

ОБЗОР МЕТОДОВ СТАТИЧЕСКОГО АНАЛИЗА ПРОГРАММНОГО КОДА

Азаренко А. В.

Факультет информационных технологий и управления, Белорусский государственный университет информатики и радиоэлектроники
Минск, Республика Беларусь
E-mail: lehaazarenko@mail.ru

Рассматривается возможность оптимизации и повышения защищенности приложений посредством статического анализа программного кода.

ВВЕДЕНИЕ

Статический анализ можно рассматривать как автоматизированный процесс обзора кода. Существенный недостаток методологии мануального обзора кода, это крайне высокая цена, поэтому компромиссным решением являются инструменты статического анализа кода, которые постоянно обрабатывают исходные тексты программ и выдают программисту рекомендации обратить повышенное внимание на определенные участки кода. Инструменты статического анализа кода используют комбинации методов статического анализа программного кода.

I. ОСНОВНЫЕ ЗАДАЧИ И ПРИЕМУЩЕСТВА

В целом задачи, решаемые с помощью методов статического анализа кода можно разделить на три категории:

1. Выявление ошибок в программном коде;
2. Рекомендации по оформлению программного кода. Некоторые платформы статического анализа позволяют проверять программный код на соответствие заданному стандарту оформления программного кода;
3. Подсчет метрик, позволяющих получить численное значение того или иного свойства программного обеспечения.

Главное преимущество статического анализ программного кода состоит в возможности существенного снижения стоимости устранения дефектов в программном обеспечении. Чем раньше ошибка выявлена, тем меньше стоимость ее исправления.

Применение методов статического анализа программного кода позволяет выявить большое число ошибок на этапе конструирования, или написания программного кода, что существенно снижает стоимость разработки всего проекта.

Статический анализ программного кода обладает рядом других преимуществ. Одним из них является полное покрытие кода, что позволяет находить дефекты и уязвимости в обработчиках редких ситуаций, мануальное тестирование которых затруднено по тем или иным причинам. Статический анализ является независимым от среды использования и комплятора, что

в свою очередь позволяет обнаруживать ошибки, которые могут проявиться в процессе длительного использования ПО.

II. ОБЗОР МЕТОДОВ НА ПРИМЕРЕ ИНТЕРВАЛЬНОГО АНАЛИЗА

Интервальный анализ вычисляет для каждой целочисленной переменной верхнюю и нижнюю границы для ее возможных значений. Интервалы являются интересными результатами анализа, поскольку эти результаты могут использоваться для оптимизации и обнаружения ошибок, связанных с проверкой границ массива, числовых переполнений и представления целых чисел. Данный случай включает решётку бесконечной высоты, и мы должны использовать специальный метод для обеспечения сходимости к фиксированной точке [1].

Решетка, описывающая одно абстрактное значение, определяется следующим образом:

$$Intervals = lift(\{[l, h] \mid l, h \in N \wedge l \leq h\}), \quad (1)$$

где

$$N = \{-\infty, \dots, -2, -1, 0, 1, 2, \dots, +\infty\} \quad (2)$$

это множество целых чисел с бесконечным количеством конечных точек и порядком интервалов, определенных путем включения:

$$[l_1, h_1] \sqsubseteq [l_2, h_2] \Leftrightarrow l_2 \leq l_1 \wedge h_1 \leq h_2 \quad (3)$$

Эта решётка не имеет конечной высоты, поскольку она содержит, к примеру, следующую бесконечную цепь:

$$[0, 0] \sqsubseteq [0, 1] \sqsubseteq [0, 2] \sqsubseteq [0, 3] \sqsubseteq [0, 4] \sqsubseteq [0, 5] \dots \quad (4)$$

Это переносится на решетку абстрактных состояний

$$States = Vars \rightarrow Intervals \quad (5)$$

До указания правил ограничения необходимо определить функцию eval, которая выполняет абстрактную оценку выражений:

$$eval(\sigma, X) = \sigma(X) \quad (6)$$

$$eval(\sigma, I) = [I, I] \quad (7)$$

$$eval(\sigma, input) = [-\infty, +\infty] \quad (8)$$

$$\begin{aligned} & eval(\sigma, E_1 \text{ op } E_2) \\ = & \widehat{op}(eval(\sigma, E_1), eval(\sigma, E_2)) \end{aligned} \quad (9)$$

