

УДК 004

О.В.Овсяк, к.т.н.

МОДЕЛІ ДЕКОМПОЗИЦІЇ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ЗАДАННЯ І ОПРАЦЮВАННЯ ОРІЄНТАЦІЇ ОПЕРАЦІЇ ЕЛІМІНУВАННЯ

Львівська філія Київського національного університету культури і мистецтв, Українська академія
друкарства, ovsjak@ukr.net

У роботі створено і описано модель декомпозиції підсистеми елімінування на субсистеми графічного вибору орієнтації та її функціонального опрацювання, а також створення графічного комп'ютерного образу операції елімінування. Розроблено і описано алгеброю алгоритмів моделі декомпозиції субсистем графічного задання і функціонального опрацювання орієнтації операції елімінування. Наведено програмну реалізацію субсистем задання орієнтації.

Ключові слова: алгебра алгоритмів, інформаційні технології, елімінування.

Вступ

Інформаційними технологіями, залежно від їхнього призначення, реалізуються відповідні алгоритми. Відомо методи неформального опису алгоритмів [1], якими є теорія алгоритмів Маркова, рекурсивні функції, машини Колмогорова, Т'юрінга, Поста, Шонгаге та інші. Алгеброю ж алгоритмів [2, 3] алгоритми інформаційних технологій і систем описуються у вигляді математичних формул. Знаків операцій алгебри алгоритмів немає серед відомих математичних знаків. Вони є графічними знаками і мають специфічну будову та їхні геометричні розміри залежать від розмірів унітермів над якими вони виконуються. Знаки операцій алгебри алгоритмів можуть створюватися засобами, наприклад, таких інформаційних систем як Word і CorelDraw. Однак використання універсальних комп'ютерних систем, з погляду необхідності виконання значної кількості дій пов'язаних з допасовуванням і масштабуванням знаків операцій до унітермів, є малоефективним. Тому було створено спеціалізовані редактори Модал [4] і Абстрактал [5]. Комп'ютерна система Абстрактал, у порівнянні з редактором Модал, забезпечує більші функціональні можливості. Цими системами використовуються спеціальні формати даних, які використовуються для збереження синтезованих формул алгоритмів у пам'яті комп'ютера. Хоча у цих системах формати представлення даних є різними але об'єднує їх те, що для кожної з інформацій про унітерми і формули алгоритмів ними відводяться абсолютні позиції і фіксована кількість байтів, а також зберігаються абсолютні координати знаків операцій. Для усунення цього недоліку у моделі комп'ютерної системи генерування програмного коду [6] з формул алгоритмів створено модель *xml* – формату опису формул алгоритмів [7]. Але для реалізації *xml* – формату необхідно створити і відповідні моделі комп'ютерного опрацювання операцій алгебри алгоритмів. Тому у даному дослідженні розглянуто елементи моделі підсистеми [7] опрацювання операції елімінування алгебри алгоритмів.

Модель декомпозиції підсистеми опрацювання операції елімінування

Операція елімінування алгебри алгоритмів має горизонтальну і вертикальну орієнтації [2, 3]. Для їхнього задання методом вибору необхідно ввести субсистему графічної візуалізації з відповідними засобами, а також субсистеми опису і опрацювання вибраних властивостей.

Заголовок субсистеми графічної візуалізації утворюємо доступністю (*pu*) з інших субсистем і підсистем, з властивістю часткової (*par*) підсистеми і наслідуванням (:) стандартної підсистеми Window [8, 9], що запишемо як *pu par @Te.%Ef: Win x:*, де % - ознака субсистеми опису графіки, *Te* – назва системи, @ - ідентифікатор підсистеми, *Ef* – назва субсистеми, : - ознака наслідування, *Win* – скорочена назва стандартної підсистеми Window [8, 9], *x:* – індекс системного простору [9].

Нехай заголовок підсистеми, яка описує можливості орієнтації має загальнодоступний метод доступу і утворює функційну частину підсистеми графічної візуалізації: *pu par @Te.Ef: Win*.

Підсистема опрацювання операції елімінування (*E*) виконується над унітермами і тому наслідує підсистему *T* [6] та має бути доступною із інших підсистем, тому для неї задаємо загальний доступ й утворюємо такий заголовок *pu @Te.E: T*.

