УДК 681.3:519.68

Болгар Ю. О., Жежерун О. П.

ЗАСТОСУВАННЯ ГЕНЕТИЧНИХ АЛГОРИТМІВ ДЛЯ УНІФІКАЦІЇ ТЕРМІВ ДРУГОГО ПОРЯДКУ З ОДНОМІСНИМИ ФУНКЦІОНАЛЬНИМИ СИМВОЛАМИ

Задача уніфікації другого порядку у загальному випадку є нерозв'язною, хоча для окремої лямбда-алгебри термів другого порядку із одномісними функціональними символами її розв'язність доведено. Стандартний алгоритм уніфікації термів другого порядку має експоненційну складність, а генетичний алгоритм із використанням спеціальних метрик як функцій пристосованості є альтернативним способом ефективного знаходження уніфікатора.

Ключові слова: лямбда-алгебра, уніфікація другого порядку, генетичний алгоритм, відстань Дамерау-Левенштейна, відстань Джаро-Уінклера.

1. Загальні поняття лямбда-алгебри

Для уніфікації термів другого порядку необхідно розширити логіку першого порядку до типізованого λ-числення, що походить від простої теорії типів Чорча [2].

Нехай маємо множину елементарних типів T_0. Тоді множина типів T визначається як найменша множина, що містить T_0 та є замкненою відносно операції $\alpha,\beta \in T \Rightarrow (\alpha \to \beta) \in T$.

Терми у типізованому λ-численні будуються із атомів, операцій аплікації та абстракції.

Атоми. Якщо V – множина змінних, а C – множина констант, причомуV і C попарно не перетинаються, то множина атомів A визначається як $A = V \cup C$.

Аплікація. Якщо e_1 – терм типу $(\alpha \to \beta)$, а e_2 – терм типу α, то аплікація $(e_1 e_2)$ є термом типу β.

Абстракція. Якщо e – терм типу β, а x – змінна типу α, то абстракція $(\lambda x.e)$ є термом типу $(\alpha \to \beta)$. λ-абстракція – це визначення анонімної функції, що бере на вхід x, а підставляє його у вираз e. Абстракція лише задає функцію, а не викликає її.

Синтаксично правильно побудовані λ-терми визначаються індуктивно:

- змінна x сама по собі є λ-термом;
- якщо t – це λ-терм, а x – змінна, то $(\lambda x.t)$ є λ-термом (побудовано за допомогою абстракції);
- якщо t і s є λ-термами, то (ts) є λ-термом (побудовано за допомогою аплікації).

Нехай E – терм, а $(\lambda x.e)$ підтерм E.

Усі входження x в $(\lambda x.e)$ називаються **зв'язаними** в E.

Будь-які входження змінної в терм E, що не є зв'язаними, називаються **вільними**.

Позначимо множину змінних, що мають хоч одне вільне входження в терм E, як $F(E)$.

Нехай e – терм і тип змінної x дорівнює типу терму e. Тоді позначимо $\underline{E\langle x,e\rangle}$ терм, який отримано заміною всіх вільних входжень x в E на e.

Введемо відношення $\underline{R(x,y,E)}$: якщо для будь-якого терма e $(\lambda y.e)$ – це підтерм E, то всі входження x в e зв'язані в E, причому $\tau(x) = \tau(y)$.

λ-конверсія перейменовує зв'язану змінну в абстракції. Нехай $(\lambda x.e)$ – підтерм E. Якщо $y \notin F(E)$ і R(x,y,E), то $(\lambda x.e)$ може бути замінено в e на $(\lambda y.e)\langle x,y\rangle$.

β-редукція замінює формальний аргумент на фактичний в аплікації. Якщо для будь-якої змінної $y \in F(E)$ виконується $R(x,y,E)$, то підтерм E виду $(\lambda x.e')e$ може бути замінено на $e'x,e$.

