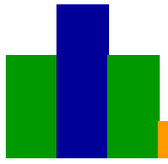
– acum elementele minimale sunt `c` şi `e`; să alegem `e`; obţinem $\begin{matrix}\texttt{cd}\\ \texttt{fhi}\end{matrix}$;

– şi aşa mai departe;

– în final obţinem un cuvânt, să zicem `abecfgdh`.

De fapt, $\flat\left(\begin{matrix}\texttt{abcd}\\ \texttt{efgh}\end{matrix}\right) =$

$\{$`abcdefgh, abcedfgh, abcefdgh, abcefgdh, abecdfgh,`
`abecfdgh, abecfgdh, abefcdgh, abefcgdh, aebcdfgh,`
`aebcfdgh, aebcfgdh, aebfcdgh, aebfcgdh`$\}$



Expresii regulare 2-dimensionale

Signatură: *se iau două seturi de operatori regulaţi care au aceeaşi parte aditivă*

$$+,\ 0,\ \cdot,\ ^\star,\ |,\ \triangleright,\ ^\dagger,\ -$$

– $(+, 0, \cdot, ^\star, |)$ - o signatură Kleene pentru direcţia verticală
– $(+, 0, \triangleright, ^\dagger, -)$ - o signatură Kleene pentru direcţia orizontală

Expresii regulate 2-dimensionale simple:

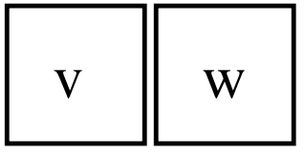
$$E ::= a(\in V) \mid 0 \mid E + E \mid E \cdot E \mid E^\star \mid | \mid E \triangleright E \mid E^\dagger \mid -$$

Mulţimea lor se notează cu *simple*-2RegExp(V).

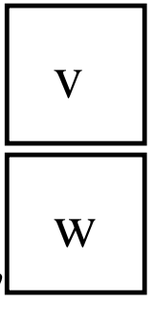
Expresiile regulate 2-dimensionale: adaugăm *intersecţia* "$\cap$"; mulţimea lor se notează cu 2RegExp(V).

Compunere şi identităţi pe griduri

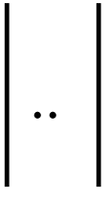
- *compunerea orizontală* $v \triangleright w$
  - este definită numai dacă $e(v) = w(w)$
  - $v \triangleright w$ este cuvântul obţinut punând v la stânga lui w



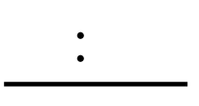
- *compunerea verticală* $v \cdot w$
  - este definită numai dacă $s(v) = n(w)$
  - $v \cdot w$ este cuvântul obţinut punând v deasupra lui w

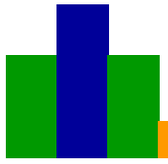


- *identitatea verticala* ε_k este:
  - cu $w(\varepsilon_k) = e(\varepsilon_k) = 0$ şi $n(\varepsilon_k) = s(\varepsilon_k) = k$



- *identitatea orizontală* λ_k este:
  - cu $w(\lambda_k) = e(\lambda_k) = k$ şi $n(\lambda_k) = s(\lambda_k) = 0$





De la expresii la mulţimi de griduri

Interpretarea (de la *expresii* la *mulţimi de griduri*) este

$$|\quad| : 2\mathsf{RegExp}(V) \quad \rightarrow \quad \mathit{LangGrids}(V)$$

- $|a| = \{a\}$; $|0| = \emptyset$; $|E+F| = |E| \cup |F|$; $|E \cap F| = |E| \cap |F|$;
- $|E \cdot F| = \{v \cdot w : v \in |E| \ \& \ w \in |F|\}$;
- $|E^\star| = \{v_1 \cdot \ldots \cdot v_k : k \in \mathbb{N} \ \& \ v_1, \ldots, v_k \in |E|\}$;
- $|_|| = \{\varepsilon_0, \ldots, \varepsilon_k, \ldots\}$;
- $|E \triangleright F| = \{v \triangleright w : v \in |E| \ \& \ w \in |F|\}$;
- $|E^\dagger| = \{v_1 \triangleright \ldots \triangleright v_k : k \in \mathbb{N} \ \& \ v_1, \ldots, v_k \in |E|\}$;
- $|-| = \{\lambda_0, \ldots, \lambda_k, \ldots\}$.

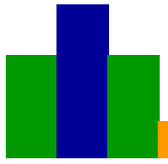
Exemple: expresii şi limbaje de griduri

- $(a \cdot d^\star \cdot g) \triangleright (b \cdot e^\star \cdot h)^\dagger \triangleright (c \cdot f^\star \cdot i)$
  reprezintă gridurile
  ```
  a b . . b c
  d e . . e f
  . . . . . .
  . . . . . .
  d e . . e f
  g h . . h i
  ```
  ;
  $(a \triangleright b^\dagger \triangleright c) \cdot (d \triangleright e^\dagger \triangleright f)^\star \cdot (g \triangleright h^\dagger \triangleright i)$ este o expresie echivalentă.

- $a^\dagger \cdot b^\dagger$ reprezintă gridurile
  ```
  aa..a
  bb..b
  ```
  ;
  versiunea platizată este limbaj context-free, dar neregulat

$$\flat(a^\dagger \cdot b^\dagger) = \{w \in \{a,b\}^* : |w|_a = |w|_b \quad \& \quad \forall w = w'w'' : |w'|_a \geq |w'|_b\}$$

($|w|_a$ reprezintă numărul de apariţii ale lui a în w).



