Анализ и извличане на компютърни програми (модели и методи за верификация на свойства на нерелационни бази от данни)

Калин Георгиев

19 юли 2013 г.

Увод: тема на изследването

- Програми, които генерират програми и програми, които анализират поведението на програми.
- В контекста на търсенето на решение на проблема за осигуряване на референтна цялост в документно-ориентирани нерелационни бази данни.

Увод: тема на изследването

- Програми, които генерират програми и програми, които анализират поведението на програми.
- В контекста на търсенето на решение на проблема за осигуряване на референтна цялост в документно-ориентирани нерелационни бази данни.

Структура на презентацията

- Увод в предметната област
- Изследователски въпрос и хипотеза на изследването
- Цели и задачи
- Формален модел на проблема
- Формално въведение на решението
- Доказателство за коректност на решението
- Практическа реализация
- Заключение

Анализ и извличане на програми

Извличане на програми: практически методи

Често повтарящи се програмни конструкции; програмни конструкции, които са сложни или дълги за ръчно описание; такива за които коректността е от особено значение

Практически методи

- Програмиране на базата на модели
- Код за потребителски интерфейс
- Web услуги
- Аспектно-ориентирано програмиране
- Генеративно програмиране
- Адаптиране на код

“Автоматичното програмиране винаги е било евфемизъм за програмиране на език на по-високо ниво спрямо езика, който е бил достъпен за програмиста в дадения момент”.
–Дейвид Парнас

Извличане на програми: практически методи

Често повтарящи се програмни конструкции; програмни конструкции, които са сложни или дълги за ръчно описание; такива за които коректността е от особено значение

Практически методи

- Програмиране на базата на модели
- Код за потребителски интерфейс
- Web услуги
- Аспектно-ориентирано програмиране
- Генеративно програмиране
- Адаптиране на код

“Автоматичното програмиране винаги е било евфемизъм за програмиране на език на по-високо ниво спрямо езика, който е бил достъпен за програмиста в дадения момент”.
–Дейвид Парнас

Извличане на програми: формални методи

- Lex & Yacc
- Системи за доказателство на теореми
- Системи за компютърна алгебра
- Трансформатори на предикати
- Извличане на код от (неконструктивни) доказателства

Анализ на програми

Анализ на програми

Динамичен анализ

- Изчерпване на състоянията
- Автоматизирано тестване
- Верификация на изпълнението
- Изолирани тестове

Статичен анализ

- Lint
- Статично типизиране
- Системи: KeY, JML, ACL2, PVS
- Системи за доказателство на теореми

Формални методи

- CTL, LTL, Автомати на Бюхи
- Код с доказателство
- Метод на Флойд и логика на Флойд-Хоар
- Индукционно правило на Скот

Анализ на програми

Динамичен анализ

- Изчерпване на състоянията
- Автоматизирано тестване
- Верификация на изпълнението
- Изолирани тестове

Статичен анализ

- Lint
- Статично типизиране
- Системи: KeY, JML, ACL2, PVS
- Системи за доказателство на теореми

Формални методи

- CTL, LTL, Автомати на Бюхи
- Код с доказателство
- Метод на Флойд и логика на Флойд-Хоар
- Индукционно правило на Скот

Анализ на програми

Динамичен анализ

- Изчерпване на състоянията
- Автоматизирано тестване
- Верификация на изпълнението
- Изолирани тестове

Статичен анализ

- Lint
- Статично типизиране
- Системи: KeY, JML, ACL2, PVS
- Системи за доказателство на теореми

Формални методи

- CTL, LTL, Автомати на Бюхи
- Код с доказателство
- Метод на Флойд и логика на Флойд-Хоар
- Индукционно правило на Скот

Нерелационни бази данни

НРСУБД: Характеристика и видове

Характеристика

- Класическият релационен модел е неприложим
- Не поддържат информацията под формата на таблици и не предоставят поддръжка на стандартния език за структурирани заявки SQL
- Често не предоставят функционалност отвъд съхранението на записите
- Нарушават ACID