η-конверсія. Якщо $x \notin F(e)$, то підтерм E виду $(\lambda x.ex)$ може бути замінено на e.

Підстановка у λ-численні – це скінченна множина підстановочних пар

$$\sigma = \{x_i, e_i \mid i \in [1..n]\} \quad \forall i,j \in [1..n]\ x_i = x_j \Rightarrow i = j.$$

Застосування підстановки σ до терму E визначається як нормальна форма терма $[\lambda x_1 \ldots x_n.E]$ $(e_1, e_2, .., e_n)$ і знаходиться індуктивно [3]:

$$\sigma(f(t_1,\ldots,t_n)) = f(\sigma(t_1),\ldots,\sigma(t_n));$$

$$\sigma(X) = \begin{cases} t, \textit{якщо пара } X \to t \textit{ є в підстановці } \sigma \\ X, \textit{в іншому випадку.} \end{cases};$$

$$\sigma(F(t_1,\ldots,t_n)) = \begin{cases} p(u), \textit{якщо пара } F \to \lambda x_1 \ldots \lambda x_n. \\ \quad \textit{и є в підстановці } \sigma \\ \textit{де } p = [x_1 \to \sigma(t_1)\ldots, x_n \to \sigma(t_n)] \\ F(\sigma(t_1),\ldots,\sigma(t_n)) \textit{ в іншому випадку.} \end{cases};$$

$$\sigma(\lambda x_1,\ldots,x_n.t) = \lambda x_1,\ldots,x_n.\sigma(t) \textit{покладаючи, що } x_i \notin Dom(\sigma).$$

Задача уніфікації у типізованому λ-численні формулюється так: маючи два λ-терми e_1 і e_2, знайти підстановку (уніфікатор) для вільних змінних з двох термів таку, що $\sigma(e_1)$ і $\sigma(e_2)$ є еквівалентними відповідно до правил конверсії.

У загальному випадку проблема уніфікації термів другого порядку є нерозв'язною [2], проте для задачі уніфікації λ-термів із одномісними функціональними символами доведено її розв'язність [1].

Наведемо приклад уніфікації двох термів з одномісними функціональними символами. Нехай маємо терми *e1=f(u(a)), e2=u(f(a)),* де *u* – змінна першого порядку із типом *t*, а *f* – другого порядку із типом $(t \to t)$.

Тоді уніфікатором цих двох термів буде підстановка $\theta = \langle \lambda x.u(x), f \rangle$

$$\theta(\mathrm{f}(\mathrm{u}(\mathrm{a}))) \to (\lambda x.u(x))u(a) \to u(u(a)) \leftarrow$$

$$\leftarrow u((\lambda x.u(x))a) \leftarrow \theta(\mathrm{u}(\mathrm{f}(\mathrm{a}))).$$

Для термів другого порядку поняття найбільшого загального уніфікатора втрачає сенс, адже терми можуть мати кілька підстановок, але жодну більш загальну, ніж інші. Отже, метою розробки алгоритму є отримання будь-якого уніфікатора з повної множини уніфікаторів.

2. Стандартний алгоритм уніфікації лямбда-термів

Розглянемо стандартний алгоритм приведення двох λ-термів t_1 та t_2 до одного вигляду за допомогою β-редукцій [5]. Маємо п'ять випадків:

1) якщо $t_1 = t_2$, то уніфікація більше не потрібна;

2) якщо Head(t_1)=Head(t_2), тобто два терми починаються однаково: $t_1=\lambda x.a(u)$, $t_2=\lambda x.a(v)$, тоді вони зводяться за допомогою редукції до виду

$$\theta(\lambda \mathrm{x.a(u)}) \to \lambda \mathrm{x.w} \leftarrow \theta(\lambda \mathrm{x.a(v)});$$

3) якщо маємо терми $t_1=\lambda x.y(x)$, $t_2=\lambda x.v'$, де *y* – змінна, а *v'* – терм, то $\theta = \left\langle \lambda \mathrm{z.t} \diagup \mathrm{y} \right\rangle$;

