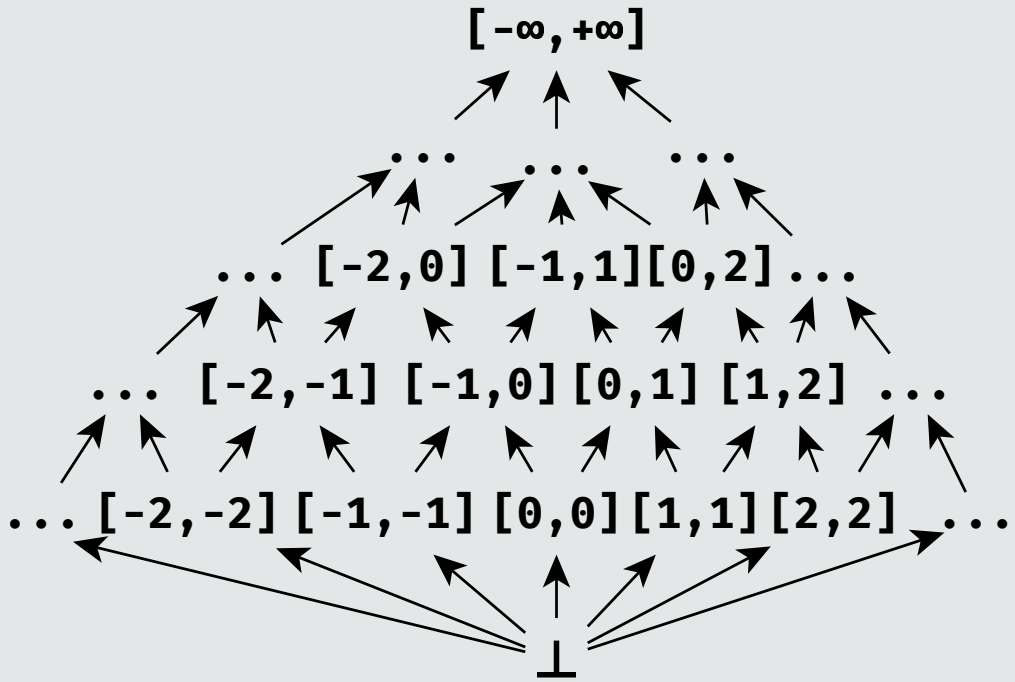
Бесконечность не предел

Как найти иголку в интервальном стоге

Марат Ахин Михаил Беляев

13 марта 2018 г.

Интервальная решетка



Есть ли тут какие-нибудь проблемы?

Проблема, собственно, одна — она бесконечная

Интервальная решетка

$$Intervals = \{[l, u] \,|\, l, u \in \mathbb{N} \wedge l \leqslant u\} \cup \{\bot\}$$

$$[l, u] \sqsubseteq [l', u'] \Leftrightarrow l' \leqslant l \wedge u \leqslant u'$$

Зачем так сложно?

- Точнее чем альтернативные варианты
- Может использоваться для многих полезных анализов
  - Распространение констант
  - Поиск выхода за границы массива
  - Упрощение выражений

Let's eval!

$$[\![x = E;]\!] = JOIN(n)[x \mapsto eval(JOIN(n), E)]$$

$$[\![n]\!] = JOIN(n) \quad \text{для любых других } n$$

$$JOIN(v) = \bigcup_{w \in succ(v)} [\![w]\!]$$

Let's eval!

$$
\begin{aligned}
eval(\sigma, INT) &= [INT, INT] \\
eval(\sigma, E_1 \ op \ E_2) &= eval(\sigma, E_1) \ \overline{op} \ eval(\sigma, E_2) \\
eval(\sigma, x) &= \sigma(x)
\end{aligned}
$$

Вроде бы все просто...

Interval arithmetics

$$[l_1, h_1] \overline{op} [l_2, h_2] = [min\ x|_{l_1}^{h_1}\ op\ y|_{l_2}^{h_2}, max\ x|_{l_1}^{h_1}\ op\ y|_{l_2}^{h_2}]$$

- Что-то слишком сложно...
  - А какая сложность получается?
  - Почему в принципе нельзя сделать проще?
  - Приведите один или более примеров, когда проще не работает

Interval comparisons

$$[l_1, h_1] \overline{op} [l_2, h_2] = [0, 1]$$

- Что-то слишком просто...
  - А почему нельзя сделать как выше?
  - Как можно сделать лучше?
  - Приведите один или более примеров, когда «как выше» не работает

Бесконечность не предел

Проблема, собственно, одна — решетка бесконечная

- Наш алгоритм неподвижной точки не работает
  - Приведите пример программы на TIP, в которой мы расходимся
- Давайте ограничим размер целых чисел!
  - В реальных программах это же правда?!
  - На практике все равно не работает...

