Modeli mobilnih procesa

Svetlana Jakšić

6.10.2008.

Sadržaj

Glava 1

Uvod

Proces je instanca računarskog programa koji se izvršava u računarskom sistemu koji ima mogućnost da istovremeno izvršava više programa. Računarski program, sam za sebe, je samo pasivan skup instrukcija, dok proces predsavlja stvarno izvršavanje tih instrukcija. Sa jednim programom može biti povezano više procesa. Na primer, otvaranje nekoliko prozora istog programa običe no povlači izvršavanje više od jednog procesa. *Konkurentnost* je svojstvo sistema u kojima se više procesa izvršava u isto vreme, pri čemu je moguća i njihova interakcija. Zbog toga prelazak u naredno stanje sistema nije jedinstveno odredjeno. Moguće su različite tranzicije, zbog čega analiza ponašanja takvog sistema može biti veoma složena. Teorija konkurentnosti razmatra širok spektar sistema i u zavisnosti od svojstva koja se apstrakuju razvija različite modele.
Mobilnost procesa i mobilnost uređaja je jedna od osobina konkurentnih sistema koja je sa pojavljivanjem World Wide Web mreže postala značajna. Cilj nam je da damo formalne modele za obe vrste mobilnosti: računarskih uređaja (npr. notebook računari, mobilni telefoni,...) i procesa (npr. applet). Geografska raspoređenost WWW ,,prirodno" zahteva mobilno računarstvo, te u ovom trenutku predstavlja najveću inspiraciju za rad u oblasti o kojoj će ovde biti reči.

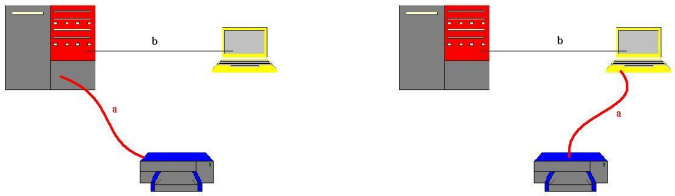
Glava 2

π-račun

U oblasti teorijskoj računarstva π-račun je procesni račun, koji su krajem osamdesetih godina XX veka razvili Robin Milner, Joachim Parrow and David Walker u [14] kao proširenje procesnog računa CCS (Calculus of Communicating Systems). π-račun je matematički model sistema u kome se povezanost među procesima može menjati kroz njihovu interakciju. Osnovni korak prelaska u naredno stanje sistema je prenos veze između dva procesa; proces koji primi vezu može da je iskoristi za dalju interakciju sa ostalim procesima. Ovo svojstvo čini π-račun pogodnim za modeliranje sistema u kojima se dostupnost elemenata sistema vremenom menja i izražavanje pojmova iz teorije kokurentnosti, kao što su prisup i resurs. Kao ilustraciju, navodimo sledeće primer: sistem koji se sastoji od servera, štampača i klijenta. Server konroliše pristup štampaču, a klijent želi da odštampa neki dokument. U početnom stanju (Slika 2.1(a)) samo server ima pristup štampaču. To je predstavljeno komunikacijskim kanalom a. Posle interakcije sa klijentom (Slika 2.1(b)) preko drugog kanala b, prisup štampaču je prebačen. To bismo u π-računu izrazili na sledeći način: server koji šalje a kroz b je $\bar{b}\langle a\rangle.S$; klijent koji prima neki kanal preko kanala b, pa potom koristi taj kanal za slanje podataka je $b(c).\bar{c}\langle d\rangle.P$. Goreopisana interakcija bi bila formulisana sa

$$\bar{b}\langle a\rangle.S|b(c).\bar{c}\langle d\rangle.P \rightarrow S|\bar{a}\langle d\rangle.P$$

Možemo videti da a ima dvostruku ulogu. U interakciji između servera i klijenta, a je objekat koji se prebačuje sa jednog na drugog. U daljoj interakciji klijenta i štampača a je ime komunikacijskog kanala. Ideja da su imena kanala u istoj kategoriji sa objektima koji se prebacuju tim kanalima leži u osnovi π-računa i odvaja ga od ostalih procesnih algebri. U ovom primeru a, b, c, d su samo *imena* koja intuitivno predstavljaju prava pristupa: a prisupa štampaču, b pristupa serveru, d pristupa nekim podacima, a c je vezano ime koje čeka da bude zamenjeno pravom pristupa koje stigne kroz a.



(a) Pre interakcije (b) Posle interakcije

Slika 2.1: Server-štampač-klijent

Ako je a jedini način da se pristupi štampaču onda možemo reći se štampač ,,pomerio" sa servera na klijenta, jer posle interakcije ništa drugo ne može da mu prisupi. Ovo je razlog zbog kojeg je π-račun nazvan računom *mobilnih procesa*. Ali π-račun je mnogo opštiji od toga. Štampač može da ima više kanala preko kojih može da dobija različite zadatke, a server te kanale može da pošalje različitim klijentima, uspostavljajući različita prava prisupa zajedničkom resursu.
Još jedan razlog zbog kojeg možemo reći da π-račun ima veću izražajnu moć od drugih procesnih algebri je mogućnost migracije domena lokalnih promenljivih. Kao i u većini procesnih algebri možemo zadati da je komunikacijski kanal lokalan (privatan) za grupu procesa. U π-računu ovu *restrikciju* zapisujemo sa $(\nu a)(P|Q)$. Nijedan drugi proces ne može neposredno da iskoristi a kao vezu sa P ili Q. Međutim ime a je objekat koji se može prenositi komunikacijskim kanalima i kao takav može biti poslat od strane procesa P ili Q nekom drugom procesu, koji ce potom moći da koristi restrikovani (vezani, privatni) kanal. Vratimo se na gorenavedeni primer i pretpostavimo da je a privatan komunikacijski kanal između servera i štampača. U π-računu to predstavljamo sa $(\nu a)(\bar{b}\langle a\rangle.S|a(e).R)$, gde je sa $a(e).R$ označen stampač. Server, i dalje, može da pošalje a kroz b. Rezultat je da, sada, tri procesa dele privatni kanal, koji se još uvek razlikuje od svih ostalih imena u svim ostalim procesima. To zapisujemo ovako

$$(\nu a)(\bar{b}\langle a\rangle.S|a(e).R)|b(c).\bar{c}\langle d\rangle.P) \rightarrow (\nu a)(S|a(e).R)|\bar{a}\langle d\rangle.P)$$

2.1 Sintaksa

U tabeli 2.1 su dati procesi koje možemo konstruisati u π-računu.

- Proces *Nil*, 0, je proces koji ne može da sprovede nijednu akciju.

$$
\begin{aligned}
P, Q ::= \; & 0 \\
& \mid \bar{a}\langle x\rangle.P \\
& \mid a(x).P \\
& \mid !P \\
& \mid (\nu x)P \\
& \mid P|Q
\end{aligned}
$$

Tabela 2.1: Procesi π-računa

- Proces *Output prefiks*, $\bar{a}\langle x\rangle.P$, šalje ime x kroz ime a i nastavlja kao P. Intuitivno, $\bar{a}$ je izlazni kanal kroz koji se šalje informacija x.
- Proces *Input prefiks*, $a(x).P$, označava da će kroz ime a biti primljeno neko ime, koje će biti zamenjeno umesto vezanog imena x.
- *Replikacija*, $!P$, je proces koji označava neograničenu replikaciju procesa P. Tj. $!P$ može da proizvede koliko god je potrebno paralelnih kopija procesa P.
- *Restrikcija* $(\nu x)P$, je proces koji se ponaša kao P, osim što je ime x lokalno. To znači da x ne može neposredno biti iskorišćeno za komunikaciju između P i okoline. Međutim, može biti iskorišćeno za komunikaciju mađu komponentama procesa P.
- *Paralelna kompozicija*, $P|Q$, predstavlja zajdničko (paralelno, istovremeno) izvršavanje procesa P i Q. Komponente P i Q se mogu izvršavati nezavisno jedna od druge, ali mogu i komunicirati ako jedna izvrši akciju input-a, a druga output-a preko istog imena.

2.2 Strukturalna kongruencija

Sintaksa procesa π-rašuna, zadata u tabeli 2.1, je na neki način suviše konkretna. Na primer, procesi $a(x).\bar{b}\langle x\rangle$ i $a(y).\bar{b}\langle y\rangle$ se razlikuju po sintaksi, iako je jedina razlika među njima izbor vezanog imena, te stoga oni predstavljaju procese sa jednakim ponašanjem: proces koji nešto primi preko kanala a, a zatim to pošalje kanalom b. Drugi primer su procesi $P \mid Q$ i $Q \mid P$ koji, oba, predstavljaju istovremeno izvršavanje procesa P i Q. Jedan od načina da izjednačimo ovakve procese je uvođenje relacije *strukturalne kongruencije* (tabela 2.2).

