X Міжнародна конференція
10th International Conference

ТЕОРЕТИЧНІ ТА ПРИКЛАДНІ АСПЕКТИ ПОБУДОВИ ПРОГРАМНИХ СИСТЕМ

THEORETICAL AND APPLIED ASPECTS OF PROGRAM SYSTEMS DEVELOPMENT

TAAPSD'2013

Праці конференції
Proceedings

25 травня – 2 червня 2013 року
Ялта

УДК: 004
Теоретичні та прикладні аспекти побудови програмних систем: матеріали міжнар. наук. конф., м. Ялта, 25 травня – 2 червня 2013 р. / редкол.: Д.Б. Буй та ін. – Кіровоград: ПП «Центр оперативної поліграфії «Авангард»», – 2013. – 196 с.

В збірнику зібрані праці X Міжнародної науково-практичної конференції «Теоретичні та прикладні аспекти побудови програмних систем» **TAAPSD'2013**, яка проходила в м. Ялта 25 травня – 2 червня 2013 р.

УДК: 004

Десята Міжнародна науково-практична конференція

ТЕОРЕТИЧНІ ТА ПРИКЛАДНІ АСПЕКТИ ПОБУДОВИ ПРОГРАМНИХ СИСТЕМ TAAPSD'2013

(Україна, Автономна республіка Крим, Ялта, 25 травня – 02 червня 2013 р.)

Міністерство освіти і науки України, Міністерство освіти і науки Автономної Республіки Крим, Республіканський вищий навчальний заклад «Кримський гуманітарний університет», Київський Національний університет імені Тараса Шевченка, Національний університет «Києво-Могилянська Академія», Кіровоградський державний педагогічний університет імені Володимира Винниченка, Міжнародний інститут прикладного системного аналізу (Австрія), Казахський національний технічний університет імені К.І. Сатпаєва (м. Алмати, Казахстан), Східноказахський державний технічний університет (м. Усть-Каменогорськ, Казахстан), Вільнюський державний університет (Литва) проводять **X Міжнародну науково-практичну конференцію «Теоретичні та прикладні аспекти побудови програмних систем» – TAAPSD'2013**.

Конференція відбулася в м. Ялта, Автономна республіка Крим, 25 травня – 02 червня 2013 р. на базі Інституту економіки та управління РВНЗ «Кримський гуманітарний університет» (*http://ieu.kgu.edu.ua/*).

НАУКОВІ НАПРЯМИ КОНФЕРЕНЦІЇ

- Теорія програмування
- Формальні методи розробки програмних систем
- Верифікація і тестування програмних систем
- Представлення даних
- Логічні методи обробки комп'ютерних знань
- Інтелектуальні інформаційні системи
- Захист та кодування інформації
- Системні методології
- Технології та сучасний інструментарій програмування
- Індустріальні основи виробництва програм
- Web-програмування та електронна комерція
- Розподілені інформаційно-обчислювальні системи і середовища
- Інноваційні методи і технології навчання
- Бази даних/знань
- Моделі предметних областей та математичні методи їх дослідження
- Інші (за погодженням з програмним комітетом)

ОФІЦІЙНІ РОБОЧІ МОВИ КОНФЕРЕНЦІЇ:

українська, російська, англійська

ПРОГРАМНИЙ КОМІТЕТ

– завідувач кафедри математичної інформатики, декан факультету кібернетики Київського Національного університету імені Тараса Шевченка, доктор фізико-математичних наук, член-кореспондент НАН України, заслужений діяч науки і техніки України, професор **А.В. Анісімов**, **голова;**

– академік НАН України, професор кафедри теорії і технології програмування Київського Національного університету імені Тараса Шевченка, доктор фізико-математичних наук, професор **В.Н. Редько**, **голова;**

– професор кафедри теорії і технології програмування Київського Національного університету імені Тараса Шевченка, доктор фізико-математичних наук, професор **Д.Б**. **Буй**, **заст. голови**;

– завідувач кафедри теорії і технології програмування Київського Національного університету імені Тараса Шевченка, доктор фізико-математичних наук, професор **М.С. Нікітченко, заст. голови;**

– завідувач кафедри інформатики та інформаційних технологій РВНЗ «Кримський гуманітарний університет», кандидат технічних наук, доцент **І.Ю. Грішін, заст. голови;**

В.В. Бублик	С.С. Шкільняк
М.М. Глибовець	Ш. Гудак
С.П. Горлач	С.Д. Погорілий
С.С. Гороховський	Ю.В. Бойко
А.Ю. Дорошенко	М.О. Сидоров
К.М. Лавріщева	С.Ф. Теленик
В.В. Пасічник	В.С. Харченко
В.Г. Скобелєв	Г.В. Сандраков
О.Л. Синявський	О.Б. Стеля
Х. Турулл-Торез	В.І. Кудін
К. Хаенсген	О.В. Авраменко
Б.С. Ахметов	С. Дапкунас

ОРГАНІЗАЦІЙНИЙ КОМІТЕТ

- **О.В. Глузман**, академік НАПНУ, ректор Республіканського вищого навчального закладу «Кримський гуманітарний університет», **голова орг. комітету**;
- **Р.Р. Ларіна,** д. е. н., професор, завідувач кафедри менеджменту інституту економіки та управління Республіканського вищого навчального закладу «Кримський гуманітарний університет», **заст. голови орг. комітету**;
- **Д.Б. Буй,** д. ф.-м. н., професор кафедри теорії та технології програмування факультету кібернетики Київського національного університету імені Тараса Шевченка, **заст. голови орг. комітету**;
- **С.В. Компан,** аспірант кафедри теорії та технології програмування факультету кібернетики Київського національного університету імені Тараса Шевченка, **секретар конференції**;
- **О.М. Секо,** Республіканський вищий навчальний заклад «Кримський гуманітарний університет», **секретар конференції**;
- **О.О. Костюк**, старший викладач кафедри мультимедійних систем факультету інформатики Національного університету «Києво-Могилянська академія»;
- **П.В. Четирбок,** старший викладач кафедри інформатики та інформаційних технологій інституту економіки та управління Республіканського вищого навчального закладу «Кримський гуманітарний університет».

ЗМІСТ

ПРИМЕНЕНИЕ КОМПЬЮТЕРНОЙ АЛГЕБРЫ ДЛЯ МОДЕЛИРОВАНИЯ И ОЦЕНКИ ЭНЕРГИИ ВОЛНОВЫХ ПРОЦЕССОВ В ДВУХСЛОЙНЫХ ЖИДКОСТЯХ

О.В. Авраменко[1], Ю.В. Гуртовой[2], В.В. Нарадовый[3]

Кировоградский государственный педагогический университет,
Кировоград, Украина

[1]oavramenko@rambler.ru, [2]hurtovyy@rambler.ru,
[3]naradvova1986@gmail.com

Решения нелинейных краевых задач в области гидромеханики методом многомасштабных разложений связано с громоздкими промежуточными выкладками и преобразованиями, которые крайне сложно, а то и невозможно, провести без помощи символьных вычислений.

Найфе использовал метод многомасштабных разложений для анализа эволюции волновых пакетов конечной амплитуды на поверхности контакта двух полубесконечных жидкостей с различными плотностями с учётом эффекта поверхностного натяжения [3], [4].

При моделировании волновых движений в более сложных гидродинамических системах применение метода многомасштабных разложений столкнулось с необходимостью применения пакетов компьютерной алгебры, как к действенному методу решения прикладных задач. Разработана и успешно применяется методология применения символьных вычислений для решения задач методом многомасштабных разложений. Так, исследованы задачи о распространении волновых пакетов на поверхности контакта гидродинамической системы «жидкое полупространство - жидкий слой с твёрдой крышкой» [5], [6], [12], [13] и о распространении волн в системе «слой с твёрдым дном – слой с твёрдой крышкой» [8] - [10].

В настоящее время изучается проблема волновых движений в системе «слой с твёрдым дном – слой со свободной поверхностью» [1], [2], [6], [11]. В этой системе существуют две поверхности (внутренняя поверхность контакта и свободная поверхность), вдоль которых распространяются волновые пакеты, что вносит дополнительные сложности при проведении аналитических и численных вычислений. Особый интерес вызывает оценка общей энергии волнового движения, а также вклад в неё внутренней и поверхностной составляющих.

Опишем математические постановки в рамках слабонелинейного приближения названных выше проблем, к которым был применён метод символьных вычислений.

Рассматривается распространение волновых пакетов на поверхности контакта $z=\eta(x,t)$ в двухслойных гидродинамических системах «жидкая среда в области Ω_1 - жидкая середа в области Ω_2» со следующими характеристиками: φ_j $(j=1,2)$ - потенциалы скоростей жидких сред, $\rho=\rho_2/\rho_1$ - отношение плотностей жидких сред, T - коэффициент поверхностного натяжения. В невозмущенном состоянии жидкие среды для трех различных гидродинамических систем размещаются в таких областях:

- система «слой с твердой крышкой $\Omega_2=\{(x,y,z),|\,x|<\infty,|\,y|<\infty,\,0<z<h)\}$ – полупространство $\Omega_1=\{(x,y,z),|\,x|<\infty,|\,y|<\infty,\,z<0)\}$»;
- система «слой с твердой крышкой $\Omega_2=\{(x,y,z),|\,x|<\infty,|\,y|<\infty,\,0<z<h_2)\}$ - слой с твердым дном $\Omega_1=\{(x,y,z),|\,x|<\infty,|\,y|<\infty,\,-h_1<z<0)\}$»;
- система «слой со свободной поверхностью $\Omega_2=\{(x,y,z),|\,x|<\infty,|\,y|<\infty,\,0<z<h_2)\}$ - слой с твердым дном $\Omega_1=\{(x,y,z),|\,x|<\infty,|\,y|<\infty,\,-h_1<z<0)\}$».

В рамках слабонелинейной модели потенциального движения жидкости после введения безразмерных переменных и коэффициента нелинейности α уравнения движения для каждой из выше представленных задач имеют вид

$$\nabla^2\phi_j=0 \quad \text{в } \Omega_j.$$

Краевые условия задаются в зависимости от характеристик системы в форме:

– кинематические условия на поверхности контакта $z=\alpha\eta(x,t)$

$$\eta_{,t}-\phi_{j,z}=-\phi_{j,x}\eta_{,x};$$

– кинематическое условие на поверхности контакта $z=\alpha\eta(x,t)$

$$\phi_{1,t}-\rho\phi_{2,t}+(1-\rho)\eta+0.5\alpha\left[(\nabla\phi_1)^2-\rho(\nabla\phi_2)^2\right]-$$
$$T(1+\alpha^2\eta_{,x}{}^2)^{-3/2}\eta_{,xx}=0;$$

– условие затухания в нижнем полупространстве $|\nabla\phi_1|\to 0$ при $z\to-\infty$;

– условие на твердом дне $\phi_{1,z}=0$;

– условие на твердой крышке $\phi_{2,z} = 0$;

– кинематическое условие на свободной поверхности $z = \alpha\eta_0(x,t)$

$$\eta_{0,t} - \phi_{2,z} = -\alpha\phi_{2,x}\eta_{0,x};$$

– кинематическое условие на свободной поверхности $z = \alpha\eta_0(x,t)$

$$\phi_{2,t} + \eta_0 + 0.5\alpha\left(\nabla\phi_2\right)^2 - T_0\left(1+\alpha^2\eta_{0,x}^2\right)^{-3/2}\eta_{0,xx} = 0.$$

Согласно методу многомасштабных разложений искомые функции представляются в виде степенных рядов по степеням малого параметра

$$\eta(x,t) = \sum_{n=1}^{k}\alpha^n\eta_n(x_0,...,x_{k-1},t_0,...,t_{k-1}) + O(\alpha^{k+1}),$$

$$\varphi_j(x,z,t) = \sum_{n=1}^{k}\alpha^n\varphi_{jn}(x_0,...,x_{k-1},z,t_0,...,t_{k-1}) + O(\alpha^{k+1}),$$

где $x_n = \alpha^n x$, $t_n = \alpha^n t$, $j = 1,2$. Далее формулируются линейные приближения поставленной нелинейной проблемы, которые последовательно решаются с использованием символьных вычислений в пакете Maple компании Waterloo Maple Inc. (www.maplesoft.com).

Ниже приведём результат расчета энергетических характеристик системы для первой линейной модели распространения внутренних и поверхностных волн в двухслойной жидкости, нижний слой которой ограничен твердым дном, а верхний – свободной поверхностью. Математическая постановка имеет вид

$$\begin{aligned}
&\varphi_{j1,x_0x_0} + \varphi_{j1,zz} = 0 \text{ в } \Omega_j,\\
&\eta_{1,t_0} - \varphi_{j1,z} = 0 \text{ на } z = 0,\\
&\eta_{01,t_0} - \varphi_{21,z} = 0 \text{ на } z = h_2,\\
&\varphi_{11,t_0} - \rho\varphi_{21,t_0} + (1-\rho)\eta_1 - T\eta_{1,x_0x_0} = 0 \text{ на } z = 0,\\
&\varphi_{21,t_0} + \eta_{01} - T_0\eta_{01,x_0x_0} = 0 \text{ на } z = h_2,\\
&\varphi_{11,z} = 0 \text{ на } z = -h_1.
\end{aligned} \qquad (1)$$

Детальный анализ решений задачи (1) приведен в [2], где показано, что существуют два независимых решения для двух пар частот $\pm\omega_1$ и $\pm\omega_2$. Вдоль поверхности контакта распространяется пакет,

состоящий из внутренней прогрессивной волны $\eta_1^{(1)}$ с частотой ω_1 и волна-отклик $\eta_{01}^{(2)}$ на поверхностную волну. Вдоль свободной поверхности распространяется аналогичный пакет из поверхностной прогрессивной волны $\eta_1^{(2)}$ с частотой ω_2 и волны-отклика $\eta_{01}^{(1)}$ на внутреннюю волну.

Исследована зависимость полученных выражений для энергии волнового движения в зависимости от различных геометрических параметров системы. В качестве проверки достоверности полученных результатов выбран один из предельных случаев [14], когда плотности верхнего и нижнего слоя равны, в результате чего система вырождается в однородный слой, ограниченный свободной поверхностью.

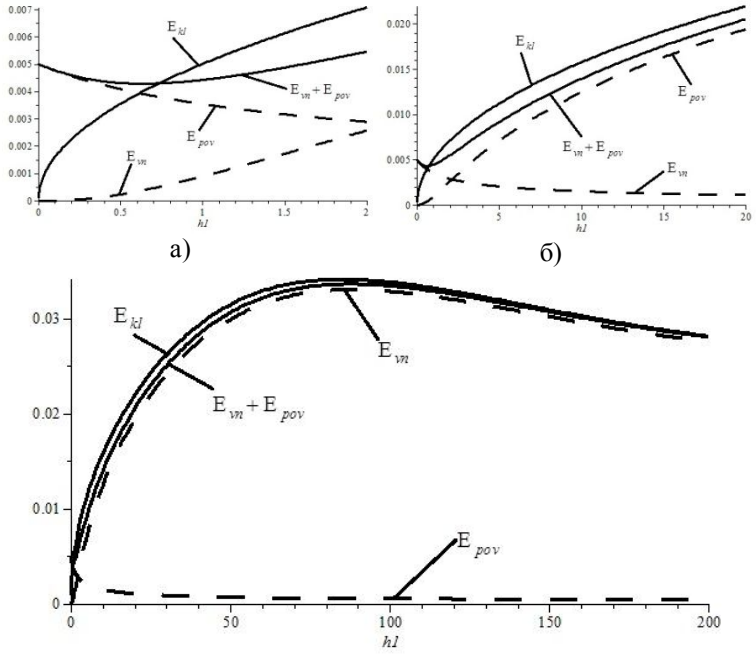


Рис.1. Зависимость энергии волнового движения от глубины нижнего слоя

На рис. 1 в разных масштабах показана зависимость от глубины нижнего слоя h_1 энергии внутренней волны E_{vn}, энергии поверхностной волны E_{pov}, общая энергия двухслойной

гидродинамической системы $E_{vn} + E_{pov}$, а также энергию для вырожденного случая E_{kl} однородного слоя конечной глубины, ограниченного свободной поверхностью.

Отметим, что полный анализ волнового движения в двухслойных гидродинамических системах предполагает большое количество громоздких аналитических преобразований. Так, вывод эволюционного уравнения в третьем и четвертом приближении связан с получением аналитических решений первых двух линейных приближений нелинейной задачи, с выводом дисперсионного соотношения первого линейного приближения и условий разрешимости второго и третьего линейного приближения нелинейной задачи. При этом особенно сложным являются процесс получения высших приближений, поскольку именно там возникает необходимость выполнять операции с функциональными матрицами и их определителями. Появляется значительное количество слагаемых, подлежащих группировкам по определенным признакам, выделяются слагаемые, содержащие множители в виде экспоненциальных функций с общим показателем. Далее следует процесс приравнивания некоторых слагаемых правой и левой части громоздких соотношений, решения системы алгебраических уравнений и другие аналитические преобразования.

Для решения проблемы выполнения математических преобразований символьные преобразования и контроля достоверности результатов реализованы программные средства в среде пакета компьютерной алгебры Maple. В процессе символьного решения производится контроль промежуточных преобразований сравнением с ранее полученными аналитическими и численными решениями, а также осуществляется проверка результатов подстановками в контрольные соотношения.

Список использованной литературы

1. Avramenko O.V., Naradovyy V.V., Selezov I.T. Multyscale modelling of the wave interaction in two-layer fluid with free surface // Збірник праць Ін-ту математики НАН України, 2010. – т.7, №2. – С. 2-9.
2. Avramenko O., Naradovyy V. Stability of wave-packets in the two-layer fluid with free surface and rigid bottom // Contemporary problems of mathematics, mechanics and computing sciences.-Harkiv, 2012. – Vol. 1.
3. Nayfeh A.H. Nonlinear propagation of wave-packets on fluid interface // Trans. ASME., Ser. E.- 1976.- 43, N4.- P. 584-588.

4. Nayfeh A.H. Second-harmonic resonance in the interaction of an air stream with capillary-gravity waves // J.Fluid Mech.- 1973.- 59.- P. 803-816.
5. Selezov I.T., Avramenko O.V. Stability of wave packets in layered hydrodynamic system subjected to the surface // International Journal of Fluid Mechanics Research.- Vol.33, №6.- 2006.- P.553-566 (1 д.а.)
6. Selezov I.T., Avramenko O.V., Gurtovyj Yu.V., Naradovyj V.V. Nonlinear interaction of internal and surface gravity waves in a two-layer fluid with free surface // J. of Math. Sciense.- 2010.- Vol.168, №4.-P. 590 -602.
7. Селезов И.Т., Авраменко О.В. Устойчивость волновых пакетов в слоистых гидродинамических системах с учетом поверхностного натяжения // Прикладна гідромеханика.- 2001.- Т.3(75), N4.- С. 38- 46.
8. Селезов И.Т., Авраменко О.В., Гуртовый Ю.В. Особенности распространения волновых пакетов в двухслойной жидкости конечной глубины // Прикладна гідромеханіка. – 2005. – Том 7(79), № 1. - С. 80-89.
9. Селезов И.Т., Авраменко О.В., Гуртовый Ю.В. Распостранение нелинейных волновых пакетов при околокритических волновых числах в двухслойной жидкой системе// Математичні методи та фізико-механічні поля. – 2007. - 50, №1. – С.91-97.
10. Селезов И.Т., Авраменко О.В., Гуртовый Ю.В. Устойчивость волновых пакетов в двухслойной гидродинамической системе //Прикладна гідромеханіка. - 2006, - 8(90), №4. - С.60-65
11. Селезов И.Т., Авраменко О.В., Нарадовый В.В. Особенности распространения слабонелинейных волн в двухслойной жидкости со свободной поверхностью // Динамические системы.- 2011.- Т.1(29), №1.-С. 53-68.
12. Селезов И.Т., Авраменко О.В. Структура нелинейных волновых пакетов на поверхности контакта жидких сред // Прикладна гідромеханика.- 2002.- Т.4(76),М.- С. 3-13.
13. Селезов И.Т., Авраменко О.В. Эволюционное уравнение третьего порядка для нелинейных волновых пакетов при околокритических волновых числах // Динамические системы.- 2001.- Вып. 17.- С. 58-67.
14. Тарапов И.Е. Механика сплошной среди. В 3 ч. Ч. 3:Механика невязкой жидкости.- Харьков: Золотые страницы, 2005.- 332 с.

МОДЕЛИРОВАНИЕ ПРОЦЕССОВ С УЧЕТОМ ФАЗОВЫХ ПЕРЕХОДОВ

[1]С.Б. Бойко, [2]Г.В. Сандраков

[1]Таврический государственный агротехнологический университет, Мелитополь, Украина,

[2]Киевский национальный университет имени Тараса Шевченко, Киев, Украина,

sandrako@mail.ru

Моделирование является важным современным инструментом для решения научных и инженерных проблем. Математическое и численное моделирование играет важную роль в понимании сути явлений и процессов, исследуемых в современной механике, физике и химии, способствуя интерпретации и даже предсказанию и открытию новых явлений. Изучение и оптимизация физико-химических и физических динамических процессов, происходящих в структурно-неоднородных веществах при больших скоростях, высоких давлениях и энергиях в области фазовых превращений, требуют достаточно точного определения значений давлений, энергий и скоростей, возникающих в различных точках этих веществ в соответствующие моменты времени. Определение значений таких величин возможно на основе математического и численного моделирования быстро протекающих нестационарных нелинейных гидродинамических процессов, учитывающего основные положения теории фазовых превращений. На данный момент в литературе практически не представлены методы исследования задач, описывающих поведение структурно-неоднородных материалов при высоких давлениях и энергиях в области фазовых превращений. Задача о построении таких моделей, учитывающих основные положения теории фазовых превращений, поставлена В.И. Юдовичем в работе [1] как одна из основных задач математической физики.

Актуальность этой задачи обусловлена интенсивным развитием разделов физики и механики, связанных с изучением процессов, происходящих при прохождении сильных ударных волн в металлах, полимерах и других твердых телах и смесях [2]. Исследование и оптимизация таких процессов необходимы для развития новых технологий, использующих методы ударного обжатия и компактификации материалов, позволяющих синтезировать новые вещества и полимеры, например, искусственные алмазы из графита. Образование новых веществ их модификаций и фаз связано с физико-химическими процессами, инициируемыми

ударными волнами при высоких давлениях и энергиях. Расчет таких волновых процессов усложняется, поскольку возникающие физические и физико-химические процессы сильно влияют на поведение инициирующих ударных волн. Кроме того, возможные фазовые переходы под действием ударного нагружения могут приводить к многофронтовым ударным волнам и к ударным волнам разгрузки [2]. Для анализа этих процессов необходимы разработка математических моделей, учитывающих возможные фазовые превращения, и построение подходящих вычислительных алгоритмов.

Один из подходов к разработке таких математических моделей представлен в [2] и основан на концепции многофазных сплошных сред, где при выводе многофазных уравнений, описывающих соответствующие модели, принимается понятие многоскоростного континуума и предположение о взаимопроникающем движении составляющих. В некотором смысле, это понятие означает одновременное присутствие нескольких материалов в каждой рассматриваемой точке пространства (дальнейшие подробности и соответствующие уравнения можно найти в [2]). Кроме того, общие многофазные уравнения, характеризующие такие многофазные сплошные среды и представленные в [2], не являются замкнутыми и содержат неопределенные слагаемые, что осложняет исследование этих уравнений и соответствующих многофазных моделей. Например, в таких уравнениях возникает проблема описания перераспределения массы и энергии между фазами в процессе развития динамики больших давлений, энергий и фазовых превращений.

Для решения этой проблемы в конкретных случаях необходимо проводить разнообразные дополнительные физические и физико-химические эксперименты, осуществление которых достаточно сложно, например, в условиях реального взрывного обжатия и нагружения материалов и веществ. Кроме того, в многофазных моделях (представленных в [2]), учитывающих фазовые превращения, предполагается, что некоторые из рассматриваемых фаз первоначально имеют нулевую плотность, которая может возрастать в некоторых точках пространства при развитии динамики ударных волн и реализации фазовых переходов. Однако, при использовании известных вычислительных алгоритмов для расчета уравнений таких многофазных моделей возникает необходимость деления на плотность при численном расчете значений соответствующих параметров. Таким образом, необходимость деления на малую или даже нулевую плотность может приводить к неустойчивости и возможной некорректности подходящего вычислительного алгоритма.

В докладе предполагается представить новый метод численного моделирования динамики смесей и структурно-неоднородных материалов в области больших деформаций и фазовых превращений, например, типа графит-алмаз. При реализации этого метода проблема описания перераспределения массы и энергии между фазами не возникает. Например, перераспределение энергии между фазами в этом методе осуществляется на основе предположения о локальном совпадении давлений в каждой из фаз. Это предположение является естественным, поскольку «при высоких давлениях свойства твердого тела приближаются к свойствам жидкостей, а для смесей жидкостей характерным является совпадение их давлений» [2]. Данный метод адаптирует и обобщает метод работы [3], где рассмотрен метод моделирования динамики структурно-неоднородной жидкости, состоящей из несущей жидкости с включениями другой жидкости (случай двухфазной сплошной среды в терминологии [2]).

Таким образом, рассматривается метод математического и численного моделирования динамики смеси или неоднородной жидкости с учетом фазовых превращений. Предполагается, что рассматриваемые жидкости являются сжимаемыми и невязкими или слабо вязкими. Неоднородности жидкости рассматриваются как малые частицы (графита) или капли одной жидкости распределенной в другой (несущей) жидкости. Общее число таких частиц может быть достаточно большим, и эти частицы могут претерпевать фазовые превращения. Динамика частиц моделируется как в методе частиц в ячейках, а несущей жидкости как в методе крупных частиц [4]. Здесь будут приведены и обобщены идеи и построения работы [3] с учетом основных положений теории фазовых превращений.

Рассматриваемый метод основан на дискретизации законов сохранения массы, моментов и энергии, которые представлены в интегральной и дифференциальной формах. Такая дискретизация является естественной, и численное расчеты реализуются как компьютерное моделирование динамики несущей жидкости, содержащей частицы, которые могут претерпевать фазовые превращения. Такой подход является адекватным физической и математической сущности рассматриваемых динамических процессов, поскольку законы сохранения остаются выполненными и на дискретном уровне в рамках принятой точности расчетов. При численной реализации рассматриваемого метода используется комбинирование метода крупных частиц и метода частиц в ячейках [4]. В этом методе используются эйлеров и лагранжев подходы одновременно, основанные на дискретизации законов сохранения массы, моментов и энергии, представленных, например, для невязких жидкостей в следующей интегральной форме

$$\int_{V(t)} \rho_t' \, d\tau = -\int_{S(t)} (\rho W) \cdot N \, ds,$$

$$\frac{d}{dt} \int_{V(t)} \rho W \, d\tau = -\int_{S(t)} p N \, ds, \qquad (1)$$

$$\frac{d}{dt} \int_{V(t)} \rho E \, d\tau = -\int_{S(t)} (pW) \cdot N \, ds,$$

где $V(t)$ и $S(t)$ обозначают объем и поверхность некоторой лагранжевой области в жидкости, N является внешней нормалью для такой области, $p = p(\rho, E)$ определяет уравнение состояния, ρ, W, E обозначают искомые неизвестные плотность, вектор скоростей и полную энергию. В случае рассмотрения трехмерной лагранжевой области, скорость представима в виде $W = (u, v, w)$.

В известном смысле [4], законы сохранения (1) являются эквивалентными законам сохранения массы, моментов и энергии, представленных в следующей дифференциальной форме

$$\frac{\partial \rho}{\partial t} + \operatorname{div}(\rho W) = 0,$$

$$\frac{\partial \rho W}{\partial t} + \operatorname{div}(\rho W \otimes W) + \nabla p = 0, \qquad (2)$$

$$\frac{\partial \rho E}{\partial t} + \operatorname{div}(\rho E W) + \operatorname{div}(pW) = 0.$$

где $W \otimes W$ обозначает тензорный квадрат вектор-функции $W = W(t, x, y, z)$.

Временная дискретизация систем уравнений (1) и (2) в рассматриваемом методе является естественной. Расчеты проводятся шаг за шагом с достаточно малым временным интервалом, начиная с заданной начальной конфигурации рассматриваемой жидкости. Пространственная дискретизация системы уравнений (2) в этом методе является более сложной и принимающей в расчет динамику движения смеси или неоднородной жидкости. Область, занимаемая такой жидкостью, разбивается на ячейки малого размера и жидкость, заполняющая каждую из таких ячеек, рассматривается как совокупность нескольких частиц. Каждой из этих частиц приписывается масса, объем, энергия и координаты, которые конкретизируются в начальный момент времени. Кроме того, плотность, скорость и полная энергия также определяются для

каждой ячейки в начальный момент. Соответствующий временной шаг моделирования движения рассматриваемой жидкости разбивается на три этапа так, чтобы выполнялись дискретные законы сохранения. Например, общая масса частиц сохраняется в процессе вычислений на каждом временном шаге при такой аппроксимации.

На первом этапе каждого временного шага вычисляются скорости частиц, обусловленные действием сил давления, что определяет эйлеров этап рассматриваемой аппроксимации, соответствующей переносу моментов и энергии в каждой ячейке, определяемому силами давления в соответствии с уравнениями моментов и энергии. Кроме того, на этом этапе учитывается возможность фазовых переходов некоторых частиц, на основе уравнений кинетики фазовых переходов. На втором этапе, принимается в расчет движение рассматриваемых частиц под действием вычисленных моментов, что соответствует лагранжеву этапу рассматриваемой аппроксимации уравнения сохранения массы. На этом этапе моделируется перенос массы из некоторой конкретной ячейки в соседние ячейки. На третьем этапе вычисляется окончательный перенос моментов и энергии, что определяет заключительный этап рассматриваемого временного шага. На этом этапе аппроксимируются уравнения моментов и энергии без учета сил давления, что моделирует перенос моментов и энергии под действием динамических сил инерции из конкретной ячейки в соседние ячейки.

Представляемый метод разработан для численного моделирования следующего физического процесса. Предположим, что частицы графита распределены равномерно в некоторой жидкости. Точнее, имеется сплошная среда с частицами графита и предполагается, что такая среда может рассматриваться при высоких давлениях как "жидкость", определяемая подходящим уравнением состояния. Рассматриваемый метод использовался для моделирования процессов прохождения сильных ударных волн в такой неоднородной среде. Это давало возможность наблюдать фазовые переходы частиц графита при проведении компьютерных экспериментов. Результаты таких экспериментов существенно зависели от плотности распределения частиц графита и интенсивности ударных волн.

Некоторые модификации рассматриваемого метода применялись также для численного моделирования динамики плазмы [5], динамики неоднородных жидкостей и структурно-неоднородных сред [3], динамики фазовых превращений [6] и являются перспективными для проведения численных расчетов параметров диффузии и абсорбции в смесях, газах и структурно-неоднородных средах.

Выводы

Будет представлен новый метод моделирования динамики смесей и структурно-неоднородных материалов в области больших деформаций и фазовых превращений. Этот метод основан на дискретизации законов сохранения, представленных в интегральной и дифференциальной формах, и уравнениях кинетики фазовых переходов. Такая дискретизация является естественной, поскольку законы сохранения остаются выполненными и на дискретном уровне в рамках принятой точности расчетов. Численные расчеты реализуются как прямое компьютерное моделирование динамики несущей жидкости, содержащей частицы, которые могут претерпевать фазовые превращения.

Список использованной литературы

1. Юдович В.И. Одиннадцать великих проблем математической гидродинамики / В.И. Юдович // Вестник молодых ученых. Серия: прикладная математика и механика. – 2003. – № 2. – С. 3–18.
2. Нигматулин Р.И. Динамика многофазных сред. Ч. 1 / Р.И. Нигматулин. – Москва: Наука, 1987. – 464 с.
3. Сандраков Г.В. Математическое моделирование динамики структурно-неоднородных жидкостей / Г.В. Сандраков, С.Б. Бойко // Журнал обчислювальної та прикладної математики. – 2011. – № 1. – С. 109 – 120.
4. Белоцерковский О.М. Метод крупных частиц в газовой динамике / О.М. Белоцерковский, Ю.М. Давыдов. – Москва: Наука, 1982. – 392 с.
5. Boyko S.B. The numerical investigation method for evaporated plasma / S.B. Boyko, V.V. Mischenko, G.V. Sandrakov // Журнал обчислювальної та прикладної математики. – 2007. – № 2. – С. 3 – 12.
6. Бойко С.Б. Математическое моделирование динамики фазовых превращений графит-алмаз / Бойко С.Б., Сандраков // Журнал обчислювальної та прикладної математики. – 2012. – № 2. – С. 24 – 46.

OPERATIONS INTERSECTION AND UNION OF CLASSES SPECIFICATIONS FOR OBJECT-ORIENTED PROGRAMMING

Dmitriy Buy[1], *Serhiy Kompan*[2]

Taras Shevchenko National University of Kyiv, Ukraine

[1]*buy@unicyb.kiev.ua*, [2]*skompan@mail.ru*

Introduction

In applying informational technologies, there is a construction problem of the so-called dependable and stable systems and infrastructures – the systems which will behave themselves stably under all, especially, critical work circumstances. Similarity of risks, and increasing actuality of their decline to an acceptable level for critical applications led to the appearance of a special term "safeware", on the analogy of the terms "hardware", "software", "firmware" etc., which combines two components: safe – secure and ware – a product, an item. This term was suggested and patented by the leading expert of NASA on the questions of infrastructural security professor N. Leveson who registered the appearance of a modern field of knowledge called safeware engineering [1]. We'll mention a fundamental statement both obvious, and elusive in its nature: it's impossible to talk about working system stability, especially of the infrastructure, if there is no formal model of its functioning which has been constructed and verified. Moreover, for the construction of a formal model, more or less complex, not a "toylike system", there should exist a mathematical apparatus with the help of which software developers create a formal model and verify it according to the source demands of a customer etc.

For the full confidence in the fact, that an informational system will be working stably (will be dependable and stable), one should single out system components, describe them formally and verify. Indeed, nowadays there is nothing instead of a "divide and rule" approach to cope with a difficulty. In fact, one of the most important components of any complex system (infrastructure) are databases. That's why there should exist an appropriate formal model. For the relational databases, such a formal model has been already constructed and explored considerably. This issue is exhaustively covered in the literature, beginning from the pioneer works by E.F. Codd (see, e.g. [2], the first textbooks [3, 4] and modern textbooks [5, 6]). We'll mention only a collection of works done by the collaborators of Taras Shevchenko National University of Kiev on the natural generalization of classical results of the databases relational approaches [7-14].

Nowadays, there are a lot of formal models of object-oriented databases (OODB) [15-20]. Each of these models elaborates OODB to a certain extent by applying a definite mathematical apparatus. The analysis of research papers dedicated to OODB has shown that authors overlook the question arising from the necessity to construct a new class specification with the two given specifications. For example, the construction of a super class with two specified classes (the interception operation of class specification), the construction of a subclass according to two super classes (the union operation of class specifications). The interception operation of class specifications is important, in our opinion, as it provides for the opportunity to construct the core of a new program with two programs which allows integrating these two programs that results in the Framework version. This paper is dedicated to the exploration of the interception and union operations of class specifications and the refining of conditions under which the interception of classes is possible.

Practical results

The authors of this paper have conducted a number of investigations in the field under research: for example, in the article [21] it has been suggested to consider an object algebraic system as a model. Formally it can be formulated like this: $< \mathrm{O}, \mathrm{K}; \Omega_{obj}; \Omega_{spec}, \leq >$, where O – a set of objects' classes, K – a set of class specification, Ω_{obj} – a set of operations over objects, Ω_{spec} – a set of operations over class specifications, and a relation $\leq \subseteq \mathrm{K} \times \mathrm{K}$ – a partial order, which clarifies inheriting. The main objective of this article is the specification of the intersection operation $\cap$ and the difference of class specifications.

Let's start with the intersection operation $\cap$. For this, we'll clarify the notion of a class: by a class we mean /it is meant a pair $K =< s, \mu >$, where s – a functional binary relation which brings to conformity an attribute with its meaning (from a universal domain D), and μ – a functional binary relation, which brings to conformity a method with its signature. Therefore [21], the relations s and μ determine class specification.

The intersection operation (of class specifications) is an operation in the form of $\cap : \mathrm{K} \times \mathrm{K} \to \mathrm{K}$, where for the values:

$< s_1, \mu_1 > \cap < s_2, \mu_2 > = < s_1 \bigcap s_2, \mu_1 \bigcap \mu_2 >$, where $\bigcap$ – a standard set-theoretical intersection.

We'll demonstrate some results as for the structure of a partially ordered set (poset) $\langle \mathrm{F}, \subseteq \rangle$, where F – a set of all the functional binary

relations (on the universal domain D), a $\subseteq$ – an ordinary set-theoretical inclusion. These results will supplement the results of the paper [22]. All undetermined notions and designations are understood in terms of this paper.

Lemma. For the arbitrary functional binary relations f and g the following equality is true: $f \cap g = (f \cap g)|(domf \cap domg)$. □

Proof. ■ Let's start with $X \stackrel{def}{=} dom\, f \cap domg$. Then we'll use generally valid properties of the set-theoretical restriction operation (a binary ratio on a set) (monotony, distributivity etc.) [19].

Firstly, we have an inclusion $dom(f \cap g) \subseteq domf \cap domg = X$; secondly, from this a chain of equalities and inequalities follows:

$f \cap g = (f \cap g)|dom(f \cap g) \subseteq (f \cap g)|X = f|X \cap g|X \subseteq f \cap g$.

Thus, $f \cap g = (f \cap g)\,|X = (f \cap g)|(domf \cap domg)$. ■

Below $\approx$ – a relation of consistency: $f \approx g \stackrel{def}{\Leftrightarrow} f|X = g|X$, where $X \stackrel{def}{=} domf \cap domg$. In [7] the main property of consistency was determined:

$f \approx g \Leftrightarrow f \cup g$ – a functional binary relation.

The following lemma's corollary forms another criterion of consistency.

Corollary (the criterion of consistency of functional binary relations). Let f, g be arbitrary functional binary relations, and $X \stackrel{def}{=} dom\, f \cap domg$. Then there will be true two equivalences: $f \approx g \Leftrightarrow dom(f \cap g) = X$, $f \not\approx g \Leftrightarrow dom(f \cap g) \subset X$. □

Proof. The proving is performed by using a lemma and inclusion $dom(f \cap g) \subseteq X$. It's important to note that the second (the first) equivalence is a formal corollary of the first one (of the second one accordingly). □

As to the structure of poset $\langle F, \subseteq \rangle$, there are two statements.

Statement 1. Poset $\langle F, \subseteq \rangle$, is a lower semilattice, at the same time $\inf\{f, g\} = f \cap g$. □

The proving results from the fact, that $\langle F, \cap \rangle$, is a commutative idempotent semigroup, and a well-known connection between such semigroups and lower semilattices (see, e.g. [23]).

More complete information about poset $\langle F, \subseteq \rangle$ will be given by the following statement.

Statement 2. (the structure of poset $\langle F, \subseteq \rangle$). The following statements are true:

1. the empty function $f_{\emptyset}$ is the smallest element («a bottom»).
2. the largest element in poset $\langle F, \subseteq \rangle$) exists if-and-only in the case when the universe D is singleton.
3. the infimum exists for any nonempty set F, though $\inf F = \bigcap_{f \in F} f$.
4. the supremum of the set F exists if-and-only in the case when the set F is restricted, though $\sup F = \bigcup_{f \in F} f$.
5. the element f is an atom only, when f is singleton.
6. poset $\langle F, \subseteq \rangle$is a relatively complete poset and a complete (upper) semilattice. □

Let's proceed to the substantial interpretation of above results.

The operation $\cap$ constructs a new class which will be basic (parental) for classes arguments. This intersection can also be empty, in this case we will get a special empty class.

As the relation $\leq$ on the specifications is introduced by component wise ($\langle s, \mu \rangle \leq \langle s', \mu' \rangle \Leftrightarrow s \subseteq s' \wedge \mu \subseteq \mu'$), that it becomes by the Cartesian product of relations $\subseteq$, in such a case all properties of the relation $\subseteq$ (statements 1, 2) shift on the relation $\leq$. The corresponding formulations are obvious and thereby are omitted.

The operation $\cup$ construct a new class which will be child (derived class) for classes arguments. The union operation (of class specifications) is an operation in the form of $\cup : K \times K \to K$, where for the values: $< s_1, \mu_1 > \cup < s_2, \mu_2 > = < s_1 \nabla s_2, \mu_1 \nabla \mu_2 >$, where ∇ is overlapping operation [24]. A union operation specification of classes clarifies multiple class inheritance. Results of union operation of class specifications were obtained in papers [21].

Results and conclusions

The model of intersection operation of class specifications has been examined. This operation has been specified as set-theoretical intersection. The specification $f \cap g$ has been interpreted as the largest total part of specifications arguments, that is the specification from which specifications-arguments can be received by inheritance (in other words, the result specification is the specification of a paternal class). The conditions of nonempty (equivalent, empty) intersection have been examined.

As for formal results, natural criteria of function consistency have been presented (corollary) which supplement the already known criteria; the structure of a partially ordered set of partial functions have been specified (statements 1, 2).

References

1. Safety of critical infrastructures: mathematical and engineering methods of analysis and ensuring / Edited by V.S. Kharchenko // N.E. Zhukovsky National Aerospace University, 2011. (in Russian)
2. Codd E.F. A relational model of data for large shared data banks . / J Communications of the ACM Vol. 13, Num. 6, June, 1970.
3. Maier D. The theory of relational databases / D. Maier. – Computer Science Press, 1983.
4. Ullman J. Database systems: The Complete Book / H. Garsia-Molina, J. D. Ullman, J. Widom – Stanford: Prentice Hall, Inc., 2002.
5. Kroenke D. M. Database processing, 11/E / D. M. Kroenke, D. Auer – Prentice Hall PTR, 2010.
6. Date C.J. An Introduction to Database Systems / C.J. Date – Addison-Wesley, 2000.
7. Buy D.B. Full image, restriction, projection, relationship compatibility / D.B. Buy, N.D. Kahuta // Theoretical and Applied Aspects of Program Systems Development: international conference, December 8-10, 2009: Abstracts. – Kyiv. – 2009. – P. 244-260. (in Ukrainian)
8. Buy D.B. The theory of multisets: bibliography, use the table in databases / D.B. Buy, J.A. Bogatiryovaa // Radio electronic and computer systems. – № 7(48). – 2010. – P. 56-62. (in Ukrainian)
9. Buy D.B Compositional semantics of recursive queries in SQL-like languages / D. B. Buy, S. A. Polyakov // Bulletin of Kyiv University. Series. phis.-math. science. – 2010. – Issue. 1. – P. 45-56. (in Ukrainian)
10. Buy D.B. Generalized table algebra, generalized tuple calculus, generalized domain calculus and theirs equivalence / D. B. Buy, I.M. Glushko // Bulletin of Kyiv University. Series. phys.-math. science. – 2011. – Issue. 1. – P. 86-95. (in Ukrainian)
11. Buy D.B. Completeness of Armstrong axioms / Buy D.B., A.V. Puzikova // Bulletin of Kyiv University. Series. phys.-math. science. – 2011. – Issue. 3. – P. 103-108. (in Ukrainian)
12. Redko V.N. Relational databases: tabular algebra and SQL-like language / V.N. Redko, J.Y. Brona, D.B. Buy, S.A. Polyakov. – Kyiv: Publishing House "AcademPeriodika", 2001. – 196 p. (in Ukrainian)
13. Buy D. Formalization of structural constraints of relationships in «entity-relationship» model / D. Buy, L. Silveystruk // Electronic Computers and Informatics'2006: international scientific conference, September 20-22, 2006, Kosice, Slovakia: proceedings. – Kosice. – 2006. – P. 96-101.
14. Buy D. Equivalence of Table Algebras of Finite (Infinite) Tables and Corresponding Relational Calculi / D. Buy, I. Glushko // Proceedings

of the Eleventh International Conference on Informatics INFORMATICS'2011, November 16-18, 2011, Rožňava, Slovakia. – P. 56-60.
15. Piskunov A.G. The formalization of the object-oriented programming paradigm [entry point]: www.realcoding.net/dn/docs/machine.pdf (in Russian)
16. Piskunov A.G. The formalization of the OOP: types, sets, classes [entry point]: agp1.hx0.ru/articles/typeSetsClasses.pdf (in Russian)
17. Chaplanova E.B. Operating specification of object-relational data model / E.B. Chaplanova // Radioelektronika, informatika, upravlinnya. – 2012. – №12. – P. 75-79. (in Russian)
18. Karel Richta, David Toth. Formal Models of Object-Oriented Databases. In Objekty 2008. Žilina: Žilinská univerzita v Žiline, Fakulta riadenia a informatiky, 2008, p. 204-217. ISBN 978-80-8070-927-3 [entry point]: http://www.ksi.mff.cuni.cz/~richta/publications/richta-toth-Objekty2008.pdf
19. Manojit Sarkar, Steven P. Reiss A Data Model and A Query Language for Object-Oriented Database. Island, Department of Computer Science Brown University Providence, Rhode, CS-92-57, December 1992 [entry point]: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.34.4531&rep=rep1&type=pdf
20. Gail M. Shaw, Stanley B. Zdonik A Query Algebra for Object-Oriented Databases. Island, Department of Computer Science Brown University Providence, Rhode, CS-89-19, March 1989 [entry point]: http://trac.common-lisp.net/elephant/raw-attachment/wiki/RelationalAlgebra/shaw89query.2.pdf
21. Buy D.B. Refinement of the multiple inheritance in the form of the overlap operation / D. Buy, S. Kompan // Visnik of Taras Shevchenko National University of Kyiv – 2012. – Issue. 4. – P. 111-119 (in Ukrainen)
22. Buy D.B. Properties related confinality and order a set of partial functions / D.B. Buy, N.D. Kahuta // Bulletin of Kyiv University. Series. phiz.-math. science. – 2006. – Issue. 2. – P. 125-135. (in Ukrainian)
23. Skornyakov L.A. Elements of the theory of structures / L.A. Skornyakov. – M.: "Nauka", 1982. – 158 p. (in Russian)
24. Buy D. Properties of set-theoretic constructions of full image and restriction / D. Buy, N. Kahuta // Visnik of Taras Shevchenko National University of Kiyv – 2005. – Issue. 2. – P. 232-240. (in Ukrainian)

ФОРМАЛЬНОЕ ОПРЕДЕЛЕНИЕ МОДЕЛИ ДАННЫХ, ИСПОЛЬЗУЕМОЙ В NoSQL БАЗАХ ДАННЫХ

Д.Б. Буй[1], С.А. Поляков[2], Ю.А. Гришко[3]

Киевский национальный университет имени Тараса Шевченко, Киев, Украина

[1]buy@unicyb.kiev.ua, [2]sergey.a.polyakov@gmail.com, [3]jul.grishko@gmail.com

Существующие NoSQL СУБД базируются на нескольких моделях данных. Можно говорить скорее о новой идеологии построения СУБД, альтернативной реляционной, чем о единой платформе, лежащей в основе NoSQL баз данных [0]. Одним из наиболее распространенных типов NoSQL СУБД являются документно-ориентированные системы, подобные MongoDB [2] и CouchDB, и основанные на формате представления документов JSON. Ниже будут рассмотрены формальные модели, описывающие структуры данных, используемые документно-ориентированных NoSQL СУБД, и изучены некоторые свойства данных моделей.

Построение ведется на основе композиционного подхода к программированию [3]. В свое время он был успешно применен при описании семантики реляционных баз данных и языка SQL [4]. Построенные модели использует в своей основе множества, мультимножества и именные множества [5].

Множество всех конечных подмножеств X будем обозначать как $2^X_{fin} = \{X' | X' \subseteq X \;\&X' - \text{конечно}\}$.

Пусть дано множество атомарных данных D и множество имен V. Именные множества, которые обозначим как D^V, определяются как конечные отображения из множества имен V во множество данных D, т. е $.D^V = \{A | A: V' \to D, V' \in 2^V_{fin}\}$.

Множество записей RC, и документов DC, строится индуктивно. А именно, множество записей ранга 0 совпадает с D^V. Обозначим это множество через RC^0. Множество документов ранга 0, обозначаемое как DC^0, есть множество всех конечных подмножеств RC^0, т.е. $DC^0 = 2^{RC^0}_{fin}$. Пусть заданы записи и документы ранга i. Тогда записи ранга i+1 задаются как $RC^{i+1} = (D \bigcup_{j=0}^{i} DC^j)^V$. Т.е. значением имени может быть либо атомарное данное, либо документ одного из предыдущих рангов. Соответственно документ ранга i+1 задается как множество всех конечных подмножеств записей ранга i+1, т.е. $DC^{i+1} = 2^{RC^{i+1}}_{fin}$.

Так как $DC^0 \subset DC^1 \subset DC^2 \subset \cdots$ монотонно возрастающая последовательность по построению, то она имеет предел $DC = \lim_{i\to\infty} DC^i = \bigcup_{i=0}^{\infty} DC^i$. Аналогично для записей $RC = \lim_{i\to\infty} RC^i = \bigcup_{i=0}^{\infty} RC^i$.

С учетом монотонности определение записей i+1-го ранга можно переписать как $RC^{i+1} = (D \cup DC^i)^V$.

По аналогии с отношением включения $\subseteq$ для множеств, для документов и записей введем отношения быть поддокументом и быть подзаписью, которые учитывают внутреннюю структуру документов и записей. Обозначим эти отношения *sdoc* (subdocument) и *srec* (subrecord) соответственно. Отношения вводятся индуктивно. Пусть $r_1, r_2 \in RC^0$. Тогда $r_1\, srec\, r_2$ тогда и только тогда, когда $r_1 \subseteq r_2$. Аналогично, для документов нулевого ранга, $d_1\, sdoc\, d_2$ тогда и только тогда, когда $\forall r_1 \in d_1 \exists r_2 \in d_2 (r_1 \subseteq r_2)$.

Пусть эти отношения определены для документов и записей i-го ранга. Тогда, для записей i+1-го ранга отношение $r_1^{i+1}\, srec\, r_2^{i+1}$, означает, что множество имен записи r_1^{i+1} включается во множество имен записи r_2^{i+1}. А значения при одинаковых именах одновременно либо являются атомарными данными и совпадают между собой, либо являются документами. Причем, тогда их ранг меньше i+1, и документ из r_1^{i+1} является поддокументом соответствующего документа из r_2^{i+1}.

d_1^{i+1} является поддокументом d_2^{i+1} тогда и только тогда, когда для любой записи $r_1 \in d_1^{i+1}$, существует запись $r_2 \in d_2^{i+1}$, для которых $r_1 srec\, r_2$. Отметим, что для r_1, вообще говоря, может существовать несколько соответствующих записей в документе d_2^{i+1}.

Введенные таким образом отношения *srec* и *sdoc* являются отношениями предпорядка. Т.е. они рефлексивны, транзитивны, но не антисимметричны.

Докажем рефлексивность. Доказательство проводится индукцией по рангу записей и документов.

База индукции. Для записей и документов нулевого ранга утверждение очевидно.

Шаг индукции. Пусть утверждение выполняется для записей и документов i-го ранга. Тогда покажем, что для записей i+1-го ранга рефлексивность также имеет место, т.е. $r^{i+1}\, srec\, r^{i+1}$. Действительно:

- множество имен включается в само себя;
- если значением имени является атомарное данное, то оно совпадает само с собой;
- если значением имени является документ, то, по построению, его ранг меньше i+1, следовательно, по предположению индукции, он является поддокументом самого себя.

Для документов i+1-го ранга рефлексивность также имеет место, так как каждая запись i+1-го ранга, как показано выше, является подзаписью самой себя.

Докажем теперь транзитивность. А именно, что имеют место формулы $r_1\ srec\ r_2\ \&\ r_2\ srec\ r_3\ \Longrightarrow\ r_1\ srec\ r_3$ и $d_1\ sdoc\ d_2\ \&\ d_2\ sdoc\ d_3\ \Longrightarrow\ d_1\ sdoc\ d_3$. Доказательство будем проводить индукцией по рангу документов и записей.

База индукции. Для документов и записей нулевого ранга транзитивность имеет место, так как отношения быть поддокументом и быть подзаписью для документов и записей, в этом случае, совпадают с отношением быть подмножеством для множеств.