4) якщо підстановка проходить лише в голові одного з термів так, щоб після підстановки вони починались однаково, тоді маємо дві альтернативи: $t_1=fe_1$, $t_2=Fe_2$:

- використовуємо правило імітації $\theta = \langle f, \lambda u.Fhu \rangle$, згідно з яким проводиться заміна всіх входжень *f* на *Fh*;
- використовуємо правило проекції $\theta = \langle f, \lambda u.u \rangle$, що відповідає видаленню з ланцюжків усіх входжень змінної *f*;

5) підстановка відбувається одночасно у двох термах відповідно до таких часткових випадків:

- $t_1=fe_1$, $t_2=ge_2$ => $\theta = \{\langle f, \lambda u.a \rangle, \langle g, \lambda u.a \rangle\}$;
- $t_1=fe_1$, $t_2=x$ => $\theta = \{\langle f, \lambda u.a \rangle, \langle x, a \rangle\}$;
- $t_1=x$, $t_2=y$ => $\theta = \{\langle x, a \rangle, \langle y, a \rangle\}$;
- $t_1=fe_1$, $t_2=b$ => $\theta = \{\langle f, \lambda u.b \rangle\}$;
- $t_1=x$, $t_2=b$ => $\theta = \{\langle x, b \rangle\}$.

Стандартний алгоритм має експоненційну складність та є напіврозв'язним, адже у випадку, якщо терми не є уніфікованими, то він зациклюється [2].

3. Обрання представлення розв'язків

В якості однієї хромосоми використано підстановку, тобто відповідність між наявними у термах змінними та їхніми можливими значеннями-термами. Для обмеження пошукового простору без втрати повноти розв'язку терми беруться із спеціально згенерованого модифікованого універсуму Ербрана та кодуються цілими числами.

Нехай H_0 – множина констант, що зустрічаються в множині диз'юнктів *S*. Якщо *S* не містить жодної константи, то припускаємо, що H_0 складається з однієї довільної константи $H_0 = \{a\}$. Нехай H_{i+1} – це об'єднання H_i і множини всіх термів виду $f^n(t_1,\ldots,t_n)$ (при всіх *n*) для всіх функцій f^n, що зустрічаються в множині диз'юнктів *S*, де $t_j\,(j = 1..n)$ належить H_i. Тоді кожне H_i називається множиною констант *i*-го рівня для *S*, а H_∞ називається ербранівським універсумом для множини *S* [4].

Для випадку термів другого порядку модифікований універсум Ербрана будується таким чином. Замість констант першого порядку використовуємо константи другого порядку. На першому рівні універсуму використовуємо всі наявні в нашій мові константи другого рівня, адже в процесі уніфікації з'являється потреба у доданні нових констант, окрім початкових. На другому рівні будуємо всі двоелементні об'єднання констант з першого рівня, на третьому – трьохелементні і так далі. Нарешті, до кожного з елементів отриманої множини застосуємо змінні другого порядку, внаслідок чого побудований нами модифікований універсум Ербрана складатиметься з елементів, що представляють праву частину λ-аплікацій у підстановці: $f \leftarrow \lambda x.\underline{FG..Hf}x$.

4. Обрання функції пристосованості

Мірою пристосованості хромосоми має бути оцінка якості обраної підстановки, тобто результат порівняння між собою початкових термів після застосування до них даної підстановки.

В даній роботі проаналізовано два варіанти функції пристосованості: відстань Дамерау-Левенштейна та відстань Джаро-Уінклера.

Відстань Дамерау-Левенштейна між двома рядками позначає мінімальну кількість операції вставки одного символу, видалення одного символу, заміни одного символу на інший або перестановки двох послідовних символів, необхідних для перетворення одного рядка на інший.