..Exemple: expresii şi limbaje de griduri

- $a^\dagger \cdot b^\dagger \cdot c^\dagger$ reprezintă gridurile $\begin{smallmatrix}\texttt{aa..a}\\ \texttt{bb..b}\\ \texttt{cc..c}\end{smallmatrix}$;

  versiunea platizată nu este nici măcar limbaj context-free

$$\flat(a^\dagger \cdot b^\dagger \cdot c^\dagger) = \{w \in \{a,b,c\}^* : |w|_a = |w|_b = |w|_c \\ \& \quad \forall \quad w = w'w'' : |w'|_a \geq |w'|_b \geq |w'|_c\}$$

- $(a+b)^{\star\dagger} = (a+b)^{\dagger\star}$

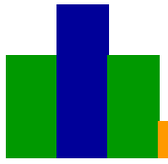
  arată că starurile orizontal şi vertical pot să comute

- $\begin{smallmatrix}\texttt{aa}\\ \texttt{bb}\end{smallmatrix} \in (a+b \triangleright b)^{\dagger\star} \setminus (a+b \triangleright b)^{\star\dagger}$

  arată că, în general, *starurile nu comută*.

- $a^\dagger \cdot b^\dagger \neq a^\star \triangleright b^\star$, *dar* $\flat(a^\dagger \cdot b^\dagger) = \flat(a^\star \triangleright b^\star)$

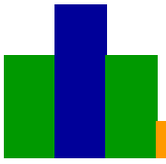
  deci operatorul de platizare poate pierde din informaţie



Expresii regulate extinse

Nevoia de a avea expresii regulate mai puternice:

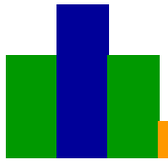
- operatorii de iterare pe $\mathsf{2RegExp}(V)$ *nu sunt suficient de puternici* *[e.g., nu se poate reprezenta limbajul L_{sq} de pătrate de a]*
- trebuie *mai multă libertate* în procesul de iterare
- o posibilitate: *cuvinte şi pavări generale*



MSC-uri extinse, griduri, RV-scenarii

Cuprins:

- Generalităţi
- MSC-uri şi extensii
- *Griduri*
  - Dualitatea spatiu-timp (pe scurt)
  - Griduri şi expresii regulate 2-dimensionale
  - *Operatorul de platizare*
  - Modelarea *inter*-actiunii prin griduri
  - Griduri vs. MSC-uri
- RV-scenarii
- Concluzii, diverse, etc.



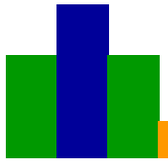
Explozia starilor şi operatorul de platizare

Aparent,

- operatorul de platizare este responsabil de cunoscuta problemă a *exploziei numărului de stări* care apare în verificarea sistemelor concurente, orientate pe obiecte

Speranţa este că

- prin extinderea tehnicilor de verificare de la cuvinte/secvenţe-de-execuţie la griduri/scenarii putem (într-o anumită măsură) evita acest fenomen



Viteză maximă

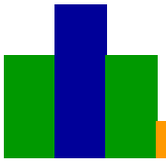
Să presupunem că toţi atomii unui grid w sunt distincţi. Atunci,

> *Propoziţie: Pentru orice $z \in \flat(w)$ se pot asocia timpi de execuţie atomilor astfel ca timpul total de execuţie pentru w dat de planificarea z să fie minim.*

[Reguli de analiză: (1) fiecare atom începe execuţia cât de curând posibil; (2) dacă doi atomi îşi termină execuţia în acelaşi timp, atunci ei pot fi puşi de operatorul de platizare în orice ordine.]

Asta arată că:

- orice procedură de planificare *statică* (e.g., pe linii, pe coloane, pe diagonale, etc.) *nu* asigură *viteza maximă*;
- trebuie considerate *toate cuvintele din platizare* ca posibile secvenţe de execuţie

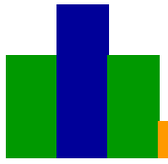


Analiza cantitativă a platizării

Rezultate experimentale (pentru griduri dreptunghiulare $m \times n$):

Numărul de execuţii secvenţiale $\varphi(m,n)$ asociate unui grid $m \times n$:

$m \backslash n$	2	3	4	5	6	7
2	2	5	14	42	132	429
3	—	42	462	6,006	87,516	1,385,670
4	—	—	24,024	1,662,804	140,229,804	13,672,405,890
5	—	—	—	701,149,020	396,499,770,810	278,607,172,289,160
6	—	—	—	—	1,671,643,033,734,960	9,490,348,077,234,178,440
7	—	—	—	—	—	475,073,684,264,389,879,228,560



..Analiza cantitativă a platizării

Rezultate teoretice:

