УДК 004.4’232

О. Овсяк
Київський національний університет культури і мистецтв, Львівська філія,
Українська академія друкарства

МОДЕЛЬ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ФОРМУВАННЯ ОПЕРАЦІЇ СЕКВЕНТУВАННЯ

Засобами розширеної алгебри алгоритмів описано модель комп’ютерного рисування знаків операції горизонтального і вертикального секвентування. Мовою об’єктного програмування C# наведено код моделі знака операції горизонтального секвентування.

Ключові слова: модель, функційний унітерм, секвентуння, декомпозиція, підсистема.

By means of the extended algebra algorithms described model of a computer drawing operation marks the horizontal and vertical sequencing. Object language programming with C # code is an operator model horizontal sequencing.

Keywords: model, featured uniterm, sequenting, decomposition, subsystem.

Вступ і формулювання задачі

У дослідженні [1] описана модель декомпозиції інформаційної системи, призначеної для комп’ютеризації синтезу, редагування і оптимізації формул алгебри алгоритмів [2, 3, 4]. Алгебра алгоритмів утворена системою операцій секвентування, елімінування, паралелення, реверсування і опреаціями циклічних секвентування, елімінування та паралелення, які виконуються над абстрактними і функційними унітермами [5, 6]. Знаки операцій алгебри алгоритмів є специфічними і мають складну конфігурацію [7]. Систему знаків операцій алгебри алгоритмів можливо створити такими універсальними комп’ютерними системами, як Word, Coral Draw, Page maker та іншими. Однак використання універсальних комп’ютерних систем потребує багато затрат праці та комп’ютерного часу і тому є малоефективним [8]. Створені спеціалізовані комп’ютерні системи [8, 9] істотно підвищують рівень автоматизації процесів набору і редагування формул алгоритмів. Але цими системами не забезпечуються процеси комп’ютерної оптимізації формул алгоритмів. Описана на рівні підсистем модель [1] інформаційної системи передбачає можливості комп’ютерної оптимізації формул алгебри алгоритмів. Але у цій моделі не описано процеси інформаційної технології формування операції секвентування, що і є задачею, яка розв’язується у цьому дослідженні.

Декомпозиція моделі інформаційної технології

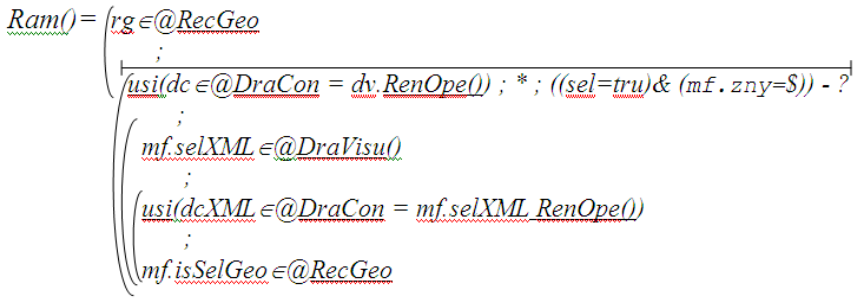
Модель інформаційної технології з метою зменшення складності її створення декомпозуємо на функційні унітерми. Модель декомпозиції описується такою формулою

$$pu\ ov\ Draw(mf \in @MaFor,\ f \in @Siz,\ bb \in @Bru,\ gb \in @Bru,\ sp \in @Pen,\ dp \in @Pen,\ x{:}y{:}mx{:}my \in @Dou) = \left(\begin{array}{l} Ram() \\ ; \\ IdeRoz() \\ ; \\ sepSiz \in @Siz \\ ; \\ RysSe() \end{array}\right.$$

де *Ram()* – унітерм рисування рамки вибору секвентування, *IdeRoz()* – ідентифікація розділювача унітермів операції секвентування, *sepSiz Î @Siz* – обчислення розмірів розділювача, *RysSe()* – рисування і видалення операцій секвентування.

Рисування рамки вибору

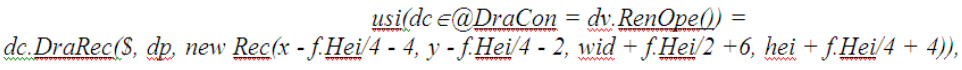
Рисування рамки вибору операції описується формулою



де *rg Î @RecGeo* – змінна стандартного типу *RecGeo*, який реалізується відомим класом `RectangleGeometry` [10, 11].