Видове

- key-value
- Графови бази данни
- Документни бази данни

НРСУБД: Характеристика и видове

Характеристика

- Класическият релационен модел е неприложим
- Не поддържат информацията под формата на таблици и не предоставят поддръжка на стандартния език за структурирани заявки SQL
- Често не предоставят функционалност отвъд съхранението на записите
- Нарушават ACID

Видове

- key-value
- Графови бази данни
- Документни бази данни

НРСУБД: Проблем с референтната цялост

- СУБД не поддържат ограничения за цялостност
- В частност външни ключове
- Проблем с изолацията

Цели и задачи

Изследователски въпрос

- Как да се изследва коректността на поведението на системи, генериращи данни в документно-ориентирани нерелационни бази данни от гледна точка на референтната цялост на данните и някои други сходни свойства
- Как да се формализират тези изисквания към данните и как от спецификации на такива изисквания да се генерират програми, анализиращи коректността на данните

Хипотеза

Свойствата за коректност на данните в нерелационни бази данни могат да се формализират по такъв начин, че от тях по автоматичен път да се синтезират програми, проверяващи тези свойства. От своя страна, тези синтезирани програми ще извършват динамичен анализ на поведението на системите, генериращи данните.

Цели

Целта на дисертационния труд е:

- запознаване с методи и средства за извличане и анализ на програми,
- мотивация на верификация на свойства на нерелационни бази данни,
- дефиниране, математическа обосновка и експериментална реализация на метод за анализ на свойства на документно ориентирани нерелационни бази данни.
- Методът да се основава на техники за извличане и анализ на програми, които се прилагат в изчислителната среда за паралелни изчисления MapReduce.

Задачи (1)

- Да се проучат основни методи и техники, свързани с извличането и анализа на програми; да се направи анализ на възможностите и приложимостта им в практиката; да се намери възможност за приложението им в платформата за паралелни изчисления MapReduce в контекста на СУНРДБ.
- Да се построи формален модел за описание и изследване на свойства на връзките между документи в нерелационни бази данни.
- Да се дефинира клас от свойства на връзките между документи, които ще бъдат анализирани автоматично.

Задачи (2)

- Да се дефинира метод за автоматично генериране на програми, проверяващи описаните типове свойства на базата на тяхна спецификация.
- Да се изследва коректността на дефинирания метод (и програмите, които се генерират чрез него).
- Да се направи експериментална реализация на метода.

Формален модел на проблема

Val-множество на допустимите стойности

- Примитивни типове: **Prim** $\subset$ **Val** - числа, низове, булеви
- Множества от двойки (етикет, стойност):
  $val = \{(l_1, v_1), ..., (l_k, v_k)\} \in$ **Val**, когато $v_i \in$ **Val**, $l_i \in$ **S**, $val[l_i] := v_i$
- Масиви от стойности: $arr = (v_1, ..., v_k) \in$ **Val**, когато $v_i \in$ **Val**,
  $arr[i] := v_i$

Тип на стойност/документ

Дефинираме изображението $type : \mathbf{Val} \to \mathbf{T}$, задаващо типа на дадена стойност.

- Ако $d \in \mathbf{N}, \mathbf{S}, \mathbf{B}$, то $type(d) = \mathbf{N}, \mathbf{S}, \mathbf{B}$, съответно. $\chi(type(d)) := prim$
- Ако $d = \{(l_1, v_1), ..., (l_k, v_k)\}$, то $type(d) = \{l_1 : type(v_1), ..., l_k : type(v_k)\}$. $\chi(type(d)) := object$
- Ако $d = (v_1, ..., v_k)$, то $type(d) = (type(v_1), ..., type(v_k))$. $\chi(type(d)) := array$

- Алтернатива на понятието схема
- Индуктивна дефиниция: позволява вложения
- Редът на полетата не е от значение

Тип на стойност/документ

Дефинираме изображението $type : \mathbf{Val} \to \mathbf{T}$, задаващо типа на дадена стойност.