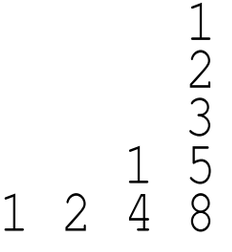
- un *grid parţial* este acea parte dintr-un grid normal care ramâne după ce s-au aplicat un număr de paşi ai procedurii de platizare
- un grid parţial este de *tipul* $(l_1; l_2; \ldots; l_m)$ dacă are l_1 elemente pe linia 1, l_2 pe linia 2, etc., unde $l_1 \leq l_2 \leq \ldots \leq l_m$
- fie $\varphi_{l_1;l_2;\ldots;l_m}$ *numărul de cuvinte* asociate de operatorul de platizare unui grid parţial de tipul $(l_1; l_2; \ldots; l_m)$
- fiecare celulă a_i a unui grid parţial are asociat un număr k_i care reprezintă *suma distanţelor* (numărul de celule) de la a_i la bordurile din nord şi vest, numărând-ul pe a_i o singură dată

..Analiza cantitativă a platizării

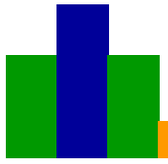
Teoremă: Pentru un grid parţial de tipul $(l_1;l_2;\ldots;l_m)$ *cu* p *celule care au distanţele asociate* $k_1,\ldots,k_p$, *avem*

$$\varphi_{l_1;l_2;\ldots;l_m} = \frac{p!}{k_1 \ldots k_p}$$

Un exemplu este în dreapta:
– are tipul (1;1;1;2;4) (9 celule);
– numerele din celule indică suma distanţelor spre nord şi vest;
– $\varphi_{l_1;l_2;\ldots;l_m} = \frac{9!}{(1)\cdot(2)\cdot(3)\cdot(1\cdot5)\cdot(1\cdot2\cdot4\cdot8)} = 189$

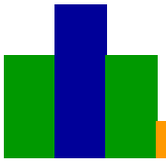


Acest rezultat este celebra teoremă Frame-Robinson-Thrall - formula din enunţ este cunoscută ca *formula cârligelor (hook formula)*.



..Analiza cantitativă a platizării

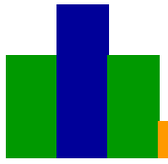
- *Corolarul 1:* $\varphi(m,n) = \frac{(m \cdot n)!}{[1 \cdot 2 \cdot \ldots \cdot n] \cdot [2 \cdot 3 \cdot \ldots \cdot (n+1)] \cdot \ldots \cdot [m \cdot (m+1) \cdot \ldots \cdot (m+n-1)]}$
- *Corolarul 2: Complexitatea lui* $\varphi(n,n)$ *este* $O(n^{n^2})$.



MSC-uri extinse, griduri, RV-scenarii

Cuprins:

- Generalităţi
- MSC-uri şi extensii
- *Griduri*
  - Dualitatea spatiu-timp (pe scurt)
  - Griduri şi expresii regulate 2-dimensionale
  - Operatorul de platizare
  - *Modelarea* inter-*actiunii prin griduri*
  - Griduri vs. MSC-uri
- RV-scenarii
- Concluzii, diverse, etc.



Modelarea inter-actiunii cu griduri

- Convenţia noastră este că într-un grid există o *cauzalitate stânga-dreapta* (procesul din dreapta poate fi executat după ce cel din stânga a fost executat), dar nu pe dos.

- Intrebare importantă:

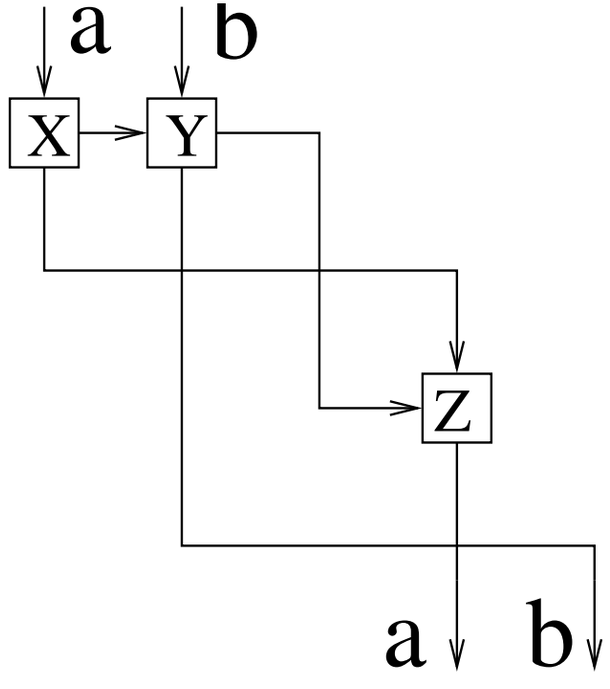
  Cum se poate modela *inter-acţiunea*? Un proces dintr-un grid poate trimite un mesaj unui proces din dreapta sa, dar nu poate primi un răspuns înapoi!

- Aparent, gridurile sunt bune spre a modela *acţiuni* (ca *trimirerea mesajelor în modelul stăpân-sclav*), dar *nu inter-acţiuni* reale (ca *apelarea unui proceduri şi aşteptarea rezultatului*)!

