ASUPRA UTILIZĂRII UNOR MECANISME FORMALE ÎN DESCRIEREA PROCESELOR DE AŞTEPTARE SPECIFICE SISTEMELOR CLIENT-SERVER

GABRIEL DIMITRIU
Student Universitatea Spiru Haret, Facultatea Matematică-Informatică
gabriel@tech.pub.ro

Rezumat: Această comunicare abordează modelarea proceselor de aşteptare care intervin în cadrul aplicațiilor client-server cu ajutorul teoriei limbajelor formale şi a structurilor AFN/AFD. Exemplificarea este realizată pentru o aplicație particulară.

Cuvinte cheie: gramatici formale, automate, client-server, modelare

1. INTRODUCERE

În contextul acestei lucrări, prin proces de aşteptare se înțelege un proces industrial, informatic sau economic în care un server oferă servicii unor clienți. Aplicațiile care implementează procese de aşteptare sunt de tip client-server.

Tratarea apelurilor într-o centrală telefonică, tratarea cererilor de către un server WEB, sisteme de înregistrare în timp real, sisteme de comandă şi control ale proceselor industriale, deservirea clienților unei baze de date, bancomate etc. sunt doar cîteva exemple de aplicații în care apare fenomenul de asteptare.

Un model client-server presupune existența următoarelor elemente: aşteptare, un server (punct de deservire) care are o lege de servire, o lege de intrare a unor entități care trebuie servite, o lege de ieşire din sistem şi o anumită configurație a şirului de aşteptare.

Legea pentru generarea şirului de aşteptare este determinată de politica serverului (coadă: *F*irst *I*n *F*irst *O*ut; stivă: *L*ast *I*n *F*ist *O*ut; listă cu priorități etc.). Alegerea politicii (disciplinei) de servire este determinată de caracteristicile procesului modelat. Şirul de aşteptare poate fi vid, finit sau infinit; de asemenea el poate să aibă o rată constantă sau o rată variabilă care să determine momentele de staționare (şomaj) ale serverului (când serverul nu are nici ce prelucra şi nici ce furniza la ieşire).

Mărimea şirului precum şi indicele de staționare specifice serverului sunt parametri importanți privind dimensionarea serverului şi alegerea legii de funcționare sau pentru construcția serverului. De exemplu, dacă timpii de aşteptare pentru o cerere devin prea mari şi un client nu poate fi servit în timp util, în cazul proceselor de timp real, atunci se poate decide crearea în paralel a unor

Analele Universității SPIRU HARET, Seria Matematică-Informatică, nr. 1 , 2005, ISSN 1841 - 7833

componente ale serverului în locul unde are loc gâtuirea sau chiar dublarea completă a serverului în cazul unor procese de importanță strategică sau când se doreşte tratarea simultană a mai multor cereri.

2. MODELARE CU AJUTORUL LIMBAJELOR FORMALE

În cele ce urmează se utilizează modelarea unui proces de aşteptare folosind teoria limbajelor formale aşa cum este propusă în [1]. Se consideră vocabularele:

$$\sum_1=\{a\},\ \sum_2=\{a,\ b\},\sum_3=\{a,b,c\} \qquad (1)$$

Fie τ o unitate de timp, suficient de mică pentru ca oricare două cereri să nu sosească simultan în coada de aşteptare şi suficient de mare ca să se poată spune că deservirea se face în cel puțin două unitați de timp τ (o unitate de timp pentru prelucrare şi o unitate de timp pentru ieşire). Cu alte cuvinte, dacă rata de sosire a cererilor în server este de o cerere la două unități de timp τ, atunci fiecare cerere este prelucrată fără să aştepte, iar serverul nu are "timpi morți". În aplicațiile practice, pentru a nu mări numărul de reguli prea mult, dar şi pentru a avea o valoare acoperitoare se va alege cea mai mare unitate de timp cu proprietățile de mai sus, cu excepția sistemelor de timp real, cu importanță strategică, când se alege cea mai mică valoare şi chiar se poate întrerupe servirea unui client, iar apoi se poate servi alt client care are prioritate mai mare. În acest caz se salvează informații şi despre starea clientului în momentul întreruperii, informații necesare pentru reluarea prelucrării clientului din momentul întreruperii, clientul întrerupt este pus în coada de aşteptare cu prioritate maximă ca, după terminarea prelucrării clientului prioritară el să fie luat primul pentru a i se putea continua prelucrarea.

Legea de intrare poate fi văzută ca o secvență infinită cu simboluri din vocabularul $\sum_2$, simbolurile *a* şi *b* generându-se la un interval de timp τ şi avind semnificația: *a* reprezintă venirea unei cereri la server în acest interval, iar *b* ca lipsa venirii unei cereri la server în acel interval. Aşadar, conform celor menționate mai sus, dacă avem şirul abbabba... clienții sunt preluați de către server fără aşteptare, iar serverul nu are "timpi morți".