Шаг индукции. Пусть транзитивность выполняется для документов и записей i-го ранга. Покажем, что тогда она будет иметь место для документов и записей i+1-го ранга.

Сначала рассмотрим записи. Пусть $r_1^{i+1}\ srec\ r_2^{i+1}$ и $r_2^{i+1}\ srec\ r_3^{i+1}$.

Так как, по определению, множество имен записи r_1^{i+1} включается во множество имен записи r_2^{i+1}, и множество имен записи r_2^{i+1} включается во множество имен записи r_3^{i+1}, то множество имен записи r_1^{i+1} включается во множество имен записи r_3^{i+1}.

Пусть значением имени n в записи r_1^{i+1} является атомарное данное. Это означает, по определению, что имя n принадлежит множеству имен записи r_2^{i+1} и его значением в этой записи выступает то же самое атомарное данное. Но тогда это имя должно принадлежать множеству имен записи r_3^{i+1} и иметь в ней такое же значение. Следовательно, если некоторое имя в записи r_1^{i+1} имеет атомарное значение, то оно имеет такое же значение в записи r_3^{i+1}.

Пусть значением имени n в записи r_1^{i+1} является некий документ d_1. Тогда значением имени n в записи r_2^{i+1} является, вообще говоря, другой документ d_2, и значением имени n в записи r_3^{i+1} также является документ d_3, причем имеет место $d_1\ sdoc\ d_2$ и $d_2\ sdoc\ d_3$. По построению, ранг документов d_1, d_2, d_3 строго меньше i+1. Следовательно, по предположению индукции, $d_1\ sdoc\ d_3$.

Теперь перейдем к документам i+1-го ранга.

Пусть $d_1^{i+1}\ sdoc\ d_2^{i+1}$ и $d_2^{i+1}\ sdoc\ d_3^{i+1}$. По определению, имеем $\forall r_1 \in d_1^{i+1} \exists r_2 \in d_2^{i+1} (r_1\ srec\ r_2)$ и $\forall r_2 \in d_2^{i+1} \exists r_3 \in d_3^{i+1} (r_2\ srec\ r_3)$. Отсюда получаем $\forall r_1 \in d_1^{i+1} \exists r_2 \in d_2^{i+1} \exists r_3 \in d_3^{i+1}$ $(r_1\ srec\ r_2 \& r_2\ srec\ r_3)$. Следовательно, $r_1\ srec\ r_3$.

В то же время антисимметричность для этих отношений не имеет место. Действительно, рассмотрим следующий пример. Пусть

$d_1 = \{\{(a,1),(b,2)\},\{(a,1)\}\}$, а $d_2 = \{\{(a,1),(b,2)\}\}$. Тогда имеем $d_1 sdoc\ d_2$ и $d_2 sdoc\ d_1$, но очевидно, что $d_1 \neq d_2$.

В реальных документно-ориентированных СУБД записи могут дублироваться. Ситуация аналогичная таблицам в реляционных СУБД, где допускается дублирование строк таблицы. Поэтому дальнейшее уточнение построенной модели данных необходимо проводить с использованием мультимножеств. А именно, рассмотрим возможность повторения записей в пределах документа.

Сначала введем некоторые определения теории мультимножеств, необходимые для построения данной модели [6].

Мультимножество α с основой U – это функция $\alpha: U \to N^+$, где U – некоторое множество, а N^+ – множество натуральных чисел без нуля. Запись M_U будет использоваться для обозначения всех конечных мультимножеств с основой U.

Теперь определим множества записей RC_M и документов DC_M для новой модели. Определения будут строиться индуктивно. Множество записей ранга 0 совпадает с совокупностью именных множеств, т.е. $RC_M^0 = D^V$. Множество документов ранга 0 определяется как совокупность всех конечных мультимножеств, основами которых выступают конечные множества записей ранга 0, т. е. $DC_M^0 = \bigcup_{d \in 2_{fin}^{RC_M^0}} M_d$. Пусть заданы записи и документы ранга i. Тогда записи ранга i+1 задаются, как и предыдущем случае $RC_M^{i+1} = \left(D \bigcup_{j=0}^{i} DC_M^j\right)^V$. Значением имени может быть либо атомарное данное, либо документ одного из предыдущих рангов. Соответственно, документы ранга i+1 задаются как конечные мультимножества, основами которых выступают конечные множества записей ранга i+1, т.е. $DC_M^{i+1} = \bigcup_{d \in 2_{fin}^{RC_M^{i+1}}} M_d$.

Теперь переопределим отношения быть поддокументом (обозначим его как $sdoc_M$) и быть подзаписью (обозначим его как $srec_M$). Отношения вводятся индуктивно.

Для записей нулевого ранга определение не меняется, т.е. $r_1\ srec_M\ r_2$ тогда и только тогда, когда $r_1 \subseteq r_2$. Для документов нулевого ранга $d_1\ sdoc_M\ d_2$ тогда и только тогда, когда $\forall r_1 \in U_{d_1} \exists r_2 \in U_{d_2}(r_1 \subseteq r_2)$, где U_{d_1} и U_{d_2} основы документов d_1 и d_2 соответственно.

Пусть эти отношения определены для документов и записей i-го ранга. Тогда, для записей i+1-го ранга отношение $r_1^{i+1}\ srec_M\ r_2^{i+1}$, определяется, так же как и выше, за исключением того, что если значениями совпадающих имен являются документы, то они находятся в отношении $sdoc_M$, а не $sdoc$.

Для документов i+1-го ранга d_1^{i+1} является поддокументом d_2^{i+1} тогда и только тогда, когда для любой записи $r_1 \in U_{d_1^{i+1}}$, существует запись $r_2 \in U_{d_2^{i+1}}$, для которой $r_1 srec_M\, r_2$. При таком определении, количество экземпляров записей не учитываться.

Введенные таким образом отношения $sdoc_M, srec_M$ также являются отношениями предпорядка. Доказательство проводится аналогично доказательству для случая множеств.

Таким образом, построена еще одна модель данных, которая уточняет структуры данных, используемы в документно-ориентированных базах данных. Модель основывается на понятие мультимножество, и допускает дубликаты записей в пределах одного документа.

В работе построены модели данных, уточняющие структуры данных, используемые в документно-ориентированных базах данных.

Введенные отношения поддокумента и подзаписи расширяют стандартное отношение подмножества (подмультимножества) на документы с учетом их внутренней структуры. В то же время, эти отношения являются предпорядком, что не позволяет говорить о полной аналогии с отношением «быть подмножеством» («быть подмультимножеством»).

Список использованной литературы

1. Sadalage J. Pramod. NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence / Pramod J. Sadalage, Martin Fowler // Addison-Wesley Professional, 2012. – 192 p.
2. Chodorow Kristina. MongoDB: The Definitive Guide / Kristina Chodorow, Michael Dirolf // O'Reilly Media, 2010. – 216 p.
3. Редько В.Н. Основания программологии / В.Н. Редько // Кибернетика и системный анализ. – 2000. – № 1. – С. 35-57.
4. Редько В.Н. Реляційні бази даних: табличні алгебри та SQL-подібні мови / В.Н. Редько, Ю.Й. Брона, Д.Б. Буй, С.А. Поляков // Київ: Видавничий дім «Академперіодика», 2001. – 196 с.
5. Буй Д.Б. Про різні способи задання іменних даних / Д.Б. Буй, Ю.Й. Брона // Вісник Київського університету. Сер. фіз.-мат. науки. – 1994. – С. 186-194.
6. Буй Д.Б. Мультимножини: означення, операції, основні властивості / Д.Б. Буй, Ю.А. Богатирьова // Матеріали 5-ї Міжнародної конференції TAAPSD'2008. – Київ. – С. 23-26.

ТЕОРІЯ НОРМАЛІЗАЦІЇ ТАБЛИЧНИХ БАЗ ДАНИХ: 2-3НФ, НФБК

Д.Б. Буй, А.В. Пузікова

Київський національний університет імені Тараса Шевченка, Київ, Україна

buy@unicyb.kiev.ua, anna_inf@mail.ru

Метою роботи є: визначити потенційні ключі у випадках, коли значення потужностей реляційної схеми і домена знаходяться у межах $|R| \le 2$ або $|D| \le 1$ та встановити їх кількість; перевірити виконання умов 2-3НФ і НФБК у зазначених випадках.

Всі неозначувані тут поняття та позначення використовуються в розумінні монографії [1]; зокрема, $t_\varnothing$ позначає порожню таблицю, яка не містить рядків і якій можна приписати довільну схему, t_ε позначає таблицю порожньої схеми Ø, яка містить тільки рядок ε також порожньої схеми: $t_\varepsilon \overset{def}{=} \{\varepsilon\}$ [1].

Нехай t – таблиця, A, B, C – атрибути таблиці t, R – схема (довільна скінченна множина атрибутів), X, Y, Z – підмножини схеми R, D – універсальний домен, F – множина функціональних залежностей (ФЗ).

Зафіксуємо множини R і F.

Означення 1. Підмножина $K \subseteq R$ називається *потенційним ключем* (відносно множини F), якщо виконується ФЗ $K \to R$ і ніяка власна підмножина $K' \subset K$ не володіє цією властивістю [2, 3, 4], тобто:

$$K \text{ – потенційний ключ} \overset{def}{\Leftrightarrow}$$
$$K \to R \in [F] \,\&\, \forall K' \subset K \Rightarrow K' \to R \notin [F].$$

Кожна таблиця t схеми R має хоча б один потенційний ключ, зокрема, $K = R$, оскільки ФЗ $R \to R$ є тривіальною.

Випадки, в яких кількість потенційних ключів не залежить від задання множини ФЗ F, вказані у табл. 1. Дана таблиця містить три рядки та чотири стовпчики. Комірки будемо позначати парами, перша компонента – номер рядка, друга – номер стовпчика. В комірці вказана кількість потенційних ключів за вказаними припущеннями (там, де можна визначити).

Таблиця 1 – Кількість потенційних ключів при різних значеннях потужностей множин R та D.

$\lvert D\rvert$ \ $\lvert R\rvert$	$\lvert R\rvert=0$	$\lvert R\rvert=1$	$\lvert R\rvert=2$	$\lvert R\rvert\geq 3$
1	2	3	4	
$\lvert D\rvert=0$	1	1	1	1
$\lvert D\rvert=1$	1	1	1	1
$\lvert D\rvert\geq 2$	1	1	1 або 2	?

Твердження 1. Заповнення табл. 1 коректне. □

Доведення.

а) $\lvert R\rvert=0$ (випадки (1,1), (2,1), (3,1)); тоді, незалежно від потужності домена, $t_{\varnothing}$, t_{ε} – всі можливі таблиці в інтерпретації. Єдиною можливою ФЗ є $\varnothing\to\varnothing$ [5], яка є тривіальною, звідси $\varnothing$ – потенційний ключ;

б) $\lvert R\rvert\geq 1$, $\lvert D\rvert\leq 1$ (випадки (1,2), (1,3), (1,4), (2,2), (2,3), (2,4)); оскільки ФЗ $\varnothing\to R$ виконується на порожній та однорядковій таблицях [5] і для $\{A\}\subseteq R$, $\varnothing\subset\{A\}$, то єдиним потенційним ключем, згідно означення 1, є порожня множина $\varnothing$;

в) $\lvert R\rvert=1$, $\lvert D\rvert\geq 2$ (випадок (3,2)); нехай для визначеності $R=\{A\}$. Оскільки ФЗ $\varnothing\to\{A\}$ – не виконується, то таблиця має єдиний потенційний ключ: $\{A\}$ (ФЗ $\{A\}\to\{A\}$ є тривіальною);

г) $\lvert R\rvert=2$ і $\lvert D\rvert\geq 2$ (випадок (3,3)); нехай для визначеності $R=\{A,B\}$ і $D=\{d_1,d_2\}$, де d_1,d_2 – різні. Оскільки ФЗ $\varnothing\to R$ не виконується на довільній дворядковій таблиці, то множина потенційних ключів може мати один ключ: $\{A\}$ або $\{A,B\}$, чи два: $\{\{A\},\{B\}\}$ (це залежить від конкретно заданої множини ФЗ);

д) $\lvert R\rvert\geq 3$, $\lvert D\rvert\geq 2$ (випадок (3,4); множина потенційних ключів залежить від конкретно заданої множини ФЗ. □

Означення 2. Атрибут $A\in R$ називається *первинним*, якщо A міститься в якому-небудь потенційному ключі K [3, 4]:

$$A\ \text{– первинний} \overset{def}{\Leftrightarrow} \exists K\ (K\ \text{– потенційний ключ}\ \&\ A\in K).$$

Означення 3. Атрибут $A \in R$ називається *непервинним*, якщо A не міститься в жодному з потенційних ключів K_i [3, 4]:

$$A \text{ – непервинний} \overset{def}{\Leftrightarrow} \forall K(K \text{ – потенційний ключ} \Rightarrow A \notin K).$$

Можна дати інше означення непервинного атрибута: нехай $K_1,...,K_n$ – потенційні ключі, тоді A – *непервинний* $\overset{def}{\Leftrightarrow}$ $A \in R \setminus \bigcup_{i=1}^{n} K_i$.

Означення 4. Повною функціональною залежністю між множинами атрибутів X, Y відносно множини ФЗ F, називається така ФЗ $X \to Y$, що множина Y залежить від всієї множини X, а не від будь-яких її власних підмножин [3, 4]:

$$X \to Y \text{ – повна} \overset{def}{\Leftrightarrow}$$
$$X \to Y \in [F] \,\&\, \forall X' \subset X \Rightarrow X' \to Y \notin [F].$$

З означення випливає, що ФЗ виду $\{A\} \to Y$ повна, а тривіальна ФЗ залежність $X \to Y$, де $Y \subset X$ – неповна.

Означення 5. Друга нормальна форма (2НФ) таблиці $t(R)$ – це або дана таблиця, якщо вона знаходиться у 1НФ і кожний непервинний атрибут характеризується повною ФЗ від кожного потенційного ключа:

$$t(R) \text{ знаходиться у 2НФ} \overset{def}{\Leftrightarrow} \forall A \text{ – непервинний атрибут,}$$
$$\forall K \text{ – потенційний ключ } (\forall K' \subset K \Rightarrow K' \to \{A\} \notin [F]),$$

або набір її проекцій $\pi_{R_1}(t), \pi_{R_2}(t),..., \pi_{R_n}(t)$, де $R = R_1 \cup R_2 \cup ... \cup R_n$, кожна з яких задовольняє вказаним властивостям, причому декомпозиція таблиці $t(R)$ є декомпозицією без втрат: $t = \pi_{R_1}(t) \otimes \pi_{R_2}(t) \otimes ... \otimes \pi_{R_n}(t)$ [4].□

Лема 1. Таблиця знаходиться у 2НФ якщо $|R| \le 2$ або $|D| \le 1$. □

Доведення. Враховуючи результати з табл. 1 для умов $|R| \le 2$ або $|D| \le 1$ розглянемо випадки:

а) $|R|=0$ або $|D|\le 1$: (1,1), (1,2), (1,3), (1,4), (2,1), (2,2), (2,3), (2,4), (3,1); потенційним ключем є $\varnothing$, яка не має власної підмножини, отже за означенням 5 таблиця знаходяться у 2НФ;

б) $|D|\ge 2$ і $|R|=1$: (3,2); потенційним ключем є одноелементна множина (ФЗ $\varnothing \to R$ не виконується), схема не містить непервинних атрибутів, отже, таблиця знаходиться у 2НФ;

в) $|D|\ge 2$ і $|R|=2$: (3,3); якщо кількість непервинних атрибутів дорівнює 1 (ключем є одноелементна множина $\{A\}$), то з урахуванням факту, що ФЗ $\varnothing \to R$ не виконується, таблиця знаходиться у 2НФ; якщо схема не містить непервинних атрибутів (потенційними ключами є множини $\{A,B\}$ або $\{\{A\},\{B\}\}$), то умова 2НФ для таблиці t не порушується. □

Лема 2. Таблиця знаходиться у 2НФ якщо виконується одна з наступних умов:

1. $\forall K\,(K$ – потенційний ключ $\&|\,K|=1)$ [6];
2. для потенційних ключів $K_1,\ldots,K_n$ виконується $\bigcup_{i=1}^{n} K_i = R$ [6]. □

Доведення. 1. Доведення випливає безпосередньо з означення 2НФ з урахуванням того факту, що ФЗ $\varnothing \to R$ не виконується на довільній таблиці, домен якої містить два або більше елементи. ▫

2. Оскільки за умови $\bigcup_{i=1}^{n} K_i = R$, де $K_1,\ldots,K_n$ – потенційні ключі, схема не містить непервинних атрибутів, то умова 2НФ для таблиці t не порушується. Зауважимо, що дана властивість виконується і для граничних випадків: $|R|=0$, $|R|=1$ і $|R|=2$.▫□

Означення 6. Атрибут $A \in R$ називається *транзитивно залежним* від множини атрибутів $X \subseteq R$, якщо існує така множина атрибутів $Y \subseteq R$, що $(X \to Y)(t) = True$ $\&(Y \to X)(t) = False$ $\&(Y \to \{A\})(t) = True$, за умови $A \notin X \bigcup Y$ [2].

Синтаксично в термінах [1] запишемо:

$$A \text{ транзитивно залежний від } X \subseteq R \overset{def}{\Leftrightarrow} \exists Y \subseteq R$$
$$(X \to Y \in [F] \ \& Y \to X \notin [F] \ \& Y \to \{A\} \in [F]$$
$$\& A \in R \setminus (X \bigcup Y).$$

Означення 7. Третя нормальна форма (3НФ) таблиці $t(R)$ – це або дана таблиця, якщо вона знаходиться у 1НФ і не містить транзитивних залежностей непервинних атрибутів від потенційних ключів:

$$t(R) \text{ знаходиться у 3НФ} \overset{def}{\Leftrightarrow} \forall X \subset R,$$
$$A \in R \setminus (\bigcup_i K_i \cup X) \ (X \to \{A\} \in [F] \Rightarrow X \to R \in [F]),$$

або набір її проекцій $\pi_{R_1}(t), \pi_{R_2}(t), ..., \pi_{R_n}(t)$, де $R = R_1 \bigcup R_2 \bigcup ... \bigcup R_n$, кожна з яких задовольняє вказаним властивостям, причому декомпозиція таблиці $t(R)$ є декомпозицією без втрат: $t = \pi_{R_1}(t) \otimes \pi_{R_2}(t) \otimes ... \otimes \pi_{R_n}(t)$ [2, 4].□

Лема 3. Таблиця знаходиться у 3НФ якщо $|R| \le 2$ або $|D| \le 1$. □

Доведення. Враховуючи результати з табл. 1 для умов $|R| \le 2$ або $|D| \le 1$ розглянемо випадки:

а) $|R| = 0$ (випадки (1,1), (2,1), (3,1)); оскільки схема R не має непервинних атрибутів, умови 3НФ виконуються тривіально;

б) $|R| = 1$, $|D| \le 1$ (випадки (1,2), (2,2)); схема $R = \{A\}$ має єдиний непервинний атрибут A і потенційним ключем є $\varnothing$, умови 3НФ виконуються тривіально;

в) $|R| = 1$, $|D| \ge 2$ (випадок (3,2)); схема $R = \{A\}$ не має непервинних атрибутів ($\{A\}$ є потенційним ключем), умови 3НФ виконуються тривіально;

в) $|R| \ge 2$, $|D| \le 1$ (випадки (1,3), (1,4), (2,3), (2,4)); нехай, для означеності $A, B \in R$, тоді виконується ФЗ $\{A\} \to \{B\}$ (врахували, що на порожній та однорядковій таблицях виконується довільна ФЗ [5]), але атрибут B не є транзитивно залежним від потенційного ключа $\varnothing$, оскільки виконується ФЗ $A \to \varnothing$. Умови 3НФ виконуються;

г) $|R|=2$, і $|D|\geq 2$ (випадок (3,3)); якщо схема має один непервинний атрибут B (ключем є одноелементна множина $\{A\}$), то врахуємо, що ФЗ $\{A\}\to\varnothing$, $\varnothing\to\{B\}$ не виконуються, таблиця знаходиться у 3НФ; якщо схема не має непервинних атрибутів, то умови 3НФ виконуються тривіально;

д) $|R|\geq 3$, $|D|\geq 2$ (випадок (3,4)); існування 3НФ залежить від конкретно заданої множини ФЗ. □

Лема 4. Таблиця знаходиться у 3НФ, якщо для потенційних ключів $K_1,...,K_n$ виконується $\bigcup_{i=1}^{n} K_i = R$. □

Доведення. Оскільки за умови $\bigcup_{i=1}^{n} K_i = R$, де $K_1,...,K_n$ – потенційні ключі, схема не містить непервинних атрибутів, то умова 3НФ для таблиці t виконується тривіально. ▫□

Лема 5. Якщо таблиця знаходиться у 3НФ, то вона знаходиться у 2НФ: $3НФ \Rightarrow 2НФ$ [2].□

Доведення проводиться від супротивного (використана ідея доведення [2]). Припустимо таблиця $t(R)$ знаходиться у 3НФ і не знаходиться у 2НФ, тобто існує така власна підмножина потенційного ключа $K'\subset K$, що для деякого непервинного атрибута A виконується ФЗ $K'\to\{A\}$. Але тоді атрибут A транзитивно залежить від потенційного ключа K, оскільки виконується ФЗ $K\to K'$. Звідси випливає, що таблиця $t(R)$ не знаходиться у 3НФ, що суперечить припущенню.□

Означення 8. Множина атрибутів, яка містить потенційний ключ, називається *суперключем*:

$$W - \textit{суперключ} \overset{def}{\Leftrightarrow} \exists K(W\supseteq K \,\&\, K - \textit{потенційний ключ}) \text{ [3].}$$

Означення 9. Нормальна форма Бойса-Кодда (НФБК) таблиці $t(R)$ – це або дана таблиця, якщо вона знаходиться у 1НФ і детермінант кожної її нетривіальної ФЗ є суперключем:

$$t(R) \textit{ знаходиться у НФБК} \overset{def}{\Leftrightarrow}$$
$$\forall X\subseteq R, \forall A\in R\setminus X(X\to\{A\}\in[F]\Rightarrow X\to R\in[F]),$$

або набір її проекцій $\pi_{R_1}(t), \pi_{R_2}(t), ..., \pi_{R_n}(t)$, де $R = R_1 \bigcup R_2 \bigcup ... \bigcup R_n$, кожна з яких задовольняє вказаним властивостям, причому декомпозиція таблиці $t(R)$ є декомпозицією без втрат: $t = \pi_{R_1}(t) \otimes \pi_{R_2}(t) \otimes ... \otimes \pi_{R_n}(t)$ [4].□

Якщо множина ФЗ F містить лише повні ФЗ, то в означенні 9 усі детермінанти є потенційними ключами.

Лема 6. $НФБК \Leftrightarrow 3НФ$ за умови $|R| \leq 2$ або $|D| \leq 1$.

Доведення проводиться розглядом усіх можливих випадків і абсолютно ідентичне до доведення твердження 2. □

Лема 7. $НФБК \Rightarrow 3НФ$. □

Доведення проводиться від супротивного. Припустимо таблиця t знаходиться у НФБК і не знаходиться у 3НФ, тобто для неключового атрибута A існує множина атрибутів $Y \subseteq R, A \notin Y$ така, що для жодного потенційного ключа K ФЗ $Y \to K$ не виконується, а ФЗ $Y \to \{A\}$ виконується. Оскільки, за припущенням, таблиця t знаходиться у НФБК, то за означенням 9 множина Y є суперключем, тобто існує такий потенційний ключ K, що виконується ФЗ $Y \to K$. Маємо протиріччя. □

Список використаних джерел

1 Редько В.Н. Реляційні бази даних: табличні алгебри та SQL-подібні мови / В.Н. Редько, Ю.Й. Брона, Д.Б. Буй, С.А. Поляков. – Київ: Видавничий дім "Академперіодика", 2001. – 198 с.

2 Мейер Д. Теория реляционных баз данных / Д. Мейер. – Москва: Мир, 1987. – 608 с.

3 Дейт К. Дж. Введение в системы баз данных, 8-е издание.: Пер. с англ. – М.: Издательский дом "Вильямс", 2005. – 1328 с. Реляційні бази даних: табличні алгебри та SQL-подібні мови / В.Н. Редько, Ю.Й. Брона, Д.Б. Буй, С.А. Поляков. – Київ: Видавничий дім „Академперіодика", 2001. – 198 с.

4 Дрибас В.П. Реляционные модели баз данных / В.П. Дрибас. – Минск: Изд-во БГУ им. В.И. Ленина, 1982. – 192 с.

5 Буй Д.Б. Критерій повноти аксіоматики Армстронга / Д.Б. Буй, А.В. Пузікова // Матеріали міжнародної конференції «Теоретичні та прикладні аспекти побудови програмних систем» – TAAPSD'2011 (Україна, Ялта, 19-23 вересня 2011 року). – С. 30-34.

6 Исаченко А.Н. Модели данных и СУБД / А.Н. Исаченко, С.П. Бондаренко – Минск: БГУ, 2007. – 205 с.

РОЗШИРЕНА МУЛЬТИМНОЖИННА ТАБЛИЧНА АЛГЕБРА: ОПЕРАЦІЇ З'ЄДНАННЯ

І.М. Глушко

Ніжинський державний університет імені Миколи Гоголя

glushkoim@gmail.com

Вступ

Реляційна модель даних в теперішній час широко використовується як у наукових дослідженнях в базах даних, так на практиці. Дана модель базується на множинах кортежів, тобто не дозволяє дублікати кортежів у відношенні [1]. Проте багато мов, орієнтованих на роботу з базами даних, вимагають реляційну модель даних з мультимножинною семантикою, що передбачає розуміння таблиць як мультимножин, тобто сукупностей з дублікатами. Питанню використання мультимножин в базах даних приділяли увагу Paul W.P.J. Grefen та Rolf A. de By [2], G. Lamperti, M. Melchiori, M. Zanella [3], A. Silbeschatz, H. Korth, S. Sudarshan [4], Д.Б. Буй, С.А. Поляков, Ю.Й. Брона, В.Н. Редько [5]. У [2, 3, 4] вводиться до розгляду мультимножинна таблична алгебра та задаються її операції. Разом з тим, в жодній із зазначених робіт не приділяється увага операціям внутрішніх та зовнішніх з'єднань над таблицями мультимножинної табличної алгебри.

Мультимножинна таблична алгебра

Всі невизначені тут поняття розуміємо в сенсі [5, 6]. Розглянемо дві множини: $\boldsymbol{A}$ – множину атрибутів і $\boldsymbol{D}$ – універсальний домен. Довільну скінченну множину атрибутів $R \subseteq \boldsymbol{A}$ назвемо схемою. Рядком схеми R називається іменна множина на парі R, $\boldsymbol{D}$, проекція якої за першою компонентою рівна R. Під таблицею розуміємо пару $\langle \psi, R \rangle$, де перша компонента ψ – це довільна мультимножина, зокрема, нескінченна, а друга компонента R – схема таблиці.

Під мультимножинною табличною алгеброю розуміємо алгебру $\langle \Psi, \Omega_{P,\Xi} \rangle$, де $\Psi = \bigcup\limits_{R \subseteq \boldsymbol{A}} \Psi(R)$ – множина усіх таблиць, $\Psi(R)$ – множина усіх таблиць схеми R, $\Omega_{P,\Xi} = \{\bigcup\nolimits_{All}^{R}, \bigcap\nolimits_{All}^{R}, \setminus_{All}^{R}, \sigma_{p,R}, \pi_{X,R}, \underset{R_1,R_2}{\otimes}, Rt_{\xi,R}, \sim_R\}_{X,R,R_1,R_2 \subseteq \boldsymbol{A}}^{p \in P, \xi \in \Xi}$ – сигнатура, P, Ξ – множини параметрів. Операції сигнатури мультимножинної табличної алгебри задано в [6].

Поповнимо сигнатуру мультимножинної табличної алгебри операціями внутрішніх і зовнішніх з'єднань та операцією напівз'єднання.

Операції внутрішнього з'єднання

Під декартовим з'єднанням таблиць схем R_1 та R_2, причому $R_1 \bigcap R_2 = \varnothing$, розуміється бінарна параметрична операція вигляду $\underset{R_1,R_2}{Cj} : \Psi(R_1) \times \Psi(R_2) \to \Psi(R_1 \bigcup R_2)$, причому

$$\langle \psi_1, R_1 \rangle \underset{R_1,R_2}{Cj} \langle \psi_2, R_2 \rangle = \langle \psi', R_1 \bigcup R_2 \rangle,$$

де $\langle \psi_1, R_1 \rangle \in \Psi(R_1), \quad \langle \psi_2, R_2 \rangle \in \Psi(R_2)$.

Основою мультимножини ψ' є множина рядків

$$\Theta(\psi') = \{ s \mid \exists s_1 \exists s_2 (s_1 \in \Theta(\psi_1) \wedge s_2 \in \Theta(\psi_2) \wedge s = s_1 \bigcup s_2) \}.$$

Кількість дублікатів знаходиться так:

$$Occ(s, \psi') = Occ(s_1, \psi_1) \cdot Occ(s_2, \psi_2),$$

де $s \in \Theta(\psi')$ і $s_1 = s \mid R_1, \quad s_2 = s \mid R_2$ – обмеження рядка s на вказані схеми (див., наприклад, [5]).

Під внутрішнім природним з'єднанням таблиць схем R_1 та R_2 розуміється бінарна параметрична операція $\underset{R_1,R_2}{\otimes}$, значеннями якої є таблиці схеми $R_1 \bigcup R_2$, що складаються з усяких об'єднань сумісних рядків вихідних таблиць.

Отже, $\underset{R_1,R_2}{\otimes} : \Psi(R_1) \times \Psi(R_2) \to \Psi(R_1 \bigcup R_2)$, де $\langle \psi_1, R_1 \rangle \underset{R_1,R_2}{\otimes} \langle \psi_2, R_2 \rangle = \langle \psi', R_1 \bigcup R_2 \rangle$, в припущенні $\langle \psi_1, R_1 \rangle \in \Psi(R_1), \quad \langle \psi_2, R_2 \rangle \in \Psi(R_2)$.

Основою мультимножини ψ' є множина рядків

$$\Theta(\psi') = \{ s \mid \exists s_1 \exists s_2 (s_1 \in \Theta(\psi_1) \wedge s_2 \in \Theta(\psi_2) \wedge \\ \wedge s_1 \approx s_2 \wedge s = s_1 \bigcup s_2) \}.$$

Кількість дублікатів знаходиться так:

$$Occ(s, \psi') = Occ(s_1, \psi_1) \cdot Occ(s_2, \psi_2),$$

де, як і раніше, $s \in \Theta(\psi')$ і $s_1 = s \mid R_1, \quad s_2 = s \mid R_2$.

Під внутрішнім з'єднанням за атрибутами $A_1,...,A_n$, причому всі $A_1,...,A_n$ попарно різні, $n \geq 1$, таблиць схем R_1 та R_2, де $R_1 \bigcap R_2 = \{A_1,...,A_n\}$ розуміється бінарна параметрична операція вигляду $\underset{A_1,...,A_n,R_1,R_2}{\otimes} : \Psi(R_1) \times \Psi(R_2) \to \Psi(R_1 \bigcup R_2)$, де $\langle \psi_1, R_1 \rangle \underset{A_1,...,A_n,R_1,R_2}{\otimes} \langle \psi_2, R_2 \rangle = \langle \psi', R_1 \bigcup R_2 \rangle$, в припущенні $\langle \psi_1, R_1 \rangle \in \Psi(R_1), \ \langle \psi_2, R_2 \rangle \in \Psi(R_2)$.

Основою мультимножини ψ' є множина рядків

$$\Theta(\psi') = s \mid \exists s_1 \exists s_2 (s_1 \in \Theta(\psi_1) \wedge s_2 \in \Theta(\psi_2) \wedge$$
$$\wedge \underset{i=1}{\overset{n}{\wedge}} s_1(A_i) = s_2(A_i) \wedge s = s_1 \bigcup s_2).$$

Кількість дублікатів знаходиться так:

$$Occ(s,\psi') = Occ(s_1,\psi_1) \cdot Occ(s_2,\psi_2),$$

де $s \in \Theta(\psi')$ і $s_1 = s \mid R_1, \ s_2 = s \mid R_2$.

Нехай $p : S \times S \xrightarrow{\sim} \{true, false\}$ – частковий бінарний предикат на множині всіх рядків S, такий, що $\forall s_1 \forall s_2 (\langle s_1, s_2 \rangle \in \operatorname{dom} p \wedge p(s_1, s_2) = true \Rightarrow s_1 \approx s_2)$.

Під внутрішнім з'єднанням за предикатом p таблиць схем R_1 та R_2 розуміється часткова бінарна параметрична операція вигляду $\underset{p,R_1,R_2}{\otimes} : \Psi(R_1) \times \Psi(R_2) \xrightarrow{\sim} \Psi(R_1 \bigcup R_2)$, де

$$\operatorname{dom} \underset{p,R_1,R_2}{\otimes} = \{\langle \langle \psi_1, R_1 \rangle, \langle \psi_2, R_2 \rangle \rangle \mid \Theta(\psi_1) \times \Theta(\psi_2) \subseteq \operatorname{dom} p\}$$

та $\langle \psi_1, R_1 \rangle \underset{p,R_1,R_2}{\otimes} \langle \psi_2, R_2 \rangle = \langle \psi', R_1 \bigcup R_2 \rangle$.

Основою мультимножини ψ' є множина рядків

$$\Theta(\psi') = \{s \mid \exists s_1 \exists s_2 (s_1 \in \Theta(\psi_1) \wedge s_2 \in \Theta(\psi_2) \wedge p(s_1, s_2) \simeq true \wedge$$
$$\wedge s = s_1 \bigcup s_2)\}.$$

Кількість дублікатів знаходиться так:

$$Occ(s,\psi') = Occ(s_1,\psi_1) \cdot Occ(s_2,\psi_2),$$

де $s \in \Theta(\psi')$ і $s_1 = s \mid R_1, \ s_2 = s \mid R_2$.

Операція природного з'єднання $\underset{R_1,R_2}{\otimes}$ є розширенням довільної іншої операції з'єднання в наступному розумінні:

$$\langle \psi_1, R_1 \rangle \underset{R_1,R_2}{Cj} \langle \psi_2, R_2 \rangle = \langle \psi_1, R_1 \rangle \underset{R_1,R_2}{\otimes} \langle \psi_2, R_2 \rangle,$$

$$\langle \psi_1, R_1 \rangle \underset{A_1,\ldots,A_n,R_1,R_2}{\otimes} \langle \psi_2, R_2 \rangle = \langle \psi_1, R_1 \rangle \underset{R_1,R_2}{\otimes} \langle \psi_2, R_2 \rangle,$$

$$\left(\langle \psi_1, R_1 \rangle \underset{p,R_1,R_2}{\otimes} \langle \psi_2, R_2 \rangle \right)_1 \preceq \left(\langle \psi_1, R_1 \rangle \underset{R_1,R_2}{\otimes} \langle \psi_2, R_2 \rangle \right)_1,$$

за умови визначення значень операцій у лівих частинах цих двох рівностей та включення ($\preceq$ – відношення включення мультимножин [5]). Запис $\left(\varphi\left(\langle \psi_1, R_1 \rangle, \langle \psi_2, R_2 \rangle\right)\right)_1$, позначає першу компоненту таблиці $\varphi\left(\langle \psi_1, R_1 \rangle, \langle \psi_2, R_2 \rangle\right)$.

Під операцією напівз'єднання двох таблиць схем R_1 та R_2 розуміється бінарна параметрична операція $\ltimes_{R_1,R_2}$, значенням якої є таблиця схеми R_1, що містить ті рядки першої таблиці, які входять у (природне) з'єднання таблиць-аргументів.

Таким чином, $\ltimes_{R_1,R_2} : \Psi(R_1) \times \Psi(R_2) \to \Psi(R_1)$, де $\langle \psi_1, R_1 \rangle \ltimes_{R_1,R_2} \langle \psi_2, R_2 \rangle = \langle \psi', R_1 \bigcup R_2 \rangle$, в припущенні $\langle \psi_1, R_1 \rangle \in \Psi(R_1)$, $\langle \psi_2, R_2 \rangle \in \Psi(R_2)$

Основою мультимножини ψ' є множина рядків

$$\Theta(\psi') = \{s_1 \mid \exists s_2 (s_1 \in \Theta(\psi_1) \wedge s_2 \in \Theta(\psi_2) \wedge s_1 \approx s_2)\}.$$

Кількість дублікатів знаходиться так: $Occ(s, \psi') = Occ(s, \psi_1)$, де $s \in \Theta(\psi')$.

Операції зовнішнього з'єднання

Для позначення відсутніх значень у результуючій таблиці, використовуємо особливий елемент універсального домену $NULL$. Позначимо через s_R^{NULL} константний рядок схеми R, тобто $s_R^{NULL} : R \to \{NULL\}$.

Використаємо логічну схему задання операцій зовнішнього з'єднання [5]. Нехай $\varphi : \Psi(R_1) \times \Psi(R_2) \xrightarrow{\sim} \Psi(R_1 \bigcup R_2)$ – деяка

часткова бінарна операція на множині таблиць, причому виконується включення

$$\left(\varphi\left(\left\langle\psi_1, R_1\right\rangle, \left\langle\psi_2, R_2\right\rangle\right)\right)_1 \preceq \left(\left\langle\psi_1, R_1\right\rangle \underset{R_1, R_2}{\otimes} \left\langle\psi_2, R_2\right\rangle\right)_1$$

для всіх $\left\langle\left\langle\psi_1, R_1\right\rangle, \left\langle\psi_2, R_2\right\rangle\right\rangle \in \mathrm{dom}\,\varphi$. Зауважимо, що операції $\underset{R_1, R_2}{Cj}$, $\underset{R_1, R_2}{\otimes}$, $\underset{A_1, \ldots, A_n, R_1, R_2}{\otimes}$, $\underset{p, R_1, R_2}{\otimes}$ саме такі.

Зафіксуємо таблиці $\left\langle\psi_1, R_1\right\rangle$, $\left\langle\psi_2, R_2\right\rangle$ з області означеності операції φ. Тоді таблиця $\left\langle\psi_1, R_1\right\rangle$ припускає наступне представлення:

$$\left\langle\psi_1, R_1\right\rangle = \left\langle\psi_1 \underset{\varphi}{\bigcap} \psi_2, R_1\right\rangle \bigcup_{All}^{R_1} \left\langle\psi_1 \underset{\varphi}{-} \psi_2, R_1\right\rangle,$$

де $\left\langle\psi_1 \underset{\varphi}{\bigcap} \psi_2, R_1\right\rangle = \left\langle\psi', R_1\right\rangle$, основою мультимножини ψ' є множина рядків $\Theta(\psi') = \{s_1 \mid s_1 \in \Theta(\psi_1) \wedge \exists s_2 (s_2 \in \Theta(\psi_2) \wedge$ $\wedge\, s_1 \bigcup s_2 \in \Theta((\varphi(\left\langle\psi_1, R_1\right\rangle, \left\langle\psi_2, R_2\right\rangle))_1))\}$, а кількість дублікатів $Occ(s_1, \psi') = Occ(s_1, \psi_1)$, де $s_1 \in \Theta(\psi')$, та $\left\langle\psi_1 \underset{\varphi}{-} \psi_2, R_1\right\rangle = \left\langle\psi'', R_1\right\rangle$, основою мультимножини ψ'' є множина рядків $\Theta(\psi'') = \{s_1 \mid s_1 \in \Theta(\psi_1) \wedge \forall s_2 (s_2 \in \Theta(\psi_2) \Rightarrow$ $\Rightarrow s_1 \bigcup s_2 \notin \Theta((\varphi(\left\langle\psi_1, R_1\right\rangle, \left\langle\psi_2, R_2\right\rangle))_1))\}$, а кількість дублікатів знаходиться так: $Occ(s_1, \psi'') = Occ(s_1, \psi_1)$, де $s_1 \in \Theta(\psi'')$.

Іншими словами, рядки з таблиці $\left\langle\psi_1 \underset{\varphi}{\bigcap} \psi_2, R_1\right\rangle$ використовуються в формуванні результату з'єднання, а рядки з таблиці $\left\langle\psi_1 \underset{\varphi}{-} \psi_2, R_1\right\rangle$ не використовуються. Представлення таблиці $\left\langle\psi_2, R_2\right\rangle$ отримаємо, замінивши ролі таблиць $\left\langle\psi_1, R_1\right\rangle$ і $\left\langle\psi_2, R_2\right\rangle$ у представленні таблиці $\left\langle\psi_1, R_1\right\rangle$.

Задамо чотири операції зовнішнього з'єднання, індуковані операцією внутрішнього з'єднання φ. Для цього розглянемо наступні природні з'єднання:

$$\left\langle \psi_1 \underset{\varphi}{-} \psi_2, R_1 \right\rangle \underset{R_1, R_2 \setminus R_1}{\otimes} \left\langle \{ s^{NULL}_{R_2 \setminus R_1} \}, R_2 \setminus R_1 \right\rangle = \left\langle \psi', R_1 \bigcup R_2 \right\rangle,$$

де основою мультимножини ψ' є множина рядків $\Theta(\psi') = \{ s_1 \bigcup s^{NULL}_{R_2 \setminus R_1} \mid s_1 \in \Theta(\psi_1 \underset{\varphi}{-} \psi_2) \}$, а кількість дублікатів $Occ(s', \psi') = Occ(s' \mid R_1, \psi_1 \underset{\varphi}{-} \psi_2)$, $s' \in \Theta(\psi')$ і

$$\left\langle \psi_2 \underset{\varphi}{-} \psi_1, R_2 \right\rangle \underset{R_2, R_1 \setminus R_2}{\otimes} \left\langle \{ s^{NULL}_{R_1 \setminus R_2} \}, R_1 \setminus R_2 \right\rangle = \left\langle \psi'', R_1 \bigcup R_2 \right\rangle,$$

де основою мультимножини ψ'' є множина рядків $\Theta(\psi'') = \{ s^{NULL}_{R_1 \setminus R_2} \bigcup s_2 \mid s_2 \in \Theta(\psi_2 \underset{\varphi}{-} \psi_1) \}$, а кількість дублікатів $Occ(s'', \psi'') = Occ(s'' \mid R_2, \psi_2 \underset{\varphi}{-} \psi_1)$, $s'' \in \Theta(\psi'')$.

Під зовнішнім лівим з'єднанням, індукованим операцією φ, розуміється часткова бінарна операція вигляду $\varphi_l : \Psi(R_1) \times \Psi(R_2) \xrightarrow{\sim} \Psi(R_1 \bigcup R_2)$, де $\mathrm{dom}\varphi_l = \mathrm{dom}\varphi$, причому

$$\varphi_l\left(\left\langle \psi_1, R_1 \right\rangle, \left\langle \psi_2, R_2 \right\rangle\right) = \varphi\left(\left\langle \psi_1, R_1 \right\rangle, \left\langle \psi_2, R_2 \right\rangle\right) \bigcup^{R_1 \cup R_2}_{All}$$

$$\bigcup^{R_1 \cup R_2}_{All} \left\langle \psi_1 \underset{\varphi}{-} \psi_2, R_1 \right\rangle \underset{R_1, R_2 \setminus R_1}{\otimes} \left\langle \{ s^{NULL}_{R_2 \setminus R_1} \}, R_2 \setminus R_1 \right\rangle.$$

Під зовнішнім правим з'єднанням, індукованим операцією φ, розуміється часткова бінарна операція вигляду $\varphi_r : \Psi(R_1) \times \Psi(R_2) \xrightarrow{\sim} \Psi(R_1 \bigcup R_2)$, де $\mathrm{dom}\varphi_r = \mathrm{dom}\varphi$, причому

$$\varphi_r\left(\left\langle \psi_1, R_1 \right\rangle, \left\langle \psi_2, R_2 \right\rangle\right) = \varphi\left(\left\langle \psi_1, R_1 \right\rangle, \left\langle \psi_2, R_2 \right\rangle\right) \bigcup^{R_1 \cup R_2}_{All}$$

$$\bigcup^{R_1 \cup R_2}_{All} \left\langle \psi_2 \underset{\varphi}{-} \psi_1, R_2 \right\rangle \underset{R_2, R_1 \setminus R_2}{\otimes} \left\langle \{ s^{NULL}_{R_1 \setminus R_2} \}, R_1 \setminus R_2 \right\rangle.$$

Під повним зовнішнім з'єднанням, індукованим операцією φ, розуміється часткова бінарна операція вигляду $\varphi_f : \Psi(R_1) \times \Psi(R_2) \xrightarrow{\sim} \Psi(R_1 \bigcup R_2)$, де $\mathrm{dom}\varphi_f = \mathrm{dom}\varphi$, причому

$$\varphi_f\left(\langle\psi_1, R_1\rangle, \langle\psi_2, R_2\rangle\right) = \varphi\left(\langle\psi_1, R_1\rangle, \langle\psi_2, R_2\rangle\right) \bigcup_{All}^{R_1 \cup R_2}$$

$$\bigcup_{All}^{R_1 \cup R_2} \left\langle \psi_1 \underset{\varphi}{-} \psi_2, R_1 \right\rangle \underset{R_1, R_2 \setminus R_1}{\otimes} \left\langle \{s_{R_2 \setminus R_1}^{NULL}\}, R_2 \setminus R_1 \right\rangle \bigcup_{All}^{R_1 \cup R_2}$$

$$\bigcup_{All}^{R_1 \cup R_2} \left\langle \psi_2 \underset{\varphi}{-} \psi_1, R_2 \right\rangle \underset{R_2, R_1 \setminus R_2}{\otimes} \left\langle \{s_{R_1 \setminus R_2}^{NULL}\}, R_1 \setminus R_2 \right\rangle.$$

Під зовнішнім з'єднанням об'єднанням розуміється часткова бінарна операція вигляду $\varphi_{\cup} : \Psi(R_1) \times \Psi(R_2) \xrightarrow{\sim} \Psi(R_1 \bigcup R_2)$, де $\mathrm{dom}\,\varphi_{\cup} = \mathrm{dom}\,\varphi$, причому

$$\varphi_{\cup}\left(\langle\psi_1, R_1\rangle, \langle\psi_2, R_2\rangle\right) = \left\langle \psi_1 \underset{\varphi}{-} \psi_2, R_1 \right\rangle \underset{R_1, R_2 \setminus R_1}{\otimes} \left\langle \{s_{R_2 \setminus R_1}^{NULL}\}, \right.$$

$$\left. R_2 \setminus R_1 \right\rangle \bigcup_{All}^{R_1 \cup R_2} \left\langle \psi_2 \underset{\varphi}{-} \psi_1, R_2 \right\rangle \underset{R_2, R_1 \setminus R_2}{\otimes} \left\langle \{s_{R_1 \setminus R_2}^{NULL}\}, R_1 \setminus R_2 \right\rangle.$$

Висновки

Отже, задано операції внутрішніх і зовнішніх з'єднань та операцію напівз'єднання над таблицями мультимножинної табличної алгебри.

Список використаних джерел

1. Codd E.F. Relational Completeness of Data Base Sublanguages / Codd E.F. // Data Base Systems. – New York: Prentice-Hall. – 1972. – P. 65-93.
2. Grefen Paul W.P.J. A Multi-Set Extended Relational Algebra. A Formal Approach to a Practical Issue / Paul W.P.J. Grefen, Rolf A. De By // 10th International Conference on Data Engineering, ICDE, February 14-18, 1994, Houston, TX, USA. – P. 80-88.
3. Lamperti G. On Multisets in Database Systems / G. Lamperti, M. Melchiori, M. Zanella // Multiset Processing: Mathematical, Computer Science, and Molecular Computing Points of View, number 2235 in Lecture Notes in Computing Since. – Berlin: Springer-Verlag, 2001. – P. 147-215.
4. Silbeschatz A. Database System Concepts: [6th Edition] / A. Silbeschatz, H. Korth, S. Sudarshan. – McGraw-Hill, 2011. – 1376 p.
5. Реляційні бази даних: табличні алгебри та SQL-подібні мови / В.Н. Редько, Ю.Й. Брона, Д.Б. Буй, С.А. Поляков. – Київ: Видавничий дім "Академперіодика", 2001. – 198 с.
6. Глушко І.М. Мультимножинна таблична алгебра / І.М. Глушко // Proceedings of the International Scientific Conference of Student and Young Scientists "Theoretical and Applied Aspects of Cybernetics" (TAAC'2011, Kyiv, February 21–25, 2011). – P. 77-79.

МОДЕЛЬ ДЛЯ ДОСЛІДЖЕННЯ ХАРАКТЕРИСТИК СИСТЕМИ КЕРУВАННЯ СТАТИСТИЧНОГО ВИМІРЮВАЛЬНОГО ІНФОРМАЦІЙНОГО КОМПЛЕКСУ

І.Ю. Грішин

Республіканський вищий навчальний заклад «Кримський гуманітарний університет», Ялта, Україна

igugri@gmail.com

Постановка проблеми в загальному вигляді й аналіз літератури

При формуванні технічного завдання на розробку вимірювального інформаційного комплексу (ВІК), а також в ході самої розробки важливо визначити відповідність проектованої системи її майбутньому призначенню і можливість виконувати поставлені задачі. В частковості, при проектуванні забезпечення льотних випробувань літаків і ракет-носіїв, надзвичайно важливо знати часові і геометричні характеристики знаходження об'єктів, що випробовується, в зоні огляду такого комплексу. До розглядуваних параметрів можна віднести: час знаходження в зоні огляду, точки входу до зони огляду і виходу з неї, взаємне розташування зони огляду і траєкторії об'єкту, якість функціонування системи керування та інші параметри.

Проведення натурних експериментів часто є неможливим на начальних стадіях розробки внаслідок їх високої вартості.

Моделювання дозволяє перевірити правильність теоретичних уявлень, уточнити сутність тих чи інших явищ, спрогнозувати проходження різних процесів. Використання методів і засобів моделювання дозволяє прискорити і скоротити витрати на розробку більш досконалих інформаційних комплексів, а також полегшує вибір місця монтажу й експлуатації, а також підготовку обслуговуючого персоналу.

Тому розробка моделей, що дозволяють оцінити можливості тих, які розробляються, або вже експлуатуються (але в стандартних умовах) ВІК, є досить актуальною проблемою.

В роботі [1] проведено аналіз основних підходів до моделювання основних елементів ВІК, наведено методи їх моделювання, що враховують статистичний характер оброблюваної інформації, однак методи моделювання цільової обстановки майже не зачіплено.

В [2] розглядаються принципи математичного моделювання радіотехнічних систем. Наводяться алгоритми моделювання на ЕОМ детермінованих і випадкових радіосигналів, лінійних і нелінійних

систем. Викладаються основні методи обробки результатів математичного моделювання. Наведені приклади математичних моделей різних радіотехнічних систем. Алгоритми, призначені для оцінки комплексних часових параметрів не розглянуто.

Роботи [3, 4] присвячено розгляду методик моделювання окремих елементів радіотехнічних систем, досить повно викладено особливості їх статистичних моделей, показано методи обробки результатів статистичного експерименту, однак питання комплексної оцінки можливостей таких систем по супроводженню об'єктів, часу їх виявлення в зоні огляду не розглянуто.

Мета статті

Задача полягає в розробці методики моделювання зовнішньої обстановки, а також зони огляду вимірювальних інформаційних комплексів, системи керування й обробки інформації для оцінки якості системи керування ВІК, що проектується.

Виклад основного матеріалу

Відомо [2], що будь-який статистичний ВІК може бути укрупнено зображено в вигляді наступної структурної схеми (рис. 1).

Рис. 1. Структура статистичного ВІК

Інформаційний зміст радіосигнал набуває в каналі розповсюдження внаслідок впливу на його параметри фізичних властивостей середовища (відображення сигналу від супроводжуваного об'єкту). Такий спосіб є характерним для ВІК.

Слід відзначити, що вивчення впливу дії перешкод на відображений від об'єкту сигнал, а також обробка інформації в ВІК

досить детально розглянуто в ряді робіт [8,9] і не є об'єктом вивчення даного дослідження.