$P \equiv P$	(Str Refl)
$P \equiv Q \Rightarrow Q \equiv P$	(Str Sim)
$P \equiv Q, Q \equiv R \Rightarrow P \equiv R$	(Str Tran)
$P \equiv Q \Rightarrow (\nu n)P \equiv (\nu n)Q$	(Str Res)
$P \equiv Q \Rightarrow P\|R \equiv Q\|R$	(Str Par)
$P \equiv Q \Rightarrow !P \equiv !Q$	(Str Repl)
$P\|Q \equiv Q\|P$	(Str Par Komut)
$(P\|Q)\|R \equiv P\|(Q\|R)$	(Str Par Asoc)
$!P \equiv P\|!P$	(Str Repl Par)
$(\nu n)(\nu m)P \equiv (\nu m)(\nu n)P$	(Str Res Res)
$(\nu n)(P\|Q) \equiv P\|(\nu n)Q$ if $n \notin fn(P)$	(Str Res Par)
$P\|0 \equiv P$	(Str Nil Par)
$!0 \equiv 0$	(Str Nil Repl)
$(\nu n)0 \equiv 0$	(Str Nil Res)
$P \equiv Q$ ako jednaki do na izbor vezanih imana	(Str α)

Tabela 2.2: Strukturalna konguencija

$$\bar{a}\langle y\rangle.P \mid a(x).Q \to P \mid Q\{x \leftarrow y\} \quad \text{(Kom)}$$
$$P \to Q \Rightarrow (\nu n)P \to (\nu n)Q \quad \text{(Res)}$$
$$P \to Q \Rightarrow P|R \to Q|R \quad \text{(Par)}$$
$$P' \equiv P, P \to Q, Q \equiv Q' \Rightarrow P' \to Q' \quad \text{(Str)}$$

Tabela 2.3: Pravila redukcije

2.3 Semantika operacija

Relacija redukcije $\to$, zadata u tebeli 2.3, opisuje prelazak procesa iz jednog u naredno stanje. Pišemo $P \to Q$, ako proces P posle neke akcije nastavlja da se ponaša kao proces Q.

- Pravilo **(Kom)** je osnovno pravilo π-računa i njime je obuhvaćen smisao komunikacije kanalima. Procesi $\bar{a}\langle y\rangle.P$ i $a(x).Q$ mogu uspostaviti komunikaciju preko kanala a, pri čemu će prvi poslati drugom ime y tim kanalom. Ime x će u procesu Q biti zamenjeno primljenim imenom y. U tabeli 2.3 je prikazana sintakasa sinhronog π-računa. Da bi se ostavrila sinhrona komunikacija, istovremeno se moraju dogoditi akcije prijema i slanja. Oba prefiksa (i input i output) blokiraju izvršavanje procesa koji ih slede.
- Pravilo **(Res)** kaže da se redukcija može nastaviti i pod restrikcijom.
- Pravilo **(Par)** objašnjava da paralelna kompozicija ne ometa redukciju.
- Pravilo **(Str)** izražava da strukturalno ekvivalentni procesi imaju iste redukcije.

2.4 Varijante i proširenja

U zavisnosti od osobina koje modeliraju razvijene različite varijante i proširenja π-računa. Pored operatora koje smo naveli, u sintaksi π-računa se mogu pojaviti i drugi, kao što su *Match*, if $x = y$ then P, i *Suma*, $P + Q$. Ubacivanjem operatora sume možemo neposredno izraziti automate u π-računu. Možemo definisati poliadičnu varijantu računa, u kojoj se kroz kanal može slati vektor imena. Komunikacija koju smo naveli u tabeli 2.3 je sinhrona, a kasnije će biti reči o asinhronoj kod koje samo input prefiks blokira izvršavanje procesa koji ga sledi, dok se output dešava slobodno.
Još jedan način za zadavanje semantike operacija je labelled transition system, gde su redukcije oblika $P \xrightarrow{\alpha} Q$. α predstavlja akciju koju proces P može

da izvede i posle koje se ponaša kao Q. Ovakav način zadavanja semantike pogodan je za uvođenje relacije bisimulacije. Bisimulacija je relacija ekvivalencije, čija jedna klasa ekvivalencije sadrži procese sa jednakim ponašanjem. Pod time podrazumevamo da ako umesto jednog procesa u nekom sistemu ubacimo drugi, iz iste klase ekvivalencije, to se neće odraziti na ponašanje sistema.

π-račun proširen lokacijama ima za cilj opis distrubuirnih sistema. U najjednostavnijoj formi, dodaju se proste lokacije u kojima nema podlokacija. Jedan od takvih računa je Distributed π (Riely i Hennessy [12]), koji je osnova za X$d\pi$ račun o kome će kasnije biti reči. Dodavanjem lokacija i operatora *go* omogućena je eksplicitna mobilnost procesa.

Spi račun (Abadi i Gordon [1]) je proširenje π računa za opis i analizu kriptografskih protokola. Na osnovu njega nastao je programski jezik *Spico* (A Stochastic Pi-Calculus with Concurrent Objects), namenjen modeliranju i analizi bioloskih sistema. Još neke od primena π-računa su *BPML* (Business Process Modeling Language): meta jezik za analiziranje procesa u poslovanju, *Nomadic Pict*, *occam-π*, *Pict*.

Glava 3

Ambijentni račun

Račun mobilnih ambijenata, Calculus of Mobile Ambients, (Cardeli i Gordon [5]) je model čija je osnovna namena opis mobilnosti procesa i uređaja, uključujući i prolazak kroz različite administrativne domene. Ambijent, u smislu u kojem ćemo ga ovde koristiti, ima sledeća svojstva:

- Ambijent je *ograničeno* mesto na kome se odvijaju procesi. Ograničenje određuje šta se nalazi unutar a šta izvan. Primeri ambijenata, sa ovog gledišta, su: web stranica (ograničena datotekom), objekat sa jednim podatkom (ograničen sobom) ili laptop (ograničen torbom i svojim portovima), za razliku od: procesa (kod kojeg je teško odrediti doseg) ili kolekcije objekata u logičkoj vezi. Postojanje ograničenja povlači potencijalno obraćanje entitetima u spoljašnjosti. Takvo fleksibilno obraćanje omogućava mobilnost ili je, u najmanju ruku, olakšava. Međutim može nastati problem ako dođe do prekida ili nestanka veza.

- Ambijent može da bude unutar drugih ambijenta. Na primer, administrativni domeni često su organizovani hijerarhijski. Ako želimo da prenesemo aktivnu aplikciju sa posla kući, aplikacija mora da napusti svoje okruženje (posao) i da se umetne u novo okruženje (kuća). Laptop može da poseduje dozvolu za iznošenje sa radnog mesta i potvrdu za ulazak ili izlazak iz zemlje.

- Ambijent možemo pomeriti u celini. Ako, na primer, povežemo laptop na novu mrežu, sve veze i podaci u njemu se shodno tome automatski promeraju.

Preciznije, razmatramo ambijente koji imaju sledeću strukturu:

- Svaki ambijent ima ime. Ime ambijenta se koristi za konrolu pristupa (ulazak, izlazak, komunikacija,...).

- Svaki ambijent sadrži kolekciju lokalnih agenata. To su procesi koji se izvršavaju unutar ambijenta i na neki način ga kontrolišu. Oni, na primer, mogu dati nalog ambijentu da se pomeri.

- Svaki ambijent ima kolekciju podambijenata. Svaki podambijent ima svoje ime, agente, podambijente itd.

3.1 Sintaksa

Prvo ćemo formalno zadati račun u celosti, a zatim objasniti pojedinačne konstrukcije. Sintaksa je definisana u tabelama 3.1, 3.2, 3.3 i 3.4. Dve glavne sintaktičke kategorije su procesi (uključujući i ambijente i agente koji izvršavaju akcije) i sposobnosti.

Supstituciju sposobnosti M umesto svakog pojavljivanja promenljive x u

$P, Q ::=$	$(\nu n)P$	restrikcija
	$\mid 0$	nil
	$\mid P \mid Q$	kompozicija
	$\mid\ !P$	replikacija
	$\mid M[P]$	ambijent
	$\mid M.P$	aktivacija sposobnosti
	$\mid (x).P$	akcija prijema (input)
	$\mid \langle M \rangle$	akcija asinhronog slanja (output)

Tabela 3.1: Procesi

$M ::=$	x	promenljiva
	$\mid n$	ime
	$\mid$ *in* M	može da uđe u M
	$\mid$ *out* M	može da izađe iz M
	$\mid$ *open* M	može da otvori M
	$\mid \varepsilon$	null(prazna)
	$\mid M.M'$	putanja

Tabela 3.2: Sposobnosti

procesu P obeležavamo sa $P\{x \leftarrow M\}$. Slično je i $M\{x \leftarrow M'\}$.

$fn((\nu n)P) \triangleq fn(P) - \{n\}$	$fn(in\ n) \triangleq \{n\}$
$fn(0) \triangleq \emptyset$	$fn(out\ n) \triangleq \{n\}$
$fn(P\|Q) \triangleq fn(P) \cup fn(Q)$	$fn(open\ n) \triangleq \{n\}$
$fn(!P) \triangleq fn(P)$	$fn(x) \triangleq \emptyset$
$fn(M[P]) \triangleq fn(M) \cup fn(P)$	$fn(n) \triangleq \{n\}$
$fn(M.P) \triangleq fn(M) \cup fn(P)$	$fn(\varepsilon) \triangleq \emptyset$
$fn((x).P) \triangleq fn(P)$	$fn(M.M') \triangleq fn(M) \cup fn(M')$
$fn(\langle M \rangle) \triangleq fn(M)$	

Tabela 3.3: Slobodna imena

$fv((\nu n)P) \triangleq fv(P)$	$fv(x) \triangleq \{x\}$
$fv(0) \triangleq \emptyset$	$fv(n) \triangleq \emptyset$
$fv(P\|Q) \triangleq fv(P) \cup fv(Q)$	$fv(in\ M) \triangleq fv(M)$
$fv(!P) \triangleq fv(P)$	$fv(out\ M) \triangleq fv(M)$
$fv(M[P]) \triangleq fv(M) \cup fv(P)$	$fv(open\ M) \triangleq fv(M)$
$fv(M.P) \triangleq fv(M) \cup fv(P)$	$fv(\varepsilon) \triangleq \emptyset$
$fv((x).P) \triangleq fv(P) - \{x\}$	$fv(M.M') \triangleq fv(M) \cup fv(M')$
$fv(\langle M \rangle) \triangleq fv(M)$	

Tabela 3.4: Slobodne promenljive

3.1.1 Objašnjenja

Sledi detaljniji opis operatora računa i neformalno objašnjenje relacije redukcije $P \to Q$ koja predstavlja prelazak procesa P u novi proces Q.