Розташування опису підсистем одна відносно другої немає значення але вони є обов'язковими складовими, що опишемо так:

$$pu \ @Te.\%Ef: Win \ x:, pu \ par \ @Te.Ef: Win, pu \ @Te.S: T.$$

Засобами алгебри алгоритмів опишемо моделі субсистем графічного задання і функційного опрацювання орієнтації операції елімінування.

Модель декомпозиції субсистеми графічного задання орієнтації

Нехай субсистема має мати графічне вікно, яке позначмо як $\%Ef$. Опишемо декомпозицію форми графічного вікна на такі параметри:

$$\%Ef = \left(\begin{array}{l} p_1; \\ p_2; \\ Tit = E_{lim}; \\ Hei = 200; \\ Wid = 300; \\ Bacgro = LiGray; \\ WinStarLoc = CenScr; \\ @Gri \end{array} \right)$$

де p_1 – унітерм підключення до використання засобів платформи *Microsoft Visual Studio .NET* [9], p_2 – унітерм задання доступу до стандартних засобів компілятора мови *XAML* [9], $Tit = E_{lim}$ – задання назви заголовку, $Hei = 200$ – опис ширини форми, $Wid = 300$ – довжина форми, $Bacgro = LiGray$ – колір фону і $WinStarLoc = CenScr$ – розташування у центрі екрана, $@Gri$ – опис стандартної підсистеми “сітка”.

Розташування і вирівнювання графічних елементів на поверхні графічного вікна виконується оперативніше якщо розбити поверхню вікна на окремі рівні частини, якими, наприклад, є квадрати. Таке розбиття досягається введенням графічного елемента яким є сітка (Gri), яка реалізується стандартним класом *Grid* [8, 9]. Текстовий ідентифікатор орієнтації операції описується введенням на сітці графічного елемента lab , який є типу підсистеми Lab , що реалізується стандартним класом *Label* [8, 9]. Вибір орієнтації виконуватимемо з використанням графічного елемента ori , який є типу стандартної підсистеми $ComB$, що реалізується стандартним класом *ComboBox* [8, 9]. Графічні елементи згрупуємо у блок, який назвемо $Вла$ і задамо його приналежність до типу стандартної підсистеми $GroB$, яка реалізується у класі *GroupBox* [8, 9]. Ці всі графічні елементи опишемо формулою:

$$@Gri = \left(\begin{array}{l} Wid = 250; \\ lab \in @Lab; \\ *; \\ sep \in @ComB; \\ ori \in @ComB; \\ Вла \in @GroB \end{array} \right)$$

де $Wid = 250$ – довжина сітки, $lab \in @Lab$ – створення елемента стандартного класу *Label*, sep , ori і $Вла$ – елементи стандартних класів *ComboBox* і *GroupBox* [8, 9], які мають такий опис:

$$lab \in @Lab = \left(\begin{array}{l} Hei = 23; \\ VerAli = Top; \\ (Орієнтація); \\ ; \\ Mar = 36; 22; 133; 0; \end{array} \right)$$

$$ori \in @ComB = \left(\begin{array}{l} Hei = 23; \\ VerAli = Top; \\ ; \\ Mar = 120; 57; 27; 0; \end{array} \right)$$

$$Вла \in @GroB = \left(\begin{array}{l} Na = gB; \\ Hea = Вла.Властивості; \\ Mar = 10, 10, 10, 45; \\ (can; \\ ok; \end{array} \right)$$

Для підтвердження зробленого вибору орієнтації вводимо графічні елементи, які назвемо ok та can і задамо їхню приналежність до типу стандартної підсистеми But , яка реалізується у стандартному класі *Button* [8, 9], що опишемо як

$$can \in @But = \left(\begin{array}{l} Hei = 95; \\ Cli = canCli; \\ (Скасувати); \\ ; \\ Mar = 36; 57; 0; 50; \end{array} \right)$$

$$ok \in @But = \left(\begin{array}{l} Hei = 23; \\ Cli = okCli; \\ (Виконати); \\ ; \\ Mar = 0; 0; 30; 10; \end{array} \right)$$

де But – стандартна підсистема *Button*, Hei , $VerAli$, Top , Mar , Cli – стандартні властивості *Height*, *VerticalAlignment*, *Top*, *Margin*, *Click* [8, 9].