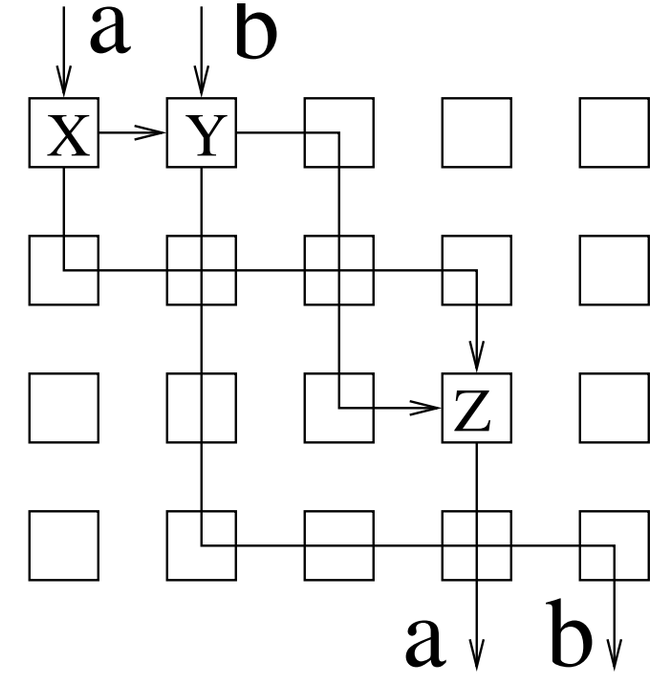
..Modelarea inter-actiunii cu griduri

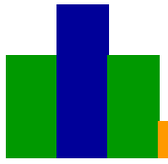
Actiune vs. inter-actiune: Avem mai jos două procese (fire de execuţie) (a) şi (b) care interacţionează şi reprezentarea interacţiei lor cu un grid.

(i) o interacţiune



(ii) gridul corespunzător



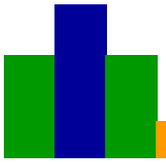


..Modelarea inter-actiunii cu griduri

- Să notăm că gridurile sunt obiecte *logice*, nu geometrice;
- O situaţie de comunicare în ambele direcţii între două procese (a) şi (b) este izomorfă cu cea prezentată în figura anterioară, în (i);
- Dacă alfabetul de griduri este suficient de bogat spre a conţine

  celule vide, identităţi, colţuri, şi intersecţii

  atunci figura din (i) poate fi reprezentată cu gridul din (ii);
- Deci, *gridurile nu sunt restrictive, putînd modela orice fel de interacţiune* (cu condiţia ca alfabetul să conţine constantele enumerate mai sus).



MSC-uri extinse, griduri, RV-scenarii

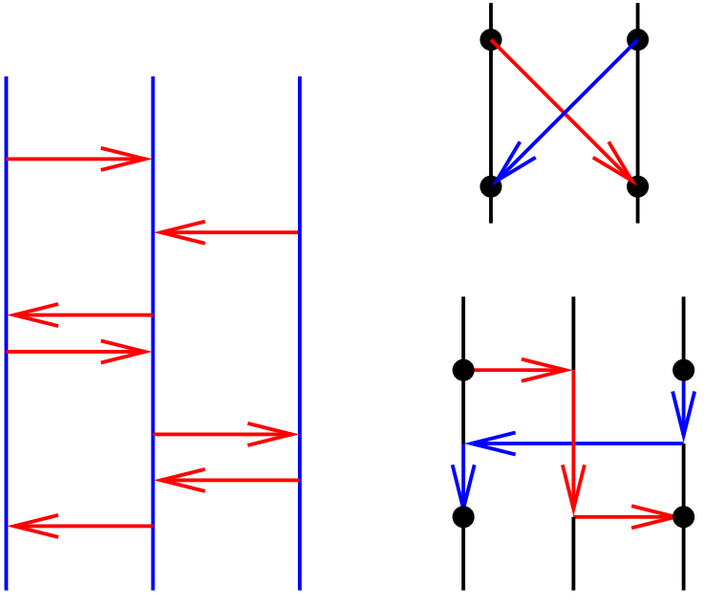
Cuprins:

- Generalităţi
- MSC-uri şi extensii
- *Griduri*
  - Dualitatea spatiu-timp (pe scurt)
  - Griduri şi expresii regulate 2-dimensionale
  - Operatorul de platizare
  - Modelarea *inter*-actiunii prin griduri
  - *Griduri vs. MSC-uri*
- RV-scenarii
- Concluzii, diverse, etc.

Griduri vs. MSC-uri

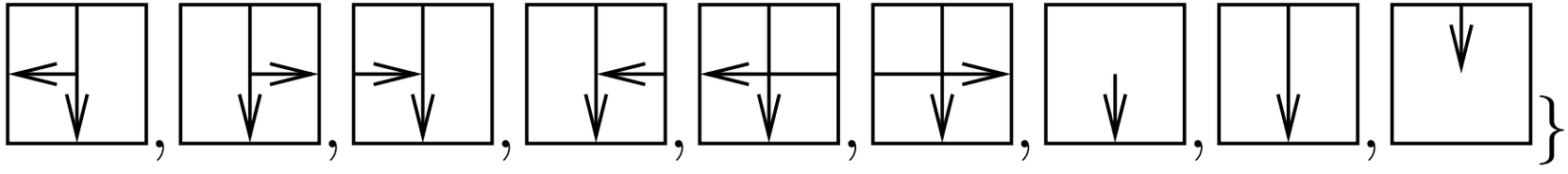
MSC-uri ca griduri

- MSC-urile pot fi reprezentate prin griduri
- distincţia spaţiu-timp se păstrează, dar se renumţă la “ciclicitatea orizontală” (gridurile sunt acum limitate orizontal)
- exemple de MSC:



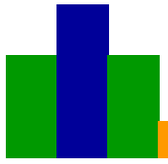
..Griduri vs. MSC-uri

Alfabet (pentru reprezentarea MSC-urilor ca griduri)

$V_{MSC} = \{$  $\}$

cu literele interpretate ca

- *sendL* - trimite un mesaj vecinului din stânga
- *sendR* - trimite un mesaj vecinului din dreapta
- *recL* - primeşte un mesaj de la vecinul din stânga
- *recR* - primeşte un mesaj de la vecinul din dreapta
- *passL* - pasează un mesaj de la dreapta la stânga
- *passR* - pasează un mesaj de la stânga la dreapta
- *init* - lansează un proces
- *void* - un proces care nu face nimic
- *end* - termină un proces



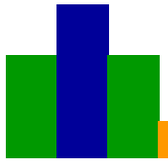
..Griduri vs. MSC-uri

Exemple

- MSC-ul anterior, cu mesaje încurcişate, poate fi reprezentat prin formula

$$(sendR \triangleright recL \triangleright void) \cdot (recR \triangleright passL \triangleright sendL) \cdot (void \triangleright sendR \triangleright recL)$$

- limbaj expresiv: se pot reprezenta *mesaje pierdute*, *procese incomplete*, chiar şi *coliziuni*, *primiri de mesaje netrimese*, *terminări de procese neîncepute*, etc.



..Griduri vs. MSC-uri

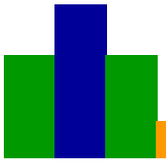
Restricţii

(α) fiecare linie este de forma $init^{\dagger}$ ori $end^{\dagger}$ ori $(sendR \triangleright passR^{\dagger} \triangleright recL + recR \triangleright passL^{\dagger} \triangleright sendL + void)^{\dagger}$;

(β) fiecare coloană este de forma $init \cdot (sendL + sendR + recL + recR + passL + passR)^{\star} \cdot end$

Teoremă: Gridurile peste V_{MSC} care satisfac condiţiile $(\alpha), (\beta)$ sunt echivalente cu MSC-urile.

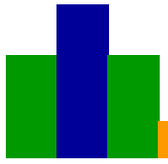
Notă: Prin reprezentarea de mai sus, se poate uşor largi clasa MSC-urilor la cele care se pot extinde nemărginit şi orizontal.



MSC-uri extinse, griduri, RV-scenarii

Cuprins:

- Generalităţi
- MSC-uri şi extensii
- Griduri
- *RV-scenarii*
  - *Tipuri de date temporale*
  - Specificatii spatio-temporale relationale
  - RV-scenarii
- Concluzii, diverse, etc.

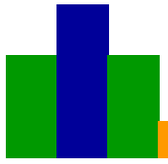


Date spatiale vs. date temporale

Banda Turing a fost primul model se memorie "*spaţială*" folosit în modelele de calculabiliate bazate pe maşini de calcul – structurile uzuale de date pot fi implementate în acest model simplu.

Dimensiunea spatiala: Banda Turing are şi o dimensiune temporală (ca şi celelalte tipuri de organizare a memoriei folosite în programarea imperativă). Specific este faptul că celulele de memorie pot fi accesate într-un timp constant. Pe scurt: *spaţiul potenţial nelimitat şi timpul limitat de acces* dă caracteristica de "*memorie spaţială*".

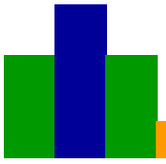
Date temporale: Vom folosi sugestia dată de dualitatea spaţiu-timp spre a crea structuri de date complexe în timp, implementabile pe structura simplă de şuvoi (stream). Aceste memorii *temporale* vor fi *potenţial nelimitate în timp, dar cu extensie spaţială limitată*.



Structurile de control ca date temporale

Structurile de control: Ele suplinesc rolul datelor *temporale* în programarea uzuală:

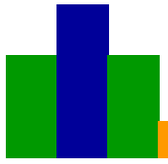
- Dacă se ia o instrucţiune dintr-un program şi de notează *timpul în care este activată*, atunci se obţine *o data temporală*!
- Dar, aceste date apar într-o *formă impură*, căci *instrucţiunile combină controlul cu datele memoriei (spaţiale)*.
- Propunerea este de a *rupe legătura structurilor de control cu datele spaţiale* şi de a *prezenta datele temporale într-o formă pură*.



Registri

Registrii:

- Un registru conţine un *număr natural* (în modelul teoretic, numerele sunt potenţial nelimitate);
- Regiştrii se pot *implementa pe o banda Turing*;
- La nivelul regiştrilor, problemele de nivel scăzut ca *identificarea poziţiei* regiştrilor pe bandă, *deplasarea* lor (dacă nu este suficient spaţiu), *operaţiile aritmetice ori logice* la nivel de bit, etc. se estomprează.



Suvoaie

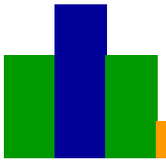
Suvoaie: Noţiune duală temporal a benzii Turing este şuvoiul.

