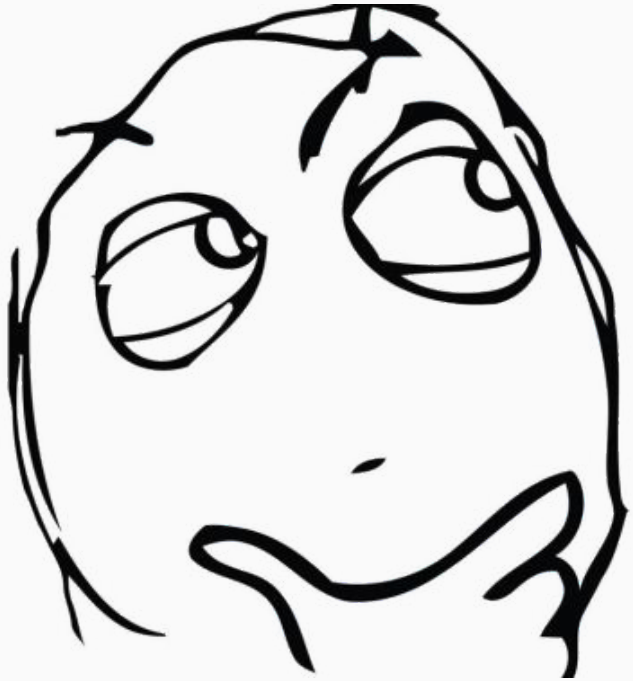
Статический анализ: начало

Марат Ахин Михаил Беляев

20 февраля 2018 г.

В предыдущей серии

- Что такое статический анализ?
- Что такое полнота?
- Что такое точность?



В предыдущей серии

Теорема Райса

Ни одно нетривиальное свойство Тьюринг-полной программы автоматически установить нельзя

Что значит «нетривиальное»?

В предыдущей серии

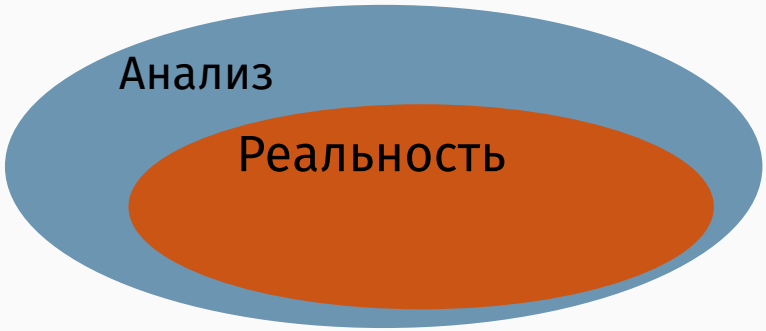
Теорема Райса

Ни одно нетривиальное свойство Тьюринг-полной программы автоматически установить нельзя

Что значит «нетривиальное»?

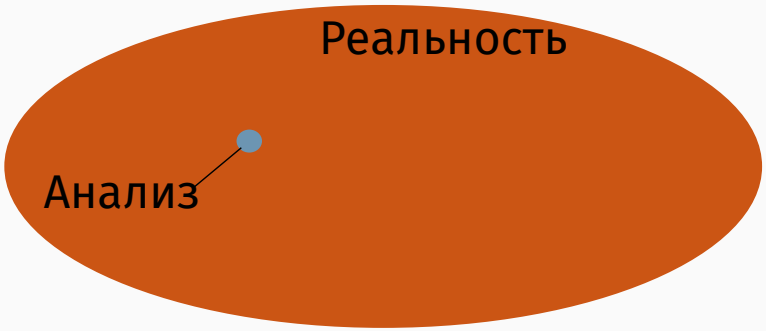
Нетривиальное — существуют программы как с ним, так и без него