Отже, для вирішення поставленої проблеми необхідно сформувати траєкторію руху об'єкту в вигляді сукупності точок, що відповідають можливим моментам проведення вимірювань її параметрів, а також оцінити приналежність цих точок зоні огляду ВІК.

Отже, розглядувана модель має включати наступні модулі (рис. 2):

формування первинної вимірювальної інформації;

формування параметрів зони огляду ВІК в просторі;

вироблення рішення щодо приналежності цілі зоні огляду і розрахунку часових параметрів;

моделювання алгоритмів обробки інформації;

моделювання системи керування.

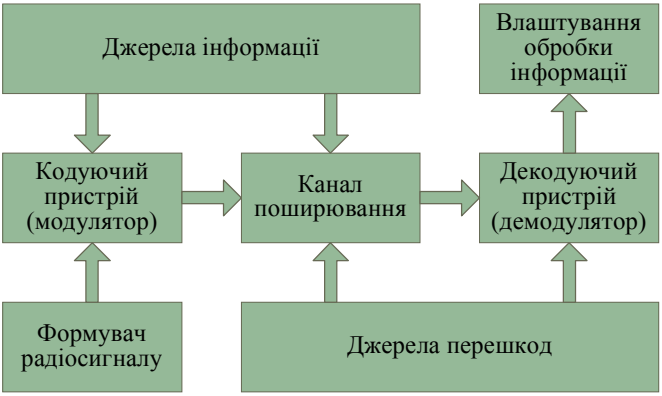


Рис. 2. Структурна схема моделі для дослідження характеристик системи керування статистичного ВІК

Виділяючи із зазначених завдань повторювані підзадачі, можна зробити висновок, що моделювання при дослідженні системи керування ВІК зводиться до сукупності таких методів:

формування заданих детермінованих функцій;

- формування випадкових величин і випадкових процесів із заданими імовірнісними характеристиками;
- лінійне перетворення сигналів відповідно до заданої ваговій функцією;
- нелінійне перетворення сигналів відповідно до заданої функцією перетворення;

- алгоритми управління;
- алгоритми первинної та вторинної обробки вимірювальної інформації.

Висновки

Розглянуто основні вимоги і методи моделювання головних елементів статистичного вимірювального інформаційного комплексу. Представлено математичний апарат, використовуваний при моделюванні, а також приклад реалізації розробленої моделі.

Список використаних джерел

1. Моделирование в радиолокации / под ред. А. И. Леонова – М . : Сов. радио, 1979. - 264 с.
2. Монаков А.А. Основы математического моделирования радиотехнических систем: учеб. Пособие / А.А. Монаков – СПб.: ГУАП, 2005. –100 с.
3. Быков В.В. Цифровое моделирование в статистической радиотехнике / В.В. Быков – М.: Сов.радио, 1971. –328 с.
4. Грішин І.Ю. Модель для оцінки часових характеристик вимірювальної інформаційної системи / Грішин І.Ю. // Вісник Національного технічного університету "Харківський політехнічний інститут". Збірник наукових праць. Тематичний випуск: Інформатика і моделювання. – Харків: НТУ "ХПІ", 2009. – № 13. С. 39—46.
5. Грішин І.Ю. Основи концепції побудови системи керування статистичними вимірювальними інформаційними системами скануючого типу / І.Ю. Грішин // // Тези Міжнародної наукової конференції «Інтелектуальні системи прийняття рішень і проблеми обчислювального інтелекту (ISDMCI’2013)», 20-24 травня 2013 р., Євпаторія. - Херсон: М-во освіти і науки України, Херсонський нац. техн. ун-т, 2013. – Т2, – Ч1. – С. 62-65.

ИСКУССТВЕННАЯ НЕЙРОННАЯ СЕТЬ И УПРАВЛЕНИЕ ИНВЕСТИЦИЯМИ ФОНДОВОГО РЫНКА

Ибраим Дидманидзе, Григол Кахиани

Батумский государственный университет Шота Руставели

ibraimd@mail.ru

Совместно с решением задач по прогнозированию на финансовых рынках с целью обеспечения эффективной инвестиционной политики, большая роль отводится правильному интерпретированию результатов прогнозирования, формализации правил торговли на фондовом рынке, анализу рисков доходности инвестиционного портфеля, методам эффективного управления капиталом и оценке обшей эффективности инвестиционной стратегий на финансовом рынке. Следовательно, с целью улучшения методов инвестирования и повышения качества управления капиталом становится построение искусственных нейронных сетей для прогнозирования временных рядов котировок ценных бумаг фондового рынка, выявление особенности их поведения.

Для подбора финансовых инструментов рынка и анализа эффективности сформированного портфеля используется портфельная теория Мерковица, согласно которой прибыльность (μ_y) и риски (σ_y^2) инвестиционного портфеля состоящего из N активов оцениваются следующим образом:

$$\sum_{k=1}^{N} w_k \mu_k ;$$

$$\sigma_y^2 = \sum_{k=1}^{N} w_k \sigma_k^2 + 2 \sum_{k=1}^{N} \sum_{i=k+1}^{N} w_i w_k \rho_{ik} \sigma_i \sigma_k ,$$

где w_i – весовой коэффициент i-го актива в портфеле, в свою очередь весовые коэффициенты активов подчиняются $\sum_{k=1}^{N} w_i = 1$ условию нормирования; ρ_{ik} – коэффициенты корреляции между i-м и j активами портфеля; $\sigma_k, \mu_k, k = \overline{1, N}$ - риск и прибыльность $k-$ го актива.

С целью прогнозирования динамики котировок финансовых инструментов фондового рынка, тоесть для решения задачи

$x_{i+1} = F(x_1, x_2, \dots, x_l)$ в работе применены дискретные модели нейронных сетей, для которых переход из l-го в (l+1) состояние происходит по следующему правилу:

$$x_i^{l+1} = f_l\left(\sum_{j=1}^{n} w_{ij}^l x_j^l\right), i = \overline{1,n}, l = \overline{0,q-1}.$$

w_{ij}^l- весовые коэффициенты берутся из условия максимума:

$$I[w] = \frac{1}{2}\sum_{i=1}^{N}\left(x_i^q - d_i\right)^2$$

функции, где в свою очередь w_i и d_i являются параметрами эталонных значений временного ряда поданных сети с целью обучения.

Список использованной литературы

1. Булашев С В . Статистика для трейдеров. — М.: Компания «Спутник +» , 2003 .
2. Дидманидзе И. Мегрелишвили З. Кахиани Г. К вопросу об использовании искусственной нейронной сети для прогнозирования валютных курсов на высоковолантильных рынках. Материалы I международной научно-практической интернет конференций «СУЧАСНІ ТЕНДЕНЦІЇ РОЗВИТКУ МАТЕМАТИКИ ТА ІІ ПРИКЛАДНІ АСПЕКТИ» - 2012, С. 166-167. Донецк. 2012.

СЖАТИЕ ИЗОБРАЖЕНИЙ

Ибраим Дидманидзе, Рима Тхилаишвили

Батумский государственный университет Шота Руставели

ibraimd@mail.ru

Несмотря на многолетние усилия по вопросам сжатия изображений, данные о полученных результатах весьма скромны, но они позволяют глубже понять масштабность и значимость проблемы.

Большинство интересных файлов имеют размер, по крайней мере, в несколько тысяч байтов. Для таких размеров доля эффективно сжимаемых файлов настолько мала, что ее невозможно выразить числом с плавающей точкой даже на суперкомпьютере (результат будет нуль).

Если каждому биту информации в идеальном случае будет соответствовать один элемент, то одно устройство ЗУ в лучшем случае будет состоять приблизительно из $7 \cdot 10^6$ элементов. При этом основные требования будут предъявляться вопросу уменьшения веса и объема управляющее - вычислительной системы (УВС).

Принципиальную важность с точки зрения уменьшения веса и объема УВС приобретает вопрос снижения мощности излучателя (передатчик информации), которая определяется как

$$P_{изл}=\frac{(4\pi r)^2 \cdot P_{пр}}{\lambda^2 g_1 g_2} \qquad (1)$$

Где $P_{изл}$ – мощность излучателя, $P_{пр}$- мощность приёмника, r – расстояние, $\lambda -$ длина волны, g_1 и g_2 - соответственно коэффициенты усиления приёмных и передающих антенн.

Квадратная зависимость $P_{изл} = f(r)^2$ от расстояния при дальних коммуникациях даже при идеально вообразимых значениях $g_1, g_2 \cong 10^4$ делает величину $P_{изл}$ очень громоздким.

Эффективным средством существенно снизит значение $P_{изл}$ и соответственно вес и габариты УВС - является применение сжатия информацию.

Согласно формуле - Хартли-Шенона имеем (без сжатия)

$$C_1 = B_1 . \lg_2 \left(1 + \frac{P'_{c\ (1)}}{P'_{ш\ (1)}}\right) \qquad (2)$$

где C_1 – емкость канала, B_1 – полоса пропускания сигнала, $P'_{изл(1)}$- мощность излучателя, $P'_{ш(1)}$ - спектральная плотность мощности шума.

При сжатии имеем

$$C_2 = B_{12}.\lg_2\left(1 + \frac{P'_{c\ (2)}}{P'_{ш\ (2)}}\right) \qquad (3)$$

где C_2 – требуемая емкость канала при сжатии, B_1 – полоса пропускания сигнала, $P'_{изл(1)}$ - мощность излучателя, $P'_{ш(1)}$ - спектральная плотность мощности шума.

Поскольку в результате сжатия C2 < C1 , коэффициент сжатия можно определить как $K_{сж} = \frac{C_1}{C_2}$.

При сжатии происходит уменьшение требуемой мощности излучателя по показательному закону, где показателем степени является коэффициент сжатия $K_{сж}$.

Уменьшение требуемой мощности непосредственно резко снижает объем и вес аппаратуры УВС.

Список использованной литературы

1. Н. Нанобашвили «Сжатие информации при склеивании монотонных пар кодовых последовательностей. Сообщения АН ГССР.93,№2,1979;

СОЗДАНИЕ СТРУКТУРИРОВАННОГО КУРСА ДИСТАНЦИОННОГО ОБУЧЕНИЯ В СРЕДЕ MOODLE

А.В. Дмитриев

Республиканское высшее учебное заведение «Крымский гуманитарный университет», Ялта, Украина

dmitriev_andrey1@mail.ru

Moodle относится к классу LMS (Learning Management System) – систем управления обучением. В нашей стране подобное программное обеспечение чаще называют системами дистанционного обучения (СДО), так как именно при помощи подобных систем во многих вузах организовано дистанционное обучение. Moodle – это свободное программное обеспечение с лицензией GPL, что дает возможность бесплатного использования системы, а также ее безболезненного изменения в соответствии с нуждами образовательного учреждения и интеграции с другими продуктами. Moodle – аббревиатура от Modular Object-Oriented Dynamic Learning Environment (модульная объектно-ориентированная динамическая обучающая среда). Благодаря своим функциональным возможностям система приобрела большую популярность и успешно конкурирует с коммерческими LMS. Moodle используется более чем в 30 000 учебных заведений по всему миру и переведена почти на 80 языков, в том числе и на русский. Более подробную информацию о Moodle можно узнать на официальном сайте проекта (http://www.moodle.org/).

Moodle дает возможность проектировать, создавать и в дальнейшем управлять ресурсами информационно-образовательной среды. Интерфейс системы изначально был ориентирован на работу преподавателей, не обладающих глубокими знаниями в области программирования и администрирования баз данных, -сайтов и т.п. Система имеет удобный интуитивно понятный интерфейс. Преподаватель самостоятельно, прибегая только к помощи справочной системы, может создать электронный курс и управлять его работой. Практически во всех ресурсах и элементах курса в качестве полей ввода используется удобный WYSIWYG HTML редактор, кроме того, существует возможность ввода формул в формате TeX или Algebra. Можно вставлять таблицы, схемы, графику, видео, флэш и др. Используя удобный механизм настройки, составитель курса может, даже не обладая знанием языка HTML, легко выбрать цветовую гамму и другие элементы оформления учебного материала.

Преподаватель может по своему усмотрению использовать как тематическую, так календарную структуризацию курса. При тематической структуризации курс разделяется на секции по темам. При календарной структуризации каждая неделя изучения курса представляется отдельной секцией, такая структуризация удобна при дистанционной организации обучения и позволяет студентам правильно планировать свою учебную работу.

Редактирование содержания курса проводится автором курса в произвольном порядке и может легко осуществляться прямо в процессе обучения. Очень легко добавляются в электронный курс различные элементы: лекция, задание, форум, глоссарий, wiki, чат и т.д. Для каждого электронного курса существует удобная страница просмотра последних изменений в курсе. Таким образом, LMS Moodle дает преподавателю обширный инструментарий для представления учебно-методических материалов курса, проведения теоретических и практических занятий, организации учебной деятельности школьников как индивидуальной, так и групповой.

Администрирование учебного процесса достаточно хорошо продумано. Преподаватель, имеющий права администратора, может регистрировать других преподавателей и студентов, назначая им соответствующие роли (создатель курса, преподаватель с правом редактирования и без него, студент, гость), распределять права, объединять студентов в виртуальные группы, получать сводную информацию о работе каждого ученика. С помощью встроенного календаря определять даты начала и окончания курса, сдачи определенных заданий, сроки тестирования. Используя инструмент Пояснение и Форум, публиковать информацию о курсе и новости. Ориентированная на дистанционное образование, система управления обучением Moodle обладает большим набором средств коммуникации. Это не только электронная почта и обмен вложенными файлами с преподавателем, но и форум (общий новостной на главной странице программы, а также различные частные форумы), чат, обмен личными сообщениями, ведение блогов.

Moodle имеет не только многофункциональный тестовый модуль, но и предоставляет возможность оценивания работы студентов в таких элементах курса как Задание, Форум, Wiki, Глоссарий и т.д. Причем оценивание может происходить и по произвольным, созданным преподавателем, шкалам. Существует возможность оценивания статей Wiki, глоссария, ответов на форуме другими участниками курса. Все оценки могут быть просмотрены на странице оценок курса, которая имеет множество настроек по виду отображения и группировки оценок.

Поскольку основной формой контроля знаний в дистанционном обучении является тестирование, в LMS Moodle имеется обширный инструментарий для создания тестов и проведения обучающего и контрольного тестирования. Поддерживается несколько типов вопросов в тестовых заданиях (множественный выбор, на соответствие, верно/неверно, короткие ответы, эссе и др.). Moodle предоставляет много функций, облегчающих обработку тестов. Можно задать шкалу оценки, при корректировке преподавателем тестовых заданий после прохождения теста обучающимися, существует механизм полуавтоматического пересчета результатов. В системе содержатся развитые средства статистического анализа результатов тестирования и, что очень важно, сложности отдельных тестовых вопросов для студентов.

В дипломном проекте рассматриваются особенности создания структурированного курса дистанционного обучения в среде Moodle. При этом особое внимание уделено вопросам обеспечения качества образования, рассмотрены факторы, определяющие качество образования. Большое внимание уделено вопросам стандартизации и рассмотрению существующих стандартов, особенно SCORM (http://www.adlnet.gov/Pages/Default.aspx) – набору спецификаций и стандартов, которые представлены разными организациями. Они все сгруппированы в три основных категории: модель объединения содержания ("Content Aggregation Model (CAM)"), средства управления работой программы ("Run-Time Environment (RTE)") и последовательность и навигация ("Sequencing and Navigation (SN)) (представлена в SCORM 2004).

Список использованной литературы

1 В.И. Солдаткин Образовательная среда сегодня и завтра. – М.: Рособразование, 2004. – 272 с.

2 В.П. Демкин, Г.В. Можаева Телекоммуникации для образования. – СПб.: «БХВ-Петербург», 2004. – 1136 с.: ил.

3 Официальный сайт LMS Moodle. Перевод статьи «Улучшения в версии Moodle 1.9» – http://docs.moodle.org/en/Release_Notes#Moodle_1.9.1

4 Н.Г. Малышев, В.А. Сердюк Международный центр дистанционного обучения: концепция и бизнес-план - М.: Минобразование России, 2004. - 402 с.

5 О.З. Лобковская, Н.Ю. Шабанова Методические указания для специальности 2202 «Автоматизированные системы обработки информации и управления» по технико-экономическому обоснованию дипломных проектов и работ, НИ РХТУ им. Д.И. Менделеева, Новомосковск, 2006г. – 40 с.

6 НПБ 105-03. Определение категорий помещений и зданий по взрывопожарной и пожарной опасности. – М.: Главное управление Государственной противопожарной службы МВД России, 1995.
7 НПБ 105-03. Нормы пожарной безопасности. – М.: Главное управление Государственной противопожарной службы МВД России, 1995.
8 СанПиН 2.2.4.548-96. Гигиенические требования к микроклимату производственных помещений. - М: Информационно-издательский центр Минздрава России, 1997, 23 с.
9 СанПиН 2.2.2/2.4.1340-03 Гигиенические требования к персональным электронно-вычислительным машинам и организации работы. Основные положения. – М.: Издательство стандартов, 2003.
10 ГОСТ 27818–88. Допустимые уровни шума на рабочих местах и методы определения.

ФРЕЙМВОРК ДЛЯ АВТОМАТИЗИРОВАННОГО СОЗДАНИЯ ВЕБ-ПРИЛОЖЕНИЙ

В.И. Егоров[1], Л.Н. Суслова[2], А.Е. Дорошенко[3]

Институт программных систем НАНУ, Киев, Украина

[1]egorov@samsonos.com , [2]suslova@samsonos.com, [3]doroshenkoanatoliy2@gmail.com

Введение

В последнее время web-приложения приобретают все больше популярности. С развитием интернета все острее встает проблема разработки интернет приложений с легко масштабируемым и изменяемым кодом. Главную роль в возможности такой масштабируемости играет архитектура проекта. Чаще всего при разработке интернет приложений используется архитектура Модель – Вид – Контроллер (Model-view-controller, MVC). Это схема использования нескольких шаблонов проектирования, с помощью которых модель данных приложения, пользовательский интерфейс и взаимодействие с пользователем разделены на три отдельные части [1]. Модель содержит объектную модель, бизнес-логику и доступ к данным, вид содержит визуальный способ отображения данных, генерацию страниц, а контроллер – маршрутизацию запросов. Таким образом, основная цель применения этой концепции состоит в разделении бизнес логики от ее визуализации. Также данная схема проектирования часто используется для построения архитектурного каркаса, когда переходят от теории к реализации в конкретной предметной области.

Ряд разработчиков специализируется только в одной из областей: либо разрабатывают графический интерфейс, либо разрабатывают бизнес-логику. Таким образом, представляется возможным добиться того, что программисты, занимающиеся разработкой бизнес-логики (модели), вообще не будут осведомлены о том, какое представление будет использоваться.

Вид и контроллер обычно поддерживаются различными фреймворками и библиотеками [2], [3], [4]. Тогда как разработка модели ложится на разработчика.

Целью исследования было изучение различных сред разработки и создание своей собственной, которая бы позволила существенно облегчить жизнь разработчика web-приложений посредством автоматизации множества рутинных процессов, предоставления возможностей минимизации кода. Стремясь создать среду, которая бы соответствовала поставленным требованиям, был разработан фреймворк SamsonPHP и шаблон WMVC – WEB Model-View-

Controller pattern, который наследует идеи стандартного паттерна MVC и учитывает особенности разработки в среде web.

За счет такого разделения повышается возможность повторного использования кода. Наиболее полезно применение данной концепции в тех случаях, когда пользователь должен видеть те же самые данные одновременно в различных контекстах и/или с различных точек зрения.

Благодаря разделению, выполняются следующие задачи. К одной модели можно присоединить несколько видов, не затрагивая при этом реализацию модели. Например, некоторые данные могут быть одновременно представлены в виде электронной таблицы, гистограммы и круговой диаграммы. Не затрагивая реализацию видов, можно изменить реакции на действия пользователя (нажатие мышью на кнопке, ввод данных), для этого достаточно использовать другой контроллер. Также повышается производительность конечных решений благодаря использованному методу автоматической генерации приложения в один скрипт [5].

Samson PHP

SamsonPHP — это фреймворк, написанный на языке программирования PHP и предназначенный для быстрого и эффективного создания веб-сайтов и веб-приложений, используя язык программирования PHP. Веб-сайты создаваемые с использованием SamsonPHP должны реализовывать шаблон программирования WMVC (Model-view-controller/Модель-представление-контроллер).

Также SamsonPHP использует принцип convention over configuration, который предписывает использование заранее установленных правил и настроек, а не полноценное конфигурирование под каждый создаваемый веб-сайт, но разработчик может переопределить каждое значение параметра фреймворка для собственных потребностей.

В основе разработки фреймоворка лежит принцип программирования DRY (don’t repeat yourself / не повторяй себя), который кардинально изменил подход к созданию веб-сайтов и веб-приложений, и сильно отразился на структуре самого фреймворка.

Модуль — как основная часть веб-приложения

В SamsonPHP два файла модели, контроллера и неограниченное количество файлов представлений которые относятся к одной сущности, принято называть – Модуль. По факту конечный веб-сайт является набором модулей, за вывод которых отвечает ядро фреймворка SamsonPHP.

Основная проблема существующих фреймворков – отсутствие полноценной модульности при разработке веб-сайта. Внутренняя

организация веб-серверов в наше время разрешает доступ только к файлам которые находятся внутри локальной директории веб-сайта (document root), тем самым ограничивая нас от использования внешних модулей. Поэтому все современные PHP фреймворки используют подход к собиранию модулей под конкретный веб-сайт либо под группу однотипных веб-сайтов с последующим физическим копированием этих файлов модулей в локальную папку каждого конкретного разрабатываемого веб-сайта. Такой подход к файловой структуре веб-сайта создает серьезные ограничения, трудности, а в некоторых случаях даже ошибки для программиста, потому-что приходится вручную или полуавтоматически подключать эти модули к веб-сайту, контролировать их версионность и в случае выхода обновления модуля, копировать его в каждый веб-сайт который его использует.

Язык программирования PHP позволяет на серверной стороне подключать файлы, которые физически находятся вне файловой структуры веб-сайта, а конкретно вне его локальной директории.

В терминологии SamsonPHP модуль, это не только набор из контроллера, модели и файлов представления, а также любых других ресурсов, таких как JavaScrip, CSS, картинки и любые другие файлы. По сути, каждый модуль SamsonPHP является мини веб-сайтом и может существовать отдельно, если он для этого достаточно подготовлен, а именно имея свой контроллер и шаблон представления. Этот новый подход к пониманию модульности в разработке веб-приложений, даёт нам полное право создавать отдельные общеиспользуемые части веб-сайта, выделяя их в полноценные модули любой сложности.

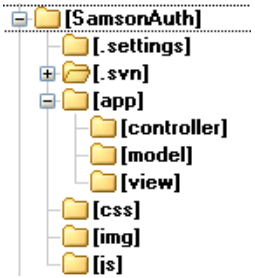


Рис. 1. Модуль SamsonAuth

Такой подход позволяет создавать библиотеки веб-ресурсов из которых конечный разработчик, сможет собрать готовый полнофункциональный веб-сайт, изменив лишь при надобности файлы представления и контроллера этих модулей.

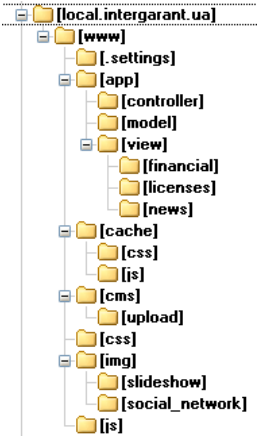


Рис. 2. Пример сайта, реализованного в SamsonPHP

Основным недостатком при физическом отделение файлов клиентских ресурсов Javascript, CSS, картинок является абсолютная невозможность их получения клиентским веб-браузером напрямую от сервера, т.к. веб-сервер не дает возможность обращаться к ресурсам веб-сайта вне его локальной директории. Для предоставления доступа к таким ресурсам во внешних модулях был разработан специальный механизм маршрутизации ResourceRouter – роутер ресурсов.

ResourceRouter

ResourceRouter(роутер ресурсов) – это внешний модуль SamsonPHP, который реализован в виде системного контроллера который обрабатывает URL запросы вида

- /src?a=@application&m=@module&r=@path

Комбинация @application и @module называется маршрутом к источнику ресурса. Для быстрого определения нужного маршрута из таблицы маршрутизации, в неё добавлен специальный хеш ключ(KEY) из него.

Механизм маршрутизации имеет следующий алгоритм, существует набор приложений(application), к которым относятся модули(module), внутри которых физически хранятся ресурсы. Основной частью ResourceRouter является таблица маршрутизации, в которой хранятся физическое размещение этих сущностей в файловой системе. Таблица заполняется и интерпретируется по следующему принципу:

- * означает любое значение параметра в маршруте
- вверху таблицы размещаются маршруты к приложению веб-сайта

- далее описываются маршруты к внутренним модулям веб-сайта
- далее описываются маршруты к внешним модулям веб-сайта
- последней строкой таблицы должен быть маршрут к локальным ресурсам

Таблица 1. Маршрутизация ресурсов

Key	Application	Module	Path
f3df58ef20f88671c61da39e4c008c15	*	system	E:\PHPPROJECTS\SamsonPHP/
2791548c939e9928b8b829a33cb223c4	local	about	about/
e9986a0a7dc8314d4055c7c13cc57494	local	blog	blog/
2919674d7c568d5967dbb9d511bcb3cd	*	auth	E:\PHPPROJECTS\SamsonAuth/
d861768d27f08f1ab398f29b9baa7611	local	*	

Теперь для завершения необходимо все пути к ресурсам внутри представлений внешних модулей и приложений заменить на специальную сокращенную функцию фреймворка

- src($path, $module = current)

которая автоматически определит, из какого модуля и приложения она вызывается и сформирует правильный URL маршрут к ресурсу:

- /src/?r=img/ajax_loader_1.gif&a=*&m=auth.

Если один из параметров маршрута упущен, то он автоматически приравнивается к “*”.

Пример разбора URL маршрута:

1. Указано приложение — *, это означает, что этот маршрут не относится к конкретному приложению, а значит, значимой частью является только модуль(auth)
2. Находим строку с Application – “*”, Module – “auth”.
3. Получаем, что физически ресурсы находятся по пути «E:\PHPPROJECTS\SamsonAuth/»
4. Добавляем физическому пути относительное размещение ресурса внутри модуля «img/ajax_loader_1.gif»
5. Получаем путь к ресурсу: «E:\PHPPROJECTS\SamsonAuth\ img\ajax_loader_1.gif»
6. Контроллер считывает содержимое определенного ресурса, если он физически существует и возвращает его, указав нужный HTTP заголовок mime-type для браузера, в случае его отсутсвия возвращается заголовок 404 ошибки

Таким образом, полностью решается задача о получении клиентского доступа к ресурсам внешних модулей веб-сайта.

PHPCompressor

PHPCompressor (сжиматель PHP) – это внешний модуль SamsonPHP, который служит для преобразования веб-сайта из среды разработки в финальный продукт, готовый к загрузке на веб-сервер. Такое преобразование необходимо для того, чтобы не копировать всю файловую структуру которая используется при разработке веб-сайта, а так же для того, чтобы веб-сайт функционировал на порядок быстрей чем его распределенная версия.

В SamsonPHP процесс перевода веб-сайта из стадии разработки в финальную рабочую стадию называют «сжатием».

Данный модуль выполняет лексические преобразования и выполняет фактическое собирание всего необходимого для функционирования веб-сайта в один финальный пусковой файл index.php. Все преобразования и операции копирования, создания, переименования ресурсов выполняются автоматически. Также специальный текстовый процессор, встроенный в модуль, имеет возможность убирать и изменять PHP код модулей, которые имеют в себе специальные директивы для оптимизации работы этих фрагментов кода в сжатой версии сайта.

Список использованной литературы

1. Web-приложения и данные: проблемы абстракции и масштабируемости / Посконин А. – Труды ИСП РАН, 2012. – 159 с.
2. «Zend Framework», [В Интернете]. URL: http://framework.zend.com/
3. «Code igniter», [В Интернете]. URL: http://ellislab.com/codeigniter
4. «Slim PHP», [В Интернете]. URL: http://www.slimframework.com/
5. Исследование методов увеличения производительности web-приложений / Ботыгин И. А. – Известия ТПУ, 2008. – №5. – 109 с.
6. Опыт создания модуля адресного доступа к контенту новостной страницы портала СДО ГОУ ВПО "ЮРГУЭС". / Лантратов О.И., Самоделов А.Н., Чумаков С.Е. – Современные проблемы науки и образования, 2010. – №6. (приложение "Технические науки"). – 5 с.
7. Архитектура семантического Web-портала / Тузовский А. Ф. – Известия ТПУ, 2006. – №7. – 142 с.

ON A STRONG NOTION OF CAUSALITY IN NONDETERMINISTIC INPUT-OUTPUT SYSTEMS

Ie. V. Ivanov

Taras Shevchenko National University of Kyiv, Ukraine
Paul Sabatier University, Toulouse, France

ivanov.eugen@gmail.com

Introduction

On an abstract level an information processing system can usually be viewed as a kind of input-output relation, where the inputs and outputs are (tuples of) certain time-varying quantities (signals). This view is frequently used in systems theory and cybernetics [1].

When one collection of input signals has only one corresponding collection of output signals, one can call such an input-output (I/O) system deterministic. An important property of deterministic I/O systems that is necessary for their physical (and real-time) implementation is causality [1]. The idea of this notion is that the value of an output signal at any time t can depend only on the values of the input signals at times $t' \le t$. Specific variants of this definition can be found in [1, 2], but assuming a system has one input and output signal and is represented as a function f: $S \to S$, where $S = T \to W$ is the set of all signals represented as total functions from a non-negative real ($T=[0,+\infty)$) or integer ($T=\{0,1,2,\ldots\}$) time domain T to some set of values $W \neq \varnothing$, the condition of causality can be expressed as follows:

Definition 1. A system f is causal, if for each input signals $i_1, i_2 \in S$ and time $t' \in T$, if $i_1(t) = i_2(t)$ for all $t \le t'$ and $o_1 = f(i_1)$, $o_2 = f(i_2)$ are the corresponding output signals, then $o_1(t) = o_2(t)$ for all $t \le t'$.

This definition gives a rise to the question of whether a non-deterministic system can be causal. This question was investigated in the literature [2-4]. In particular, in the work [2] the authors argue that non-determinism and causality are independent properties of input-output systems and propose the following notion of a non-deterministic causal system. Assume that a non-deterministic input-output system is represented as a (multi)function F: $S \to 2^S \setminus \{\varnothing\}$, where S is a set of signals as we described above. Then the condition of a causal (or non-anticipative) system can be expressed as follows:

Definition 2. A system F is causal (or non-anticipative), if for each input signals $i_1, i_2 \in S$ and time $t' \in T$, if $i_1(t) = i_2(t)$ for all $t \le t'$, then $\{ o|_A \mid o \in F(i_1) \} = \{ o|_A \mid o \in F(i_2) \}$, where $A=[0, t']$ and $o|_A$ denotes a restriction of a function (output signal) on a set.

It is clear that in deterministic case, i.e. if $F(i)$ is a singleton set for each i , this definition is equivalent to Definition 1.

Strong non-anticipation

Although Definition 2 seems to be a natural generalization of the notion of a causal deterministic system to the nondeterministic case, it has certain counter-intuitive consequences which we will describe below. Assume that W (the set of signal values) coincides with the set of real numbers $\mathbf{R}$. Let g: $\mathbf{R}\to\mathbf{R}$ be an arbitrary function.

Definition 3. An input-output system represented by a (multi)function F: $S\to 2^S \setminus \{\varnothing\}$ is called a g-limit system, if the following conditions hold for any input signal $i\in S$:

(1) if a limit $L=\lim_{t\to+\infty} i(t)$ exists and is finite, then $F(i)$ is the set of all signals $o\in S$ such that $\lim_{t\to+\infty} o(t) = g(L)$;

(2) if a limit $\lim_{t\to+\infty} i(t)$ does not exist or is infinite, than $F(i)=S$.

Lemma 1. If F is a g-limit system for some g: $\mathbf{R}\to\mathbf{R}$, then F is causal in sense of Definition 2.

When a function g is discontinuous, intuitively, this result seems to contradict the idea behind the notion of a causal system. This idea is that at any time such a system does know the future values of the input signals and thus cannot use them to determine the current output value. But consider, for example, the case when g is the signum function (i.e. $g(0)=0$, $g(x)=1$, if $x>0$, and $g(x)<0$, if $x<0$). Then a g-limit system outputs a signal which converges to 1 (as $t\to+\infty$) whenever the input signal converges to a positive number (as $t\to+\infty$). Moreover, it outputs a signal which converges to 0 whenever the input signal converges to 0. This implies that when a system receives a decreasing positive input signal which tends to 0, it decides to output values which are close to 0 starting from some time t. Intuitively, after reading the input signal until time t, the system decides that 0 is a more likely limit of the input signal than a positive value, but such a decision cannot be based on the past values of the input signal, so it requires some knowledge of the future of the input signal.

These observations can be formalized as follows. Consider two I/O systems represented by (multi)functions F_1, F_2: $S\to 2^S \setminus \{\varnothing\}$.

Definition 4. F_1 is a refinement of F_2, if $F_1(i) \subseteq F_2(i)$ for all $\in S$.

Lemma 2. Let F is a g-limit system for some g: $\mathbf{R}\to\mathbf{R}$. Then F has a deterministic causal refinement iff g is continuous.

Here a deterministic causal refinement is (single-valued) choice function (selector) of the multifunction F which satisfies Definition 1.

This proposition implies that for a discontinuous function g, a g-limit system has no deterministic causal refinement. Informally, this means that there is no way one can choose a single output signal (system's response) for each input signal from the possibilities offered by a system in such a way that output signal are causally related with input signals.

In order to formalize the idea behind the notion of a causal system (no prior knowledge of the future) in nondeterministic case more precisely, let us introduce the following notion.

Definition 5. A system F is strongly causal (or strongly non-anticipative), if for each pair i, o such that $o \in F(i)$ there exists a deterministic causal refinement F' of F such that $o \in F'(i)$.

This definition requires the operation of a system to be determined by the operations of its deterministic causal refinements. Informally, the operation of a strongly causal system F can be interpreted as a two-step process:

(1) before receiving the input signals, the system F (non-deterministically) chooses a deterministic causal refinement F' (one can call this a response strategy);

(2) the system F' receives input signals of F and produces the corresponding output signals (response) which become the output signals of F.

Informally, it is clear that in this scheme at any time the system F does not need to know the future of its input signals in order produce corresponding output signals.

Theorem 1. If a nondeterministic system F is strongly causal, then it is causal in sense of Definition 2. The converse does not always hold.

References

1. Oppenheim A. Signals and Systems / A. Oppenheim, A. Willsky, S. Nawab. – Pearson Education, 1998.
2. Foo N. Realization for Causal Nondeterministic Input-Output Systems / N. Foo, P. Peppas // Studia Logica. – 2001. – Vol. 67. – P. 419 -437.
3. Windeknecht T. Mathematical systems theory: Causality / T. Windeknecht // Mathematical Systems Theory. – 1967. – Vol. 1. – P. 279-288.
4. Arbib M. Automata theory: the rapprochement with control theory / R. Kalman, P. Falb and M. Arbib. – Topics in Mathematical Systems Theory. – McGraw Hill, 1969.

ОБ'ЄКТНИЙ АНАЛІЗ МОДЕЛЮВАННЯ МОДЕЛЕЙ ПРЕДМЕТНИХ ОБЛАСТЕЙ З ВИКОРИСТАННЯМ ТЕОРІЇ ФРЕГЕ

А.Л. Колесник

Інститут програмних систем НАН України, 03187,
Київ, проспект Академіка Глушкова, 40,

swabber@gmail.com

Вступ

Процес побудови моделі предметної області (ПрО) або домену, орієнтованої на її розуміння людиною і тому його звуть концептуальним моделюванням. Проблема, яку моделюють ПрО, формулюється у просторі її понять або на перехресті декількох пов'язаних між собою областей [1-3].

Найпершим аспектом моделювання домену усіх відомих підходів може вважатися його понятійна база, тобто система понять, за допомогою якої формулюються усі аспекти проблеми. У ментальній діяльності людини застосовується ряд відношень, які дозволяють будувати похідні поняття та встановлювати між ними зв'язки. Серед таких відношень назвемо найпоширеніші:

– узагальнення є звуження істотних ознак поняття, при цьому розширюється коло охоплених поняттям об'єктів, тобто його обсяг;

– конкретизація є додавання істотних ознак, завдяки чому зміст поняття розширюється, а обсяг поняття звужується;

– агрегація є об'єднання ряду понять у нове поняття, істотні ознаки нового поняття при цьому можуть бути або сумою ознак компонент або суттєво новими.;

– асоціація є найбільш загальним відношенням, що утверджує наявність зв'язку між поняттями, не уточнюючи залежність їх змісту та обсягів.

Для моделювання моделей програмних систем (ПС) з об'єктів в рамках наукової тематики ІПС НАНУ визначено формальні концепції: узагальнення понятійній структури предметної області ПС на основі теорії Фреге; теоретико-множинного впорядкування об'єктів (об'єднання, різниці, декартового добутку тощо); логіко-алгебраїчного подання множин об'єктів як алгебраїчної системи на сукупності предикатів певної сигнатури; поведінки об'єктів в залежності від терміну їх станів і поведінки.

Сутність об’єктного проектування доменів

Об’єктно-орієнтований підхід визначає стратегію побудови об’єктної моделі (ОМ) ПрО згідно теорії Г.Буча, відповідно якої увись «світ» складається з об’єктів [1].

Визначення об'єкту, як базисного поняття об’єктної моделі ОМ предметної області, застосовується на двох припущеннях:

– для кожного об'єкта однозначно визначається його відповідність до сутності ПрО, включаючи відображення необхідних зв'язків та особливостей поведінки;

– засіб подання ПрО за об'єктним підходом визначає міру повноти відображення сутності та поведінки як об'єкту, або міру адекватності його сприйняття замовником.

Повнота відображення сутності, що задається об'єктом, відповідає мірі абстракції та її відображенню у час його опису. Для одночасного урахування обох вимог у запропонованої концепції запроваджується понятійна структура, що відповідає поняттю трикутника Фреге (рис.) згідно якої об'єкт є *денотатом*. Символ використовується для ідентифікації об'єкту, а денотат відповідає рівню знань виконавця про сутність модельованого світу, що відбивається об'єктом.

Природно, що денотат можна ідентифікувати різним чином, використовуючи для цього обрані алфавіти, та одному об'єкту можуть відповідати декілька концептів, відображаючи обраний рівень абстракції. Кожному трикутнику Фреге у обраній логічній системі буде відповідати певний об'єкт з власним ім'ям та певним змістом.

Тобто о**б’єкт** – це іменована частина дійсної реальності з певним рівнем абстракції відносно вибраної ПрО та з цілком обумовленої поведінкою. Поданням об’єкту є понятійна структура у вигляді трикутника Фреге.

Об'єкт як понятійна структура відповідно трикутнику Фреге, складається з власного ідентифікатору, денотату, чи то образу предмету або абстрактного елементу, на який вказує цей ідентифікатор, і концепту, що визначає змістовність цього денотату, виходячи з мети концептуального моделювання. Кожний денотат та концепт об’єкту визначається декомпозиційним чи композиційним шляхом.

Рис. Трикутник Фреге

Під декомпозиційним поданням денотату розуміється сутність дійсної реальності, яка відповідає певному об'єкту і подається як сукупність однорідних чи неоднорідних предметів.

Композиційне визначення денотату це кількість однорідних чи неоднорідних предметів з вибраної предметної області.

Концепти об'єктів формуються на основі концепту початкового об'єкту або без врахування його. Концепт нового композиційного об'єкту формується на основі однакових чи різних концептів початкових об'єктів.

Зміни рівня деталізації (абстракції) концептів визначається операціями включення чи виключення однієї чи кількох властивостей при формуванні концепту нового об'єкту з однаковим денотатом. Кожна з операцій має певний пріоритет та арність, а також пов'язана з відповідними допустимими змінами денотатів та концептів.

Нехай $O=(O_1, O_2, \ldots O_n)$ – множина об'єктів на певному рівні об'єктного аналізу і

$O_i = O_i(Name_i, Den_i, Con_i)$, де $Name_i$, Den_i, Con_i – знак (ім'я), денотат та концепт відповідно. $P=(P_1, P_2, \ldots P_r)$ – множина предикатів, на основі яких визначаються концепти об'єктів – $Con_i = (P_{i1}, P_{i2}, \ldots, P_{is})$.

До базових операцій відносяться операції декомпозиції і композиції об'єктів:

decomds (O_i): $O_i \rightarrow (O_{i1}, \ldots O_{ik})$,

decomdn(O_i): $O_i \rightarrow (O_{i1}, \ldots, O_{ik})$,

comds$(O_{i1}, \ldots O_{ik}): (O_{i1}, \ldots O_{ik}) \rightarrow O_i$,

comdn $(O_{i1}, \ldots O_{ik}): (O_{i1}, \ldots O_{ik}) \rightarrow O_i$,

Розширення концепту. Якщо $(P_t \in P)$, $(P_t \notin Con_i)$ і $P_t(O_i)$ приймає значення істини, то

conexp$(O_i, P_t): O_i \rightarrow O_i`$,

де $O_i` = O_i`(Name_i, Den_i, Con_i`)$, $Con_i \cup \{P_t\} = Con_i`$

Звуження концепту. Якщо $P_t \in Con_i$, то connar $(O_i, P_t): O_i \rightarrow O_i`$,

де $O_i` = O_i`(Name_i, Den_i, Con_i`)$, $Con_i` = Con_i \setminus P_t$

При розгляду об'єктів ПрО використовується принцип відміни кожного окремого елементу ПрО з припущенням, що він визначається через його особливість – зовнішню і внутрішню.

Об'єкт, як певна сутність ПрО перебуває в різних станах і має певний набір операцій, які надають іншим об’єктам послуги (сервіси) для виконання методів. Об'єкти поділяються по класам за їх особистими властивостями і в суперкласи за загальними характеристиками (feature).

Кожний об'єкт має власний стан і набір операцій, які впливають на стан інших об'єктів. Об'єкти приховують інформацію про значення станів, операцій і обмежують доступ до них. Перехід від об'єктної моделі до моделі ПС для об'єктів визначаються інтерфейси, які відображають зв'язки між об’єктами, що реалізують їх методи.

До базових концепцій проектування ОМ відносяться такі:

– концепція узагальнення понятійній структури ПрО, що відповідає формальному поняттю трикутника Фреге;

– теоретико-множинна концепція з операціями над об’єктами (об’єднання, різниці, декартового добутку тощо);

– логіко-алгебраїчна концепція для опису множини об’єктів як алгебраїчної системи з визначеною сукупністю предикатів певної сигнатури;

– концепція поведінки об’єктів в залежності від терміну їх функціонування та сформованих ними станів, що фіксуються в діаграми переходів станів і використовуються при виконанні об’єктів.

Ці концепції реалізуються на рівнях проектування ОМ з забезпеченням її повноти та коректності. Головною рисою рівнів проектування є зовнішні та внутрішні характеристики об’єктів та їх опис на заданій множині об’єктів з встановленням типів взаємовідношень: спеціалізація, агрегація, класифікація, асоціація, екземплярізація тощо.

Математичні операції проектування ОМ:

операція об'єднання U(A,B), де *A* і *B* – об'єкти з статусом множин. Результат використання: новий об'єкт-множина, що є об'єднанням множин *A* та *B;*

операція перехрещення П(А,В), де *А* і *В* – об'єкти-множини. Результат використання: новий об'єкт-множина, що є перехрещенням множин *А* та *В*;

операція різниці D = А П В при *В* $\subset$ *А*. Результат використання: новий об'єкт-множина з усіма елементами *А*, які не входять у *В*;

операція симетричної різниці S = А В, де *А П В*, та ні *А* $\subset$ *В*, ні *В* $\subset$ *А*. Результат використання: новий об'єкт-множина з множиною елементів, які належать *А* або ("або" – розподільне) *В*.

Чотирирівніве проектування об'єктної моделі

При виконанні операцій проектування ОМ визначаються нові об'єкти і проводиться структурна їх впорядкованість відповідно висхідної технології.

Процес аналізу ПрО та побудови ОМ виконується за чотирма рівнями абстракції відповідно вищезазначеним аспектам концепцій. Їх сутність розглядаються нижче.

Узагальнюючий рівень проектування ОМ. Цей рівень подає найвищу міру абстракції відображення ПрО у ОМ з використанням розглянутої концепції узагальнення з метою визначення сутностей ПрО та подання їх у вигляді об'єктів ОМ ПрО як базових понять моделі. Отриманий результат – це специфікації об'єктів ПрО з структурою: <ім'я об'єкту> <концепт>,

де *ім'я об'єкту* – ідентифікатор з символьного рядку літер та десяткових цифр;

концепт – текст, що визначає <денотат об'єкту>.

Структурно-впорядковуючий рівень. Об'єкти вже визначені на узагальнюючому рівні абстракції. Кожний об'єкт подається як множина або елемент множини. Нехай $O'=(O_1, O_2, \ldots O_n)$, $OS= (OS_1, OS_2, \ldots OS_m)$ і $OS \subseteq O'$. Тоді IS: $O' \rightarrow OS$ є обмеження тотожного відображення O' на себе. Нехай $S=(S_1, S_2, \ldots S_m)$ – множина об'єктів, що визначені на даному рівні, і TS: $OS \rightarrow S$, де кожному OS_i відповідає S_i. Тоді визначається відображення між об'єктами на узагальнюючому та структурно-впорядковуючому рівнях *TS*IS:* $O' \rightarrow S$.

Поведінковий рівень. Об'єкти вже визначені на характеристичному рівні абстракції. Визначаються життєві цикли об'єктів (послідовність станів об'єктів в залежності від параметра часу). Нехай $B=(B_1, B_2, \ldots B_k)$ – множина об'єктів, які визначені на поведінковому рівні, і кожному A_i відповідає B_i $(A_i \rightarrow B_i(t))$. Тоді визначається відображення ТВ між об'єктами на характеристичному та поведінковому рівнях ТВ: А $\rightarrow$ В.

Висновок

Представлено об'єктне проектування доменів (предметних областей), Розглянуто чотири рівні проектування ОМ. На першому рівні використовуються механізми Фреге для визначення основних понять об'єкта – ім'я, денотат та концепт з використанням операції над денотатами і концептами. На інших рівнях формуються властивості і характеристики об'єктів, їх структурне упорядкування та поведінка.

Список використаних джерел

1. *Лаврищева Е.М., Грищенко В.Н.* Сборочное программирование. Основы индустрии программных продуктов.– К.: Наук.думка, 2009.– 377 с.
2. *Лавріщева К.М.* Взаємодія програм, систем й операційних середовищ // Проблеми програмування, №3, 2011.– с.13–24.
3. *Лавріщева К.М., Слабоспицька О.О., Коваль Г.І., Колесник А.О.* Теоретичні аспекти керування варіабельністю в сімействах програмних систем.–Вісник КНУ, серія фіз.–мат. наук. – 2011. – №1. – С. 151–158.
4. *К.М. Лавріщева, Г.І. Коваль та др.* Нові теоретичні засади технології виробництва сімейств програмних систем у контексті ГП – Електронна монографія, ДРНТІ.– №67–УК–2011 від 5.10.11.–377 с.
5. *Лаврищева К.М., Колесник А.Л.* Концептуальні моделі розподілених програмних систем // Проблеми програмування, №2, 2013.– с. 3–18.

ФОРМАЛЬНЫЕ МЕТОДЫ ВЕРИФИКАЦИИ НА ОСНОВЕ СЕТЕЙ ПЕТРИ

С.Л. Крывый[1], А.Н. Максимец[2]

Киевский национальный университет имени Тараса Шевченка, Киев, Украина

[1]*sl.krivoi@gmail.com,* [2]*maksymets@gmail.com*

Введение. Данная работа является результатом построения экспериментальной системы верификации структурных свойств параллельных и распределенных систем на основе использования сетей Петри (СП). В данной работе пердставлен технологический процесс и методы анализа СП.

Сети Петри. Определение 1. *Сетью Петри* (СП) называется тройка (P,T,F), где P и T непересекающиеся множества, элементы которых называются *местами* и *переходами* соответственно, а $F \subseteq (P \times T) \cup (T \times P)$ – отношение инцидентности. Элементы из F называются стрелками, а места – вершинами, имея в виду графическое представление сети. Если $(x,y) \in F$, то x называется входной вершиной y, а y – выходной вершиной x. Множества входных и выходных вершин для x обозначается ${}^{\bullet}x$ и $x^{\bullet}$ соответственно.

Сеть (P,T,F) называется размеченной, если задана функ-ция разметок $M : P \to N$, где N – множество натуральных чисел. Если $M(p) = m$, то это значит, что функция M ставит в вершину p ровно m фишек. Разметка сети, в случае когда $|P| = n$ и множество P упорядочено, представляется вектором $M = (m_1, m_2, \ldots, m_n)$, где $m_i = M(p_i)$, $p_i \in P$, $i = 1,2,\ldots,n$.

Сетью Петри называется четверка (P,T,F,W,M_0), где (P,T,F) – сеть, M_0 – начальная разметка ее мест, а $W : F \to N \setminus \{0\}$ – функция кратности дуг СП [2].

Классы СП и их свойства. Приведём классификацию СП и методы исследования их важнейших свойств. С этой целью введем некоторые определения общего характера.

Определение 2. *Пусть* (P,T,F,W,M_0) *некоторая СП, а* $Q \subseteq P$, $Q \neq \varnothing$ *– подмножество её мест. Множество* Q *называют*

а) **дедлоком (сифоном)** тогда и только тогда, когда ${}^\bullet Q \subseteq Q^\bullet$;

б) **ловушкой** тогда и только тогда, когда $Q^\bullet \subseteq {}^\bullet Q$.

Анализ структурных свойств СП. Рассмотрим классы СП, в частности, так называемые чистые СП, и методы анализа их структурных свойств. Структурные свойства СП – это свойства, которые не зависят от начальной разметки СП в том смысле, что они выполняются для произвольной начальной разметки СП.

Определение 3. *СП* (S, W, M_0) *называется* **чистой**, *если из того, что* $(p,t) \in F$ *вытекает что* $(t,p) \notin F$ *или* ${}^\bullet t \cap t^\bullet = \varnothing$.

К структурным свойствам чистых СП относятся следующие свойства [6]:

- структурная живучесть;
- управляемость;
- структурная ограниченность;
- консервативность и частичная консервативность;
- повторяемость и частичная повторяемость;
- непротиворечивость и частичная непротиворечи-вость.

Определение 4 . *СП называется*

– **структурно живой**, если существует живая начальная разметка этой сети;

– **полностью управляемой**, если произвольная разметка достижима в этой СП из произвольной другой ее разметки;

– **структурно ограниченной**, если она ограничена(т.е. $\exists k \in \mathrm{N}\ \forall p \in P : M(p) \le k$) для любой конечной начальной разметки этой сети;

– **консервативной**, если в процессе ее функционирования суммарное число фишек в сети остается постоянным (т.е. для любых двух достижимых разметок M и M_1 имеет место $\sum_{p \in P} M(p) = \sum_{p \in P} M_1(p)$);

– (частично) **повторяемой**, если существует начальная разметка M_0 СП и последовательность срабатываний переходов σ такие, что каждый (некоторый) переход принадлежит к σ бесконечно часто;

– (частично) **непротиворечивой**, если существует начальная разметка M_0 СП и последовательность срабатываний переходов σ, ведущая из M_0 к M_0, такие, что каждый (некоторый) переход встречается в σ хотя бы один раз.

Методы анализа этих свойств зависят от класса СП, для которых они исследуются. Классификация СП выполняется путем введения определенных ограничений на СП.

