Паралелни програмски систем за пребројавање коначних структура

Александар Пејовић

ментор:
др Силвиа Гилезан

План

- Циљ данашњег излагања је да се представе како сама теза тако и добијени резулати
- Излагање је организовано на следећи начин:
  - Уводни пример
  - Проблематика тезе
  - Приказ система
  - Примене
  - Закључак

- Теза се бави проблематиком из теорије коначних модела
- Попут већине развијених области математике, и теорија коначних модела је сама по себи апстрактна
- Зато ћемо излагање започети увођењем основних концепата теорије коначних модела на једном свакодневном примеру

Уводни пример

- Посматрајмо спортски турнир на коме учествују 3 играча А, Б и Ц
- Игра се по принципу свако са сваким и победник је играч са највише победа
- Занимају нас питања попут тога да ли увек постоји победник и ако постоји под којим условима

Уводни пример

Турнир са победником

Играч	Победа
А	2
Б	1
Ц	0

Уводни пример

Турнир са победником

Играч	Победа
А	2
Б	1
Ц	0

Турнир без победника

Играч	Победа
А	1
Б	1
Ц	1

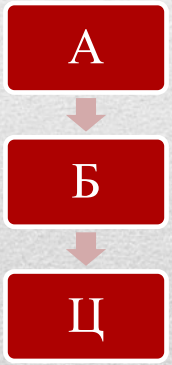
Уводни пример

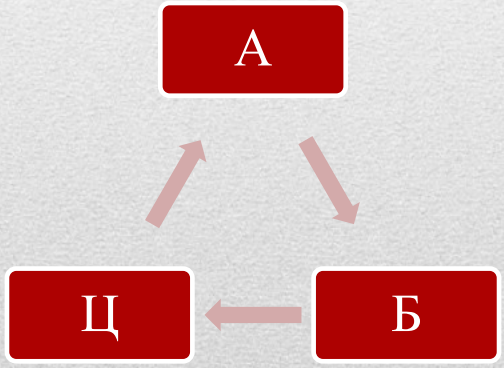
Турнир са победником

Играч	Победа
А	2
Б	1
Ц	0

Турнир без победника

Играч	Победа
А	1
Б	1
Ц	1





Уводни пример

- У свакодневном смислу дати пример је разумљив
- Али да би могли да резонујемо о турнирима са метематичког аспекта то није довољно
- Потребно је дефинисати шта је то турнир, односно шта значи победити

Уводни пример

- Турнире моделујемо помоћу теорије T језика $L = \{>\}$, где је $>$ симбол бинарне релације *победити*
  - $x > y$ читамо као x је победио y
- Аксиоме теорије T су
  1. $\forall x \forall y \; x > y \vee y > x$
  2. $\forall x \forall y \; x > y \Rightarrow \neg(y > x)$
- Другим речима, турнири су управо модели теорије T

Уводни пример

- Основни проблем који се разматра у тези је генерисање и пребројавање коначних структура, односно коначних модела теорија првог реда
- Главни резултат тезе представља развој низа алгоритама и софтверског пакета базираног на њима за паралелно генерисање и пребројавање

Проблематика тезе

Проблематика тезе

- Место ове тезе може се сагледати у контексту коначне комбинаторике и теорије графова, односно генерално дискретне математике као важног дела савремене математике
- Наиме у овим областима, главни проблеми су поред класификације структура које се у њима јављају, управо питања егзистенције, пребројавања и генерисања модела
  - OEIS (The On-Line Encyclopedia of Integer Sequences) има преко 300 хиљада уноса

Проблематика тезе

- Треба истаћи и да ефикасно решавање поменутих проблема има и практичне примене
- Оне се крећу од оптимизације и верификације логичких кола, па све до генерисања распореда часова
- Такође постоје и други софтвери који се баве истом проблематиком
  - Mace4 (развио W. McCune)
  - Paradox (развили K. Claessen и N. Sörensson)
  - PSATO (развили H. Zhang, M. P. Bonacina и J. Hsiang)

- Наш систем по могућностима спада у групу генералних система
- Замишљен је као Python-ova библиотека за развој наменских генератора, односно наменских бројача коначних модела
- Скоро цео је имплементиран у Python-у
- Само су хардверски убрзане примитиве имплементиране у OpenCL-у

Приказ система

- Логички гледано, систем се састоји од неколико модула
  - Модул за унос теорија првог реда
  - Модул за превод у исказне теорије
  - Модул за парцијалну евалуацију исказних формула
  - Рачунско језгро

Приказ система

Унос теорија првог реда

- Унутар система формуле су представљене као апстрактна синтаксна стабла
  - $p \wedge q \vee \neg r$
- Ова репрезентација је погодна за програмску манипулацију
- Међутим, није згодна за ручни унос
  - `Or(And(Var('p'), Var('q')), Not(Var('r')))`