Legea de servire poate fi văzută ca un şir peste vocabularul $\sum_3$ având următoarele semnificații ale simbolurilor pe timpul unei perioade de timp τ: *a* – dacă o cerere este transmisă serverului, *b* – dacă serverul este ocupat cu deservirea altei cereri, *c* – dacă o cerere este satisfăcută şi i se trimite răspuns. Astfel, durata servirii unei cereri este egală cu lungimea şirului delimitat de două simboluri *a* şi *c* consecutive. Cererile clienților sunt identice însă timpul lor de servire va fi diferit în funcție de momentul în care sosesc şi nu de natura lor (cazul este cel a unei reguli FIFO fără priorități, dar cu cereri identice).

Legea de ieşire va fi văzută ca un şir peste vocabularul $\sum_2$ (*a* este ieşirea din server ce corespunde simbolului *c* din legea de servire, iar *b* afirmă că nici un client nu primeşte răspuns şi corespunde simbolurilor *a* şi *c* din legea de servire).

Şirul de aşteptare este un cuvînt peste vocabularul $\sum_2$: *a* – reprezintă venirea unei cereri, *b*- orice încercare de tratare a unei cereri care eşueaza din cauza lipsei de cereri.

Considerând subcuvintele şirurilor prezentate obținem patru limbaje: LI (limbajul de intrare), LS (limbajul de servire ale cărui cuvinte încep întotdeauna cu simbolul *a*, deoarece reprezintă prima cerere care este primită de catre server,

echivalent cu timpul 0), LO (limbajul de ieşire) care sunt infinite întotdeauna şi LC (limbajul care reprezintă şirul de aşteptare) care poate fi infinit, nul (în cazul în care nu vine nici o cerere) sau de lungimea 1 (cînd cererile vin exact cu rata de prelucrare a serverului, aşa cum ar fi cazul ideal).

Folosind notaţiile de mai sus legile de funcţionare ale serverului sunt:

1. dacă în LS apare un simbol a, atunci cererea curentă din LI este tratată de server, dacă cererea care trebuia să fie tratată este b atunci se tratează o cerere din LC;

2. dacă în LS apare un simbol a, în LI apare un simbol b iar LC este vid, atunci serverul rămâne în aşteptare până la apariţia unui simbol a în LI iar LS ramîne în a;

3. dacă LS este b atunci orice cerere care vine reprezentată printr-un simbol a în LI este introdusă în LC.

Cu aceste legi de funcţionare se poate formula următorul algoritm dupa care sistemul funcţionează:

1. dacă nu avem a la intrare şi coada este goală treci la pasul 1;

2. dacă avem a la intrare atunci prelucreaza a şi treci în starea b, dacă nu treci la pasul 6;

3. dacă în timp ce serverul prelucrează apare un simbol a la intrare atunci salveaza a în coadă;

4. treci în starea c. Scoţi la ieşire a şi dacă în acelaşi timp apare a la intrare salvează a în coadă;

5. treci în starea a şi mergi la pasul 2;

6. dacă coada nu este goală atunci extragi un element din coadă, prelucrezi a, treci în starea b şi mergi la pasul 3.

3. **MODELARE CU AJUTORUL AUTOMATELOR**

Pe parcursul acestei secţiuni se presupun cunoscute notaţiile şi teminologia teoriei automatelor [2].

Pentru modelarea procesului de aşteptare cu ajutorul automatelor vom presupune că deservirea area loc exact în doua unitati de timp (1 unitate de timp pentru prelucrare şi o unitate de timp pentru ieşire) aşadar şirul starilor automatului peste vocabularul $\sum_2$ va fi abcabc...

3.1 **Modelare folosind automat finit nedeterminist**

Algoritmul descris mai sus se poate modela cu ajutorul unui automat nedeterminist cu 6 stări, aceste stări sunt date de tripletul (LS,LC,LO) cu semnificaţiile din secţiunea 2. Aşadar mulţimea stărilor este: K={(a,b,b), (a,a,b), (b,b,b), (b,a,b,), (c,b,a), (c,a,a)} cu semnificaţiile:

- (a,b,b)=se aşteaptă cerere de la clienţi, coada este goală şi la ieşire nu se furnizează nimic
- (a,a,b)=se aşteaptă cerere de la clienţi,în coadă se află cel puţin un client şi la ieşire nu se furnizează nimic
- (b,b,b)=se prelucrează informaţia pentru client, coada este goală şi la ieşire nu se furnizează nimic
- (b,a,b)=se prelucrează informaţia pentru client,în coadă se afla cel puţin un client şi la ieşire nu se furnizează nimic

- (c,b,a)=se furnizează la ieşire un element a şi în coada nu se află nici un client
- (c,a,a)=se furnizează la ieşire un element a şi în coadă se află cel putin un client.

