

Vážení prítomní, volám sa Michal Kováč a chcel by som vám prezentovať výsledky mojej dizertačnej práce s názvom Biologicky motivované výpočtové modely.

V úvode prezentácie vám predstavím rôzne výpočtové modely motivované biológiou. Najviac sme sa venovali P systémom, preto budem pokračovať formálnou definíciou a prehľadom rôznych variantov P systémov.

V druhej časti predstavím 4 témy nášho výskumu, z čoho 3 články boli publikované. V našej práci sme skúmali viaceré varianty P systémov a to konkrétne Sekvenčné P systémy s inhibítormi, Sekvenčné P systémy s aktívnymi membránami, Sekvenčné P systémy s množinami namiesto multimnožín, z čoho všetky spomenuté témy boli publikované. Dalším variantom P systémov, ktorým sme sa zaoberali bola Detekcia prázdnoty membrán.

## Biologicky motivované výpočtové modely

### └ Prehľad problematiky

### └ Biologicky motivované modely

### └ Biologicky motivované výpočtové modely

Dvojaké uplatnenie:

- reálne modely živých systémov
  - virtuálne biologické experimenty
  - verifikácia správnosti chápania ich činnosti
- modely na popis iných systémov

Biologicky motivované výpočtové modely majú dvojaké uplatnenie. Jednak v rámci biológie môžu slúžiť ako reálne modely správania sa živých systémov, na ktorých môžeme robiť rôzne virtuálne biologické experimenty, prípadne verifikovať správnosť nášho chápania ich biologickej činnosti.

Na druhej strane môžu slúžiť ako modely na popis aj iných ako biologických systémov, čo otvára rad teoretických informatických otázok, napr. výpočtová sila alebo analýza behaviorálnych vlastností.

## Biologicky motivované výpočtové modely

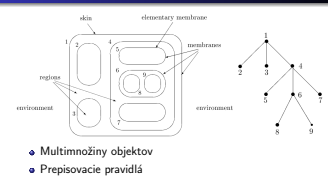
### └ Prehľad problematiky

### └ Biologicky motivované modely

### └ Biologicky motivované výpočtové modely

- Neurónové siete (od 1943)
- Celulárne automaty (od 1968)
- Evolučné algoritmy (od 1954)
- L systémy (od 1968)
- Swarm Intelligence (od 1989)
- P systémy (od 1998) [?]
- ...

Dlho skúmané modely ako neurónové siete, celulárne automaty, evolučné algoritmy, L systémy, či swarm intelligence, si už našli svoje uplatnenie v praxi, kým membránové systémy sú ešte len v začiatkoch svojho vývoja.



Membránové systémy sú inšpirované bunkami. Základom je preto membránová štruktúra, ktorá pozostáva z regiónov, ktoré sú oddelené membránami. Tvorí to hierarchickú štruktúru, ktorá sa dá zobrazit' aj ako strom.

Obsahom regionov sú multimnožiny objektov, ktoré v realite predstavujú napr. molekuly, vírusy, enzýmy alebo proteíny.

Objekty medzi sebou môžu interagovať. Táto interakcia je definovaná prepisovacími pravidlami.

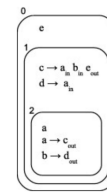
- $u \rightarrow v$ , where
- $u \in \mathbb{N}^E$
  - $v = v'$  or  $v = v'\delta$ , where  $\delta \notin \Sigma$
  - $v' \in \mathbb{N}^{\Sigma \times (\{here, out\} \cup \{a_i \mid 1 \leq i \leq m\})}$

Prepisovacie pravidlá majú ľavú a pravú stranu. Na ľavej strane sú reaktanty, čo je multimnožina objektov.

Na pravej strane sú produkty, čo je multimnožina objektov, pričom pre každý objekt sa definuje, či ostáva v aktuálnom regione, alebo ide cez membránu do vonkajšieho regionu alebo cez membránu s daný označením do vnútorného regionu.

Delta je špeciálny symbol, ktorý nepatrí abecede, ktorý keď je prítomný, tak po aplikovaní pravidla sa rozpustí membrána, v ktorej sa pravidlo aplikovalo a obsah membrány sa vyleje von.

Pravidlo je aplikovateľné v danom regione, ak sú reaktanty obsiahnuté v multimnožine objektov, ktorá sa aktuálne nachádza v danom regione.



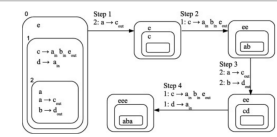
Toto je ukážka P systému, ktorý generuje Fibonacciho postupnosť. Má tri membrány a pracuje s piatimi objektami a,b,c,d,e. V membráne s označením 1 sú dve pravidlá. Prvé pravidlo prepíše objekt c na objekty a,b,e, a pošle objekty a,b do vnútornej membrány a objekt e do vonkajšej membrány. Druhé pravidlo prepíše objekt d na objekt a a pošle ho do vnútornej membrány. Vo vnútornej membráne sú dve pravidlá. Prvé prepíše a na c a pošle ho von. Druhé pravidlo prepíše b na d a pošle ho von.

- Krok výpočtu
  - Sekvenčný
  - Paralelný
  - Maximálne paralelný

Postupné uplatňovanie pravidiel definuje výpočet. V jednom kroku výpočtu sa uplatní:

- presne jedno pravidlo (sekv. mod)
- aspoň jedno pravidlo (paralelný mod)
- maximálna multimnožina pravidiel

V pôvodnej definícii, ktorú uvádza Paun, sa používa maximálny paralelizmus.



V úvodnej konfigurácii ukážkového P systému je aplikovateľné iba prvé pravidlo vo vnútornej membráne. A sa prepíše na c a pošle von. V ďalšom kroku sa v prepíše na abe, e sa pošle von, ab dnu. V ďalšom kroku sa vďaka maximálnemu paralelizmu musia aplikovať vo vnútornej membráne obidve pravidlá, preto sa objekty ab prepíšu na cd a pošlú von. V ďalšom kroku sa vďaka maximálnemu paralelizmu musia aplikovať obidve pravidlá, preto pôjdu dve áčka a jedno béčko do vnútornej membrány a jedno éčko von.

Tento výpočet nikdy neskončí a každý druhý krok sa do vonkajšej membrány pošle počet éčok zodpovedajúci ďalšiemu prvku Fibonacciho postupnosti.

- Jazyk nad postupnosťami/multimnožinami objektov
  - Generatívny mód: postupnosť/multimnožina objektov vypustených do okolitého prostredia
  - Akceptačný mód: vstupnú multimnožinu vložíme do špecifickej membrány, ak výpočet zastaví, akceptujeme

P systém definuje jazyk rôznymi spôsobmi. Môže to byť jazyk nad slovami - postupnosťami objektov alebo jazyk nad multimnožinami. V generatívnom mode môžeme zobrať objekty vypustené do prostredia počas výpočtu a túto postupnosť objektov alebo multimnožinu objektov zahrnúť do jazyka. Keďže pre daný P systém vďaka nedeterminizmu existuje viac možných výpočtov, veľkosť definovaného jazyka môže byť aj väčšia ako 1.

- Jazyk nad posupnosťami/multimnožinami objektov
- Generatívny mód: postupnosť/multimnožina objektov vypustených do okolitého prostredia
- Akceptačný mód: vstupnú multimnožinu vložíme do špecifickej membrány, ak výpočet zastaví, akceptujeme

V akceptačnom mode vstupnú multimnožinu vložíme do špecifickej membrány a spustíme výpočet, ktorý ak zastaví, tak vstupnú multimnožinu zahrnieme do jazyka.