- Зато смо развили DSL за креирање AST-ова
- Основна идеја DSL-а је једноставна
  - Формуле пишемо као одговарајуће Python-ове изразе
  - А као резулат евалуације добијамо њихове AST-ове
- Предност овог дизајна
  - Не захтева развој додатног парсера
  - Добро се интегрише са остатком система

Унос теорија првог реда

- Пример исказне формуле
  - `p & q | ~r`
- Пример формула логике првог реда
  - `A[i : S](p(i, i))`
  - `A[i : S].E[j : S](p(i, j))`
  - `A[k : S].A[i, j : D](~p(i, k) | ~p(j, k))`

Унос теорија првог реда

Превод у исказне теорије

- Нека је L коначан језик првог реда, а L_A језик L проширен симболима константи – именима елемената коначног домена A. Скуп исказних слова $\mathcal{P}$ дефинишемо на следећи начин

$$\mathcal{P} = \left\{ p_{F_{a_1 \ldots a_k b}} \middle| a_1, \ldots, a_k, b \in A \right\} \cup \left\{ q_{R_{a_1 \ldots a_k}} \middle| a_1, \ldots, a_k \in A \right\}$$

Превод у исказне теорије

- Затим рекурзивно дефинишемо пресликавање превођења $*$ из скупа Sent_{L_A} свих $L_{\omega\omega}$-реченица из L_A у скуп $L_\omega^{\mathcal{P}}$ исказних формула над $\mathcal{P}$
- У основи, пресликавање $*$ преводи универзалне и егзистенцијалне квантификаторе редом у коначне коњункције и дисјункције параметризованих формула без квантификатора
  - Нека исказно слово p_{ij} значи $i \leq j$, тада се рефлексивност $\forall x\ x \leq x$ преводи у $\bigwedge_i p_{ii}$

- Имплементирана је као AST трансформација
- Резултат је ново AST са редукованим бројем исказних променљивих у коме нема константи
- Наиме, по замени исказне променљиве додељеном вредношћу, врши се симболичко рачунање применом идентитета попут
  - `p & 0 → 0`
  - `p & 1 → p`
  - `p & ~p → 0`
  - `p & p → p`
  - `~~p → p`
  - …

Парцијална евалуација исказних формула

- Нуди предефинисане јединице за извршавање
  - `GPURunner`
  - `CPURunner`
- Имплементира хардверски убрзане примитиве
  1. `eval(expression)`
  2. `count(expression)`
  3. `findmodel(expression)`
  4. `exist(expression)`

Рачунско језгро

- Свака примитива има свој OpenCL кернел
- Међутим, сви кернели деле заједничку функцију `f`
- Ова функција имплементира паралелизовану евалуацију исказне формуле

$$t^{\Omega_n}(b_1, \ldots, b_n)$$

и представља срж наших рачунских кернела

Рачунско језгро

Free vectors	Segments							
	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7
ω_0	0000	0000	0000	0000	1111	1111	1111	1111
ω_1	0000	0000	1111	1111	0000	0000	1111	1111
ω_2	0000	1111	0000	1111	0000	1111	0000	1111
ω_3	0011	0011	0011	0011	0011	0011	0011	0011
ω_4	0101	0101	0101	0101	0101	0101	0101	0101

Рачунско језгро

- Парцијална уређења су модели теорије T језика $L = \{\leq\}$, где је $\leq$ бинарни релацијски симбол. Аксиоме теорије T су
  1. $x \leq x$ (рефлексивност)
  2. $x \leq y \wedge y \leq z \Rightarrow x \leq z$ (транзитивност)
  3. $x \leq y \wedge y \leq x \Rightarrow x \equiv y$ (антисиметричност)