Restrikcija
Operator restrikcije, $(\nu n)P$, stvara novo (jednistveno) ime n u domenu procesa P. Kao i u π-računu, restrikcija νn može da proširi domen imena prema spolja ili da ga, ako je moguće, smanji prema unutra. Za razliku od π-računa, ime n koje je vezano operatorom restrikcije nije ime kanala, već ime ambijenta. Operator restrikcije se slaže sa relacijom redukcije, što je izraženo sledećim pravilom:

$$P \to Q \Rightarrow (\nu n)P \to (\nu n)Q$$

Nil
Proces 0 je proces koji ne radi ništa. On ne može da se redukuje.

Pralalelna kompozicija
Kao i u π-računu, istovremeno izvršavanje procesa označeno je binarnim operatorom koji je komutativan i asocijativan: $P|Q$. Podleže pravilu:

$$P \to Q \Rightarrow P|R \to Q|R$$

Ovo pravilo, primenjeno direktno, je dovoljno za redukciju leve strane. Redukcija desne strane se izvodi pomoću komutativnosti.

Replikacija
Replikacija je tehnički pogodan način za predstavljanje iteracije i rekurzije. Proces $!P$ označava nevezanu replikaciju procesa P tj. može da proizvede koliko god je potrebno paralelnih kopija procesa P. Ekvivalentan je sa $P|!P$. Nema redukcijskih pravila za $!P$, preciznije rečeno proces P pod ! ne može da se redukuje pre nego što se razvije na $P|!P$.

Ambijenti
Ambijent obeležavamo sa $n[P]$, gde je n ime ambijenta, a P proces koji se izvršava unutar ambijenta. U $n[P]$ se podrazumeva da se proces P aktivno izvršava i da može biti paralelna kompozicija nekoliko procesa. Naglasimo da se P izvršava i kada se njegovo okruženje (ambijent) pomera. Izvršavanje procesa P izražavamo pravilom da bilo koja redukcija P postaje redukcija $n[P]$:

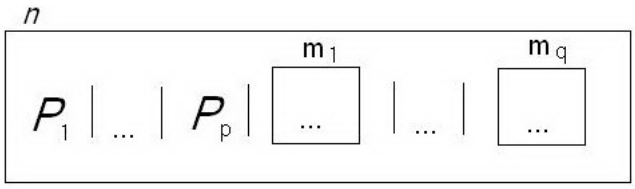
$$P \to Q \Rightarrow n[P] \to n[Q]$$

U opštem slučaju, ambijent ima strukturu stabla (drveta). Svaki čvor ovog drveta, pored podambijenata, može sadržati kolekciju (neambijentalnih) procesa koji se paralelno izvršavaju. Za te procese kažemo da se izvršavaju u

ambijentu, za razliku od onih koji se izvršavaju u podambijentima. Prema tome, opšti oblik ambijenta je:

$$n[P_1 \mid \ldots \mid P_p \mid m_1[\ldots] \mid \ldots \mid m_q[\ldots]\,] \qquad (P_i \neq n_i[\ldots])$$

Ako zagrade prikažemo kao ,,kutije", ambijent u opštem obliku, izgleda ovako:



Možemo imati dva ambijenta sa istim imenom, bilo ugnježdena, bilo istog nivoa. Šta više, $!n[P]$ generiše beskonačno ambijenta istog imena. Na taj način, na primer, možemo jednostavno modelirati replikaciju servera.

Akcije i sposobnosti

Operacije koje menjaju hijerarhijsku strukturu ambijenata su operacije koje se mogu inerpretirati kao, na primer, prolazak kroz firewall. Zbog toga su te operacije ograničene sposobnostima. Zahvaljujući njima, ambijent može da dozvoli drugim ambijentima da izvode neke operacije, nemorajući da im otkrije svoje pravo ime. Sposobnosti se mogu prenositi kanalima kao vrednosti.

Proces $M.P$ izvršava akciju kojom upravlja sposobnost M i nastavlja dalje kao proces P. Proces P ne pocinje da se izvršava pre akcije. Za svaku sposobnost M imamo posebno pravilo redukcije procesa $M.P$. Posmatramo tri vrste sposobnosti: jedna za ulazak u ambijent, druga za izlazak iz ambijenta i treća za otvaranje ambijenta. Za dato ime M, sposobnost *in* M obezbeđuđe dozvolu za ulazak u M, sposobnost *out* M dozvolu za izlazak a sposobnost *open* M dozvolu za otvaranje M. Posedovanje jedne ili više ovih sposobnosti nije dovoljno za rekonstrukciju početnog imena od kojeg su izdvojene.

Sposobnost ulaska

Sposobnost ulaska, *in* m, se može koristiti u akciji:

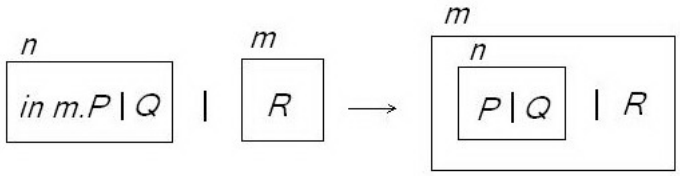
$$in\ m.P$$

koja daje nalog ambijentu koji okružuje *in* $m.P$ da uđe u ambijent m. Ako ambijent m trenutno ne postoji, ova operacija blokira izvršavanje procesa P

dok se odgovarajući ambijent ne pojavi. Ako postoji više od jednog ambijenta sa imenom m, bilo koji može biti izabran. Pravilo redukcije glasi:

$$n[in\ m.P|Q]|m[R] \rightarrow m[n[P|Q]|R]$$

Ili, ako zagrade predstavimo kutijama:



Posle uspešne redukcije proces $in\ m.P$ nastavlja kao P. I P i Q se, nakon redukcije, nalaze na nižem nivou u drvetu ambijenata.

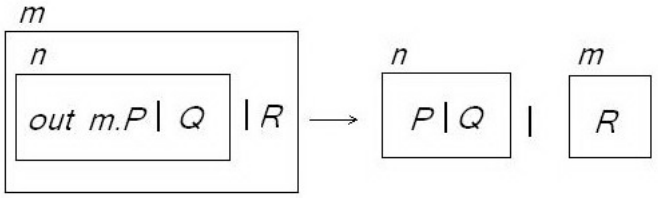
Sposobnost izlaska
Sposobnost izlaska, $out\ m$, se može koristiti u akciji:

$$out\ m.P$$

koja daje nalog ambijentu koji okružuje $out\ m.P$ da izađe iz ambijenta m. Ako ambijent-roditelj nema ime m, ova operacija blokira izvršavanje procesa P dok se odgovarajući ambijent-roditelj ne pojavi. Pravilo redukcije glasi:

$$m[n[out\ m.P|Q]|R] \rightarrow n[P|Q]|m[R]$$

To jest:



Posle uspešne redukcije proces $out\ m.P$ nastavlja kao P. I P i Q se, nakon redukcije, nalaze na višem nivou u drvetu ambijenata.

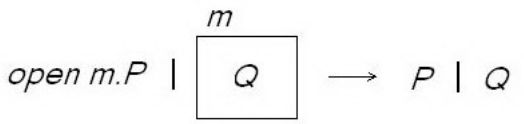
Sposobnost otvaranja
Sposobnost otvaranja, $open\ m$, se može koristiti u akciji:

$$open\ m.P$$

Ova akcija obezbeđuje način za uklanjanje ograničenja ako ambijenta imena m, koji se nalazi na istom nivou kao i *open*. Pravilo redukcije glasi:

$$open\ n.P|n[Q] \rightarrow P|Q$$

To jest:



Ako ambijent m trenutno ne postoji, ova operacija blokira izvršavanje procesa P dok se odgovarajući ambijent ne pojavi. Ako postoji više od jednog ambijenta sa imenom m, bilo koji može biti izabran.

Prenosive vrednosti
Objekti koji se mogu prenositi komunikacionim kanalima su imena i sposobnosti. U realnoj situaciji, razmena imena bi trebalo da bude retka, znajući da ime ambijenta ima veliku kontrolu nad njim. Umesto toga, obično bi trebalo razmenjivati vezane sposobnosti da bi se postigla kontrolisana interakcija između ambijenata. Korisno je kombinovati višestruke sposobnosti u putanje, $(M.M')$, posebno kada su neke od tih sposobnosti predstavljene sa ulaznim promenljivima. Primetimo da se, zbog komunikacije, među sposobnostima nalaze i imena. Ime predstavlja sposobnost kreiranja ambijenta tog imena. Postoji razlika između ν -vezanih imena i input vezanih promenljivih. Promenljive mogu biti instancirane imenima sposobnosti. U praksi, ne moramo da pravimo lesksičku razliku iazmeđu ove dve vrste, mada često koristimo n, m, p, q za imena a w, x, y, z za promenljive.