Pre väčšinu známych modelov sú generatívny aj akceptačný mod rovnako silné, u P systémoch to nie je vždy tak, preto sa oplatí skúmať obidva mody.

- $u \rightarrow v$
- Kooperatívne ( $u \in \mathbb{N}^E$ ) (PsRE [?])
  - Nekooperatívne ( $u \in \Sigma$ ) (PsCF [?])
  - Nekooperatívne s inhibítormi ( $u \rightarrow v \mid \text{inh}, \text{inh} \subseteq \Sigma$ ) (PsETOL [?])
  - Katalytické ( $cu \rightarrow cv, u \in \Sigma, c \in C \subseteq \Sigma$ )
    - $\leq 2$  katalyzátormi (PsRE [?])
    - $\leq 1$  katalyzátorom (otvorený problém)
    - $\leq 1$  katalyzátorom a inhibítormi (PsRE [?])

Literatúra spomína rôzne spôsoby definovania prepisovacieho pravidla. Pôvodná definícia, ktorú uvádza Paun, používa kooperatívne pravidlá v znení, ako som uviedol. Takto definované P systémy sú Turingovsky úplné.

Nekooperatívne pravidlá neumožňujú interakciu medzi objektami, takže na ľavej strane je vždy iba jeden objekt. Takto definované P systémy sú ekvivalentné Parikhovmu zobrazeniu bezkontextových jazykov.

Pravidlá s inhibítormi umožňujú špecifikovať množinu objektov, inhibítorov, z ktorých ak aspoň jeden je prítomný v regióne, tak dané pravidlo sa nemôže uplatniť. Takto definované P systémy sú ekvivalentné Parikhovmu zobrazeniu triedy jazykov ETOL.

- Kooperatívne ( $u \in \mathbb{N}^E$ ) (PsRE [?])
- Nekooperatívne ( $u \in \Sigma$ ) (PsCF [?])
- Nekooperatívne s inhibítormi ( $u \rightarrow v \mid \text{inh}, \text{inh} \subseteq \Sigma$ ) (PsETOL [?])
- Katalytické ( $cu \rightarrow cv, u \in \Sigma, c \in C \subseteq \Sigma$ )
  - s 2 katalyzátormi (PsRE [?])
  - s 1 katalyzátorom (otvorený problém)
  - s 1 katalyzátorom a inhibítormi (PsRE [?])

Katalytické pravidlá umožňujú objektom interagovať iba s objektom z množiny katalyzátorov. Dva katalyzátory stačia na Turingovskú úplnosť. Výpočtovú silu P systémov s jedným katalyzátorom nevieme zaradiť, je to otvorený problém. Ak ale umožníme pravidlá s inhibítormi, dosiahneme Turingovskú úplnosť.

- Maximálny paralelizmus vs. sekvenčný mód
- Sekvenčné P systémy s kooperatívnymi pravidlami (VASS [?])
  - s prioritami (PsRE [?])
  - s aktívnymi membránami (PsRE [?])
  - s inhibítormi (PsRE [?])

Maximálny paralelizmus je veľmi silná vlastnosť. Globálny časovač reakcií vo väčšine prípadov tvorí hranicu toho, čo je, a čo nie je Turingovsky úplné. Ani v bunke sa nenachádza taký časovač, podľa ktorého by sa reakcie synchronizovali. Preto sa hľadajú spôsoby, ako túto vlastnosť odľahčiť, prípadne, akými spôsobmi by sa dal rozšíriť sekvenčný mod, aby sa dosiahla Turingovská úplnosť.

---

Sekvenčné P systémy s kooperatívnymi pravidlami nie sú Turingovsky úplné, lebo sú ekvivalentné s Vector Addition Systems a s Petriho sieťami.

---

Ak sa pridajú pravidlá s prioritami, alebo s aktívnymi membránami, alebo s inhibítormi, takto definované P systémy sú už Turingovsky úplné.


2018-01-13

## Biologicky motivované výpočtové modely

- └ Skúmané varianty P systémov
- └ Sekvenčné P systémy s inhibítormi
- └ Vlastné výsledky

Vlastné výsledky

Vlastné výsledky

Teraz nasleduje druhá časť prezentácie, v ktorej predstavím vlastné výsledky. 

2018-01-13

## Biologicky motivované výpočtové modely

- └ Skúmané varianty P systémov
- └ Sekvenčné P systémy s inhibítormi
- └ Sekvenčné P systémy s inhibítormi

Sekvenčné P systémy s inhibítormi

1. Sekvenčné P systémy s inhibítormi

Prvý variant P systémov, ktorým sme sa zaoberali, sú sekvenčné P systémy s inhibítormi.



- └ Skúmané varianty P systémov
- └ Sekvenčné P systémy s inhibítormi
- └ Sekvenčné P systémy s inhibítormi

- Pravidlo s inhibítormi:  $u \rightarrow v|I, I \subseteq \Sigma$
- Turingovská úplnosť pre akceptačný aj generatívny mód
- 

Toto rozšírenie umožňuje k pravidlám pridať množinu inhibítorov  $I$ , z ktorých ak sa aspoň jeden nachádza v aktuálnej membráne, pravidlo nie je aplikovateľné.

Dokázali sme Turingovskú úplnosť pre akceptačný aj generatívny mód.

Tieto výsledky sme prezentovali na konferencii Computability in Europe v roku 2014 v Budapešti a náš článok je publikovaný v zborníku.

- └ Skúmané varianty P systémov
- └ Sekvenčné P systémy s inhibítormi
- └ Prehľad simulácie pre akceptačný mód

- Simulácia registrového stroja
- Obsah registra  $x$  sa reprezentuje početnosťou objektu  $x$
- Objekt pre každú inštrukciu
- SUB inštrukcia sa simuluje pomocou inhibítora
  - $i : SUB(x, j, k)$
  - $ix \rightarrow j$
  - $i \rightarrow k|_{-x}$

Pre akceptačný mód sme ukázali simuláciu registrového stroja. Obsah registra  $x$  sa reprezentuje početnosťou objektu  $x$ . Navyše máme objekt pre každú inštrukciu. Inhibítory sme využili na simuláciu inštrukcie SUB.

- └ Skúmané varianty P systémov
- └ Sekvenčné P systémy s inhibítormi
- └ Prehľad simulácie pre akceptačný mód

- Registrový stroj  $M = (n, P, i, h, Lab)$
- P systém  $(\Sigma, \mu, w, R)$ 
  - $\Sigma = Lab \cup \# \cup a_j, 1 \leq j \leq n$
  - $w = i \cup a_j^n, n_i$  je počiatočná hodnota registra  $i$
  - $\forall (e : add(j), k, l) \in P :$ 
    - $e \rightarrow a_j k \in R$
    - $e \rightarrow a_j l \in R$
  - $\forall (e : sub(j), k, l) \in P :$ 
    - $ea_j \rightarrow k \in R$
    - $e \rightarrow l, a_j \in R$
  - $ha_j \rightarrow h\# \in R$
  - $\# \rightarrow \# \in R$

Formálne, máme registrový stroj  $M$ , kde  $n$  je počet registrov,  $P$  je množina inštrukcií označených značkami z množiny  $Lab$ ,  $i$  je označenie počiatočnej inštrukcie,  $h$  je označenie koncovej inštrukcie.

Zostrojíme P systém s jednou membránou, v abecede budú označenia inštrukcií, symboly zodpovedajúce jednotlivým registrom a špeciálny ukončovací symbol mriežka.