- Ако $d \in \mathbf{N}, \mathbf{S}, \mathbf{B}$, то $type(d) = \mathbf{N}, \mathbf{S}, \mathbf{B}$, съответно. $\chi(type(d)) := prim$
- Ако $d = \{(l_1, v_1), ..., (l_k, v_k)\}$, то $type(d) = \{l_1 : type(v_1), ..., l_k : type(v_k)\}$. $\chi(type(d)) := object$
- Ако $d = (v_1, ..., v_k)$, то $type(d) = (type(v_1), ..., type(v_k))$. $\chi(type(d)) := array$

- Алтернатива на понятието схема
- Индуктивна дефиниция: позволява вложения
- Редът на полетата не е от значение

Съвпадение на типове

$type(d_1) \iff type(d_2)$, ако е изпълнено едно от следните твърдения:

- $type(d_1) = type(d_2)$ или
- $type(d_1) = (\mathcal{T}_{1_1}, ..., \mathcal{T}_{1_k})$, $type(d_2) = (\mathcal{T}_{2_1}, ..., \mathcal{T}_{2_l})$ за някое k и l, и $\mathcal{T}_{1_1} = \mathcal{T}_{1_2} = ... = \mathcal{T}_{1_k} = \mathcal{T}_{2_1} = \mathcal{T}_{2_2} = ... = \mathcal{T}_{2_l}$, или
- $type(d_1) = (\mathcal{T}_1, ..., \mathcal{T}_k)$, $type(d_2) = []$, независимо от $\mathcal{T}_1, ..., \mathcal{T}_k$. Релацията е в сила и за симетричния случай. Или
- $type(d_1) = \{l_1 : t_1, ..., l_k : t_k\}$, $type(d_2) = \{l_1 : t'_1, ..., l_k : t'_k\}$ и $type(t_i) \iff type(t'_i)$.

Унифицира хомогенни масиви с различен брой елементи. Унифицира празния масив с всеки хомогенен масив.

Подтип

Нека t и $t' \in \mathbf{T}$, казваме, че t е подтип на t' и означаваме с $t \leq t'$, ако е изпълнено едно от следните условия:

- $t \iff t'$
- $\chi(t) = \chi(t') = \mathit{object}, \forall(f : t_1) \in t, \exists(f : t_2) \in t' : (t_1 \leq t_2)$.

Съвместимост

Съвместимост на стойност $v \in \mathbf{Val}$ с тип $t \in \mathbf{T}$.

Казваме, че стойността v е съвместима с типа t и означаваме с $t \models v$, тогава и само тогава, когато $type(v) \leq t$.

Съдържане на поле в тип

$t \vdash l$, тогава и само тогава, когато $\chi(t) = object$ и $\exists(f, t') \in \mathbf{T}$. За всяко поле, такова че $t \vdash l$, дефинираме изображението $[] : \mathbf{T} \to \mathbf{T}$ по следния начин: $t[l] = t'$.

Дефиницията за съдържане е продължена индуктивно и позволява "навлизане" във вложените записи: $t \vdash (l_1, ..., l_k)$ за $t \in \mathbf{T}$ и $(l_1, ..., l_k) \in \mathbf{S}^*$.

Колекции и документи

Колекция наричаме всяко множество от документи (елементи на **Doc**), а база данни - всяко множество от колекции.

Полу-схема на колекция C

Типа $s(C) \in \mathbf{T}$, такъв че:

- $\forall d \in C : (type(d) \leq s(C))$.
- $\forall t \in \mathbf{T}$, такова че $s(C) \leq t$, $\exists d \in C : (type(d) \not\leq t)$.

Ключ в колекция

Нека $key \in \mathbf{S}$ (или $\mathbf{S}^*$) е такова, че $s(C) \vdash key$. key наричаме ключ за колекцията C, тогава и само тогава, когато
$\forall d_1 \neq d_2 \in C : d_1[key] \neq d_2[key]$.