I/O komunikacija u ambijentu
Najjednostavniji mehanizam komunikacije koji možemo predstaviti je anonimna komunikacija unutar ambijenta. Akcija slanja (output) oslobađa sposobnost (ili ime) a u lokalni etar ambijenta koji ga okružuje. Akcija prijema (input) prihvata sposobnost iz lokalnog etra i pridružuje je promenljivoj unutar njenog domena. Imamo redukciju:

$$(x).P|\langle M\rangle \rightarrow P\{x \leftarrow M\}$$

Ovakav mehanizam lokalne komunikacije se uklapa u ideju ambijenata. Preciznije, komunikaciju na daljinu, kao ni mobilnost na "veliku"razdaljinu, nije poželjno obavljati automatski jer postoji mogućnost da poruke moraju proći

kroz, recimo, firewall ili neku drugu vrstu kontrole. Međutim, ovaj jednostavni mehanizam je dovoljan, kao što ćemo videti, za utapanje asinhronog π-računa.

Nepravilnost u sintaksi

Da bismo dobili da i imena i sposobnosti mogu da budu i input i output, imamo samo jednu sintaksnu kategoriju koja uključuje oboje. Međutim, na taj način se mogu dobiti termi bez smisla, kao na primer $n.P$ iz $(x).x.P \mid \langle n \rangle$. Do ove nepravilnosti je doslo zbog potrebe za uvođenjem promenljivih za sposobnosti ($(x).x.P$) i imena ($(x).x[P]$). Ova nepravilnost se može prevazići tipskim sistemom koji razdvaja imena od sposobnosti, kao u [6] ili modifikacijom sintakse, kao u [7].

3.1.2 Strukturalna konguencija

U tabeli 3.5 data je relacija strukturalne kongruencije, $\equiv$. Strukturalna kongruencija je relacija ekvivalenciije. Procesi su grupisani u klase ekvivalencije relacije, koje predstavljaju procese ekvivalentne do na trivijalno sintaksno restrukturiranje.

Takođe, izjednačavamo procese do na preimenovanje vezanih imena:

$$(\nu n)P = (\nu m)P\{n \leftarrow m\} \text{ if } m \notin fn(P)$$

$$(x).P = (y).P\{x \leftarrow y\} \quad \text{if } y \notin fv(P)$$

Strukturalno ekvivalentne procese smatramo identičnim.

3.2 Semantika operacija

3.2.1 Pravila redukcije

Ponašanje procesa je formalno zadato relacijom redukcije u tabeli 3.6, dok smo objašnjenja u poglavlju 3.1.1. Ako se proces P redukuje na proces Q iz više koraka tj. $P \to Q_1 \to \cdots \to Q_k \to Q$, to zapisujemo $P \to^* Q$. Preciznije, $\to^*$ je refleksivno i tranzitivno zatvorenje $\to$.

3.2.2 Primer

Provera mobilnog agenta

Za neki proces koji je nalazi na najvišem nivou u nekom ambijentu možemo reći da je privilegovan, jer može da utiče da mobilnost okružujućeg ambijenta i može da otvara podambijente. Pretpostavimo da takav, privilegovani, proces želi da napusti svoj *Home* ambijent, pa de se potom vrati i da povrati

$P \equiv P$	(Str Refl)
$P \equiv Q \Rightarrow Q \equiv P$	(Str Sim)
$P \equiv Q, Q \equiv R \Rightarrow P \equiv R$	(Str Tran)
$P \equiv Q \Rightarrow (\nu n)P \equiv (\nu n)Q$	(Str Res)
$P \equiv Q \Rightarrow P\|R \equiv Q\|R$	(Str Par)
$P \equiv Q \Rightarrow !P \equiv !Q$	(Str Repl)
$P \equiv Q \Rightarrow M[P] \equiv M[Q]$	(Str Amb)
$P \equiv Q \Rightarrow M.P \equiv M.Q$	(Str Akcija)
$P\|Q \equiv Q\|P$	(Str Par Komut)
$(P\|Q)\|R \equiv P\|(Q\|R)$	(Str Par Asoc)
$!P \equiv P\|!P$	(Str Repl Par)
$(\nu n)(\nu m)P \equiv (\nu m)(\nu n)P$	(Str Res Res)
$(\nu n)(P\|Q) \equiv P\|(\nu n)Q$ if $n \notin fn(P)$	(Str Res Par)
$(\nu n)(m[P]) \equiv m[(\nu n)P]$ if $n \neq m$	(Str Res Amb)
$P\|0 \equiv P$	(Str Nil Par)
$(\nu n)0 \equiv 0$	(Str Nil Res)
$!0 \equiv 0$	(Str Nil Repl)
$P \equiv Q \Rightarrow M(x).P \equiv M(x).Q$	(Str Prijem)
$\varepsilon.P \equiv P$	(Str ε))
$(M.M').P \equiv M.M'.P$	(Str .)

Tabela 3.5: Strukturalna konguencija

$n[in\ m.P\|Q]\|m[R] \rightarrow m[n[P\|Q]\|R]$	(Red In)
$m[n[out\ m.P\|Q]\|R] \rightarrow n[P\|Q]\|m[R]$	(Red Out)
$open\ n.P\|n[Q] \rightarrow P\|Q$	(Red Open)
$P \rightarrow Q \Rightarrow (\nu n)P \rightarrow (\nu n)Q$	(Red Res)
$P \rightarrow Q \Rightarrow n[P] \rightarrow n[Q]$	(Red Amb)
$P \rightarrow Q \Rightarrow P\|R \rightarrow Q\|R$	(Red Par)
$(x).P\|\langle M\rangle \rightarrow P\{x \leftarrow M\}$	(Red Kom)

Tabela 3.6: Redukcuja

svoje privilegije. Ambijent *Home* ne može da dozvoli bilo kom posetiocu da postane privilegovan, iz bezbednosnih razloga, te stoga izvorni proces mora na neki način biti proveren.
Rešenje je dato ispod. Proces najvišeg nivoa stvara novo (privatno) ime, n, koje će se koristiti kao "tajni ključ" između tog procesa i *Home* ambijenta; *open* n stoji u *Home* da bi moglo da proveri proces, kada se vrati. Proces napušta ambijent *Home* u obliku ambijenta *Agent*. Po povratku u *Home*, ambijent *Agent* otkriva ambijent n, koji se otvara sa *open* n da bi proces P nasavio izvršavanje kao proces najvišeg nivoa.

$Home[(\nu n)(open\ n \mid Agent[out\ home.in\ home.n[out\ Agent.open\ Agent.P]])]$

$\equiv \quad (\nu n)Home[open\ n|Agent[out\ home.in\ home.n[out\ Agent.open\ Agent.P]]]$

$\rightarrow \quad (\nu n)(Home[open\ n]|Agent[in\ home.n[out\ Agent.open\ Agent.P]])$

$\rightarrow \quad (\nu n)Home[open\ n|Agent[n[out\ Agent.open\ Agent.P]]]$

$\rightarrow \quad (\nu n)Home[open\ n|n[open\ Agent.P]|Agent[\ ]]$

$\rightarrow \quad (\nu n)Home[0|open\ Agent.P|Agent[\ ]]$

$\rightarrow \quad (\nu n)Home[0|P|0]$

$\equiv \quad Home[P]$

3.2.3 Kontekstualna ekvivalencija

Kontekstualna ekvivalencija je standardan način iskazivanja da dva procesa imaju isto ponašanje: dva procesa su kontekstualno ekvivalentna ako i samo ako svaki put kada ih ubacimo u prizvoljno okruženje, oni mogu da izvršavaju iste elementarne obzervacije.
U našem slucaju, kontekstualnu ekvivalenciju formulišemo u odnosu na obzervaciju prisustva ambijenata najvišeg nivoa. Kažemo da proces P *pokazuje ambijent* n, pišemo $P \downarrow n$, samo ako je P proces koji sadrži ambijent najvišeg nivoa sa imenom n. Kažemo da proces P *kasnije pokazuje ambijent* n, pišemo $P \Downarrow n$, samo ako nakon nekog broja redukcija P pokazuje ambijent n. Formalno definišano:

$$P \downarrow n \quad \triangleq P \equiv (\nu m_1 \ldots m_i)(n[P']|P'') \quad \text{gde } n \notin \{m_1 \ldots m_i\}$$
$$P \Downarrow n \quad \triangleq P \rightarrow^* Q \text{ i } Q \downarrow n$$

Sada definišimo kontekstualnu ekvivalenciju u zavisnosti od predikata $P \Downarrow n$. Neka je kontekst $C()$ proces koji ima nula ili više praznina. Tada za svaki proces P, proces $C(P)$ dobijamo popunjavanjem svake praznine u C sa kopijom P (neka slobodna imena procesa P mogu postati vezana). *Kontekstualna*

ekvivalencija,$P \simeq Q$, je definisana sa:

$$P \simeq Q \quad \triangleq \text{za sve } n \text{ i } C(), C(P) \Downarrow n \Leftrightarrow C(Q) \Downarrow n$$

Konačno, pišemo $P \rightarrow^* \simeq Q$ ako postoji R takvo da $P \rightarrow^* R$ and $R \simeq Q$. Naredne dve propozicije iskazuju svojstva konekstualne ekvivalencije. Neka je relacija $\mathcal{R}$ predcongruencija ako i samo ako za sve P, Q i $C()$ ako $P\mathcal{R}Q$ onda $C(P)\mathcal{R}C(Q)$. Ako je $\mathcal{R}$ takođe i refleksivna, simetrična i tranzitivna kažemo da je kongruencija. Na primer, strukturalna kongruencija je kongruencija. Štaviše jednostavno se pokazuje da je i kontekstualna ekvivalencija kongruencija.

Propozicija 3.1 *Kontekstualna ekvivalencija je kongruencija.*

Lema 3.2 *Ako je* $P \equiv Q$ *i* $Q \downarrow n$ *onda* $P \downarrow n$.