V počiatočnej konfigurácii membrána obsahuje označenie počiatočnej inštrukcie registrového stroja a objekty  $a_j$  podľa počiatočného stavu registrov.

- └ Skúmané varianty P systémov
- └ Sekvenčné P systémy s inhibítormi
- └ Prehľad simulácie pre akceptačný mód

- Registrový stroj  $M = (n, P, i, h, Lab)$
- P systém  $(\Sigma, \mu, w, R)$ 
  - $\Sigma = Lab \cup \# \cup a_j, 1 \leq j \leq n$
  - $w = i \cup a_j^n, n_i$  je počiatočná hodnota registra  $i$
  - $\forall (e : add(j), k, l) \in P :$ 
    - $e \rightarrow a_j k \in R$
    - $e \rightarrow a_j l \in R$
  - $\forall (e : sub(j), k, l) \in P :$ 
    - $ea_j \rightarrow k \in R$
    - $e \rightarrow l, a_j \in R$
  - $ha_j \rightarrow h\# \in R$
  - $\# \rightarrow \# \in R$

Pre všetky inštrukcie ADD, ktoré zvýšia hodnotu registra  $j$  o jedna a nasledujúca inštrukcia je  $k$  alebo  $l$  (rozhoduje sa nedeterministicky) budeme mať pravidlá, ktoré prepíše symbol zodpovedajúci označeniu inštrukcie na symbol  $a_j$  a symbol zodpovedajúci označeniu nasledujúcej inštrukcie.

Uvažujme inštrukciu SUB, ktorá ak má register  $j$  kladnú hodnotu, tak ju zníži o jedna, a ak je register  $j$  prázdny, tak ju nezníži, ale bude iná nasledujúca inštrukcia. Pre takáto inštrukciu budeme mať v P systéme dve pravidlá. Prvé pravidlo je kooperatívne, ktoré skonzumuje symbol  $a_j$  a vytvorí symbol  $k$  zodpovedajúci označeniu nasledujúcej inštrukcie. Druhé pravidlo je s inhibítormi a aplikovať sa môže len, ak sa v regione nenachádza symbol  $a_j$ .

- Registrový stroj  $M = (n, P, i, h, Lab)$
- P systém  $(\Sigma, \mu, w, R)$ 
  - $\Sigma = Lab \cup \# \cup a_j, 1 \leq j \leq n$
  - $w = i \cup a_j^n, n_j$  je počiatočná hodnota registra  $i$
  - $\forall (e : add(j), k, l) \in P :$ 
    - $e \rightarrow a_j k \in R$
    - $e \rightarrow a_j l \in R$
  - $\forall (e : sub(j), k, l) \in P :$ 
    - $ea_j \rightarrow k \in R$
    - $e \rightarrow l, a_j \in R$
  - $ha_j \rightarrow h \# \in R$
  - $\# \rightarrow \# \in R$

Ked uvažujeme registrové stroje, kde musia byť všetky registre prázdne, aby sa výpočet korektne ukončil, máme pravidlá, ktoré pri dosiahnutí koncovej inštrukcie  $h$ , ak je niektorý register neprázdny, vytvorí sa špeciálny symbol, ktorý tam už ostane navždy, a vďaka poslednému pravidlu sa dosiahne, že výpočet nezastaví.

Dokázali sme, že pre každý registrový stroj vieme zostrojiť sekvenčný P systém s inhibítormi, ktorý v akceptačnom móde zastaví na rovnakých vstupoch ako registrový stroj.

- Simulácia maximálne paralelného P systému  $\Pi_1$  pomocou sekvenčného P systému s inhibítormi  $\Pi_2$
- Každý maximálne paralelný krok  $\Pi_1$  simulujeme sekvenčnými krokmi  $\Pi_2$
- Maximálne paralelný krok rozdeľujeme na 4 fázy:
  - RUN
  - SYNCHRONIZE
  - SENDDOWN
  - RESTORE

V generatívnom móde sme sa rozhodli pre simuláciu maximálne paralelného P systému  $\Pi_1$  pomocou sekvenčného P systému s inhibítormi  $\Pi_2$ .

---

Každý maximálne paralelný krok  $\Pi_1$  simulujeme pomocou niekoľkých sekvenčných krokov  $\Pi_2$ .

## Biologicky motivované výpočtové modely

- └ Skúmané varianty P systémov
  - └ Sekvenčné P systémy s inhibítormi
    - └ Prehľad simulácie pre generatívny mód

### Prehľad simulácie pre generatívny mód

- Simulácia maximálne paralelného P systému  $\Pi_1$  pomocou sekvenčného P systému s inhibítormi  $\Pi_2$ .
- Každý maximálne paralelný krok  $\Pi_1$  simulujeme sekvenčnými krokmi  $\Pi_2$ .
- Maximálne paralelný krok rozdeľujeme na 4 fázy:
  - RUN
  - SYNCHRONIZE
  - SENDDOWN
  - RESTORE

Kedže pravidlá sa uplatňujú simultánne vo všetkých membránach, tento proces treba synchronizovať. Maximálne paralelný krok rozdeľujeme na 4 fázy. Fáza je reprezentovaná špeciálnym objektom a všetky pravidlá sú kooperatívne, na ľavej strane každého pravidla je fáza, ku ktorej sa pravidlo viaže.

## Biologicky motivované výpočtové modely

- └ Skúmané varianty P systémov
  - └ Sekvenčné P systémy s inhibítormi
    - └ Prehľad simulácie pre generatívny mód

### Prehľad simulácie pre generatívny mód

- Simulácia maximálne paralelného P systému  $\Pi_1$  pomocou sekvenčného P systému s inhibítormi  $\Pi_2$ .
- Každý maximálne paralelný krok  $\Pi_1$  simulujeme sekvenčnými krokmi  $\Pi_2$ .
- Maximálne paralelný krok rozdeľujeme na 4 fázy:
  - RUN
  - SYNCHRONIZE
  - SENDDOWN
  - RESTORE

V prvej fáze RUN v  $\Pi_2$  po jednom prepisujeme symboly pomocou pravidiel zodpovedajúcim pravidlám v  $\Pi_1$ , akurát produkty si označujeme, aby neboli znovu použité, kým neskončí simulácia jedného maximálne paralelného kroku. Pomocou inhibítorov zistíme moment, kedy sa už v  $\Pi_2$  nedá aplikovať žiadne ďalšie pravidlo, ktoré by sa mohlo zahrnúť do multimnožiny pravidiel aplikovaných v  $\Pi_1$ . To nám zaručí, že aplikované pravidlá sú maximálnou multimnožinou a môžeme prejsť do simulácie ďalšieho maximálne paralelného kroku.

- Simulácia maximálne paralelného P systému  $\Pi_1$  pomocou sekvenčného P systému s inhibítormi  $\Pi_2$
- Každý maximálne paralelný krok  $\Pi_1$  simulujeme sekvenčnými krokmi  $\Pi_2$
- Maximálne paralelný krok rozdeľujeme na 4 fázy:
  - RUN
  - SYNCHRONIZE
  - SENDDOWN
  - RESTORE

V druhej fáze SYNCHRONIZE sa v každom regione čaká na ostatné regiony, aby sa spustil ďalší maximálne paralelný krok.

Pošle sa synchronizačný token do vonkajšej membrány. V nej, keď sa pozbierajú tokeny zo všetkých membrán, tak vonkajšia membrána pošle signál všetkým membránam, aby mohli začať ďalší maximálne paralelný krok.