Проблемою звичайної відстані Дамерау-Левенштейна є висока часова складність: O(M*N*max(M,N)), де M, N – довжини рядків. А оскільки швидкість обрахунку пристосованості є ключовою для швидкодії генетичного алгоритму, то в алгоритмі застосовано спрощений варіант відстані Дамерау-Левенштейна, що носить назву оптимального вирівнювання рядків (optimal string alignment). Основна відмінність від самої відстані Дамерау-Левенштейна полягає в тому, що оптимальне вирівнювання рядків накладає одне обмеження: будь-який підрядок не може бути редагований більш ніж один раз. Унаслідок цього обмеження, алгоритм можна звести до складності O(M*N).

Щоб знайти відстань оптимального редагування рядків, використаємо алгоритм Вагнера-Фішера. Побудуємо матрицю $D[M,N]$, де M і N – довжини порівнюваних рядків, а $D(i,j)$ – відстань Дамерау-Левенштейна між *i* символами першого рядка, та *j* символами другого рядка за такою формулою:

$$D(i,j)=\begin{cases}\min\begin{pmatrix}A,\\ D(i-2,j-2)+transposeCost\end{pmatrix};\\ ;i>1,j>1,S[i]=T[j-1],S[i-1]=T[j]\\ A;\ в\ іншому\ випадку\end{cases}.$$

$$A=\begin{cases}0;i=0,j=0\\ i;j=0,i>0\\ j;i=0,j>0\\ D(i-1,j-1);S[i]=T[j]\\ \min\begin{pmatrix}цццццц1)+\\ D(i-1,j)+deleteCost,\\ D(i-1,j-1)+replaceCost\end{pmatrix};j>0,i>0,S[i]\neq T[j]\end{cases}$$

Вага операцій видалення, вставки, перестановки та заміни – однакова:
insertCost=deleteCost=replaceCost=transposeCost=1.

Відстань Джаро-Уінклера

Відстань Джаро-Уінклера ґрунтується на кількості схожих символів та їхньому порядку, співставленні та перестановці символів. Вона не є метрикою в математичному сенсі, адже не задовольняє властивість трикутника. Обчислювальна складність відстані Джаро-Уінклера – O (M*N). Відстань є нормалізованою: 1 відповідає точному співпадінню, а 0 – абсолютно різним рядкам. Для сумісності із використанням відстані Дамерау-Левенштейна, кінцеве значення відстані Джаро-Уінклера обертаємо так, щоб 0 відповідав за співпадіння рядків.

Відстань Джаро d_j між двома рядками s_1 та s_2 обчислюється таким чином:

$$d_j=\begin{cases}0, & якщо\ m=0\\ \frac{1}{3}\left(\frac{m}{|s_1|}+\frac{m}{|s_2|}+\frac{m-t}{m}\right) & в\ іншому\ випадку\end{cases},$$

де *m* – кількість символів, що співпадають, *t* – половина кількості транспозицій.

Символи називаються такими, що співпадають, якщо вони є однаковими та розташовані на відстані не більш ніж $\frac{\max(|s_1|,|s_2|)}{2}-1$.

Кількість транспозицій обчислюється так: для всіх символів, що співпадають, з першого рядка сумується кількість символів, що співпадають, з другого рядка, проте знаходяться на інших позиціях. Потім отримана сума ділиться на два.

Уінклер додав модифікацію до алгоритму Джаро, роблячи більш схожими ті рядки, що мають однакові префікси. Тоді відстань Джаро-Уінклера d_w обчислюється так:

$$d_w=d_j+\left(lp\left(1-d_j\right)\right),$$

де l – довжина спільного префікса (до 4 символів), p – сталий масштабуючий коефіцієнт (зазвичай 0.1).

Відстань Джаро-Уінклера варто використовувати для рядків невеликої довжини, що якраз підходить до умов задачі. Врахування однакових префіксів також має краще відображати ідею уніфікації термів, адже важливо, щоб терми співпадали від самого початку. Виходячи із опису обраних функцій пристосованості, генетичний алгоритм буде розв'язувати задачу мінімізації.