Надо что-то менять (с)

Widening

*Давайте бегать по **конечной** версии нашей решетки*

- $\omega(L)$ — образ решетки L
  - $\omega(L) = \{y \in L | \exists x \in L : y = \omega(x)\}$
  - Должен иметь конечную высоту, чтобы все это имело смысл
- ω — функция «расширения»
  - $fix(\omega \circ f)$ сходится к переаппроксимации $fix(f)$
  - ω «загрубляет» информацию

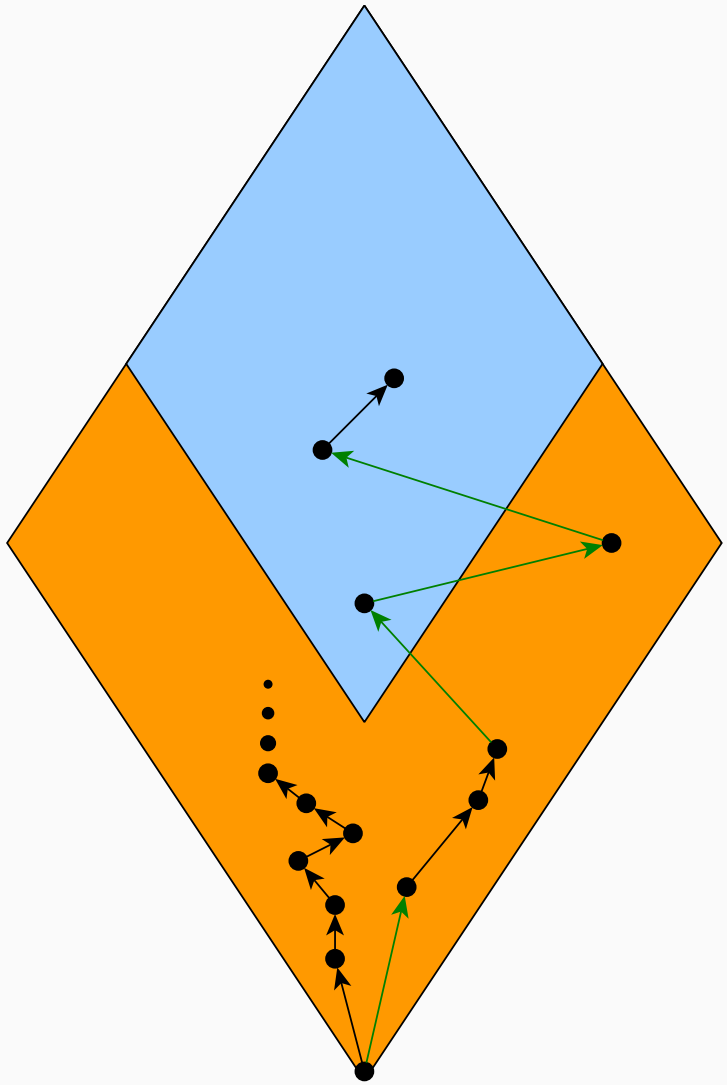
Widening

- ω должна быть
  - экстенсивной
  - монотонной
  - идемпотентной[1]

$$f^i(\bot, \ldots, \bot) \sqsubseteq (w \circ f)^i(\bot, \ldots, \bot) \sqsubseteq fix(w \circ f)$$