Полнота vs точность



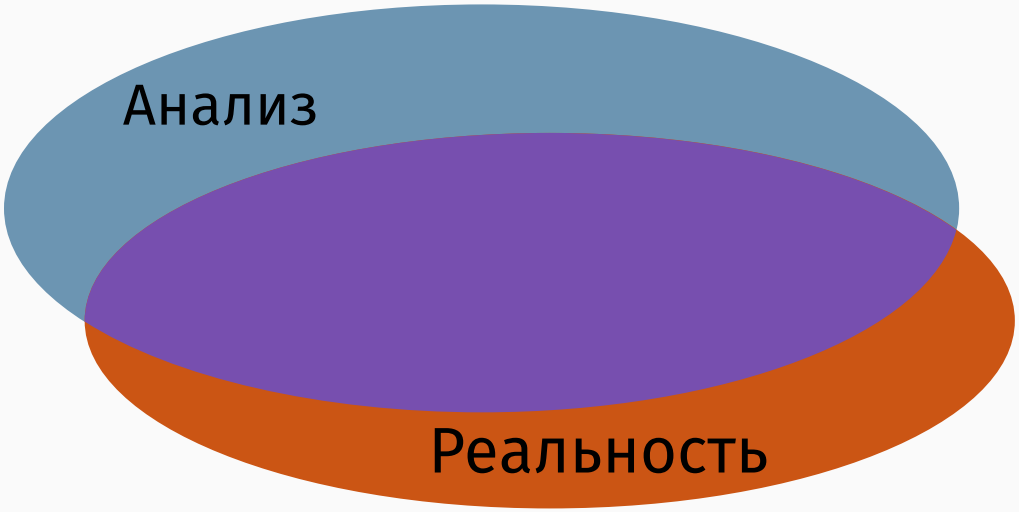
Ориентация на точность vs ориентация на полноту

Полнота vs точность



Но может быть и вот так

Полнота vs точность



В жизни обычно бывает вот так

Полнота vs точность

- **Насколько сложно сделать 100% полный анализ?**
  Нинасколько. Например, если вы ищете ошибки, просто находите их **везде**.
- **Насколько сложно сделать 100% точный анализ?**
  Немного сложнее. Давайте будем находить только совсем тривиальные случаи.

Будете ли вы пользоваться таким анализом?

Полнота vs точность

- **Насколько сложно сделать 100% полный анализ?**
  Нинасколько. Например, если вы ищете ошибки, просто находите их **везде**.
- **Насколько сложно сделать 100% точный анализ?**
  Немного сложнее. Давайте будем находить только совсем тривиальные случаи.

Будете ли вы пользоваться таким анализом?

На практике мы хотим улучшить одну характеристику, сохраняя другую в разумных пределах.
Зачастую это можно сделать только крайне сложными алгоритмами.

Полнота vs точность vs производительность

- Никому не нужен анализ с точностью в районе 0
- Никому не нужен анализ с полнотой в районе 0
- Никому не нужен анализ, который анализирует 10 строк 10 лет

Полнота, точность, производительность: выбираем любые три

Начнём уже что-то анализировать!

- На прошлой лекции мы сказали, что сигнатурный анализ — не самый распространённый тип СА
- Какой же самый?

Начнём уже что-то анализировать!

- На прошлой лекции мы сказали, что сигнатурный анализ — не самый распространённый тип СА
- Какой же самый?

- Анализы, встроенные в компилятор
    - Но это нечестно, вы их результатов не видите

Начнём уже что-то анализировать!

- На прошлой лекции мы сказали, что сигнатурный анализ — не самый распространённый тип СА
- Какой же самый?

- Анализы, встроенные в компилятор
    - Но это нечестно, вы их результатов не видите

- Анализ типов!

Анализ типов

- Да, анализ типов, даже самый простой — это тоже СА
- Он аппроксимирует динамические свойства программы статически

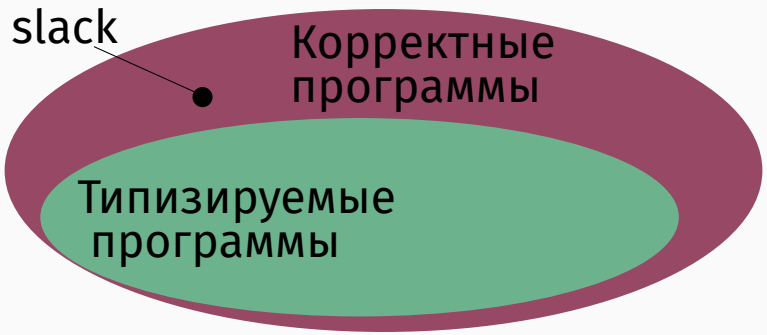
Какие гарантии даёт анализ типов?

