УДК 004.7 **doi: 10.26906/SUNZ.2018.2.087**

А.С. Свиридов

Харківський національний університет радіоелектроніки, Харків

МЕТОД ВИЗНАЧЕННЯ ОЗНАК ПРОГРАМНИХ РЕАЛІЗАЦІЙ АЛГОРИТМІВ НА ОСНОВІ СИНТАКСИЧНОГО АНАЛІЗУ

Розглянуто існуючі підходи та методи виявлення ознак. Запропоновано використання синтаксичного аналізу в парі з семантикою. Проведено аналіз особливостей, що виникають при об'єднанні алгоритмів, описано можливість приналежності алгоритмів різних груп та призначень. Запропонований метод дозволяє визначати ознаки програмної реалізації алгоритмів. Використання отриманих ознак дозволяє групувати схожі за ознаками алгоритми.

Ключові слова: *метод, ознака, синтаксичний аналізатор, семантика, програмний код.*

Вступ

До існуючих видів запису алгоритмів можна віднести такі як словесний, формульно-словесний, графічний і описаний на формальній алгоритмічній мові. За такої необхідності, такі способи записів, як словесний, формульно-словесний і графічний перетворюються в спосіб, описаний на формальній алгоритмічній мові. Незалежно від способу запису алгоритмів їх можна представити у вигляді наступних основних схем: лінійні, розгалужувальні, а також циклічні алгоритми. Лінійні алгоритми являють собою пряму послідовність операцій або дій. Розгалужувальні алгоритми не мають прямої послідовності дій, через те, що на одному або декількох етапах необхідно провести операцію вибору, оскільки для кожного вибору існує своє продовження алгоритму. Циклічний алгоритм є ускладненою версією розгалужувального і лінійного алгоритмів, оскільки, наприклад, при невиконанні деякої умови необхідно повернуться на кілька операцій вище (в початок циклу).

В даний час всі алгоритми, які в подальшому будуть реалізовані за допомогою комп'ютерної техніки, описуються за допомогою мов програмування. При описі алгоритму на мові програмування використовуються, так звані, функції. Функція, як і сам алгоритм, виконується послідовно, що, в свою чергу, дозволяє провести деяку аналогію з описом лінійних алгоритмів, виконання яких проводиться послідовно. Мови програмування для реалізації деяких дій використовують цикли та розгалуження. Застосування послідовного виконання команд-дій, використання циклів і елементів розгалуження дозволяє зробити висновок про те, що, функція, яка описана на мові програмування, є алгоритмом представленим до цього в словесному або графічному вигляді.

Існує величезна кількість різних способів подання алгоритмів. Залежно від уявлення таких алгоритмів вони можуть використовуватися за допомогою різних засобів. В даній статті розглядаються алгоритми, реалізацію яких можна представити у вигляді програмного коду [1]. Оскільки в сучасному світі більшість підприємств, заводів та виробництв використовують сучасну техніку, на даний момент подання алгоритмів у вигляді програмного коду є самим поширеному в світі [2]. Таке широке розподілення реалізацій призвело до того, що існує величезна кількість аналогічних алгоритмів, при цьому іноді розробникам простіше реалізувати алгоритм «з нуля» ніж шукати його реалізацію. Навіть за умови, що розробник знайшов схожий на необхідний алгоритм, дуже часто у нього не вистачає інформації для підтвердження своєї думки про цей алгоритм. Відсутність деякої стандартизації і єдиної системи з пошуку алгоритму зводиться до того що глобальна мережа все далі і далі наповнюється не завжди коректними і правильними алгоритмами [3].

Постановка задачі

Існуючі алгоритми найчастіше представлені у вигляді наборів, які представляють собою збірку алгоритмів, націлених найчастіше на вирішення задач в одній області. Такі збірки алгоритмів, наприклад, представлені у вигляді бібліотек програмних реалізацій алгоритмів, не використовують автоматизований процес наповнення таких наборів. Відсутність автоматизації наповнення таких збірок може призводити до появи копій алгоритмів, а також мати реалізацію одного і того ж самого алгоритму з різним ступенем синтаксичної наповненості. Відсутність додаткової інформації про алгоритми може призвести до того, що, коли потрібно буде обрати один алгоритм із набору схожих за призначенням алгоритмів, відсутність аргументованого вибору серед цих алгоритмів може привести до реалізації більш трудомісткого алгоритму замість менш трудомісткого, за умови того що результати застосування алгоритмів будуть однаковими.