Формування умови рисування рамки вибору операції секвентування передбачає перевірку значень змінної вибору формули (*(sel=tru) – ?*), відсутність режимів знищення усієї формули (*mf.zny≠delAl*), першого (*mf.zny≠delFir*) і другого (*mf.zny≠delSec*) унітермів та вложеної (*mf.zny≠delChi*) формули, що описується умовою *(mf.zny=$)-?*, якою ідентифікується порожнє значення змінної *mf.zny*. Умовний унітерм матиме такий опис *((sel=tru)& (mf.zny=$) – ?*).

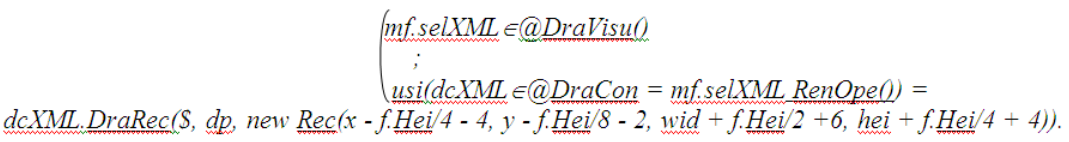
Для створення рамки вибору секвентування вводимо змінну *dv Î @DraVis* стандартного типу *DraVis*, який реалізується відомим класом `DrawingVisual` [10, 11]. Сам функційний унітерм рисування рамки вибору описується формулою



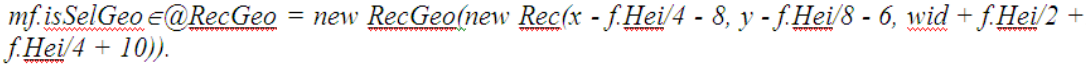
де *usi* – реалізується стандартним дескриптором `using` [10, 11], *DraCon* – підсистема, яка реалізується стандартним класом `DrawingContext` [10, 11], *RenOpe(),DraRec()* та *Rec()* – типові функційні унітерми, які реалізуються відомими процедурами `RenderOpen()`,`DrawRectangle()` та `Rect()`[10, 11], *Hei* – унітерм, який реалізується типовою властивістю `Height` [10, 11].

Висвітлення рамки вибору секвенції на екрані комп'ютера описується функційним унітермом *mf.canDra.AddVisu(dv)*, де *AddVisu()* – функційний унітерм, який реалізується відомою процедурою `AddVisual()`[10, 11].

Формули алгоритмів записуються у пам'ять комп'ютера і зчитуються з неї у вигляді *xml* – подібного формату даних. Для здіснення виділення формул, зчитуваних із пам'яті комп'ютера, вводимо такий функційний унітерм

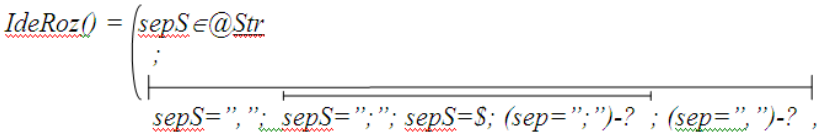


Для запамятовування рамки вибору вводимо функційний унітерм



Ідентифікація розділювача

Функційний унітерм ідентифікації розділювача операції секвентування



де *sepS Î @Str* – введення текстової змінної, *sep* – змінна, у якій зберігається значення розділювача унітермів операції секвентування.

Обчислювання розмірів розділювачів

Функційний унітерм обчислення розмірів розділювача унітермів

sepSiz ∈@Siz= new @Siz(TexLeng(sepS), TexHei(sepS)),

де *sepSiz Î @Siz* – означення змінної *sepSiz* типу підсистеми *Siz*, який реалізується стандартним класом `Size [10, 11]`, *TexLeng(sepS)* – функційний унітерм обчислення довжини розділювача унітермів, *TexHei(sepS)* – функційний унітерм обчислення висоти розділювача унітермів.