Dokaz Ako je $Q \downarrow n$, onda je $Q \equiv (\nu m_1, \ldots, m_k)(n[Q'] \mid Q'')$ gde $n \notin \{m_1, \ldots, m_k\}$. Koristeći pretpostavku $P \equiv Q$, pomoću tranzitivnosti dobijamo da je $P \equiv (\nu m_1, \ldots, m_k)(n[Q'] \mid Q'')$, pa je zato $P \downarrow n$.
□

Lema 3.3 *Ako je* $P \equiv Q$ *i* $Q \Downarrow n$ *onda* $P \Downarrow n$.

Dokaz Po definiciji iz $Q \Downarrow n$ sledi $Q \rightarrow^* R$ i $R \downarrow n$. Kako znamo (Red $\equiv$) i lemu 3.2, imamo da $P \rightarrow^* R$, pa sledi da $P \Downarrow n$.
□

Lema 3.4 *Ako* $P \equiv Q$ *onda* $P \simeq Q$.

3.3 Kodiranje π-računa

Jedno od merila ekspresivne moći Ambijentnog računa je mogućnost kodiranja π-računa. Kodiranje koje ovde dajemo je relativno jednostavno. Ključna ideja za potupunu translaciju je predstavljanje imena kanala.
Kanal jednostavno predstavljamo ambijentom: ime kanala je ime ambijenta. Komunikacija preko kanala je predstavljena lokalnom komunikacijom, unutar ambijenta. Ime *io* ćemo koristiti za prenos ulaznih i izlaznih zahteva na kanal. Kanal otvara sve takve zahteve i dozvoljava njihovu interakciju.

$buf\ n$	$\triangleq n[!open\ io]$	bafer kanala
$(ch\ n)P$	$\triangleq (\nu n)(buf\ n \mid P)$	novi kanal
$n(x).P$	$\triangleq (\nu p)(io[in\ n.(x).p[out\ n.P]] \mid open\ p)$	prijem kanalom
$n\langle M \rangle$	$\triangleq io[in\ n.\langle M \rangle]$	asinhrono slanje kanalom

$P, Q ::=$	$(\nu n)P$	restrikcija
	$\mid P\|Q$	kompozicija
	$\mid\ !P$	replikacija
	$\mid M(x).P$	akcija prijema
	$\mid M\langle M'\rangle$	akcija asinhronog slanja

Tabela 3.7: Procesi

$M ::=$	x	promenljiva
	$\mid n$	ime

Tabela 3.8: Izrazi

Ove definicije zadovoljavaju očekivanu redukciju $n(x).P \mid n\langle M\rangle \rightarrow^* P\{x \leftarrow M\}$ ako imamo bafer kanala $buf\ n$:

$$
\begin{aligned}
& buf\ n \mid n(x).P \mid n\langle M\rangle \\
& \equiv (\nu p)(n[!open\ io] \mid io[in\ n.(x).p[out\ n.P]] \mid open\ p \mid io[in\ n.\langle M\rangle]) \\
& \rightarrow^* (\nu p)(n[!open\ io \mid io[(x).p[out\ n.P]] \mid io[\langle M\rangle]] \mid open\ p) \\
& \rightarrow^* (\nu p)(n[!open\ io \mid (x).p[out\ n.P] \mid \langle M\rangle] \mid open\ p) \\
& \rightarrow (\nu p)(n[!open\ io \mid p[out\ n.P\{x \leftarrow M\}]] \mid open\ p) \\
& \rightarrow (\nu p)(n[!open\ io] \mid p[P\{x \leftarrow M\}] \mid open\ p) \\
& \rightarrow (\nu p)(n[!open\ io] \mid P\{x \leftarrow M\}) \\
& \equiv buf\ n \mid P\{x \leftarrow M\}
\end{aligned}
$$

Gorenavedene definicije kanala su pogodne za utapanje komunikacije na kanalima u ambijentni račun (pod uslovom da ime io ne koristimo u druge svrhe). Međutim, komunikacija na takvim kanalima funkcioniše samo unutar jednog ambijenta. Drugim rečima, sa ovog stanovišta, proces iz π-računa se uvek nalazi u istom ambijentu. Prema tome, pojam mobilnosti u π-računu (razmena imena preko kanala) je donekle različit od smisla mobilnosti.
Da bismo precizirali ideju ove translacije, dajemo formalizaciju asinhronog π-računa u tabelama 3.7, 3.8, 3.9, 3.10, 3.11, 3.12. Smatramo da su sva imena n koja su vezana restrikcijom različita od promenljivih x vezanih input prefiksom. Imamo odvojene funkcije fn i fv za slobodna imena i slobodne promenljive, redom.

Kodiranje asinhronog π-računa u ambijentni račun dato je presilkavanjem iz tabele 3.13. Preslikavamo svaki proces najvišeg nivioa u kontekst skupa imena S. Za S možemo uzeti skup svih slobodnih imena procesa. Ovo

$$
\begin{aligned}
&fn((\nu n)P) \triangleq fn(P) - \{n\}\\
&fn(P|Q) \triangleq fn(P) \cup fn(Q)\\
&fn(!P) \triangleq fn(P)\\
&fn(M(x).P) \triangleq fn(M) \cup fn(P)\\
&fn(M\langle M'\rangle) \triangleq fn(M) \cup fn(M')\\
&fn(x) \triangleq \emptyset\\
&fn(n) \triangleq \{n\}
\end{aligned}
$$

Tabela 3.9: Slobodna imena

$$
\begin{aligned}
&fv((\nu n)P) \triangleq fv(P)\\
&fv(P|Q) \triangleq fv(P) \cup fv(Q)\\
&fv(!P) \triangleq fv(P)\\
&fv(M(x).P) \triangleq fv(M) \cup (fv(P) - \{x\})\\
&fv(M\langle M'\rangle) \triangleq fv(M) \cup fv(M')\\
&fv(x) \triangleq \{x\}\\
&fv(n) \triangleq \emptyset
\end{aligned}
$$

Tabela 3.10: Slobodne promenljive

$P \equiv P$	(Str Refl)
$P \equiv Q \Rightarrow Q \equiv P$	(Str Sim)
$P \equiv Q, Q \equiv R \Rightarrow P \equiv R$	(Str Tran)
$P \equiv Q \Rightarrow (\nu n)P \equiv (\nu n)Q$	(Str Res)
$P \equiv Q \Rightarrow P\|R \equiv Q\|R$	(Str Par)
$P \equiv Q \Rightarrow !P \equiv !Q$	(Str Repl)
$P \equiv Q \Rightarrow M(x).P \equiv M(x).Q$	(Str Prijem)
$P\|Q \equiv Q\|P$	(Str Par Komut)
$(P\|Q)\|R \equiv P\|(Q\|R)$	(Str Par Asoc)
$!P \equiv P\|!P$	(Str Repl Par)
$(\nu n)(\nu m)P \equiv (\nu m)(\nu n)P$	(Str Res Res)
$(\nu n)(P\|Q) \equiv P\|(\nu n)Q$ if $n \notin fn(P)$	(Str Res Par)
$(\nu n)P \equiv P$ if $n \notin fn(P)$	(Str Res fn)

Tabela 3.11: Strukturalna konguencija

$$
\begin{array}{ll}
n\langle m\rangle \mid n(x).P \to P\{x \leftarrow m\} & \text{(Red Kom)} \\
P \to Q \Rightarrow (\nu n)P \to (\nu n)Q & \text{(Red Res)} \\
P \to Q \Rightarrow P|R \to Q|R & \text{(Red Par)} \\
P' \equiv P, P \to Q, Q \equiv Q' \Rightarrow P' \to Q' & \text{(Red } \equiv \text{)}
\end{array}
$$

Tabela 3.12: Redukcija

$$
\begin{array}{lll}
\langle\!\langle P\rangle\!\rangle_S & \triangleq \langle\!\langle S\rangle\!\rangle \mid \langle\!\langle P\rangle\!\rangle & \text{gde je } S \text{ skup imena} \\
\langle\!\langle\{n_1,\ldots,n_k\}\rangle\!\rangle & \triangleq n_1[!open\ io] \mid \ldots \mid n_k[!open\ io] & \\
\langle\!\langle(\nu n)P\rangle\!\rangle & \triangleq (\nu n)(n[!open\ io] \mid \langle\!\langle P\rangle\!\rangle) & \\
\langle\!\langle P|Q\rangle\!\rangle & \triangleq \langle\!\langle P\rangle\!\rangle \mid \langle\!\langle Q\rangle\!\rangle & \\
\langle\!\langle !P\rangle\!\rangle & \triangleq\ !\langle\!\langle P\rangle\!\rangle & \\
\langle\!\langle M(x).P\rangle\!\rangle & \triangleq (\nu p)(io[in\ M.(x).p[out\ M.\langle\!\langle P\rangle\!\rangle]] \mid open\ p) & \\
\langle\!\langle M\langle M'\rangle\rangle\!\rangle & \triangleq io[in\ M\langle M'\rangle] &
\end{array}
$$

Tabela 3.13: Kodiranje asinhronog π-računa

kodiranje uključuje i sinhroni π-račun, jer se on može kodirati u asinhroni π-račun. Ovakvim kodiranjem smo poštovali semantiku asinhronog π-računa, u smislu da se redukcijski korak u asinhronom π-računu može simulirati sa sekvencom redukcijskih koraka i ekvivalencija u ambijentom računu, kao što je i pokazano u teoremi 3.6. Pretpostavljamo da *io* ne pripada skupu imena π-računa.