- Un *şuvoi (engl. "stream")* este o secvenţă finită ori infinită de date ordonate în timp; se reprezintă sub forma

  $a_0 \frown a_1 \frown a_2 \frown \ldots,$
  unde $a_0, a_1, a_2, \ldots$ sunt datele sale ("tokens")
  la momentele de timp $0, 1, 2, \ldots$, respectiv;

- Suvoaiele vor fi mereu *finite*, dar *nelimitate* în timp;

- Un şuvoi poate fi gândit ca rezultatul *observării datelor transmise pe un canal* de comunicaţie – el afişază o dată (de tipul canalului) în fiecare moment de timp.

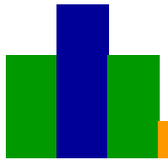


Voci

Voce: *Vocile* se definesc similar cu regiştrii. O voce este o *structură temporală de date* care conţine un *număr natural* (în modelul teoretic, numerele sunt potenţial nelimitate).

Implementare: Vocile se pot *implementa pe un şuvoi* ca regiştrii pe o bandă. De exemplu, se poate specifica o voce dînd timpul de start şi lungimea.

Exemplu: O voce v care conţine 2005 este definită de o adresă temporală t_0 (unde începe pe şuvoi) şi de o lungime (aici 4, dacă se foloseşte reprezentarea zecimală)
– celula ataşată şuvoiului va afişa cifrele 2,0,0,5 în momentele t_0, t_0+1, t_0+2, t_0+3.



..Voci

Detalii irelevante:

- La nivelul specificaţiei (or al programelor interactive de nivel înalt) multe detalii *nu* ne interesează. Exemple:

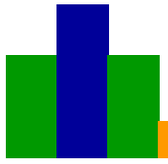
  – *unde anume începe* efectiv o voce pe şuvoi;

  – dacă vocile se *reprezintă continuu* (toate cifrele sale sunt vecine pe şuvoi), ori *alternant* (pe rând, câte o cifră din fiecare voce), ori altfel

  – *cum se fac operatiile aritmetice şi logice* la nivel de bit temporal,

  – etc.

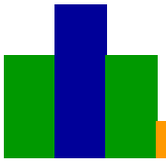
- Detalii, ca cele de mai sus, vor fi *necesare* la nivelul unei specificaţiei rafinate care să fie implementată pe o maşină concretă care are restricţii spaţio-temporale clare (cod maşină)



Alte date temporale

Alte date temporale:

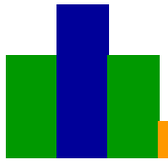
- Multe din tipurile de date uzuale au o versiune temporală – le vom nota adăugând un "`t`" în faţa datelor uzuale.
- Exemple: `tBool` (booleeni)
  `tInt` (întregi, de diverse lungimi)
  `tArray` (vectori)
  `tLinkedList` (liste înlănţuite)
  etc.



MSC-uri extinse, griduri, RV-scenarii

Cuprins:

- Generalităţi
- MSC-uri şi extensii
- Griduri
- *RV-scenarii*
  - Tipuri de date temporale
  - *Specificatii spatio-temporale relationale*
  - RV-scenarii
- Concluzii, diverse, etc.



Specificatii relationale

Specificatii spatio-temporale relationale:

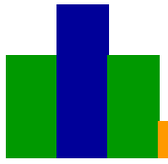
- O *specificaţie spaţio-temporală relaţională* este o relaţie

$$S \subseteq (\mathbb{N}^m \times \mathbb{N}^p) \times (\mathbb{N}^n \times \mathbb{N}^q)$$

  între regiştri şi voci de intrare şi de ieşire;

- *Tipul specificaţiei* este indicat prin notaţia $S : (m, p) \rightarrow (n, q)$ unde m (resp. p) este numărul de *voci de intrare* (resp. *regiştri de intrare*) şi n (resp. q) este numărul de *voci de ieşire* (resp. *regiştri de inşire*).

- *Pe elemente*, o specificaţie este descrisă ca o relaţie între tuple concrete $\langle v \mid r \rangle \mapsto \langle v' \mid r' \rangle$ unde v, v' (resp. r, r') sunt tuple de voci (resp. regiştri).



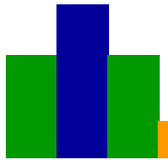
Exemple de specificatii relationale

Exemple: *Constatele de bază* sunt

$$c0 = \boxplus, c1 = \boxplus, c2 = \boxplus, c3 = \boxplus, c4 = \boxplus, c5 = \boxplus$$

Ele au o interpretare relaţională naturală, anume:

$c0 = \emptyset$;
$c1 = \{\langle \mid x\rangle \mapsto \langle \mid x\rangle : x \in \mathbb{N}\}$;
$c2 = \{\langle x \mid \rangle \mapsto \langle x \mid \rangle : x \in \mathbb{N}\}$;
$c3 = \{\langle \mid x\rangle \mapsto \langle x \mid \rangle : x \in \mathbb{N}\}$ (convertor spaţiu-timp);
$c4 = \{\langle x \mid \rangle \mapsto \langle \mid x\rangle : x \in \mathbb{N}\}$ (convertor timp-spaţiu);
$c5 = \{\langle x \mid y\rangle \mapsto \langle x \mid y\rangle : x, y \in \mathbb{N}\}$.