Модель декомпозиції субсистеми функціонального опрацювання орієнтації

Нехай орієнтація задається користувачем у діалоговому вікні редактора формул алгоритмів. Для опрацювання заданої орієнтації вводимо унітерм $Orie$ і опишемо його такою формулою справа, де $Orie$ – назва складного унітерма, який є типу $E.Ori$ (у ньому E є назвою підсистеми опрацювання операції елімінування, $Orie$ – секвентна область значень змінної

$$pu\ Ori \in E.Ori = \left(\begin{array}{l} E.Ori.Hor; E.Ori.Ver; (\%ori.SelInd = 0) - ? \\ ; \\ \%ori.SelInd = 0; \%ori.SelInd = 1; (val = E.Ori.Hor) - ? \\ ; \\ * \\ ; \\ set = - ? \\ ; \\ get - ? \end{array} \right)$$

орієнтації); *E.Ori.Hor* – значення горизонтальної орієнтації (*Hor*); *E.Ori.Ver* – значення вертикальної орієнтації; (*%ori. SellInd = 0*)-? – перевірка задання у графічному вікні горизонтальної орієнтації операції елімінування (задана орієнтація визначається аналізом значення стандартної властивості *SelectedIndex* [8, 9] елемента *ori* графічного інтерфейсу); (*val = E.Ori.Hor*) – ? – перевірка збіжності значення системної змінної *value* [8, 9] зі значення змінної орієнтації *E.Ori*.

Вводимо функційний унітерм *Ef()* задання можливих значень змінної орієнтації, які опишемо формулою

$$pr\ Ef() = \overbrace{IniCom(); \% ori. SellInd=0; \% ori. It. + ((\text{"Hor"}) \& (\text{"Ver"}))}^{\text{}} ,$$

де *IniCom()* – функційний унітерм ініціалізації компонентів графічного вікна задання орієнтації, який описується стандартною процедурою *InitializeComponent*[8, 9]; *%ori. SellInd=0* – задання значення графічного елемента *ori*; *%ori. It. + ((\text{"Hor"}) \& (\text{"Ver"}))* – графічному елементу *ori* приписування *Items.Add*[8, 9] можливих значень орієнтації (*"Hor"* та *"Ver"*) операції елімінування.

Задаємо функційний унітерм *Ok()* опрацювання цоку лівою клавішею мишки, коли курсор знаходиться на графічному елементі *ok*. *Ok()* опишемо виразом

$$pr\ Ok(s \in @Obj, e \in @RouEveArg) = \overbrace{DialRes=t; Clo().}^{\text{}} ,$$

де *pr* – ідентифікатор обмеження доступу до функційного унітерма у рамках підсистеми у якій він описаний; *s* – змінна типу підсистеми *Obj*, яка реалізується стандартним класом *Object* [8, 9]; *e* – змінна типу підсистеми *RouEveArg*, яка реалізується стандартним класом *RoutedEventArgs* [1, 2]; *DialRes* – унітерм, який реалізується стандартною системною властивістю *DialogResult* [8, 9]; *t* – логічне значення; *Clo()* – функційний унітерм, який реалізується стандартною системною процедурою *Close*[8, 9], а *.* – ознака завершення виконання функційного унітерма і повернення у місце його вибору.

Також вводимо функційний унітерм *Can()* опрацювання цоку лівою клавішею мишки, коли курсор знаходиться на графічному елементі *can*, який опишемо формулою

$$pr\ Can(s \in @obj, e \in @RouEveArg) = \overbrace{DialRes=f; Clo().}^{\text{}} ,$$

де *f* – логічне значення.

Враховуючи введені унітерми формула декомпозиції підсистема опрацювання вибору орієнтації і розділювача описується виразом справа.