Nedeterminarea provine din definirea stărilor (b,a,b), (c,a,a) si (a,a,b) în care nu se specifică numărul de elemente care se află în coadă, ci doar că există elemente în coadă.

Starea inițială este q_0={(a,b,b)} iar mulțimea stărilor finale este F={(c,b,a),(c,a,a)}. Deşi automatul considerat nu are stări finale în adevăratul sens al cuvântului, el mergând la infinit iar ieşirea fiind şirul dat de limbajul LO, şi starea de aşteptare în timp mort este q= q_0.

Alfabetul de intrare este Σ={a,b} cu a = a venit o cerere de la un client şi b = nu a venit o cerere de la un client.

Schema automatului este prezentată în figura 1:

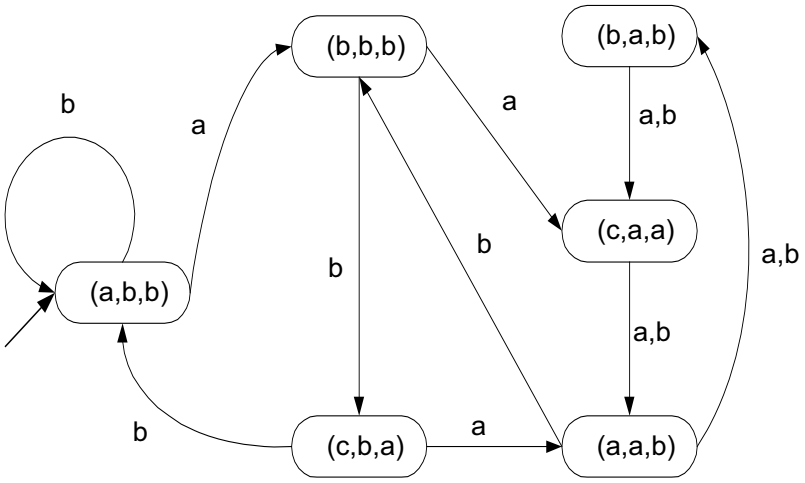

Figura 1.

Tabelul 1 prezintă funcția de stare δ :

Tabelul 1

δ	A	b
(a,b,b)	{(b,b,b)}	{(a,b,b)}
(a,a,b)	{(b,a,b)}	{(b,b,b),(b,a,b)}
(b,b,b)	{(c,a,a)}	{(c,b,a)}
(b,a,b)	{(c,a,a)}	{(c,a,a)}
(c,b,a)	{(a,a,b)}	{(a,b,b)}
(c,a,a)	{(a,a,b)}	{(a,a,b)}

Se observă ca nedeterminarea apare în momentul când vine simbolul b şi automatul este în starea (a,a,b) deoarece dacă este un singur element în coada de aşteptare, atunci trece în starea (b,b,b), altfel trece în starea (b,a,b).

Aşadar cvintetul $(\sum,K,q_0,F,\delta)$ formează un atomat finit nedeterminist.

3.2 **Modelare folosind automatului finit determinist**

Prima modificare pe care cititorul ar fi tentat să o faca este ca să introducă un contor care specifică cîte elemente sunt în coadă, şi în funcție de acel contor, să se ia o decizie de trecere în stările (b,b,b) sau (b,a,b). Dar făcînd această modificare ceea ce rezultă nu mai este un automat în adevaratul sens al cuvîntului deoarece atunci vom avea un alfabet foarte mare pentru contor.

Propunem următoarea modificare a automatului finit nedeterminist prezentat în secțiunea 3.1 pentru a rezulta un automat finit determinist.

În locul alfabetului de intrare $\sum$ vom avea alfabetul $\sum^1$={(a,b,b), (a,a,b), (a,a,a), (a,b,a), (b,b,b), (b,a,b), (b,a,a), (b,b,a)}. Se observă că $\sum^1$:{a,b} x {a,b} x {a,b}, aşadar noul alfabet este dintr-un spațiu tridimensional față de fostul alfabet de intrare care era dintr-un spațiu unidimensional. În acest triplet avem asignate valorile (LI(i),LC(i),LC(i+1)) sau (valoarea de intrare, prima cerere din coadă, a doua cerere din coadă) toate la momentul respectiv deci pentru $\forall$i, cu aceste modificări implementarea cozii de aşteptare se modică astfel încăt să permită citirea primului şi celui de-al doilea element fără a face extragerea lor efectivă. Se face convenția că dacă nu este nici o cerere în coadă acesta se asignează cu b iar cererea următoare din coadă este tot asignată cu b deoarece în coadă nu sunt decât a, iar situația următoare nu este posibilă LC(10)=abaaaab deoarece în coadă trebuie să fie numai a iar b defineşte sfârşitul cozii dar în schimb este posibil LC(10)=aaaaabb. în cazul în care se adaugă în coadă LC(10) încă un element el devine LC(11)=aaaaaabb. Aşadar din $\sum^1$ trebuie să scoatem tripletele (a,b,a) şi (b,b,a) deoarece mărim inutil alfabetul de intrare, aceste elemente neputând veni niciodată.