Обчислення довжини і ширини розділювача

pr (l ∈@Dou) TexLeng(tex ∈@Str)=F_Tex(tex ∈@Str).Wid,
pr (h ∈@Dou) TexHei(tex ∈@Str)=F_Tex(tex ∈@Str).Hei,

де *pr* – специфікатор доступу до функційного унітерма, який реалізується типом `private[10, 11]`, *l Î @Dou* – призначена для збереження дожини розділювача змінна *l* типу *Dou*, який реалізується стандартним класом `double [10, 11]`, *h* – змінна, призначена для збереження ширини розділювача, *tex Î @Str* – текстова змінна типу *Str*, який реалізується стандартним класом `string [10, 11]`, *Wid* – унітерм, використовуваний для обчислення довжини, який реалізується стандартною властивістю `Width [10, 11]`, *Hei* – унітерм використовуваний для обчислення висоти, який реалізується стандартною властивістю `Height [10, 11]`, *F_Tex(tex Î @Str)* – функційний унітерм нормалізації (врахування регіональних особливостей) тексту розділювача *tex*, який описується формулою

```
pr (ft∈@FormTex) F_Tex(t∈@Str) =(s∈@FontSt= @FontSts.Nor
                                ;
                                s= @FontSts.Nor
                                ;
                                tf∈@T_f =new @T_f(@MaFor.fF, s, @F_W.Ligh, @F_S.Medi)
                                ;
                                ft∈@FormTex=new @FormTex(t, Sy.Gl.@CuI.CurCul,
                                                         @FlDir.LeToRig, tf, @MaFor.f, @Bru.Bla)
                                ;
                                ft.TexAli  = @TexAli.Le,
```

у якій *ft Î @FormTex* – означення вихідної змінної *ft* типу *FormTex*, який реалізується стандартним класом `FormattedText [10, 11]`, *t Î @Str* – вхідна змінна *t* типу *Str*, *s Î @FontSt* – введення змінної *s* типу підсистеми *FontSt*, яка реалізується стандартним класом `FontStyle [10, 11]`, *= @FontSts.Nor* – приписування змінній стандартної властивості унітерму *Nor* підсистеми *FontSts*, яка реалізується властивістю `Normal` класу `FontStyles [10, 11]`, *s= @FontSts.Nor* – повторна нормалізація, *tf Î @T_f* – створення змінної типу підсистеми *T_f*, яка реалізується стандартним класом `Typeface [10, 11]`, *=new @T_f(@MaFor.fF, s, @F_W.Ligh, @F_S.Medi)* – приписування створеній змінній шрифтів *fF*, які означено у головній підсистемі *MaFor*, а також властивостей *Ligh* і *Medi*, які означено у підсистемах *F_W* і *F_S* та реалізовано у властивостях `Light` і `Medium` стандартних класів `FontWeights` і `FontStretches [10, 11]`, *ft Î @FormTex* – створення змінної *ft* підсистеми *FormTex*, яка реалізується стандартним класом `FormattedText [10, 11]`, *=new @FormTex(t, Sy.Gl.@CuI.CurCul, @FlDir.LeToRig, tf, @MaFor.fSiz, @Bru.Bla)* – приписування створеній змінній значення тексту *t*, який відформатований з врахуванням регіональних особливостей *CurCul*, орієнтації тексту *LeToRig*, розміру (кегля) шрифта *f* і кольору кісточки *Bla* (нею рисується текст), які означено у підсистемах *Sy.Gl.@CuI*, *FlDir*, *MaFor*, *Bru* та відповідно реалізуються властивостями `CurrentCulture, LeftToRight, Black` стандартних класів `System.Globalization.CultureInfo, FlowDirection, Black [10, 11]`, *ft.TexAli = @TexAli.Le* – опис лівої (*Le*) прив'язки (*TexAli*) відформатованого тексту з використанням унітерму підсистеми *TexAli*, яка реалізується класом `TextAlignment` з властивостями `TextAlignment` і `Left [10, 11]`.

Загальна модель операції секвентування

Перед виходом на на загальну модель необхідно перевірити, чи має місце режим видалення (*(mf.znyF=tr) – ?*) операцій секвентування. Ознакою цього є значення *tr* змінної *mf.znyF*. За

наявності цього режиму виконуватиметься алгоритм видалення секвентування *VydAl()*. Інакше вибирається загальна модель операціїї секвентування *M()*. Ці процеси описуються формулою

$$RysSe() = \overline{VydAl();\ M()\ ;\ (mf.znyF=tr)\ -\ ?}\ .$$

Загальна модель рисування знаків операції секвентування утворена розпізнаванням орієнтації (горизонтальна або вертикальна), заданням режимів видалення унітермів, врахуванням співвідношень між довжиною і шириною знака операції секвентування, заданням початкових координат знака операції, вибором функційних унітермів рисування знаків, вибором функційних унітермів введення операції на графічне поле їхнього відображення, обчисленням координат розташування унітермів і розділювача між ними та вибором методів рисування унітермів і розділювача.