Определение 5. *СП* (P,T,F,W,M_0) *называется*

– ***машиной состояний*** *(МС)* $\Leftrightarrow \ \forall t \in T \ \ | t^\bullet | = | ^\bullet t | \le 1$;

– ***синхрографом*** *(СГ)* $\Leftrightarrow \ \forall p \in P \ \ | p^\bullet | = | ^\bullet p | \le 1$;

– ***свободного выбора*** *(СВ)* $\Leftrightarrow \forall p, p' \in P$

$p \ne p' \to (p^\bullet \cap p'^\bullet = \varnothing \vee | p^\bullet | = | p'^\bullet | \le 1)$;

– ***расширенной*** *СП свободного выбора (РСВ)* $\Leftrightarrow$

$\forall p, p' \in P \ \ (p \ne p' \to (p^\bullet \cap p'^\bullet = \varnothing \vee p^\bullet = p'^\bullet)$;

– ***простой*** *СП (ПСП)* $\Leftrightarrow \quad \forall p, p' \in P$

$(p \ne p' \to (p^\bullet \cap p'^\bullet = \varnothing \vee | p^\bullet | \le 1 \vee | p'^\bullet | \le 1))$;

– ***асимметричной*** *СП (АСП)* $\Leftrightarrow$

$\forall p, p' \in P \ \ (p \ne p' \ p^\bullet \cap p'^\bullet = \varnothing \vee p^\bullet \subseteq p'^\bullet \vee p'^\bullet \subseteq p^\bullet)$.

Для этих классов сетей, как следует из определений, имеют место такие включения:

СГ $\subset$ СВ; МС $\subset$ СВ; СВ $\subset$ РСВ $\subset$ АСП; СВ $\subset$ ПСП $\subset$ АСП.

Поиск дедлоков и ловушек в СП. Поиск делоков и ловушек и основанный на них метод анализа живучести СП базируется на использовании методов решения диофантовых ограничений.

Напомним, что когда произвольный переход имеет выходное место, принадлежащее дедлоку Q, то в Q должно включаться и его выходное место. А для ловушки наоборот. Из этого определения вытекает, что когда дедлок Q является пустым множеством, то он всегда остается пустым множеством в процессе функционирования СП. Для ловушки ситуация противоположная: если хотя бы одно ее место получило фишку, то ловушка постоянно остается непустым множеством в процессе функционирования СП.

Дедлоки и ловушки можно найти в СП с помощью решения системы логичных уравнений, которые описывают эти свойства, или эквивалентной ей системы линейных неравенств над множе-ством $\{0,1\}$.

Ловушка называется помеченной начальной разметкой, если хотя бы одно место этой ловушки получает, по крайней мере, одну фишку. Очевидно, что объединение двух дедлоков (ловушек) снова будет дедлоком (ловушкой). Дедлок (ловушка) называется базисным (-ой), если он (она) не может быть представлена в виде объединения других

дедлоков (ловушек). Все дедлоки (ловушки) СП можно породить с помощью объединения базисных дедлоков (ловушек). Дедлок (ловушка) называется минимальным (-ой), если он (она) не включает других дедлоков (ловушек). Минимальные дедлоки (ловушки) являются базисными, но не все базисные дедлоки (ловушки) являются минимальными.

Важность построения множеств дедлоков и ловушек для данной СП состоит в том, что с помощью анализа полученных множеств можно определить, жива ли данная СП или нет. Это вытекает из таких утверждений.

СП свободного выбора как и расширенная СП свободного выбора живая тогда и только тогда, когда каждый дедлок этой СП включает ловушку, помеченную начальной разметкой M_0.

При использовании СП для верификации свойств систем очень важным является определение пути, ведущего к ошибке или к подозрительному месту или переходу в СП. Этот путь скрыт в Т-инвариантах, однако Т-инварианты содержат информацию только о том, какие переходы срабатывают, но не несут никакой информации о том, в какой последовательности они срабатывают. Если установлено, что СП ограничена, то граф достижимых разметок такой СП является конечным автоматом. Применяя к этому автомату алгоритм анализа, построим регулярное выражение, а по нему определяются кратчайшие пути, ведущие в подозрительные места СП.

Технологический процесс исследования свойств СП. Исходя из сказанного выше, предлагается следующий техноло-гический процесс исследования свойств СП. Этот процесс основывается на методах линейной алгебры, теориях графов, конечных автоматов и формальных языков.

Многие свойства СП проверяются средствами линейной алгебры, в частности, средствами решения систем линейных уравнений и неравенств в области натуральных чисел. Вторая группа средств – это средства построения и обработки графов и деревьев. И третья группа – это средства работы с конечными автоматами и языками (в частности, с регулярными языками).

Первая группа средств базируется на оригинальном методе построения минимального порождающего множества решений систем линейных уравнение в области натуральных чисел. Этот метод решения назван TSS-методом и описан в [1, 3]. С помощью этого алгоритма генерируются инварианты СП без какой либо дополнительной обработки. Использование этого алгоритма позволяет решить проблемы:

а) достижимости в подклассе СП свободного выбора,

b) недостижимости разметки в общем случае,

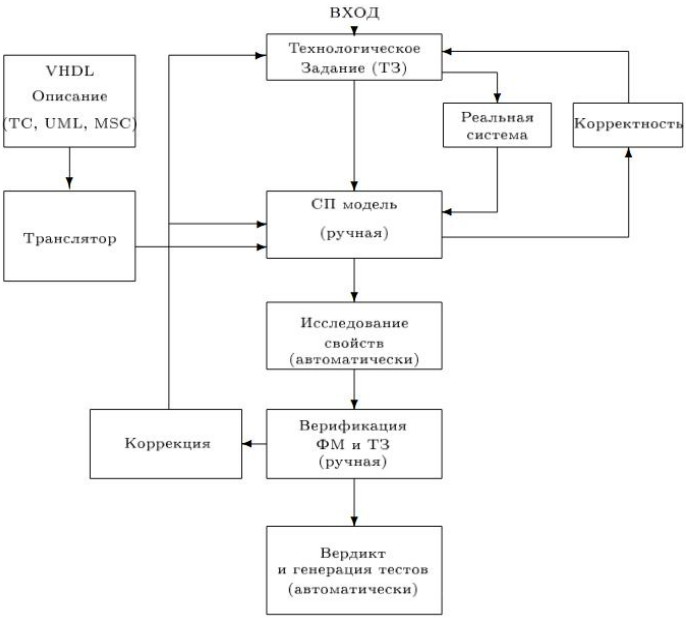
c) ограниченности СП для единственного ее места,

d) вычисления ловушек и дедлоков СП,

e) структурных свойств для данной СП.

Вторая группа средств базируется на хорошо разработанных алгоритмах построения, представления и обхода графов [4].

А третья группа тоже имеет хорошую алгоритмическую поддержку в виде алгоритмов анализа и преобразования регулярных языков и регулярных выражений, результаты представлены в [5].



РиРис. 1. Схема технологического процесса

Рис. 2. Конкретизация этапов анализа.

Заключение. Сравнение методов анализа СП с логическими методами анализа перечисленных свойств показывает что:

– некоторые свойства могут быть выражены СП более натурально и более просто, чем в логических формализмах;

– некоторые свойства (ограниченность, возможность возвра-та к начальному состоянию, определение дедлоков и ловушек) вовсе не могут быть выражены или очень трудно выражаются средствами логических формализмов;

– проверка соответствующих свойств средствами СП более простая.

Список использованных источников

1. Крывый С.Л. О некоторых методах решения и критериях совместности систем линейных диофантовых уравнений в области натуральных чисел. / ж. Кибернетика и системный анализ. – 1999. – N 4. – С. 12-36.
2. Hack M. H. T. Decidability Questions for Petri Nets. / Ph. D. Thesis, M.I.T. – 1976.
3. Kryvyi S. A Criteria of compatibility systems of linear diophantine constraints. / Lecture Notes in Computer Science. Spring. Verlag. – 2002. – 2328. – P. 264-271.
4. Jonathan L. Gross, Jay Yellen Handbook of Graph Theory. / CRC Press. – 2004. – P. 57-68.
5. Perrin D., Pin J-E. Infinite words: Automata, Semigroups, Logic and Games. – Elsevier: Academic Press. – 2004. – 538 p.
6. Murata T. Petri Nets: Properties, Analysis and Applications. / Proceed. of the IEEE. – 1999. – v. 77. – N 4. – P. 541-580.

ОНТОЛОГІЧНЕ ПОДАННЯ ЖИТТЄВОГО ЦИКЛУ ПС ДЛЯ ЗАГАЛЬНОЇ ЛІНІЇ ВИРОБНИЦТВА ПРОГРАМНИХ ПРОДУКТІВ

К.М. Лавріщева

Інститут програмних систем НАН України, 03187, Київ,
проспект Акад. Глушкова,40

lavryscheva@gmail.com

Анотація. В статті обгрунтований онтологічний підхід до подання моделі життєвого циклу стандарту ISO/IEC, включаючи опис загальних класів основних, організаційних та керуючих процесів. Для процесу тестування наведений онтологічний опис понять, концептів і задач в термінах системи Protégé, результатом якої є загальноприйнята мова XML. Ці онтологічні описи є базисом лінії виробництва програм за життєвим циклом в інструментально-технологічному комплексі (ІТК ІПС НАНУ) веб-сайту http://sestudy.edu-ua.net.

Вступ

Найпершим аспектом моделювання домену усіх відомих підходів може вважатися його понятійна база, тобто система понять, за допомогою якої формулюються усі аспекти проблеми. Понятійна база визначає не тільки термінологію, якою мають користуватися носії інтересів, що приймають участь у процесі аналізу вимог, а також суттєві відношення між поняттями та їх інтерпретацією [1]. Серед відношень назвемо найважливі:

– узагальнення для звуження істотних ознак поняття з розширенням коло понять об’єктів та їх обсягу;

– конкретизація, як додавання істотних ознак, завдяки чому зміст поняття розширюється, а обсяг його звужується;

– агрегація, як об'єднання ряду понять у нове поняття, істотні ознаки нового поняття можуть бути або сумою ознак компонентів або суттєво новими;

– асоціація, як найбільш загальне відношення, що утверджує наявність зв'язку між поняттями, не уточнюючи залежність їх змісту та обсягів.

Сукупність термінології, понять, характерних для них відношень та парадигми їхньої інтерпретації у межах домену прийнято називати *онтологією доменних знань*.

У самому загальному випадку, онтологія – це *угода про спільне використання понять, що включає засоби представлення предметних знань і домовленості про методи мірковувань*.

Онтології представляються семантичними мережами, у яких вузлами є поняття, а дугами – зв'язки чи відношення, асоціації між ними. На даний час онтологічні підходи отримали широке розповсюдження у вирішенні проблем представлення знань, семантичної інтеграції інформаційних ресурсів, інформаційного пошуку тощо. Зараз міжнародна спільнота пропонує різні форми подання і опису онтологій у вигляді, зручному для цілей розроблення ПС. Наприклад, *Web Ontology Language (OWL), Ontology-Driven Software Developmen (ODSD)* тощо. Вони дозволяють одержувати описи класів об'єктів предметної області, що відображають поняття та знання про них. Онтології деяких предметних областей подаються знаннями, словниками понять, концептів та відношень між ними. Для опису онтологій предметних областей використовуються різні мови OWL, XML, UML, BPL тощо. Аналіз шляхів моделювання предметної області засобами DSL показує наявність наступних систем: ODM (Organizational Domain Modeling), FODA (Feature-Oriented Domain Analysis), DSSA (Domain-Specific Software Architectures), Eclipse-DSL, DSL Tools VS.Net тощо.

Мова XML (Extensible Markup Language) фактично стала стандартом розмітки різноманітних даних предметних областей для їх зберігання і обміну між різними застосуваннями. Вона є засобом автоматичної трансформації описів моделей предметних областей в сучасних онтологічних мовами до XML-схем, придатних для роботи з ними різних прикладних застосувань.

В рамках викладання дисципліни «Програмна інженерія» в Київському національному університеті імені Тараса Шевченко студентами вивчені сучасні стандарти життєвого циклу (ЖЦ) ISO/IEC 12207 -2007 та загальних типів даних GDT (General Data Types) ISO/IEC 11404 – 2006, а також апробовані базові засоби загального опису онтологій: DSL Tools VS.Net та Protégé. На прикладі окремих фрагментів цих доменів онтологічними засобами побудовані відповідні моделі з метою їх використання в практичної діяльності при побудові програмних продуктів [2]. Деякими студентами кафедри ІС і ТТР захищені дипломні роботи по даної тематиці з використання онтологічних засобів для подання знань щодо цих доменів за допомогою понять - *класів, аксіом, слотів, фасетів* тощо.

В даної роботі викладається семантика цих доменів, їх онтологічний опис та продемонстровані результати в мові DSL з використання онтологічних систем. Для реалізації студентами фрагментів стандартного домену – життєвий цикл ПС використано DSL Tools VS.Net, а для його під домену – «процес тестування» використано систему Protégé [3, 4].

Онтологічний опис стандарту ЖЦ

Створення онтології ЖЦ стандарт ISO/IEC 12207. ЖЦ базується на загальної структурі змісту процесів поданих у табл.1. [2].

Таблиця 1. Процеси життєвого циклу в стандарті ISO/IEC 12207

№ п/п	Процес (підпроцес)
1. Категорія «Основні процеси»	
1.1	Замовлення (договір)
1.1.1	Підготовка замовлення, вибір постачальника
1.1.2	Моніторинг діяльності постачальника, приймання споживачем
1.2	Постачання (придбання)
1.3	Розроблення
1.3.1	Виявлення вимог
1.3.2	Аналіз вимог до системи
1.3.3	Проектування архітектури системи
1.3.4	Аналіз вимог до ПЗ системи
1.3.5	Проектування ПЗ
1.3.6	Конструювання (кодування) ПЗ
1.3.7	Інтеграція ПЗ
1.3.8	Тестування ПЗ
1.3.9	Системна інтеграція
1.3.10	Системне тестування
1.3.11	Інсталяція ПЗ
1.4	Експлуатація
1.4.1	Функціональне використання
1.4.2	Підтримка споживача
1.5	Супроводження
2. Категорія «Процеси підтримки»	
2.1	Документування
2.2	Керування конфігурацією
2.3	Забезпечення гарантії якості
2.4	Верифікація
2.5	Валідація
2.6	Загальний огляд
2.7	Аудит
2.8	Вирішення проблем
2.9	Забезпечення застосовності продукту
2.10	Оцінювання продукту

3. Категорія «Організаційні процеси»		
3.1	Керування	
	3.1.1	Керування на рівні організації
	3.1.2	Керування проектом
	3.1.3	Керування якістю
	3.1.4	Керування ризиком
	3.1.5	Організаційне забезпечення
	3.1.6	Вимірювання
	3.1.7	Керування знаннями
3.2	Удосконалення	
	3.2.1	Упровадження процесів
	3.2.2	Оцінювання процесів
	3.2.3	Удосконалення процесів

Як видно з табл. 1, усі процеси в даному стандарті поділяються на три категорії:

– основні процеси;

– процеси підтримки;

– організаційні процеси.

Для кожного з процесів визначені види діяльності (дії – activity), задачі, сукупність результатів (виходів) діяльності і розв'язання задач, а також деякі специфічні вимоги. У стандарті наведено перелік робіт для основних, організаційних процесів і процесів підтримки, але не спосіб їх виконання і не форма подання результатів.

Для даних процесів проведено виділення базових понять, концептів і зв'язків. На основі цього опису система DSL Tools VS.Net дає наступне графічне подання основних процесів ЖЦ у вигляді класів процесів та відношень між ними. Зпочатку розробляється візуальну модель домену ЖЦ. В ній DSL Tools надає опису класів та відношень між ними для подання основної логіки домену. В кожному класі описуються наявні методи та поля необхідні для функціонування домену ЖЦ. Нижче наведено приклад основних процесів ЖЦ стандарту ISO/IEC 12207, отриманих у середовищ DSL Tools (рис.1).

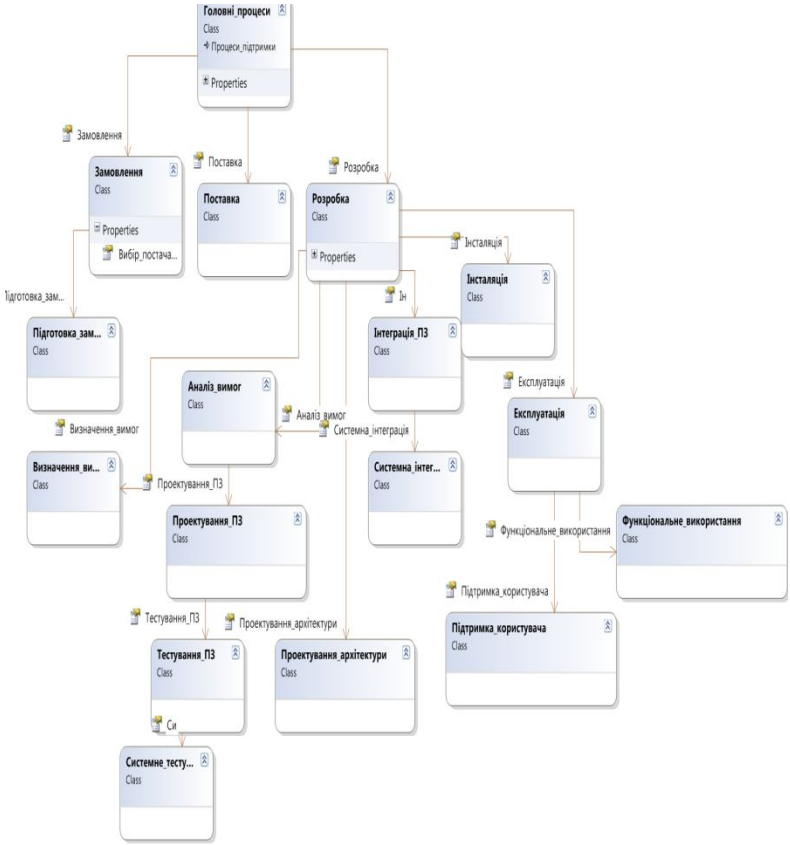


Рис. 1. Графічне подання головних процесів ЖЦ

Графічний вигляд процесів ЖЦ згенерованийц до мови XML. Фрагмент опису основних процесів подано і в XML.

```
<?xml version="1.0" encoding="utf-8"?>
 <AssociationLine Name="Визначення_вимог"
Type="Main.Визначення_вимог" ManuallyRouted="true"
FixedFromPoint="true" FixedToPoint="true">
 </AssociationLine>
 <AssociationLine Name="Інтеграція_ПС" Type="Main.Інтеграція_ПС"
ManuallyRouted="true"
 <AssociationLine Name="Інсталяція" Type="Main.Інсталяція"
ManuallyRouted="true"
 <AssociationLine Name="Експлуатація" Type="Main.Експлуатація"
ManuallyRouted="true"
 <Property Name="Визначення_вимог" />
```

<Property Name="Проектування_архітектури" />
<Property Name="Інтеграція_ПС" />
<Property Name="Інсталяція" />
<Property Name="Аналіз_вимог" />
<Property Name="Експлуатація" />….
</Class>
<Property Name="Проектування_ПС" />
</ShowAsAssociation>

Виходячи з цього XML опису процесів ЖЦ наведеним процесам і операціям додати опис семантичної програми в одної з сучасних мов програмування, що реалізує функції цих основних процесів ЖЦ. Для прикладу узятий процес тестування ЖЦ. Він анотований засобами представлення знань засобами Protégé. Це зовсім новий вид опису відсутний в практиці програмування і орієнтований на застосування на лінії тестування сучасних програм [1-3].

Опис онтології процесу тестування ЖЦ

Для представлення процесу тестування використовується онтологічна система Protege. В ній знання подаються *класами, слотами, фасетами* та *аксіомами*.

Подібну можливість надають також і інші інструменти, наприклад, діаграми класів в UML системи Rational Rose, які можуть відображатися в програмний код на декількох мовах програмування.

Поняття тестування в домені ЖЦ поделяються на дві групи: *прості* і *складні*.

Прості поняття тестування. Це такі у процесі тестування: Тестер *(Tester),* Контекст *(Context),* Дія *(Activity),* Метод *(Method),* Артефакт *(Artefact) і Середовище (Environment).*

Просте поняття можуть мати атрибути. В якості *атрибутів* вибирані такі під-поняття, які характеризують базове (батьківське) поняття і можуть приймати конкретні значення. Дамо короткий зміст цих понять:

Tester – визначає суб'єкт або об'єкт, який виконує тестування. Група тестування має лідера, який є атрибутом поняття, а його ім'я – значенням атрибута. Тобто є атрибути: *ім'я, тип, обов'язки*. Атрибут тестера – обов'язки – описує, що може робити тестер в процесі тестування. Поняття обов'язки – складне поняття, яке повинне визначатися на основі простих понять.

Таким чином, для поняття тестер можна виділити наступні атрибути: ім'я (TESTER_NAME), тип (TESTER_TYPE), обов'язки.

Атрибут тестера – *обов'язки* – описує, що може робити тестер в процесі тестування. Поняття обов'язки – складне поняття, яке повинне визначатися на основі простих понять.

Приклад. Фрагменти XML-документа, які задовольняють схемі.

Опис тестера з прізвищем *Іванов*

```
<TESTER TESTER_TYPE="HUMAN" TESTER_NAME="Іванов" />
Опис групи тестування с керівником Петров и програмним компонентом.
<TESTER TESTER_TYPE="GROUP" TESTER_NAME="ATEAM" TESTER_LEADER="Петров">
<TESTER TESTER_TYPE="HUMAN" TESTER_NAME="Петров" />
<TESTER TESTER_TYPE="SOFTWARE" TESTER_NAME="ANAGENT" />
</TESTER>
```

Контекст - в процесі розроблення визначає відповідні рівні, методи тестування, входи і виходи задач тестування. В онтології поняття контекст у тестуванні визначає один атрибут: CONTEXT_TYPE (Рівень_тестування) з призначеними значеннями:

Рівень_тестування = {модульне, інтеграційне, системне, регресійне}

Дія складається з понять, що деталізують кроки процесу тестування: планування тестування, розроблення (генерація) тестів, виконання тестів, оцінка результатів, вимірювання тестового покриття, генерація звітів тощо. Для цього поняття визначається один атрибут – Тип дії (ACTIVITY_TYPE) – з можливими значеннями:

Тип_дії = {планування, розробка тестів, виконання тестів, перевірка результатів, оцінка покриття, підготовка звіту}

Метод – це поняття, якому відповідає декілька методів тестування. Наприклад, для модульного тестування застосовні методи структурного тестування та функціонального тестування. В той же час, існують різні підходи до класифікації методів тестування. Наприклад, кожний метод по відношенню до початкового коду може класифікуватися на «білу скриньку» або базоване на коді тестування (program-based) і «чорну скриньку» або базоване на специфікації тестування (specification-based).

Фрагмент XML-схеми поняття:

```
<!-- METHOD -->
<xs:element name="METHOD">
 <xs:complexType>
 <xs:attribute name="METHOD_NAME" use="required">
 <xs:simpleType>
 <xs:restriction base="xs:token">
 <xs:enumeration value="CONTROL_FLOW_TESTING"/>
 <xs:enumeration value="DATA_FLOW_TESTING"/>
```

```
    <xs:enumeration
value="STATEMENT_COVERAGE_TESTING"/>
    <xs:enumeration value="BRANCH_COVERAGE_TESTING"/>
```

Методи, базовані на коді, підрозділяються на: структурні; підсіву помилок; - мутаційні. Структурні методи підрозділяються на дві групи: тестування потоку керування і тестування потоку даних. Методи тестування потоку керування (control-flow methods) включають покриття операторів, покриття гілок і різні критерії покриття шляхів. Ці конкретні методи тестування є екземплярами різних підкласів методів тестування.

Аналогічним чином можна класифікувати методи «чорної скриньки» або засновані на специфікації: функціональні; базовані на припущенні про помилки; евристичні і ін.

З іншого боку, по відношенню до процесу пошуку помилок і відмов, всі методи можна розділити на систематичні (пошуку помилок) і стохастичні (статистичні) – виявлення відмов.

Таким чином, для представлення методу тестування введемо наступні атрибути:

- ім’я (назва методу), наприклад, «розбиття на категорії»;
- тип методу, наприклад, структурне, базоване на помилках;
- підхід, базований на коді, на специфікаціях, статистичний.

Таке розбиття методів дозволяє однозначно класифікувати кожний метод тестування і розширювати онтологію.

Артефакт. Кожна дія з тестування може включати декілька артефактів, таких як об’єкт тестування, проміжні дані, результати тестування, плани, набори тестів, скрипти тощо. Ці артефакти називають «тестовими активами». Об’єкти тестування можуть бути різних типів: початковий код, HTML файли, XML файли, вбудовані зображення, звук, відео, документи і ін. Всі ці артефакти відображаються у онтології. Кожний артефакт також асоціюється з місцем свого зберігання, даними, історією створення і перегляду, наприклад, творця, час оновлення, номер версії тощо.

Enviroment. Програмне середовища де виконується тестування як правило натується такими даними: ім’я продукту, тип продукту і версію. Поняття середовище розбивається на два під-поняття: апаратне і програмне, а кожне з них містить атрибути.

Атрибути під-поняття апаратне середовище: *назва пристрою, модель, виробник*.

Атрибути під-поняття програмне середовище: *назва продукту, тип продукту і версія*. Можливими значеннями атрибута тип продукту можуть бути, наприклад:

Середовище = {ОС, БД, Компілятор, Веб-броузер}

Складні поняття процесу тестування. До складних понять відносимо такі: *обов'язки тестера (capability) і задача (task)*. Ці поняття визначаються за допомогою основних простих понять.

В розподіленій системі взаємодія між компонентами виконується за допомогою інтерфейсів (повідомлень). Після оброблення повідомлення, компонент, який його отримав, повертає *відповідь*. Тому в онтологію доцільно ввести додаткові поняття повідомлення і відповідь. З кожним повідомленням можна зв'язати атрибути *Тип* і *Значення* (Вміст). Можливі значення типу повідомлень задаватимемо переліком.

Тип = {директивне, декларативне, ..}

З кожною відповіддю можна зв'язати її стан. Стан будемо задавати у вигляді атрибута з двома можливими значеннями: *Стан_Відповіді = {Успіх, Відмова}*.

Приклад опису варіанту програми тестування (рис.2) на мові, яке наведено в інструментально-технологічному комплексі –ІТК сайта http://sestudy.edu-ua.net [3].

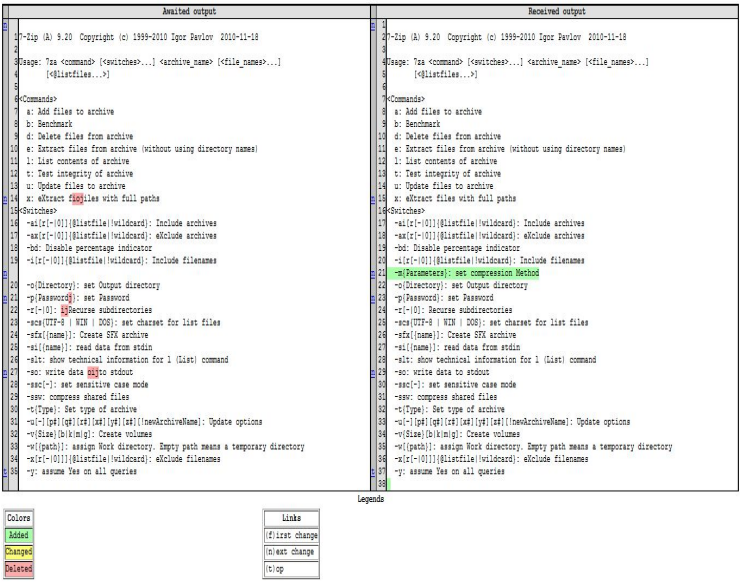


Рис 2. Програма тестування з помилками програм

Висновок. Досвіт навчання і використання онтологічного підходу широко поширений зараз у світі для подання різних предметних областей відображено на прикладі домену ЖЦ і процесу тестування програм у ІТК ІПС НАНУ у вигляді лінії.

Список використаних джерел

1. Лавріщева К.М., Коваль Г.М., Бабенко Л.П., Слабоспицька О.О., Ігнатенко П.П. Нові теоретичні засади технології виробництва сімейств програмних систем у контексті генерувального програмування – Сайт ДНБ ім. Ак. Вернадського, 2012. – Available at http://www.nbuv.gov.ua/

2. Лавріщева К.М. Програмна інженерія. – Підручник. – К.: Академперіодика, 2008. –319 с.

3. Лавріщева К.М. Інструментально-технологічний комплекс для розроблення й навчання прийомам виробництва програмних систем, Вісник НАНУ, 2012, №3. – С. 67-79. – Available at http://www.nbuv.gov.ua/portal/all/herald/2012_3/a4-3.pdf

4. Lavrischeva E., Ostrovski A., Radetskyi I. Approach to E–Learning Fundamental Aspects of Software Engineering //8– th international Conf. ICTERI– 2012 "ICT in Education, Research and Industrial Applications".– Publ. springer –sbm.com/ocs/home/ ICTERI–2012 Proc. 8-th Int. Conf. ICTERI 2012, Kherson, Ukraine, June 6-10, 2012. – Available at http://ceur-ws.org/Vol-848/ICTERI-2012 -CEUR- WS-paper-17-p-176-187.pdf .

5. Андон П.І., Лавріщева К.М., Слабоспицька К.М. Методологія побудови ліній виробництва програмних продуктів і їх практичне застосування // Міжнародний науковий конгресс «Інформаційне суспільство в Україні» (24-25 жовтня, 2012). Держ агентство з питань науки, іновацій та інформатизації України – Available at http://www.ict-congress.com.ua/attachments/article/102

АГРЕГИРОВАННАЯ МОДЕЛЬ МАКРОЭКОНОМИЧЕСКОЙ СИСТЕМЫ

Р.Р. Ларина, И.Ю. Гришин

Республиканское высшее учебное заведение «Крымский гуманитарный университет», Ялта, Украина

larinarr@inbox.ru, igugri@gmail.com

Предлагаемая модель предназначена для изучения основных тенденций развития макроэкономической системы, таких как закономерности процесса расширенного воспроизводства, роста национального дохода, соотношения фондов потребления и накопления и др. Она позволяет оценить долгосрочные последствия принимаемых решений в области распределения бюджетных средств.

Структурная схема агрегированной модели экономической системы приведена на рис.1.

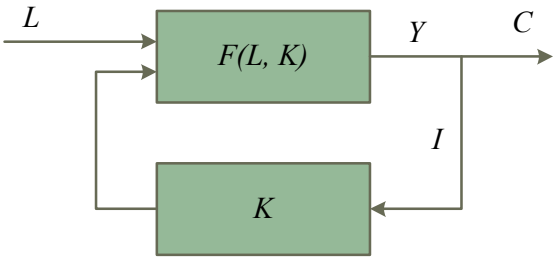


Рис.1. Структурная схема агрегированной модели экономической системы

На вход модели поступают капитальные вложения в виде основных фондов (*K*) и людские ресурсы (рабочая сила) (*L*). В результате процесса производства, который отражается преобразованием *F(K, L)* производится конечный продукт *Y*. Этот продукт идет на конечное потребление *(C)* и на инвестиции *(I)*.

Как видно из схемы, величина *L* является независимым фактором производства, а величина *K* включена в замкнутый контур системы.

Динамику этой модели можно описать следующей системой дифференциальных уравнений (1), которые вытекают из схемы и введенных обозначений.

$$
\begin{aligned}
&Y = F\left(L, K\right),\\
&y = C + I,\\
&\dot{K} = I - \mu K,\\
&I = SY,\\
&C = \left(1 - S\right)Y,\\
&\dot{L} = \eta L.
\end{aligned}
\qquad (1)
$$

Здесь величина μ - норма износа основных фондов, *S* - доля конечной продукции, идущая на воспроизводство основных фондов; а η -темп роста населения (трудовых ресурсов).

Здесь все величины зависят от времени, но аргумент *t* опущен для простоты записи.

Функция *F(L,K)* – производственная функция.

После подстановок и упрощения система (1) принимает вид

$$
\begin{aligned}
&\dot{K} = SF(L, K) - \mu K,\\
&C = (1 - S)F(L, K),\\
&L = L_0 e^{\eta t}.
\end{aligned}
\qquad (2)
$$

Введем новые переменные:

$k = \dfrac{K}{L}$ – фондовооруженность в рассматриваемой экономической системе;

$l = \dfrac{C}{L}$ – потребление на душу населения;

$\varphi\left(k\right) = \dfrac{F\left(L, K\right)}{L}$ – производительность труда.

После подстановки этих переменных в (2), преобразования и упрощения получим окончательную систему уравнений, описывающих динамику экономической системы, учитывающую начальные условия функционирования

$$
\begin{aligned}
&\dot{k} = S\varphi\left(k\right) - \left(\mu + \eta\right)k,\\
&l = \left(1 - S\right)\varphi\left(k\right),\\
&k\left(0\right) = k_0.
\end{aligned}
\qquad (3)
$$

Полученная модель (3) может бать использована для проведения макроэкономического анализа. К примеру,

проанализируем траекторию движения системы при постоянной величине *S,* каким образом будут меняться величины *Y, K, C*. Очевидно, что *S* должна удовлетворять ограничению $0 \le S \le 1$.

В результате анализа (3) следует, что динамику системы определяет только первое уравнение. Отобразим это на графике (рис. 2).

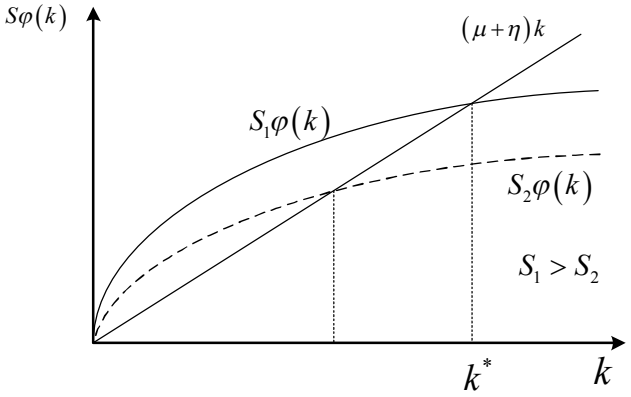


Рис. 2. Анализ динамики макроэкономической системы

Из анализа рис. 2 следует, что при постоянном значении *S* система имеет состояние устойчивого равновесия, определяемого условием

$$S\varphi(k) = (\mu+\eta)k\,. \qquad (4)$$

С помощью предложенной модели можно осуществить анализ влияния изменения величины доли конечной продукции, идущей на воспроизводство основных фондов, на поведение макроэкономической системы в целом, а также определить оптимальное значение этой величины. Могут быть решены и многие другие задачи макроэкономического анализа.

ПІДГОТОВКА СТУДЕНТІВ ДО ВИКОРИСТАННЯ ХМАРНИХ ТЕХНОЛОГІЙ

І.В. Лупан[1], П.І. Андронатій[2]

Кіровоградський державний педагогічний університет
імені Володимира Винниченка

[1]ilupan@kspu.kr.ua, [2]pashaandronatiy@gmail.com

Одним із провідних напрямків розвитку сучасної освіти є перехід на нову концептуальну модель, яку називають "відкритою освітою". Вона полягає у "необхідності забезпечення рівного доступу до якісної освіти для всіх тих, хто повинен навчатися …, хто має бажання, потребу … навчатися впродовж життя і хто має для цього можливості" [5]. До інструментів систем відкритої освіти відносять науково-освітні інформаційні мережі, спеціальні технології підтримки віртуальної навчальної діяльності (якими є, наприклад, сервіси web 2.0), глобальні мережі вчителів-новаторів, спеціальні технології підвищення ефективності проектування та використання комп'ютерно-орієнтованих систем навчального призначення, технології мережного е-дистанційного навчання, електронні предметно-інформаційні ресурси, сучасні мобільні засоби, тощо.

Більшість з перелічених інструментів можуть бути реалізовані в рамках моделі хмарних обчислень, яка полягає у наданні користувачам динамічного доступу до послуг, обчислювальних ресурсів та додатків через Інтернет [11]. Web 2.0 застосовується у хмарних обчисленнях як модель надання сервісів. У свою чергу Web 2.0 визначають як концепцію, комплексний підхід до організації, реалізації та підтримки web-ресурсів.

Інтерес до хмарних технологій як таких та до їх використання в освітній діяльності в Україні зростає, свідченням чого, зокрема є Всеукраїнський науково-методичний семінар "Хмарні технології в освіті" [13]. Судячи з публікацій, методисти досить ретельно вивчають дидактичні аспекти застосування найрізноманітніших сервісів [наприклад, 8]. Однак найчастіше поки використовуються wiki-технології.

Можливості й способи використання Вікі-технології у педагогічній практиці висвітлено в публікаціях Е.Д. Патаракіна, Е.Ю. Кулик [9], Л.Н. Рулієне, К. Браунгардт [12] та багатьох, багатьох інших. Wiki-технологія – це технологія побудови web-систем, призначених для колективної розробки, зберігання, структуризації тексту, гіпертексту, файлів, мультимедіа, тощо. Одним з прикладів використання wiki в освітньому процесі вищого навчального закладу є ресурс MachineLearning.ru [3], зокрема,

сторінки В.К. Воронцова зі статистичного аналізу даних.

Використання технології wiki дозволяє викладачеві залучити до створення wiki-конспекту дисципліни усіх студентів. При цьому досить легко вести облік виконаної кожним самостійної роботи, удосконалювати зміст конспекту, формувати та редагувати посилання на корисні ресурси, тощо. Робота над конспектом (редагування, поширення та поглиблення тематики) може продовжуватися наступними курсами студентів.

За таким принципом будується більшість wiki-проектів, розміщених на wiki-порталі Кіровоградського державного педагогічного університету імені Володимира Винниченка Вікі-КДПУ [2]. Ресурси порталу постійно поповнюються, тож його навіть як приклад wiki-спільноти наведено у діючому підручнику інформатики для школи [7, с. 279]. Портал працює на движку MediaWiki [1], який поширюється за ліцензією GNU/GPL. Одним з таких є проект "Історія інформатики", започаткований для майбутніх вчителів інформатики [14].

Робота над wiki-проектом потребує від студентів деяких початкових знань та навичок. Це робота з електронною поштою та пошуковими системи Інтернету, вивчення сервісів Веб 2.0 для збереження фотографій, схем, малюнків, відео- та аудіо-записів і таке інше, опанування тегів розмітки wiki-сторінок та засобів введення та редагування математичних формул. Студенти-майбутні вчителі інформатики набувають їх під час вивчення відповідних курсів протягом усього навчання. Студенти ж спеціальності "Статистика" набувають необхідних компетенцій, виконуючи навчальний проект під час вивченні курсу "Інформатика та програмування". За розробленою авторами статі програмою, завдання проекту для студентів полягає у проведенні ознайомлювального дослідження з деякої довільно обраної студентом тематики з обов'язковим збором статистичного матеріалу та оформленням результатів дослідження засобами web 2.0-інструментарію [6]. Сюди входить пошук необхідних матеріалів у Всесвітній мережі, оформлення посилань на них на сторінці власного проекту, створення такої сторінки на wiki-порталі КДПУ, наповнення її таблицями, рисунками, посиланнями на Інтернет-джерела, відеоресурси та презентації, тощо. Крім того, що таке завдання завжди викликає інтерес студентів, набуті навички вони активно використовують у подальшому при створенні wiki-конспектів з інших дисциплін.

Список використаних джерел:

1. General Overview: http://mediawiki.org.
2. http://wiki.kspu.kr.ua

3. MachineLearning.ru: Профессиональный информационно-аналитический ресурс, посвященный машинному обучению, распознаванию образов и интеллектуальному анализу данных. Режим доступу: http://www.machinelearning.ru/wiki/
4. Биков В.Ю. Технології хмарних обчислень – провідні інформаційні технології подальшого розвитку інформатизації системи освіти України/ В.Ю.Биков // Комп'ютер у школі та сім'ї. – 2011. − №6. – С.3-11.
5. Биков В.Ю., Мушка І.В. Електронна педагогіка та сучасні інструменти систем відкритої освіти. //Інформаційні технології і засоби навчання. – 2009. − №5 (13). – Режим доступу: http://www.ime.edu-ua.net/em13/emg.html
6. Інформатика та програмування. − http://wiki.kspu.kr.ua/ index.php/Інформатика_та_програмування_(27_група
7. Інформатика-11 (академічний рівень, профільний рівень) / Лисенко Т.І., Ривкінд Й.Я., Чернікова Л.А., Шакотько В.В. – К.: Вид-во «Генеза», 2011. – 300 с.
8. Крибель С.С., Шобухова В.В. Использование социальных сетей в образовании. // Информатика и образование. – 2012. − №4. − С.66-68.
9. Кулик Е.Ю., Патаракин Е.Д. (28 января 2006). WikiWiki в организации учебного процесса. Режим доступу: http://heap.altlinux.org/pereslavl2006/kulik/abstract.html
10. Патаракін Є.Д. Створення учнівських, студентських і викладацьких спільнот на базі мережевих сервісів Веб 2.0. − К.: Навчально-методичний центр "Консорціум із удосконалення менеджмент-освіти в Україні", 2007. − 88 с.
11. Романченко В. Облачные вычисления на каждый день. http://www.3dnews.ru/editorial/cloud_computing/ Дата: 06/09/2009
12. Рулиене Л.Н., Браунгардт К. Роль Wiki в развитии современного образовательного процесса. Режим доступу: http://ruliene.bsu.ru/wp-content/uploads/Ruliene_UlanUde.pdf
13. Хмарні технології в освіті: матеріали Всеукраїнського наук.-метод. Інтернет-семінару (Кривий Ріг – Київ – Черкаси – Харків, 21 грудня 2012 р.). – Кривий ріг: Вид. відділ КМІ, 2012. – 173 с.
14. Історія інформатики (проект для студентів спеціальності "Математика та інформатика"). Режим доступу: http://wiki.kspu.kr.ua/ index.php/Історія_інформатики

ПРОБЛЕМА СПРОСТОВНОСТІ У МОНОТОННИХ ЛОГІКАХ ФЛОЙДА-ХОАРА

М.С. Нікітченко, А.В. Криволап

Київський національний університет імені Тараса Шевченка, Київ, Україна

nikitchenko@unicyb.kiev.ua, krivolapa@gmail.com

Монотонна логіка Флойда-Хоара для часткових квазіарних предикатів

У статті [1] показано, що класична логіка Флойда-Хоара не є монотонною, якщо її розширити на випадок часткових квазіарних предикатів. Тому було дано визначення композиції Флойда-Хоара, яка була б монотонною за всіма аргументами. Монотонність розуміється відносно визначеності предикатів та функцій. Інтуїтивне визначення наступне: композиція є монотонною, якщо при довизначенні предикатів та функцій, що є її аргументами, результат композиції не зміниться на тих даних, на яких він вже був визначений.

Всі поняття, що не визначені в даній роботі, будемо розглядати в смислі [1–3]. Зокрема, V – множина імен (змінних), A – множина базових значень, ${}^{V}A$ – множина номінативних даних, $Pr^{V,A}$ – множина квазіарних предикатів над ${}^{V}A$, $Prg^{V,A}$ – множина бікваазіарних функцій над ${}^{V}A$, які є одним з варіантів подання семантики програм. Тоді композиція Флойда-Хоара буде мати тип $FH : Pr^{V,A} \times Prg^{V,A} \times Pr^{V,A} \to Pr^{V,A}$, тобто за передумовою, програмою та післяумовою вона видає відповідний предикат.

Композиція визначена наступним чином:

$$FH(p, pr, q)(d) = \begin{cases} T, \text{ якщо } p(d)\downarrow = F \vee q(pr(d))\downarrow = T, \\ F, \text{ якщо } p(d)\downarrow = T \wedge q(pr(d))\downarrow = F, \\ \text{невизначено в інших випадках.} \end{cases}$$

В статті [1] було доведено, що така композиція буде не лише монотонною, але й неперервною. Під істинністю будемо як і в статті [1] розуміти неспростовність.

Монотонна композиція найслабкішої передумови для випадку часткових квазіарних предикатів

Для того, щоб мати змогу перевіряти чи існує спростування для конкретної трійки Флойда-Хоара, потрібно визначити також композицію найслабкішої передумови, яка б за післяумовою та програмою видавала б деяку передумову таку, щоб відповідна трійка була б істинною. Відповідне перетворення предикатів було введено Дейкстрою для випадку тотальних предикатів в [4]. Якщо розглядати часткові предикати, то очевидно, що визначень композиції найслабкішої передумови може бути декілька. Визначимо відповідну композицію виходячи з того, що вона має бути монотонною, отримана передумова має дозволяти показати що трійка Флойда-Хоара спростовна та передумова може бути просто обчислена за післяумовою та програмою. Позначимо p^T області істинності, а p^F області хибності предиката p. Тоді композицію найслабкішої передумови

$$(WP : Prg^{V,A} \times Pr^{V,A} \to Pr^{V,A})$$

будемо визначати наступним чином:

$$WP(pr,q)(d) = \begin{cases} T, \text{ якщо } q(pr(d))\downarrow = T, \\ F, \text{ якщо } q(pr(d))\downarrow = F, \\ \text{невизначено в інших випадках.} \end{cases}$$

Іншими словами області істинності та хибності найслабкішої передумови є повними прообразами областей істинності та хибності післяумови відносно програми. Формально це можна записати :

$$WP(pr,q) = p \Leftrightarrow p^T = pr^{-1}[q^T] \wedge p^F = pr^{-1}[q^F].$$

Лема 1. Композиція $WP : Prg^{V,A} \times Pr^{V,A} \to Pr^{V,A}$ – монотонна за всіма аргументами, іншими словами, виконується наступне:

$$pr \subseteq pr' \wedge q \subseteq q' \Rightarrow WP(pr,q) \subseteq WP(pr',q').$$

Покажемо, як за побудовою біквазіарної функції (тобто програми) ми можемо визначити найслабкішу передумову. Для цього розглянемо всі композиції програмного рівня [3] , окрім композиції циклу, а саме: композицію присвоєння, композицію послідовного виконання, композицію умовного оператору, та композицію тотожної програми. Тоді можна довести таке твердження.

Теорема 1. Мають місце наступні рівності:

- $WP(id,q) = q$,

- $WP(AS^x(fa),q) = S^x(q,fa)$,
- $WP(f \bullet g,q) = WP(f,WP(g,q))$,
- $WP(IF(b,f,g),q) = (b \to WP(f,q)) \wedge (\neg b \to WP(g,q))$.

Можна побачити, що в такому випадку зберігається властивість $WP(pr,q)(d)\downarrow = F \Leftrightarrow q(pr(d))\downarrow = F$. Отже проблему пошуку спростування трійки Флойда-Хоара $\{p\}pr\{q\}$ можна звести до проблеми пошуку спростування $p \to WP(pr,q)$, що виражає наступна лема.

Лема 2. $(p \to WP(pr,q))(d)\downarrow = F \Leftrightarrow FH(p,pr,q)(d)\downarrow = F$.

Таким чином, ми отримуємо можливість працювати зі звичайною імплікацією, а не з композицією Флойда-Хоара. Якщо розглядати обмежений випадок, в якому програма задається без використання композиції циклу, або програма апроксимується простішими програмами без циклів, то найслабкіша передумова рекурсивно обчислюється за допомогою наведених правил. Це дозволяє звести формули трьохосновної програмної логіки до формул двохосновної композиційно-номінативної логіки предикатів [5], яка є простішою та більш дослідженою, ніж програмна логіка.

Список використаних джерел

1. Нікітченко М.С., Криволап А.В. Семантичні властивості монотонних логік Флойда-Хоара / М.С. Нікітченко, А.В. Криволап // Вісник Київського національного університету імені Тараса Шевченка. Сер.: фіз.-мат. науки. – 2012. – Вип. 3. – С. 215-222.
2. Никитченко Н.С. Композиционно-номинативный подход к уточнению понятия программы / Н.С. Никитченко // Пробл. программирования. – 1999. – № 1. – С. 16–31.
3. Нікітченко М.С. Теорія програмування. Частина **1**: [навчальний посібник] / М.С. Нікітченко. – Ніжин: Видавництво НДУ імені Миколи Гоголя, 2010. – 129 с.
4. Edsger W. Dijkstra Guarded commands, nondeterminacy and formal derivation of program / Edsger W. Dijkstra // Communications of the ACM. – 1975. – №18(8). – С. 453–457.
5. Nikitchenko M.S., Tymofieiev V.G.: Satisfiability in Composition-Nominative Logics // Central European Journal of Computer Science.– 2012.– Vol. 2, issue 3.– P. 194–213.

ПРО ДЕЯКІ ПІДХОДИ ДО МОДЕЛЮВАННЯ ПОВЕДІНКИ ВІДВІДУВАЧІВ ТЕМАТИЧНОГО ПОРТАЛУ НА ОСНОВІ МАРКОВСЬКИХ ПРОЦЕСІВ ПРИЙНЯТТЯ РІШЕНЬ

О.В. Олецький

Національний Університет «Києво-Могилянська Академія»

oletsky@ukr.net

Поведінку відвідувачів веб-сайту, зокрема тематичного порталу, в загальних рисах можна охарактеризувати як послідовність переходів між окремими сторінками сайту, спрямовану на досягнення тієї чи іншої мети. Природним чином постає проблема: на основі дослідження факторів, які впливають на формування цих послідовностей, здійснити реорганізацію гіпертекстових посилань так, щоб структура порталу стала більш дружньою для відвідувача, а процес навігації – більш цілеспрямованим.

Процес навігації на тематичному порталі очевидним чином пов'язаний з процесом прийняття рішень, оскільки кожний перехід – це по суті і є рішення про те, відвідання якої сторінки видається йому найбільш доцільним. Важливу роль у дослідженні цих процесів може відіграти математичне та імітаційне моделювання, в рамках якого слід розглядати:

- моделі, які описують структуру та інформаційне наповнення тематичного порталу;
- моделі, які характеризують процес досягнення мети відвідувача;
- моделі, які описують власне прийняття рішень.

Як базову модель, що описує структуру гіпертекстових посилань, природно розглядати структуру, яку можна назвати навігаційним графом. Вузли такого графа відповідають окремим сторінкам, дуги – зв'язкам між ними. Точніше, від вузла A до вузла B йде орієнтована дуга, якщо A містить гіпертекстове посилання на B.

Але вузли навігаційного графа – це не просто деякі абстрактні вершини. Вони відповідають документам, які, в свою чергу, пов'язані з поняттями предметної області. Тому переходи між вузлами – це фактично переміщення в деякому просторі понять, і врахування цього дозволяє зробити аналіз навігації по сайту більш предметним і семантично-орієнтованим.

У роботах [1, 2] розвивається підхід, спрямований на опис зв'язків між онтологією предметної області та документами, які складають основу інформаційного наповнення порталу з урахуванням специфіки веб-орієнтованих систем. В рамках цього

підходу розглядається наступна формалізація зв'язків "онтологія - документ".

Нехай W – множина вузлів онтології предметної області. Будемо вважати, що $W=T\cup V$, де T – множина термінальних концептів (змістовно – множина термінів, які можуть зустрічатися в документах), V – множина метасимволів, тобто нетермінальних концептів, понять, які не можуть безпосередньо з'являтися в документах, але які впливають на їх формування та ранжування за релевантністю. Можна вважати, що $T\cap V=\varnothing$, оскільки у випадку, якщо слово з тексту водночас є важливим поняттям онтології, ми можемо просто формально перейменувати відповідний нетермінальний концепт.

Аналогічно, множину документів D подамо у вигляді $D=DT\cup DV$, де DT – множина документів як таких, DV – множина категорій документів.

Введемо наступні множини можливих зв'язків:

R – множина типів зв'язків між концептами предметної області (в першу чергу – нетермінальними);

L – множина типів зв'язків між концептами предметної області та документами;

H – множина типів зв'язків між документами.

Тоді модель інформаційного наповнення онтологічно-орієнтованого тематичного порталу можна записати у вигляді відношення

$$M = W^*\cup Z^*\cup D^*, \qquad (1)$$

де W^* - множина кортежів (w_1, r, w_2); $w_1,w_2\in W$, $r\in R$;

Z^* - множина кортежів (w,l,d); $w\in W$, $l\in L$; $d\in D$;

D^* - множина кортежів (d_1, h, w_2); $d_1,d_2\in D$, $h\in H$.

Співвідношення (1) можна прийняти як деяке робоче наближення до формалізованої моделі. Але воно ще залишається досить інтуїтивним: зокрема, більш точно було б говорити не про те, що зв'язок належить до множини типів зв'язків, а про те, що цей зв'язок є екземпляром деякого класу з множини можливих класів зв'язків. Але на цьому етапі ця відмінність не грає вирішальної ролі.