Lema 3.5 *Supstitucija*

$$\langle\!\langle P\rangle\!\rangle\{x \leftarrow m\} = \langle\!\langle P\{x \leftarrow m\}\rangle\!\rangle$$

□

Propozicija 3.6 *1. Ako $P \equiv P'$ važi u π-računu i ako je S skup imena, tada $\langle\!\langle P\rangle\!\rangle_S \simeq \langle\!\langle P'\rangle\!\rangle_S$.*

2. Ako $P \to P'$ važi u π-računu i ako je $S \supseteq fn(P)$, tada $\langle\!\langle P\rangle\!\rangle_S \simeq \to^* \simeq \langle\!\langle P'\rangle\!\rangle_S$.

Dokaz U ovom dokazu, koristićemo osnovna svojstva relacije $\simeq$,

1. Indukcijom po dužini izvođenja $P \equiv P'$ pokazujemo da $\langle\!\langle P\rangle\!\rangle \simeq \langle\!\langle P'\rangle\!\rangle$. Odatle sledi $\langle\!\langle P\rangle\!\rangle_S \simeq \langle\!\langle P'\rangle\!\rangle_S$ (tj $\langle\!\langle S\rangle\!\rangle|\langle\!\langle P\rangle\!\rangle \simeq \langle\!\langle S\rangle\!\rangle|\langle\!\langle P'\rangle\!\rangle$) zbog kongruentnosti $\simeq$.

(Struct Refl), **(Struct Sym)**, **(Struct Tran)**, **(Struct Par)**, **(Struct Repl)**, **(Struct Par Komut)**, **(Struct Par Asoc)**, **(Struct Repl Par)**. Slede direktno iz definicija i indukcijske hipoteze.

(Struct Res) $Q \equiv Q' \Rightarrow (\nu n)Q \equiv (\nu n)Q'$. Po indukcijskoj hipotezi važi $\langle\langle Q\rangle\rangle \simeq \langle\langle Q'\rangle\rangle$. Kako je $\simeq$ kogruencija, dobijamo da $(\nu n)(n[!open\ io] \mid \langle\langle Q\rangle\rangle) \simeq (\nu n)(n[!open\ io] \mid \langle\langle Q'\rangle\rangle)$. Tj. da je $\langle\langle(\nu n)Q\rangle\rangle \simeq \langle\langle(\nu n)Q'\rangle\rangle$

(Struct Res Res) $(\nu n)(\nu m)Q \equiv (\nu m)(\nu n)Q$.

$$\begin{aligned}\langle\langle(\nu n)(\nu m)Q\rangle\rangle &= (\nu n)(n[!open\ io] \mid (\nu m)(m[!open\ io] \mid \langle\langle Q\rangle\rangle)) \\ &\equiv (\nu m)(m[!open\ io] \mid (\nu n)(n[!open\ io] \mid \langle\langle Q\rangle\rangle)) \\ &= \langle\langle(\nu m)(\nu n)Q\rangle\rangle\end{aligned}$$

(Struct Prijem) $Q \equiv Q' \Rightarrow M(x).Q \equiv M(x).Q'$. Po indukcijskoj hipotezi imamo: $\langle\langle Q\rangle\rangle \simeq \langle\langle Q'\rangle\rangle$. Kako je $\simeq$ kongruencija, imamo da je $(\nu p)(io[in\ M.(x).p[out\ M.\langle\langle Q\rangle\rangle]] \mid open\ p) \simeq (\nu p)(io[in\ M.(x).p[out\ M.\langle\langle Q'\rangle\rangle]] \mid open\ p)$. To jest da je $\langle\langle M(x).Q\rangle\rangle \simeq \langle\langle M(x).Q'\rangle\rangle$.

(Struct Res Par) $(\nu n)(Q'|Q'') \equiv Q \mid (\nu n)Q''$ ako $n \notin fn(Q')$. Primetimo da važi ili $fn(\langle\langle Q'\rangle\rangle) = fn(Q')$ ili $fn(\langle\langle Q'\rangle\rangle) = fn(Q') \cup \{io\}$, i da je $n \neq io$ po globalnoj konvenciji. Zbog toga, iz $n \notin fn(Q')$ sledi da $n \notin fn(\langle\langle Q'\rangle\rangle)$. Dalje je

$$\begin{aligned}\langle\langle(\nu n)(Q' \mid Q'')\rangle\rangle &= (\nu n)(n[!open\ io] \mid (\langle\langle Q'\rangle\rangle \mid \langle\langle Q''\rangle\rangle)) \\ &\equiv \langle\langle Q'\rangle\rangle \mid (\nu n)(n[!open\ io] \mid \langle\langle Q''\rangle\rangle) \\ &= \langle\langle Q \mid (\nu n)Q''\rangle\rangle\end{aligned}$$

.

(Struct Res fn) $(\nu n)Q \equiv Q$ ako $n \notin fn(Q)$. Kao i u prethodnom slučaju, $n \notin fn(\langle\langle Q\rangle\rangle)$. Tada je$\langle\langle(\nu n)Q\rangle\rangle = (\nu n)(n[!open\ io] \mid \langle\langle Q\rangle\rangle) \equiv \langle\langle Q\rangle\rangle \mid (\nu n)n[!open\ io]$. Po !!!!!!!, imamo da je $(\nu n)n[!open\ io] \simeq 0$. Pa je $\langle\langle(\nu n)Q\rangle\rangle \simeq \langle\langle Q\rangle\rangle$.

2. Indukcijom po dužini izvođenja $P \to P'$.

(Red Kom) $n\langle m\rangle \mid n(x).Q \to Q\{x \leftarrow m\}$. Treba da pokažemo da ako je $S \supseteq fn(n\langle m\rangle \mid n(x).Q)$, onda je $\langle\langle n\langle m\rangle \mid n(x).Q\rangle\rangle_S \simeq\to^*\simeq \langle\langle Qx \leftarrow m\rangle\rangle_S$. Znamo da je $\langle\langle n\langle m\rangle \mid n(x).Q\rangle\rangle = \langle\langle S\rangle\rangle \mid io[in\ n.\langle m\rangle] \mid (\nu p)(io[in\ n.(x).p[out\ n.\langle\langle Q\rangle\rangle]] \mid open\ p)$. Po pretpostavci, n se nalazi u S, pa se zato $n[!open\ io]$ nalazi u $\langle\langle S\rangle\rangle$. Dalje se, izračunavanjem sa početka poglavlja i refleksivnošću $\simeq$, dobija $\langle\langle n\langle m\rangle \mid n(x).Q\rangle\rangle_S \simeq\to^*\simeq \langle\langle S\rangle\rangle \mid \langle\langle Q\rangle\rangle\{x \leftarrow m\}$. Po lemi o supstituciji !!! ref, desna strana je

jednaka sa $\langle\langle S\rangle\rangle \mid \langle\langle Q\{x \leftarrow m\}\rangle\rangle$, a to je jednako sa $\langle\langle Qx \leftarrow m\rangle\rangle_S$.

(Red Res) $Q \equiv Q' \Rightarrow (\nu n)Q \equiv (\nu n)Q'$. Treba da pokažemo da ako je $S \supseteq fn((\nu n)Q)$ onda je $\langle\langle(\nu n)Q\rangle\rangle_S \simeq\rightarrow^*\simeq \langle\langle(\nu n)Q'\rangle\rangle_S$. Ako je $S \supseteq fn((\nu n)Q)$, onda je $S \cup \{n\} \supseteq fn(Q)$. Pošto izjednačavamo terme do na preimenovanje vezanih promenljivih, možemo da pretpostavimo da $n \notin S$. Po indukcijskoj hipotezi važi $\langle\langle Q\rangle\rangle_{S\cup\{n\}} \simeq\rightarrow^*\simeq \langle\langle Q'\rangle\rangle_{S\cup\{n\}}$. Višetrukom primenom (Res Res) i kongruentnosti $\simeq$, izvodimo da je $(\nu n)\langle\langle Q\rangle\rangle_{S\cup\{n\}} \simeq\rightarrow^*\simeq (\nu n)\langle\langle Q'\rangle\rangle_{S\cup\{n\}}$. Kako je

$$\begin{aligned}(\nu n)\langle\langle Q\rangle\rangle_{S\cup\{n\}} &= (\nu n)(\langle\langle S \cup \{n\}\rangle\rangle \mid \langle\langle Q\rangle\rangle)\\ &\equiv (\nu n)(n[!open\ io] \mid \langle\langle S\rangle\rangle \mid \langle\langle Q\rangle\rangle)\\ &\equiv \langle\langle S\rangle\rangle \mid \langle\langle(\nu n)Q\rangle\rangle\\ &= \langle\langle(\nu n)Q\rangle\rangle_S\end{aligned}$$

i slično $(\nu n)\langle\langle Q'\rangle\rangle_{S\cup\{n\}} \equiv \langle\langle(\nu n)Q'\rangle\rangle_S$, dobijamo da je $\langle\langle(\nu n)Q\rangle\rangle_S \simeq\rightarrow^*\simeq \langle\langle(\nu n)Q'\rangle\rangle_S$.

(Red Par) $Q \rightarrow Q' \Rightarrow Q|R \rightarrow Q'|R$. Po indukcijskoj hipotezi je $\langle\langle Q\rangle\rangle_S \simeq\rightarrow^*\simeq \langle\langle Q'\rangle\rangle_S$. Višetrukom primenom (Res Par) i kongruentnosti $\simeq$, izvodimo da je $\langle\langle Q\rangle\rangle_S \mid \langle\langle R\rangle\rangle \simeq\rightarrow^*\simeq \langle\langle Q'\rangle\rangle_S \mid \langle\langle R\rangle\rangle$,to jest da je $\langle\langle Q|R\rangle\rangle_S \simeq\rightarrow^*\simeq \langle\langle Q'|R\rangle\rangle_S$.