Реалізація синтаксичного аналізатора дозволяє отримувати структуровані дані (структурувати дані), шляхом аналізу тексту/коду/даних. При застосуванні структурного аналізатора для алгоритмів або файлу, що містить набори алгоритмів у вигляді результату, можна отримати структуру алгоритму. За допомогою цієї структури даних можна виявити додаткові параметри, обчислити складність алгоритмів, а також пе-

ревірити на наявність копій серед наборів алгоритмів. Таким чином, завдання автоматичного виявлення наборів ознак в даний час є актуальним.

Мета статті - розробка методу, заснованого на синтаксичному аналізі програмної реалізації алгоритму, який дозволить виявляти ознаки алгоритмів.

Метод визначення ознак

Метод, що пропонується, включає в себе три основні блоки: аналіз вхідних алгоритмів; синтаксичний аналіз коду; виділення ознак. Перший блок є вхідним блоком до методу, та основна його задача полягає в перевірці кількості алгоритмів на вході, а також аналіз мови програмування. Другим блоком є блок синтаксичного аналізу коду; в ньому вперше використано синтаксичний аналізатор, задачею якого є представити данні алгоритму у вигляду структурної інформації. На наступному блоку, завдяки семантиці, структурні данні аналізуються та шукаються ознаки цього алгоритму. Блок-схема методу представлена на рис. 1.

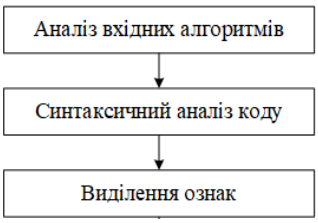

Рис. 1. Схема методу визначення ознак програмних реалізацій алгоритмів на основі синтаксичного аналізу

Нижче наведено детальний опис кожного з блоків запропонованого методу.

Аналіз вхідних алгоритмів. Програмна реалізація алгоритму є кодом, написаним на мові програмування, що реалізує поставлену задачу.

Інформація, що надходить на вхід системи аналізу алгоритмів, може представлятися у вигляді одного алгоритму або у вигляді набору алгоритмів, наприклад, у вигляді бібліотеки, яка підключається для використання функцій (алгоритмів).

Програмні реалізації алгоритмів найчастіше бувають двох типів. Перший тип представляється у вигляді, алгоритму, що не має на даний момент свого аналога або такий алгоритм, який є модифікованою (поліпшеною) версією (частиною) класичного алгоритму. Відмінною особливістю такого типу алгоритмів є те, що програмний код може мати недоліки:

- бути не оптимізованим;
- не завжди реалізовувати поставлену задачу;
- бути не алгоритмом цілком, а частковою реалізацією класичного алгоритму [4].

При цьому структурною особливістю таких алгоритмів є представлення таких алгоритмів не у вигляді готової реалізованої бібліотеки (набору функцій), а окремою версією алгоритму в єдиному екземплярі. Варто також відзначити, що, навіть за умови наявності деяких недоліків, такі реалізації, з точки зору складності алгоритму, можуть виявитися простішими за класичні алгоритми, модифікованою (поліпшеною) версією яких вони є.

Другий тип – це алгоритми, які знаходяться в деяких базах, найчастіше представлених у вигляді готових бібліотек. Такі бібліотеки є збіркою (набором) класичних алгоритмів. Такий тип характеризується наявністю опису, також найчастіше відповідністю конкретній тематиці. Наприклад, математична бібліотека може містити алгебраїчні або геометричні реалізації функцій, а бібліотека для попередньої обробки зображень включає набір алгоритмів, за допомогою яких в залежності від вхідного зображення можна буде використовувати її різні функції.

На вхід методу можна подавати як перший тип алгоритмів, так і другий, представлений у вигляді бібліотеки. Алгоритм, що подається на вхід системи, найчастіше написаний на одній з найпопулярніших мов програмування в залежності від призначення даного алгоритму. При цьому деякі алгоритми можуть бути написані на різних мовах, хоча по суті можуть робити одні і ті ж операції. Наприклад, як згадувалося вище, однією з таких може бути бібліотека математичних функцій. Така бібліотека представлена на різних мовах програмування, оскільки математичні функції можуть використовуватися, як, наприклад, в задачах аналізу відеофайлу, так й при реалізації тих чи інших додатків. Таким чином, наступним кроком є визначення мови програмування, на якій було написано код, що реалізує цей алгоритм або набір алгоритмів.