5. Обрання кінцевої умови

Оскільки глобальні мінімуми функцій пристосованості нам наперед відомі, то протягом тестування алгоритму застосовано два підходи до визначення критерію зупинки:

– у першому випадку кінцем алгоритму стала збіжність популяції у заданому глобальному мінімумі, що дорівнює нулю для обох функцій пристосованості (FC1);

– у другому випадку просимульовано реальну ситуацію використання генетичного алгоритму, коли глобальний мінімум не відомо. Отже, критерієм зупинки алгоритму стала відсутність покращення пристосованості популяції або досягнення заданої максимальної кількості поколінь (FC2).

Алгоритм вважається завершеним успішно, якщо пристосованість найкращої знайденої хромосоми повністю відповідає глобальному мінімуму функції пристосованості. Можливість похибки у даній задачі виключається, адже алгоритм або знаходить уніфікатор, або ні.

6. Опис генетичного алгоритму

1. Первинний аналіз вхідних термів, виділення змінних другого порядку.

Відповідно до теорії типізованої лямбда-алгебри, вхідні терми мають задовольняти таким умовам:

– всередині запису терму не повинні зустрічатися змінні або константи першого порядку, інакше порушиться типізація;

– аналогічно, терми повинні закінчуватися на константу або змінну першого порядку;

– якщо терми починаються із констант другого рівня, то вони мають бути ідентичними, адже інакше не можливо буде провести уніфікацію.

2. Побудова модифікованого універсуму Ербрана заданої глибини. Глибина залежить від розмірності пошукового простору (кількості змінних).

3. Створення початкової популяції розміру N та заповнення неї елементами з ербранівського універсаму.

4. Оцінюємо популяцію за допомогою обраної функції пристосованості (Дамерау-Левенштейна або Джаро-Уінклера).

5. Турнірним відбором набираємо хромосоми у батьківський пул.

6. Випадковим чином обираємо хромосоми для схрещування із батьківського пулу.

7. Із заданою ймовірністю p_c проводимо доточкове схрещування хромосом. Нащадків додаємо до поточної популяції.

8. До нащадків застосовуємо із заданою ймовірністю p_m щільнісну мутацію: мутований ген змінює своє значення на випадково обраний елемент ербранівського універсуму.

9. Сортуємо популяцію, що складається із батьків та нащадків, за пристосованістю.

10. Обираємо N найкращих хромосом.

11. Якщо досягнуто кінцевої умови, то кінець алгоритму. Інакше, переходимо на крок 4.

7. Параметри алгоритму

Реалізований генетичний алгоритм має такі параметри:

– кількість хромосом: 50;

– ймовірність кросоверу: 0,7;

– ймовірність мутації: 0,2;

– максимальна кількість поколінь: 100;

– максимальна кількість поколінь без покращення середньої пристосованості: 15;

– кількість прогонів: 1000.

У роботі досліджено ефективність алгоритму для задачі із розмірністю 2, 4 та 6, що відповідає сумарній кількості змінних другого порядку у вхідних термах. Тестування розробленого алгоритму проходило шляхом знаходження таких усереднених характеристик протягом багатьох прогонів:

– середня кількість поколінь, які потребував алгоритм для досягнення умови зупинки (avGenNum);

– середня доля успіху;

– відсоток успішних прогонів (successRate);

– середній рівень пристосованості популяції (avFitness);

– середнє відхилення найкращої знайденої пристосованості від реального значення (avDelta).

8. Аналіз ефективності алгоритму із різною конфігурацією

Результати алгоритму для розмірності 2.

Проаналізуємо результати, отримані для роботи генетичного алгоритму на двовимірній задачі, тобто уніфікації термів із двома змінними другого порядку. Таблиця містить зазначені вище характеристики для використання в якості функцій пристосованості відстані Дамерау-Левенштейна (DL), нормалізованої відстані Дамерау-Левенштейна (NDL) та відстані Джаро-Уінклера (JW).

