Задания по курсу «Статический анализ программ»

5 июня 2024

Алексей Трифонов

Лекция 2

Что будет, если в нашу систему ввести тип Bool?

Так как появляются типы `true` и `false`, которые до этого использовались как `1` и `0` соответственно, добавляются и изменяются следующие правила:

$$E_1 == E_2 : [\![E_1]\!] = [\![E_2]\!] \wedge [\![E_1 == E_2]\!] = \mathrm{bool}$$
$$E_1 > E_2 : [\![E_1]\!] = [\![E_2]\!] = \mathrm{int} \wedge [\![E_1 > E_2]\!] = \mathrm{bool}$$
$$E_1 \texttt{ op } E_2 : [\![E_1]\!] = [\![E_2]\!] = [\![E_1 \texttt{ op } E_2]\!] = \mathrm{int}$$
$$\mathrm{output}\ E : [\![E]\!] = \alpha$$
$$\mathrm{if}\ (E)S : [\![E]\!] = \mathrm{bool}$$
$$\mathrm{if}\ (E)S_1\ \mathrm{else}\ S_2 : [\![E]\!] = \mathrm{bool}$$
$$\mathrm{while}\ (E)S : [\![E]\!] = \mathrm{bool}$$

Precision не изменяется, так как система остается sound. Recall — ухудшится, так как в новой реализации больше нельзя будет использовать результаты выражений которые ранее возвращались как `int`.

Что будет, если в нашу систему ввести тип Array?

Синтаксис (arr – $T[]$, T – $[\![\mathrm{value}]\!]$, idx – int):

```
arr[idx] = x;
y = arr[idx];
arr = {};
arr = {2, 4, 8};
```

Правила типизации:

$$E[E_{\mathrm{idx}}] : [\![E]\!] = \alpha[] \wedge [\![E[E_{\mathrm{idx}}]]\!] = \mathrm{int} \wedge [\![E[E_{\mathrm{idx}}]]\!] = \alpha$$
$$E[E_{\mathrm{idx}}] = E_{\mathrm{val}} : [\![E]\!] = \alpha[] \wedge [\![E_{\mathrm{idx}}]\!] = \mathrm{int} \wedge [\![E[E_{\mathrm{val}}]]\!] = \alpha$$
$$\{\} : [\![\{\}]\!] = \alpha[]$$
$$\{E_1, E_2, ..., E_n\} : [\![\{E_1, E_2, ..., E_n\}]\!] = [\![E_1]\!] = [\![E_2]\!] = ... = [\![E_n]\!] \wedge [\![\{E_1, E_2, ..., E_n\}]\!] = [\![E_1]\!][]$$

Типизируйте следующую программу

```
main() {
   var x, y, z, t;               //
   x = {2, 4, 8, 16, 32, 64};    // [|x|] = [|{2, 4, 8, 16, 32, 64}|] ∧ [|2|] = [|4|]
= [|8|] = ... = [|64|]
   y = x[x[3]];                  // [|y|] = [|x[x[3]]|] = alpha_x ∧ [|x|] = alpha_x[]
∧ [|x[3]|] = int ∧ [|x|] = alpha_x'[] ∧ [|3|] = int
   z = {{}, x};                  // [|z|] = [|{{},x}|] ∧ [|x|] = [|{}|]
   t = z[1];                     // [|t|] = [|z[1]|] = alpha_z ∧ [|z|] = alpha_z[] ∧
[|1|] = int
```

```
    t[2] = y;                   // [|t|] = alpha_t[] ∧ [|2|] = int ∧ [|y|] = alpha_t
}
```

$$
\begin{aligned}
[\![x]\!] &= \text{int}[] \\
[\![y]\!] &= \text{int} \\
[\![z]\!] &= \text{int}[][] \\
[\![t]\!] &= \text{int}[]
\end{aligned}
$$

Подумайте, что происходит в получившейся реализации, если в программе есть рекурсивный тип?

Программа типизируется, так как используется унификация на основе Union-Find.

Лекция 3

У решетки есть максимальный и минимальный элементы ($\top$, $\bot$). Являются ли они точной верхней или нижней гранью какого-либо подмножества S?

Да. $\top$ является точной верхней границей, а $\bot$ нижней например у подмножества (количества > 1) единичной решетки. Или в решетке которая указана в примере, $\{1, -, 0, +0, -0\}$.

```
        ?
      /   \
    +0     -0
   /  \   /  \
  +     0     -
  |     |    /
  1     |   /
    \   |  /
        ⊥
```

Уникальны ли они?

