Министерство образования и науки Российской Федерации
Московский физико-технический институт (государственный университет)

Физтех-школа прикладной математики и информатики
Кафедра дискретной математики

Выпускная квалификационная работа бакалавра по направлению 010900
«Прикладные математика и информатика»

Формальная верификация программ и их асимптотик

Студент 7910б группы
Григорянц С. А.

Научный руководитель
Дашков Е. В.

Долгопрудный
2021

Содержание

Глава 1

Введение

Сегодня, все больше отраслей компьютерных технологий нуждаются в формальной верификации. В первую очередь, в этот список входят программы, связанные с транспортом, коммуникацией, медициной, компьютерной безопасностью, криптографией и банковским делом. В этих критических сферах, последствия отсутствия формальной верификации в некоторых проектах привели к очень плачевным последствиям [2]. Фундаментальным отличием формальной верификации от классического тестирования ПО заключается в том, что, в то время как классическое тестирование проверяет систему лишь на каком-то подмножестве возможных входов, формальная верификация ПО позволяет утверждать о корректности системы на всех возможных входах. Такого вида гарантии конечно же дают несравнимо большую уверенность в корректности работы данного ПО, даже учитвая то, что остальные части системы, такие как компилятор, аппаратное обеспечение, и, в конце концов, само ПО, используемое для верификации, могут дать сбои. Ибо мы таким образом убеждаемся в том, что сам код, который является частью ПО, ошибок не содержит. Можно привести в пример много проектов, использующих формальную верификацию для довольно сложных систем. Например, в верификации компиляторов – проект Jinja [4, 5], проект Verisoft [6, 9], а также проект CompCert [1, 7]. В блокчейне – проект Scilla [8]. Таким образом, мы можем убедиться в том, что формальная верификация ПО и вправду очень востребованна.

Но наряду с верификацией программ, также встает вопрос верификации асимптотики применяемого алгоритма. Действительно, довольно часто мы хотим убедиться не только в том, что алгоритм корректен, но также и в том, что его асимптотика соответствует нашим ожиданиям. На самом деле, баги, связанные с асимтотикой алгоритма, могут возникать довольно часто только для конкретных входов, что делает классический подход тестирования неприемлемым для их отыскания. например, рассмотрим следующую реализацию бинарного поиска (на языке Python).

Рис. 1. Проблемный бинарный поиск

```
# Requires t to be a sorted array of integers.
# Returns k such that i <= k < j and t[k] = v
# or -1 if there is no such k.
def bsearch(t, v, i, j):
    if j <= i:
        return -1
    k = i + (j-i) // 2
    if v == t[k]:
        return k
    elif v < t[k]:
        return bsearch(t, v, i, k)
    return bsearch(t, v, i+1, j)
```

Проблема этого кода состоит в том, что при попадании в правую часть списка, вместо того, чтобы рассматривать интервал $[k+1; j)$, мы рассматриваем интервал $[i+1; j)$. Конечно, сам алгоритм корректно реализует бинарный поиск, но асимптотика при вводе, скажем, последнего элемента массива, превращается из логарифмической в линейную. На этом примере хорошо видно, что даже формальной верификации алгоритма и классического тестирования его на проверку асимптотики работы не всегда гарантирует нам корректность этой самой асимптотики. В связи с этим, возникает потребность в том, чтобы иметь также возможность формально верифицировать не только корректность алгоритма, но и его асимптотику.

В данной работе мы покажем, как формальная верификация алгоритмов и их асимптотик может быть реализована с помощью Сепарационной Логики с временными кредитами [3]. С помощью этой теории, мы формально верифицируем алгоритм нахождения наибольшей общей подпоследовательности.

Список литературы

[1] Xavier Leroy et al. *The CompCert verified compiler.* URL: http://compcert.inria.fr/.

[2] N. Dershowitz. *SOFTWARE HORROR STORIES.* URL: https://www.cs.tau.ac.il/~nachumd/verify/horror.html.

[3] Armaël Guéneau, Arthur Charguéraud, François Pottier. "A Fistful of Dollars: Formalizing Asymptotic Complexity Claims via Deductive Program Verification". *Programming Languages and Systems.* под ред. Amal Ahmed. Cham: Springer International Publishing, 2018, с. 533—560. ISBN: 978-3-319-89884-1.

[4] Gerwin Klein, Tobias Nipkow. "A Machine-Checked Model for a Java-like Language, Virtual Machine, and Compiler". *ACM Trans. Program. Lang. Syst.* **28** 4 (июль 2006), с. 619—695. ISSN: 0164-0925. DOI: 10.1145/1146809.1146811. URL: https://doi.org/10.1145/1146809.1146811.

[5] Gerwin Klein, Tobias Nipkow. "Verified Bytecode Verifiers". *TCS* **298** (2003), с. 583—626.

[6] Dirk Leinenbach, Wolfgang Paul, Elena Petrova. "Towards the Formal Verification of a C0 Compiler: Code Generation and Implementation Correctnes". *Proceedings of the Third IEEE International Conference on Software Engineering and Formal Methods.* SEFM '05. USA: IEEE Computer Society, 2005, с. 2—12. ISBN: 0769524354. DOI: 10.1109/SEFM.2005.51. URL: https://doi.org/10.1109/SEFM.2005.51.

[7] Xavier Leroy. "Formal Certification of a Compiler Back-End or: Programming a Compiler with a Proof Assistant". *SIGPLAN Not.* **41** 1 (янв. 2006), с. 42—54. ISSN: 0362-1340. DOI: 10.1145/1111320.1111042. URL: https://doi.org/10.1145/1111320.1111042.

[8] Ilya Sergey, Amrit Kumar, Aquinas Hobor. *Scilla: a Smart Contract Intermediate-Level LAnguage.* 2018. arXiv: 1801.00687 [cs.PL].

[9] Martin Strecker. *Compiler Verification for C0 (intermediate report).*

Благодарности

Благодарности идут тут.