Ключът е по-скоро уникален идентификатор на документите в колекцията.

Връзка "едно към много"

Нека $key_1 \in \mathbf{S}^*$ е ключ за колекцията C_1 и $p \in \mathbf{S}^*$ е такъв етикет, че $s(C_2) \vdash p$. Двойката (key, p) наричаме връзка "едно към много" тогава и само тогава, когато $range_{C_2}(p) \subseteq range_{C_1}(key)$. key наричаме ключ на връзката, а p - референция.

Формулировка на проблема за връзки "едно към много"

$$bad_{C_1,C_2}(key, p) = \{d | d \in C_2 \wedge |d[p]| - range_{C_1}(key)) \neq \emptyset\}$$

Т.е. множеството *bad* се състои от тези елементи, които "нарушават" връзката "едно към много".

Твърдение

Нека $key_1 \in \mathbf{S}^*$ *е ключ за колекцията* C_1 *и* $p \in \mathbf{S}^*$ *е такъв етикет, че* $s(C_2) \vdash p$. (key, p) *е връзка "едно към много" тогава и само тогава, когато* $bad_{C_1,C_2}(key, p) = \emptyset$.

Програмен модел MapReduce

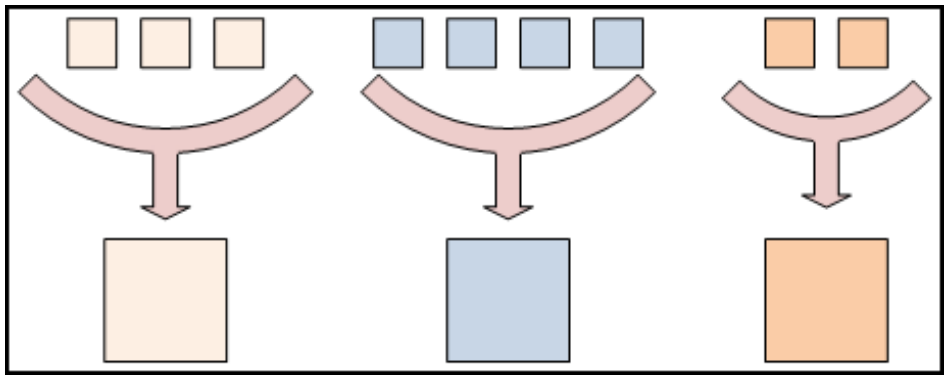
Map, Reduce, MapReduce

$map : \mathbf{Val} \rightarrow 2^{\mathbf{Prim} \times \mathbf{Val}}$

$map[C] = \bigcup \{map(d) | d \in \mathbf{C}\}$

$map[C]_k = \{key | \exists (key, v) \in map[C]\}$

$map[C] \restriction key =$
$\{v | v \in \mathbf{Val} : \exists (key, v) \in map[C]\}$



$reduce : \mathbf{Prim} \times 2^{\mathbf{Val}} \rightarrow \mathbf{Val}$

$$mr(C, map, reduce) = \{(key, reduce(key, map[C] \restriction key)) | key \in map[C]_k\}$$

$$mr_2(C_1, C_2, map_1, map_2, reduce) =$$

$$\left\{ \begin{array}{l} (key, reduce(key, map_1[C_1] \restriction key \bigcup map_2[C_2] \restriction key) \\ | key \in map_1[C_1]_k \bigcup map_2[C_2]_k \end{array} \right\}$$