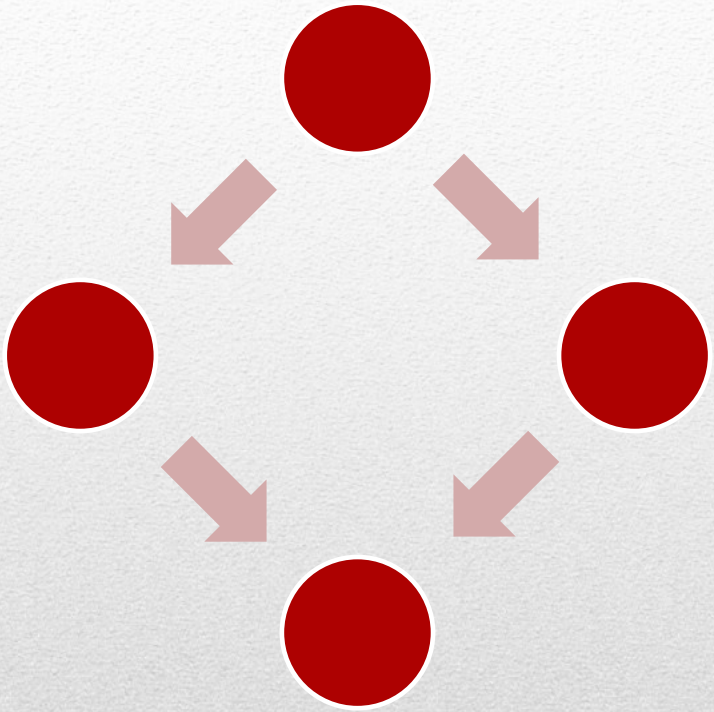
Примене

- Поступак превођења датих аксиома у исказну теорију доводи до експлозије исказних променљивих
  - Ако имамо k унарних релација $|\mathcal{P}| = kn$
  - Ако имамо једну бинарну релацију $|\mathcal{P}| = n^2$
  - Ако имамо једну бинарну операцију $|\mathcal{P}| = n^3$

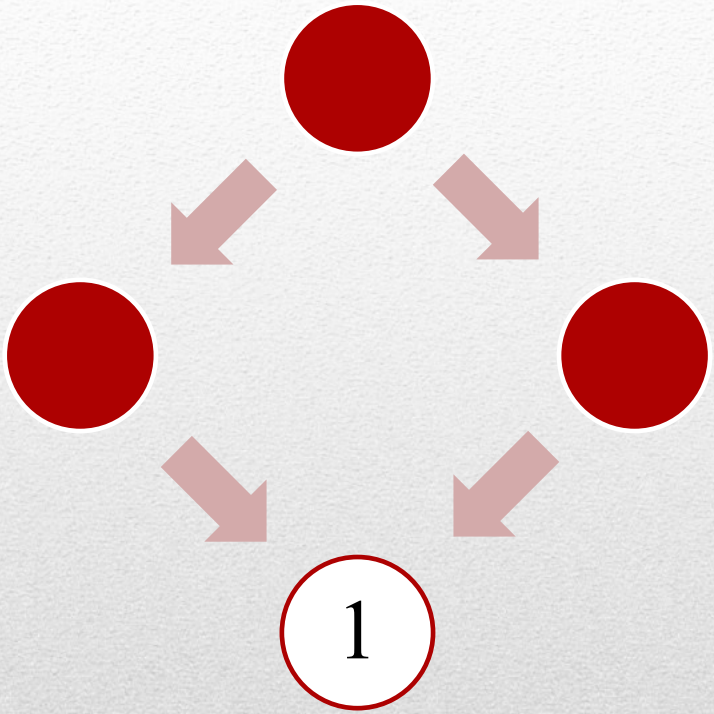
Примене

- Зато нам је била потребна нека стратегија за редуковање броја исказних променљивих
  - Могуће су различите стратегије
  - Углавном се своде на додељивање вредности одређеним променљивама
- У тези смо развили стратегију која се заснива на теорији дефинабилности