Dalšie dve fázy sú čisto technického charakteru. Vo fáze SENDDOWN sa zvlášť posielajú objekty do vnútorných membrán a vo fáze RESTORE sa všetko v rámci regionu pripravuje na ďalší maximálne paralelný krok.

- Sekvenčné P systémy s inhibítormi sú Turingovsky úplné
- Podobné výsledky pre Petriho siete
- Rozpúšťanie, vytváranie membrán, pravidiel s prioritami
- Výskum iných obmedzení pravidiel

Ukázali sme, že v akceptačnom aj v generatívnom móde sú sekvenčné P systémy s inhibítormi Turingovsky úplné.

Hoci tieto výsledky nie sú veľmi prekvapivé, nakoľko podobné výsledky s inhibítormi už boli ukázané pre Petriho siete, prínos týchto simulácií je aj v ukázaní spôsobu konverzie medzi rôznymi modelmi, čo môže pomôcť v ďalšom výskume.


Další výskum môže nadviazať a doplniť simuláciu o iné aspekty P systémov, napríklad rozpúšťanie, vytváranie membrán, pravidiel s prioritami, ako aj skúsiť iné obmedzenie pravidiel, napríklad obmedzenie kooperácie alebo obmedzenie sily inhibítorov.

2018-01-13

## Biologicky motivované výpočtové modely

- └ Skúmané varianty P systémov
  - └ Sekvenčné P systémy s aktívnymi membránami
    - └ Sekvenčné P systémy s aktívnymi membránami

## 2. Sekvenčné P systémy s aktívnymi membránami

Druhý variant P systémov, ktorým sme sa zaoberali, sú sekvenčné P systémy s aktívnymi membránami. 

## Sekvenčné P systémy s aktívnymi membránami

2018-01-13

## Biologicky motivované výpočtové modely

- └ Skúmané varianty P systémov
  - └ Sekvenčné P systémy s aktívnymi membránami
    - └ Sekvenčné P systémy s aktívnymi membránami

- Pravidlo, ktoré vytvorí membránu:  $u \rightarrow [v]_j$ ,  $u \in N^*$ ,  $v \in N^*$ ,  $1 \leq j \leq m$
- Bez limitu počtu aplikovaní pravidla na vytvorenie membrány (PsRE [?])
- Rozhodnuteľnosť existencie nekonečného výpočtu
- Nerozhodnuteľnosť existencie konečného výpočtu
- 

Pravidlo, ktoré vytvorí membránu, na pravej strane špecifikuje označenie membrány a multimnožinu objektov, ktoré sa v nej budú po vytvorení nachádzať.

---

Ak pre sekvenčné P systémy povolíme pravidlá vytvárajúce nové membrány, a nestanovíme limit na počet aplikovaní takýchto pravidiel, dosiahneme Turingovskú úplnosť, ako ukázal Ibarra v roku 2005. Ukázal aj, že pri obmedzení počtu aplikovaní takýchto pravidiel je to dá simulovať variantom bez takýchto pravidiel.

- Pravidlo, ktoré vytvorí membránu:  $u \rightarrow [v]_i$ ,  $u \in N^*$ ,  $v \in N^*$ ,  $1 \leq i \leq m$
- Bez limitu počtu aplikovaní pravidiel na vytvorenie membrány (PsRE [?])
- Rozhodnuteľnosť existencie nekonečného výpočtu
- Nerozhodnuteľnosť existencie konečného výpočtu

Pre tento variant sme analyzovali rozhodnuteľnosť behaviorálnych vlastností. Podarilo sa nám dokázať, že existencia nekonečného výpočtu je rozhodnuteľný problém a existencia konečného výpočtu je nerozhodnuteľný problém.

Tieto výsledky sme prezentovali na konferencii Computability in Europe v roku 2015 v Bukurešti a článok bol publikovaný v zborníku z tejto konferencie.

- Problém zastavenia je definovaný pre deterministické modely
- Zovšeobecnenie: Existencia (ne)konečného výpočtu

Spomenuté behaviorálne vlastnosti sa podobajú na problém zastavenia. Ten je ale definovaný iba pre deterministické modely.

Kedže pre netederministické modely môže existovať výpočet, ktorý zastaví, aj výpočet, ktorý nezastaví, má zmysel pýtať sa dve rôzne otázky: či existuje konečný výpočet a či existuje nekonečný výpočet.

- Membránová konfigurácia  $(T, l, c)$ , kde
  - $T$  je stromová štruktúra
  - $l : V(T) \rightarrow \{1, \dots, m\}$
  - $c : V(T) \rightarrow \mathbb{N}^{\Sigma}$
- Aktívny P systém je  $(\Sigma, C_0, R_1, R_2, \dots, R_m)$ , kde
  - $\Sigma$  je abeceda
  - $C_0$  je počiatočná membránová konfigurácia
  - $R_i$  je množina pravidiel

Aby sa pri dôkazoch lepšie manipulovalo s konfiguráciou, upravili sme definíciu aktívneho P systému, kde sme izolovali pojem membránová konfigurácia.

Je to trojica  $(T, l, c)$ , kde  $T$  je stromová štruktúra

---

$l$  je označenie membrán - zobrazenie vrcholov na čísla

---

$c$  je zobrazenie vrcholov stromu  $T$  na multimnožinu symbolov, čo predstavuje obsah membrány.

---

Aktívny P systém je  $m+2$  tica, kde  $\Sigma$  je abeceda,  $C_0$  je počiatočná membránová konfigurácia a  $R_i$  je množina pravidiel asociovaná s označením membrány  $i$ .

- Existencia konečného výpočtu je nerozhodnuteľný problém
- Redukcia na halting problem

Podarilo sa nám dokázať, že existencia konečného výpočtu pre sekvenčné P systémy s aktívnymi membránami je nerozhodnuteľný problém.

---

Dôkaz je pomocou redukcie. Ibarra v článku uvádza simuláciu, vďaka ktorej môžeme tvrdiť, že ak by sme vedeli rozhodovať existenciu konečného výpočtu pre sekvenčné P systémy s aktívnymi membránami, potom by sme vedeli rozhodovať existenciu konečného výpočtu pre registrové stroje, čo je už známy nerozhodnuteľný problém.



- └ Skúmané varianty P systémov
- └ Sekvenčné P systémy s aktívnymi membránami
- └ Existencia nekonečného výpočtu

- Existencia nekonečného výpočtu je rozhodnuteľný problém
- Obmedzenie na počet membrán

Skúmali sme aj opačný problém - existenciu nekonečného výpočtu. Podarilo sa nám dokázať, že je to rozhodnuteľný problém.

Dôkaz uvádzame iba pre obmedzenie na počet membrán, ktoré sa nachádzajú v ľubovoľnej konfigurácii.

Aj keď toto obmedzenie nie je veľmi realistické z biologického hľadiska, výsledok je aj tak zaujímavý, lebo sekvenčné P systémy s aktívnymi membránami sú Turingovsky úplné aj s týmto obmedzením - pri simulácii registrového stroja sa v každej konfigurácii nachádzajú najviac tri membrány.