Map, Reduce, MapReduce

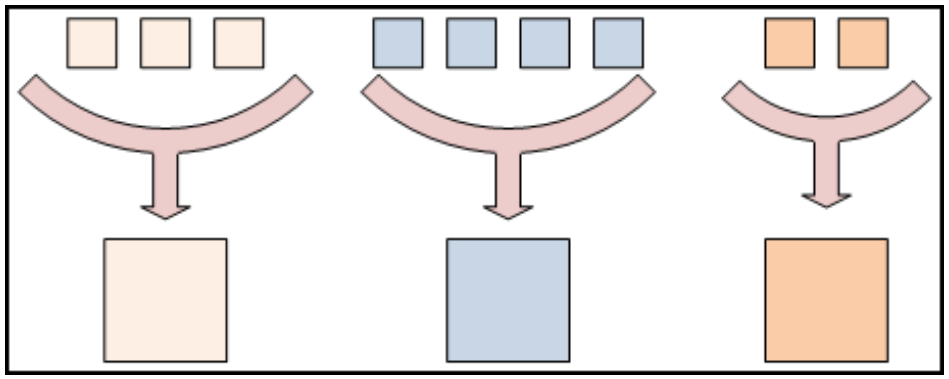
$map : \mathbf{Val} \rightarrow 2^{\mathbf{Prim} \times \mathbf{Val}}$

$map[C] = \bigcup \{map(d) | d \in \mathbf{C}\}$

$map[C]_k = \{key | \exists (key, v) \in map[C]\}$

$map[C] \restriction key =$
$\{v | v \in \mathbf{Val} : \exists (key, v) \in map[C]\}$

$reduce : \mathbf{Prim} \times 2^{\mathbf{Val}} \rightarrow \mathbf{Val}$



$$mr(C, map, reduce) = \{(key, reduce(key, map[C] \restriction key)) | key \in map[C]_k\}$$

$$mr_2(C_1, C_2, map_1, map_2, reduce) =$$

$$\left\{ \begin{array}{l} (key, reduce(key, map_1[C_1] \restriction key \bigcup map_2[C_2] \restriction key) \\ | key \in map_1[C_1]_k \bigcup map_2[C_2]_k \end{array} \right\}$$

Map, Reduce, MapReduce

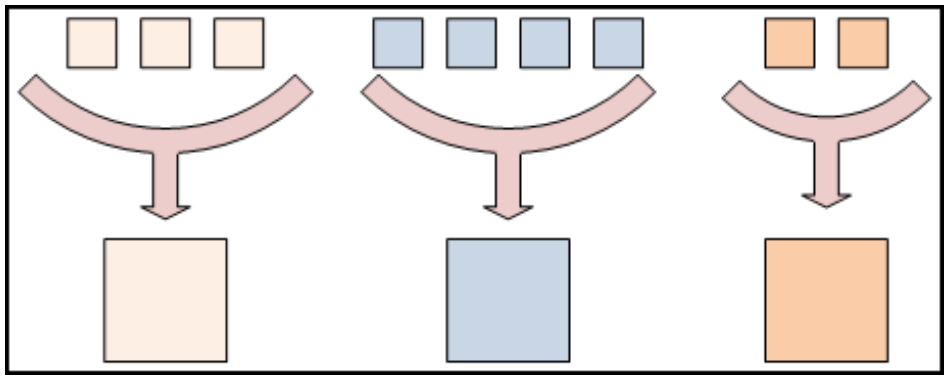
$map : \mathbf{Val} \rightarrow 2^{\mathbf{Prim} \times \mathbf{Val}}$

$map[C] = \bigcup \{map(d) | d \in \mathbf{C}\}$

$map[C]_k = \{key | \exists (key, v) \in map[C]\}$

$map[C] \restriction key =$
$\{v | v \in \mathbf{Val} : \exists (key, v) \in map[C]\}$

$reduce : \mathbf{Prim} \times 2^{\mathbf{Val}} \rightarrow \mathbf{Val}$



$$mr(C, map, reduce) = \{(key, reduce\,(key, map\,[C] \restriction key))\,|key \in map\,[C]_k\}$$

$$mr_2(C_1, C_2, map_1, map_2, reduce) =$$

$$\left\{ \begin{array}{l} (key, reduce(key, map_1[C_1] \restriction key \bigcup map_2[C_2] \restriction key) \\ |key \in map_1[C_1]_k \bigcup map_2[C_2]_k \end{array} \right\}$$