Після визначення мови програмування, далі необхідно провести синтаксичний аналіз цього коду для виявлення ознак алгоритму. Однак, на даному етапі, за умови, що на вхід методу подавалася нова бібліотека, необхідно аналізувати кожен алгоритм окремо, що знаходиться всередині бібліотеки.

Відповідно, необхідно провести додатковий синтаксичний аналіз, заснований на пошуку ключових слів мови, на якій було написано бібліотеку. Ключові слова в мові програмування є зарезервованими словами, за допомогою яких компілятор може знайти початок і кінець функцій, змінних і т.д. Ґрунтуючись на роботі компілятора, за допомогою додаткового синтаксичного аналізу можна здійснити поділ бібліотеки на окремі функції (алгоритми). За умови, що на вхід методу подавалась не бібліотека, а один алгоритм, то немає необхідності проводити додатковий синтаксичний аналіз, який використовується для бібліотеки. Відповідно, на першому етапі відбувається аналіз введеної інформації. У вигляді результату додаткового синтаксичного аналізу отримуються структуровані дані алгоритмів бібліотек.

Синтаксичний аналіз коду. Синтаксичний аналізатор повинен розпізнати структуру поданого на вхід методу алгоритму, а саме синтаксичні залежності ключових слів. В результаті має бути або побудовано синтаксичне дерево, або виявлені складові. Зазвичай граматика будується так, щоб на виході будувалося синтаксичне дерево, що дозволяє виконувати

різноманітні трансформації лексичного змісту з перепогодження залежних ключових слів, а також легко виділяти семантику, зокрема – застосовувати алгоритм зважування альтернативних варіантів побудови дерева [5]. Пропозиція може допускати кілька альтернативних варіантів зв'язування ключових слів. В цьому випадку аналізатор намагається застосовувати деякі евристики і базу знань, але може в кінці повернути кілька варіантів синтаксичного дерева. До такої поведінки відноситься синтаксичний аналіз знизу, оскільки він сильно обмежений в ресурсах по припиненню експоненціального зростання числа варіантів [6]. Наявність декількох підсумкових варіантів розбору може означати не тільки недостатній набір правил у мовній моделі мов програмування, але і властиву даній пропозиції неоднозначність, яка усувається тільки з урахуванням більш широкого контексту.

Аналізовані блоки коду можуть мати різну складність, включати невідому типізацію даних або відступ від нормативного синтаксису. Щоб ефективно справлятися з різними задачами, синтаксичний аналізатор застосовує кілька різних алгоритмів, включаючи низхідний синтаксичний аналіз і синтаксичний аналіз знизу, а також він застосовує семантичний аналіз для уточнення результатів у випадку коли програмний код має недоліки, що ускладнюють читання. Низхідний синтаксичний аналіз є досить точним. Синтаксичний аналіз знизу працює досить швидко і здатний проаналізувати навіть дуже довгі блоки коду та ігнорувати незрозумілі фрагменти коду.

Низхідний синтаксичний аналіз, або аналіз через синтез, починається з припущення про структуру блоку коду, а потім уточнює і деталізує цей код, опускаючись на рівень конкретних ключових слів/змінних. Іншими словами, цей алгоритм ініціює розбір з початкового нетерміналу S – тобто усі слова в блоці коду виявляються пов'язані в єдину структуру.

Синтаксичний аналіз знизу починає розбір з конкретних ключових слів, пов'язуючи до них спочатку пари тип-змінна, потім приєднує до цих пар нові доповнення або інші пов'язані пари. Поступово процес зв'язування доходить до початкового нетерміналу S – тобто усі слова в пропозиції виявляються пов'язані в єдину структуру. З одного боку, завдання структурного синтаксичного аналізатора – визначити синтаксичну структуру алгоритму. Інакше кажучи, він розпізнає синтаксично коректні (або майже коректні) блоки коду. Результатом розпізнавання є не тільки факт визнання коректності блоків коду з точки зору синтаксису мови програмування. Синтаксичний аналізатор також видає інформацію щодо синтаксичних взаємин ключових слів/змінних і визначає деякі ознаки. З іншого боку, алгоритм розпізнавання синтаксису спроектований так, що він породжує по набору заданих правил можливі блоки і, зіставляючи їх з наявними, підтверджує одне зі своїх припущень, розуміючи синтаксис та значення коду.