(Red $\equiv$) $Q' \equiv Q, Q \rightarrow R, R \equiv R' \Rightarrow Q' \rightarrow R'$. Po indukcijskoj hipotezi i (1.) je $\langle\langle Q'\rangle\rangle_S \simeq \langle\langle Q\rangle\rangle_S, \langle\langle Q\rangle\rangle_S \simeq\rightarrow^*\simeq \langle\langle R\rangle\rangle_S, \langle\langle R\rangle\rangle_S \simeq \langle\langle R'\rangle\rangle_S$, pa imamo $\langle\langle Q'\rangle\rangle_S \simeq\rightarrow^*\simeq \langle\langle R'\rangle\rangle_S$ zbog tranzitivnosti $\simeq$.

□

Napomenimo da možemo definisati ambijentni račun bez komunikacije koji je dovoljno izražajan da bi bio Turing-kompletan.
Mobilnost procesa u π-računu predstavljena je mobilnošću veza koje taj proces poseduje. Stoga, ne možemo reći da u π-računu imamo ekplicitnu mobilnost. Međutim mnogi fundamentalni koncepti i tehnike iz ovog računa uključeni su u ambijentni račun.

Glava 4

Mobilni procesi i dinamički web podaci

Ovde predstavljamo, X$d\pi$-račun (Gardner i Maffeis [10]), matematički model distribuiranog sistema u kojem su raspoređeni podaci. Ovaj račun je jedan od pokušaja da se povežu istraživanja mobilnih procesa i podataka za web aplikacije. Web podaci, kao što je XML, igraju ključnu ulogu u razmeni informacija između globalno distribuiranih aplikacija. Ovakvi podaci se ne čuvaju samo statički. Često im se može obraćati indirektno, na primer korišćenjem hiperlinkova, service call-a ili skriptova za dinamički prisrup podacima. To zahteva kompleksnu koordinaciju između podataka i procesa raspoređenih na različitim lokacijama. Peer-to-peer (decentralizovani) distribuirani sistemi za upravljanje podacima, kakve ovde razmatramo, u kojima svaka komponenta ima jednake osnovne funkcije i istovremeno, i generiše i konzumira informacije. Svaka koponenta (*lokacija*) se sastoji od iz dva dela. U jednom se nalaze XML podaci, a u drugom aktivni procesi. Procesi, koji se ovde razmatraju, opisuju agente koji mogu da komuniciraju međusobno, prisupaju podacima ili da migriraju na druge lokacije da bi tamo nastavili izvršavanje. Definicije procesa mogu biti sastavni deo podataka i mogu biti odabrane za pokretanje od strane drugih procesa.

4.1 Sinatksa

U X$d\pi$-računu peer-to-peer sitem se modelira pomoću grupe lokacija (*mreža*), pri čemu sadržaj svake lokacije čini jedan term koji predstavlja podatke (*drvo*) i jedan (*proces*) koji predstavlja servise koje pruža lokacija-peer, agente koje se izvršavaju prilikom prelaska u druge delove sistema i lokalne agente koji čekaju komande agenata sa drugih lokacija.

Lokacija

$$l[T \parallel P]$$

Na lokaciji l imamo drvo sa podacima T i proces P. Lokacija l je dobro definisana ako ni drvo T ni proces P ne sadrže pojavljivanja slobodnih promenljivih.

Mreža

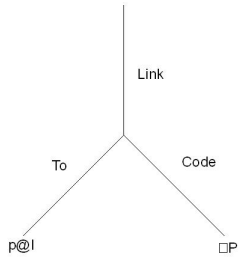
$$\mathbf{N} ::= 0 \mid \mathbf{N} \mid \mathbf{N} \mid l[T \parallel] \mid (\nu c)\mathbf{N}$$

Mreže predstavljamo paralelnom kompozijom pojedničnih lokacija. U dobro definisanoj mreži sve lokacije imaju različita imena. Operator ν, kao i do sada, označava restrikciju.

Drvo

Model za podatke je neuređeno drvo sa korenom i označenim granama. Podaci se nalaze u listovima. Tabela 4.1 sintaksu drveta.

< `Link` >
< `To` >
$p@l$
< /`To` >
< `Code` >
P
< /`Code` >
< /`Link` >



`Link`[`To`[$p@l$]|`Code`[$\Box P$]]

Zbog pojednostavljavanja sintakse razmatramo samo dve vrste podataka: skriptove i pokazivače.

Skript, $\Box P$, je statičan proces u drvetu na nekoj lokaciji, koji čeka da bude aktiviran od strane procesa sa iste lokacije. Π može da bude proces ili promenljiva.

Putanja, p, pronalazi čvorove u drvetu. Tabela 4.2 daje pravila formiranja putanje. U nekoj putanji, "`a`" označava korak po duž grane označene sa `a`, " // " označava proizvoljan čvor, ".." korak unazad, "$\bullet$ " putanju od korena to trenutnog čvora, "x" promenljivu, a "/ " kompoziciju putanja. Kažemo da je neka putanja, lokalna putnja, ako sadrži "$\bullet$ ". Lokalne putanje su dozvoljene samo u skriptovima. String "$\bullet$ " će biti zamenjen stvarnom putanjom sa lokacije na kojoj je skript aktiviran.

$T ::=$	$\emptyset$	prazno drvo
	$\mid x$	promenljiva
	$\mid T \mid T$	kompozicija dva drveta
	$\mid \mathtt{a}[T]$	grana označena sa `a` sa poddrvetom T
	$\mid \mathtt{a}[\square\Pi]$	grana označena sa asa skriptom $\square\Pi$
	$\mid \mathtt{a}[p@\lambda]$	grana označena sa asa pokazivačem $p@\lambda$

Tabela 4.1: Sintaksa drveta

$p ::= \mathtt{a} \mid // \mid .. \mid \bullet \mid x \mid p \,/\, p$

Tabela 4.2: Sintaksa putanje

λ može biti promenljiva ili ime lokacije. *Pokazivač*, $p@\lambda$, pokazuje na skup čvorova na lokaciji λ, koji se nalaze na *putanji* p.

Procesi

Procesi koje ovde razmatramo su, u osnovi, procesi iz $d\pi$-računa [12], gde je lokalna komunikacija, modelirana procesima iz π-računa, proširena migracijom na lokacije (komanda `go`). Imamo još dva procesa za lokalnu interakciju procesa i drveta: `run` i `update` komande. Procese najčesće obeležavamo sa P, Q, R, a kanale sa c. Sintaksa procesa data je u tabeli 4.3.
Vrednost, v, može biti kanal, skript, lokacija, putanja ili drvo. Sintaksa vrednosti data je u tabeli 4.4.
Argument komande `go` može biti ime lokacije, promenljiva ili simbol $\circlearrowleft$, koji sme da se pojavi samo u skriptu (da označi lokaciju na kojoj će skript biti aktiviran).
Komanda `update` ima dva argumenta. Prvi je *šablon* χ, a drugi *izraz* V, čija je sintaksa data u tabeli 4.5. Šablon (traženi oblik podatka) može biti oblika skript, pokazivač ili drvo. Izrazi (podaci) mogu biti skript, pokazivač ili drvo. U $\mathtt{update}_p(\chi, V).P$, promenljive iz χ se mogu pojaviti i u V i u P i vezane su. Ovo je razlog zbog kojeg je dozvoljeno pojavljivanje promenljivih u skriptovima, pokazivačima, drvetima i procesima.

4.2 Semantika operacija

Relacijom redukcije u Xdπ oppisane su tri vrste intrakcije

- Međusobna komunikacija procesa
- Prelazak procesa sa jedne na drugu lokciju

$P ::=$	0	nil
	$\mid P \mid P$	paralelna compozicija
	$\mid (\nu c)P$	novi kanal c
	$\mid \bar{c}\langle v \rangle$	output vrednosti v kroz kanal c
	$\mid c(z).P$	input parametrizovan promenljivom z
	$\mid !c(x).P$	replikacija input procesa
	$\mid \texttt{go}\ \lambda.P$	migracija na lokaciju λ
	$\mid \texttt{go}\ \circlearrowleft.P$	migracija na izvornu lokaciju
	$\mid \texttt{run}_p$	run komanda
	$\mid \texttt{update}_p(\chi, V).P$	update komanda

Tabela 4.3: Sintaksa procesa

$$v ::= c \mid \Box P \mid l \mid p \mid T$$

Tabela 4.4: Sintaksa vrednosti

$$\begin{aligned} \chi ::= &\ \Box x \mid y@x \mid x \\ V ::= &\ \Box \Pi \mid p@\lambda \mid T \end{aligned}$$

Tabela 4.5: Šabloni i izrazi

(com) $l[\ T\ \|\ \bar{c}\langle v\rangle \mid c(z).P \mid Q\ ] \to l[\ T\ \|\ P\{v/z\} \mid Q\ ]$

(com!) $l[\ T\ \|\ \bar{c}\langle v\rangle \mid !c(z).P \mid Q\ ] \to l[\ T\ \|\ !c(z).P \mid P\{v/z\} \mid Q\ ]$

(stay) $l[\ T\ \|\ \texttt{go}\ l.P \mid Q\ ] \to l[\ T\ \|\ P \mid Q\ ]$

(go) $l[\ T_1 \mid\mid \texttt{go}\ m.P \mid Q\ ] \mid m[\ T_2\ \|\ R\ ] \to l[\ T_1\ \|\ Q\ ] \mid m[\ T_2\ \|\ P \mid R\ ]$