Map, Reduce, MapReduce

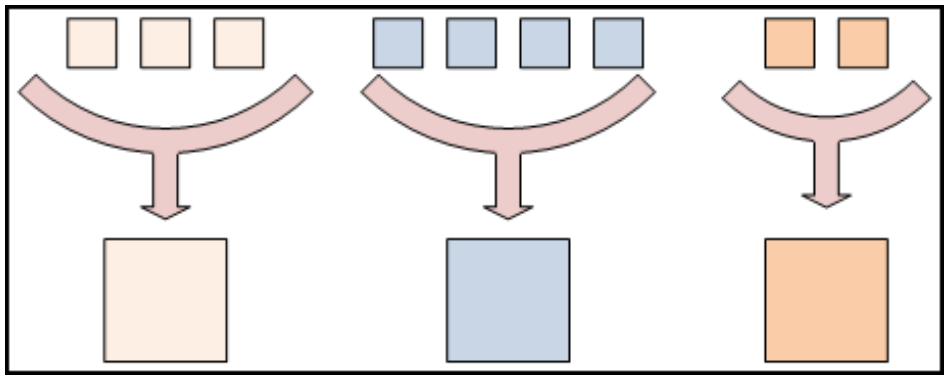
$map : \mathbf{Val} \rightarrow 2^{\mathbf{Prim} \times \mathbf{Val}}$

$map[C] = \bigcup \{map(d) | d \in \mathbf{C}\}$

$map[C]_k = \{key | \exists (key, v) \in map[C]\}$

$map[C] \restriction key =$
$\{v | v \in \mathbf{Val} : \exists (key, v) \in map[C]\}$

$reduce : \mathbf{Prim} \times 2^{\mathbf{Val}} \rightarrow \mathbf{Val}$



$$mr(C, map, reduce) = \{(key, reduce\,(key, map\,[C] \restriction key))\,|key \in map\,[C]_k\}$$

$$mr_2(C_1, C_2, map_1, map_2, reduce) =$$

$$\left\{ \begin{array}{l} (key, reduce(key, map_1[C_1] \restriction key \bigcup map_2[C_2] \restriction key) \\ |key \in map_1[C_1]_k \bigcup map_2[C_2]_k \end{array} \right\}$$

Метод за верификация на връзки и агрегатни свойства

Връзки "Едно към много"

Нека (key, p) е кандидат за връзка "едно към много".

Дефиниция ($Map^s_{base} : \mathbf{S} \to (\mathbf{Doc} \to 2^{\mathbf{Prim} \times \mathbf{Val}})$)

$$Map^s_{base}(k) = \lambda d.\{d[k] : (1, 0)\}$$

Дефиниция ($Map^s_{ref} : \mathbf{S} \to (\mathbf{Doc} \to 2^{\mathbf{Prim} \times \mathbf{Val}})$)

$$Map^s_{ref}(p) = \lambda d.\{k : (0, 1) | k \in |d[p]|\}$$

Дефиниция ($reduce_c$)

$$reduce_c(k, \{(a_1, b_1), ..., (a_k, b_k)\}) = (\sum a_i, \sum b_i)$$

Връзки "Едно към много"

$$R_1 = mr_2(C_1, C_2, Map^s_{base}(key), Map^s_{ref}(p), reduce_c)$$

Референтната цялостност относно двойката (key, p) може да се установи чрез преглеждане на съответната тройка $key : (c_{base}, c_{ref}) \in R_1$:

- $c_{base} = 0$ ще означава, че k се среща като референция в C_2, но не и като основен ключ в C_1.
- $c_{base} > 1$ ще означава, че k се среща като основен ключ в C_1 повече от веднъж.

