

Верификация неструктурированных программ методом индуктивных утверждений

Ф.М. Пучков¹

¹Московский государственный университет им. М.В. Ломоносова, Институт проблем информационной безопасности, fedormex@gmail.com

Важная с начала создания компьютерных программ задача верификации (аудита) программного кода со временем становится все более актуальной. Наряду с другими традиционными причинами, это связано с активным использованием свободного программного обеспечения в составе подлежащих защите автоматизированных систем. В настоящее время существует большое количество теоретических подходов к верификации императивных программ [1-3]. Большая часть из них применима для анализа структурированных программ, то есть программ, состоящих из простых операторов, условных операторов (if), а также операторов цикла с простым условием выхода (while, for). Тем не менее, в реальных программах количество неструктурированных циклов в реальных программах достаточно велико и может составлять до 40% от общего количества циклов [4]. Например, неструктурированными являются циклы, содержащие в своем теле операторы break, return или continue, а также циклы, содержащие побочное действие в условии выхода. В таких условиях, неприменимы методы, подобные описанным в работе [3].

```
int endsWith(const char* str, const
char* suffix) {
    int i = strlen(suffix);
    int j = strlen(str) - i;
    if(j < 0) return 0;
    while (--i >= 0)
        if(suffix[i] != str[i+j])
            return 0;
    return 1;
}
```

Неструктурированной программой будем называть программу, заданную в виде блок-схемы — ориентированного графа, вершинам которого соответствуют операторы программы, а ребра определяют порядок выполнения указанных операторов. В графе должна быть выделена начальная и конечная вершина; любой оператор дол-

жен быть достижим из начального, и находиться на ориентированном пути в конечную вершину.

Разработанный метод верификации программ состоит в построении в каждой точке (вершине) программы условий, называемых далее инвариантами, от значений переменных программы и состоянии ее памяти, которые должны быть истинны всякий раз, когда программа проходит через данную точку. Базовый алгоритм построения инвариантов может быть представлен, как процесс последовательного обхода инструкций программы и генерирования в них инвариантов на основе ранее полученных инвариантов с учетом семантики проходимых инструкций, а также с использованием априорно заданных индуктивных утверждений (гипотез). При этом важным является порядок обхода вершин управляющего графа. В докладе будет представлен новый алгоритм обхода вершин, исследованы его свойства и проведено сравнение с другими известными алгоритмами.

В работе будет также предложен способ проверки истинности индуктивных гипотез (инвариантов цикла).

Литература

- [1] Floyd R. W. Assigning meanings to Programs. // Proc. Symposium in Appl. Math. Vol. 19. Mathematical Aspects of Computer Science. AMS, Providence, P. 19-32., 1967.
- [2] Cousot P., Cousot R. Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints // In Proc. POPL'77, P. 238-252. ACM Press, 1977.
- [3] Dahl O.J., Dijkstra E.W., Hoare C.A.R. Structured Programming. Eds.: Academic Press Ltd., 1972, pp. 234.
- [4] Chalin P. Adjusted Verification Rules for Loops Are More Complete and Give Better Diagnostics for Less Dependable Software Research Group, Department of Computer Science and Software Engineering, Concordia University, Montréal, Canada.