Розглянемо спочатку випадок із першою кінцевою умовою (FC1), описаний в таблиці 1. Згідно зі встановленою умовою закінчення алгоритму, доля успіху завжди становила 100 %, адже алгоритм працював доти, поки не знайде уніфікатор термів. Аналогічно середня пристосованість популяції та середнє відхилення дорівнюють нулю. Щодо кількості поколінь, які знадобились алгоритму для досягнення результату, то, як бачимо, найкращий показник продемонструвала ненормалізована відстань Дамерау-Левенштейна, а найгірший – її нормалізований варіант.

Таблиця 1. **Результати алгоритму для розмірності 2 для першої кінцевої умови**

	FC1			
	avGenNum	successRate,%	avFitness	avDelta,%
DL	24,51	100	0	0
NDL	58,255	100	0	0
JW	27,41	100	0	0

Другий спосіб задання кінцевої умови продемонстровано у таблиці 2, що найшвидше збігається популяція при використанні нормалізованої відстані Дамерау-Левенштейна, але вона має найгіршу долю успіху та найбільшу похибку у знайдених результатах. Найближче до реального мінімуму наближається алгоритм, що використовує відстань Джаро-Уінклера. Незважаючи на те, що його доля успіху трохи гірша, ніж у Дамерау-Левенштейна, середнє відхилення знайденого результату – у сім разів менше.

Таблиця 2. **Результати алгоритму для розмірності 2 для другої кінцевої умови**

	FC2			
	avGenNum	successRate,%	avFitness	avDelta,%
DL	24,369	61,8	0,385	7
NDL	23,895	52,2	0,077	7,4
JW	25,039	59,9	0,011	1,1

Результати алгоритму для розмірності 4.

Розглянемо тепер результати, які показав реалізований алгоритм для задачі уніфікації термів із чотирма змінними другого порядку. Як бачимо, загальна динаміка показників не відрізняється від двовимірного випадку. Помітне лише зростання середньої кількості поколінь, які проходить генетичний алгоритм до зупинки.

При використанні першого варіанта кінцевої умови найгірший показник середньої кількості поколінь показала у таблиці 3 відстань Джаро-Уінклера, а найкращий – ненормалізована відстань Дамерау-Левенштейна.

Таблиця 3. **Результати алгоритму для розмірності 4 для першої кінцевої умови**

	FC1			
	avGenNum	successRate,%	avFitness	avDelta,%
DL	31,016	100	0	0
NDL	43,871	100	0	0
JW	82,841	100	0	0

Для другого варіанта умови закінчення алгоритму, описаного у таблиці 4, середня кількість поколінь, за які збігається популяція, є приблизно однаковим для всіх трьох варіантів функції пристосованості. Щодо долі успіху, то для цієї конфігурації найкращий показник, як і у двовимірному випадку, показала ненормалізована функція Дамерау-Левенштейна. А от найменшу похибку у знайдених результатах продемонструвала відстань Джаро-Уінклера.

Таблиця 4. **Результати алгоритму для розмірності 4 для другої кінцевої умови**

	FC2			
	avGenNum	successRate,%	avFitness	avDelta,%
DL	27,879	66	0,363	5,9
NDL	27,045	49,2	0,064	6,2
JW	26,912	46	0,016	1,5

Результати алгоритму для розмірності 6.

Для задачі із розмірністю 6 змінено параметри алгоритму та збільшено розмір популяції для урізноманітнення її генофонду і попередження передчасної збіжності. Отже, подальше тестування проходило для 100 хромосом у популяції.

Отже, для першого варіанта кінцевої умови бачимо у таблиці 5, що алгоритму із використанням відстані Джаро-Уінклера знадобилося втричі більше поколінь для досягнення глобального екстремуму, аніж алгоритму із відстанню Дамерау-Левенштейна.