Пълнота и коректност на метода

Връзки "Едно към много"

Твърдение (Коректност)

Нека k и n са такива, че $t = k : (0, n) \in R_1$. Тогава $k \notin range_{C_1}(key)$ и броят на всички документи d, такива че $d \in C_2$ и $k \in |d[p]|$, е точно n.

Твърдение (Пълнота)

Нека k е такъв ключ, че $k \notin range_{C_1}(key)$, а n е броят на на елементите на множеството $B = \{d \in C_2 | k \in |d[p]|\}$. Тогава, ако $n > 0$, съществува тройка $t = k : (0, n) \in R_1$.

Твърдение (Връзка между R_1 и bad)

$t = k : (0, n) \in R_1$ за някакъв ключ k и някакво число n тогава и само тогава, когато съществува поне един документ $d \in bad_{C_1, C_2}(key, p)$, такъв че $k \in |d[p]|$.

Твърденията за останалите случаи са със сходни формулировки

Елементи от практическата реализация

Множество специфични детайли

```
var emitter_references = function
                         (listFieldPath, idFieldInListPath){

  var map_function = "function (){"+

    "var list = this" + unwindString (listFieldPath) + ";" +

    "for (elementLabel in list){" +

      "emit (list[elementLabel]" +
            unwindString(idFieldInListPath) +
            ",{\"sums\":[0,1]});}}";

  return eval(map_function);

}
```

Примерна спецификация на свойство

```
var specification  = {
 "database":"socialbook",
 "many-to-one-reference":{
      "primary-collection":"PrimaryRecords",
      "primary-key":"id",
      "reference-collection":"ReferencingRecords",
      "reference-list":"authorsInformation",
      "reference-key":"id"}
}
```

Заключение

Приноси

- **Направен е изчерпателен обзор на областите синтез и анализ на компютърни програми** в глава 2 (“Обзор”).
- **Създаден/дефиниран е формален модел на нерелационните бази данни, позволяващ теоретични разглеждания на свойствата на техни представители.** Дефинициите на понятията и свойствата, свързани с проблема, както и някои леми, свързани с тях, са изложени в глава 3 (“Формален модел на проблема...”).
- **Формално е обоснован метод за синтезиране на програми, извършващи анализ за коректността на поведението на системи, генериращи данни в нерелационни СУБД.** (Глава 4 – “Формален модел на решението”).

Приноси

- **Направено е доказателство на пълнотата и коректността на метода за синтезиране на програми** в глава 5 (“Пълнота и коректност на метода”).
- **Направена е експериментална реализация на създадения метод за конкретен тип нерелационна СУБД.** В глава 6 (“Елементи от практическата реализация”) са дадени подробни сведения за практическата реализация на метода с реална СУНРДБ.

Бъдещи направления

Ако реферираният документ притежава полето X, то всеки съответен на него реферираш документ притежава полето Y и сумата на стойностите на всички полета Y е не по-голяма от стойността на полето X

Бъдещи направления

- Характеризация на проверимите свойства

Бъдещи направления

- Други приложения на въведените модели

Публикации

- Georgiev, K., *"Referential integrity and dependencies between documents in a document oriented database"*, GSTF Journal of Computing, Volume 2, Number 4, January 2013, pages 24-28
- Георгиев, К., *"Анализ и извличане на компютърни програми"*, Годишник на секция Информатика, Съюз на учените в България, Том 5, 2012, 1-31
- Georgiev, K., Trifonov, T., *"Verification of Java programs and applications of the Java Modelling Language in computer science education"*, Information Systems & Grid Technologies, Seventh International Conference, ISGT'13, приета за печат

Доклади

- Георгиев, К., Трифонов, Т., *"Автоматични средства за верификация в помощ на обучението по теоретична информатика"*, доклад на конференция по повод 50 годишнината на катедра Математическа логика и приложения, Гьолечица, 17-18 ноември 2012
- Георгиев, К., *"Анализ и извличане на обектно-ориентирани програми"*, доклад на пролетна научна сесия на ФМИ, 16 март 2013

Благодаря за вниманието!