[1] на самом деле, нет, но на практике чаще всего да

Widening in practice

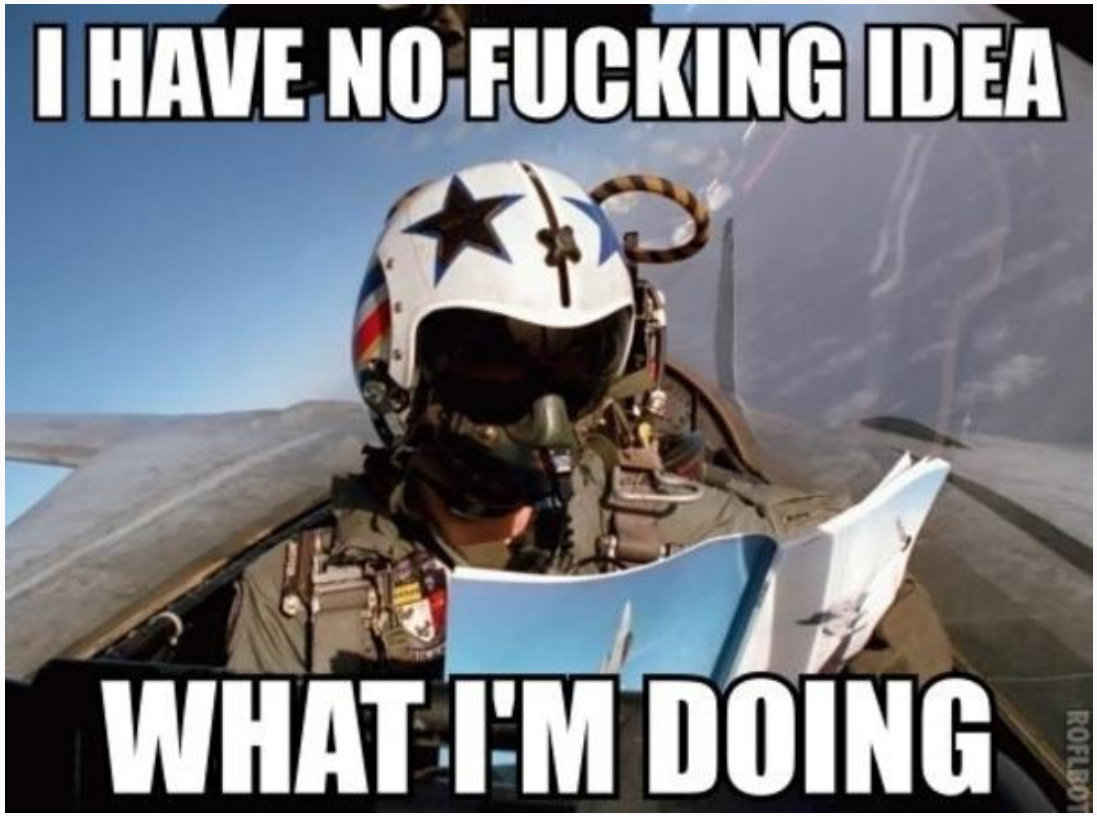


Widening for intervals

Какую конечную версию нашей интервальной решетки можно придумать?

Widening for intervals

Какую конечную версию нашей интервальной решетки можно придумать?



Widening for intervals

Программы обычно работают с какими-то числами

- Возьмем множество чисел $B \subset N$
  - Например, множество всех констант в программе
  - Любое ли множество нам подходит?
- Будем загрублять все интервалы до ближайших интервалов над B

$$\omega([a, b]) = [max\{i \in B | i \leqslant a\}, min\{i \in B | i \geqslant b\}]$$
$$\omega(\bot) = \bot$$

Расходимся по одному (c)

```
var x,y;
y = 0;
while (input) {
    x = 7;
    x = x + 1;
    y = y + 1;
} // here be wat?
```

$$[x \mapsto \bot, y \mapsto [0, 0]]$$

$$[x \mapsto [8, 8], y \mapsto [0, 1]]$$

$$[x \mapsto [8, 8], y \mapsto [0, 2]]$$

$$[x \mapsto [8, 8], y \mapsto [0, 3]]$$

$$\ldots$$

Попробуем расширяться

```
var x,y;
y = 0;
while (input) {
    x = 7;
    x = x + 1;
    y = y + 1;
} // here be wat?
```

$$[x \mapsto \bot, y \mapsto [0, 0]]$$

$$[x \mapsto [7, ?], y \mapsto [0, 1]]$$

$$[x \mapsto [7, ?], y \mapsto [0, 7]]$$

$$[x \mapsto [7, ?], y \mapsto [0, ?]]$$

$$B = \{0, 1, 7\}$$

Что-то не хватает...