- └ Skúmané varianty P systémov
- └ Sekvenčné P systémy s aktívnymi membránami
- └ Existencia nekonečného výpočtu

- Graf dosiahnuteľnosti
- Čiastočné usporiadanie  $\leq$ :
  - $C_1 = (T_1, h, c_1)$
  - $C_2 = (T_2, h, c_2)$
  - $C_1 \leq C_2$ , ak  $\exists$  izomorfizmus  $f: T_1 \rightarrow T_2$  taký, že:
    - $\forall d \in T_1$  platí:
      - $h(d) = h(f(d))$
      - $c_1(d) \leq c_2(f(d))$
  - $C_1 \leq C_2 \Rightarrow$  každé pravidlo v  $C_1$  je aplikovateľné v  $C_2$ .

Dôkaz využíva graf dosiahnuteľnosti. V prípade jednej membrány konfigurácia obsahuje iba multimnožinu objektov, preto sa dá použiť štandardná konštrukcia grafu dosiahnuteľnosti pre Petriho siete.

Vo všeobecnosti potrebujeme ale rozšíriť definíciu čiastočného usporiadania konfigurácií.

- Graf dosiahnuteľnosti
- Čiastočné usporiadanie  $\leq$ :
  - $C_1 = (T_1, h, c_1)$
  - $C_2 = (T_2, h, c_2)$
  - $C_1 \leq C_2$ , ak  $\exists$  izomorfizmus  $f: T_1 \rightarrow T_2$  taký, že:
    - $\forall d \in T_1$  platí:
      - $h(d) = h(f(d))$
      - $c_1(d) \subseteq c_2(f(d))$
  - $C_1 \leq C_2 \Rightarrow$  každé pravidlo v  $C_1$  je aplikovateľné v  $C_2$ .

Majme konfigurácie  $C_1$  a  $C_2$ .

$C_2$  pokrýva  $C_1$  (zapisujeme  $C_1$  je menšia, alebo rovná ako  $C_2$ ), ak existuje izomorfizmus  $f$ , ktorý pre každú membránu zachováva označenia a zachováva obsah, tým, že obsah každej membrány v  $C_1$  je multimnožina obsiahnutá v obsahu zodpovedajúcej membrány v  $C_2$ .

Táto definícia nám umožňuje tvrdiť, že ak  $C_2$  pokrýva  $C_1$ , potom každé pravidlo aplikovateľné v  $C_1$  je aplikovateľné v  $C_2$ .

- Dicksonova lemma: Pre každú nekonečnú postupnosť  $n$ -tíc nad  $\mathbb{N}$   $\{a_i\}_{i \in \mathbb{N}}$  existujú  $i < j$ :  $a_i \leq a_j$
- Pre každú nekonečnú postupnosť konfigurácií existuje  $C_1, C_2$ :  $C_1 \rightarrow^* C_2$  a  $C_1 \leq C_2$ .
- Kodovanie konfigurácií  $enc(C_1) \leq enc(C_2) \Rightarrow C_1 \leq C_2$

Na tomto mieste by som chcel spomenúť Dicksonovu lemu. Tá tvrdí, že pre každú nekonečnú postupnosť  $n$ -tíc nad prirodzenými číslami existujú indexy  $i < j$ :  $a_i \leq a_j$ , pričom sa porovnávajú jednotlivé pozície v  $n$ -tici.

Dokážeme nasledovné tvrdenie: Pre každú nekonečnú postupnosť konfigurácií existuje  $C_1, C_2$ :  $C_1 \rightarrow^* C_2$  a  $C_1 \leq C_2$ .

Definovali sme kodovanie konfigurácií do  $n$ -tíc s vlastnosťou, že ak  $enc(C_1) \leq enc(C_2)$ , potom  $C_1 \leq C_2$ . Vďaka tomuto kodovaniu a pomocou Dicksonovej lemy dokážeme aj pôvodnú vetu, že pre každú nekonečnú postupnosť konfigurácií existuje  $C_1, C_2$ :  $C_1 \rightarrow^* C_2$  a  $C_1 \leq C_2$ .

- Traverzuj graf dosiahnuteľnosti
- Dosiahnutá konfigurácia  $C_2$ , taká, že na ceste z počiatočnej konfigurácie existuje  $C_1 \leq C_2 \Rightarrow \text{YES}$ .
- Ak traverzovanie skončilo  $\Rightarrow \text{NO}$ .

Algoritmus, ktorý rozhoduje existenciu nekonečného výpočtu je teda nasledovný:

Traverzuj graf dosiahnuteľnosti.


Ak sa dosiahne konfigurácia  $C_2$ , taká, že na ceste z počiatočnej konfigurácie existuje  $C_1$  taká, že  $C_2$  pokrýva  $C_1$ , tak nekonečný výpočet existuje. Ak traverzovanie skončilo, tak nekonečný výpočet neexistuje.

- Existencia nekonečného výpočtu je rozhodnuteľná.
- Existencia konečného výpočtu je nerozhodnuteľná.

Dokazali sme, že existencia nekonečného výpočtu sa pri sekvenčných P systémov s aktívnymi membránami s obmedzením na počet membrán dá rozhodovať. Otvorený problém ostáva, či to platí pre variant bez tohto obmedzenia. Veríme, že áno, ale nemáme k tomu dôkaz.

---

Tiež sme dokázali, že existencia konečného výpočtu je nerozhodnuteľná. Dokázali sme to pomocou redukcie na problém zastavenia.

Tretí variant P systémov, ktorým sme sa zaoberali, sú sekvenčné P systémy s množinami namiesto multimnožín. 

- Inšpirácia z Reaction systems
- Nakoľko realistické je reprezentovať presný počet objektov?
- Nepraktická analýza kvôli veľkosti stavového priestoru

Inšpirácia pre tento variant pramenila z formalizmu Reaction systems. Nahradili sme obsah membrány, kde namiesto multimnožín uvažujeme množiny objektov.

---

K tomuto rozhodnutiu nás viedli dve otázky.

Nakoľko realistické je reprezentovať presný počet objektov?

Niekedy nás zaujíma iba výskyt, napríklad či sa v membráne nachádza vírus, alebo nie.

---

Ak uvažujeme multimnožiny, máme problém explozie stavového priestoru, ktorý sa potom neprakticky analyzuje. V prípade množín je stavový priestor menší, čo umožňuje jednoduchšiu analýzu.

- [7]: počty objektov sa ignorujú
  - Maximálny paralelizmus  $\rightarrow$  determinizmus.
  - Ekvivalencia s konečnosťovými automatmi.
  - S aktívnymi membránami je model univerzálny.
- [7]: "min-enabled" computational step (= sekvenčný mód)
  - Ekvivalencia s konečnosťovými automatmi.
- Vlastnosti:
  - Pravidlá bez konfliktu (objekty sa môžu zúčastniť ako reaktanty súčasne vo viacerých pravidlách).
  - Ak je objekt použitý aspoň v jednom pravidle ako reaktant, bude spotrebovaný.

Alhazov v roku 2005 uvažoval o P systémoch, kde sa ignorovali počty objektov. Pri maximálnom paralelizme je výpočet deterministický, lebo pravidlá nie sú navzájom konfliktné a maximálna multimnožina simultánne aplikovateľných pravidiel je v každej konfigurácii iba jedna.

Takto definované P systémy sú ekvivalentné s konečnosťovými automatmi, čo sa týka výpočtovej sily.

S aktívnymi membránami je model univerzálny.

- [7]: počty objektov sa ignorujú
  - Maximálny paralelizmus  $\rightarrow$  determinizmus.
  - Ekvivalencia s konečnosťovými automatmi.
  - S aktívnymi membránami je model univerzálny.
- [7]: "min-enabled" computational step (= sekvenčný mód)
  - Ekvivalencia s konečnosťovými automatmi.
- Vlastnosti:
  - Pravidlá bez konfliktu (objekty sa môžu zúčastniť ako reaktanty súčasne vo viacerých pravidlách).
  - Ak je objekt použitý aspoň v jednom pravidle ako reaktant, bude spotrebovaný.