Примене



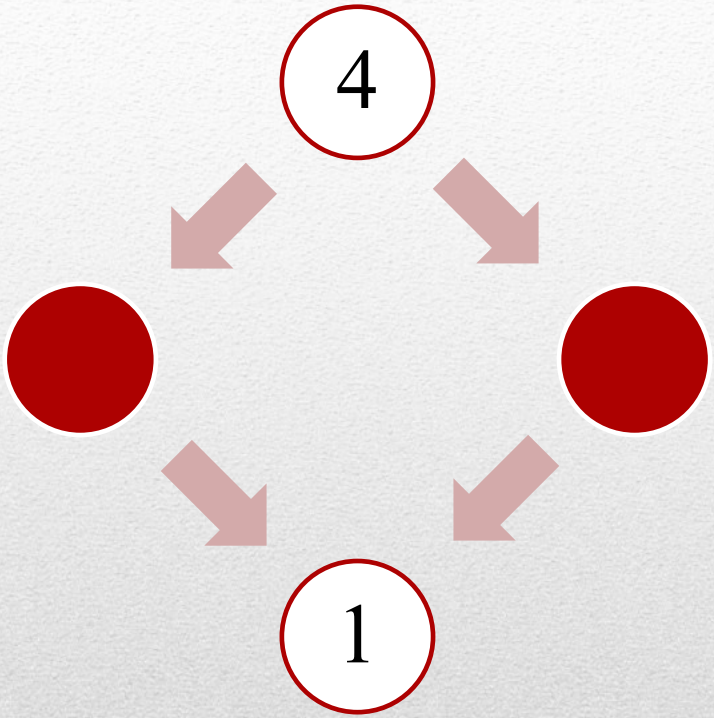
Примене



најмањи елемент

Примене

највећи елемент

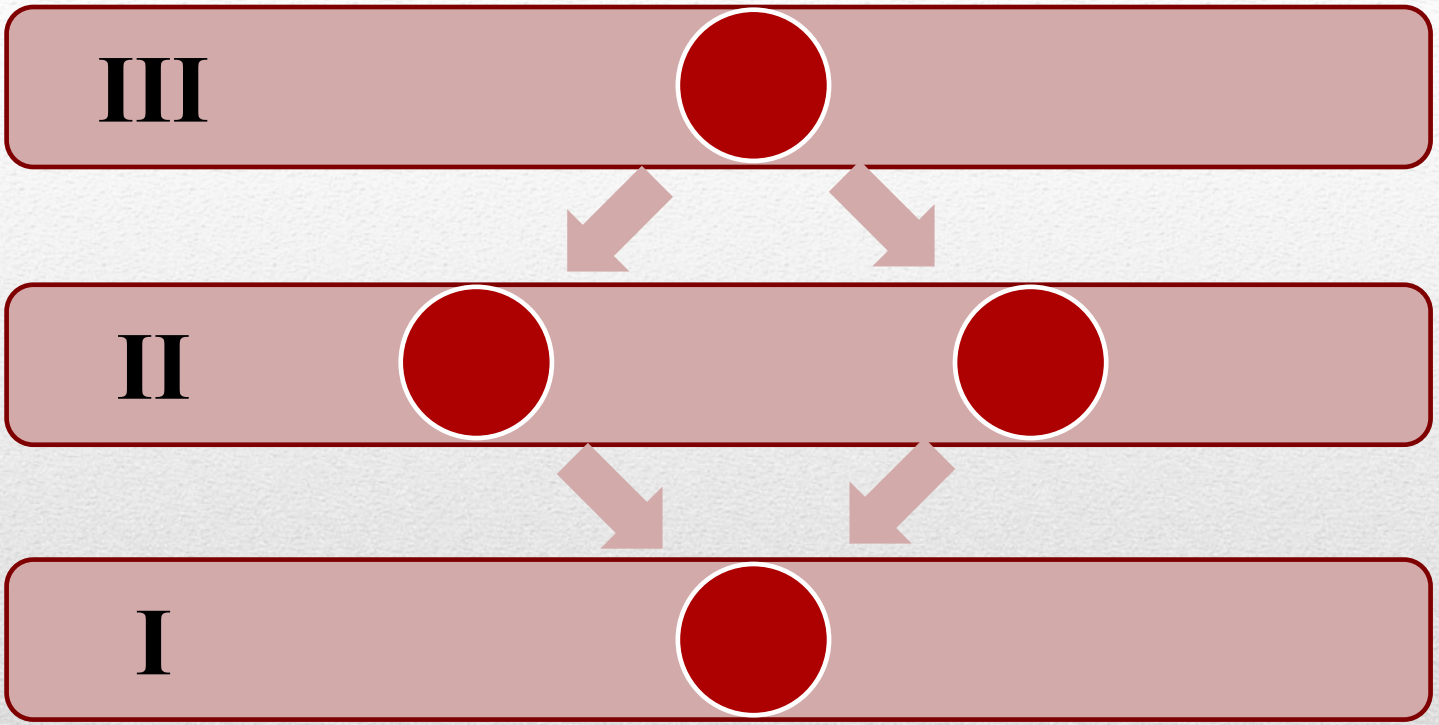


најмањи елемент

Примене

- Елементи попут најмањег и највећег су примери дефинабилних константи
- То значи да постоји формула $\varphi(x)$ језика L која их дефинише
  - $\exists g \forall x \; x \leq g$
- Ова идеја може да се генерализује и на дефинабилне подскупове

Примене



Примене

- Алгоритам за пребројавање парцијалних уређења се састоји из следећих корака
  1. Генеришемо све могуће конфигурације нивоа датог домена
     - Рецимо за домен од 3 елемента: 3, 1+2, 2+1, 1+1+1
  2. За сваку конфигурацију нивоа
     a) Генеришемо формуле које ту конфигурацију дефинишу
     b) За тако добијену дефиницију конфигурације нађемо број модела помоћу система
     c) На основу нађеног броја реконструишемо укупан број модела у датој конфигурацији
  3. Нађемо укупан број модела сумирањем броја модела у свим конфигурацијама

Примене

- У тези смо приказани алгоритам користили за бројање мрежа
- Успели смо да пребројимо мреже на доменима до 15 елемената
- Другим речима, успели смо да повећамо домен са којим можемо да радимо преко 2 пута

Примене

- Семантика нашег система је предикатска логика првог реда и зато можемо генерисати и бројати моделе било које класе коначних структура које се могу изразити овом логиком
- Очигледно ограничење представља раст броја исказних промењљивих у опису дате класе модела
- Међутим, са добрим избором адекватне потклассе модела и вештом редукцијом промењљивих, могу се добити нови и занимљиви резултати у рачунарској дискретној математици

Закључак

Хвала!