Попробуем расширяться (2)

```
var x,y;
y = 0;
while (input) {
    x = 7;
    x = x + 1;
    y = y + 1;
} // here be wat?
```

$$[x \mapsto \bot, y \mapsto [0, 0]]$$

$$[x \mapsto [7, +\infty], y \mapsto [0, 1]]$$

$$[x \mapsto [7, +\infty], y \mapsto [0, 7]]$$

$$[x \mapsto [7, +\infty], y \mapsto [0, +\infty]]$$

$$B = \{-\infty, 0, 1, 7, +\infty\}$$

Проблемы widening

- Результаты являются переаппроксимацией
- Построение B не всегда является очевидным
    - Обсудим чуть позже

Давайте попробуем уточнить переаппроксимацию

Narrowing

$$fix = \sqcup f^{i}(\bot)$$

$$fix\omega = \sqcup(\omega \circ f)^{i}(\bot)$$

$$fix \sqsubseteq fix\omega$$

Narrowing

$$fix = \sqcup f^{i}(\bot)$$

$$fix\omega = \sqcup(\omega \circ f)^{i}(\bot)$$

$$fix \sqsubseteq fix\omega$$

$$fix \sqsubseteq f(fix\omega) \sqsubseteq fix\omega$$

Narrowing

$$fix = \sqcup f^i(\bot)$$

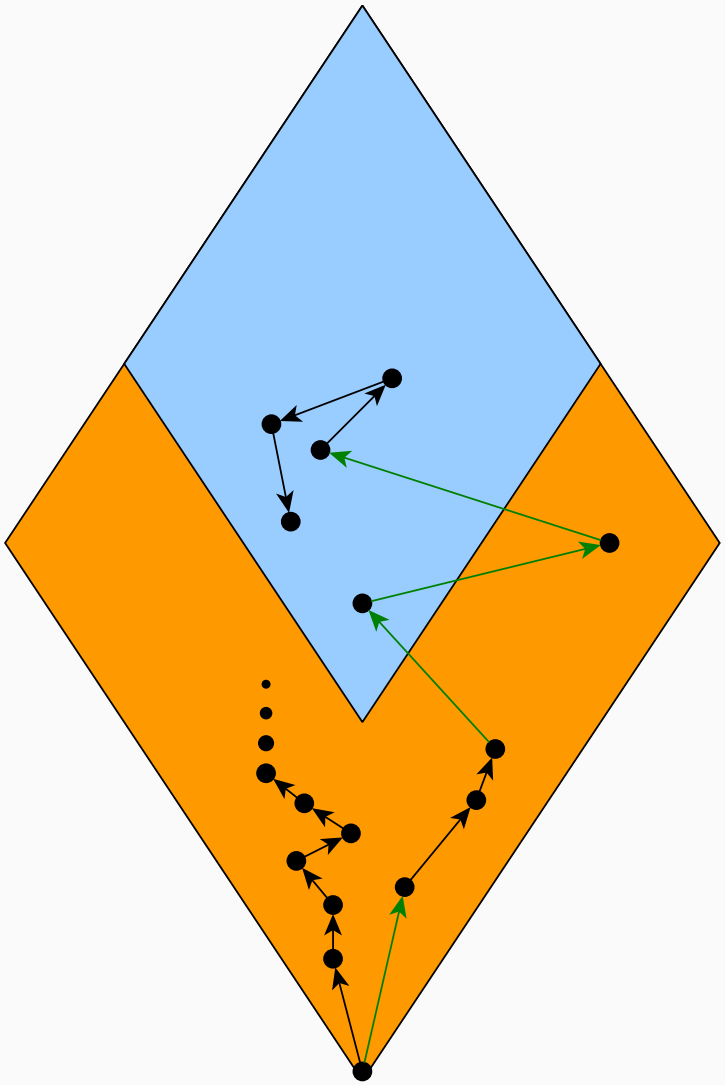
$$fix\omega = \sqcup(\omega \circ f)^i(\bot)$$

$$fix \sqsubseteq fix\omega$$

$$fix \sqsubseteq f(fix\omega) \sqsubseteq fix\omega$$

Применение f к результатам widening уточняет результат (sic!)

Narrowing in practice



Попробуем сужаться

```
var x,y;
y = 0;
while (input) {
    x = 7;
    x = x + 1;
    y = y + 1;
} // here be wat?
```

$$[x \mapsto \bot, y \mapsto [0, 0]]$$

$$[x \mapsto [7, +\infty], y \mapsto [0, 1]]$$

$$[x \mapsto [7, +\infty], y \mapsto [0, 7]]$$

$$[x \mapsto [7, +\infty], y \mapsto [0, +\infty]]$$

$$\ldots$$

$$[x \mapsto [8, 8], y \mapsto [0, +\infty]]$$

$$B = \{-\infty, 0, 1, 7, +\infty\}$$

Проблемы narrowing

- Результаты все еще могут являться переаппроксимацией
- Если множество B подобрано плохо, уточнять результат мы будем очень и очень долго

```
// The infamous McCarthy's 91 function
f(x) {
    var r;
    if (x > 100) {
        r = x - 10;
    } else {
        r = f(f(x + 11));
    }
    return r;
}
```

Как подобрать B?