При моделюванні процесу досягнення цілей відвідувачами веб-порталу ми розглядаємо два підходи:

- драстичний (все або нічого), при якому відвідувача цікавить лише конкретна інформація та успіх досягається лише на сторінці, яка містить цю інформацію, а всі інші розглядаються лише як проміжні ланки на шляху до неї;

- кумулятивний, при якому кожна сторінка може мати самостійну інформаційну цінність, і сумарний інформаційний виграш від процесу навігації певним чином накопичується.

При цьому слід враховувати певні ресурсні обмеження: обмежений загальний час, обмежена кількість переходів, обмежений час перебування на кожній сторінці тощо. При цьому такі обмежень варто описувати в рамках теорії нечітких множин.

Очевидно, ключову роль має відігравати дослідження процесів, пов'язаних з прийняттям рішень щодо навігації по порталу. Відповідні моделі можна будувати в рамках теорії навчання з підкріпленням на основі марковських процесів прийняття рішень [3]. які в загальних рисах можна охарактеризувати наступними компонентами:

- множина станів S;
- множина дій A;
- функція заохочення R, яка описує винагороду, отриману за дію a в стані s;
- функція переходу між станами T (взагалі кажучи, вона може мати імовірнісний характер).

При моделюванні процесу навігації стани в найпростішому випадку відповідають окремим сторінкам. Але з огляду на описану вище модель у вигляді співвідношення (1) кожний стан характеризується деяким набором пов'язаних з ним термінів онтології. Тому можна говорити про певні групи станів, які мають спільні характеристики.

Дії відповідають переходам за гіпертекстовими посиланнями. Функція заохочення має відповідати додатковому виграшу, який відвідувач отримав внаслідок переходу.

У найпростішому випадку переходи від стану до стану при кожному конкретному виборі дії можна вважати детермінованими. Але якщо говорити про групи станів, функція переходів може набути імовірнісного характеру.

Дослідження може здійснюватися як шляхом аналізу поведінки відвідувачів реального порталу, так і на основі агентно-орієнтованого імітаційного моделювання. В обох випадках мова має йти про аналіз деяких рекурентних співвідношень, які дозволяють будувати оцінку корисності поточного стану на основі негайного виграшу від здійснення певної дії та оцінок корисності його наступників. Міра корисності має будуватися з урахуванням того, що найбільш перспективні переходи пов'язані зі сторінками, схожими на поточну, але не максимально схожими. Ці міри схожості теж природно описати в термінах нечітких множин; деякі підходи до побудови подібних мір розглядалися в [1, 2].

Серед можливих напрямків практичного застосування підходу, що описується, можна відмітити наступні.

1. Оптимізація структури навігаційного графа. Можна сказати, що структура навігаційного графа повинна визначатися закономірностями. які випливають з онтологічного аналізу предметної області. Можна сказати, що структура навігаційного графа тематичного порталу повинна визначатися закономірностями. які випливають з онтологічного аналізу предметної області. Якщо піти далі, можна ставити задачу автоматизованої побудови навігаційного графу на основі аналізу онтології предметної області. Теоретично можна уявляти собі повнозв'язний навігаційний граф, кожна сторінка якого зв'язана з усіма іншими, але на практиці таку структуру реалізувати неможливо і недоцільно. Користувач бажає бачити не всі можливі переходи, а лише найбільш важливі для нього. Тому постає задача динамічного формування навігаційних графів. В рамках цього підходу слід розрізняти потенційний навігаційний граф, який безпосередньо формується на основі онтології та фактичний навігаційний граф, який утворюється на основі потенційного на основі певних критеріїв оптимізації. Тут процес досягнення цілі більше тяжіє до драстичної функції – відвідувач прагне якомога скоріше перейти до потрібної йому сторінки.

2. Як зворотну задачу можна розглядати побудову онтології певної предметної області на основі аналізу відповідного навігаційного графа. Ці задачі можна розв'язувати незалежно одну від іншої або паралельно.

3. Автоматизований підбір контекстної реклами з найкращими показниками CTR (ефективність власне посилання; ймовірність переходу за посиланням) та CTB (умовна релевантність цільової сторінки; ймовірність здійснення тієї чи іншої дії).

4. Використання в навчальному процесі. Тут вузли навігаційного графа пов'язані з окремими навчальними матеріалами. Можна ввести достатньо адекватний критерій досягнення цілі: підвищення проценту правильних відповідей на тестові запитання. Відповідно до цього процес досягнення цілі описується кумулятивною моделлю. Тоді можна сформулювати проблему персоналізації навчального процесу за рахунок автоматичного підбору посилань, які б вели до найбільш корисних ресурсів.

5. Аналіз поведінки відвідувачів веб-сайту на основі методик Data Mining, зокрема Web Usage Mining [4]. Одну з найбільш типових задач Web Usage Mining можна в найбільш загальних рисах охарактеризувати наступним чином: знаючи історію навігації даного відвідувача, тобто послідовність сторінок $P_1, \ldots P_n$, що були переглянуті цим відвідувачем, виявити закономірності здійснених переходів, і на основі цього – вірогідність того, що він перейде за

деяким посиланням на сторінку *q*, а також оцінити міру його зацікавленості в цій сторінці. Серед найбільш перспективних напрямків тут можна відмітити:

- побудова дерев рішень з метою формування наборів правил "якщо *A*, то *B*";
- пошук асоціацій і алгоритм Apriori з метою виявлення закономірностей типу "*A* і *B* часто зустрічаються разом";
- кластерний аналіз з метою розбиття профілів відвідувачів та історій їх навігацій на кластери

Список використаних джерел

1. *Олецький О.В.* Організація онтологічно-орієнтованих засобів автоматизованого добору інформаційних ресурсів на тематичному порталі. //Наукові записки НаУКМА. Т.99. Комп'ютерні науки. – К., 2009. – С. 66-69.
2. *Олецький О.В.* До проблеми моделювання потоку відвідувань на онтологічно-орієнтованому тематичному порталі. //Моделювання та інформаційні технології. Збірник наукових праць. Спеціальний випуск. Т.2.- К.,2010. – С.321-326.
3. *Рассел С., Норвиг П.* Искусственный интеллект: современный подход. – М.: Изд. дом «Вильямс», 2006. – 1408 с.
4. *Барсегян А.А., Куприянов М.С., Степаненко В.В., Холод И.И.* Технологии анализа данных: Data Mining, Visual Mining, Text Mining, OLAP. – СПб: БХВ-Петербург, 2007. – 384 с.

АРХИТЕКТУРНЫЕ МОДЕЛИ НЕЧЕТКИХ ИНТЕЛЛЕКТУАЛЬНЫХ МУЛЬТИАГЕНТНЫХ СИСТЕМ

И.Н. Парасюк, С.В. Ершов

Институт кибернетики имени В.М.Глушкова НАН Украины,
Киев, Украина

iv.para@yandex.ru, sershv@ukr.net

В настоящее время большое внимание уделяется методам представления знаний с помощью нечеткой логики высшего типа. Как известно [1,2], средствами нечеткой логики можно построить методологии для вычислений со словами – основы представления знаний и моделирования рассуждений в интеллектуальных системах. Хорошо различимые границы “информационных гранул”, представляемых лингвистическими термами, могут быть описаны нечеткими функциями принадлежности. Это основа теории нечетких множеств типа 1. Однако, когда границы плохо различимы, например, выражены некоторыми зонами, их можно выражать нечеткими множествами типа 2 и выше. У таких множеств отдельные значения принадлежности задаются функциями принадлежности. Чаще всего, имеется ввиду лингвистическая неопределенность, связанная с различными оттенками смысла слов в моделях знаний, основанных на правилах [1]. Нечеткими множествами типа 2 можно характеризовать степени неопределенности, связанные с принятием решений в интеллектуальных системах, в том числе мультиагентных. Основные понятия, характеризующие нечеткую логику высшего типа, то есть функции и степени принадлежности второго типа, а также понятие следа неопределенности описаны в [3,4].

Уместно отметить, что создание агентов, использующих преимущества нечеткой логики такого типа, требует разработки архитектурных моделей нечетких агентов, моделей коллективного поведения группы агентов и методов генерации платформно-специфических моделей таких агентов для перспективных высокопроизводительных платформ. Это важно, например, при решении таких прикладных задач, как исследование эффективности процессов эвакуации людей в чрезвычайных ситуациях, поведения транспортных средств в условиях неопределенности (неполной информации) и ряда других процессов.

В докладе представлены и обоснованы архитектурные модели интеллектуальных мультиагентных систем, основанные на нечеткой логике высшего типа, что позволяет более информативно представить степень неопределенности лингвистического представления фактов и знаний (убеждений) при моделировании

структуры и группового поведения интеллектуальных агентов в нечеткой среде [5-7].

Основу функционирования интеллектуального агента, основанного на нечеткой логике типа 2, составляет система правил нечеткого вывода ЕСЛИ-ТО, но множества антецедентов или консеквентов этих правил – нечеткие множества (или нечеткие числа) типа 2. Основными компонентами архитектурной модели такого нечеткого агента являются: фузификатор, система правил, средства нечеткого вывода и процессор выходных значений. Процессор выходных значений состоит из редуктора типа, создающего на выходе нечеткое множество типа 1 и дефузификатора, генерирующего соответствующее ему четкое значение.

Нечеткое множество типа 2 задается функцией принадлежности $\widetilde{A} = \left\{\left((x,u), \mu_{\widetilde{A}}(x,u)\right) \middle| \forall x \in X, \forall u \in J_x\right\}$, где $0 \le \mu_{\widetilde{A}}(x,u) \le 1$. При этом $J_x \in [0,1]$ представляет первичную степень принадлежности элемента *x*, а $\mu_{\widetilde{A}}(x,u)$ – нечеткое множество (типа 1) вторичной принадлежности.

Компонент "Фузификатор" отображает каждое входное значение $x = (x_1,..., x_p) \in X_1 \times X_2 \times ... \times X_p$ в нечеткое множество $\widetilde{A}_x$ на *X*. Один из простейших методов фузификации – построение множества-синглетона. $\widetilde{A}_x$ представляет собой нечеткий синглетон 2-го типа, если $\mu_{\widetilde{A}x}(x) = 1/1$ для всех $x = x'$ и $\mu_{\widetilde{A}x}(x) = 1/0$ для всех остальных $x \ne x'$.

Таким образом, для агента с *p* входами $x_1 \in X_1$, ... , $x_p \in X_p$ и одним выходом $y \in Y$, если предположить, что всего имеется *M* правил, то *i*-е правило агента можно представить следующим образом:

$$R^i : \text{ЕСЛИ } x_1 = \widetilde{F}_1^i \text{ И } ... \text{ И } x_p = \widetilde{F}_p^i \text{ ТО } y = \widetilde{G}^i, \quad i = 1,...,M.$$

Основываясь на нечеткой логике типа 2, нечеткий выход комбинирует правила и дает отображение входных нечетких множеств в выходные нечетких множества (типа 2). Необходимо вычислить объединения ⊔ и пересечения П, задаваемые как нечеткие отношения 2-го типа. Если $\widetilde{F}_1^i \times ... \times \widetilde{F}_p^i = \widetilde{A}^i$, то правило можно записать в виде $R^i : \ \widetilde{F}_1^i \times ... \times \widetilde{F}_p^i \to \widetilde{G}^i =$

$= \widetilde{A}^i \to \widetilde{G}^i$, $i = 1,...,M$. Правило R^i задает функцию принадлежности $\mu_{R^i}(x,y) = \mu_{R^i}(x_1,...,x_p,y)$, где $\mu_{R^i}(x,y)$ можно представить как $\mu_{R^i}(x,y) = \mu_{\widetilde{A}^i \to \widetilde{G}^i}(x,y) =$ $= \mu_{\widetilde{F}_1^i}(x_1) \prod ... \prod \mu_{\widetilde{F}_p^i}(x_p) \prod \mu_{\widetilde{G}^i}(y)$.

При использовании при разработке агента интервальных нечетких множеств типа 2 и умножения в качестве t-нормы, результат пересечения всех множеств антецедента $F^i(x) = \prod_{k=1}^{p} \mu_{\widetilde{F}_k^i}(x_i)$ дает в результате интервал срабатывания, представляемый парой значений $F^i(x) = \left[\breve{f}^i, \hat{f}^i\right]$, где $\breve{f}^i = \breve{\mu}_{\widetilde{F}_1^i}(x_1) * ... * \breve{\mu}_{\widetilde{F}_p^i}(x_p)$ – нижняя граница принадлежности, а $\hat{f}^i = \hat{\mu}_{\widetilde{F}_1^i}(x_1) * ... * \hat{\mu}_{\widetilde{F}_p^i}(x_p)$ – ее верхняя граница.

В общем случае значение, поступающее на вход нечеткого правила R^i, задается нечетким множеством типа 2 с функцией принадлежности $\mu_{\widetilde{A}_x}(x) = \mu_{\widetilde{x}_1}(x_1) \prod ... \prod \mu_{\widetilde{x}_p}(x_p) =$

$= \prod_{k=1}^{p} \mu_{\widetilde{x}_k}(x_k)$, где $\widetilde{x}_k$ (k =1,..., p) – обозначения нечетких множеств каждого из входов после фузификации. Каждое правило R^i задает результирующее нечеткое множество 2 типа такое, что $\mu_{\widetilde{B}^i}(y) = \mu_{\widetilde{A}_x \circ R^i}(y) = \coprod_{x \in X} \left[\mu_{\widetilde{A}_x}(x) \prod \mu_{R^i}(x,y)\right]$.

Данное выражение представляет собой соотношение между нечетким множеством 2-го типа, которое активирует одно правило вывода и нечетким множество 2-го типа, полученным в результате его применения.

Компонент “Редуктор типа” создает нечеткое множество типа 1, называемое центроидом, которое в дальнейшем преобразуется дефузификатором в четкое значение. В непрерывном случае построение центроида нечеткого множества типа 2 оказывается сложной вычислительной задачей. Эта задача упрощается, если функции принадлежности второго порядка имеют интервальный характер.

Задача компонента “Дефузификатор” – на основе этих значений определить единственное значение, являющееся

результатом применения ко входным значениям $x = (x_1,...,x_p)$ системы нечетких правил: $y(x) = (y_l + y_r)/2$.

Каждый компонент обобщенной архитектурной модели интеллектуального агента на основе нечеткой логики выполняет роли поставщика и потребителя значений переменных, представленных нечеткими множествами 2 типа, обычными нечеткими множествами (1-го типа) и четкими значениями на числовой шкале. Сочетание ролей внутри одного компонента однозначно определяет тип компонента, представленный в данной архитектуре. Например, сочетание ролей "Потребитель: нечеткое значение типа 2" и "Поставщик: нечеткое значение типа 1" указывает на компонент, называемый редуктором типа. Сочетание ролей "поставщик-потребитель" демонстрирует зависимости по данным, что приводит к невозможности параллельного выполнения указанных компонентов. Однако, существует возможность конвейерной обработки, при которой в каждом слое (ярусе) задействован один из компонентов, передающий полученное после обработки значение компоненту-потребителю в следующем ярусе. Такая многоярусная схема представляется целесообразной в том случае, если нечеткие агенты постоянно получают новые входные значения, но не чаще чем среднее время реализации нечеткого вывода или понижения типа средствами нечеткой логики типа 2. Заметим, что компонент нечеткого вывода также может быть представлен в виде двухъярусной схемы, в которой на первом ярусе параллельно определяются значения, являющиеся результатом действия каждого отдельного правила, а во втором ярусе осуществляется агрегирование (объединение) полученных значений, для чего применяется специально определенная операция суммы нечетких множеств типа 2, или понижение типа множества.

Архитектура иерархической нечеткой мультиагентной системы включает подробное представление отдельного нечеткого агента с целью упрощения нечетких баз правил с явным учетом отдельных агентов и взаимодействия между ними. Дополнительная эффективность достигается за счет возможности лучшего представления одновременного воздействия уменьшенного количества входов на отдельные (упрощенные) базы правил. Однако, при этом теряется точность из-за накопления ошибок в результате повторяющегося применения фузификации, нечеткого вывода, редукции типа и дефузификации в течение нескольких циклов работы такой системы. Для повышения точности редукция типа с дефузификацией и дальнейшей фузификацией промежуточных переменных могут отсутствовать, а нечеткое значение,

представляющее собой нечеткое множество типа 2 непосредственно передается между агентами.

Обобщением такой модели мультиагентных систем является ее представление как виртуальной сетки (grid). Каждый агент представлен вершиной сетки, в то время как взаимодействие между агентами – соединениями между этими вершинами. Модель мультиагентной системы с $p \times q$ вершинами $\{N_{11}\ldots N_{p1}\},\ldots,$ $\{N_{1q}\ldots N_{pq}\}$, $p \times q$ входами вершин $\{x_{11}\ldots x_{p1}\},\ldots,\{x_{1q}\ldots x_{pq}\}$, принимающими лингвистические значения из любых допустимых входных множеств, $p \times q$ выходами вершин $\{y_{11}\ldots y_{p1}\},\ldots,\{y_{1q}\ldots y_{pq}\}$, выдающими лингвистические значения из любых допустимых множеств выходов, состоит из p горизонтальных уровней и q вертикальных ярусов (слоев) и может быть описана следующим образом:

Ярус 1 …… Ярус q
Уровень 1 $N_{11}(x_{11}, y_{11})$……$N_{1q}(x_{1q}, y_{1q})$
……
Уровень p $N_{p1}(x_{p1}, y_{p1})$……$N_{pq}(x_{pq}, y_{pq})$

Структура сетки определяет местоположение агентов, а также их входы и выходы. В этом случае, каждый вход и выход может быть как скаляром, так и вектором, представляющим набор нечетких значений типа 2. Уровни в этой структуре сетки представляют собой иерархии узлов в плане субординации в пространстве, тогда как ярусы задают временную зависимость с точки зрения последовательности выполнения.

В качестве целевой платформы для выполнения мультиагентных систем используется кластерная система СКИТ-3. Для поддержки взаимодействия интеллектуальных агентов была создана специальная библиотека, которая включает ряд шаблонов, классов и методов для организации поведения агентов – циклического, одноразового и многоразового, а также для обмена разными типами асинхронных сообщений с использованием MPI в качестве носителя. Вариант библиотеки реализован для использования возможностей программно-апаратного стека CUDA. Библиотека дает возможность организовать модель системы как виртуальную сетку (grid) агентов, в которой агентам разрешается взаимодействовать тольки в пределах ограниченной дистанции (количество переходов по сетке). Из-за распределения глобального состояния между процессорами, часть состояния отдельных агентов может сохраняться у соседних агентов. При циклическом выполнении алгоритмов нечеткого вывода элементы этого состояния могут запрашиваться до начала выполнения следующей итерации.

Специальный примитив используется для барьерной синхронизации выполнения итераций (этапов выполнения) нечетких агентов. Однако, такой подход не учитывает разнообразия типов памяти и время задержки передачи сообщений между агентами, которые используют такие типы памяти. Поэтому, для уменьшения затрат на коммуникацию нами предложен специальный метод, основанный на виртуальной сетке агентов над иерархией типов памяти агентов.

Описанные выше архитектурные модели мультиагентных систем на основе нечеткой логики высшего типа позволяют более информативно представить степень неопределенности системы нечетких правил при спецификации поведения таких агентов и систем. Разработан моделе-ориентированный подход для порождения платформно-специфических моделей нечетких мультиагентных систем, функционирующих в перспективных высокопроизводительных средах, в том числе кластерной системе СКИТ.

Список использованной литературы

1. Zadeh L.A. Fuzzy Logic = Computing With Words / L.A. Zadeh // IEEE Transactions on Fuzzy Systems. – 1996. – Vol. 4. – P. 103-111.
2. Zadeh L.A. The concept of a linguistic variable and its application to approximate reasoning / L.A. Zadeh // Information Sciences. – 1975. – Vol.8, № 8. – P.199–249, P.301—357.
3. Mendel J.M. Type-2 sets made simple / J.M. Mendel, R.I. John // IEEE Trans. on Fuzzy Systems. – 2002. – Vol.10, № 2. – P.117–127.
4. Karnik N.N. Type-2 fuzzy logic system / N.N. Karnik, J.M. Mendel, Q. Liang // IEEE Trans. on Fuzzy Systems. – 1999. – Vol.7, № 6. – P.643–658.
5. Парасюк И.Н. Трансформационный подход к разработке интеллектуальных агентов на основе нечетких моделей / И.Н. Парасюк, С.В. Ершов // Проблеми програмування. – 2011. – № 2. – С. 62–78.
6. Парасюк И.Н. Нечеткие модели мультиагентных систем в распределенной среде / И.Н. Парасюк, С.В. Ершов // Проблеми програмування. – 2010. – № 2–3. – С. 330–339.
7. Ершов С.В. Принципы построения нечетких мультиагентных систем в распределенной среде / С.В. Ершов // Компьютерная математика. – 2009. – № 2. – С. 54–6

ПРОГРАМНЕ СЕРЕДОВИЩЕ ДЛЯ ПОБУДОВИ БІНАРНИХ РОЗВ'ЯЗУВАЛЬНИХ ДІАГРАМ

С.Д. Паращук[1], Е.А. Гончаренко

Кіровоградський державний педагогічний університет
Кіровоград, Україна

[1]sparashchuk@gmail.com

Бінарні розв'язувальні діаграми (Binary Decision Diagram, BDD) широко використовуються в системах автоматизованого проектування, методах формальної верифікації, для стиснення чорно-білих та кольорових зображень, для прискорення виконання запитів баз даних. Загалом BDD є зручними для представлення булевих функцій від великої кількості змінних. Це дає можливість ефективно їх використовувати в задачах штучного інтелекту, перевірки правильності електронних схем, програм, протоколів та інше [1].

Бінарна розв'язувальна діаграма є канонічною формою представлення булевої функції $f(x_1,x_2,\ldots,x_n)$ від *n* змінних у вигляді редукованого орієнтованого ациклічного графу, у якому відсутні повторення у структурі, з одною кореневою вершиною і в загальному випадку двома листками, поміченими 0 і 1. Коренева і проміжні вершини помічені змінними x_i, із кожної вершини виходять рівно два ребра, які відповідають значенням 0 і 1 змінної x_i.

Існують вільно розповсюджувані програмні бібліотеки, у яких реалізовані алгоритми маніпуляцій з BDD. Ці бібліотеки включають програми для побудови BDD за формульним записом булевої функції, виконання булевих операцій над BDD та інше.

Нами розроблене програмне середовище для побудови BDD та виконання операцій над ними. У середовищі реалізована можливість візуального представлення булевої функції за допомогою таблиці істинності, формули, бінарного розв'язувального дерева. Для кожної з форм отримується графічне зображення функції у вигляді BDD. Також підтримується візуалізація операцій над функціями (BDD). Дане програмне середовище розроблене для навчальних цілей і дозволяє наочно представити булеві функції та операції над ними.

Список використаних джерел

1. Карпов Ю. Г. MODEL CHECKING. Верификация параллельных и распределенных программных систем. – СПб.: БХВ-Петербург, 2010. – 560 с.

ИНТЕЛЛЕКТУАЛЬНЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

П.Н. Петренко, П.В. Четырбок

Республиканское высшее учебное заведение «Крымский гуманитарный университет», Ялта, Украина

zknf1994@mail.ru

Интеллектуальные информационные технологии (ИИТ) (англ. *Intellectual information technology, IIT*) — это информационные технологии, помогающие человеку ускорить анализ политической, экономической, социальной и технической ситуации, а также - синтез управленческих решений. При этом используемые методы не обязательно должны быть логически непротиворечивы или копировать процессы человеческого мышления.

Использование ИИТ в реальной практике подразумевает учет специфики проблемной области, которая может характеризоваться следующим набором признаков:

- качество и оперативность принятия решений;
- нечеткость целей и институциальных границ;
- множественность субъектов, участвующих в решении проблемы;
- хаотичность, флюктуируемость и квантованность поведения среды;
- множественность взаимовлияющих друг на друга факторов;
- слабая формализуемость, уникальность, нестереотипность ситуаций;
- латентность, скрытость, неявность информации;
- девиантность реализации планов, значимость малых действий;
- парадоксальность логики решений и др.

Гносеологический фундамент ИИТ наиболее явно видится в работах Канта, Гегеля, Гуссерля. Собственно же явную историю ИИТ удобно начать с середины XX века, когда появился термин «Искусственный интеллект» (Artificial Intelligence). История ИИТ начинается с середины 1970-х годов и связывается с совместным практическим применением интеллектуальных информационных систем, систем интеллекта, систем и информационных систем. Эта история связана также с развитием трех научных направлений: компьютерной философии, компьютерной психологии и продвинутой компьютерной науки (англ.Advanced computer science) и дополняется прогрессом в создании: ситуационных центров, информационно-

аналитических систем, инструментариев эволюционных вычислений и генетических алгоритмов, систем поддержки общения человека с компьютером на естественном языке, когнитивным моделированием, систем автоматического тематического рубрицирования документов, систем стратегического планирования, инструментариев технического и фундаментального анализа финансовых рынков, систем менеджмента качества, систем управления интеллектуальной собственностью и др.

С середины 1940-х вплоть до ранних 1970-х гг. создание ИИТ рассматривалось преимущественно в рамках логического решения задач. Этот период развития ИИТ характеризуется сравнительно большой определенностью и низкой динамичностью объекта управления. Вместе с тем уже в 1943 году появились «продукции Поста» и методы решения некорректных (обратных) задач на метризуемых пространствах, а в 1947 году для моделирования сложных экономических ситуаций активно начали использоваться методы причинного нелогического вывода, которые позже легли в основу методов системной динамики, немонотонных вычислений, когнитивного моделирования.

Создание центров управления полетами, организация штабных работ с применением средств визуализации и автоматизации, зарубежные публикации на тему создания специальных ситуационных центров вдохновили в 1970-е годы инженеров на создание ситуационных комнат для совершенствования управления крупными социальными и институциональными системами. В создании таких комнат и интеллектуальных технологий больше внимания стало придаваться средствам визуализации, диалоговым системам, помогающим использовать базы знаний и модели для решения плохо структурированных проблем. В середине 1970-х годов на основе ИИТ в корпоративном мире начинают развиваться системы поддержки решений для эффективного управления ресурсами, осуществления контроллинга. Ряд замечательных практических идей и результатов, например, связанных с теорией нейронных сетей, многоагентных и активных систем, оптических и голографических процессоров, появилось именно в это время. Тот период можно отметить успехами в создании всеобъемлющих моделей ситуационного управления регионами в периоды кризисов. Его характеризует вера в практически неограниченные возможности искусственного интеллекта.

В середине 1980-х годов был отмечен крах иллюзий относительно неограниченных возможностей успешной формализации процессов мышления с помощью систем логической обработки естественного языка. Вместе с тем появились интеллектуальные технологии для ограниченной поддержки исследовательской и профессиональной деятельности лиц, принимающих решения. Практическое применение получили подходы, основанные на использовании достоверного и

правдоподобного вывода, немонотонных логик и нечетких систем, лингвистических процессоров. Тогда же появилась явная потребность в оптических и квантовых вычислениях – для решения многомерных и слабо распараллеливающихся задач. Видимые успехи появились в сфере обработки текстов естественного языка, высококачественного поиска документов, слежения за динамичными объектами управления, решения задач распознавания образов, имитационного моделирования, статистической обработки данных, решения транспортных задач, построения нечетких контроллеров. В конце 1980-х внимание разработчиков ИИТ все больше акцентируется на исследовании адаптивных свойств информационных систем, учитывающих умственную активность человека при осуществлении речевых актов, дискурса и принятии решений.

С начала 1990 ИИТ все активней используются в стратегическом менеджменте, управлении ресурсами, реинжиниринге, создании ситуационных центров. Все более заметно внедряются интеллектуальные информационные технологии аналитической обработки больших массивов информации, технологии поддержки решений. В 1990-х годах в совокупности и взаимосвязи развиваются: экспертные системы реального времени, интеллектуальные агенты, активные системы, достоверный и правдоподобный вывод, эволюционные и квантовые вычисления, когнитивные модели, ситуационные центры и пр. Эксклюзивное место в развитии ИИТ с середины 1990-х заняла разработка необходимых условий конвергентности (сходимости) процессов управления, поиска информации и синтеза управленческих решений, направленных на обеспечение необходимых условий устойчивой сходимости этих процессов к намечаемым целям.

С 2000 года начал приобретать новое звучание процесс электронизации деятельности органов власти, бизнеса и населения. Концепция электронной демократии, предполагающая: осуществление гражданского контроля, проведение выборов и референдумов, поддержку процессов самоорганизации населения, обеспечение возможности участия населения в принятии государственных решений, расширение технологической возможности обмена мнениями – также предусматривает расширение возможностей интеллектуальных информационных технологий.

Концепции электронной коммерции, включающие: маркетинг, управление корпоративными ресурсами, повышение качества продукции и услуг, расширение доступа к капиталу, электронные торги, развитие инноваций, поддержку процессов самоорганизации бизнеса – не могла не активизировать работы по дальнейшему развитию систем поддержки решений с помощью ИИТ.

Список використаних джерел

1. Браславский П.И., Вовк Е.А., Маслов М.Ю. Фасетная организация интернет-каталога и автоматическая жанровая классификация документов //Компьютерная лингвистика и интеллектуальные технологии. Тр. междунар. семинара "Диалог-2002". Т. 2. - М.: Наука, 2002. - С.83-93.
2. Гаврилова Т.А., Хорошевский В.Ф. Базы знаний интеллектуальных систем. - СПб.: Питер, 2000.
3. Храмцов П. Информационно-поисковые системы Internet // Открытые системы, - 1996. - №3.

ДИАГОНАЛЬНОЕ ПРОИЗВЕДЕНИЕ И НАСЛЕДОВАНИЕ ФУНКЦИЙ СПЕЦИФИКАЦИИ РОДИТЕЛЬСКОГО КЛАССА

А.Г.Пискунов[1], И.А.Петренко[2]

[1]Компьютерная Академия Шаг, Киев,
[2] студент кафедры КИ, радиофизический факультет Киевского национального университета имени Тараса Шевченко, Киев

После некоторого обзора различных трактовок понятий типа и класса [2, 3, 4, 9, 10, 11] авторы становились на следующих определениях. Тип трактуется как спецификация, а именно, четыре раздела, содержащих:

- обозначение одного множества или нескольких множеств (в теоретико-множественном смысле). В самом абстрактном случае, есть только обозначение множества без какого либо уточнения его структуры. Это обозначение будет использоваться в сигнатуре функций. Обозначение желательно использовать даже если известна структура множества, как в случае, рассмотренном ниже. Это множество будет называться доменом типа;
- набор сигнатур функций, определенных на этом(-их) множестве(-ах);
- аксиомы, описывающие поведение функций;
- предусловия использования функций.

Класс трактуется, как реализация его спецификации в некотором языке программирования.

Как показывает практика использования метода формальных спецификаций RAISE [6] множества из спецификации записываются при помощи обычных математических обозначений, функции используются в обычном математическом смысле, без привычных

для программистов скрытых параметров. Такой подход позволил на известном примере нарушения принципа подстановки Лисков [5] формально решить вопрос соотношения понятий типа, класса, наследования и выделения подтипа (см. [7, 8]).

При этом остался нерешенным вопрос можно ли выразить наследуемые функции спецификации подкласса через функции спецификации надкласса. Рассмотрим пример наследования из [4]:

```
class cell is
    var contents: Integer := 0;
    method get(): Integer is
        return self.contents;
    end;
    method set(n: Integer) is
        self.contents := n;
    end;
end;
```

```
subclass reCell of cell is
    var backup: Integer := 0;
    override set(n: Integer) is
        self.backup := self.contents;
        super.set(n);
    end;
    method restore() is
        self.contents := self.backup;
    end;
end;
```

Класс *cell* имеет поле класса *contents* целого типа *Integer* и два метода *set* и *get*. Обозначим через *Q(c)* домен класса *c*, тогда *Q(cell)* будет доменом класса *cell* и синонимом множества целых чисел *Integer.*

Сигнатуры функций из спецификации класса имеют следующий вид $get : Q(cell) \to Integer$ и $set : Integer \to Q(cell)$, а *contents* имеет тип *Q(cell).* Функции спецификации класса, множеством значений которых является домен класса, будем называть функциями состояния. Функции спецификации класса, множество значений которых не имеет в качестве прямого сомножителя домен класса, будем называть функциями выхода.

Далее, наследуемый класс *reCell* имеет уже два поля целого типа *contents* и *backup*, при этом домен *Q(reCell)* оказывается декартовым произведением $Q(cell) \times Integer$, а наследуемая функция выхода

имеет сигнатуру $get : Q(reCell) \to Integer$. Как и выше, *Q(reCell)* является синонимом для $Integer \times Integer$. Очевидно, что функция *get* из спецификации класса *cell* (далее, get_{cell}) и функция *get* из спецификации класса *reCell* (далее, get_{reCell}) являются совершенно различными функциями. При этом, поскольку поле *backup* просто игнорируется наследуемым методом *get*, выполняются следующие соотношения: $get_{reCell} = \lambda(q,i) : Q(reCell) \bullet get_{cell}(q)$ где пара $(q,i) \in Q(reCell)$, или, как указано выше $(q,i) \in Q(cell) \times Integer$ (смысл лямбда-выражения напоминается ниже). Таким образом, функция get_{reCell} оказывается композицией проекции множества *Q(reCell)* на первый сомножитель декартового произведения $Q(cell) - \pi$ и функции get_{cell}. А именно: $get_{reCell} = get_{cell} \circ \pi$.

На таком простом примере легко удалось выразить функцию спецификации наследника через функцию спецификации родителя. Но всегда ли существует способ записи функций наследника через функции родителя, если при наследовании происходит добавление полей? В таком случае степень декартового призведения домена класса наследника оказывается больше степени домена класса родителя. Поэтому любому не статическому наследуемому методу в спецификации наследника должна соответствовать функция, никогда не совпадающая с соответствующей функцией спецификации родителя.

Как было показано в работе [1], оказывается, что любые наследуемые функции из спецификации наследника удается записать через проекции на сомножители декартового произведения, композицию, диагональное произведение и функцию спецификации родителя.

Композиция функций. **П**рименение одной функции к результату другой функции называется композицией функций и обозначается символом ○.

Диагональное произведение функции (см. [12, Частично рекурсивные функции]). Пусть заданы отображения $g_1 : X \to Y_1, ..., g_k : X \to Y_k$. Тогда отображение

$(g_1,...,g_k): X \to Y_1 \times ... \times Y_k$, определенное условием $(g_1,...,g_k)(x) = (g_1(x),...,g_k(x))$ для каждого $x \in X$, называется диагональным произведением отображений $g_1,...,g_k$. Сами отображения $g_1,...,g_k$ называются компонентами диагонального произведения.

Проекция. Отображение $\pi_{X_j} : X_1 \times ... \times X_k \to X_j$, заданное как $\pi_{X_j}(x_1,...,x_k) = x_j$, где *1 <= j <= k* и $x_j \in X_j$, будет называться проекцией.

Лямбда выражение. Далее, символ лямбда λ будет использоваться для записи лямбда выражений над функциями.

Выражение $\lambda val_1 : T_1,..., val_n : T_n \bullet expression(val_1,..., val_n)$ будет определять функцию с сигнатурой $T_1 \times .... \times T_n \to T$, где *T* – множество значений выражения $expression(val_1,..., val_n)$. Например, выражение $\lambda x : Integer, y : Integer \bullet x + y$ определяет функцию сложения двух целых чисел с сигнатурой $Integer \times Integer \to Integer$.

Наследование функций. Пусть есть множества Q_l, Q_1, Q_r, Q_2, связанные соотношением $Q_2 = Q_l \times Q_1 \times Q_r$ – домен класса наследника, а Q_1 – домен класса родителя. Пусть $q_1 \in Q_1, q_l \in Q_l, q_r \in Q_r$, а *X* и *Y* – некоторые множества. Тогда функция $g_2 : Q_2 \times X \to Q_2 \times Y$ находится в отношении наследования с функцией $g_1 : Q_1 \times X \to Q_1 \times Y$, если $g_2 = \lambda(q_l, q_1, q_r, x) : Q_2 \times X \bullet (q_l, \pi_{Q_1}(g_1(q_1, x)), q_r, \pi_Y(g_1(q_1, x)))$, где $x \in X$, то есть, функция g_2 является диагональным произведением функций следующего вида

$$g_2 = (\pi_{Q_l}, \pi_{Q_1} \circ g_1 \circ (\pi_{Q_1}, \pi_X), \pi_{Q_r}, \pi_Y \circ g_1 \circ (\pi_{Q_1}, \pi_X))$$

В более простых случаях, для функций состояния с сигнатурой $f : Q \times X \to Q$ или функций выхода $f : Q \times X \to Y$ для некоторого домена спецификации *Q* и множеств *X, Y* отношения наследования будут задаваться более простыми выражениями.

Наследование функций состояния. Функция $g_2 : Q_2 \times X \to Q_2$ находится в отношении наследования с функцией $g_1 : Q_1 \times X \to Q_1$ если $g_2 = \lambda(q_l, q_1, q_r, x) : Q_2 \times X \bullet (q_l, g_1(q_1, x), q_r)$, где $x \in X$, то есть, функция g_2 является диагональным произведением функций следующего вида: $g_2 = (\pi_{Q_l}, g_1 \circ (\pi_{Q_1}, \pi_X), \pi_{Q_r})$.

Наследование функций выхода. Функция $f_2 : Q_2 \times X \to Y$ находится в отношении наследования с функцией $f_1 : Q_1 \times X \to Y$ если $f_2 = \lambda(q_l, q_1, q_r, x) : Q_2 \times X \bullet f_1(q_1, x)$, где $x \in X$,

то есть, функция f_2 является композицией функций следующего вида $f_2 = f_1 \circ (\pi_{Q_1}, \pi_X)$.

Список использованной литературы

1. A.G. Piskunov. Formalization Of The Oop Paradigm: Inheritance Of Abstract Automata / A.G. Piskunov // Вестник Киевского национального университета имени Тараса Шевченко. – Сер.: Физ.-мат. науки. – 2011. – Вып. 11. – С. 40-44.
2. Benjamin Pierce. Types and Programming Languages / Benjamin Pierce. – London: The MIT Press, 2002.
3. Dines Bj0rner. Software Engineering 1. Abstraction and Modelling / Dines Bj0rner. – Интернет ресурс: http://www2.imm.dtu.dk/~db/Software_Engineering.
4. Martin Abadi. A theory of objects / Martin Abadi, Luca Cardelli. – Springer, 1996. –Интернет ресурс: http:/books.google.com.ua/books?id=4xT3LgCPP5UC).
5. Robert C.Martin. The Liskov Substitution Principle / Robert C.Martin. / C++ Report, March 1996. – Интернет ресурс: http://www.objectmentor.com/resources/articles/lsp.pdf.
6. А.Г. Пискунов. The RAISE Method Group: Алгебраическое проектирование класса / А.Г. Пискунов. – 2007. – Интернет ресурс: http://www.realcoding.net/article/view/4538.
7. А.Г. Пискунов. Об одном примере нарушения принципа подстановки Лисков / А.Г. Пискунов. – 2010. – Интернет ресурс: http://www.realcoding.net/articles/ob-odnom-primere-narusheniya-printsipa-podstanovki-liskov.html.
8. А.Г. Пискунов. Нарушения спецификации класса при наследовании / А.Г. Пискунов, И.А. Петренко // Доклады девятой

международной конференции TAAPSD'2012. – Киев. – 2012.– С. 207-208.
9. А.Г. Пискунов. Типы, множества и классы / А.Г. Пискунов, С.М. Петренко. – 2010. – Интернет ресурс: http://www.realcoding.net/content/typeSetsClasses.pdf.
10. Бенджамин Пирс. Типы в языках программирования / Бенджамин Пирс. – М.: Лямбда Пресс Добросвет, 2012. – Интернет ресурс: с.656, http://newstar.rinet.ru/~goga/tapl/tapl-toc.html.
11. Пискунов А.Г. Классы и типы в языках, основанных на классах / Пискунов А.Г. – 2010. – Интернет ресурс: http://agp.hx0.ru/oop/TvsC.pdf.
12. Манин Ю.И. Вычислимое и невычислимое / Манин Ю.И. – М.: Сов. радио, 1980. – С. 128. – Интернет ресурс: http://agp.hx0.ru/arts/manin2.djvu.

ВИКОРИСТАННЯ МЕТОДУ РОБІНСОНА ДЛЯ ПЕРЕВІРКИ ПРОЦЕДУРНОЇ СЕМАНТИКИ ЛОГІЧНИХ ПРОГРАМ

О.В. Присяжнюк, З.П. Халецька, В.В. Котяк

Кіровоградський державний педагогічний університет
імені Володимира Винниченка, Кіровоград, Україна

olena_drobot@mail.ru, zoya.kh@mail.ru, v_kotyak@kspu.kr.ua

Варто звернути увагу на те, що вивчення методу резолюцій Робінсона повинно створити у студентів певне уявлення про правила логічного аналізу та розуміння змісту автоматичного доведення теорем, що лежать в основі логічного програмування.

Студенти спеціальності «Інформатика» на першому курсі знайомляться з методом резолюцій на базі алгебри висловлень, що застосовується для перевірки виконуваності формул та правильності логічних наслідків. Більш ґрунтовне вивчення методу відбувається на другому курсі для застосування його в логіці предикатів та формальних теорій першого порядку під час вивчення дисципліни «Теорія алгоритмів та математична логіка». При цьому на методі Робінсона не акцентується особлива увага, а він розглядається як один з можливих варіантів виведення логічного висновку. В подальшому навчанні, зокрема в логічному програмуванні, якому студенти навчаються в ході дисципліни «Інтелектуальні системи», використання методу резолюцій має більш практичний характер, а саме для перевірки процедурної семантики роботи програми на коректність та ефективність.

Перевірка коректності роботи програми – одна з важливих задач програмування. Сучасні напрямки перевірки правильності програм – це формальні специфікації, методи доведення, верифікація та тестування. Програма на мові Пролог дуже близька до запису специфікацій, оскільки являє собою набір послідовності формул з логіки предикатів першого порядку, який описує об’єкти та відношення між ними у предметній області задачі, що дозволяє поєднувати процес написання програми з її верифікацією.

У практичному програмуванні на Пролозі основний фактор – це ефективність програм. Основний оцінювальний параметр – число виконуваних уніфікацій і число спроб уніфікацій в процесі обчислення. Цей параметр пов’язаний із часом роботи програми.

Процес створення програми на Пролозі включає декларативний та процедурний рівні програмування. Аналіз програми лише на декларативному рівні програмування не дозволяє

реалізувати ефективний пошук альтернативних рішень у випадку їх комбінаторного перебору, а також виявити логічні помилки у програмі.

На процедурному рівні програмування важливий порядок термів в сукупності однойменних правил і фактів, а також порядок хвостовий цілей в тілі речення, оскільки процедурна семантика синтаксично правильної програми повністю визначається механізмом співставлення термів та прийнятою у Пролозі стратегією пошуку рішень. Порядок цілей (підцілей) впливає на кількість перевірок, що виконуються програмою при спробі задовольнити поточну ціль.

Процес, у результаті якого Пролог-система встановлює, чи задовольняє об'єкт ціль, містить у собі логічний вивід і дослідження різних варіантів. Все це робиться автоматично самою Прологом-системою й, як правило, приховано від користувача.

Головним компонентом інтерпретатора мови Пролог є універсальний механізм розв'язання задач, принцип дії якого заснований на правилі резолюції. Для перевірки процедурної семантики програми і, як наслідок, розуміння як саме інтерпретатор Прологу проводить доведення цілі, при проведенні лабораторних робіт, особливо на початковому етапі, студентам пропонується:

- описати задачу за допомогою фраз Хорна;
- написати запит, згідно якому потрібно вияснити, чи є конкретна атомарна формула наслідком поточної множини фраз;
- для перевірки запита «вручну» застосовувати правило резолюцій до тих пір, поки не буде отримано пустий диз'юнкт, що означає успіх, або ж, поки чергову ціль буде неможливо уніфікувати і з одним диз'юнктом, що означає невдачу.

Перевірка процедурної семантики програми із застосуванням аналізу формування логічного висновку за правилом резолюцій Робінсона дозволяє студентам краще зрозуміти процедурний сенс програми, а в ряді випадків підвищити ефективність роботи програми за рахунок зменшення кількості перегляду можливих альтернатив в процесі узгодження цілі, а отже уникнути комбінаторного переповнення пам'яті.

Список використаних джерел

1. Братко И. Алгоритмы искусственного интеллекта на языке PROLOG, 3-е издание \\ Пер. С анг. –М. Изд.дом «Вильямс», 2004. – 640 с.
2. Глибовець М.М., Олецький О.В. Штучний інтелект. –К.:Вид.дім «КМ Академія», 2002. – 366 с.

МОДЕЛИ НЕЧЕТКОЙ ЛОГИКИ В БИОИНФОРМАТИКЕ

А.И. Провотар

Киевский национальный университет имени Тараса Шевченко,
Киев, Украина

aprowata@unicyb.kiev.ua

Нечеткие системы логического вывода. Под нечеткой системой логического вывода (алгоритмом) [2-5] понимают упорядоченное множество нечетких инструкций, которые при выполнении дают приближенное (нечеткое) решение проблемы.

Пусть x и y - входная и выходная лингвистические переменные [2,3]; A и B - некоторые нечеткие множества, задающие значения элементов терм-множеств переменных x и y соответственно. Простейшим нечетким алгоритмом может быть такая конструкция:

вход (x);
если *x есть* A, ***то*** *y есть* B;
выход (y).

Инструкция "***если*** *x есть* A, ***то*** *y есть* B" интерпретируется как нечеткая импликация $A \rightarrow B$ и, следовательно, задается нечетким отношением на декартовом произведении областей определения (четких множествах) $\boldsymbol{X}$ входной переменной и $\boldsymbol{Y}$ выходной переменной. Выходное значение алгоритма определяется с помощью композиционного правила. А именно, если на вход подается нечеткое множество A', то на выходе получаем нечеткое множество B', которое определяется по формуле

$$B'(y) = \max_{x \in X} \min\,(A'(x), \min\{A(x), B(y)\}), y \in Y.$$

Более сложный нечеткий алгоритм образует конструкция вида:

вход (x);
если *x есть* A_1, ***то*** *y есть* B_1;
если *x есть* A_2, ***то*** *y есть* B_2;
. . .
если *x есть* A_m, ***то*** *y есть* B_m;
выход (y),

где A_i и B_i - нечеткие множества.

Существует два основных способа определения выхода B'. В обоих используется так называемое понятие *агрегации* правил, т.е.

учет сумарного эффекта от работы всех правил. Оператор агрегации ***Agg*** действует как *s*-норма [2], но разрешается использование произвольной *t*-нормы.

Первый способ определения выхода состоит в предварительной агрегации нечетких отношений $R = \mathbf{Agg}(R_1, R_2, ..., R_m)$. Результат B' при заданном входе A' определяется при помощи композиционного правила: $B' = A' \circ R$. Если оператор агрегации есть операцией нахождения максимума, то B' определяется по формуле

$$B' = A' \circ \bigcup_{i=1}^{m} R_i.$$

Второй способ состоит в определении выходов для каждого правила при помощи использования композиции $B_i' = A' \circ R_i$, $i = 1, \ldots, m$. Дальше осуществляется агрегация полученных выходов по правилу $B' = \mathbf{Agg}(B_1', B_2', \ldots, B_m')$, т.е.

$$B' = \bigcup_{i=1}^{m} (A' \circ R_i).$$

Утверждение. При использовании max−min композиций совместно с операцией максимума в роли оператора агрегации результаты, полученные обоими механизмами логического вывода, будут эквивалентными, т.е. справедливо соотношение

$$A' \circ \bigcup_{i=1}^{m} R_i = \bigcup_{i=1}^{m} (A' \circ R_i).$$

Более интересной представляется ситуация, когда алгоритм имеет не один, а несколько входов:

вход ($x_1, x_2, \ldots, x_n$);
если x_1 *есть* $A_{11} \wedge x_2$ *есть* $A_{12} \wedge \ldots \wedge x_n$ *есть* A_{1n} ***то*** y *есть* B_1;
если x_1 *есть* $A_{21} \wedge x_2$ *есть* $A_{22} \wedge \ldots \wedge x_n$ *есть* A_{2n} ***то*** y *есть* B_2;
. . .
если x_1 *есть* $A_{m1} \wedge x_2$ *есть* $A_{m2} \wedge \ldots \wedge x_n$ *есть* A_{mn} ***то*** y *есть* B_m;
выход (y),

где x_j, $j = 1,\ldots, n$ - входные лингвистические переменные, y – выходная лингвистическая переменная; A_{ij} и B_i - нечеткие множества. Логическая связка "$\wedge$" интерпретируется как *t*-норма нечетких множеств. В отличии от случая с одной входной переменной, представление импликации в виде отношения в алгоритмах со многими входными параметрами невозможно. В связи с этим используется другая процедура нахождения выхода, которая использует так называемые уровни истинности правил типа ***если*** x_1 *есть* $A_{i1} \wedge x_2$ *есть* $A_{i2} \wedge \ldots \wedge x_n$ *есть* A_{in} ***то*** y *есть* B_i.

В случае двух входов x_1 і x_2, процедура выполнения алгоритма будет состоять из следующих шагов:

1. Для каждого правила R, $i = 1, \ldots, m$ вычисляем уровень истинности правила

$$\alpha_i = \min\left[\max_{X_1}(A_1'(x_1) \wedge A_{i1}(x_1)), \max_{X_2}(A_2'(x_2) \wedge A_{i2}(x_2)) \right];$$

2. Для каждого правила вычисляем индивидуальные выходи

$$B_i'(y) = \min(\alpha_i, B_i(y));$$

3. Вычисляем агрегатный выход

$$B'(y) = \max(B_1', B_2', \ldots, B_m').$$

Эта процедура называется max−min процедурой или процедурой логического вывода Мамдани (импликация интерпретируется как операция минимум, агрегация выходов правил – как операция максимум).

Утверждение. При использовании max−min композиций и логического вывода Мамдани результаты будут эквивалентными, т.е. справедливо соотношение

$$B'(y) = \max_{x \in X}(A'(x) \wedge (R(x,y)) = \max_{i=1}^{m}(\alpha_i \wedge B_i(y)).$$

Вероятность и возможность в нечетких системах логического вывода. Для того, чтобы определять вероятности событий в пространстве элементарных событий, вводится понятие распределения вероятностей. Это числовая функция $\boldsymbol{P}$, которая присваивает число $\boldsymbol{P}(A)$ элементарному событию A. Область определения функции $\boldsymbol{P}$ расширяется на множество 2^S.

Нечетким событием A в некотором (непустом) универсальном пространстве $\boldsymbol{X}$ (обозначается $A \subseteq \boldsymbol{X}$) называется множество пар

$$A = \{(x, \mu_A(x)), x \in \boldsymbol{X}\},$$

где μ_A: $\boldsymbol{X} \to [0,1]$ – функция принадлежности нечеткого множества A.

Для того, чтобы определять возможность события в пространстве элементарных событий, вводится понятие распределения возможностей $\boldsymbol{\Pi}$.

Если события A_1, A_2, …, A_n взаимоисключающие, то вероятность объединения этих событий равна суме вероятностей этих событий, а пересечение – их произведению. Возможность объединения событий равна максимуму возможностей, а пересечение равно минимуму возможностей этих событий.

Если A – нечеткое множество в пространстве $\boldsymbol{X}$, описывающее четкое событие в этом пространстве, т. е.

$$A = \{(x, \mu_A(x)), x \in \boldsymbol{X}\},$$

то вероятность этого события может быть вычислена по формуле

$$\boldsymbol{P}(A) = \sum_{i \in A} \mu_A(x)\boldsymbol{P}(x),$$

а возможность – по формуле

$$\boldsymbol{\Pi}(A) = \max_{x \in A} \{\mu_A(x)\boldsymbol{\Pi}(x)\}$$

Рассмотрим следующее нечеткое событие (множество) в пространстве $\boldsymbol{X}$

$$A = \{(x, \mu_A(x)), x \in \boldsymbol{X}\}.$$

α-сечение события (множества) A определяется как следующее обычное множество

$$A_\alpha = \{x, \mu_A(x) \geq \alpha\}.$$

Соответственно, вероятность такого события есть

$$\boldsymbol{P}(A_\alpha) = \sum_{x \in A_\alpha} \boldsymbol{P}(x).$$

Здесь A_α есть объединением взаимоисключающих элементарных событий. Вероятность A_α – это сума вероятностей элементарных событий из A_α.