Модель функційного унітерма наведена формулою (1), де *zS Î @DraVisu* – змінна *zS* типу підсистеми *DraVisu*, яка реалізується відомою підсистемою `DrawingVisual`, *(ori=Ori.Hor)-?* – перевірка, чи значення змінної орієнтації *ori* є горизонтальна орієнтація (*Ori.Hor*), *sH Î @Dou=Mat.Sqr(wid) + 2* – із значення довжини *wid* знаку операції обчислення його щирини, *usi (dc Î @DraCon=zS.RenOpe())* – створення змінної *dc* типу підсистеми *DraCon* і приписування їй змінної *zS* відкритої функційним унітермом *RenOpe()*, *po Î @Poi(x, y+sH)* – обчислення і приписування координат створеній змінній *po* типу підсистеми *Poi*, яка реалізована класом `Point [10, 11]`, *DraSe(mf, dc, po)* – функційний унітерм рисування горизонтального знаку операції секвентування, *mf.canDra.AdVis(zS)* – для висвітлення знака створеної операції горизонтального секвентування додавання *AdVis()* її до *canDra*, *x = x +f.Hei / 2* та *y = y + sH* – обчислення абсциси та ординати для вирисовування першого унітерму операції секвентування, *(tA±$) – ?* – перевірка наявності першого унітерму, *tA.Dra(mf, f, bb, gb, sp, dp, x, y, mx, my)* – вибір функційного унітерма рисування першого унітерма, *x=x + tA.wid* – додавання до абсциси довжини першого унітерма, *x = x + f.Hei / 2* – корекція абсциси на кегель шрифта унітерма, *po = new Poi(x, y)* – задання нових координат змінній *po*, *DraT_M(mf, po, sepS)* – функційний унітерм рисування розділювача унітермів операції секвентування, *x = x + sepS.Wid + f.Hei / 2* – зміна абсциси з врахуванням довжини розділювача і розміру кегля, *(tB±$) – ?* – перевірка наявності другого унітерма, *tB.Dra(mf, f, bb, gb, sp, dp, x, y, mx, my)* – вибір функційного унітерма рисування другого унітерма операції секвентування. У випадку невиконання умови u_2, виконуються унітерми *mf.zny=$* – присвоєння змінній *zny* значення *$*, а *mf.oCy=fa* – змінній *oCy* значення *fa*.

Якщо ж умова *(ori=Ori.Hor)-?* не виконується, то з елімінування за цією умовою вибирається формула *H()*, яка описує рисування знака операції вертикального секвентування та його унітермів (2) та є аналогічною формулі (1).

$$H() =$$

$$\overline{\begin{array}{l}
\left(\begin{array}{l} sW \in @Dou=Mat.Sqr(wid) + 2\ ; \\ ; \\ usi\ (dc \in @DraCon= zS.RenOpe()) = \left(\begin{array}{l} po \in @Poi(x+sW,\ y) \\ ; \\ DraSeW(mf,\ dc,\ po) \\ ; \\ mf.canDra.AdVis(zS) \end{array}\right. \end{array}\right. \left(\begin{array}{l} mf.zny=\$\ ;\ u_2\text{-?} \\ ; \\ mf.oCy=fal \end{array}\right. \\
;\ x = x + sW \\
;\ y = y + f.Hei / 2 \\
;\ \overline{\left(\begin{array}{l} tA.Dra(mf,\ f,\ bb,\ gb,\ sp,\ dp,\ x,\ y,\ mx,\ my);\ *\ ;\ (tA\pm\$) - ? \\ y=y + tA.hei\ ; \end{array}\right.} \\
\left(\begin{array}{l} x = x + f.Hei / 2\ ; \\ po = new\ Poi(x,\ y)\ ; \\ DraT_M(mf,\ po,\ sepS); \\ \left(\begin{array}{l} x = x + sepS.Wid + f.Hei / 2\ ; \\ tB.Dra(mf,\ f,\ bb,\ gb,\ sp,\ dp,\ x,\ y,\ mx,\ my);*;\ (tB\pm\$) - ? \end{array}\right. \end{array}\right.
\end{array}} \qquad (2)$$