Kleijn a Koutny v roku 2011 skúmali rôzne mody výpočtu pre P systémy pracujúce s množinami a sekvenčný bol tiež spomenutý pod názvom "min-enabled".

Ukázala sa ekvivalencia s konečnosťovými automatmi.

Sekvenčné P systémy pracujúce s množinami majú tieto vlastnosti: Pravidlá sú bez konfliktu, lebo objekty sa môžu zúčastniť ako reaktanty súčasne vo viacerých pravidlách. Ak je objekt použitý aspoň v jednom pravidle ako reaktant, bude spotrebovaný.

# Biologicky motivované výpočtové modely

## └ Skúmané varianty P systémov

### └ Sekvenčné P systémy s množinami namiesto multimnožín

#### └ Aktívny P systém

- Membránová konfigurácia  $(T, l, c)$ , kde
  - $T$  je stromová štruktúra
  - $l : V(T) \rightarrow \{1, \dots, m\}$
  - $c : V(T) \rightarrow \mathbb{N}^{\Sigma}$
- Aktívny P systém je  $(\Sigma, C_0, R_1, R_2, \dots, R_m)$ , kde
  - $\Sigma$  je abeceda
  - $C_0$  je počiatočná membránová konfigurácia
  - $R_i$  je množina pravidiel

Na tomto mieste by som chcel znovu pripomenúť definíciu aktívneho P systému.

# Biologicky motivované výpočtové modely

## └ Skúmané varianty P systémov

### └ Sekvenčné P systémy s množinami namiesto multimnožín

#### └ Iné spôsoby vytvárania membrány

- Problémy pôvodnej definície:
  - Vytváranie membrány, ktorá už existuje
  - Posielanie objektu do neexistujúcej membrány
- Inject-or-create
- Wrap-or-create

Pravidlá, ktoré vytvárajú nové membrány, majú isté problémy. Napríklad, čo sa stane, ak sa dva krát po sebe vytvorí membrána s tým istým označením? Tu máme dve možnosti. Bud vytvorenie druhej membrány v definícii nejakým spôsobom zakážeme, aby sme zachovali invariant, kde existuje iba jedna membrána s daným označením. Alebo povolíme dve susedné membrány s rovnakým označením, ale potom treba riešiť situáciu, keď sa posiela objekt do membrány, či sa pošle do jednej, alebo do oboch.

Podľa pôvodnej definície je pravidlo neaplikovateľné aj v prípade, keď sa posiela objekt do membrány s označením, ktoré sa v aktuálnom regióne nenachádza.

- Problémy pôvodnej definície:
  - Vytváranie membrány, ktorá už existuje
  - Posielanie objektu do neexistujúcej membrány
- Inject-or-create
- Wrap-or-create

Preto sme vymysleli alternatívne definície vytvárania membrány. Inject or create zjednocuje pravidlo pre posielanie a pravidlo pre vytváranie membrány. V prípade, ak membrána s daným označením existuje, tak sa daný objekt do nej pošle. Ak neexistuje, tak sa daný objekt zabalí do novej membrány.

Wrap or create ponecháva explicitné pravidlo na vytvorenie novej membrány, ale ak membrána s daným označením už existuje, tak ju zabalí do novej membrány s tým istým označením.

	membrány	čas
original	$O(n)$	$O(n)$
original	$O(\log(n))$	$O(\log(n))$
inject-or-create	$O(\log(n))$	$O(\log(n))$
wrap-or-create	$O(n)$	$O(1)$

Pri všetkých variantoch sme ukázali Turingovskú úplnosť pomocou simulácie registrového stroja. Jednotlivé simulácie sme medzi sebou porovnali s ohľadom na dve veličiny. Merali sme maximálny počet membrán v niektorej konfigurácii P systému v závislosti od najvyššej hodnoty registra. A tiež sme merali počet krokov výpočtu P systému potrebných na simuláciu jedného kroku registrového stroja. V prvom riadku tabuľky je jednoduchá simulácia podľa pôvodnej definície, ktorá nie je veľmi efektívna. Na jeden krok registrového stroja je potrebných až  $O(n)$  krokov.

	membrány	čas
original	$O(n)$	$O(n)$
original	$O(\log(n))$	$O(\log(n))$
inject-or-create	$O(\log(n))$	$O(\log(n))$
wrap-or-create	$O(n)$	$O(1)$

Pomocou binárneho označenia membrán sa nám podarilo optimalizovať túto simuláciu na logaritmický čas.

Pomocou sémantiky inject or create sa sa podarilo simulovať registrový stroj podobným spôsobom s tou istou zložitosťou.

Sémantika wrap or create sa ukázala ako vhodnejšia, čo sa týka časovej zložitosti, ale počet vytvorených membrán v simulácii sa znížiť nepodarilo.


•

Tieto výsledky sme prezentovali a boli publikované v zborníku z konferencie Concurrency, Specification and Programming v roku 2015 v Rzesove. Rozšírená verzia článku bola zaslaná do karentovaného časopisu Fundamenta Informaticae.



- └ Skúmané varianty P systémov
- └ Detekcia prázdnoty membrán
- └ Detekcia prázdnoty membrán

## 4. Detekcia prázdnoty membrán

Zaoberali sme sa aj inými variantami P systémov, ktoré rôznymi spôsobmi umožňujú detekciu prázdnoty membrán a využívajú to pri výpočte. 

- └ Skúmané varianty P systémov
- └ Detekcia prázdnoty membrán
- └ Detekcia prázdnoty membrán

- Objekty vyhýbajúce sa prázdnej membráne
- Mutovanie objektov pri poslaní do prázdnej membrány
- Objekt reprezentujúci vákuum

Uvažovali sme napríklad objekty vyhýbajúce sa prázdnej membráne. Pravidlo posielajúce objekt do prázdnej membrány sa síce uplatní, ale daný objekt ostane v aktuálnej membráne.

---

V inom variante sa daný objekt síce pošle do membrány, ale ak je prázdna, tak sa z neho stane iný objekt.

---

A posledný variant obsahuje špeciálne objekty, ktoré reprezentujú vákuum. Takýto objekt sa vytvorí automaticky v prázdnej membráne, nemôže byť vytvorený prepisovacím pravidlom. Môže ale byť na ľavej strane pravidiel, teda interagovať s inými objektami. Pri týchto variantoch sme dosiahli iba čiastočné, alebo triviálne výsledky, ktoré neboli vhodné na publikáciu.

2018-01-13

## Biologicky motivované výpočtové modely

└ Skúmané varianty P systémov

└ Detekcia prázdnoty membrán

└ Vyjadrenia k posudkom (doc. Sosík)

Vyjadrenia k posudkom (doc. Sosík)

- The statement of Theorem 4.1.2 should be reformulated, although intuitive meaning is clear. *PsRE* does not equal to mentioned P systems but to the family of number sets they generate.
- **Theorem 4.1.2:** Sequential P systems with cooperative rules and inhibitors can simulate register machines and thus equal *PsRE*.

Komentár ohľadom znenia vety 4.1.2, kde sa tvrdí, že P systémy sú ekvivalentné *PsRE* - takto sa to často používa v literatúre.