Таблиця 5. **Результати алгоритму для розмірності 6 для першої кінцевої умови**

	FC1			
	avGenNum	successRate,%	avFitness	avDelta,%
DL	36,88	100	0	0
NDL	53,206	100	0	0
JW	138,01	100	0	0

Для другого варіанта умови закінчення алгоритму, результати різних конфігурацій, наведені у таблиці 6, аналогічні до попередніх випадків із 2- та 4-розмірним пошуковим простором: найуспішнішим є застосування відстані Дамерау-Левенштейна, а найточнішим – Джаро-Уінклера.

Таблиця 6. **Результати алгоритму для розмірності 6 для другої кінцевої умови**

	FC2			
	avGenNum	successRate,%	avFitness	avDelta,%
DL	48,487	75,6	0,251	2,7
NDL	47,79	63,1	0,031	3,1
JW	49,728	38,7	0,012	1,2

Узагальнюючи всі проведені випробовування реалізованого алгоритму, для визначення того, яку функцію пристосованості доцільніше використовувати у генетичному алгоритмі, що уніфікує терми другого порядку із одномісними функціональними символами, необхідно оцінити розмірність пошукового простору. Відтак, відстань Джаро-Уінклера показала погані результати для випадку із шістьма змінними, адже однією із особливостей відстані є її застосування для порівняння коротких за довжиною рядків. Отже, при збільшенні довжини результуючих термів відстань Джаро-Уінклера використовувати недоцільно.

Для випадку, коли генетичний алгоритм працює до досягнення наперед заданого глобального екстремуму, найкраще себе показало застосування відстані Дамерау-Левенштейна. Для задач будь-якої розмірності алгоритм із такою конфігурацією використав найменшу кількість поколінь.

Що стосується симуляції реальної роботи генетичного алгоритму, коли наперед не відомо глобальний екстремум функції пристосованості, то всі варіанти обчислення відстані між результуючими термами показали однакову середню кількість поколінь, що проходили до збіжності популяції. Щодо показників долі успіху та середньої похибки, то функції показали протилежні результати. Успішнішим є застосування відстані Дамерау-Левенштейна, але точніші значення уніфікаторів знаходив алгоритм із використанням відстані Джаро-Уінклера.

Список літератури

1. Жежерун А. П. Разрешимость проблемы унификации для языков второго порядка с одноместными функциональными символами / А. П. Жежерун // Кибернетика. – 1979. – № 5. – С. 120–125.
2. Huet G. P. A Unification Algorithm for Typed λ-Calculus / G. P. Huet // Theoretical Computer Science. – 1975. – №. 1. – P. 27–57.
3. Levy J. Simplifying the Signature in Second-Order Unification / J. Levy, M. Villaret // Journal: Applicable Algebra in Engineering Communications and Computing. – 2009. – Vol. 20, №. 5–6. – P. 427–445.
4. Mordechai Ben-Ari. Mathematical Logic for Computer Science. – Third Edition, 2012. – P. 341.
5. Snyder W. H. Hinger order unification revisited: complete sets transformations / W. Snyder, J. Gallier // Dep. of Compand Inform. Science University at Pensilvania. – 2004. – P. 1–48.

I. Bolgar, O. Zhezherun

THE APPLICATION OF GENETIC ALGORITHMS FOR UNIFICATION OF SECOND-ORDER TERMS WITH UNARY FUNCTIONAL SYMBOLS

Second-order unification problem is undecidable in general case, but the decidability has been proven for lambda-algebra of second-order terms with unary functional symbols. The standard algorithm of second-order unification has an exponential complexity, while genetic algorithm with the use of special metrics as fitness function is an alternative and effective way of finding a unificator.

Keywords: lambda-algebra, second-order unification, genetic algorithm, Damarau-Levenshtein distance, Jaro-Winkler distance.

Матеріал надійшов 13.05.2014