Говорят, что возможность того, что вероятность события A_α есть $\boldsymbol{P}(A_\alpha)$ равна α. Используя такую интерпретацию, определяется нечеткая вероятность $\boldsymbol{P}(A)$ нечеткого события A соответствующая α, а именно

$$\boldsymbol{P}(A) = \{(\boldsymbol{P}(A_\alpha), \alpha), \alpha \in [0,1]\}.$$

Список использованной литературы

1. *Cholewa W., Czogala E.* Podstawy systemow ekspertowych. Warszawa: Prace IBIB PAN, N28, 1989.
2. *Д. Рутковская, М.Пилиньский, Л.Рутковский.* Нейронные сети, генетические алгоритмы и нечеткие системы. М.: Телеком, 2006. – 382 с.
3. *J. Leski*. Systemy neuronowo-rozmyte. Warszawa: Naukowo-Techniczne, 2008. – 690 с.
4. *Л. Катеринич, А. Провотар.* Диагностирование на нейронных сетях в системе Гомеопат // XIII-th International Conference: Knowledge Dialogue Solution. – Sofia, 2007. – V1. – P.64-68.
5. *Zadeh L.A*. Fuzzy sets as a basis for a theory of possibility //Fuzzy Sets ana Systems, 1978, N1, p. 3–28.

ИСПОЛЬЗОВАНИЕ ГРАФОВЫХ БАЗ ДАННЫХ ПРИ ОБРАБОТКЕ РЕЛЯЦИОННЫХ БАЗ ДАННЫХ

Н.Н. Рубан

Донбасский государственный педагогический университет,
Славянск, Украина

rubannn@gmail.com

Введение

Достаточно часто при обработке данных, которые хранятся в реляционной базе, возникает потребность в их анализе. Но зачатую, реляционная структура не может удовлетворить все необходимые потребности, либо обработка такой структуры сложна и трудоемка. Поэтому предлагается совместное использование различных по структуре баз данных, а именно связка MySQL [2] (реляционная) и Neo4j [1] (графовая).

Графовая база данных – это база данных, структура которой представлена в виде графа, при этом данные хранятся либо в вершинах графа, либо в их ребрах. Рассмотрим один из возможных механизмов взаимодействия MySQL и Neo4j.

Опишем условия, которые были выбраны для доступа к данными:

1. База данных MySQL является основным хранилищем данных.
2. Neo4j является вторичным хранилищем данных, которое содержит в себе подмножество основных данных и отвечает следующим условиям:
   a). поддержка OLTP (Online Transaction Processing), доступ к заранее сформированным наборам данных в реальном времени, которые соответствуют определенным условиям.
   b). поддержка пакетного режима – сбор и обработка данных с некоторой задержкой по времени (либо по расписанию).

Таким образом, необходимо иметь быстрый доступ к имеющемуся набору данных, а также возможность синхронизации данных между двумя системами.

Основная часть

Рассмотрим данный подход на примере. Смоделируем базу данных отношений между поставщиками и клиентами в сфере продажи автомобилей. Представим каждый элемент в виде объекта со своими методами (рис. 1).

Объекты связаны посредством идентификатора, который является первичным ключом, индексирование происходит по данному ключу. Методы каждого объекта следят за сохранением

данных в структуру Neo4j, преобразовывая их в узлы, а связи – в ребра графа (рис. 2).

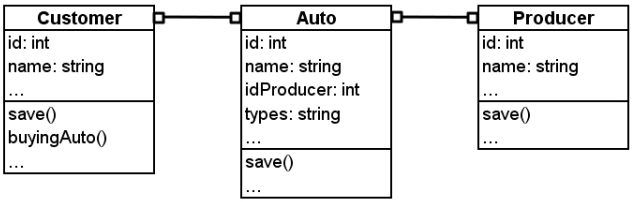


Рис. 1 Модель реляционной базы данных

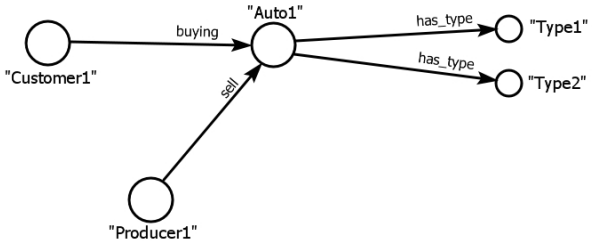


Рис. 2 Модель графовой базы данных

При импорте данных в реляционную базу данных важно следить за корректным преобразованием типов и сохранением данных. Для этого можно использовать SQL-запросы, выбирая при этом только необходимые данные, либо использовать встроенную систему API, которая позволяет импортировать/экспортировать данные.

Воспользуемся SQL-запросами:

SELECT id, name from customers where

Customer customer=new Customer(rs.getInt(“id”), rs.getString(“name”));

customer.save(); //создаем или обновляем узел «Customer»; при создании индексируем по id.

SELECT id, name from auto where

Auto auto=new Auto(rs.getInt(“id”),rs.getString(“name”));

auto.setType(rs.getString(“type”));

auto.save();//создаем или обновляем узел «Auto»; создаем узел «type» если он не существует; создаем отношение между «type» и «Auto»; при создании индексируем по id.

SELECT id, name, type from merchants where

Producer producer=new Producer(rs.getInt(“id”), rs.getName(“name”));

producer.save(); // создаем или обновляем узел «Producer»; при создании индексируем по id.

SELECT customer_id,auto_id,buying_date from customerBuying
Customer customer=repository.getById(rs.getInt(“customer_id”));
customer.buyingAuto(rs.getInt(“product_id”),rs.getDate(“buying_date”);
//создаем отношение между «Customer» и «Auto» и устанавливаем дату покупки как свойство связи.

После того как данные из реляционной базы импортированы и обе системы (MySQL и Neo4j) работают, становится актуальным вопрос синхронизации данных. В зависимости от поставленных задач синхронизацию можно проводить после конкретной операции, либо запускать по расписанию. Либо предусмотреть более сложную процедуру, которая будет отвечать необходимым требованиям и выполнятся после очередного события.

Выводы

Таким образом, рассмотрен метод, который позволяет ускорить обработку информации в реляционной базе данных. При этом проблема использования данных различных типов из различных источников сводится к обычному парсингу. Хотя первичный импорт данных и может занять определенное время, также следует учитывать и то, что это не частый процесс и данное время будет компенсировано за счет ускорения обработки информации в графовой структуре.

Список использованной литературы

1. Jonas Partner, Aleksa Vukotic, Nicki Watt. Neo4j in Action / Jonas Partner, Aleksa Vukotic, Nicki Watt – Manning Publications, 2012. – 350p.
2. Кузнецов М.В. MySQL 5 / Кузнецов М.В., Симдянов И.В. – Спб.: БХВ-Петербург, 2006 – 1024с.: ил.
3. Ian Robinson. Graph Databases / Ian Robinson, Jim Webber, Emil Eifrem – O'Reilly Media, 2013.– 178p.
4. Shashank Tiwari. Professional NoSQL / Shashank Tiwari –Wrox, 2011. – 408p.
5. Эрик Редмонд Семь баз данных за семь недель. Введение в современные базы данных и идеалогию NoSQL / Эрик Редмонд, Джим Р. Уилсон. [под редакцией Жаклин Картер / пер. с англ. Слинкин А.А.] – М.: ДМК Премм, 2013. – 384с.: ил.

ПРИКЛАДНЫЕ ВОПРОСЫ ТЕОРИИ ОБОЗНАЧЕНИЙ[1]

Х.М. Рухая, Л.М. Тибуа

Тбилисский Государственный Университет,
Сухумский Государственный Университет

khimuri.rukhaia@viam.sci.tsu.ge

Резюме: Работа является естественным продолжением статьи [1], в том смысле, что для создания общей теории программирования надо применить тот самый метод, который применяется для создания теории обозначений [2]. В частности, для операторов программирования нужно найти логическуюформу и включить их в ряд языковых символов логической теории.

В качестве примера в статье обобщен оператор присваивания, где переменной присваивается не только числовое значение, но и любой терм [3]. Заметим, что переменная, к которой происходит присваивание терма, встречается как в терме, так и в формуле. Соответственно, оператор присваивания имеет две логические формы, т.е. имеем два типа оператора присваивания Rx и Sx. Результатом действия первого из них является формула, а другого – терм.

Ключевое слово: частичныйквантор, логико-специальный оператор, оператор присваивания, теория обозначений, общая теория программирования.

удк: 510.62; 519.682.1; 519.682.2

Одной из существенных особенностей современных математических теорий (как формальных, так и содержательных) является ограничение и сведение, в некотором разумном смысле, к минимуму алфавита теории. Благодаря этому класс предложений теории становится обозримым, точно определяется понятие доказательства, четко ставится ряд математических проблем и облегчается их решение.

Хотя ограничение алфавита теории теоретически имеет ряд важных существенных преимуществ, тем не менее оно вызывает ряд существенных затруднений. Именно, если бы мы пользовались только символами ограниченного алфавита теории, мы были бы

[1]Работа выполнена в ИПМ имени И.Н. Векуа ТГУ и в Сухумском государственном университете по поддержке научного фонда Грузии им.Шота Руставели (Грант № D/16/4 - 120/11)

вынуждены писать весьма длинные формы (формулы и термы). Практически невозможно как письменное представление таких форм, так и восприятие их содержания на основе их писменного представления. В частности, современный математический язык существенно отходит от обычного (математического) языка и прямое его использование становится невозможным. Для преодоления этих трудностей вводят сокращающие символы и пользуются сокращенными формами. Этим ограниченный язык современной математической теории заменяется языком расширенной сокращающими символами теории, причем последний, оставаясь современным математическим языком, очень близок обычному математическому языку.

Совершенно идентичное положение имеется в случае искусственных языков. Здесь в роли современной математической теории выступает машинный язык Персонального Компютера (в следующем ПК), а в роли расширенной сокращающими символами теории выступают языки программирования. Каждый язык программирования получается введением новых операторов, которые естественно можно назвать сокращающими символами. Язык программирования, противовес машинного языка, близок к обычному.

При составлении программ на языке программирования программист использует общие законы связывающие язык программирования и машинный язык ПК; т.е. общие законы о свойствах операторов (сокращающих символов) языка программирования и о свойствах предложении языка программирования. Но такие общие законы пока не доказаны – пока их использование опирается на интуицию. Это вызывает необходимость отладки программ, что состоит в том, что с целью проверки правильности программы, используют ее для решения простых задач соответствующего рода. Известно, что 70% усилий для программного обеспечения ПК, приходит на отладки программ. При этом, отладка программ осуществляется не безупречным образом. Отладки программ станут излишними, если упомянутые общие законы будут доказаны. Но для такого доказательства необходимо отыскать математическое понятие сокращающего символа (т.е. оператора языка программирования). Иначе говоря, необходимо создать общую теорию языков программирования.

Аналогичное положение было и в математике до 1977 г. Первая попытка создания рациональной теории сокращающих символов на основе введения математического понятия сокращающего символа (вместо интуитивного понятия сокращающего символа) предпринята Ш.С. Пхакадзе в монографии [2] «Некоторые вопросы теории обозначений». Математическое понятие сокращающего

символа вводится на основе рационального ограничения правил введения сокращающих символов. В упомянутой монографии раз и навсегда доказываются важные общие свойства сокращающих символов и сокращеррых форм, если только сокращающие символы вводятся с помощью упомянутого ограниченного семейства правил введения сокращающих символов. При этом указанное семейство правил так богато, что с помощью этих правил можно ввести почти все сокращающие символы, которые применяются в математических теориях. Например, показано, что на основе упомянутого семейства правил введения сокращающих символов можно ввести все сокращающие символы, которые вводятся в монографии Н. Бурбаки «Теория множеств» [4] в явном или в неявном виде. Указанную теорию можно назвать общей теорией сокращающих символов так как она одновременно служит обширному классу математических теорий.

Упомянутая теория сокращающих символов дает возможность существенно повысить логическую строгость математических текстов. А это является необходимым условием для создания автоматических устройств, ориентированных на обработку математических текстов. Создание таких автоматических устройств является частью проблемы искусственного интелекта.

Само собой напрашивается создание общей теории производных операторов (теорию сокращающих символов) для языков программирования с помощью тех же методов, которые применяются для создания теории обозначений. Создание такой теории будет означать создание такой общей теории языков программирования, в которой будут доказуемы общие законы, устанавливающие связь между машинными языками и языками программирования.

Для достижения указанных целей требуется следующее:

Изучить наиболее важные языки программирования. Изучить сокращающие символы (операторы) этих языков и на основе ограничения правил их введения ввести рациональное понятие сокращающего символа(оператора) для языков программирования и развить теорию сокращающих символов для искуственных языков (для языков программирования). От понятия сокращающего символа требуется, чтобы, с одной стороны, оно было настолько общим, что его объем содержал бы все операторы известных искусственных языков, а с другой стороны, условия ограничивающие правила введения сокращающих символов (операторов языков программирования) были настолько сильными, чтобы было возможным установить те нужные общие свойства, которые применяют программисты при составлении программ. Создание такой теории фактически означает создание общей теории

программирования. Это даст программистам возможность составлять надежные программы, не нуждающиеся в проверке с помощью отладки. Кроме того, создание такой общей теории программирования даст возможность выработать рекомендации о том, в каком направлении должна развиваться вычислительная техника, какие автоматические устройства следует создавать с целью обработки математических текстов, какие операции следует выполнять ПК, каковым следует быть машинным языкам и т.д.

Следует особо отметить одну трудность, с которой мы сталкиваемся при реализации упомянутой цели. Определения сокращающих символов языков программирования не даётся в явном виде. Языки программирования создаются почти независимо от машинного языка. Потом создают трансляторы, переводящий программу (текст, написанный на языке программирования) на машинный язык. Поэтому требуется тщательное изучение действия трансляторов. Это даст возможность найти определения операторов (сокращающих символов) языка программирования в явном виде. Необходимость преодоления этой трудности очевидна.

Аналогичные вопросы ставятся в работе Виктора Михаиловича Глушкова "Некоторые проблемы теории автоматов и искусственного интелекта" [3]. В частности, он считает необходимым разработать практический формальный язык для записи математических предложений и их доказательств. При этом этот язык должен относиться к формальным языкам математической логики, а также он должен содержать оператор присваивания общего вида - т.е. переменным должны присваиваться не только числовые значения, но и значения соответствующие различным математическим понятиям.

Нами построена **τSR** логика [1], язык которого содержит операторы подстановки Sx и Rx, являющиеся операторами присваивания общего вида. Чтобы показать, что это так, нам надо проанализировать оператор присваивания, выявить его сущность и найти его место в класификации операторов классической математической логики.

В основном операторы логических теорий можно разбить на шесть групп: Логический реляционный; Логический субстантивный; Специальный реляционный; Специальный субстантивный; Логико-специальный реляционный; Логико-специальный субстантивный.

В этих классификациях под логических (соответственно специальных, соответственно логико-специальных) операторах подразумываются только те операторы, операндами которых являются только формулы (соответственно только термы, соответственно формулы и термы), а под реляционных (соответственно субстантивных) операторах подразумываются

только те операторы, результатом действия которых являются формулы (соответственно термы).

Каждому n–местному логико-специальному оператору приписывается показатель логичности в виде некоторой непустой упорядоченной системы натуральных чисел (n□,n□,…,nk), 1≤n□<n□<…<nk≤n. Показатель логичности указывает все те номера операнда, где стоят формулы. Некоторые операторы являются составными т.е. кванторами. Весом такого квантора является (m, n), где m- число операторных букв, а n – число операндов. Среди них некоторые кванторы являются частичными, т.е. они связывают буквы не всех операндов. К таким операторам приписываются показатели связанности ввиде непустой упорядоченной систеьы натуральных чисел (n□,n□,…,ne), 1≤n□<n□<…<ne≤n, т.е. n□,n□,…,ne номера всех тех операндов, где происходит связывание букв. Например, формальной характеристикой операторов подстановки Rx и Sx является следующее:

1. Оператор Rx (RxTA) являяется двухместным логико-специальным реляционным частичным квантором, показатель логичности и связанности которого является (2) (т.е. вторым операндом оператора Rx является формула, где происходит связывание всех свободных вхождений буквы x.

Легко увидеть, что первый операнд оператора Rx является термом. При этом все свободные вхождения буквы x в первом операнде остаются свободными.

2. Оператор Sx (SxTU) является двухместным специальным субстантивным (т.е. оба операнда – термы, а результат действий также терм) частичным квантором, показателем связаанности которого явля\ется (2) (т.е. квантор Sx связывает все свободные вхождения буквы x только во втором операнде).

Формула RxTA читается так: «Формула полученная из формулы A заменой всех свободных вхождений буквы x в A на T». Мы предполагаем, что «T свободно в A для x» т.е. при замене все свободные вхождения буквы терма T остаются свободными в T после подстановки (это достигается переименованием связанных букв в A).

Терм SxTU читается так: «терм полученный из терма U, заменой всех свободных вхождений буквы x в U на T». Здесь также предполагаем, что «T свободно в U для x».

Предположим, что T есть x+1, а U есть x. Тогда терм SxTU будет терм Sx x+1 x, который читается так: «терм полученный из терма x заменой x на x+1», т.е. терм Sx x+1 x равносильно терму x+1.

Нетрудно увидеть, что терм Sx x+1 τy(x<1) равносильно терму τy[(x+1)<1]. Однако, терм Sx x+y τy[x<y] неравносильно терму τy[(x+y)<y], так как свободная буква y первого операнда x+y оказался связанным в термах τy[(x+y)<y]. Т.е. терм x+y не является свободным в τy[x<y] для x. Следовательно, надо переименовать все связанные вхождения буквы y в τy[x<y], например на z, которая не встречается в терме τy[x<y] и в полученном терме τz[x<z] надо произвести подстановку x+y вместо x, т.е. получим: τz[(x+y)<z].

Чтобы убедится в том, что операторы Sx и Rx являются обобщением оператора присваивания, надо вспомнить что такое оператор присваивания и найти общего с операторами Sx и Rx.

«Оператор присваивания – один из основных операторов в языках программирования, предназначенный для задания или изменения значения одной или нескольких переменных» [5].

«Оператор присваивания в обычных языках программирования можно рассматривать как действие, которое переводит компьютер из одного состояния в другое. Новое состояние присваивания (U,V,W) отличается от предыдущего состояния W тем, что ячейка U содержит V». ([6], стр. 161).

Из ниже приведенных примеров нетрудно увидеть, что оператор присваивания является частным случаем операторов Sx и Rx.

Пример 1. Терм SxTU читается так же «терм T присваивается x в терме U», частным случаем которого является терм Sx T МНОЖЕСТВО(x). Этот терм равносильно терму МНОЖЕСТВО(T). Терм МНОЖЕСТВО(T) читается так же «T множество» или в тексте доказательства "обозначим через T множество" и т.д. (см. [3] стр. 11).

Пример 2. Формула RxTA можно прочитать также «терм T присваивается x в формуле A», частным случаем которой является формула Rx T Группа(x). Эта формула равносильна формуле Группа(T). которая читается так «T есть группа» или в тексте доказательства «пусть T есть группа» и т.д. (см. [3] стр. 11).

Пример 3. В работе [3] (см. стр. 11) оператор «ngm» рассматривается, как одноместный специальный субстантивный оператор. Следовательно, присваивание T: = ngm(x) является типа SxTU т.е. SxT ngm(x) терм равносилен терму «ngm(T)» и читается так: «подмножество множества T». Если оператор «ngm» рассмотрим ни как в работе [3], а как двухместный специальный реляционный оператор, то формула ngm(x,y) читается так: «x есть подмножество множества y», а оператор присваивания T: = ngm(x,y) является типа RxT A, т.е. формула RxT ngm(x,y) равносильна

формуле ngm(T, y) и следовательно читается «T есть подмножество множества y».

Нетрудно доказать следующие своиства операторов Sx и Rx:

1. $[Rx\ \ T\ \ \neg A] \leftrightarrow \neg [Rx\ \ T\ \ A]$.
2. $[Rx\ \ T\ [\ A \vee B]] \leftrightarrow [[Rx\ \ T\ A] \vee [Rx\ T\ B]]$.
3. $[Rx\quad T\quad [A \wedge B] \leftrightarrow [[Rx\ T\ A] \wedge [Rx\ T\ B]]$.
4. $[Rx\quad T\ [A \rightarrow B]] \leftrightarrow [[Rx\ \ T\ A] \rightarrow [Rx\ T\ B]]$.
5. $[Rx\quad T\ [A \leftrightarrow B]] \leftrightarrow [[Rx\ \ T\ A] \leftrightarrow [Rx\ T\ B]]$.
6. $[Rx\ T\quad [T_1 = T_2]] \leftrightarrow [[Sx\ T\ T_1] = [Sx\ \ T\ T_2]]$.
7. $[RxTA] \leftrightarrow [RyTRxyA]$, где *y*-буква не имеющая вхождений в *A*.
8. $[RxTRyUA] \leftrightarrow [RySxTURxTA]$, где *y*-буква не имеющая вхождений в *T*.
9. $[SxT_1SyT_2U] = [SySxT_1T_2SxT_1U]$, где *y*-буква не имеющая вхождений в T_1.
10. $[\tau xA] = [\tau y\ Rxy\ A]$, где *y*-буква не имеющая вхождений в *A*.
11. $[SyT\tau xA] = [\tau xRyTA]$, где *x*-буква не имеющая вхождений в T.

Список использованной литературы

1. Rukhaia Kh; Tibua L; Chankvetadze G; Dundua B. One Method of constructing a formal system; Aplied Mathematics,Informatics and Mechanics(AMIM)T.11N 2; 2006
2. Ш.С. Пхакадзе Некоторые вопросы теории обозначений. Изд. ТГУ, Тбилиси, 1977г.
3. В.М. Глушков; Некоторые проблемы теории автоматов и искусственного интелекта. «Кибернетика» № 2, 1970г
4. Н. Бурбаки; Теория множеств. Изд. Мир, М. 1965г.
5. Словарь по кибернетике. Под редакции академика В.С. Михалевича. Киев 1989г.
6. Р. Ковальский. Логика в решении проблем. М. 1990г.

СВОЙСТВА НАСЫЩЕНИЯ И АКТИВНОГО ДОПОЛНЕНИЯ ДЛЯ ТАБЛИЧНЫХ АЛГЕБР

А.С. Сенченко

Донбасский государственный педагогический университет, Славянск, Украина

senchenko@pisem.net

Введение

В настоящее время системы управления базами данных широко используются практически во всех сферах деятельности человека. Наиболее распространённой является реляционная модель данных, впервые предложенная Э. Коддом в 1970 году [1]. С математической точки зрения реляционная база данных является конечным набором конечных отношений различной размерности между заранее определёнными множествами элементарных данных.

Табличные алгебры, введённые В.Н. Редько и Д.Б. Буем, построены на основе реляционных алгебр Э. Кодда и существенно их уточняют. Они составляют теоретический фундамент языков запросов современных табличных баз данных. Элементы носителя табличной алгебры уточняют реляционные структуры данных, а сигнатурные операции построены на базе основных табличных манипуляций в реляционных алгебрах и языке SQL.

В ставшей уже классической монографии по табличным алгебрам [2] найдено и доказано много различных свойств операций табличных алгебр. В настоящей работе найдены необходимые и достаточные условия, при которых некоторые включения, доказанные в [2], превращаются в равенства.

Основные определения

Зафиксируем некоторое непустое множество $A = \{A_1, \ldots, A_n\}$, элементы которого называются атрибутами. Произвольное конечное подмножество $R = \{A'_1, \ldots, A'_k\} \subseteq A$ назовем схемой, причем схема может являться пустым множеством. Строкой s схемы R называется множество пар $s = \{(A'_1, d_1), \ldots, (A'_k, d_k)\}$, проекция которого по первой компоненте равна R. Таблицей схемы R называется конечное множество строк схемы R. Далее в работе рассматриваем таблицы схемы R с количеством атрибутов k. На множестве всех таких таблиц введены такие операции:

1) объединение $\bigcup$ двух таблиц – таблица, состоящая из тех и только тех строк, которые принадлежат хоть одной из исходных таблиц;

2) пересечение $\bigcap$ двух таблиц – таблица, состоящая из тех и только тех строк, которые принадлежат одновременно обеим исходным таблицам;

3) разность $T_1 - T_2$ двух таблиц – таблица, состоящая из тех и только тех строк, которые принадлежат таблице T_1 и не принадлежат таблице T_2.

Для введения операции активного дополнения нам необходимо дать определение двух понятий, используемых в работе в дальнейшем. Активным доменом атрибута A_i относительно таблицы T называется множество $D_{A_i,T} = \{d \mid \exists s \in T \wedge (A_i, d) \in s\}$, состоящее из всевозможных значений атрибута A_i в таблице T. Насыщением $C(T)$ называется таблица $\prod_{A \in R} D_{A,T}$, где $\prod$ – оператор прямого (декартового) произведения всех атрибутов схемы T. Активным дополнением таблицы T называется таблица $\widetilde{T} = C(T) - T$.

Кроме этих четырёх операций на множестве всех таблиц введены операции проекции, селекции, соединения (в некоторых источниках, например в [3], эта операция называется эквисоединением), деления таблиц и операция переименования атрибутов; эти операции не будут использованы в настоящей работе, поэтому их определения не приводим. Табличной алгеброй называют частичную алгебру с носителем – множеством всех таблиц произвольной схемы и приведёнными выше девятью операциями (насыщение рассматривается как вспомогательная операция). В табличной алгебре выделяют две пустые таблицы: таблицу T_ε, схема которой является пустым множеством и таблицу $T_\varnothing$ – пустое множество строк произвольной (в том числе и непустой) схемы.

Основные результаты

В монографии [2] в подразделе о свойствах насыщения и активного дополнения сформулирован и доказан ряд свойств этих операций, большая часть которых являются включениями. Автором были найдены необходимые и достаточные условия (в виде пяти теорем), при которых шесть из этих включений превращаются в

равенства для непустых таблиц. Доказательства этих теорем являются достаточно громоздкими, поэтому они не будут показаны в настоящей работе, а будут опубликованы в ближайшее время. Кроме формулировки к каждому утверждению будут приведены примеры, в которых будет показано, что выполнение/невыполнение этих критериев приводит к выполнению/невыполнению равенства.

Теорема 1 (дистрибутивность насыщения по объединению). Равенство $C(T_1 \bigcup T_2) = C(T_1) \bigcup C(T_2)$ выполняется тогда и только тогда, когда выполняется хоть одно из двух условий:

а) хотя бы для $k-1$ атрибута значения их активных доменов относительно таблиц T_1 и T_2 попарно совпадают, то есть, существует не более одного атрибута, для которого значения активного домена относительно таблиц T_1 и T_2 различается;

б) значение активного домена каждого атрибута относительно одной таблицы является подмножеством значения активного домена соответствующего атрибута относительно другой таблицы, то есть выполняются включения $\forall i\ D_{A_i,T_1} \subseteq D_{A_i,T_2}$ или $\forall i\ D_{A_i,T_2} \subseteq D_{A_i,T_1}$

□

Пример 1.1.

Пусть $T_1 = \begin{matrix} A & B & C \\ 1 & 2 & 3 \\ 2 & 2 & 3 \\ 2 & 3 & 3 \end{matrix}$ и $T_2 = \begin{matrix} A & B & C \\ 2 & 2 & 1 \\ 2 & 3 & 1 \\ 3 & 3 & 1 \end{matrix}$.

Значения активных доменов $D_{A,T_1} = \{1,2\}, D_{B,T_1} = \{2,3\}, D_{C,T_1} = \{3\}$ и $D_{A,T_2} = \{2,3\}$, $D_{B,T_2} = \{2,3\}, D_{C,T_2} = \{1\}$ не удовлетворяют ни одному из условий, равенство не должно выполняться. Действительно, таблица $C(T_1 \bigcup T_2)$ содержит 4 строки (например, $\{(A,1),(B,2),(C,1)\}$), которые не содержатся ни в $C(T_1)$, ни в $C(T_2)$, следовательно не содержатся в $C(T_1) \bigcup C(T_2)$.

Пример 1.2.

Пусть $T_1 = \begin{array}{ccc} A & B & C \\ 1 & 2 & 3 \\ 2 & 2 & 3 \\ 2 & 3 & 3 \end{array}$ и $T_2 = \begin{array}{ccc} A & B & C \\ 1 & 2 & 4 \\ 1 & 3 & 4 \\ 2 & 3 & 4 \end{array}$.

Значения активных доменов $D_{A,T_1} = \{1,2\}, D_{B,T_1} = \{2,3\}, D_{C,T_1} = \{3\}$ и $D_{A,T_2} = \{1,2\}$, $D_{B,T_2} = \{2,3\}, D_{C,T_2} = \{4\}$ удовлетворяют условию (а), равенство должно выполняться. Действительно,

$$C(T_1 \cup T_2) = \begin{array}{ccc} A & B & C \\ 1 & 2 & 3 \\ 1 & 2 & 4 \\ 1 & 3 & 3 \\ 1 & 3 & 4 \\ 2 & 2 & 3 \\ 2 & 2 & 4 \\ 2 & 3 & 3 \\ 2 & 3 & 4 \end{array} \quad C(T_1) = \begin{array}{ccc} A & B & C \\ 1 & 2 & 3 \\ 1 & 3 & 3 \\ 2 & 2 & 3 \\ 2 & 3 & 3 \end{array} \quad C(T_2) = \begin{array}{ccc} A & B & C \\ 1 & 2 & 4 \\ 1 & 3 & 4 \\ 2 & 2 & 4 \\ 2 & 3 & 4 \end{array}$$

Пример 1.3.

Пусть $T_1 = \begin{array}{ccc} A & B & C \\ 1 & 2 & 3 \\ 2 & 2 & 1 \\ 2 & 3 & 3 \end{array}$ и $T_2 = \begin{array}{ccc} A & B & C \\ 2 & 2 & 1 \\ 2 & 3 & 1 \end{array}$.

Значения активных доменов $D_{A,T_1} = \{1,2\}, D_{B,T_1} = \{2,3\}, D_{C,T_1} = \{1,3\}$ и $D_{A,T_2} = \{2\}, D_{B,T_2} = \{2,3\}, D_{C,T_2} = \{1\}$ удовлетворяют условию (б),

равенство должно выполняться. Действительно,

$$C(T_1 \cup T_2) = \begin{array}{ccc} A & B & C \\ 1 & 2 & 1 \\ 1 & 2 & 3 \\ 1 & 3 & 1 \\ 1 & 3 & 3 \\ 2 & 2 & 1 \\ 2 & 2 & 3 \\ 2 & 3 & 1 \\ 2 & 3 & 3 \end{array}, C(T_1) = \begin{array}{ccc} A & B & C \\ 1 & 2 & 1 \\ 1 & 2 & 3 \\ 1 & 3 & 1 \\ 1 & 3 & 3 \\ 2 & 2 & 1 \\ 2 & 2 & 3 \\ 2 & 3 & 1 \\ 2 & 3 & 3 \end{array}, C(T_2) = \begin{array}{ccc} A & B & C \\ 2 & 2 & 1 \\ 2 & 3 & 1 \end{array}.$$

Теорема 2 (дистрибутивность насыщения по пересечению). При $C(T_1) \cap C(T_2) \neq T_\varnothing$ эквивалентны утверждения:

а) выполняется равенство $C(T_1 \cap T_2) = C(T_1) \cap C(T_2)$;

б) выполняются равенства $\forall i \; D_{A_i, T_1 \cap T_2} = D_{A_i, T_1} \cap D_{A_i, T_2}$;

в) для каждого элемента x, из множества $D_{A_i, T_1} \cap D_{A_i, T_2}$ существует такая строка $s \in T_1 \cap T_2$, что $(A_i, x) \in s$. □

Пример 2.1.

Пусть $T_1 = \begin{array}{ccc} A & B & C \\ 1 & 2 & 1 \\ 1 & 3 & 3 \\ 2 & 3 & 1 \\ 4 & 2 & 1 \\ 4 & 2 & 3 \end{array}$ и $T_2 = \begin{array}{ccc} A & B & C \\ 1 & 1 & 1 \\ 1 & 2 & 1 \\ 3 & 1 & 3 \\ 3 & 2 & 4 \\ 4 & 2 & 3 \end{array}$.

Значения активных доменов $D_{A,T_1} = \{1,2,4\}, D_{B,T_1} = \{2,3\}, D_{C,T_1} = \{1,3\}$ и $D_{A,T_2} = \{1,3,4\}$, $D_{B,T_2} = \{1,2\}, D_{C,T_2} = \{1,3,4\}$; общие значения активных доменов $(A,1),(A,4),(B,2),(C,1),(C,3)$. Общие строки $\{(A,1),(B,2),(C,1)\}$ и $\{(A,4),(B,2),(C,3)\}$ покрывают все общие значения доменов, равенство должно выполняться. Действительно,

$$C(T_1 \cap T_2) = \begin{array}{ccc} A & B & C \\ 1 & 2 & 1 \\ 1 & 2 & 3 \\ 4 & 2 & 1 \\ 4 & 2 & 3 \end{array}, \quad C(T_1) \cap C(T_2) = \begin{array}{ccc} A & B & C \\ 1 & 2 & 1 \\ 1 & 2 & 3 \\ 4 & 2 & 1 \\ 4 & 2 & 3 \end{array}.$$

Пример 2.2.

Пусть $T_1 = \begin{array}{ccc} A & B & C \\ 1 & 1 & 3 \\ 1 & 2 & 2 \\ 1 & 4 & 2 \\ 4 & 1 & 3 \\ 4 & 4 & 2 \end{array}$ и $T_2 = \begin{array}{ccc} A & B & C \\ 1 & 1 & 3 \\ 1 & 2 & 1 \\ 2 & 1 & 3 \\ 2 & 2 & 1 \\ 4 & 1 & 3 \end{array}$.

Значения активных доменов $D_{A,T_1} = \{1,4\}, D_{B,T_1} = \{1,2,4\}, D_{C,T_1} = \{2,3\}$ и $D_{A,T_2} = \{1,2,4\}$, $D_{B,T_2} = \{1,2\}, D_{C,T_2} = \{1,3\}$; общие значения активных доменов $(A,1),(A,4),(B,1),(B,2),(C,3)$. Общие строки $\{(A,1),(B,1),(C,3)\}$ и $\{(A,4),(B,1),(C,3)\}$ не покрывают все общие значения доменов (не покрыто значение $(B,2)$), равенство не должно выполняться. Действительно,

$$C(T_1 \cap T_2) = \begin{array}{ccc} A & B & C \\ 1 & 1 & 3 \\ 4 & 1 & 3 \end{array}, \quad C(T_1) \cap C(T_2) = \begin{array}{ccc} A & B & C \\ 1 & 1 & 3 \\ 1 & 2 & 3 \\ 4 & 1 & 3 \\ 4 & 2 & 3 \end{array}.$$

Теорема 3 (закон двойного отрицания). Равносильны утверждения:

а) выполняется равенство $C(\widetilde{T}) = C(T)$;

б) выполняются равенства $\widetilde{\widetilde{T}} = T$;

в) для каждого значения x активного домена каждого атрибута A_q существуют такие значения $d_1,\ldots,d_{q-1},d_{q+1},\ldots,d_k$ активных доменов

соответствующих атрибутов $A_1,\ldots,A_{q-1},A_{q+1},\ldots,A_k$, что строка $s=\{(A_1,d_1),\ldots,(A_{q-1},d_{q-1}),(A_q,x),(A_{q+1},d_{q+1}),\ldots,(A_k,d_k)\}\notin T$.□

Пример 3.1.

$$T=\begin{array}{ccc} A & B & C \\ 1 & 2 & 2 \\ 1 & 3 & 1 \\ 2 & 2 & 2 \\ 2 & 2 & 3 \end{array}.$$

Значения активных доменов $D_{A,T}=\{1,2\}, D_{B,T}=\{2,3\}, D_{C,T}=\{1,2,3\}$. Для каждого значения активного домена каждого атрибута условие из (в) выполняется, равенства (а) и (б) должны выполняться. Действительно,

$$\widetilde{T}=\begin{array}{ccc} A & B & C \\ 1 & 2 & 1 \\ 1 & 2 & 3 \\ 1 & 3 & 2 \\ 1 & 3 & 3 \\ 2 & 2 & 1 \\ 2 & 3 & 1 \\ 2 & 3 & 2 \\ 2 & 3 & 3 \end{array},\quad \widetilde{\widetilde{T}}=\begin{array}{ccc} A & B & C \\ 1 & 2 & 2 \\ 1 & 3 & 1 \\ 2 & 2 & 2 \\ 2 & 2 & 3 \end{array};\quad \widetilde{\widetilde{T}}=T \text{ и } C\left(\widetilde{T}\right)=C(T).$$

Пример 3.2.

$$T=\begin{array}{ccc} A & B & C \\ 1 & 2 & 2 \\ 1 & 3 & 1 \\ 1 & 3 & 2 \\ 2 & 2 & 2 \\ 2 & 3 & 2 \end{array}.$$

Значения активных доменов $D_{A,T} = \{1,2\}, D_{B,T} = \{2,3\}$, $D_{C,T} = \{1,2\}$. Для значения $2 \in D_{C,T}$ все соответствующие строки в таблице T есть; условие из (в) не выполняется, равенства (а) и (б) не должны выполняться. Действительно,

$$\widetilde{T} = \begin{array}{ccc} A & B & C \\ 1 & 2 & 1 \\ 2 & 2 & 1 \\ 2 & 3 & 1 \end{array}, \quad \widetilde{\widetilde{T}} = \begin{array}{ccc} A & B & C \\ 1 & 3 & 1 \end{array}; \quad \widetilde{\widetilde{T}} \neq T \text{ и } C\left(\widetilde{\widetilde{T}}\right) \neq C(T).$$

Теорема 4 (закон преобразования разности). Равенство $T_1 \bigcap \widetilde{T}_2 = T_1 - T_2$ при $T_1 - T_2 \neq T_\varnothing$ выполняется тогда и только тогда, когда выполняется включение $\forall i \; D_{A_i,T_1} \subseteq D_{A_i,T_2}$.□

Пример 4.1.

$$\text{Пусть } T_1 = \begin{array}{ccc} A & B & C \\ 1 & 2 & 1 \\ 1 & 3 & 1 \\ 2 & 3 & 1 \\ 2 & 3 & 3 \end{array} \text{ и } T_2 = \begin{array}{ccc} A & B & C \\ 1 & 2 & 1 \\ 2 & 2 & 2 \\ 2 & 3 & 1 \\ 4 & 2 & 3 \end{array}.$$

Все включения активных доменов выполняются, равенство должно выполняться. Действительно,

$$T_1 \bigcap \widetilde{T}_2 = \begin{array}{ccc} A & B & C \\ 1 & 3 & 1 \\ 2 & 3 & 3 \end{array}, \; T_1 - T_2 = \begin{array}{ccc} A & B & C \\ 1 & 3 & 1 \\ 2 & 3 & 3 \end{array}.$$

Пример 4.2.

$$\text{Пусть } T_1 = \begin{array}{ccc} A & B & C \\ 1 & 2 & 1 \\ 1 & 3 & 1 \\ 2 & 2 & 3 \\ 4 & 2 & 3 \\ 4 & 3 & 1 \end{array} \text{ и } T_2 = \begin{array}{ccc} A & B & C \\ 1 & 1 & 1 \\ 1 & 1 & 3 \\ 2 & 1 & 3 \\ 2 & 3 & 3 \\ 4 & 3 & 1 \end{array}.$$

Одно включение не выполняется (D_{B,T_1} не является подмножеством D_{B,T_2}), равенство $T_1 \bigcap \widetilde{T}_2 = T_1 - T_2$ должно не выполняться. Действительно,

$$T_1 \bigcap \widetilde{T}_2 = \begin{array}{ccc} A & B & C \\ 1 & 3 & 1 \end{array}, \; T_1 - T_2 = \begin{array}{ccc} A & B & C \\ 1 & 2 & 1 \\ 1 & 3 & 1 \\ 2 & 2 & 3 \\ 4 & 2 & 3 \end{array}.$$

Теорема 5 (закон де Моргана). Равенство $\widetilde{T}_1 \bigcap \widetilde{T}_2 = \left(T_1 \tilde{\bigcup} T_2 \right)$ при $\left(T_1 \tilde{\bigcup} T_2 \right) \neq T_{\varnothing}$ выполняется тогда и только тогда, когда выполняется одно из четырёх взаимоисключающих условий:

1) $\forall i \; D_{A_i,T_1} = D_{A_i,T_2}$;

2) $\forall i \; D_{A_i,T_2} \subseteq D_{A_i,T_1}$, и для всех индексов q, $x \in D_{A_q,T_1} - D_{A_q,T_2}$ и всех $d_1,\ldots,d_{q-1},d_{q+1},\ldots,d_k$, принадлежащих соответственно $D_{A_1,T_1},\ldots,D_{A_{q-1},T_1},D_{A_{q+1},T_1},\ldots,D_{A_k,T_1}$, строки $\{(A_1,d_1),\ldots,(A_{q-1},d_{q-1}),(A_q,x),(A_{q+1},d_{q+1}),\ldots,(A_k,d_k)\} \in T_1$;

3) $\forall i \; D_{A_i,T_1} \subseteq D_{A_i,T_2}$, и для всех индексов q, $x \in D_{A_q,T_2} - D_{A_q,T_1}$ и всех $d_1,\ldots,d_{q-1},d_{q+1},\ldots,d_k$, принадлежащих соответственно $D_{A_1,T_2},\ldots,D_{A_{q-1},T_2},D_{A_{q+1},T_2},\ldots,D_{A_k,T_2}$, строки $\{(A_1,d_1),\ldots,(A_{q-1},d_{q-1}),(A_q,x),(A_{q+1},d_{q+1}),\ldots,(A_k,d_k)\} \in T_2$;

4) Существует такой индекс q, для которого существуют $x \in D_{A_q,T_1} - D_{A_q,T_2}$, $y \in D_{A_q,T_2} - D_{A_q,T_1}$ и $D_{A_q,T_2} \bigcap D_{A_q,T_1} \neq \varnothing$; для всех $i \neq q$ выполняются равенства $D_{A_i,T_1} = D_{A_i,T_2}$;0 для всех $x \in D_{A_q,T_1} - D_{A_q,T_2}$, всех $y \in D_{A_q,T_2} - D_{A_q,T_1}$ и всех $d_1,\ldots,d_{q-1},d_{q+1},\ldots,d_k$, принадлежащих соответственно $D_{A_1,T_1},\ldots,D_{A_{q-1},T_1},D_{A_{q+1},T_1},\ldots,D_{A_k,T_1}$, строки

$\{(A_1,d_1),\ldots,(A_{q-1},d_{q-1}),(A_q,x),(A_{q+1},d_{q+1}),\ldots,(A_k,d_k)\}\in T_1$ и строки $\{(A_1,d_1),\ldots,(A_{q-1},d_{q-1}),(A_q,y),(A_{q+1},d_{q+1}),\ldots,(A_k,d_k)\}\in T_2$.□

Пример 5.1.

Пусть $T_1=\begin{array}{ccc} A & B & C \\ 1 & 2 & 1 \\ 2 & 2 & 1 \\ 2 & 2 & 2 \\ 2 & 3 & 1 \\ 2 & 3 & 2 \\ 3 & 3 & 1 \end{array}$ и $T_2=\begin{array}{ccc} A & B & C \\ 1 & 2 & 2 \\ 3 & 2 & 2 \\ 3 & 3 & 1 \end{array}$.

Значения активных доменов $D_{A,T_1}=\{1,2,3\}, D_{B,T_1}=\{2,3\}, D_{C,T_1}=\{1,2\}$, $D_{A,T_2}=\{1,3\}, D_{B,T_2}=\{2,3\}, D_{C,T_2}=\{1,2\}$. Условие (2) выполняется: для $2\in D_{A,T_1}-D_{A,T_2}$ все необходимые строки в T_1 есть, поэтому равенство должно выполняться. Действительно,

$\left(T_1\tilde{\cup}T_2\right)=\begin{array}{ccc} A & B & C \\ 1 & 3 & 1 \\ 1 & 3 & 2 \\ 3 & 2 & 1 \\ 3 & 3 & 2 \end{array}$, $\tilde{T}_1\cap\tilde{T}_2=\begin{array}{ccc} A & B & C \\ 1 & 3 & 1 \\ 1 & 3 & 2 \\ 3 & 2 & 1 \\ 3 & 3 & 2 \end{array}$.

Пример 5.2.

Пусть $T_1=\begin{array}{ccc} A & B & C \\ 1 & 2 & 1 \\ 2 & 2 & 1 \\ 2 & 3 & 1 \\ 2 & 3 & 2 \\ 3 & 3 & 1 \end{array}$ и $T_2=\begin{array}{ccc} A & B & C \\ 1 & 2 & 2 \\ 3 & 2 & 2 \\ 3 & 3 & 1 \end{array}$.

Значения активных доменов $D_{A,T_1}=\{1,2,3\}, D_{B,T_1}=\{2,3\}, D_{C,T_1}=\{1,2\}$, $D_{A,T_2}=\{1,3\}, D_{B,T_2}=\{2,3\}, D_{C,T_2}=\{1,2\}$. Условие (2) не выполняется: для $2\in D_{A,T_1}-D_{A,T_2}$ в таблице T_1 нет строки

$\{(A,2)(B,2),(C,2)\}$, условия (1), (3), (4) тоже не выполняются, поэтому равенство не должно выполняться. Действительно,

$$\left(T_1 \tilde{\cup} T_2\right) = \begin{array}{ccc} A & B & C \\ 1 & 3 & 1 \\ 1 & 3 & 2 \\ 2 & 2 & 2 \\ 3 & 2 & 1 \\ 3 & 3 & 2 \end{array}, \quad \tilde{T}_1 \cap \tilde{T}_2 = \begin{array}{ccc} A & B & C \\ 1 & 3 & 1 \\ 1 & 3 & 2 \\ 3 & 2 & 1 \\ 3 & 3 & 2 \end{array}.$$

Выводы

В работе исследованы свойства операций насыщения и активного дополнения табличных алгебр. Найдены критерии, при которых некоторые включения из [2] превращаются в равенства. Результаты работы могут быть использованы в теории обобщенных табличных алгебр и, на наш взгляд, для оптимизации запросов в реляционных базах данных.

Автор благодарит Дмитрия Борисовича Буя за постановку задачи и полезные замечания.

Список использованой литературы

1. Codd E.F. A Relational Model of Data for Large Shared Data Banks / E. F. Codd // Communications of the ACM. – 1970. – V. 13, N. 6. – P. 377–387.
2. Реляційні бази даних: табличні алгебри та SQL-подібні мови / [В.Н. Редько, Ю.Й. Брона, Д.Б. Буй, С.А. Поляков]. – Київ: Видавничий дім «Академперіодика», 2001. – 198 с.
3. Мейер Д. Теория реляционных баз данных: [пер. с англ.] / Д. Мейер. – Москва: Мир, 1987. – 608 с.

ДЕЯКІ ПИТАННЯ СПЕЦИФІКАЦІЙ ОБМЕЖЕНЬ ПРИ ПРОЕКТУВАННІ ПРОГРАМНИХ СИСТЕМ

Л.М. Сільвейструк, О.В. Шишацька

Київський національний університет імені Тараса Шевченка,
Київ, Україна

slm-klm@ukr.net, oshyshatska@gmail.com

Вступ

Моделювання семантики – одна з найважчих задач в проектуванні програмних систем. Для цього в різних моделях використовуються обмеження цілісності, що виражають обмеження, які накладаються предметною областю, визначаються зв'язки між компонентами і описується поведінка системи. В роботі наведено класифікації обмежень та розглянуто обмеження в моделі «сутність-зв'язок».

Поняття обмеження. Класифікація обмежень

Модель програмної системи – це набір тверджень, що математично строго визначає структурні властивості об'єктів системи, їх поведінку. При аксіоматичному підході вона задається описом структури і операторів. Властивості структур задаються аксіомами – формальними твердженнями, достатньо простими, щоб бути самоочевидним. Семантика задається аксіомами, або обмеженнями цілісності, що описують допустимі стани і послідовності станів системи. Загальний систематизований вступ до теорії обмежень цілісності в базах даних наведено в [1].

Обмеження цілісності можна поділити на: *статичні обмеження цілісності* – представляють семантику усіх можливих станів системи; *динамічні обмеження цілісності* – описують поведінку системи в часі, тобто коректність послідовності її станів.

Статистичні обмеження цілісності можуть бути розподілені на наступні класи: *структурні обмеження* (використовуються при проектуванні системи і накладаються на її структуру, наприклад включення, виключення, функціональні залежності); *семантичні обмеження* (використовуються при проектуванні системи і накладаються на її семантику, наприклад, функціональні, багатозначні залежності); *обмеження реалізації* (використовуються для представлення системи, наприклад, включення, об'єднання, залежності, породжені кортежами); *обмеження проектування* (використовуються для побудови зручного інтерфейсу, наприклад, загальні функціональні і узагальнені функціональні залежності).

Можна показати, що ці типи обмежень можуть бути використані і для динамічних обмежень цілісності. Динамічні обмеження використовують для підтримки структури даних. Немає загального підходу до використання динамічних обмежень цілісності. За своїм значенням вони можуть бути розподілені на *обмеження переходу* (описують операції над системою, зміну її станів, наприклад, перед- і післяумови, що накладаються на операції зміни даних) та *тимчасові формули* (обмеження на послідовності станів).

Ця класифікація включає і приховані і явні обмеження. Різниця між цими типами обмежень залежить від використовуваної моделі. В реляційній моделі всі обмеження представлені разом. Це приводить до складнощів в їх класифікації. Однак, для деяких типів обмежень цілісності таких як функціональні залежності і залежності включення, аксіоматизація можлива лише в рамках логіки першого порядку. При ER-підході структурні обмеження моделюються внутрішніми обмеженнями, такими як залежності включення, які задані в структурі схеми. Більшість розширень моделі «сутність-зв'язок» підтримують різні типи функціональних залежностей, такі як один-до-одного, один-до-багатьох та кардинальні обмеження. Перевагою таких обмежень є простота опису при проектуванні системи.

Обмеження в моделі «сутність-зв'язок»

У роботах [2, 3] було розглянуто обмеження, що накладаються на типи зв'язків у моделі «сутність-зв'язок». Розглянемо типові обмеження, що накладаються на атрибути: ключ, обмеження домена, обмеження кардинальності.

Одне з базових понять, яке використовується при проектуванні програмних систем, – це атрибут (attribute). За допомогою атрибутів визначаються властивості об'єктів. Так, типу сутності та типу зв'язку відповідає певний набір атрибутів. Тип сутності може характеризуватися принаймні одним атрибутом, а тип зв'язку може і не мати атрибутів.

Має місце поняття множини значень, які може приймати атрибут – *домен атрибуту* (*domain*). Домен визначає всі потенціальні значення, що можуть бути присвоєні атрибуту. Зрозуміло, що різні атрибути можуть використовувати один домен.

Як зазначалось вище, можна розміщувати атрибути на типі зв'язку, але не потрібно зловживати такою можливістю; в якості альтернативи використовують наступне: вводять в модель новий тип сутності з тими ж атрибутами та з'єднують його з типом зв'язку (знищуючи його атрибути), отримуючи новий тип зв'язку, який має арність на одиницю більше (див., наприклад, [4]).

Сутності та зв'язки характеризуються значеннями своїх атрибутів. Розглядаючи типи сутностей та типи зв'язків на рівні атрибутів і доменів, вони уточнюються наступним чином.

Нехай $A_1, A_2, ..., A_n$ атрибути типу сутності E або типу зв'язку R, а D – універсальний домен (для спрощення позначень розглядається один домен для всіх атрибутів).

Типом сутності E називається множина відображень вигляду $e: \{A_1, A_2, ..., A_n\} \to D$, де e – це сутність типу сутності E.