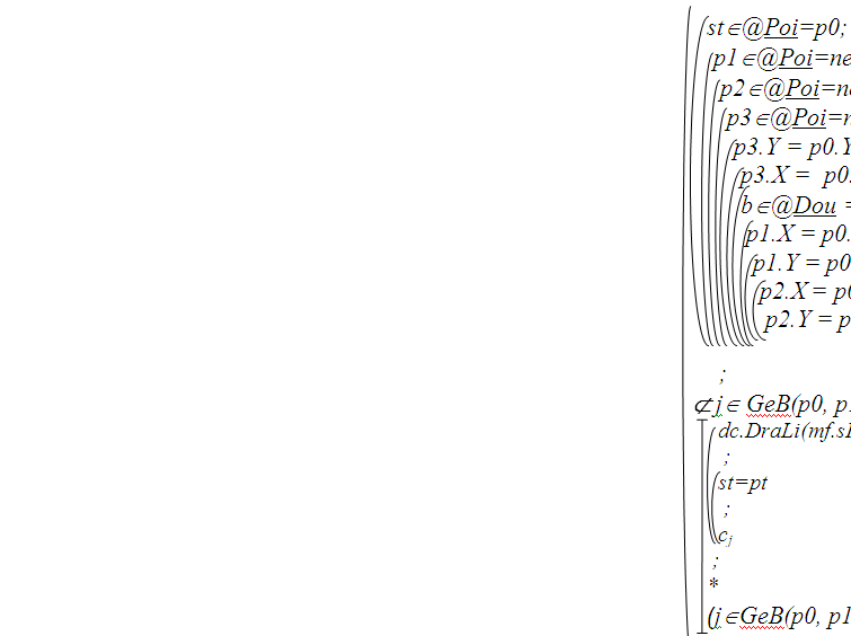
Модель рисування знака операції горизонтального секвентування

Задаємо для функційного унітерма специфікатор доступу ***pu*** і три вхідні параметри ***mf, dc, pt*** та вибір ним функційного унітерма ***DraBezW()***

pu DraSeH(mf∈@MaFor, dc∈@DraCo, pt∈@Poi) = DraBezH(mf, dc, new Poi(pt.X, pt.Y-1, wid),

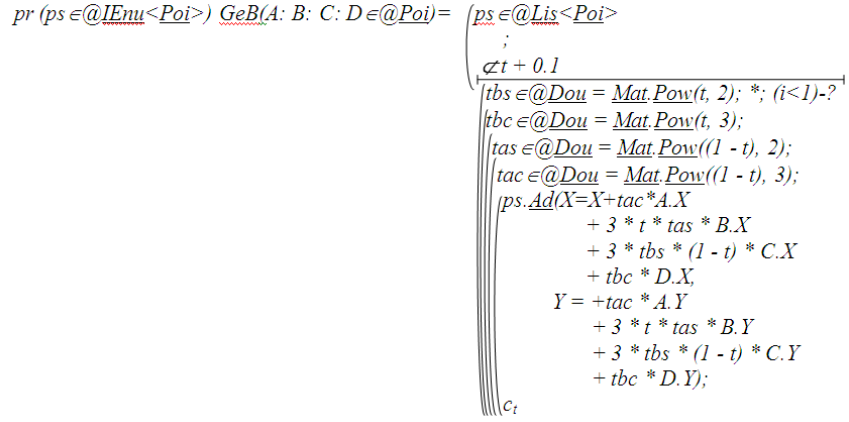
де ***pt.X, pt.Y-1*** – координати початку рисування знака операції секвентування, а ***wid*** – довжина цього знака.

Функційний унітерм ***DraBezW()*** використовуватимо тільки у підсистемі секвентування, тому задамо до нього метод обмеженого доступу ***pr***, який реалізується специфікатором `private` [10, 11] і опишемо формулою



де ***st, p1, p2, p3*** – змінні базових координат знака операції секвентування, ***b*** – змінна для збереження обчисленої ширини знака операції секвентування, ***j*** – змінна циклу кількість значень якої описується функційним унітермом ***GeB()***, ***DraLi()*** – функційний унітерм для опису використовування знака операції секвентування відрізками прямих ліній.