Не всегда. В решетке которая содержит только один элемент — он является $\top$ и $\bot$ одновременно.

Как выглядит $\top$ произведения решёток $L_1, L_2, ..., L_n$? А $\bot$?

По определению произведения, верхняя: $\top L_1, \top L_2, ..., \top L_n$. Нижняя: $\bot\ L_1, \bot\ L_2, ..., \bot\ L_n$

Какая высота решётки произведений?

Высота решетки произведения равна сумме высот решеток-множителей.

Какие точные грани решётки отображений?

По определению:

$$
A \to L = \{[a_1 \to x_1, a_2 \mapsto x_2, ...] | A = \{a_1, a_2, ...\} \wedge x_1, x_2, ... \in L\}
$$
$$
f \sqsubseteq \Leftrightarrow \forall a_i : f(a_i) \sqsubseteq g(a_i), \text{где } f, g \in A \to L
$$

Для решетки отображений $A \to L$: точная верхняя грань — $\forall a : A.a \to \top$, точная нижняя грань — $\forall a : A.a \to \bot$.

Какая высота решётки отображений?

По определению выше, высота решётки отображений равна произведению мощности множества A на высоту решётки $L : |A| * h(L)$

Можно ли выразить анализ типов с предыдущей лекции как анализ над решетками?
Да. Можно использовать плоскую решетку от множества возможных типов, где $\top$ будет любой тип, а $\perp$ — невозможность вывода типизации.

Можно ли выразить анализ над решётками как анализ типов?
Если ввести `Any` как $\top$ (как в TypeScript, Kotlin) и `Nothing` как $\perp$ (или `!` как в Rust), тогда при помощи механизма наследования можно задать отношения между типами в решетке.

Лекция 4

Какая сложность структурного алгоритма для liveness analysis?
Сложность структурного алгоритма в общем случае — $O(n \cdot h \cdot k)$, где n — количество узлов CFG, h — высота решетки, а k — время вычисления constraint функции.

Для данного анализа — $O(n \cdot c^2)$, где n — количество узлов, c — количество переменных, h – высота решетки ($c = h - 1 \sim h$). Циклы не влияют на оценку.

Распишите и решите систему ограничений для примера

```
var x, a, b;          // {}
x = input;            // {}
a = x - 1;            // {x - 1}
b = x - 2;            // {x - 1, x - 2}
while (x > 0) {       // {x - 1, x > 0}
  output a * b - x;   // {x - 1, x > 0, a * b, a * b - x}
  x = x - 1;          // {a * b, x - 1}
}                     // {a * b, x - 1}
output a * b;         // {x - 1, x > 0}
```

Лекция 5

Предложите решетку для реализации анализа размера переменных
Используем интервальную решетку для аппроксимации возможных значений переменных программы во время выполнения. Множество возможных значений будет включать в себя значения от минимального до максимального возможного, а так же $-\inf$ и $+\inf$. С помощью widening сводим решетку.

Лекция 6

Напишите вариант программы, для которой анализ открытости-закрытости файлов не показывает корректный результат

```
if (x == 42) {
  open();
}

if (x == 42) {
  flag = 1;
}

...

if (flag == 1) {
  close();
}
```

Так как текущий анализ следит только за flag не понимает что при $x == 42$ будут выполняться оба условия, то в конце получим $(\text{flag} = 1) \to *$.

Предложите, каким образом можно решить описанные в лекции проблемы в этой ситуации

Добавить правила которые будут учитывать ситуации в программе, когда после $(\text{flag} = 0) \to *$ может идти $(\text{flag} = 1)$.

Лекция 8

Напишите вариант программы, для которой контекстно-чувствительный анализ знаков требует коэффициент $k > 1$

```
factorial(arg) {
   if (arg > 0) { return rec(arg-1); }
   return arg;
}

xyz(n) {
  factorial(-n);
}

main() {
   output xyz(42);
   output xyz(-42);
}
```

Приведите пример решётки, для которой контекстно-чувствительный анализ в функциональном стиле является более ресурсозатратным, чем контекстно-чувствительный анализ по месту вызова с глубиной 2

Решетка из булеана переменных для анализа живости переменных: $\text{States} = \text{Var} \to 2^{\text{Var}}$.

- По месту вызова с глубиной 2: $|\text{Nodes}| * 2 * |\text{Var}| * 2^{|\text{Nodes}|}$
- Функциональный стиль: $|\text{Nodes}| * |\text{Var}| * 2^{|\text{Nodes}|} * |\text{Var}| * 2^{|\text{Nodes}|}$