(run) $$\frac{p(T) \rightsquigarrow_{p,l,\Box x,\Box x} T, \{\{\Box P_1/\Box x\}, \ldots, \{\Box P_n/\Box x\}\}}{l[\ T\ \|\ \texttt{run}_p \mid Q\ ] \to l[\ T\ \|\ P_1 \mid \ldots \mid P_n \mid Q\ ]}$$

(update) $$\frac{p(T) \rightsquigarrow_{p,l,\chi,V} T', \{\texttt{s}_1, \ldots, \texttt{s}_n\}}{l[\ T \parallel \texttt{update}_p(\chi, V).P \mid Q\ ] \to l[\ T' \parallel P\texttt{s}_1 \mid \ldots \mid P\texttt{s}_n \mid Q\ ]}$$

Tabela 4.6: Pravila redukcije

- Interakcija procesa sa lokalnim podacima uz pomoć `run` i `update` komandi

4.2.1 Pravila redukcije

Pravila redukcije su data u tabeli 4.6. Međusobna komunikacija procesa data je pravilima (com) i (com!). Prelazak procesa na lokaciju različitu od polazne dat je pravilom (go), a na istu pravilom (stay). Interakcija procesa sa lokalnim podacima opisana je pravilima (run) i (update). Pravila redukcije (run) i (update) bazirana su na definiciji update funkcije, $\rightsquigarrow$, koja je data u tabeli u 4.7. Ova funkcija za parametre uzima putanju p, lokaciju l, šablon χ i vrednost V, dok njen argument može biti drvo kod kojeg neki čvorovi mogu biti podvučeni), skript ili pokazivač koji, takođe, mogu biti podvuceni. U je oznaka za termove obogacene podvlačenjem. Najinteresantija pravila su (Up) i (Id) u kojima se pomoću funkcije `match` vrać a supstitucija (traženi podaci) ili se nastavlja sa daljom proverom u slučaju da podvučeni podaci ne odgovaruju šablonu.

(Empty tree)	$\emptyset \rightsquigarrow_\theta \emptyset, \emptyset$
(Script)	$\Box P \rightsquigarrow_\theta \Box P, \emptyset$
(Pointer)	$p@ \rightsquigarrow_\theta p@l, \emptyset$
(Node)	$\dfrac{U \rightsquigarrow_\theta V, \Theta}{\mathtt{a}[U] \rightsquigarrow_\theta \mathtt{a}[V], \Theta}$
(Par)	$\dfrac{U_1 \rightsquigarrow_\theta T_1, \Theta_1 \quad U_2 \rightsquigarrow_\theta T_2, \Theta_2}{U_1 \mid U_2 \rightsquigarrow_\theta T_1 \mid T_2, \Theta_1 \cup \Theta_2}$
(Id)	$\dfrac{\mathtt{match} U \chi \text{ undefined} \quad U \rightsquigarrow_\theta V, \Theta}{\mathtt{a}[\underline{U}] \rightsquigarrow_\theta \mathtt{a}[V], \Theta}$
(Up)	$\dfrac{\mathtt{match} U \chi = \mathtt{s} \quad V \rightsquigarrow_\theta V', \Theta \quad \theta = p, l, \chi, V}{\mathtt{a}[\underline{U}] \rightsquigarrow_\theta \mathtt{a}[V'], \{\mathtt{s}\{l/\circlearrowleft, p/\cdot\}\} \cup \Theta}$

Tabela 4.7: $\rightsquigarrow$ funkcija

4.2.2 Primer

Kao ilustraciju pravila redukcije Xdπ računa dajemo seledeći primer. Neka se na lokaciji l nalaze drvo $T = \mathtt{a}\ [\ \mathtt{b}\ [\ \Box P\ ]\ |\ [\ \Box Q\ ]\ ]$ i proces $\mathtt{update}_{\mathtt{a/b}}(\Box x, \mathtt{c}[p@l]).\mathtt{go}\ m.\mathtt{update}_q(y, \mathtt{c}[\Box x])$. Tada primenom pravila (update) dolazi do redulcije:

$$l[\ T \parallel \mathtt{update}_{\mathtt{a/b}}(\Box x, \mathtt{c}[p@l]).\mathtt{go}\ m.\mathtt{update}_q(y, \mathtt{c}[\Box x])] \rightarrow$$

$$l[T' \parallel \mathtt{go}\ m.\mathtt{update}_q(y, \mathtt{c}[\Box P]) \mid \mathtt{go}\ m.\mathtt{update}_q(y, \mathtt{c}[\Box Q])$$

gde je $T' = \mathtt{a}\ [\ \mathtt{b}\ [\ \mathtt{c}\ [\ p@l\ ]\ ]\ |\ [\ \mathtt{c}\ [\ p@l\ ]\ ]\ ]$. Posle ove redukcije na lokaciji l se nalazi drvo T' koje je dobijeno od drveta T pomoću funkcije $\rightsquigarrow$. Odnosno svi podaci oblika skript na putanji `ab` u drvetu T su zamenjeni sa $\mathtt{c}\ [\ \Box P\ ]$. Proces $\mathtt{update}_{\mathtt{a/b}}(\Box x, \mathtt{c}[p@l]).\mathtt{go}\ m.\mathtt{update}_q(y, \mathtt{c}[\Box x])$ koji se prvobitno nalazio na lokaciji l se redukovao na $\mathtt{go}\ m.\mathtt{update}_q(y, \mathtt{c}[\Box P]) \mid \mathtt{go}\ m.\mathtt{update}_q(y, \mathtt{c}[\Box Q])$. Da bi se to postiglo primenjena je supstitucija dobijena funkcijom $\rightsquigarrow$.

U Xdπ račun uvedeni su tipovi [9] i dokazano je da ovakav tipiziran sistem ispunjava svojstva bezbednosti.

Glava 5

Zaključak

Možemo reći da su danas koncepcija računarskog sistema i mobilnost dobile značajnu ulogu i da još uvek treba raditi na adekvatnoj teoretskoj osnovi. Računaski sistemi i procesi u njima, raspoređeni po čitavom svetu razmenjuju podatke, pomeraju se sa jedne na drugu lokaciju, međsobno interaguju, te stoga globalno računarstvo postaje oblast sve interesantnija za istraživanje. Internet i WWW pružaju obezbeđuju infrastrukturu koja je rasprostranjena po čitavoj planeti. Međutim Web krši mnoge poznate pretpostavke o ponašanju distribuiranih sistema, te je stoga neophodno raditi na razvoju novih teoretskih modela.
U daljem radu planiramo da razmatramo sistem sa sledećim osobinama:

- procesi sa različitim ulogama
- uloga procesa određuje koji deo drveta taj proces može da vidi (view)
- ako neki proces želi da čita određene podatke, nijedan proces ne može da promeni te podatke pre čitanja
- proces više uloge može da promeni view niže uloge

i da za njega razvijemo odgovarajući model.

Bibliografija

[1] Abadi, M.,A.D. Gordon, A calculus for cryptographic protocols: the spi calculus. *Proc. of the Fourth ACM Conference on Computer and Communications Security*, 36-47, 1997.

[2] Boudol, G., Asynchrony and the p-calculus. *Technical Report 1702, INRIA, Sophia-Antipolis*, 1992

[3] Chiara Braghin, Daniele Gorla, Vladimiro Sassone: Role-based access control for a distributed calculus. Journal of Computer Security 14(2): 113-155 (2006)

[4] Adriana B. Compagnoni, Elsa L. Gunter: Types for Security in a Mobile World. TGC 2005: 75-97

[5] Cardelli, L., A.D. Gordon, Mobile ambients, *Foundations of Software Science and Computational Structures*, Maurice Nivat (Ed.), Lecture Notes in Computer Science 1378, Springer, 140-155. 1998.

[6] Luca Cardelli, Andrew D. Gordon: Types for Mobile Ambients. *POPL 1999*: 79-92

[7] Mario Coppo, Mariangiola Dezani-Ciancaglini, Elio Giovannetti, Ivano Salvo: Mobility Types for Mobile Processes in Mobile Ambients. *Electr. Notes Theor. Comput. Sci. 78*: (2003)

[8] Mads Dam: On the Decidability of Process Equivalences for the pi-Calculus. Theor. Comput. Sci. 183(2): 215-228 (1997)

[9] Dezani-Ciancaglini M., S. Ghilezan, J. Pantovic and D. Varacca: Security Types for Dynamic Web Data. *Theoret. Comput. Sci.*: 156–171 (2008)

[10] Gardner, P., Maffeis, S.:Modelling Dynamic Web Data, *Theoretical Computer Science*, vol 342;104–131, (2005).

[11] Daniele Gorla, Matthew Hennessy, Vladimiro Sassone: Security Policies as Membranes in Systems for Global Computing CoRR abs/cs/0506061: (2005)

[12] Hennessy M.,Riely J.:Resource Access Control in Systems of Mobile Agents,*Information and Computation*, vol 173;82–120, (2002)

[13] Milner, R.: Communicating and Mobile Systems: the π-Calculus. Cambridge University Press, Cambridge (1999).

[14] Milner, R., J. Parrow, D. Walker, A calculus of mobile processes, Parts 1-2. *Information and Computation*, 100(1), 1-77. 1992

[15] Parrow, J.: An introduction to the π-Calculus. *Handbook of Process Algebra*, Elsevier (2001).

[16] Silvano dal Zilio, Mobile Processes: A Commented Bibliography (2002)