Для отримання списку поточних координат знака операції секвентування введено функційний унітерм ***GeB()***, який має чотири вхідні (***A, B, C, D***) і один вихідний (***ps***) параметри, ***ps Î @IEnu<Poi>*** – вихідний параметр типу підсистеми ***IEnu<Poi>***, яка реалізується відомим класом `IEnumerable<Point>` [10, 11], ***ps Î @Lis<Poi>*** – змінна типу підсистеми ***Lis<Poi>***, який реалізується відомим класом `List<Point>` [10, 11], ***tbs, tbc, tas, tac*** – змінні для збереження поточних значень для обчислення координат ліній, із яких утворюється знак операції секвентування.

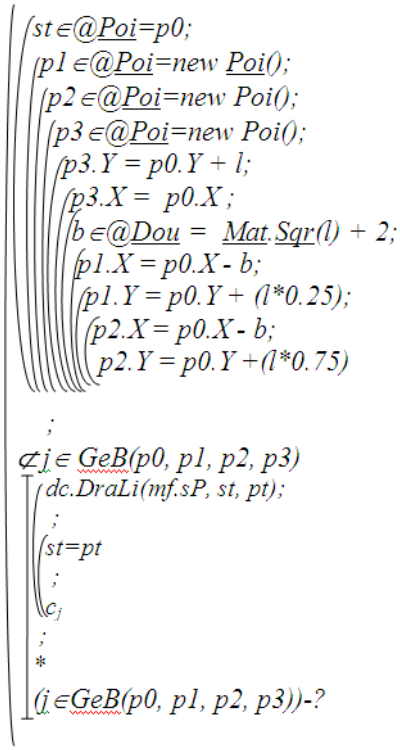


Модель рисування знака операції вертикального секвентування

Такими формулами, які аналогічні формулам опису створення знака операції горизонтального секвентування, описується модель вирисовування знака операції вертикального секвентування

pu DraSeW(mf∈@MaFor, dc∈@DraCo, pt∈@Poi) = DraBezW(mf, dc, new Poi(pt.X-1, pt.Y, hei),

pr DraBezW(mf∈@MaFor, dc∈@DraCo, p0∈@Poi, l∈@Dou)=



Код моделі операції горизонтального секвентування

Програма моделі мовою **C#** є такою:

```
        public void DrawSek(MainForm mf, DrawingContext dc, Point pt)
        {
            DrawBezier_M(mf, dc, new Point(pt.X, pt.Y – 1), width);
        }

        private void DrawBezier_M(MainForm mf, DrawingContext dc, Point
p0, double length)
        {
            Point start = p0;
            Point p1 = new Point(), p2 = new Point(), p3 = new Point();

            p3.Y = p0.Y;
            p3.X = p0.X + length;

            double b = Math.Sqrt(length) + 2;

            p1.X = p0.X + (length * 0.25);
            p1.Y = p0.Y – b;


            p2.X = p0.X + (length * 0.75);
            p2.Y = p0.Y – b;
```

```
            foreach (Point pt in GetBezierPoints(p0, p1, p2, p3))
            {
                dc.DrawLine(mf.solidPen, start, pt);

                    start = pt;
            }
        }


        private IEnumerable<Point> GetBezierPoints(Point A, Point B,
Point C, Point D)
        {
            List<Point> points = new List<Point>();

            for (double t = 0.0d; t <= 1.0; t += 1.0 /10)
            {
                double tbs = Math.Pow(t, 2);
                double tbc = Math.Pow(t, 3);
                double tas = Math.Pow((1 – t), 2);
                double tac = Math.Pow((1 – t), 3);

                points.Add(new Point
                {
                    X = +tac * A.X
                        + 3 * t * tas * B.X
                        + 3 * tbs * (1 – t) * C.X
                        + tbc * D.X,
                    Y = +tac * A.Y
                        + 3 * t * tas * B.Y
                        + 3 * tbs * (1 – t) * C.Y
                        + tbc * D.Y
                });
            }

            return points;
        }
```

Підсумки

Декомпозицією моделі на підмоделі зменшено складність побудови математичної моделі інформаційної технології комп'ютерного рисування знака операції секвентування.