- Только указатели можно разыменовывать
- Только функции можно вызывать
- Условия в `if` должны быть булевыми
- Сравнивать можно только выражения одного типа
- ...

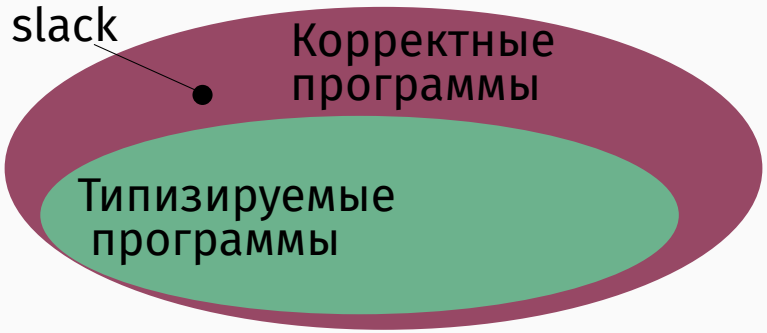
Анализ типов

Главный вопрос: могут ли ошибки работы со значениями разных типов произойти во время выполнения?

- Это свойство *нетривиально*, следовательно, неразрешимо
- На практике берётся аппроксимация снизу
  (или сверху, это как посмотреть)



Анализ типов



- **Type soundness**: если что-то протипизировалось, то оно во время выполнения не упадёт!
  - По крайней мере, с ошибками системы типов языка
- В любом языке есть программы, которые являются корректными, но отвергаются type checker'ом
- Во многих языках система типов *unsound*:
  - Массивы в Java
  - `void*` и `union` в C
  - `reinterpret_cast` в C++

Slack

Slack — это область корректных программ, которые, тем не менее, не типизируются

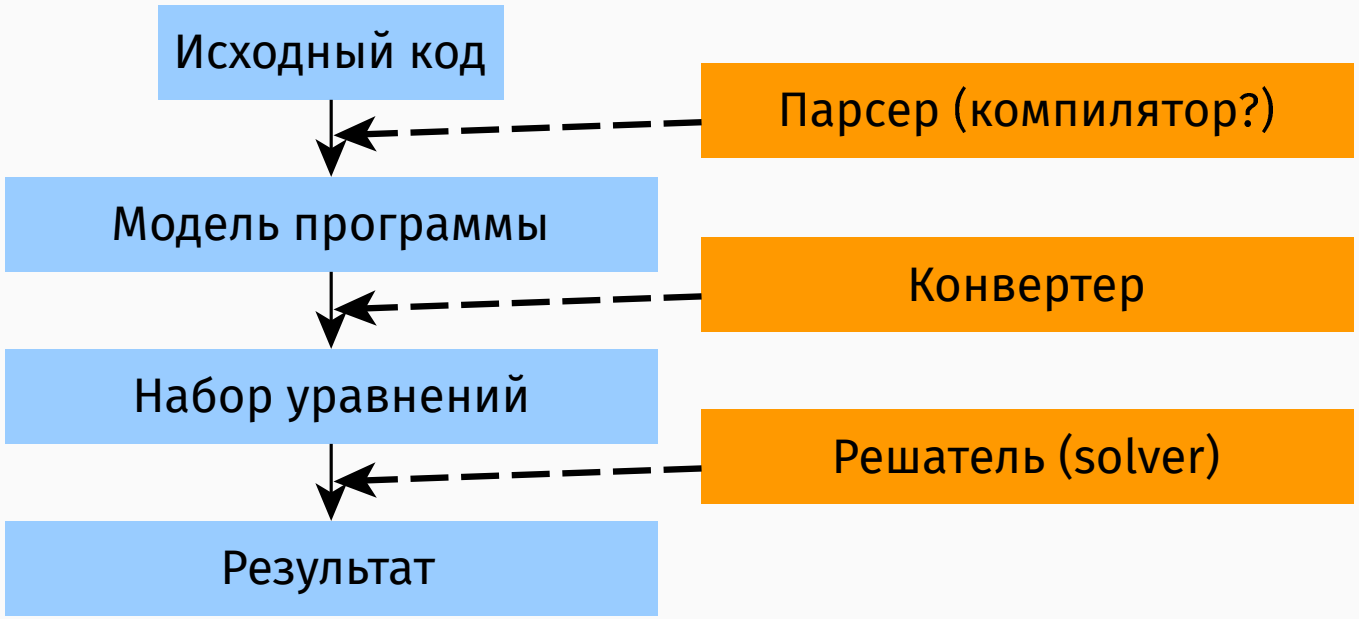
```
String toString() {
    Object x = foo();
    if(x instanceof String) {
        return x; // Ну точно String же!
    }
    return "";
}
```