Абстрактные арифметические операторы определяются как:

$$\begin{aligned} \widehat{op}([l_1, h_1], [l_2, h_2]) = & \left[\min_{x \in [l_1, h_1], y \in [l_2, h_2]} x \right. \\ & \left. \text{op } y \quad \max_{x \in [l_1, h_1], y \in [l_2, h_2]} x \text{ op } y \right] \end{aligned} \quad (10)$$

Например:

$$\widehat{+}([1, 10], [-5, 7]) = [1 - 5, 10 + 7] = [-4, 17] \quad (11)$$

Функция JOIN является обычной для форвардного анализа:

$$JOIN(v) = \prod_{w \in pred(v)} [w] \quad (12)$$

Теперь мы можем указать правило ограничения для присвоений:

$$\begin{aligned} X = E : [v] = & JOIN(v)[X \\ \mapsto & eval(JOIN(v), E)] \end{aligned} \quad (13)$$

Для остальных узлов ограничение является тривиальным:

$$[v] = JOIN(v) \quad (14)$$

Теорема Клини о неподвижной точке утверждает следующее: всегда существует уникальное наименьшее приемлемое решение для такого анализа. Однако, так как интервальный анализ использует решетку с бесконечной высотой, теорема неподвижной точки не применима. В таком случае возникает вопрос: имеют ли ограничение интервального анализа наименьшее приемлемое решение для любой программы? Ответ положительный.

Один из способов определить тот факт, что наименьшая неподвижная точка существует, является более продвинутый вариант теоремы о неподвижной точке, который основывается на преобразовании итерационных последовательностей и выполнение без предположения о конечной высоте. Другим способом является использование другой теоремы о неподвижной точке, также известной, как теорема Тарского о неподвижной точке: в конечной решетке L каждая монотонная функция $f : L \rightarrow L$ имеет уникальную наименьшую неподвижную точку, полученную из $lfp(f) = \prod \{x \in L | f(x) \sqsubseteq x\}$.

Доказательство довольно просто. Допустим $D = \{x \in L | f(x) \sqsubseteq x\}$ и $d = \prod D$. Сначала покажем, что d это неподвижная точка f , то есть

$f(d) = d$. Предположим $x \in D$. Тогда $d \sqsubseteq x$, потому что d это нижняя граница D . Из монотонности f получаем $f(d) \sqsubseteq f(x)$ и $f(x) \sqsubseteq x$, так как $x \in D$. Таким образом, $f(d) \sqsubseteq x$, $f(d)$ также нижняя граница D . Так как d это наибольшая нижняя граница D , получаем $f(d) \sqsubseteq d$. Из монотонности f получем $f(f(d)) \sqsubseteq f(d)$. Из этого следует, что $f(d) \in D$ и так как d является нижней границей D , получаем $d \sqsubseteq f(d)$. Из антисимметрии \sqsubseteq мы получаем $f(d) = d$. Чтобы убедиться в том, что d это уникальная наименьшая неподвижная точка f , предположим, что d' это какая-либо неподвижная точка f , то есть $f(d') = d'$. Тогда $d' \in D$ и, так как d это нижняя граница D , мы получаем $d \sqsubseteq d'$, а из антисимметрии \sqsubseteq следует тот факт, что наименьшая неподвижная точка уникальна.

Из-за того, что ограничение функций для интервального анализа монотонно, данная теорема свидетельствует о том, что наименьшая неподвижная точка существует. Таким образом, ограничения для интервального анализа всегда имеют четко определенное наиболее точное решение. Тем не менее, мы не можем вычислить наименьшую неподвижную точку, используя алгоритмы фиксации точки. Интервальный анализ решетки имеет бесконечную высоту, так что применение алгоритмов фиксации точки может никогда не заканчиваться: для решетки L_n последовательность приближений

$$f^i(\perp, \dots, \perp_n) \quad (15)$$

никогда не должна сходиться. Мощным инструментом для обхода подобного рода задачи является применение методов расширения и сужения.

III. Выводы

Метод интервального анализа используется для решения проблемы неопределенного потока мощности в централизованном кластере, так что в интервал включаются все решения, что, в свою очередь, позволяет покрыть все возможные варианты использования. Однако стоит отметить тот факт, что ни один из методов статического анализа программного кода не является ультимативным, поэтому выбор того или иного метода статического анализа программного кода зависит от большого числа факторов таких, как среда разработки и применения, архитектурные особенности продукта и т.д.

СПИСОК ЛИТЕРАТУРЫ

1. Андрес Меллер и Махаэль И. Шварцбах, Статический анализ программ / А. Меллер, М. И. Шварцбах // 2019. – С. 75-77.