Побудована математична модель описує рисування рамки вибору операції секвентування, ідентифікацію та обчислення розмірів розділювачів унітермів операції секвентування, обчислення розмірів унітермів операції секвентування і автоматичне рисування знаків горизонтального і вертикального секвентування унітермів.

Програмною реалізацією і апробацією виконано верифікацію побудованої математичної моделі, якою підтверджено опис математичною моделлю інформаційних процесів автоматичного вирисовування знаків операції секвентування і можливість комп'ютерної оптимізації секвенційних формул алгоритмів на підставі властивостей операції секвентування.

1.Овсяк О. Класи інформаційної системи генерування коду /О. Овсяк // Вісник Тернопільського державного технічного університету: "Тернопільський національний технічний університет імені Івана Пулюя". – 2010. – № 1. – С. 171 – 176. 2. Owsiak W., Owsiak A. Rozszerzenie algebry algorytmów /Pomiary, automatyka, kontrola. – № 2, 2010. – S. 184 – 188. 3. Owsiak W., Owsiak A., Owsiak J. Teoria algorytmów abstrakcyjnych i modelowanie matematyczne systemów informacyjnych. – Opole: Politechnika opolska, 2005. – 275 s. 4. Ovsyak V.K.: Computation Models and Algebra of

Algorithms. http://www.nbuv.gov.ua/Portal/natural/ VNULP/ISM/ 2008_621/01.pdf 5.Овсяк В. Алгоритми: аналіз методів, алгебра впорядкувань, моделі, моделювання. – Львів: Світ, 1996. – 132 с. 6. Овсяк В. Алгоритми: методи побудови, оптимізації, дослідження вірогідності. – Львів: Світ, 2001. – 160 с. 7. Овсяк О. Модель абстрактної підсистеми комп'ютерної інформаційної системи генерування коду /О. Овсяк //Компютерні науки та інформаційні технології. Вісник національного університету Львівська політехніка. – №686. 2010. – С.127 – 136. 8. 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" / В.М. Бритковський – Львів, 2003. – 18 с. 9. 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" / А.С. Василюк – Львів, 2008. – 20 с. 10. Petzold C. Programowanie Windows w języku C#. –Warszawa: „RM", 2002. – 1161 s. 11. Мэтью Мак-Дональд. Windows presentation foundation в .NET 3.5 с примерами на C# 2008. – М.–СПб.– К.: "Apress", 2008. – 922 с.

УДК 004.852; 004.94

П. Кравець
Національний університет "Львівська політехніка",
кафедра інформаційних систем та мереж

САМООРГАНІЗАЦІЯ СТРАТЕГІЙ СТОХАСТИЧНОЇ ГРИ НА ПРИКЛАДІ ПОБУДОВИ ЛАТИНСЬКИХ КВАДРАТІВ

Досліджується проблема ігрової самоорганізації розподіленої системи на прикладі розв'язування задачі побудови латинських квадратів. Суть самоорганізації полягає у перетворенні локально скоординованих стратегій гравців у глобальну координацію розв'язків стохастичної гри. Сформульовано ігрову задачу, розроблено метод та алгоритм її розв'язування, виконано комп'ютерне моделювання стохастичної гри для побудови ортогональних та звичайних латинських квадратів. Отримано та проаналізовано характеристики самоорганізації стохастичної гри.

Ключові слова: стохастична гра, координація стратегій, самоорганізація системи, латинський квадрат.

The problem of game self-organizing of the distributed system on an example of the problem solving of Latin squares construction is investigated. The essence of self-organizing consists in transformation of the locally-co-ordinated strategies of players to global coordination of stochastic game solution. The game problem is formulated, the method and algorithm of its solution are developed, computer modelling of stochastic game for construction of orthogonal and normal Latin squares is executed. Characteristics of self-organizing of stochastic game are received and analysed.

Keywords: stochastic game, coordination of strategies, self-organization of system, Latin square.

Вступ

Безкоаліаційна стохастична гра – це повторювальна математична гра в умовах невизначеності матриць виграшів, у якій гравці після незалежного випадкового вибору чистих стратегій на основі побудованого на динамічних змішаних стратегіях імовірнісного механізму отримують випадковий виграш (або програш), який використовують для адаптивної модифікації змішаних стратегій так, щоб максимізувати функції середніх виграшів (або мінімізувати функції середніх програшів).