- Выбрать значения из спецификации программы
- Выбрать значения из самой программы
- Выбрать случайные значения
- Выполнить несколько итераций анализа над полной решеткой и выбрать значения из результатов

Как ускорить narrowing?

- Классическая схема — сперва widening, потом narrowing
  - Почему это работает плохо?
- Можно пробовать чередовать widening и narrowing, учитывая структуру программы
  - Результаты, скорее всего, будут точнее
  - Кроме того, это еще может быть быстрее
  - Почему?

Как улучшить все

Более общий widening

- $\nabla : LxL \to L$ — это widening-оператор
  - $l_1 \sqsubseteq (l_1 \nabla l_2) \sqsupseteq l_2 \; \forall l_1, l_2 \in L$
  - для любой возрастающей цепочки $x_0 \sqsubseteq x_1 \sqsubseteq \ldots$ последовательность $y_{i+1} = y_i \nabla x_{x+1}$ сходится
- Теперь ищем неподвижную точку через $y_{i+1} = y_i \nabla f(y_i)$

Как улучшить все

Более общий narrowing

- $\bigtriangledown : LxL \longrightarrow L$ — это narrowing-оператор
  - $l_1 \sqsubseteq l_2 \Rightarrow l_1 \sqsubseteq (l_1 \bigtriangledown l_2) \sqsubseteq l_2 \; \forall l_1, l_2 \in L$
  - для любой нисходящей цепочки $x_0 \sqsupseteq x_1 \sqsupseteq \ldots$ последовательность $y_{i+1} = y_i \bigtriangledown x_{x+1}$ сходится
- Теперь ищем неподвижную точку через $y_{i+1} = y_i \bigtriangledown f(y_i)$

Warrowing

Более общее все

- $\Diamond : LxL \rightarrow L$ — это warrowing-оператор
  - $l_1 \Diamond l_2 = \begin{cases} l_1 \bigtriangleup l_2 \text{ if } l_2 \sqsubseteq l_1 \\ l_1 \bigtriangledown l_2 \text{ otherwise} \end{cases}$
- Ускоряет сходимость
- Увеличивает точность итогового решения
- **Не работает**[2]

[2] для стандартных решателей монотонных фреймворков

Алгоритмы поиска неподвижной точки (revised)

function StructuredWorkListFixPointAlgorithm(f)

$(x_1, x_2, \ldots, x_n) \leftarrow (\bot, \bot, \ldots, \bot)$

$WL \leftarrow \{v_1, v_2, \ldots, v_n\}$ // *priority queue*

while $WL \neq \emptyset$ **do**

$\{v_i\} \cup WL \leftarrow WL$ // *w.r.t. priority*

$y = f_i(x_1, x_2, \ldots, x_n)$

if $x_i \neq y$ **then**

$x_i \leftarrow y$

$WL \leftarrow WL \cup \mathbf{v_i} \cup deps(v_i)$

end if

end while

return x

end function

На подумать

- Допустим, мы хотим реализовать оптимизирующий компилятор для языка TIP. Среди прочего, для работы ему требуется информация о размерах различных переменных.
  - bool (1 bit)
  - byte (8 bit signed)
  - char (16 bit unsigned)
  - int (32 bit signed)
  - bigint (any integer)
  - any (any thing)
- `(T) E` — операция приведения типов

На подумать

- Предложите решетку для реализации анализа размера переменных
    - Нужно описать не только решетку для одного абстрактного значения, но и все другие решетки, требуемые для анализа целой программы
- Опишите правила вычисления различных выражений
- Придумайте нетривиальный пример программы на TIP для получившегося анализа и посмотрите, что для него получается

На поделать

- Допишите реализацию метода `widen` в `IntervalAnalysisWorklistSolverWithWidening`
  - `src/tip/analysis/IntervalAnalysis.scala`
- Реализуйте придуманный вами анализ размера переменных
  - `src/tip/analysis/VariableSizeAnalysis.scala`
  - (такого файла там ещё нет =))

В следующей серии

Почему ветвления — зло!

akhin@kspt.icc.spbstu.ru belyaev@kspt.icc.spbstu.ru