Типом зв'язку R називається множина відображень вигляду $r: \{A_1, A_2, ..., A_m\} \to D$, причому множина атрибутів $\{A_1, A_2, ..., A_m\}$ містить атрибути типу зв'язку R та атрибути всіх сутностей-учасниць даного типу зв'язку, де r – це зв'язок типу зв'язку R [5]. У роботі [5] Чен уточнює атрибут як функцію, що відображає тип сутності (тип зв'язку) в домен.

Розрізняють різні класи атрибутів: *прості* (*simple*) та *складові* (*composite*) *атрибути* (*attributes*), *однозначні* (*single-valued*) та *багатозначні* (*multivalued*) *атрибути*, *базові* (*stored*) та *похідні* (*derived*) *атрибути*, *ключі* (*keys*).

Простий атрибут – атрибут, який складається з одного компонента з незалежним існуванням. Складовий атрибут – атрибут, який складається з декількох компонентів, кожен з яких характеризується незалежним існуванням.

Прості атрибути іноді називають *атомарними*. Складовий атрибут представляє деяку композицію простих (або складових) атрибутів. Такі взаємозв'язки вказують на ієрархію атрибутів.

Рішення про моделювання атрибута у вигляді простого чи складового залежить лише від проектанта.

Однозначний атрибут – атрибут, який містить одне значення для будь-якої сутності типу сутності (будь-якого зв'язку типу зв'язку). Багатозначний атрибут – атрибут, який може містити декілька значень для сутності типу сутності (зв'язку типу зв'язку).

В більшості реалізацій моделі «сутність-зв'язок» багатозначний атрибут необхідно знищувати шляхом створення нових слабкого типу сутності та бінарного типу зв'язку виду «один до багатьох» (слабкий тип сутності та бінарний тип зв'язку виду «один до багатьох» розглядаються у наступних розділах).

Похідний атрибут – атрибут, який представляє значення, похідне від значення зв'язаного з ним атрибута або деякої множини атрибутів, які належать деякому (не обов'язково даному) типу сутності (типу зв'язку) або множині типів сутностей (типів зв'язків). Базовими атрибутами вважаються атрибути, які не є похідними.

Серед похідних атрибутів потрібно розрізняти атрибути, що використовують значення декількох атрибутів однієї сутності, та атрибути, що використовують значення декількох атрибутів всіх сутностей в межах одного типу сутності. В другому випадку похідний атрибут є атрибутом типу сутності, так як використовує всі значення сутностей цілого типу (ситуація аналогічна використанню агрегатних функцій).

Також похідні атрибути можуть обраховуватися на основі декількох взаємопов'язаних атрибутів різних типів сутностей; при цьому необхідно уточнити типи зв'язків між даними типами сутностей.

Виділяють особливий вид атрибутів – ключі. *Ключ* – атрибут або підмножина атрибутів, яка унікальним чином визначає сутність в складі типу сутності (зв'язок в складі типу зв'язку); іншими словами, дві різні сутності (два різних зв'язка) у межах типу сутності (типу зв'язку) не можуть мати однакові значення всіх атрибутів, які складають ключ.

У термінах реляційного підходу ключ K функціонально обумовлює всі атрибути A_i типу сутності (типу зв'язку), які не входять до ключа $(K \to A_i)$ (див., наприклад, [4, с. 108]).

Дане твердження можна уточнити за допомогою поняття обмеження відношення P (зокрема, функції) за множиною X – $\mathrm{P}|\mathrm{X}$ [6].

Множина атрибутів K є ключем, якщо виконується твердження $\forall e_1 \ \forall e_2 \ (e_1, e_2 \in \mathrm{E} \wedge e_1|\mathrm{K} = e_2|\mathrm{K} \Rightarrow e_1 = e_2)$.

Виділяють наступні види ключів:

- *потенційний ключ* (*candidate key*) – атрибут або набір атрибутів, які унікально ідентифікують сутності типу сутності (зв'язки типу зв'язку);
- *первинний ключ* (*primary key*) – деякий вибраний потенціальний ключ типу сутності (типу зв'язку) для спрощення роботи з сутностями (зв'язками);
- *альтернативний ключ* (*alternative key*) – потенціальний ключ, який не є первинним ключем.

Зрозуміло, що тип сутності або тип зв'язку має один первинний ключ, а потенційних ключів може бути декілька.

Для отримання додаткової інформації про об'єкти моделі на них накладаються обмеження. Одне з обмежень, яке застосовують до атрибутів типу сутності (типу зв'язку), це визначення ключа.

Інше обмеження – це обмеження на значення елементів схеми (структури) даних – *обмеження домену* (*domain constraint*), яке являє

собою явне задання множини допустимих значень, а також співвідношення між значеннями атрибутів, які повинні виконуватися. Множину значень можна задавати декількома способами, у тому числі перерахуванням допустимих значень, завданням стандартних типів (int, float, string і т.п.) або завданням умови приналежності. При описі функцій приналежності та при визначенні допустимих відношень можливо застосовувати операції порівняння, логічні зв'язки та інші засоби мови, що використовується для визначення обмежень цілісності.

На атрибути може накладатися також *обмеження кардинальності* (*cardinality constraints*), яке відображає найменшу та найбільшу можливу кількість значень атрибуту для сутності (зв'язку) і задається у вигляді відповідного діапазону.

Мінімальна кардинальність атрибута показує кількість екземплярів атрибуту, що повинні існувати, щоб об'єкт був допустимий. Зазвичай – це значення рівне 0 або 1. Якщо воно рівне 0, атрибут не зобов'язаний мати значення, а якщо 1, то атрибут зобов'язаний мати значення.

Максимальна кардинальність атрибута показує максимальну кількість екземплярів атрибуту, які може мати об'єкт. Зазвичай значення рівне 1 або N. Якщо воно рівне 1, атрибут може мати не більш одного екземпляра, якщо воно рівне N, то гранична кількість екземплярів не задана.

Особливість даного обмеження в моделі «сутність-зв'язок» є те, що значення мінімальної та максимальної кардинальності атрибута може бути деякими натуральними числами, тоді в моделі чітко регламентується як мінімальна так і максимальна кількість екземплярів атрибута.

Кардинальність атрибута зображається у вигляді пари значень (M, N), де M – значення мінімальної кардинальності, а N – значення максимальної кардинальності.

При побудові моделі даних одне з базових питань – це коректність побудови моделі. Тому встановлення логічних взаємозв'зків між об'єктами моделі та їх обмеженнями – важливе питання проектування.

При розгляді атрибутів можна встановити наступні зв'язки:

- якщо атрибут ключовий, то мінімальна та максимальна кардинальність атрибуту дорівнюють 1;
- якщо атрибут однозначний, то максимальна кардинальність атрибуту дорівнює 1;
- якщо атрибут многозначний, то максимальна кардинальність атрибуту більше 1 або N.

Висновки

В роботі розглянуто обмеження в моделі «сутність-зв'язок», . В подальшому планується уточнити вище викладенні логічні зв'язки для перевірки коректності побудови атрибутів та їх обмежень у моделі.

Список використаних джерел

1. Тальхайм Б. Обзор семантических ограничений для моделей баз данных / Б.Тальхайм // Интеллектуальные системы. – 1998. – Т.3. – Вып. 3-4. – С. 307-351.
2. Буй Д.Б. Формалізація моделі „сутність-зв'язок" / Д.Б. Буй, Л.М. Сільвейструк // Київ: Видавничо-поліграфічний центр „Київський університет". – 2011. – 186 с.
3. Сільвейструк Л.М. Узагальнена кардинальність участі в моделі «сутність-зв'язок» / Л.М. Сільвейструк // Вісник Київського національного університету імені Тараса Шевченка. Серія фізико-математичні науки. – 2011. – Вип. 2. – С. 142-146.
4. Гарсиа-Молина Г. Системы баз данных: [полный курс.: пер. с англ.] / Г. Гарсиа-Молина, Дж. Ульман, Дж. Уидом. – Москва: „Вильямс", 2004. – 1088 с.
5. Chen P.P. The entity-relationship model – towards a unified view of data / P.P. Chen // ACM Transactions on Database Systems. – March 1976. – Vol. 1, No. 1. – P. 9-36. – Переклад на російську мову можна знайти за адресою: http://www.osp.ru/dbms/1995/03/271.htm.
6. Буй Д.Б. Властивості теоретико-множинних конструкцій повного образу та обмеження / Д.Б. Буй, Н.Д. Кахута // Вісник Київського університету. Сер.: фіз.-мат. науки. – 2005. – Вип. 2. –С. 232-240.

ПРОВЕРКА ВЫПОЛНИМОСТИ ФОРМУЛ ЛИНЕЙНОЙ АРИФМЕТИКИ НАД КОНЕЧНЫМ КОЛЬЦОМ

В.В. Скобелев

Институт прикладной математики и механики НАН Украины,
Донецк, Украина

vv_skobelev@iamm.ac.donetsk.ua

Введение

Применение алгебраических моделей при решении задач защиты информации обосновывает актуальность исследования автоматных моделей над конечными кольцами. Анализ таких автоматов естественно приводит к задаче проверки выполнимости формул арифметики над конечным кольцом.

В последнее время проблеме выполнимости формул разрешимых теорий 1-го порядка [1] уделяется значительное внимание, а полученные результаты нашли применение для широкого класса задач, включающего в себя как формальную верификацию, так и составление расписаний.

В настоящей работе исследуется задача проверки выполнимости формул линейной арифметики над ассоциативным конечным кольцом [2].

Основные понятия

Будем рассматривать конечные кольца $\mathsf{K} = (K, +, \cdot)$ с ненулевым умножением (в противном случае задача проверки выполнимости формул над кольцом сводится к аналогичной задаче для абелевой группы). Множество всех рассматриваемых колец разбивается на следующие шесть непустых непересекающихся подмножеств:

1) множество коммутативных колец с единицей;
2) множество коммутативных колец без единицы;
3) множество некоммутативных колец с единицей;
4) множество некоммутативных колец с левыми единицами, но без правых единиц;
5) множество некоммутативных колец с правыми единицами, но без левых единиц;
6) множество некоммутативных колец без односторонних единиц.

Указанное разбиение колец дает возможность выделить случаи, в которых решение исследуемой задачи существенно различается.

Простейшими атомами линейной арифметики $LA(\mathsf{K})$ над кольцом K назовем формулы $ax \Diamond b$, $xa \Diamond b$ и $a_1 x a_2 \Diamond b$, где $a, a_1, a_2, b \in K$ – фиксированные элементы, а $\Diamond \in \{=, \neq\}$ (так как в

конечном кольце невозможно определить никакое естественное отношение линейного порядка, то «неравенства» в конечном кольце определяются именно отношением $\neq$), а атомами $LA(\mathsf{K})$ назовем системы линейных уравнений и линейных неравенств. Таким образом, любая формула $LA(\mathsf{K})$ является конъюнкцией некоторого множества простейших атомов и атомов $LA(\mathsf{K})$.

В [3] предложен метод решения систем уравнений над конечным кольцом, основанный на использовании классов левых и правых ассоциированных элементов кольца (в случае некоммутативного кольца) или классов ассоциированных элементов кольца (в случае коммутативного кольца). Такой подход дает возможность свести поиск решений к поиску обратимых элементов кольца (что намного проще, чем поиск неизвестных по всему кольцу). Этот метод эффективно используется при решении исследуемой задачи.

Основные результаты

На основе слоистой техники (layering technique) построены $LA(\mathsf{K})$- решатели $\mathsf{S}^{(1)}_{LA(\mathsf{K})}$, $\mathsf{S}^{(2)}_{LA(\mathsf{K})}$, $\mathsf{S}^{(3)}_{LA(\mathsf{K})}$ и $\mathsf{S}^{(4)}_{LA(\mathsf{K})}$, каждый из которых представляет собой иерархию трех модулей, что:

1) $LA(\mathsf{K})$-решатель $\mathsf{S}^{(1)}_{LA(\mathsf{K})}$ предназначен для анализа выполнимости простейших термов вида $ax = b$, $xa = b$ и $a_1 x a_2 = b$;

2) $LA(\mathsf{K})$-решатель $\mathsf{S}^{(2)}_{LA(\mathsf{K})}$ предназначен для анализа выполнимости простейших термов вида $ax \neq b$, $xa \neq b$ и $a_1 x a_2 \neq b$;

3) $LA(\mathsf{K})$-решатель $\mathsf{S}^{(3)}_{LA(\mathsf{K})}$ предназначен для анализа выполнимости термов вида $a_{i1}x_1 + \ldots + a_{in}x_n = b_i \quad (i = 1, \ldots, m)$, $x_1 a_{i1} + \ldots + x_n a_{in} = b_i \quad (i = 1, \ldots, m)$ и $u_{i1} + \ldots + u_{in} = b_i$ $(i = 1, \ldots, m)$, где $u_{ij} \ (i \in \mathbf{N}_m; j \in \mathbf{N}_n)$ – выражение типа $a_{ij}x_j$, $x_j a_{ij}$ или $a'_{ij} x_j a''_{ij}$, причем, по крайней мере, два из этих типов присутствуют в терме;

4) $LA(\mathsf{K})$-решатель $\mathsf{S}^{(4)}_{LA(\mathsf{K})}$ предназначен для анализа выполнимости термов вида $a_{i1}x_1 + \ldots + a_{in}x_n \neq b_i \quad (i = 1, \ldots, m)$, $x_1 a_{i1} + \ldots + x_n a_{in} \neq b_i \quad (i = 1, \ldots, m)$ и $u_{i1} + \ldots + u_{in} \neq b_i$

$(i=1,\ldots,m)$, где u_{ij} $(i\in\mathbf{N}_m; j\in\mathbf{N}_n)$ – выражение типа $a_{ij}x_j$, x_ja_{ij} или $a'_{ij}x_ja''_{ij}$, причем, по крайней мере, два из этих типов присутствуют в терме.

Отметим, что $LA(\mathsf{K})$-решатель $\mathsf{S}^{(4)}_{LA(\mathsf{K})}$ заменяет анализируемый терм системой уравнений, которая выполнима тогда и только тогда, когда выполним исходный терм, после чего запускается $LA(\mathsf{K})$-решатель $\mathsf{S}^{(3)}_{LA(\mathsf{K})}$.

Из разбиения множества рассматриваемых колец на шесть непустых непересекающихся подмножеств вытекает, что истинна следующая теорема.

Теорема 1. Каждый $LA(\mathsf{K})$-решатель $\mathsf{S}^{(1)}_{LA(\mathsf{K})}$, $\mathsf{S}^{(2)}_{LA(\mathsf{K})}$, $\mathsf{S}^{(3)}_{LA(\mathsf{K})}$ и $\mathsf{S}^{(4)}_{LA(\mathsf{K})}$ является полным и непротиворечивым.

Исследуется асимптотическая сложность предложенных решателей. Доказано, что истинны следующие теоремы.

Теорема 2. Для любого поля $\mathrm{GF}(p^k)$ ($p\in\mathbf{N}$ – простое число, $k\in\mathbf{N}$) временная сложность $LA(\mathrm{GF}(p^k))$-решателя $\mathsf{S}^{(1)}_{LA(\mathrm{GF}(p^k))}$ равна

$$T=\begin{cases}O(\log p), \text{если } p\to\infty \text{ и } k \text{ фиксировано}\\ O(k), \text{если } k\to\infty \text{ и } p \text{ фиксировано}\\ O(k\log p), \text{если } p\to\infty \text{ и } k\to\infty\end{cases}. \qquad (1)$$

Теорема 3. Для любого кольца вычетов $\mathsf{Z}_{p^k}=(\mathbf{Z}_{p^k},+,\cdot)$ ($p\in\mathbf{N}$ – простое число, $k\geq 2$) временная сложность $LA(\mathsf{Z}_{p^k})$-решателя $\mathsf{S}^{(1)}_{LA(\mathsf{Z}_{p^k})}$ равна

$$T=\begin{cases}O(\log p), \text{если } p\to\infty \text{ и } k \text{ фиксировано}\\ O(\log k), \text{если } k\to\infty \text{ и } p \text{ фиксировано}\\ O(\log pk), \text{если } p\to\infty \text{ и } k\to\infty\end{cases}. \qquad (2)$$

Теорема 4. Для любого поля $\mathrm{GF}(p^k)$ ($p\in\mathbf{N}$ – простое число, $k\in\mathbf{N}$) временная сложность $LA(\mathrm{GF}(p^k))$-решателя $\mathsf{S}^{(2)}_{LA(\mathrm{GF}(p^k))}$ определяется формулой (1).

Теорема 5. Для любого кольца вычетов $\mathsf{Z}_{p^k} = (\mathbf{Z}_{p^k}, +, \cdot)$ ($p \in \mathbf{N}$ – простое число, $k \geq 2$) временная сложность $LA(\mathsf{Z}_{p^k})$-решателя $\mathsf{S}^{(2)}_{LA(\mathsf{Z}_{p^k})}$ определяется формулой (2).

Теорема 6. Для любого поля $\mathrm{GF}(p^k)$ ($p \in \mathbf{N}$ – простое число, $k \in \mathbf{N}$) временная сложность $LA(\mathrm{GF}(p^k))$-решателя $\mathsf{S}^{(2)}_{LA(\mathrm{GF}(p^k))}$ равна

$$T = (O(\min\{m, n\}(t_1 + mn^2 t_2)) \quad (p^k \to \infty).$$

Отметим, что для колец без единицы временная сложность каждого из $LA(\mathsf{K})$-решателей $\mathsf{S}^{(1)}_{LA(\mathsf{K})}$, $\mathsf{S}^{(2)}_{LA(\mathsf{K})}$, $\mathsf{S}^{(3)}_{LA(\mathsf{K})}$ и $\mathsf{S}^{(4)}_{LA(\mathsf{K})}$ может быть субэкспонентой и совпадать со сложностью поиска решений соответствующей системы уравнений или неравенств.

Таким образом, проверка выполнимости формулы линейной арифметики $LA(\mathsf{K})$, представленной конъюнкцией систем уравнений и неравенств над кольцом K, сводится к последовательной активации модулей $\mathsf{S}^{(1)}_{LA(\mathsf{K})}$, $\mathsf{S}^{(2)}_{LA(\mathsf{K})}$, $\mathsf{S}^{(3)}_{LA(\mathsf{K})}$ и $\mathsf{S}^{(4)}_{LA(\mathsf{K})}$.

Простейшие атомы типа равенство, выполнимость которых установлена модулем $\mathsf{S}^{(1)}_{LA(\mathsf{K})}$, могут быть использованы для подстановок в остальные уравнения и неравенства, что в ряде случаев дает возможность снизить сложность последующего анализа. Аналогичным образом, системы уравнений, выполнимость которых установлена модулем $\mathsf{S}^{(3)}_{LA(\mathsf{K})}$, могут быть использованы для подстановки в неравенства, которые будут анализироваться модулем $\mathsf{S}^{(4)}_{LA(\mathsf{K})}$. Такая подстановка дает возможность уменьшить число переменных, и, следовательно, упростить анализ системы неравенств.

Заключение

Полученные результаты показывают, что при проверке выполнимости формул линейной арифметики над конечным кольцом возникает ситуация, принципиально отличная от ситуации, которая имеет место для поля рациональных чисел или для кольца целых чисел.

Детальный анализ задачи проверки выполнимости формул нелинейной арифметики над конечными полями и кольцами вычетов

представляет собой возможное направление дальнейших исследований.

Другое направление связано с детальным анализом задачи выполнимости формул арифметики над простейшими некоммутативными кольцами. К таким кольцам, в частности, относятся кольца матриц над конечными полями.

Список использованной литературы

1. Sebastiani R. Lazy Satisfiability modulo theories / R. Sebastiani // Journal on Satisfiability, Boolean Modeling and Computation. – 2007. N 3. – P. 141-224.
2. Курош А.Г. Лекции по общей алгебре / А.Г. Курош. – М.: Наука, 1973. – 400 с.
3. Skobelev V.V. On systems of polynomial equations over finite rings / V.V. Skobelev // Наукові записки НаУКМА. – 2012. – Т. 138. – P. 15-19.

ПРОБЛЕМА ПРОВЕРКИ ВЫПОЛНИМОСТИ ФОРМУЛ РАЗРЕШИМЫХ ТЕОРИЙ (ОБЗОР)

В. В. Скобелев[1], *В.Г. Скобелев*[2]

Институт прикладной математики и механики НАН Украины, Донецк, Украина

[1]*vv_skobelev@iamm.ac.donetsk.ua*
[2]*skbv@iamm.ac.donetsk.ua*

Применение информационных технологий практически во всех сферах деятельности человечества привело к необходимости автоматизации процессов управления, проектирования, анализа безопасности и эффективности, а также сопровождения реальных (в том числе, информационных) систем с критической областью применения.

Широкий класс реальных задач (организация потоков информации в компьютерных сетях, формальная верификация систем, представление и обработка знаний, планирование, исследование операций, тестирование дискретных устройств и т.д.), а также задач дискретной математики (основанных на использовании теоретико-множественных, теоретико-числовых, логических, графовых и сетевых моделей) имеет следующее общее свойство: решение задачи естественно сводится к проверке выполнимости некоторой формулы.

Именно это свойство и лежит в основе разработки средств автоматизированного решения (т.е. решателей) всех таких задач.

В математической логике «выполнимость формулы» означает существование хотя бы одной интерпретации, в которой эта формула истинна.

Проверку выполнимости формулы F исчисления высказываний можно осуществить с помощью двоичного размеченного ориентированного ранжированного дерева D_F, построенного в соответствии со следующими правилами:

1. Корень дерева D_F отмечен формулой F.

2. Осуществляется выбор висячей вершины v дерева D_F и переменной x, входящей в формулу F_v, являющуюся отметкой вершины v.

3. Из вершины v выходят две дуги, ведущие в вершины v_l и v_r следующего уровня. Дуга, ведущая в вершину v_l (соответственно, в вершину v_r), имеет отметку «$x=1$» (соответственно, «$x=0$»). Отметка вершины v_l (соответственно, v_r – формула, полученная из формулы F_v подстановкой $x=1$ (соответственно, подстановкой $x=0$), упрощенная на основе тождеств исчисления высказываний.

Построение дерева D_F осуществляется до тех пор, пока либо появится вершина с отметкой «1», либо каждая висячая вершина имеет отметку «0». В 1-м случае формула F – ыполнимая, а во 2-м случае – невыполнимая (т.е. тождественно ложная во всех интерпретациях).

Приведенная выше схема представляет класс рекурсивных алгоритмов, каждый из которых определяется правилами выбора раскрываемой вершины v, а также переменной x, входящей в формулу F_v.

Известно, что проверка выполнимости формул исчисления высказываний является NP-полной задачей. Значительные усилия исследователей были затрачены на разработку методов, позволяющих упростить решение этой задачи на практике.

Один из классов таких методов основан на применении правила резолюции: следствием дизъюнктов $A \vee B$ и $\overline{A} \vee C$ является дизъюнкт $B \vee C$ (этот дизъюнкт называется резольвентой).

Представление исследуемой формулы исчисления высказываний в виде КНФ и применение правила резолюции лежит в основе DPLL-процедуры (DPLL – сокращение от Davis-Putnam-Logemann-Loveland).

Для исчисления предикатов ситуация иная. Известно, что не существует алгоритм, осуществляющий проверку выполнимости формул исчисления предикатов. Однако существуют алгоритмы, которые для любой выполнимой формулы исчисления предикатов за конечное число шагов устанавливает ее выполнимость. Идея построения таких алгоритмов состоит в следующем.

Замкнутая (т.е. не содержащая свободных переменных) формула F_1 преобразуется в эквивалентную формулу F_2, не содержащую логические связки, отличные от связок $\wedge$, $\vee$ и $\bar{}$. Формула F_2 приводится к предваренной нормальной форме

$$F_4 = (Q_1 x_1)\ldots(Q_n x_n)F_3,$$

где $Q_1,\ldots,Q_n$ – кванторы, а F_3 – бескванторная формула. Элиминацией кванторов существования формула F_4 приводится к сколемовской нормальной форме

$$F_6 = (\forall y_1)\ldots(\forall y_m)F_5,$$

где F_5 – бескванторная формула.

Из теоремы Эрбрана вытекает, что невыполнимость формулы F_6 (т.е. используется метод доказательства «от противного») эквивалентна существованию невыполнимого конечного подмножества множества формул, полученных подстановкой в формулу F_5 всевозможных термов, которые можно построить при помощи предметных констант и функциональных символов, встречающихся в формуле F_5.

Таким образом, проверка невыполнимости формулы F_6, по своей сути, сводится к проверке того, что конечное множество формул исчисления высказываний является невыполнимым.

Если исследуемая задача сводится к проверке выполнимости формулы математической логики (т.е. исчисления высказываний или исчисления предикатов), то она может быть решена непосредственным применением SAT-решателя, т.е. той или иной программной системы, предназначенной для проверки выполнимости формул математической логики.

Если же задача может быть адекватно представлены только в виде такой формулы 1-го порядка некоторой разрешимой теории T, что для ее решения требуется применение специфических методов теории T, то говорят о выполнимости по модулю теории T. В этом случае используется обозначение SMT(T), где SMT – сокращение фразы «Satisfiability Modulo Theory».

Для решения таких задач применяется T-решатель, представляющий собой ту или иную программную систему,

состоящую из взаимодействующие между собой модулей, осуществляющих вычисления в рамках теории T.

Целью настоящего доклада и является анализ современного состояния исследований проблемы проверки SMT(T).

Вначале в докладе рассматриваются методы, используемые при построении современных SAT-решателей, основанных на использовании DPLL-процедуры.

Затем рассматриваются основные понятия и принципы, лежащие в основе построения современных программных средств, предназначенных для проверки SMT(T)для различных теорий T.

Охарактеризован «ленивый» подход (lazy approach) к решению этой проблемы.

В настоящее время этот подход является основным при построении программных средств, предназначенных для проверки SMT(T) для различных теорий T.

«Ленивый» подход основан на организации эффективного взаимодействия SAT-решателя и T-решателя, т.е. процедур, предназначенных для анализа ограничений, сформулированных в терминах атомов теории T.

В докладе охарактеризована общая структура T-решателя, построенного на основе «ленивого» подхода.

Охарактеризованы методы, предназначенные для повышения эффективности вычислений, осуществляемых современными T-решателями.

Детализация структуры T-решателя проиллюстрирована рядом примеров.

В докладе представлен список литературы, в которой изложены различные подходы и методы, используемые при построении современных программных средств, предназначенных для проверки SMT(T) для различных теорий T.

В заключение кратко перечислены некоторые новые направления, формирование которых происходит на современном уровне исследований проблемы проверки SMT(T).

МЕХАНІЗМИ РОЗРОБЛЕННЯ СІМЕЙСТВА ПРОГРАМНИХ ПРОДУКТІВ ЗА ЙОГО ОБ'ЄКТНО-КОМПОНЕНТНОЮ МОДЕЛЛЮ

О.О. Слабоспицька

Київський національний університет імені Тараса Шевченка, Київ, Україна

ols.07@mail.ru

Вступ

Перспективною формою індустріального виробництва програмних продуктів (ПП) масового використання з обмеженими витратами й термінами та гарантованою якістю є наразі Сімейство ПП (СПП). Це набір ПП з керованою множиною спільних властивостей (features), відповідних потребам споживачів у цільовому домені, які розробляються попередньо визначеним способом із спільної множини ресурсів повторного використання [1].

У роботі надано формальний розвиток моделі властивостей СПП у парадигмі збіркового програмування [2] для вдосконалення уніфікованої автоматизованої підтримки розроблення й рефакторингу ПП та еволюції СПП за передбачених і, насамперед, непередчених змін середовища його розроблення й використання.

Об'єктно-компонентна модель СПП

Традиційна для СПП модель властивостей (МВ) – ієрархія спільних та змінних властивостей ПП з СПП із спеціальними відношеннями [1],[3] між ними. Властивість – вимога, функція чи нефункціональна характеристика ПП, запитана їх споживачами.

Пропонується адаптація об'єктно-компонентного методу моделювання ПС К.М.Лавріщевої-В.М.Грищенка [2] до МВ з властивостями двох типів – функціями та даними для них. Результатом є

Означення 1. Об'єктно-компонентна модель СПП – це п'ятірка

$$OCM = \langle r; OA; IA; CA; B \rangle; OA = \langle (F, R_F), (D, R_D), OO \rangle, IA = \langle I, R_I \rangle, \quad (1)$$
$$CA = \langle C, R_C, CO \rangle; R_S = \{CH_S, VC_S, IM_S, EQ_S, EX_S\}, S \in \{F, D, I, C\},$$

де r – ім'я СПП; OA, IA, CA – відповідно об'єктна, інтерфейсна й компонентна підмоделі СПП; B – апріорна умова їх коректності.

Ім'я r – спільний корінь зв'язних орієнтованих графів з (1): об'єктів домену СПП, що подають усі передбачені для ПП функції ((F, R_F)) і дані ((D, R_D)); інтерфейсів підтримки їх взаємодії (I, R_I); програмних компонентів повторного використання (КПВ), що реалізують інтерфейси (C, R_C). Дуги цих графів – пари вершин

множини S: об'єкти функцій $f \in F$ і даних $d \in D$, інтерфейси $in \in I$ та КПВ $c \in C$, пов'язані властивими МВ [3] відношеннями $E \in R_S$ з (1), поданими далі. В об'єктній (OA) і компонентній (CA) підмоделях OO й CO – операції добору та, відповідно, реа-лізації функцій і даних для компонентних ПП з СПП. В інтерфейс-ній підмоделі IA виконання методів функцій $f \in F$ на об'єктах да-них $d(f) \in D$, припустимих чи обов'язкових для f, подано як під-тримка в інтерфейсі in реалізації допоміжних складних об'єктів o:

$$o = (f, d(f)) \in O \subseteq F \times D; I = \left\{ in = \mathrm{e}(o) = \left\langle id, M(f), D(d(f)) \right\rangle \right\}, \quad (2)$$

де id – унікальне ім'я in, $M(f)$ – сигнатура методів f; $D(d(f))$ – перелік атрибутів $d(f)$, що зіставляються змінним з $M(f)$.

Умова коректності OCM (елемент B в (1)) – це відсутність в елементарних функцій і об'єктів даних (листків графів (F, R_F) і (D, R_D)) спільних методів та, відповідно, атрибутів.

Множину R_S відношень на $S \in \{F, D, O, I, C\}$ з (1) описує

Означення 2. Об'єкти функцій $f_i \in F, f_j \in FG \subset F$ пов'язані:

1) прямим (CH_F) або (m,n)-варіантним підпорядкуванням ($VC_F^{m,n}, 0 \le m < n \lhd |F|$), якщо реалізація в ПП "батька" f_i випли-ває з реалізації всіх чи, відповідно, від m до n з FG "нащадків" f_j;

2) імплікацією IM_F або зіставленням EQ_F, якщо реалізація f_i потребує f_j або, відповідно, вони мають реалізуватися разом;

3) виключенням EX_F, якщо реалізація f_i виключає f_j.

Відношення з множини R_D мають той же зміст, що й відно-шення 1)-3) з R_F, але $VC_F^{m,n}$ відповідає $VC_D^{p,q}, 0 \le p < q \lhd |D|)$.

Щодо множини R_O, складні об'єкти $o_i, o_j \in O$ (2) пов'язані прямим підпорядкуванням, імплікацією, зіставленням і виключен-ням, якщо їх відповідники CH_F, IM_F, EQ_F, EX_F пов'язують функ-ції f_i, f_j або ж вони однакові/непов'язані, а відношення CH_D, IM_D, EQ_D, EX_D має місце для $d(f_i), d(f_j)$. За (m,n,p,q)-варіантно-го підпорядкування, $VC_F^{m,n}$ пов'язує f_i, f_j ($p, q = 0$) або ж це не так в описаному сенсі, і $VC_D^{p,q}$ пов'язує $d(f_i), d(f_j)$ ($m, n = 0$).

Відношення з множин R_I й R_C мають місце для інтерфейсів і реалізуючих КПВ, якщо їх однойменні, щойно визначені, відповідники з множини R_O пов'язують підтримувані об'єкти o_i, o_j.

За варіантного підпорядкування на S (див.(1)), "батько" зветься точкою варіабельності (у функціях, даних, складних об'єктах, інтерфейсах, КПВ), а "нащадки" – варіантами його реалізації у ПП.

Відношення R_I, R_C дозволяють опис КПВ $c \in C$ з CA в (1).

Означення 3. Інтерфейс $in \in I$ для об'єкта o (2) реалізується КПВ з єдиним вхідним інтерфейсом $II.c$ – цим in без імені id, доповне-ним, якщо це точка варіабельності, сигнатурою mch сервісного ме-тоду вибору варіантів in, та множиною $IO.c$ вихідних інтерфейсів:

$$c = \kappa(in) = \langle id, II.c, S.c \cup [ch.c], IO.c \rangle; II.c = \langle M(f) \cup [mch], D(d(f)) \rangle,$$
$$IO.c = \cup_{0 \le m < n \le |F|, 0 \le p < q \le |D|} \{ir \in I \mid inVC_I^{m.n.p,q} ir, inCH_I ir\}), \qquad (3)$$

де id – унікальне ім'я c, успадковане від in; $S.c$, $ch.c$ – програмні реалізації методів $M(f)$ для атрибутів $D(d(f))$ та для методу mch.

Трактуванням виразів (2),(3) з позицій загальної алгебри [4] є

Лема. Композиція відображень (2),(3) $\tau = \kappa \circ \varepsilon$ – ізоморфізм алгеб- раїчних моделей типу 2 $\langle O, R_o \rangle, \langle I, R_i \rangle, \langle C, R_C \rangle$, що зіставляє точ-ки варіабельності й варіанти складних об'єктів, інтерфейсів і КПВ.

Наслідок. Листку $l = (fl, d(fl))$ графу (O, R_O), що поєднує елемен-тарну функцію з її даними, відповідають листки графів IA, CA (1): елементарний інтерфейс $il = \langle id, M(fl), D(d(fl)) \rangle$ і КПВ $cl = \kappa(il)$: $II.cl = IO.cl = il \setminus \langle id \rangle; S.cl_i \cap S.cl_j = \varnothing$ (за умовою B з (1)); $S.cl$ – модулі для предметно-орієнтованих методів $M(fl)$ на $D(d(fl))$.

На підставі леми формалізуємо передбачені властивості ПП.

Означення 4. У СПП із моделлю (1) конфігурація властивостей ПП – зв'язний підграф g графу (O, R_O) (2) з коренем r і вершинами $SO \subseteq O$, де зберігаються їх бінарні відношення з R_O, крім виклю-чення. Предикат конфігурації – функція $P: 2^O \to \{0;1\}$, одинична для підмножин-конфігурацій і нульова – для решти. Компонентна конфігурація – подібний до g підграф q графу (C, R_c)

Зауваження. Образ конфігурації g $\tau(g) = \{\tau(o), o \in SO\}$ – компо-нентна конфігурація; P індукує на C предикат PC:

$PC(SC)=1$, якщо $\tau^{-1}(SC)=\{\tau^{-1}(c), c\in SC\}$ – конфігурація; інакше $PC(SC)=0$.

Запровадимо операції OO з OA, CO з CA (див. (1)).

Означення 5. Нехай $OO=\{\cup,\cap,\neg\}; CO=\{\cup^c,\cap^c,\neg^c\}$. Результат об'єднання $\cup$, перетину $\cap$ й доповнення $\neg$ множин властивостей (відповідно, однойменних операцій $\cup^c,\cap^c,\neg^c$ на 2^C) – конфігурація властивостей (відповідно, КПВ), де множина вершин утворена об'єднанням, перетином і доповненням операндів, що є конфігураціями, а предикат P (відповідно, PC) одиничний, або $\varnothing$, якщо P (PC) нульовий і/або хоч один з операндів – не конфігурація.

З означення, леми й результатів у галузі алгебри [4] випливає

Теорема. Алгебрачні системи $\left\langle 2^O, OO, \varnothing, O\right\rangle, \left\langle 2^C, CO, \varnothing, C\right\rangle$ – бу-льові алгебри типу $\langle 2,2,1\rangle$ з нулями $\varnothing$ і одиницями O, C, пов'язані ізоморфізмом $\tau=\kappa\circ\varepsilon$ вигляду (2),(3).

Використання об'єктно-компонентної моделі СПП

Розглянута модель OCM (1) є підґрунтям уніфікованого розвитку й автоматизованої підтримки механізмів розроблення СПП:

1) ітеративного узгодженого моделювання на підставі коду наявних ПП [3] та очікуваних потреб їх споживачів і розробників;

2) побудови ПП та їх варіантів у межах СПП і поза ними [5];

3) обґрунтованого моніторингу [6] відповідності зазначеним потребам варіабельності СПП – "здатності ПП або артефакта ефективно розвиватися, змінюватися, налаштовуватися або конфігуруватися для використання в певному контексті " [1],[5],[6];

4) еволюції СПП за непередбачених змін внутрішнього й зовнішнього середовища його розроблення й використання [6].

Реалізація механізмів сприятиме підвищенню якості ПП з СПП завдяки можливості гарантування якості КПВ (ретельним тестуванням, добором сертифікованих COTS) та автоматизованого формування і збирання в сучасних компонентних середовищах (MS.Net, J2EE тощо) їх конфігурації, адекватної потребам у ПП.

Список використаних джерел

1. Pohl K., Bockle G., van der Linden F.J. Software Product Line Engineering: Foundations, Principles and Techniques. – Springer-Verlag, 2005. – 473 p.

2. Лаврищева Е.М., Грищенко В.Н. Сборочное программирование. Основы индустрии программных продуктов // 2-изд. доп. и перераб. – Киев: Наук. думка, 2009. – 372 с.
3. She S. et al. Reverse Engineering Feature Models // ICSE ’11, May 21–28, 2011, Waikiki, Honolulu, HI, USA.
4. Артамонов В.А., Салий В.Н., Скорняков Л.А. Общая алгебра.Т.1,2 – М., Наука, 1991. – 348 с.
5. Лавріщева К.М та ін. Теоретичні аспекти керування варіабельністю в сімействах програмних систем // Вісн. КНУ ім. Т. Шевченка. Сер. фіз.-мат. науки. – 2011. – № 1 – С. 45-53.
6. Слабоспицька О.О. Підхід до підтримки обгрунтованої еволюції сімейства програмних систем // Праці 9-ї міжнар. конф. TAAPSD’2012 – Київ, 2012. – С. 274-278.

ПРОГРАММНЫЙ КОМПЛЕКС ДЛЯ МОДЕЛИРОВАНИЯ И ОПТИМИЗАЦИИ ПОТОКА ГРУНТОВЫХ ВОД

О.Б. Стеля, И.О. Стеля, В.И. Кудин

Киевский национальный университет имени Тараса Шевченко, Киев, Украина

oleg.stelya@gmail.com, igor.stelia@gmail.com

Особенностью данного программно-моделирующего комплекса KRISFLOW является его направленность на решение плановых задач потока грунтовых вод для больших территорий со сложными гидрогеологическими условиями, требующими введения значительных объемов информации, представленной в картографическом и табличном виде. Для удобного введения исходной информации разработан графический пользовательский интерфейс.

Положенная в основу моделирующего комплекса математическая модель должна учитывать всю сложность питания водоносных горизонтов, источниками которого являются потери на фильтрацию из каналов различного назначения, питание грунтовых вод за счет орошения, осадки, отбор грунтовых вод дренажными скважинами, возможный переток между водоносными горизонтами. Также должны учитываться такие объекты, как расположенные внутри области водоемы и различные условия на границе области.

Используемая математическая модель основана на системе уравнений, которые описывают нестационарный плановый поток грунтовых вод в связанных между собой водоносных горизонтах:

$$S_1(x)\frac{\partial u_1(x,t)}{\partial t} = \sum_{i=1}^{2}\frac{\partial}{\partial x_i}\left(k_1(x)b_1(x,u_1(x,t))\frac{\partial u_1(x,t)}{\partial x_i}\right) -$$

$$-\sum_{m=1}^{p} q_m(t)\delta(x-r_m) + \sigma(x)(u_2(x,t)-u_1(x,t)) + W(x,t),$$

$$x=(x_1,x_2)\in \Omega_1 \setminus \sum_{l=1}^{n_1}\overline{\Xi}_l, \qquad 0<t\le T,$$

$$S_2(x)\frac{\partial u_2(x,t)}{\partial t} = \sum_{i=1}^{2}\frac{\partial}{\partial x_i}\left(k_2(x)b_2(x)\frac{\partial u_2(x,t)}{\partial x_i}\right) +$$

$$+\sigma(x)(u_1(x,t)-u_2(x,t)), \quad x=(x_1,x_2)\in \Omega_2, \quad 0<t\le T,$$

где $u_1(x,t)$ – гидравлический напор безнапорного горизонта, $k_1(x)$ – коэффициент фильтрации безнапорного горизонта, $S_1(x)$ – коэффициент влагоемкости, $b_1(x,u_1(x,t))$ – толщина насыщенной части пласта, q_m – объем води, который отбирается m-той дренажной скважиной ($0 \le q_m(t) \le Q$), $\delta(x-r_m)$ – дельта-функция Дирака в точке r_m, точки $x = r_m$, $m = \overline{1,p}$ определяют положение дренажных скважин в области, $W(x,t)$ – определяет суммарное поступление влаги на свободную поверхность грунтовых вод, $\sigma(x) = k_0(x)/b_0(x)$ ($k_0(x)$ – коэффициент фильтрации слабопроницаемого разделяющего слоя, $b_0(x)$ – его толщина), $u_2(x,t)$ – гидравлический напор напорного горизонта, $S_2(x)$ – коэффициент влагоемкости, $k_2(x)$ – коэффициент фильтрации напорного горизонта, $b_2(x)$ – мощность водоносного горизонта.

На границе области Ω_1 задаются условия $u_1(x,t) = g(x,t)$, $k_1(x)b_1(x,u_1(x,t))\dfrac{\partial u_1(x,t)}{\partial \mathrm{n}} = R(x,t,u_1)$, где n – вектор внешней нормали к границе области. Аналогичные условия задаются на границе области Ω_2. Задаются следующие начальные условия

$$u_1(x,t) = \phi_1(x,t),\ x \in \overline{\Omega}_1,\ u_2(x,t) = \phi_2(x,t),\ x \in \overline{\Omega}_2,\ t = 0.$$

Внутренние поверхностные водоемы, содержащиеся в области моделирования, обозначены Ξ_l, а их границы - $\partial\,\Xi_l$, $l = \overline{1,n_1}$. На этих границах задаются условия $u_1(x,t) = g_l(t),\ x \in \partial\,\Xi_l,\ l = \overline{1,n_1}$, где $g_l(t)$ соответствует отметке поверхности воды в l-м водоеме.

Для решения краевой задачи используется метод конечных разностей.

Программный комплекс предназначен для решения плановых задач со значительными объемами вводимой информации. Введение такой информации требует согласования исходных данных различного характера, представленных в табличном и картографическом виде. Данный программный комплекс ориентирован на использование на больших ирригационных системах для оптимизации их функционирования. В связи с этим,

модель требует введения данных по распределению орошаемых площадей с нормами полива на них, данных о расположении каналов различного назначения с объемами потерь на фильтрацию из них, о расположении дренажных скважин с фактическими объемами откачки и др.

Характер и структура информации необходимой для функционирования модели определили архитектуру программного комплекса, состоящего из трех блоков:

- блока, реализующего численные методы решения краевых задач для моделирования потока грунтовых вод;
- графического пользовательского интерфейса для введения исходных данных и их хранения в файлах специальной структуры;
- программного блока для связи с географической информационной системой (ГИС).

Программные средства графического интерфейса позволяют настраивать модель на исследуемый объект. Работа с графическим интерфейсом является интуитивно понятной и описана в руководстве пользователя. С помощью экранных форм вводимые данные структурируются и записываются в файлы для дальнейшего использования блоком решения краевых задач и ГИС для их отображения на карте. Данные, хранимые в файлах, непосредственно привязываются к дискретной модели (соответствующей расчетной разностной сетке) на цифровой карте объекта. Пример экранной формы приведен на рисунке 1.

Форма предназначена для задания расположения орошаемых полей их административной принадлежности и информации по объемам полива. Поля задаются номерами узлов сетки, в которых осуществляется полив. Номера узлов могут задаваться как вручную, так и в автоматизированном режиме с помощью нажатия стрелок в направлении смещения на карте в нужный узел дискретной модели.

После ввода с помощью экранной формы и записи данных в файл они могут автоматически отображаться на цифровой карте в ГИС. Таким образом, ГИС используется, как на начальном этапе для привязки модели к исследуемой области, введения координатно-привязанных данных в модель и их контроля, так и на этапе отображения результатов моделирования и автоматизированного построения тематических карт по результатам расчетов.

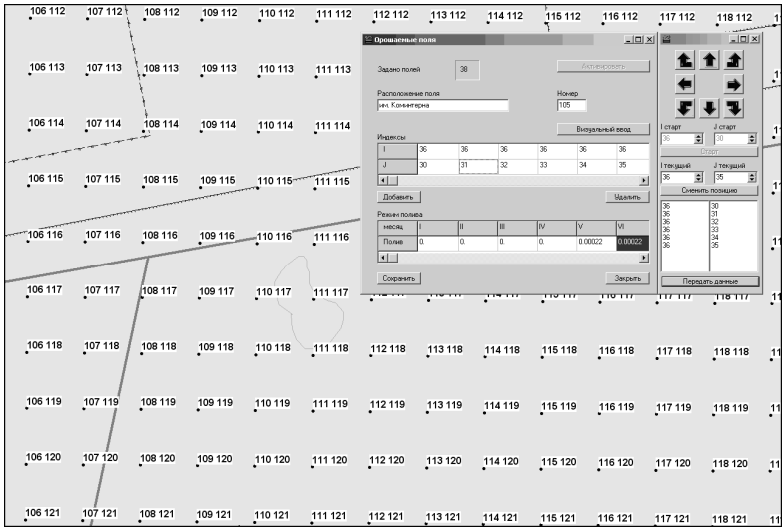


Рис. 1. Цифровая карта с экранной формой для ввода исходных данных

Данные для динамического построения карт хранятся в файлах специальной структуры формируемых с помощью пользовательского интерфейса. В ГИС в автоматизированном режиме строятся такие слои цифровой карты, как узлы разностной сетки, граница области моделирования, дренажные скважины, наблюдательные скважины системы мониторинга, оросительные каналы, орошаемые поля, поверхностные водоемы и др.

Программный комплекс позволяет в автоматическом режиме отображать результаты моделирования на цифровой карте в виде абсолютных отметок поверхности грунтовых вод или в виде тематических карт глубин залегания грунтовых вод, данные мониторинга, а также графическую и атрибутивную информацию, используемую в модели. Такие карты удобны с точки зрения анализа ситуации на объекте моделирования и принятия решений.

В качестве примера использования программного комплекса рассмотрена область, расположенная в Херсонской области Украины на территории Краснознаменской оросительной системы. Для моделирования выбрана западная ее часть общей площадью около 460 квадратных километров. Такой выбор обоснован тем, что применяемая на ней сельскохозяйственная и ирригационная практика характерна для юга Украины, а область моделирования имеет хорошо выраженные гидрогеологические границы представленные Краснознаменским каналом (с севера и северо-востока) и берегом

Черного моря (с юга). С другой стороны изучаемая область имеет слабую естественную дренированность земель и требует постоянных мероприятий по искусственному водопонижению. Целью исследования являлось моделирование уровня грунтовых вод в зависимости от распределения орошения по полям и работы дренажной системы. Результаты моделирования использовались для оценки эффективности существующего дренажа, оценки целесообразности введения в эксплуатацию дополнительных дренажных скважин, анализа развития ситуации в долгосрочном прогнозе.

Для ввода исходных данных в модель использовались такие слои цифровой карты как: дренажные скважины, орошаемые поля, поверхностные водоемы, каналы различного назначения с делением по типу покрытия, а также скважины системы мониторинга.

С картографическими объектами связаны файлы с различной атрибутивной информацией. Так, например, для дренажных скважин в файле содержится для каждой скважины: номер скважины, наименование населенного пункта, в котором она расположена, индексы узла разностной сетки, с которым она совпадает, ежесуточный дебит. Для визуализации результатов моделирования и принятия решения использовались слои карты с населенными пунктами, коммуникациями, береговой линией Черного моря, границей области моделирования.

При численном решении задачи применялась равномерная сетка с величиной ячейки 100х100 метров, что привело к 46525 расчетным узлам. Размеры ячейки сетки выбирались исходя из точности используемых для моделирования данных. Слой карты с узлами разностной сетки строился динамически и использовался для введения исходных данных и отображения результатов моделирования в цифровом виде. При расчетах использовался временной шаг 0.5 суток. На вертикальных участках, ограничивающих область с запада и востока, заданы однородные условия второго рода, на участках границы вдоль магистрального канала и берега моря – условия первого рода, интерполированные по данным наблюдений за уровнем грунтовых вод. Были заданы: пространственное и временное распределение орошения по хозяйствам, осадки, расположение каналов и объемы потерь из них, а также объемы и режим откачки дренажными скважинами. Калибровка модели осуществлялась на протяжении трех лет. В процессе калибровки с помощью математического моделирования влагопереноса были уточнены объемы питания грунтовых вод за счет орошения, а также потери на фильтрацию из каналов.

Полученный модельным путем уровень грунтовых вод сравнивался с измеренным в наблюдательных скважинах системы

мониторинга. Время моделирования составляло один год, при этом рассчитанные уровни грунтовых вод использовались в качестве исходных данных при моделировании на следующий год.

Тематическая цифровая карта с областью моделирования и, построенными в автоматическом режиме, глубинами залегания грунтовых вод приведена на рисунке 2. Прогнозные расчеты позволили оценить эффективность работы вертикального дренажа при изменении объемов и площадей орошения.

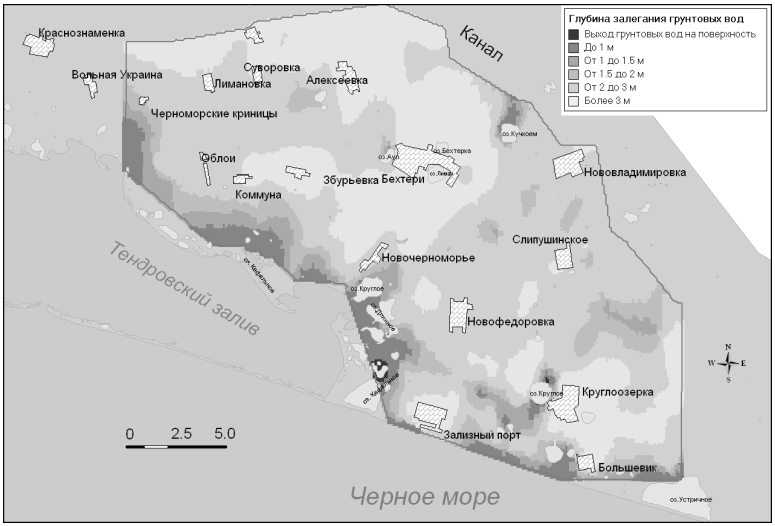


Рис. 2. Результаты моделирования глубин залегания грунтовых вод

ФОРМАЬНИЙ ОПИС ВЗАЄМОДІЇ ФУНКЦІОНАЛЬНИХ І ІНТЕРФЕЙСНИХ ОБ'ЄКТІВ В РОЗПОДІЛЕНИХ ПРОГРАМНИХ СИСТЕМАХ

А.Ю. Стеняшин

Інститут програмних систем НАН України, 03187, Київ, проспект Акад.Глушкова, 40,

e–mail: stenyaschin@gmail.com

Вступ

В роботі пропонується використання об'єктне-орієнтованого підходу (ООП) до опису взаємодії функціональних та інтерфейсних об'єктів при проектуванні розподілених програмних систем (РПС).

Об'єктне проектування РПС базується на концепції декомпозиції об'єктів, як методів чи функцій предметної області, що можуть бути розробленими і розповсюдженими по різних вузлах інформаційного середовища [1-3]. Головною проблемою проектування є забезпечення взаємозв'язку функціональних об'єктів. Зараз використовуються широко розповсюджене поняття інтерфейсу, як проміжного не функціонального об'єкту, який призначений для забезпечення інформаційного зв'язку різних функціональних об'єктів предметної області. Прикладом такого інтерфейсу є stub і skeleton, що реалізовані в системі CORBA і забезпечують зв'язок об'єктів клієнта і серверу між собою..