Такий аналіз проводиться з метою створення структури даних на основі алгоритму, що дозволить в подальшому використовувати отримані дані для виділення ознак даного алгоритму.

Виділення ознак. Інформація, що отримана після етапу синтаксичного аналізу програмного коду, представляється у вигляді деякої структури даних, яку необхідно обробити для виділення ознак. Виділення ознак буде відбуватися шляхом використання семантики. Семантика уявляє з себе формалізацію значень конструкцій мов програмування за допомогою побудови їх формальних математичних моделей. За допомогою семантики будуть розглядатися основні структурні елементи функцій. Такими елементами є: змінні (локальні і глобальні); цикли; розгалуження. Необхідність використання семантики, а також основне призначення даного процесу полягає в тому, що мови програмування дозволяють описувати однакові за змістом операції різними способами. Наприклад, на рис. 2 представлено дві реалізації програмного коду.

```
i=0; while(i<5){i++;}
```

```
i=0; do{i++;}while(i<=4);
```

Рис. 2. Програмні реалізація частки коду завдяки ключовому слову while та парі слов do-while

Зрозуміло, що ці два фрагменти коду виконують одне і те ж, результати їх роботи ідентичні.

Це може призводити, по-перше, до підвищення складності реалізації функції, якщо розглядати з точки зору оптимізації часу виконання алгоритмів, по-друге до появи більш складних конструкцій, які є аналогами вже існуючих функцій, наприклад, в бібліотеці. При цьому, в залежності від мови програмування на якій буде описано алгоритм, різні способи реалізації однієї і тієї ж частини коду можуть займати різний час виконання. В залежності від мови, на якій описано алгоритм, будуть застосовуватися різні семантичні підходи. Основними підходами є такі семантики: аксіоматична; денотаційна; інтерпретаційна; трансляційна; трансформаційна [7 – 9].

Реалізація семантичного підходу виглядає наступним чином (однак, в залежності від складності алгоритму, реалізація може змінюватись). Спочатку відбувається обчислення кількості послідовних дій, кількість внутрішніх процедур і умов (розгалужень).

Першим кроком на етапі виділення ознак є пошук аналогічних алгоритмів з бази або бібліотеки. Цей крок необхідний для того, щоб виявити семантичні збіги на основі яких можна буде виділити ознаки, в базі знаходиться алгоритм, який за конструкцією схожий з алгоритмом, що було подано на вхід методу. Після цього здійснюється перевірка на однотипність коду. Якщо інформація про алгоритм, який поступив на вхід метода, є досить схожою на інформацію алгоритму з бази, на основі відмінностей та схожості семантичних конструкцій робиться висновок чи необхідно додавати до групи ознак з бази ознаку отриману з нового алгоритму. Якщо знайдена ознака структурно схожа з вже наявною в базі групою з описом, то відбувається додавання цієї ознаки до знайденої групи ознак. Ознаки збираються в групи ознак.

На етапі порівняння схожого по структурі алгоритму, що знаходиться в базі та алгоритму, поданого на вхід методу, відбувається наступне. Здійснюється тотожне порівняння локальних і глобальних змінних, порівняння операторів розгалуження, циклів на основі семантичного підходу, що, в свою чергу, дозволяє визначити чи є поданий на вхід методу алгоритм (для початку) аналогом алгоритму з бази, за умови, що кількість послідовних дій, циклів і розгалужень є ідентичною, але при цьому локальні змінні та/або глобальні змінні не відповідають один одному то відбувається порівняння типів даних цих змінних. За умови, якщо, наприклад, в функцію передається значення з плаваючою комою, а в іншу функцію передається цілочисельні значення, а не значення, що може приймати лише «true» або «false», то можна зробити висновок про те, що перша функція є модифікованою/покращеною версією другої функції. Такий аналіз функції дозволяє не тільки порівнювати функції, а також перетворювати часто повторювані елементи функцій до деяких блоків даних, на основі яких можна робити модифікації існуючих функцій, замінюючи складні блоки коду простішими еквівалентами, які виконують ті ж самі дії. Такі блоки кодів, залежно від їх складності, можуть бути показником або відмінною рисою функцій/алгоритмів, які, у свою чергу, є відмінною рисою алгоритму, який можна виділити.