2018-01-13

## Biologicky motivované výpočtové modely

└ Skúmané varianty P systémov

└ Detekcia prázdnoty membrán

└ Vyjadrenia k posudkom (doc. Sosík)

Vyjadrenia k posudkom (doc. Sosík)

- The statement of Theorem 4.1.2 should be reformulated, although intuitive meaning is clear. *PsRE* does not equal to mentioned P systems but to the family of number sets they generate.
- **Theorem 4.1.2:** Sequential P systems with cooperative rules and inhibitors can simulate register machines and thus **generate** *PsRE*.

Avšak správne má byť, že P systémy generujú triedu jazykov *PsRE*.

- Symbols in rules in Section 4.4.2 are sometimes separated by commas (Ex. 4.4.1), sometimes not (p. 87). In Section 4.2, separators  $|$  are sometimes used, sometimes not.
- **Example 4.4.1:**  $x_j, t_i \rightarrow x_i, t_j$
- **p. 87:**  $zst \rightarrow yst$
- **Definition 2.6.3** ... As elements of a multiset can also be strings, we separate them with the pipe symbol, e.g.  $element|element|other\_element$
- **Proof 4.3.1**  $e|a \rightarrow k \uparrow$

Další komentár bol ohľadom zápisu multimnožiny.

Chyby pramenili často z toho, že v rôznej literatúre sa používalo rôzne označenie. Oddelovanie čiarkami je chybné. Najviac prehľadné je podľa mňa zápis pomocou stringu, čiže bez oddeľovačov, ale v prípade, ak sú prvky multimnožiny viacznakové, tak sa oddeľia zvislou čiarou. Avšak v dôkaze 4.3.1 máme zvislú čiaru použitú aj v prípade jednoznakových prvkov multimnožiny.

- In rule 6 at p. 84, label 1 of the membrane should be  $i$ .
- $6 : x_j t_i \rightarrow [1 y_k t_i]_1$

Další komentár bol ohľadom označenia membrány v dôkaze. Pri simulácii operácie  $ADD(i)$  registrového stroja máme pravidlo, ktoré vytvorí membránu s označením  $i$ .

2018-01-13

## Biologicky motivované výpočtové modely

- └ Skúmané varianty P systémov
- └ Detekcia prázdnoty membrán
- └ Vyjadrenia k posudkom (doc. Sosík)

Vyjadrenia k posudkom (doc. Sosík)

- In rule 6 at p. 84, label 1 of the membrane should be  $i$ .
- $6 : x_1 t_1 \rightarrow [y_1 t_1]_1$

Má tam byť  $i$  namiesto 1.

2018-01-13

## Biologicky motivované výpočtové modely

- └ Skúmané varianty P systémov
- └ Detekcia prázdnoty membrán
- └ Vyjadrenia k posudkom (doc. Sosík)

Vyjadrenia k posudkom (doc. Sosík)

- It is not clear where Proof 4.4.1 ends. Example 4.4.1 presents the general part of the proof (translation of rules of a register machine into a P system), hence it should be denoted as an example.
- Dôkaz 4.4.1 má dve strany, Example 4.4.1 je jeden odstavec v strede. Potom ešte pokračuje dôkaz.

Dôkaz 4.4.1 má dve strany, Example 4.4.1 ukazuje konfiguráciu P systému pre ukážkovú konfiguráciu registrového stroja. Je to jeden odstavec v strede dôkazu, ktorý potom ešte pokračuje. Mohli sme dôkaz lepšie štruktúrovať.

- Dôkaz zrejme vyžaduje drobnú úpravu pre prípad  $M(a_i) > 1$  v pravidle  $r_j$  na str. 60
  - Áno, dôkaz funguje len pre pravidlá s ľavou stranou veľkosti nanajvýš 2
  - $a|RUN \rightarrow a|RUN| \rightarrow$
  - $\forall r_j \in R_i$  such that
- $$r_j = a_1^{M(a_1)} a_2^{M(a_2)} \dots a_n^{M(a_n)} \rightarrow a_1^{N(a_1)} a_2^{N(a_2)} \dots a_n^{N(a_n)}$$
- we will have the following rules ( $\forall 0 \leq m_k \leq \min(M(a_k), 1)$ ):
- $$a_1^{M(a_1)-m_1} a_2^{M(a_2)-m_2} a_3^{M(a_3)-m_3} \dots a_n^{M(a_n)-m_n} |RUN \rightarrow a_1^{N(a_1)-m_1} a_2^{N(a_2)-m_2} \dots a_n^{N(a_n)-m_n} |RUN$$

Dôkaz zrejme vyžaduje drobnú úpravu pre prípad  $M(a_i) > 1$

Áno, máte pravdu. Dôkaz funguje len pre pravidlá s ľavou stranou veľkosti nanajvýš 2.

V simulácii sa vytvárajú objekty označené bodkou, v regione môže byť nanajvýš jeden objekt označený bodkou. Tento sa dá používať pri prepisovacích pravidlách rovnako, ako by to bol objekt bez bodky.

- Dôkaz zrejme vyžaduje drobnú úpravu pre prípad  $M(a_i) > 1$  v pravidle  $r_j$  na str. 60
  - Áno, dôkaz funguje len pre pravidlá s ľavou stranou veľkosti nanajvýš 2
  - $a|RUN \rightarrow a|RUN| \rightarrow$
  - $\forall r_j \in R_i$  such that
- $$r_j = a_1^{M(a_1)} a_2^{M(a_2)} \dots a_n^{M(a_n)} \rightarrow a_1^{N(a_1)} a_2^{N(a_2)} \dots a_n^{N(a_n)}$$
- we will have the following rules ( $\forall 0 \leq m_k \leq \min(M(a_k), 1)$ ):
- $$a_1^{M(a_1)-m_1} a_2^{M(a_2)-m_2} a_3^{M(a_3)-m_3} \dots a_n^{M(a_n)-m_n} |RUN \rightarrow a_1^{N(a_1)-m_1} a_2^{N(a_2)-m_2} \dots a_n^{N(a_n)-m_n} |RUN$$

Ale umožňuje zistiť, kedy už pravidlo, ktoré má na ľavej strane 2 rovnaké objekty, nie je aplikovateľné.

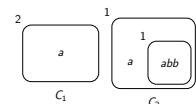
Pre pravidlá s ľavou stranou obsahujúcou 3 rovnaké objekty by sme potrebovali, aby  $M(a_i) = 2$ . Dalo by sa to spraviť zavedením symbolov s dvomi bodkami, ktoré by mali v každej membráne nanajvýš 1 výskyt. Navyše by symbol s dvomi bodkami mohol vzniknúť iba, ak je prítomný symbol s jednou bodkou. Tým pádom by sme vedeli zistiť, že dané pravidlo s tromi rovnakými symbolmi na ľavej strane nie je aplikovateľné. Analogicky by to bolo aj pre väčšie počty symbolov na ľavej strane.

- Pozor na formuláciu v dôkaze 4.2.6. Nekonečná postupnosť môže byť aj konštantná a vtedy rastúci pár neexistuje. Analogicky v dôkaze 4.2.7 treba rastúci pár zameniť za neklesajúci pár.
- Áno, má tam byť neklesajúci. Hoci je uvedené znamienko  $\leq$ , v texte je použité "increasing".

V dôkazoch treba rastúci pár zameniť za neklesajúci pár.

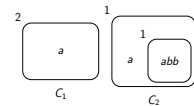
Áno, má tam byť neklesajúci. Hoci je uvedené znamienko  $\leq$ , v texte je použité "increasing".