Specificatii compuse

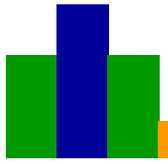
Specificatii compuse: Specificaţiile se pot *compune orizontal şi vertical*, dacă tipurile corespund: Date două specificaţii

$$S_1 : (m_1, p_1) \to (n_1, q_1) \text{ şi } S_2 : (m_2, p_2) \to (n_2, q_2)$$

- *compunerea orizontală* $S_1 \triangleright S_2$ este definită dnd $n_1 = m_2$; tipul lui $S_1 \triangleright S_2$ este $(m_1, p_1 + p_2) \to (n_2, q_1 + q_2)$; rezultatul este cel aşteptat:

$$\langle v \mid r_1, r_2 \rangle \mapsto \langle v'' \mid r_1', r_2' \rangle \text{ in } S_1 \triangleright S_2 \quad \text{dnd}$$
$$\exists v'. \langle v \mid r_1 \rangle \mapsto \langle v' \mid r_1' \rangle \text{ in } S_1 \wedge \langle v' \mid r_2 \rangle \mapsto \langle v'' \mid r_2' \rangle \text{ in } S_2$$

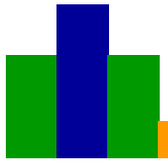
- *compunerea verticală* $S_1 \cdot S_2$ este definită similar: este definită dnd $q_1 = p_2$, iar tipul rezultatului este $(m_1 + m_2, p_1) \to (n_1 + n_2, q_2)$.



..Specificatii compuse

Pare un pic straniu că tipul specificaţiilor compuse *creşte*, dar la o analiză mai atentă e justificabil:

- Creşterea tipului pe orizontală revine la faptul că dacă avem mai multe procese/obiecte, fiecare necesită date de intrare pentru iniţializare; acest lucru se evită folosind *constructori* – în loc să dau aceste date din afară, ele sunt create în program la iniţializarea proceselor/obiectelor; pe scurt, extern tipul unui proces/obiect, exceptând pe cel principal, este $0 \rightarrow 0$;
- O analiză similară ar putea fi făcută pentru dimensiunea verticală, dar necesită concepte privind rv-programe (programe cu regiştri şi voci) care nu au fost încă introduse în curs.



MSC-uri extinse, griduri, RV-scenarii

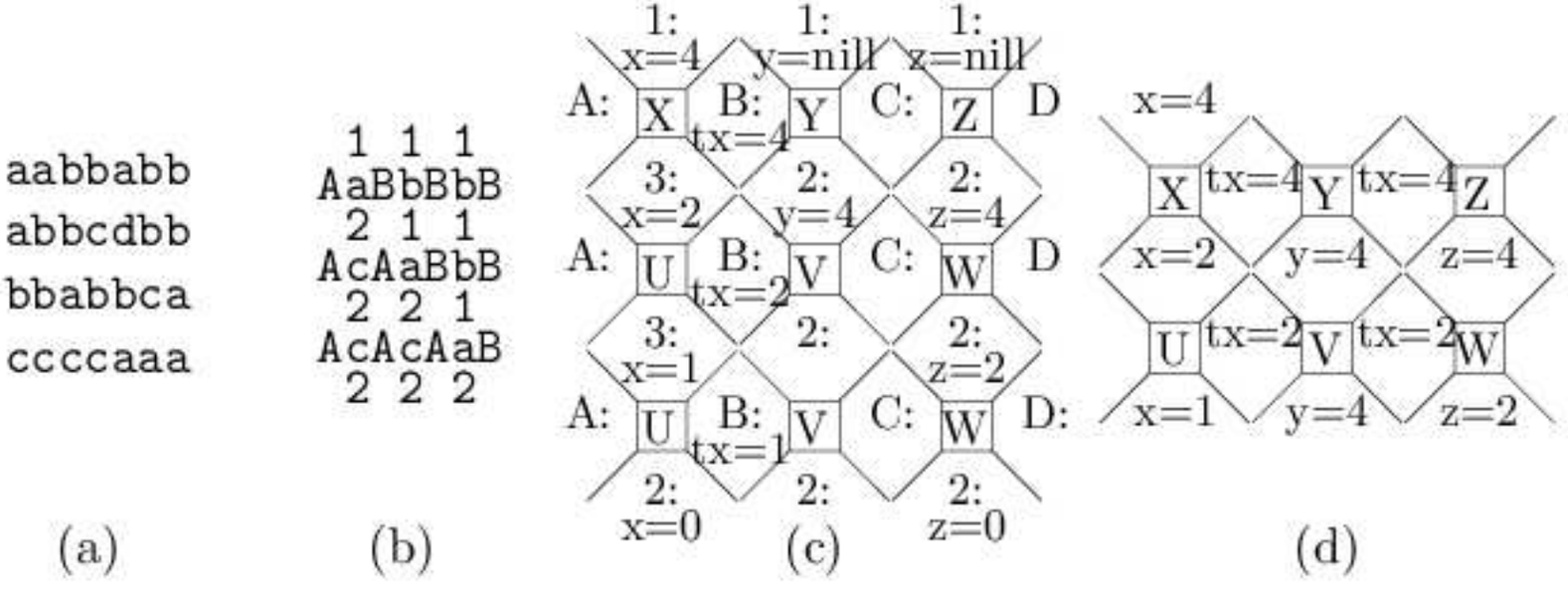
Cuprins:

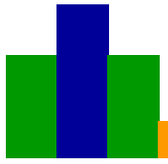
- Generalităţi
- MSC-uri şi extensii
- Griduri
- *RV-scenarii*
  - Tipuri de date temporale
  - Specificatii spatio-temporale relationale
  - *RV-scenarii*
- Concluzii, diverse, etc.