Програмна реалізація моделі субсистеми віконного задання орієнтації

Код програмної реалізації описаний мовою *XAML* [9]:

```
<Window x:Class="TermEdit.SequenceForm"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title="Секвентування" Height="170" Width="290" Background="LightGray"
  WindowStartupLocation="CenterScreen">
  <Grid Width="270">
    <Label Height="23" Margin="36,22,133,0" Name="label1" VerticalAlignment="Top"> Орієнтація:
  </Label>
    <Label Margin="36,57,0,50" Name="label2" HorizontalAlignment="Left" Width="95"> Розділювач:
  </Label>
    <ComboBox Margin="120,57,27,53.5" Name="separator" />
    <ComboBox Height="23" Margin="120,22,27,0" Name="orientation" VerticalAlignment="Top" />
    <GroupBox Header="Властивості" Margin="10,10,10,45" Name="groupBox1" />
    <Button Height="23" Margin="53,0,0,10" Name="cancel" Click="cancel_Click"
  VerticalAlignment="Bottom" HorizontalAlignment="Left" Width="75">Скасувати</Button>
    <Button Height="23" Margin="0,0,30,10" Name="ok" Click="ok_Click" VerticalAlignment="Bottom"
  HorizontalAlignment="Right" Width="74">Виконати</Button>
  </Grid>
</Window>
```

(Orie
; Ef()
;
(Ok()
;
Can()).

Програмна реалізація моделі субсистеми функціонального опрацювання орієнтації

Програмний код написано мовою *C#* і він має такий вигляд:

```
namespace TermEdit
{
    public partial class EliminationForm : Window
    {
        public Elimination.Orientation Orientation
        {
            get
            {
                if (orientation.SelectedIndex == 0)
                    return Elimination.Orientation.Horizontal;
                else
                    return Elimination.Orientation.Vertical;
            }
            set
            {
                if (value == Elimination.Orientation.Horizontal)
                    orientation.SelectedIndex = 0;
                else
                    orientation.SelectedIndex = 1; }
        }
        public EliminationForm()
        {
            InitializeComponent();
            orientation.SelectedIndex = 0;
            orientation.Items.Add("Горизонтальна");
            orientation.Items.Add("Вертикальна");
        }
        private void ok_Click(object sender, EventArgs e)
        {
            DialogResult = true;
            Close();
            return;
        }
        private void cancel_Click(object sender, RoutedEventArgs e)
        {
            DialogResult = false;
            Close();
            return;
        }
    }
}
```

Висновки

1. Декомпозицією підсистеми елімінування на субсистеми зменшено складність створення моделі підсистеми.
2. Декомпозицією субсистем графічного задання і функціонального опрацювання зменшено складність побудови їхніх математичних моделей.
3. Алгебра алгоритмів є засобом побудови математичних моделей як графічних так і функціональних субсистем.
4. Програмною реалізацією і апробацією створені математичні моделі субсистем графічного задання і функціонального опрацювання верифіковані, чим підтверджено опис ними орієнтації операції елімінування.

Список літературних джерел

1. В.А.Успенский, А.Л.Семенов. Теория алгоритмов: основные открытия и приложения. – Москва: “Наука”, 1987. – 288 с.
2. В.К.Овсяк. Засоби еквівалентних перетворень алгоритмів інформаційно-технологічних систем //Доповіді Національної академії наук України, №9, 1996. – С.83-89.
3. Owsiak W., Owsiak A. Rozszerzenie algebry algorytmów /Pomiary, automatyka, kontrola. – № 2, 2010. – S. 184 – 188.
4. Бритковський В.М. Моделювання редактора формул секвенційних алгоритмів: автореф. дис. на здобуття наук. ступеня канд. тех. наук: спец. 01.05.02 “Математичне моделювання та обчислювальні методи” / Бритковський В.М. - Львів, 2003. - 18 с.
5. Василюк А.С. Підвищення ефективності математичного і програмного забезпечення редактора формул алгоритмів: автореф. дис. на здобуття наук. ступеня канд. тех. наук: спец. 01.05.02 “Математичне та програмне забезпечення обчислювальних машин і систем” / Василюк А.С. - Львів, 2008. - 20 с.
6. Овсяк О. Класи інформаційної системи генерування коду /О.Овсяк //Вісник Тернопільського державного технічного університету: “Тернопільський національний технічний університет імені Івана Пулюя“. – № 1, 2010. – С. 171 – 176.
7. Овсяк О.В. Граматика мови текстового опису формул алгоритмів /О.Овсяк //Вісник Тернопільського державного технічного університету, № 3, 2010. – С.106 – 110.
8. Petzold C. Programowanie Windows w języku C#. –Warszawa: „RM”, 2002. – 1161 s.
9. Мэтью Мак-Дональд. Windows presentation foundation в .NET 3.5 с примерами на C# 2008. – Москва, Санкт-Петербург, Киев: “Apress”, 2008. – 922 с.