Mulțimile stărilor şi a stărilor finale rămîn identice cu cele prezentate în subsecțiunea 3.1 pentru automatul finit nedeterminist, dar se modifică funcția automatului care este acum $\delta:\sum^1 xK \rightarrow K$.

Cu notațiile şi explicațiile de mai sus avem schema automatului finit determinist prezentată în figura 2: în care prin x a fost codificată orice valoare din alfabetul $\sum$={a,b}.

Astfel, avem tabelul 2. Pentru funcția δ în care prin ϕ s-a reprezentat situația imposibilă, deoarece în starea respectivă coada este ori plină şi (x,b,x) nu poate apărea sau coada este goală şi (x,a,x) nu poate apărea.

Aşadar cvintetul $(\sum^1,K,q_0,F,\delta)$ formează un atomat finit determinist, care se observă că are aceleaşi stări finale şi funcționare ca şi automatul finit nedeterminist.

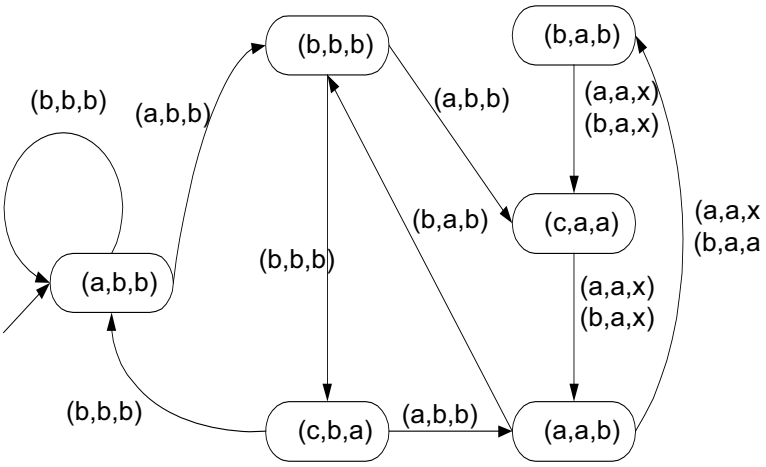

Figura 2

Tabelul 2

δ	(a,b,b)	(b,b,b)	(a,a,b)	(a,a,a)	(b,a,b)	(b,a,a)
(a,b,b)	(b,b,b)	(a,b,b)	ϕ	ϕ	ϕ	ϕ
(a,a,b)	ϕ	ϕ	(b,a,b)	(b,a,b)	(b,b,b)	(b,a,b)
(b,b,b)	(c,a,a)	(c,b,a)	ϕ	ϕ	ϕ	ϕ
(b,a,b)	ϕ	ϕ	(c,a,a)	(c,a,a)	(c,a,a)	(c,a,a)
(c,b,a)	(a,a,b)	(a,b,b)	ϕ	ϕ	ϕ	ϕ
(c,a,a)	ϕ	ϕ	(a,a,b)	(a,a,b)	(a,a,b)	(a,a,b)

4. **CONCLUZII**

Se observă că fără o informație asupra modului în care operează sistemul nu se poate face o simulare iar automatele pentru cazul în care prelucrarea durează mai multe unități de timp se complică foarte mult deoarece pentru fiecare unitate de timp în plus la prelucrare apare cîte o stare deci dacă perioada de timp cît se face prelucrarea este n atunci numărul stărilor atutomatului va fi n+5, ceea ce se vede că nu mai este realizabil. În acel caz se poate face doar o simulare cu ajutorul unui program sau pe hîrtie a sistemului cum a fost prezentată în [1] şi nu o modelare, caz în care se cunosc LI, LO, LS ale sistemului şi se urmăreste doar funcționarea sistemului .

REFERENCES

1. Gh. Păun, *Mecanisme generative de procese economice*. Ed. Tehnică,Bucureşti , 1980, pag 184-188
2. I. Văduva, *Modele de simulare*. Editura Universității din Bucureşti, 2004.

Abstract: In this paper we present the waiting processes modeling that appear in client-server applications with the help of formal language theory and of NFA/DFA structures for a particular application.