RV-scenarii

Definiţii

- un *RV-scenariu* este un grid îmbogăţit cu date în jurul fiecărei litere (*RV* - Programs with *R*egisters and *V*oices)
- datele se pot da pe diferite nivele de abstracţie, de la un simplu nume, la variabile şi valori complet definite
- exemple

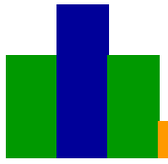




..RV-scenarii

Tipuri

- tipul unei interfeţe este $t_1; t_2; \ldots; t_k$, unde fiecare t_k este tipul asociat unui celule din scenariu care participă la interfaţă (ca la reprezentări spaţio-temporale)
- tipul vid se notează cu 0 ori *nill* şi poate fi liber adăugat ori scos din descrierea interfeţei
- tipul unui scenariu este notat $f : \langle w|n \rangle \rightarrow \langle e|s \rangle$ (specifică tipurile interfeţelor de vest, nord, est, şi sud)
- exemplu anterior (dreapta) are tipul $\langle nill; nill|sn; nill; nill \rangle \rightarrow \langle nill; nill|sn; sn; sn \rangle$, unde *sn* este tipul "intreg spaţial"



..RV-scenarii

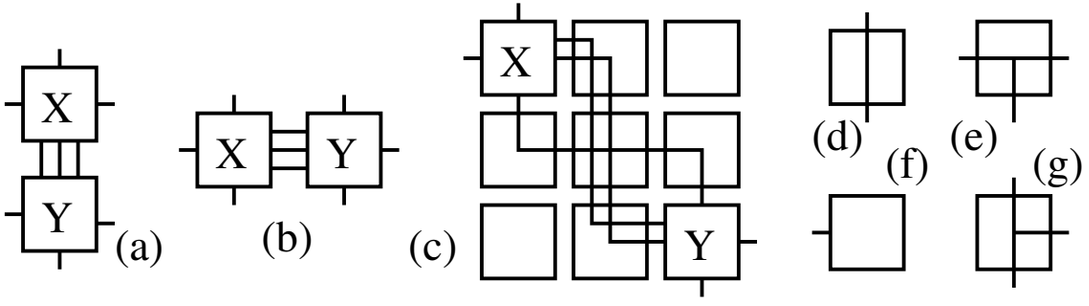
Egalitatea interfeţelor

- două interfeţe $t = t_1;t_2;\ldots;t_k$ şi $t' = t'_1;t'_2;\ldots;t'_{k'}$ sunt *egale*, scris $t = t'$, dacă $k = k' \wedge \forall i : t_i = t'_i$
- două interfeţe $t = t_1;t_2;\ldots;t_k$ şi $t' = t'_1;t'_2;\ldots;t'_{k'}$ sunt *egale până la inserarea de nill*, scris $t =_n t'$, dacă devin egale după inserarea de termeni *nill*

Constante (vezi figura de mai jos, desenul (c))

- *identităţi* (celula 2, rândul 2), *recorder* R (celula 2, rândul 1), *speaker* S (celula 1, rândul 2), *celulă vidă* Λ (celula 3, rândul 1), *transformed recorder* (desen (e)), *transformed speaker* (desen (g)).

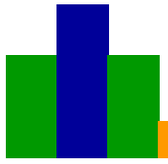
Operaţii cu scenarii



Compunerea orizontală

- date două scenarii $f_i : \langle w_i | n_i \rangle \rightarrow \langle e_i | s_i \rangle, i = 1,2$, *compunerea orizontală* $f_1 \triangleright f_2$ este definită numai dacă $e_1 =_n w_2$
- rezultatul se obţine inserând o linie goală pentru fiecare element *nill* introdus în interfată estică a lui f_1, respectiv vestică a lui f_2, apoi, ca la griduri, rezultatul $\overline{f_1}$ se pune la stănga lui $\overline{f_2}$
- tipul rezultatului este $f_1 \triangleright f_2 : \langle w_1 | n_1; n_2 \rangle \rightarrow \langle e_2 | s_1; s_2 \rangle$ şi este unic până la inserţia de elemente *nill*

Compunerea verticală - similar



..Operaţii cu scenarii

Compunerea diagonală (este operaţie derivată)

- *compunerea diagonală* $f_1 \bullet f_2$ este definită numai dacă avem $e_1 =_n w_2$ şi $s_1 =_n n_2$
- rezultatul este definit de formula

$$f_1 \bullet f_2 = (f_1 \triangleright R_1 \triangleright \Lambda_1) \cdot (S_2 \triangleright Id \triangleright R_2) \cdot (\Lambda_2 \triangleright S_1 \triangleright f_2)$$

cu constante R, S, Id, Λ adecvate (vezi (c) din figura de mai sus)

Extensia la mulţimi de scenarii (A, B sunt mulţimi de scenarii)

- *compunerea orizontală*

$$A \triangleright B = \{f_a \triangleright f_b | \text{ cu } f_a : \langle w_a | n_a \rangle \rightarrow \langle e_a | s_a \rangle, f_a \in A \\ f_b : \langle w_b | n_b \rangle \rightarrow \langle e_b | n_b \rangle, f_b \in B \\ \text{şi astfel încât } e_a =_n w_b\}$$

- *compunerea verticală* şi *compunere diagonală*- similar