Висновки

Запропоновано метод, який дозволяє виявляти ознаки програмних реалізацій алгоритмів. Виявлення ознак програмної реалізації алгоритмів в значній мірі дозволяє автоматизувати процес створення баз алгоритмів. В даний час такий процес не є повністю автоматизованим, що призводить до того, що алгоритми, які використовуються для реалізації завдань можуть бути некоректними або час, витрачений на створення такого нового алгоритму є виправданим. Автоматичний процес виявлення ознак з алгоритму в майбутньому може дозволити реалізувати концепцію автоматичного підбору алгоритмів під конкретну задачу, що дозволить скоротити час, витрачений на реалізацію таких алгоритмів до мінімуму. А застосування синтаксичного аналізу може дозволити знаходити аналогічні алгоритми, що в перспективі дозволить позбутися некоректних копій алгоритмів.

Напрямки подальших досліджень буде направлено на модернізацію даного методу з метою можливості додавання класифікації алгоритмів на основі розуміння ознаки самого алгоритму і групи ознак, що мають спільні характерні особливості. Така класифікація у вигляді наявності стандартизованих і згрупованих алгоритмів дозволить підбирати оптимальні алгоритми на основі існуючих даних для певних завдань.

Список літератури

1. A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "Cnn features off-the-shelf: an astounding baseline for recognition," in Proc. of the IEEE Conf. on Comp. Vision and Pattern Recognition Workshops, 2014, pp. 806–813

2. Альфред В. Ахо, Теория синтаксического анализа, перевода и компиляции. Т. 2, ,2012, 487 с.

3. Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн, Алгоритмы. Построение и анализ, Книга по требованию,2016, стр 1328

4. A. Kumar and so on "Ask me anything: Dynamic memory networks for natural language processing," CoRR, abs/1506.07285, 2015

5. Джесси Рассел, Синтаксический анализ, Книга по требованию,2013, 250 с.

6. K. Cho, B. Van Merrienboer, C. Gulcehre, D. Bahdanau, ¨ F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," arXiv preprint arXiv:1406.1078, 2014.

7. J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation." in EMNLP, vol. 14, 2014, pp. 1532–1543.

8. M. Denil, A. Demiraj, N. Kalchbrenner, P. Blunsom, and N. de Freitas, "Modelling, visualizing and summarising documents with a single convolutional neural network," 26th Int. Conf. on Computational Linguistics, pp. 1601–1612, 2014

9. P. D. Turney and P. Pantel, "From frequency to meaning: Vector space models of semantics," Journal of artificial intelligence research, vol. 37, pp. 141–188, 2010

Надійшла до редколегії 22.02.2017

Рецензент: д-р техн. наук, доц. М.А. Павленко, Харківський національний університет Повітряних Сил імені Івана Кожедуба, Харків.

МЕТОД ОПРЕДЕЛЕНИЯ ПРИЗНАКОВ ПРОГРАММНЫХ РЕАЛИЗАЦИИ АЛГОРИТМОВ НА ОСНОВЕ СИНТАКСИЧЕСКОГО АНАЛИЗА

А.С. Свиридов

Рассмотрены существующие подходы и методы выявления признаков. Предложено использование синтаксического анализа в паре с семантикой. Проведен анализ особенностей, возникающих при объединении алгоритмов, описана возможность принадлежности алгоритмов различных групп и назначений. Предложенный метод позволяет определять признаки программной реализации алгоритмов. Использование полученных признаков позволяет группировать схожие по признакам алгоритмы.

Ключевые слова: *метод, признак, синтаксический анализатор, семантика, программный код.*

METHOD OF SYMBOLS DETERMINING BY PARSER FOR SOFTWARE-BASED ALGORITHMS

A.S. Svyrydov

Existing approaches and methods of detecting signs are considered. The use of syntactic analysis in conjunction with semantics is proposed. The analysis of the features that arise when combining algorithms is carried out, the possibility of belonging to the algorithms of different groups and appointments is described. The proposed method allows to determine the signs of program algorithms realization. The use of the obtained features allows grouping similar algorithms.

Keywords: *method, sign, parser, semantics, program code.*