Це реалізовано в межах об'єктної моделі (ОМ) в системі CORBA, яка має повну і чітку об'єктну орієнтацію, призначену для визначення ОМ із функціональних об'єктів із забезпеченням їх взаємодії в межах РПС при виконанні цих обєктів в сучасних гетерогених середовищах.

Модель ОМ є еталонною і використовується при проектуванні складних систем, як стандарт об'єктної структури РПС. Об'єкти ОМ визначають властивості функцій та завдають типи даних при опису інтерфейсних об'єктів, що забезпечують взаємодію обєктів у сучасному операційному середовищі. Об'єкти групуються в класи відповідно властивостей їх функцій і можуть створювати супер класи чи підкласи об'єктів. Над об'єктами класу виконуються операції ООП - наслідування, полімоффізму, агрегації тощо Для опису інтерфейсних об'єктів РПС пропонується спеціальна мова опису вхідних і вихідних параметрів, що передаються між різнорідними об'єктами.

Засоби опису взаємодії об'єктів РПС

В рамках наукового проекту в ІПС НАНУ розроблений об'єктна-компонентний підхід, який орієнтований на проектування РПС з компонентів повторного використання (КПВ) і сервісів. Цей підхід подається на чотирьох рівнях. Перший з них відповідає побудові ПС із об'єктів. В даної роботі розглядаються об'єкти двох типів.

Під *об'єктами 1-го типу* будимо розуміти об'єкти або компоненти, які відповідають прикладним функціям предметної області (ПрО) і забезпечують рішення деяких задач ПрО. До *об'єктів 2-го типу* відносимо об'єкти-інтерфейси, що включають операції виклику методів об'єктів 1-го типу і грають роль посередників для забезпечення взаємозв'язку між функціональними об'єктами 1-го типу, які можуть бути віддаленими один від одного по різним середовищам.

Інтефейсні об'єкти виконують зв'язок об'єктів 1-типу з сервісними функціями - віддаленими процедурами мережі, розташованими у серверній частині РПС. Його загальними функціями є:: підготовка зовнішніх даних (параметрів) клієнта, набір викликів цих процедур або сервісу до сервера, обробка різних помилок, повернення даних від сервера до клієнта тощо.

Операції взаємодії за допомогою інтерфейсів визначаються викликами і протоколами. Самі об'єкти описуються мовами програмування (МП), а їх інтерфейси в мові IDL.

Структура об'єкту 2-типу, чи посереднику незалежна від МП, вона визначена в IDL. В ній дається опис вхідних даних - in, вихідних – out, і сумісних - inout. Це мова стандартизує структуру посередника, опис параметрів та дає можливість перебудовувати не однакові по типу і структурі, дані, що передаються по мережі.

РПС складається із двох частин, кожна з яких може виконуватися у різних процесах, а взаємозв'язок між ними виконують інтерфейсні функції. Перша частина РПС — серверна програма, а друга — клієнтська.

Мовні засоби введемо на прикладі проектування програми з використанням базової множини мови. При цьому будемо розглядати два типи об'єктів-компонентів: прикладні, які виконують функції клієнта ПрО; інтефейсні, які виконують зв'язок об'єктів клієнта з сервісними функціями - віддаленими процедурами мережі, розташованими у серверній частині РПС [36].

В загальні функції інтерфейсного посередника клієнта входять: підготовка зовнішніх даних клієнта (параметрів), набір. викликів цих процедур або сервісу до сервера, обробка різних помилок, повернення даних від сервера до клієнта.

Загальні функції інтерфейсного посередника сервера містять: очікування повідомлень клієнта та їх обробка; запуск віддаленої

процедури та передача їй параметрів клієнта; повернення результатів процедури клієнту, знищення віддаленої процедури та інші.

Структура посередника незалежно від мови у цілому однакова. Це дає змогу стандартизувати структуру посередника та визначити Її за допомогою однієї з вказаних мов. РПС складається із двох частин, кожна з яких виконується у різних процесах, а взаємодіють вони шляхом виклику інтерфейсних функцій. Перша частина — серверна програма, а друга — клієнтська (далі просто сервер та клієнт).

Формальні операції над об'єктами двох типів

Введемо формальне визначення РПС у термінах процесів–об'єктів та за допомогою запропонованої мови специфікацій об'єктів. Будемо вважати такі визначення:

F	Множина функцій
I	Множина інтерфейсів
O	Множина об'єктів-процесів
$O_k \in O, In(O_k)$	Множина вхідних інтерфейсів (які об'єкт використовує як клієнт)
$O_k \in O, Out(O_k)$	Множина вихідних інтерфейсів (які об'єкт надає як сервер)

Визначення взаємодій об'єкту (взаємодія першого типу). Результатом взаємодії двох об'єктів буде об'єкт, у якого множина вхідних інтерфейсів співпадає з множиною вхідних інтерфейсів об'єкта-сервера, а множина вихідних інтерфейсів — з множиною вихідних інтерфейсів об'єкта-клієнта:

$$O_k = (Out(O_k), In(O_k))$$
$$O_l = (Out(O_l), In(O_l))$$
$$O_k \cdot O_l = (Out(O_k), In(O_l))$$

Не всі об'єкти можуть взаємодіяти один з одним, тому накладемо певні умови. Взаємодія об'єктів $O_k \cdot O_l$ є коректною, якщо об'єкт-сервер повністю забезпечує сервіс, необхідний об'єкту-клієнту $\forall I_m \in In(O_k) \Rightarrow \exists I_n \in Out(O_l) \wedge I_m = I_n$

Інтерфейс буде вхідним для об'єкту, якщо об'єкт отримує сервіс за допомогою цього інтерфейсу. І вихідним, якщо за допомогою цього інтерфейсу об'єкт надає сервіс об'єктам.

Визначення проекції об'єкту. Результатом проекції об'єкту на інтерфейс буде об'єкт, у якого множина вхідних інтерфейсів містить один елемент – цей інтерфейс, а множина вихідних інтерфейсів містить тільки ті інтерфейси, які необхідні для надання сервісу цього інтерфейсу. $O_k = \left(Out\left(O_k\right), In\left(O_k\right)\right)$

$$O_k\left[I_m\right] = if\left(I_m \in Out\left(O_k\right)\right) then\left(\left\{I_m\right\}, \left\{I_n : I_n \in In\left(O_k\right) \wedge exec\left(I_m, I_n\right)\right\}\right) else\left(\{\ \}, \{\ \}\right)$$

де $exec\left(I_m, I_n\right)$ - предикат, що вказує на необхідність інтерфейсу I_n для виконання I_m інтерфейсу.

Визначимо проекцію об'єкту на множину інтерфейсів:

$$O_k\left[\left\{I_1, I_2, \ldots, I_M\right\}\right] = \left(\bigcup_{m=1}^{M} Out\left(O_k\left[I_m\right]\right), \bigcup_{m=1}^{M} In\left(O_k\left[I_m\right]\right)\right)$$

Результатом проекції об'єкту на об'єкт є проекція об'єкту на множину вхідних інтерфейсів об'єкта: $O_k\left[O_l\right] = O_k\left[In\left(O_l\right)\right]$

З визначення операцій проекції об'єкта та взаємодії між об'єктами випливає рівність: $O_k \cdot O_l = O_k \cdot O_l\left[O_k\right]$.

Операція проекції об'єкту має вищий пріоритет, ніж операція взаємодії (що позначається конкатенацією), тобто: $\left(O_k \cdot O_l\right)\left[I_m\right] = O_k\left[I_m\right] \cdot O_l$

Визначення операції наслідування (взаємодія другого типу). Успадкування серед об'єктів РПС відбувається на рівні інтерфейсів. Так, якщо об'єкт успадковує інтерфейси іншого об'єкту ($O_k \leftarrow O_l$), то він надає сервіс всієї множини вихідних інтерфейсів цього об'єкту. Тобто:

$$O_k \leftarrow O_l \Rightarrow Out\left(O_k\right) \subseteq Out\left(O_l\right)$$

Це положення випливає із ООП з програмування РПС, де виконання програми може змінюватися динамічно, а не статично.

Отже, успадкування об'єктом іншого об'єкту — це об'єкт, у якого множина вихідних інтерфейсів містить всі інтерфейси як першого, так і другого типу об'єктів, а множина вхідних інтерфейсів містить тільки ті інтерфейси, які необхідні для надання сервісу цих інтерфейсів.

$$O_k \leftarrow O_l = \begin{pmatrix} Out\left(O_k\right) \cup Out\left(O_l\right), \\ \left\{I_m : \left(I_m \in In\left(O_k\right) \cup In\left(O_l\right)\right) \wedge \exists I_n \in Out\left(O_k \leftarrow O_l\right) : exec\left(I_n, I_m\right)\right\} \end{pmatrix}$$

При операції успадкування об'єкт, що унаслідується, делегує іншому об'єктові свої інтерфейси.

З визначення операції наслідування інтерфейсів об'єктів випливають властивості:

транзитивності

$\forall O_{1,2,3} \in O : O_1 \leftarrow O_2, O_2 \leftarrow O_3 \Rightarrow O_1 \leftarrow O_3$.

симетричності $\forall O_k \in O \Rightarrow O_k \leftarrow O_k$.

Розкриття проекції над об'єктами, між якими існує взаємодія другого типу: $(O_1 \leftarrow O_2)[\ ] = O_1[\ ] \leftarrow O_2[\ ]$.

Визначення мови і операцій взаємодії об'єктів в РПС

Основне призначення мови специфікації полягає в тому, щоб описати основні властивості і функцій РПС. Мова специфікацій близька до мови опису інтерфейсів в IDL CORBA [3].

Ця мова специфікації інтерфейсів об'єктів подається за допомогою правил Бекуса-Наура (БНФ) і має вигляд:

<інтерфейс об'єкта> ::= **Object** *<ім'я_Об'єкту>* ***:{*** <множина вихідних інтерфейсів>*}; {*<множина вхідних інтерфейсів>*}* **end**

<множина вхідних інтерфейсів> ::= <множина інтерфейсів>

<множина вихідних інтерфейсів> ::= <множина інтерфейсів>

<множина інтерфейсів> ::= ∅ | <інтерфейс>; <множина інтерфейсів>*;*

<інтерфейс> ::= **Interface <***ім'я_інтерфейсу***>*: *{*<множина функцій>*}* **end**

<множина функцій> ::= ∅ | <функція>; <множина функцій>*;*

<функція> ::= **function <***ім'я_функції***>*: <множина параметрів> **end**

<множина параметрів> ::= <параметр> | <параметр>, <множина параметрів>

<параметр> ::= <тип> (<вид параметру>)

<вид параметру> ::= **in | out | inout**

Тип описує множину типів даних, які підтримуються мовами програмування (C#, Pascal т.і.) та забезпечують взаємодію між процесами, а <вид параметру> це: **in** — вхідний параметр, **out** — вихідний параметр, **inout** — сумісний параметр.

Множини вхідних та вихідних інтерфейсів мають різну семантику, але однаковий синтаксис. На формальному рівні взаємодію між об'єктами через інтерфейс будемо описувати за допомогою операції конкатенації (·).

Послідовне середовище РПС можна описати за допомогою програми, що виконується у цьому середовищі. Оскільки, основною відмінністю між послідовним та розподіленим середовищем є можливість паралельної еволюції об'єктів у останньому, то необхідно ввести операцію, що описує цю властивість. За допомогою цієї операції виникає можливість розширення класу РПС, тобто опису інших РПС, що не ввійшли до опису операцій над об'єктами.

Визначення операцій над об'єктами

Надалі елемент множини елементів РПС позначимо Р та дамо визначення операцій над ним [5-10].

Визначення паралельного виконання РПС. Результатом паралельної виконання двох РПС буде середовище: $P_o || P_r$

Якщо у розподіленому середовищі РПС виконується не лише два РПС, а декілька, тоді розподілене середовище описується так: $P_o || ... || P_r$

Операцію паралельного виконання можна розширити і перенести на множину об'єктів, оскільки РПС є об'єктом. Але зауважимо, що результатом паралельного виконання об'єктів буде не об'єкт, а середовище, у якому між об'єктами не виникає взаємодії ні першого, ні другого типу: $O_k \| \ldots \| O_l$

Розширимо раніше визначені операції над об'єктами.

*Визначення взаємодії об'єкту і середовища (*взаємодія першого типу*).* Результатом взаємодії об'єкта і середовища буде об'єкт, у якого множина вхідних інтерфейсів співпадає з множиною вхідних інтерфейсів об'єкта-сервера, а множина вихідних інтерфейсів — об'єднання множин вихідних інтерфейсів об'єктів середовища.

$$O_k \cdot \left(O_{l_1} \| \ldots \| O_{l_n}\right) = \left(Out\left(O_k\right), \bigcup_{j=1}^{n} In\left(O_{l_j}\right) \right)$$

Для подальшого використання вважаємо, що множина вхідних інтерфейсів при такій взаємодії є порожньою (накладемо таке обмеження). Оскільки при взаємодії об'єкта $O_k \cdot \left(O_{l_1} \| \ldots \| O_{l_n}\right)$ виникає не детермінованість взаємодії (який із об'єктів середовища взаємодіє з об'єктом-сервером), тобто $O_k \cdot \left(O_{l_1} \| \ldots \| O_{l_n}\right) = \left(Out\left(O_k\right), \{\ \}\right)$.

Взаємодія об'єкту і середовища $O_k \cdot \left(O_{l_1} \| \ldots \| O_{l_n}\right)$ є коректною, якщо виконується умова: середовище повністю забезпечує сервіс, необхідний об'єкту-клієнту:

$$\forall I_m \in In\left(O_k\right) \Rightarrow \exists I_n \in \bigcup_{j=1}^{n} Out\left(O_{l_j}\right) \wedge I_m = I_n$$

Визначення проекції середовища. Результатом проекції середовища на інтерфейс буде середовище, у якого всі об'єкти є проекціями об'єктів середовища.

$$\left(O_k \| \ldots \| O_l\right)\left[I_m\right] = O_k\left[I_m\right] \| \ldots \| O_l\left[I_m\right]$$

Аналогічно визначаємо проекцію середовища на множину інтерфейсів та на об'єкт. З визначення операцій проекції об'єкта та взаємодії між об'єктами випливає рівність:

$$O_k\left(O_{l_1} \parallel \ldots \parallel O_{l_n}\right) = O_k\left(O_{l_1} \parallel \ldots \parallel O_{l_n}\right)\left[O_k\right].$$

Визначення операції успадкування (взаємодія другого типу). У*спадкування* об'єктом інтерфейсів від РПС є об'єкт, що успадковує інтерфейси всіх об'єктів середовища. Дана операція в ООП також називається множинним успадкуванням. Надалі будемо використовувати цей термін як синонім.

При успадкуванні інтерфейсів від середовища виникає не детермінованість коли існують два (або декілька) об'єкти, що надають сервіс по одному інтерфейсу. Для виключення не детермінованості запропоновано вважати, що операція паралельного виконання (в даному випадку) не симетрична, тобто: $O_k \leftarrow \left(O_{l_1} \parallel O_{l_2}\right) \neq O_k \leftarrow \left(O_{l_2} \parallel O_{l_1}\right)$.

Класи РПС, що описуються за допомогою мови специфікації. Описавши операції паралельного виконання РПС та об'єктів розширимо множину РПС, що описуються мовою специфікації. Тепер об'єкт може отримувати сервіс не тільки від одного, але і від багатьох об'єктів. Всі запити щодо інтерфейсу та сервіс об'єкт O_{k_1} отримує тільки від об'єкту O_{k_2}, який в свою чергу, якщо виникає необхідність, отримує сервіси від $O_{l_1} \ldots O_{l_n}$. Тобто між об'єктами O_{k_1} та O_{k_2} існує взаємодія першого типу, а об'єкти $O_{k_1} \ldots O_{k_n}$ виконуються паралельно, і визначають РПС класу $O_{k_1} \cdot O_{k_2} \cdot \left(O_{l_1} \parallel \ldots \parallel O_{l_n}\right)$.

Запити щодо інтерфейсу та сервіс об'єкт O_{k_1} отримує від об'єкту O_{k_2}, якщо цей об'єкт надає сервіс по цьому інтерфейсу, в противному разі від першого із об'єктів $O_{l_1} \ldots O_{l_n}$, який надає цей сервіс. Тобто об'єкт O_{k_2} наслідує інтерфейси від взаємодії другого типу.

Використвуючи ці формальні операції можливо побудувати абстрактную модель РПС, а потім її довести до програмної реалізації в операційному середовищі, в якому є засоби підтримки класів і операцій над їх екземплярами, а також принципів взаємодії різнорідних об'єктів.

Висновки

В роботі розглянуто підхід до проектування РПС на основі об'єктів 1 і 2 типів, які відображають функції предметної області і принципи взаємодії об'єктів функцій між собою в межах РПС. Запропоновано мову взаємодії цих об'єктів, набор формальних операцій представлення функцій об'єктів та механізмів взаємодії об'єктів 2- типу з іншими у середовищі РПС. Введені операції забезпечуються операціями ООП в часті успадкування, агрегації і поліморфізму.

Запропоновані результати досліджень формального опису процесів взаємодії об'єктів, які можуть використовуватися при розробці нових РПС спеціального призначення для виконання у сучасних гетерогенних середовищах.

Список використаних джерел

1. Гради Буч. Объектно-ориентированное проектирование с примерами применения. - Киек: Диалектика, 1992. - 519 с.
2. Орфали Р., Харки Д., Эдвардс Д. Основы CORBA: Пер. с англ. – М., МАЛИП, 1999. – 318 с.
3. В. Эммерих. Конструирование распределенных объектов. Методы и средства программирования интероперабельных объектов в архитектурах OMG/Corba, Microsofts/COM, Java/Rmi.-M.: Мир, 2002.-510 с.
4. Грищенко В.Н. Подход к формализации объектно-ориентированной методологии //Проблемы программирования. - Киев: 1997. - №1.- с.33-39.
5. В.М. Грищенко. Підхід до аналізу поведінки об`єктних систем при моделюванні предметних областей // Проблемы программирования.-Киев: 1998.- №4.- с. 67-75.
6. Буч Г., Рамбо Д., ДжекобсонА. Язык UML. Руководство пользователя.– М.: ДМК, 2000, 430с.
7. Андон Ф.И., Лаврищева Е.М. Методы инженерии распределенных компьютерніх приложений.– Киев, Наукова думка, 1997. - 228с.
8. Грищенко В.М. Підходи та моделі взаємодії програмних компонентов на основі XML. ж. Проблемы программирования, 2000, №3–4 .-с.28-37.
9. Лаврищева Е.М. Сборочное программирование. Основы индустрии программных продуктов.- Наукова Думка- 2009.- 371с.

A.A.Letichevsky and D.R.Gilbert.A general theory of action language. Kibernetika, (1):16-36, January-Febrary.- 1998.

ОПЕРАЦІЇ НАД СУТНОСТЯМИ В ІНТЕЛЕКТУАЛЬНОМУ ОБ'ЄКТНОМУ СЕРЕДОВИЩІ

Д.О. Терлецький

Київський національний університет імені Тараса Шевченка,
Київ, Україна

dmytro.terletskyi@gmail.com

Вступ. Поняття «множина» є центральним для теорії множин і одним із фундаментальних понять для математики в цілому, але окрім цього це поняття має важливе та глобальне значення, в життєдіяльності людини. Ми використовуємо множини повсякденно в своїй розумовій діяльності в процесі сприйняття, аналізу, порівняння, пошуку, класифікації, тощо. Ми свідомо або підсвідомо створюємо множини, оперуємо ними, застосовуємо до них всі відомі нам теоретико-множинні операції і не тільки. Проте аналізуючи визначення поняття «множина», що наведені в [1], виникають питання про походження «конкретних» множин.

Постановка задачі. Виходячи з основ класичної теорії множин [2], можна зробити висновок, що «нові» множини можна отримувати шляхом теоретико-множинних операцій над «вже існуючими» множинами і це дійсно так. Проте не зникають питання про походження цих так званих «вже існуючих» множин, їхню кількість, їх види і так далі. Виникають питання про можливості автоматизації процесу створення множин для машини, без сторонньої допомоги; автоматизації класифікації та ідентифікації елементів множин; автоматичної генерації множин, що відносяться до певного класу. Тобто виникає питання про можливість практичної реалізації для машин здатності оперувати такою базовою категорією людського мислення як «множина».

Пропонований підхід. Враховуючи всі особливості класичної теорії множин [2], пропонується деяке її розширення та формалізація. Зокрема вводиться новий рівень – рівень об’єктів, які є складовими для створення множин, що по суті в деякому сенсі розширює класичну теорію і виступає деяким конструктивним поясненням існування множин. Також вводиться концепція класу об’єктів, що дозволяє в певному сенсі формалізувати класифікацію самих об’єктів. При цьому розглядається скінченна версія теорії множин в якій відкидаються поняття «одноелементна множина» та «пуста множина», що дозволяє в певному сенсі назвати запропонований підхід – підходом до створення деякої альтернативної теорії множин, яка частково спирається на ідеї викладені в [3].

Концепція об'єкта. Відомо, що будь-яка множина складається з елементів, які її утворюють (визначають, формують). В якості елементів множини можуть виступати будь-які предмети, явища нашої уяви або навколишнього світу, називатимемо такі елементи *об'єктами*. Розглянемо такий об'єкт, як натуральне число, очевидно, що кожне натуральне число повинне бути цілим та бути додатнім – що є по суті характеристичними властивостями натуральних чисел, звідси не важко переконатися, що 8 — це дійсно натуральне число, а −4 або 2.9 – не є такими. Аналізуючи вище наведені факти, можна зробити висновок, що будь-який об'єкт володіє певними властивостями, які є для нього характеристичними і визначають його, як деяку сутність та дозволяють відрізняти його від інших об'єктів.

Об'єкт A – це носій деякої сукупності властивостей та ознак $P(A) = (p_1(A),...,p_n(A))$ (специфікації) об'єкту A, що визначають його, як деяку сутність. Позначатимемо довільний об'єкт наступним чином:

$$A = A / P(A) = A/(p_1(A),...,p_n(A)).$$

Очевидно, що специфікація кожного об'єкту містить певну кількість властивостей, тому доцільно ввести поняття розмірності об'єктів. Розмірність об'єкта $D(A)$ – це натуральне число n, яке визначає кількість властивостей об'єкту A, що входять до його специфікації $P(A)$.

Об'єкти A і B є однотипними (різнотипними) тоді і тільки тоді коли вони мають однакову розмірність і їм властива (не властива) одна і та ж специфікація, тобто $D(A) = D(B)$ і $P(A) \equiv P(B)$.

Властивості об'єктів глобально можна поділити на два типи – якісні та кількісні. Кількісні властивості представлятимемо наступним чином

$$p_i(A) = (v(p_i(A)), u(p_i(A))),$$

де $p_i(A)$ – це ідентифікатор i-ї властивості, $v(p_i(A))$ – це кількісне значення і $u(p_i(A))$ – це одиниці вимірювання кількісного значення властивості. Якісні властивості представлятимемо наступним чином

$$p_i(A) = f_i(A),$$

де $f_i(A)$ – це функція, що демонструє виконання чи не виконання властивості $p_i(A)$ для об'єкту A і приймає значення з множини

$M = \{0,1\}$.

Концепція класу. В загальному об'єкти можна поділити на конкретні (реально матеріально існуючі) та абстрактні. Кожен конкретний об'єкт, не залежно від того коли і як він був створений, є нічим іншим, як матеріальною реалізацією свого абстрактного образу – прототипу. Кожен прототип по суті, є абстрактною специфікацією для створення майбутніх реальних об'єктів. Окрім властивостей об'єктів також варто виділити операції (методи), які можна застосувати до об'єктів враховуючи особливості їх специфікації. В програмуванні [4], часто використовують специфікацію та сигнатуру об'єкта без самого об'єкту називаючи це типом або класом об'єкту, тому використавши цю ідею сформулюємо визначання неоднорідного класу.

Клас об'єктів – це абстрактна специфікація та набір операцій (сигнатура) певної кількості довільних об'єктів. Позначатимемо клас об'єктів наступним чином:

$$T = (P(T), F(T)) = ((p_1(T),...,p_n(T)),(f_1(T),...,f_k(T)))$$

де T – ім'я класу, $P(T)$ – абстрактна специфікація, а $F(T)$ – набір операцій (методів), які можна застосовувати до об'єктів класу T враховуючи особливості їх специфікації.

Операції над об'єктами. Використовуючи концепцію об'єкта та класу об'єктів, введемо деякі операції над об'єктами.

Множина об'єктів S – це об'єднання $\cup$ в єдине ціле не менше ніж двох довільних об'єктів довільних класів, тобто

$$S = (A_1 / T_1) \cup (A_2 / T_2) \cup ... \cup (A_n / T_n) = \{A_1, A_2,..., A_n\} / T_S,$$

і $T_1 \cup T_2 \cup ... \cup T_n = T_S$ де T_S – клас новоутвореної множини, специфікація якого має наступну структуру

$$P(T_S) = (Core(T_S), pr_1(A_1),..., pr_n(A_n)),$$

де $Core(T_S)$ – це ядро специфікації класу T_S, що включає в себе властивості, які є спільними для специфікацій $P(A_1),...,P(A_n)$; $pr_1(A_1),..., pr_n(A_n)$ – проекції об'єктів, що складаються з властивостей, які властиві лише цим об'єктам.

Перетин $\cap$ двох довільних об'єктів A_1 і A_2, що володіють специфікаціями $P(A_1)$ та $P(A_2)$ і належать до класів $T(A_1)$ та $T(A_2)$ відповідно, — це абстрактний об'єкт класу T_A, специфікація якого визначається наступним чином

$$P(T_A)=\begin{cases} p_{i_1}(A_1), Eq(p_{i_1}(A_1), p_{i_2}(A_2))=1; \\ p_i(T_A), Eq(p_{i_1}(A_1), p_{i_2}(A_2))=k, k\in(0,1); \\ break, i_1 \vee i_2 > \min(D(A_1), D(A_2)). \end{cases}$$

Різниця / двох довільних об'єктів A_1 і A_2, що володіють специфікаціями $P(A_1)$ та $P(A_2)$ і належать до класів $T(A_1)$ та $T(A_2)$ відповідно, — це абстрактний об'єкт класу T_S, специфікація якого визначається наступним чином

$$P(T_A)=\begin{cases} p_{i_1}(A_1), Eq(p_{i_1}(A_1), p_{i_2}(A_2))=0; \\ p_i(T_A), Eq(p_{i_1}(A_1), p_{i_2}(A_2))=k, k\in(0,1); \\ p_{i_1}(A_1), i_1 > i_2; \\ break, i_1 > D(A_1). \end{cases}$$

Симетрична різниця ÷ двох довільних об'єктів A_1 і A_2, що володіють специфікаціями $P(A_1)$ та $P(A_2)$ і належать до класів $T(A_1)$ та $T(A_2)$ відповідно, — це абстрактний об'єкт класу T_S, специфікація якого визначається наступним чином

$$P(T_A)=\begin{cases} p_{i_1}(A_1), p_{i_2}(A_2), Eq(p_{i_1}(A_1), p_{i_2}(A_2))=0; \\ p_i(T_A), Eq(p_{i_1}(A_1), p_{i_2}(A_2))=k, k\in(0,1); \\ p_{i_1}(A_1), i_1 > D(A_2); \\ p_{i_2}(A_2), i_2 > D(A_1); \\ break, i_1 \vee i_2 > \max(D(A_1), D(A_2)). \end{cases}$$

Результатом операції клонування об'єкта A буде новий об'єкт (клон) A_i, $i=\overline{1,n}$, який володітиме такою ж специфікацією буде належати до того ж класу, що й об'єкт A, тобто $P(A)\equiv P(A_i)$ та $T_A = T_{A_i}$

$$Clon(A) = (A_i/(p_1(A),...,p_n(A)))$$

де A_i – i-й клон об'єкту A.

Розглянемо деякі об'єкти A, B та C, що володіють специфікаціями $P(A)$, $P(B)$ та $P(C)$, та належать до класів

T_A, T_B та T_C відповідно. Тепер застосуємо до них операцію об'єднання та створимо нову множину

$$S = (A / T_A) \cup (B / T_B) \cup (C / T_C) = \{A, B, C\} / T_S,$$

де T_S – клас множини S, специфікація якого в загальному випадку матиме вигляд:

$$P(T_S) = (Core(T_S), pr(A), pr(B), pr(C)).$$

Зрозуміло, що ядро $Core(T_S)$ існуватиме лише тоді коли специфікації $P(A), P(B)$ та $P(C)$ міститимуть хоча б одну спільну властивість.

Тепер за допомогою операції клонування двічі A та B.

$$Clon(A) = (A_1 / (p_1(A), ..., p_n(A)))$$
$$Clon(A) = (A_2 / (p_1(A), ..., p_n(A)))$$
$$Clon(B) = (B_1 / (p_1(B), ..., p_m(B)))$$
$$Clon(B) = (B_2 / (p_1(B), ..., p_m(B)))$$

Після чого застосуємо до об'єктів та їх клонів операцію об'єднання та створимо нову множину.

$$S = A \cup B \cup A_1 \cup A_2 \cup B_1 \cup B_2 = \{A, B, A_1, A_2, B_1, B_2\} / T_S$$

де T_S – клас множини S, специфікація якого в загальному випадку матиме вигляд:

$$P(T_S) = (Core(T_S), pr(A), pr(B)).$$

Очевидно, що множина S буде мультимножиною, оскільки $A \equiv A_1 \equiv A_2$, $B \equiv B_1 \equiv B_2$. Враховуючи це, можна зробити висновок, що мультимножини можна створювати за допомогою суперпозиції операцій об'єднання та клонування об'єктів, тобто

$$S = \bigcup_{i=1}^{n} \left(A_i \bigcup_{j=1}^{n_i} \left(Clon_j(A_i) \right) \right).$$

Приклади. Розглянемо два об'єкти A_1 та A_2 Об'єкту A_1 відповідатиме слово української мови «кольоровий», а об'єкту A_2 – слово «барви». Опишемо їх специфікацією $P(A_i), i = \overline{1,8}$, де p_1 – це «к-ть літер в слові», p_2 – це «к-ть голосних літер в слові», p_3 – це «к-ть приголосних літер в слові», p_4 – це, «рід слова», p_5 – це,

«кількісна форма слова» і p_6 – це «набір літер з яких складається слово». Тепер застосуємо до них операції об'єднання, перетину, різниці та симетричної різниці (див. Таблиця 1).

Таблиця 1. Результати операцій над об'єктами A_1 та A_2

$A_i / p_j(A_i)$	p_1	p_2	p_3	p_4	p_5	p_6
A_1	10	4	6	ч.	одн.	{к,о,л,ь,о,р,о,в,и,й}
A_2	5	2	3	с.	множ.	{б,а,р,в,и}
$A_1 \cup A_2$	(10, 5)	(4, 2)	(6, 3)	(ч.,с.)	(одн., множ.)	({к,о,л,ь,о,р,о,в,и,й}, {б,а,р,в,и})
$A_1 \cap A_2$	5	2	3	×	одн.	{р,в,и}
A_1 / A_2	5	2	3	×	×	{к,о,л,ь,й}
A_2 / A_1	×	×	×	×	множ.	{б,а}
$A_1 \div A_2$	5	2	3	×	множ.	{к,о,л,ь,й,б,а}

Висновки. В даній роботі вводиться концепція об'єкту та класу об'єктів, пропонується конструктивний метод побудови неоднорідних множин, який дозволяє не лише будувати (генерувати) множини, а й класифікувати їх, що дозволяє дещо по іншому розглянути задачі класифікації та ідентифікації об'єктів. Отримані результати можуть бути застосовані в області штучного інтелекту, зокрема в проектуванні та розробці інтелектуальних інформаційних систем.

Список використаних джерел

1. Г. Кантор. Труды по теори множеств. – Москва: Наука, 1985. – 430 с.
2. Р. Р. Столл. Множества. Логика. Аксиоматические теории. – Москва: Просвищение, 1968. – 231 с.
3. П. Вопенка. Математика в альтернативной теории множеств. – Москва: Мир, 1983. – 150 с.
4. Б. Страуструп. Язык программирования С++. Специальное издание. – Издательство: Бином, Невский Диалект, 2004. – 1054 с.

РОЗПІЗНАВАННЯ ОБ'ЄКТІВ З ВИКОРИСТАННЯМ КРИТЕРІЮ НА ОСНОВІ ВЕКТОРНОЇ МІРИ БЛИЗЬКОСТІ ОБРАЗІВ У ПРОСТОРІ ПОХИБОК

П.В. Четирбок

Республиканський вищий навчальний учбовий заклад
«Крымский гуманитарный университет», Ялта, Україна

petr58@ukr.net

Метою роботи є розробка нової інформаційної технології розпізнавання сигналів електрооптичних зображень з використанням критерію на основі векторної міри близькості образів у просторі похибок.

Об'єктом дослідження є об'єкті електрооптичних зображень.

Предмет дослідження є методи розпізнавання об'єктів електрооптичних зображень.

Методи дослідження ґрунтуються на теорії нечітких та гібридних нейронних мереж.

Для досягнення поставленої мети в роботі розв'язаний комплекс задач, зокрема, науково-технічних, у рамках яких проведена класифікація існуючих методів й алгоритмів розпізнавання образів, визначена узагальнена методика побудови автоматизованих систем розпізнавання образів й алгоритмів навчання нейронних мереж; прикладних, у рамках яких розроблений інформаційно-програмний комплекс, що реалізує розроблені методи й алгоритми, комплекс є ядром універсальної системи інтелектуального аналізу даних, що розв'язує задачі розпізнавання об'єктів електрооптичних зображень; експериментальних задач, у рамках яких проведені експериментальні дослідження ефективності роботи реалізованих моделей, методів й алгоритмів, розроблені рекомендації.

Отримані в роботі результати наочно демонструють ефективність використання розроблених моделей, методів й алгоритмів для розв'язання задач розпізнавання сигналів. За допомогою мультиспектральной електрооптичної системи, що працює в трьох діапазонах – червоному, зеленому й синьому, були отримані знімки прибережної смуги океану й було потрібно розпізнати об'єкти у вигляді геометричних фігур на водній поверхні та на піску. Для цих цілей з огляду на складність задачі, а також великий рівень перешкод запропоновано використати нечіткі нейронні мережі, зокрема NEFClass. Для вибору даних з електрооптичних зображень використовується система ENVI та можливість системи картографувати, тобто суміщати отримані з

різних спектральних камер зображення по контрольних точках. Це дозволяє отримати мультиспектральне зображення.
У роботі вирішені наступні задачі:
1. Побудовано вирішальне правило для класифікації об′єктів у формі електрооптичних зображень, яке забезпечує надійне розпізнавання в умовах наявності шумових складових зображень.
2. Розроблено модель асоціативної пам'яті з використанням критерію на основі векторної міри близькості образів у просторі похибок. Асоціативна пам'ять надійно запам'ятовує m образів, $m \le n$, де n – розмірність простору вхідних образів. Критерій дозволяє створити модель розподіленої пам'яті.
3. Застосування критерію векторної міри близькості образів у просторі похибок дозволило скоротити мінімальну кількість помилково класифікованих зразків об'єктів електрооптичних зображень. Так без використання критерію всі алгоритми дали приблизно 4% помилкових класифікацій, а з використанням критерію градієнтний алгоритм дав 1% помилкових класифікацій. При цьому похибка не зростає при зростанні складності моделі.

Список використаних джерел

1. Четырбок П.В. Динамический подход в обучении нейронных сетей с использованием скалярного критерия для распознавания образов / Четырбок П.В. // – Луганськ: Вісник Східноукраїнського національного університету імені Володимира Даля. – 2009.– №1(131) ч. 2. – С. 112 – 114.
2. Четырбок П.В. Алгоритм сжатия информации с использованием скалярного критерия распознавания образов /Четырбок П.В. // Системні технології. – Дніпропетровськ: Регіональний міжвузівський збірник наукових праць. – 2008. – №6. – С.185 –190.
3. Четырбок П.В. Скалярный критерий распознавания образов – функционал качества системы управления / Четырбок П.В. // Штучний інтелект. – Донецьк: ІПШІ «Наука і освіта». – 2009. – №4. – С. 100 – 103.
4. Rozsoha O.O, Chetyrbok P.V. Scalar Criterion for the Pattern Recognition /Chetyrbok P.V. // The 5 International conference “Information technologies and management” – Latvia, Riga – 2007. – S.56 – 57.

ОДИН ПІДХІД ДО ІНДЕКСАЦІЇ У ОНТОЛОГІЧНО-КЕРОВАНІЙ ІНФОРМАЦІЙНО-ПОШУКОВІЙ СИСТЕМІ

А.С. Шабінський

Національний університет «Києво-Могилянська академія»,
Київ, Україна

anton.shabinskiy@gmail.com

Онтологічно-керована інформаційно-пошукова система

Називатимемо деяку інформаційно-пошукову систему (ІПС) онтологічно-керованою (ОКІПС), якщо схожість документів (оцінка релевантності) у ній визначається не напряму, а через інтерпретацію документів у термінах онтології предметної області (концепти та відношення). У процесі пошуку релевантних документів ОКІПС оперує концептами різного рівня абстракції (поняттями, темами) на противагу ІПС, що працює з явними елементами представлення документа (наприклад, слова та фрази у тексті). ОКІПС може оперувати структурними елементами документів на інших етапах, наприклад, індексації колекції, попередньої обробки документів, інтерпретації документів у термінах онтології, аналізу та обробки результатів пошуку тощо. Надалі під документами та пошуком ми матимемо на увазі відповідно текстові документи та текстовий інформаційний пошук; під ІПС ми розумітимемо систему, яка не керується онтологією.

Проблема індексації в ОКІПС

Класичний пошуковий індекс складається зі словника та списків словопозицій. [1] Така структура індексу дозволяє знаходити спочатку окремі терміни у словнику, а потім визначати документи, що містять ці терміни. Ця структура і спосіб роботи з нею підходять для звичайних ІПС, які не керуються онтологіями для оцінки схожості документів. Далі розглянемо схематичне зображення пошукового процесу у ІПС та ОКІПС, яке подане на рис. 1. Зображені процеси передбачають найпоширеніший варіант перебігу подій, коли на вхід подається документ (запит), для якого потрібно знайти релевантні відповідники. Процес пошуку в ОКІПС має один принциповий додатковий крок – інтерпретацію вхідного запиту у термінах онтології. Надалі у обох системах відбувається схожий процес: подані на вхід елементи запиту відшукуються у індексі. Для ІПС такими елементами є терміни, нормалізовані з вхідного документу на кроці попередньої обробки. Для ОКІПС таким елементами є концепти, виокремлені з запиту на кроці інтерпретації.

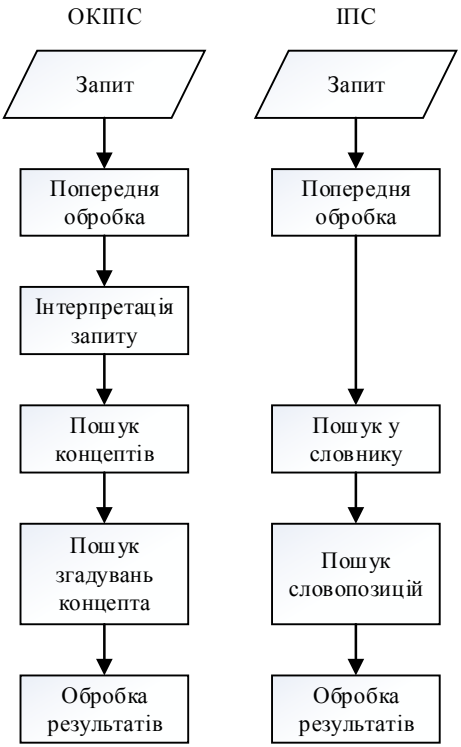


Рис. 1. Схематичний процес пошуку у ІПС та ОКІПС

Очевидно, що структура для індексу в ОКІПС мусить бути іншою: замість слів у словнику – концепти, замість словопозицій – згадування концептів у документі. Процес побудови такої структури – онтологічно-орієнтована індексація колекції документів – суттєво відрізняється від процесу текстової індексації.

Онтології, тематичні карти та тематичні моделі

Тематичні карти [2] [3] є однією з форм представлення онтологій. Тематична карта виражає онтологію у вигляді концептів – тем – та відношень (асоціацій) між ними. Ієрархія тем може бути скільки завгодно складною. Тематичні карти багато в чому подібні до стандарту W3C RDF [4] [5], хоча останні орієнтовані на ресурси, а не на теми як такі. Онтологія в ОКІПС може мати будь-яку форму і походження, але ми зосередимося на тематичних картах з двох причин: по-перше, вони дозволяють інтуїтивно моделювати зміст документів, інтерпретуючи їх як розкриття певних тем у деяких пропорціях; по-друге, тематичні карти придатні до автоматизованої побудови з мінімальною участю людини-експерта. Додатково

тематичні карти забезпечені стандартизованим XML синтаксисом [6] та специфікацією прикладного програмного інтерфейсу (API) з різноманітними імплементаціями [7].

Особливу привабливість тематичних карт як онтологій для ОКІПС забезпечують ймовірнісні тематичні моделі. Ймовірнісні тематичні моделі (ТМ) – це алгоритми для виявлення тематичного наповнення документів у великих неструктурованих колекціях. [8] Найпростішою тематичною моделлю є приховане розміщення Діріхле (latent Dirichlet allocation) [9]. ТМ здатні статистично визначити групу розподілів на словнику – теми – та визначити розподіл на цих темах у колекції. У контексті побудови ієрархічних тематичних карт особливої уваги заслуговують корельована тематична модель [10] та модель розміщення патінко (pachinko) [11]. Ці ТМ дозволяють визначити . Такі тематичні моделі дозволяють автоматизувати побудову онтологій для ОКІПС, отримуючи при цьому досить виразні і потужні структури.

Розглянемо індексацію текстових колекцій у ОКІПС із тематичною картою у якості онтології.

Онтологічно-орієнтована індексація

У пропонованому нами підході індекс ОКІПС суміщається із онтологією (тематичною картою). Така карта-індекс об'єднує в собі одночасно інформацію щодо розподілу на темах для кожного документу, розподіли на словнику для кожної теми, розподіли на темах для відношень «супер-тема – підтема» та виконує роль індексу ОКІПС, за яким може відбуватися як класичний пошук семантично схожих документів, так і огляд колекції документів через переходи від теми до теми. Описана структура зображена на рис. 2.

Практично структура має такий характер. Дерево тем, яке є аналогією словника у класичному текстовому індексі, значно компактніше. Як відомо, закон Хіпса [12] дозволяє приблизно оцінити кількість термінів за кількістю лексем, і для мільйона лексем дає оцінку в ~39 тис. термінів. Кількість тем, навпаки, не зростає настільки швидко з розміром колекції і не перевищує кількох сотень для колекції в 20000 документів (точні значення залежать від колекції та задачі). Очевидно, дерево тем доцільно зберігати в оперативній пам'яті.

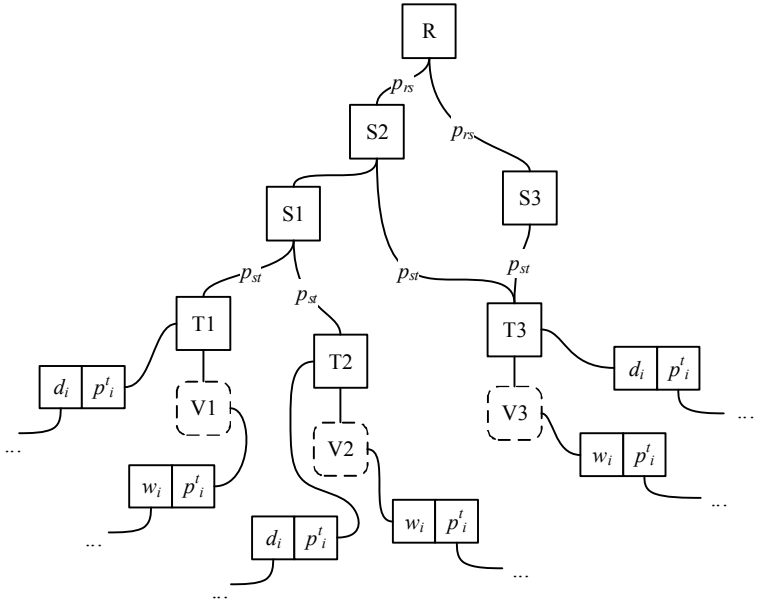


Рис. 2. Онтологічно-орієнтований індекс. *S*, *T* – супер-теми, теми; *V* – розподіли на словнику, де w_i – слово, p^t_i – ймовірність слова *i* для теми *t*; темі відповідає список ID документів d_i із ймовірностями p^t_i з розподілу на темах (ймовірність теми *t* для документу *i*); p_{st} та p_{rt} – розподіли на підтемах.

Кожна листова тема в дереві містить указники на два списки: ID документів (з ймовірностями з розподілу на темах) та слів словника (з ймовірностями з розподілу на словнику для цієї теми).

Створення карти-індексу відбувається у два етапи:

а) навчання моделі на колекції

б) обробка отриманих результатів моделі для побудови індексу.

Актуалізація індексу

У динамічних колекціях (що постійно поповнюються) постає проблема актуалізація індексу. У класичних пошукових системах з текстовими індексами через складність динамічної індексації часто вдаються до повною переіндексації колекції. Процес навчання тематичних моделей значно складніший за звичайну індексацію і унеможливлює часту переіндексацію у ОКІПС. Проте, завдяки наявній тренованій моделі при додаванні нових документів, що відповідають за змістом початковій навчальній колекції, можна

здійснювати вивід розподілу на темах для документу і оновлювати індекс відповідно до отриманих результатів.

Зовсім інша проблема постає при додаванні документів, які не відповідають природі навчальних даних. Наприклад, при появі у колекції документів нової тематики. Вивід тем для таких документів дає незадовільні результати. Наразі проблема актуалізації індексу при появі нових тем залишається у розробці.

Список використаних джерел

1. Manning, Christopher D., Raghavan, Prabhakar та Schütze, Hinrich. Introduction to Information Retrieval. – New York : Cambridge University Press, 2008.
2. An Introduction to Topic Maps // Ahmed, Kal та Moore, Graham. – 5: Microsoft Corporation, 2005 p., The Architecture Journal, сс. 3-9.
3. ISO/IEC JTC1/SC34/WG3. Topic Maps — Part 1: Overview and Basic Concepts. http://www.itscj.ipsj.or.jp/sc34/open/1045.htm.
4. Beckett, Dave , McBride, Brian. RDF/XML Syntax Specification. W3C. http://www.w3.org/TR/REC-rdf-syntax/.
5. Berners-Lee, Tim. Notation3 (N3): A readable RDF syntax. World Wide Web Consortium. http://www.w3.org/DesignIssues/Notation3.
6. ISO/IEC JTC1/SC34/WG3. Topic Maps — XML Syntax. http://www.isotopicmaps.org/sam/sam-xtm/.
7. Common Topic Map Application Programming Interface. http://www.tmapi.org.
8. Blei, David M. Probabilistic Topic Models. – Communications of the ACM. 2012 p., T. 55, 4.
9. Latent Dirichlet Allocation //Blei, David M., Ng, Andrew Y. and Jordan, Michael I. – Cambridge, MA : MIT Press, 2003, Journal of Machine Learning Research, Vol. 3, pp. 993-1022.
10. A correlated topic model of SCIENCE // Blei, David та Lafferty, John. – 1, 2007 p., The Annals of Applied Statistics, T. 1, сс. 17-35.
11. Pachinko Allocation: DAG-Structured Mixture Models of Topic Correlations // Li, Wei та McCallum, Andrew – Pittsburg, 2006. Proceedings of the 23rd International Conference on Machine Learning.
12. Heaps, Harold Stanley. Information Retrieval: Computational and Theoretical Aspects – Academic Press, 197

БАЗЫ ЗНАНИЙ

Н.С. Ялова, П.В. Четырбок

Республиканское высшее учебное заведение
«Крымский гуманитарный университет», Ялта, Украина

zknf1994@mail.ru

База знаний, БЗ (англ. Knowledge base, KB) — это особого рода база данных, разработанная для управления знаниями (метаданными), то есть сбором, хранением, поиском и выдачей знаний. Раздел искусственного интеллекта, изучающий базы знаний и методы работы со знаниями, называется инженерией знаний.

Под базами знаний понимается совокупность фактов и правил вывода, допускающих логический вывод и осмысленную обработку информация. Например, в языке Пролог базы знаний описываются в форме конкретных фактов и правил логического вывода над базами данных и процедурами обработки информации, представляющих сведения и знания о людях, предметах, фактах событиях и процессах в логической форме.

В зависимости от уровня сложности систем, в которых применяются базы знаний, различают:

1) БЗ всемирного масштаба — например, Интернет или Википедия
2) БЗ национальные — например, Википедия
3) БЗ отраслевые — например, Автомобильная энциклопедия
4) БЗ организаций
5) БЗ экспертных систем
6) БЗ специалистов

Знания в базе знаний можно разделить на алгоритмические и неалгоритмические.

- **алгоритмические (процедурные) знания** - это алгоритмы (программы, процедуры), вычисляющие функции, выполняющие преобразования, решающие точно определенные конкретные задачи. Пример: любая библиотека программ.
- **неалгоритмические знания** - состоит из мысленных объектов, называемых понятиями. Понятие обычно имеет имя, определение, структуру (составные элементы), оно связано с другими понятиями и входит в какую-то систему понятий. Другие неалгоритмические знания - это связи между понятиями или утверждения о свойствах понятий и связях между ними.

На практике во многих экспертных системах и СБЗ содержимое базы знаний подразделяют на "факты" и "правила". Факты - элементарные единицы знания (простые утверждения о

характеристиках объекта), правила служат для выражения связей, зависимостей между фактами и их комбинациями. Таким образом, классификацию знаний можно представить следующим образом:

- понятия (математические и нематематические)
- факты
- правила, зависимости, законы, связи
- алгоритмы и процедуры

Прямое использование знаний из базы знаний для решения задач обеспечивается **механизмом получения решений** (inference engine – машина вывода) – процедурой поиска, планирования, решения. Механизм решения дает возможность извлекать из базы знаний ответы на вопросы, получать решения, формулируемые в терминах понятий, хранящихся в базе. Примеры запросов:

- найти объект, удовлетворяющий заданному условию;
- какие действия нужно выполнить в такой ситуации и т.д.

Интерфейс – обеспечивает работу с базой знаний и механизмом получения решений на языке высокого уровня, приближенном к профессиональному языку специалистов в той прикладной области, к которой относится СБЗ.

Для создания СБЗ могут использоваться следующие средства:

1. Традиционные языки программирования - C, Basic, Pascal, Lisp и др. Особо в этом ряду стоит выделит язык функционального программирования Lisp. Его основные свойства: данные представляются в виде списков, для получения решений используется рекурсия.
2. Языки представления знаний (такие как Prolog) - имеют специфические средства описания знаний и встроенный механизм поиска вывода.
3. Пустые оболочки экспертных систем - содержат реализации некоторого языка представления знаний и средства организации интерфейса пользователя. Позволяют практически полностью исключить обычное программирование при создании прикладной экспертной системы.

Список использованной литературы

1. Системы управления базами данных и знаний. /Под ред. А.Н.Наумова. М., 1991.

X МІЖНАРОДНА КОНФЕРЕНЦІЯ
10TH INTERNATIONAL CONFERENCE

ТЕОРЕТИЧНІ ТА ПРИКЛАДНІ АСПЕКТИ ПОБУДОВИ ПРОГРАМНИХ СИСТЕМ
THEORETICAL AND APPLIED ASPECTS OF PROGRAM SYSTEMS DEVELOPMENT

TAAPSD'2013

ПРАЦІ КОНФЕРЕНЦІЇ
PROCEEDINGS

Підп. до друку 14.05.2013. Формат 60x84 1/16. Папір офсет. Друк різограф. Ум. др. арк. 11,16. Тираж 100. Зам. № 3819.

ЦЕНТР ОПЕРАТИВНОЇ ПОЛІГРАФІЇ «АВАНГАРД»

м. Кіровоград, вул. Пашутінська,12, оф. 4.
Тел./факс: **24-86-34, 27-02-24,**
моб. **/050/ 531-73-72, 341-04-33.**
http://avangard.kr.ua, e-mail: **info@avangard.kr.ua**