- Ak porovnávam kódy ako refazce,  $enc(C_1) < enc(C_2)$  môže platiť aj v situácii, keď príslušné "stromy" nie sú izomorfné, čo podľa môjho názoru znamená, že dôkaz Lemy 4.2.5 neplatí (opačná implikácia platí).
- $enc(C_1) = 1001\ 0000\ 0000\ 0000$ ,  
 $enc(C_2) = 0000\ 0000\ 1010\ 1210$



Keď stromy nie sú izomorfné, tak podľa preorderu dostanú iný order number, a teda v kódovaní bude nenulová iná časť. Tým pádom nebude platiť ani  $enc(C_1) < enc(C_2)$  ani  $enc(C_2) < enc(C_1)$ . Porovnanie kódovaní nie je alfabeticke, ale podľa pozícií. Na to, aby bolo jedno kódovanie väčšie ako druhé, musí byť aspoň na jednej pozícii väčšie a na ostatných väčšie alebo rovné. Inými slovami - ak sú kódovania porovnateľné  $enc(C_1) < enc(C_2)$  alebo  $enc(C_2) < enc(C_1)$ , potom už nutne musí byť nenulová tá istá časť a teda stromy sú izomorfné.

- Ak porovnávať kódy ako refazce,  $enc(C_1) < enc(C_2)$  môže platiť aj v situácii, keď príslušné "stromy" nie sú izomorfné, čo podľa nášho názoru znamená, že dôkaz Lemy 4.2.5 neplatí (opačná implikácia platí).
- $enc(C_1) = 1001\ 0000\ 0000\ 0000$ ,  
 $enc(C_2) = 0000\ 0000\ 1010\ 1210$



Na tomto príklade máme stanovený limit na počet membrán 2. Existujú len dve navzájom neizomorfné stromové štruktúry s nanajvýš dvomi vrcholmi, buď je to samostatný vrchol alebo koreň plus jeden vrchol. Preto kodovanie pozostáva z dvoch častí. Uvádzam príklady dvoch membránových konfigurácií. Prvá má poradové číslo 1, preto je nenulová iba prvá časť kodovania. Druhá má poradové číslo 2, preto je nenulová len druhá časť kodovania. Kodovanie jednej časti je zreťazenie kodovania jednotlivých membrán po prechode preorderom. V prvej konfigurácii máme jeden objekt *a* a nula objektov *b*, preto sa kodovanie začína 10. Označenie membrány je 2, preto kodovanímá na príslušnej pozícii 1. Kodovanie druhej konfigurácie začína rovnako, 10, ale keďže má označenie 1, tak pokračuje 10. Vnútoraná membrána obsahuje jedno *a* a dve *b*, preto kodovanie pokračuje 12 a keďže má označenie 1, tak sa kodovanie zakončí 10.

- Je nutné dávať dávať umelý predpoklad na ohraničenie počtu membrán zvonku cez zablokovanie aplikovateľnosti pravidiel vytvárajúceho novú membránu v situácii, ktorá by viedla k prekročeniu stanoveného počtu membrán keď aktívne P-systémy s obmedzeným sumárnym počtom membrán sú univerzálne?
- V dôkaze využívame tento limit pri stanovení počtu navzájom neizomorfných stromov.
- Nekonečná postupnosť membránových štruktúr  $\Rightarrow$  dve štruktúry  $T_1, T_2$  s nejakou vlastnosťou, vďaka ktorej budeme môcť tvrdiť, že postupnosť je nekonečná.
  - $T_1$  je podstrom  $T_2$
  - $parent(v_1) = v_2$  v  $T_1$ , potom  $parent'(v_1) = v_2$  v  $T_2$

Je nutné dávať dávať umelý predpoklad na ohraničenie počtu membrán?

V dôkaze využívame tento limit pri stanovení počtu navzájom neizomorfných stromov. Bez tohto umelého predpokladu by dôkaz nefungoval. Máme hypotézu, že by to šlo pomocou tvrdenia, že ak máme nekonečnú postupnosť membránových štruktúr, kde susedné prvky postupnosti sa líšia len vytvorením alebo rozpustením membrány, tak sa v nej nájdu dva prvky postupnosti s nejakou vlastnosťou, vďaka ktorej budeme môcť tvrdiť, že postupnosť je nekonečná.

# Biologicky motivované výpočtové modely

## └ Skúmané varianty P systémov

### └ Detekcia prázdnoty membrán

### └ Vyjadrenia k posudkom (doc. Pardubská)

#### Vyjadrenia k posudkom (doc. Pardubská)

- Je nutné dávať dávať umelý predpoklad na ohraničenie počtu membrán zvonku cez zablokovanie aplikovateľnosti pravidiel vytvárajúceho novú membránu v situácii, ktorá by viedla k prekročeniu stanoveného počtu membrán keď aktívne P-systémy s obmedzeným sumárnym počtom membrán sú univerzálne?
- V dôkazoch využívame tento limit pri stanovení počtu navzájom neizomorfných stromov.
- Nekonečná postupnosť membránových štruktúr  $\Rightarrow$  dve štruktúry  $T_1, T_2$  s nejakou vlastnosťou, vďaka ktorej budeme môcť tvrdiť, že postupnosť je nekonečná.
  - $T_1$  je podstrom  $T_2$
  - $\text{parent}(v_1) = v_2$  v  $T_1$ , potom  $\text{parent}^*(v_1) = v_2$  v  $T_2$

Hľadali sme danú vlastnosť. Ako prvé sme skúsili, že membránová štruktúra je podstromom štruktúry, ktorá sa nachádza niekde neskôr v postupnosti. Toto sa ukázalo ako nedostatočná požiadavka. Nevyplývalo z toho, že postupnosť je nekonečná. Vlastnosť, kde sa zachováva, že vzťah parent-child sa zachováva (akurät sa medzi ne môže dostať nová membrána), sa tiež ukázala byť nedostatočná. Predpokladáme, že existuje nejaká štrukturálna vlastnosť, ktorá sa zachová, umožní tvrdiť, že postupnosť je nekonečná.

# Biologicky motivované výpočtové modely

## └ Skúmané varianty P systémov

### └ Detekcia prázdnoty membrán

### └ Vyjadrenia k posudkom

#### Vyjadrenia k posudkom

- Mohli by ste vysvetliť motivácie pre definované modifikácie P-systémov v závere kapitoly 4?
- V pôvodnej definícii:
  - Posielanie do membrány je definované iba pre prípad, kedy cieľová membrána existuje.
  - Vytvorenie novej membrány bolo definované iba pre prípad, kedy cieľová membrána neexistuje
  - Prirodzene sa žiadalo zjednotiť tieto dva pojmy, aby výsledný jeden pojem bol definovaný vo všetkých prípadoch, aj keď cieľová membrána existuje, aj keď neexistuje  $\Rightarrow$  inject-or-create

Motivácia vôbec niečo modifikovať pramenila z toho, že v pôvodnej definícii bolo posielanie do membrány definované iba pre prípad, kedy cieľová membrána existuje. Vytvorenie novej membrány bolo definované iba pre prípad, kedy cieľová membrána neexistuje. Prirodzene sa žiadalo zjednotiť tieto dva pojmy, aby výsledný jeden pojem bol definovaný vo všetkých prípadoch, aj keď cieľová membrána existuje, aj keď neexistuje. Tak vznikla modifikácia inject or create.

Wrap or create na druhej strane zabezpečuje, aby pravidlo pre vytvorenie membrány v každej situácii nejakú membránu vytvorilo.