Одна из основных проблем теории типов — уменьшение области slack

Constraint-based analysis

Проверка типов — это пример анализа на основе ограничений (уравнений)

(aka constraint-based analysis)



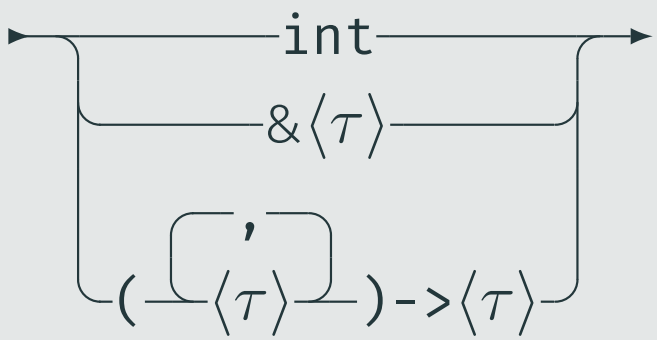
А что у нас в TIP?

- Стоп, в TIPe нет типов!

А что у нас в TIP?

- Стоп, в TIP нет типов!
- В смысле, всё — это числа
- Но есть указатели и функции!

$\langle \tau \rangle$



Примеры корректных типов:

- `&&int`
- `(int, (int, int) -> &int) -> int`

Система типов TIP?

- Всё, о чём мы говорим дальше, является частным случаем системы типов Хиндли-Дамаса-Милнера
- Гугл в помощь

$$
\begin{array}{lcl}
\mathcal{T}[\![\cdot]\!] & : & \mathbb{E} \to (\mathbb{X} \to \mathbb{P})^{\perp}_{=_{\mathbb{P}}} \to (\mathbb{X} \cup \{\kappa\} \to \mathbb{P})^{\perp}_{=_{\mathbb{P}}} \\
\mathcal{T}[\![x]\!] & = & \overline{\mathbf{lca}} \circ \lambda \overline{\rho}_{\mathbb{M}} \,.\, \bigcap \{\alpha^{\mathcal{X}}_{\mathbb{M}1}(\rho[\kappa \mapsto S\rho]) \mid S \in \mathcal{C}[\![x]\!] \wedge \rho \in \gamma^{\mathcal{X}}_{\mathbb{M}}(\overline{\rho}_{\mathbb{M}})\} \circ \overline{\mathbf{ground}} \\
 & \overset{(1)}{=} & \lambda \rho_{\mathbb{P}} \,.\, \overline{\mathbf{lca}}(\bigcap_{\rho \in \mathbb{X} \to \mathbb{U}_{\perp}} \{\alpha^{\mathcal{X}}_{\mathbb{M}1}(\rho[\kappa \mapsto \rho(x)]) \mid \overline{\mathbf{ground}}(\rho_{\mathbb{P}}) \subseteq \alpha^{\mathcal{X}}_{\mathbb{M}1}(\rho)\}) \\
 & \overset{(2)}{=} & \lambda \rho_{\mathbb{P}} \,.\, \overline{\mathbf{lca}}\left(\bigcap_{\rho \in \mathbb{X} \to \mathbb{U}_{\perp}} \left\{ \bigcup \left\{ \overline{\rho}_{\mathbb{M}} \subseteq \mathbb{X} \cup \{\kappa\} \to \mathbb{M} \;\middle|\; \begin{array}{l} \forall \rho_{\mathbb{M}} \in \overline{\rho}_{\mathbb{M}}, y \in \mathbb{X} \cup \{\kappa\} \,.\, \rho_{\mathbb{M}}(y) \in \alpha_{\mathbb{M}}(\rho[\kappa \mapsto \rho(x)](y)) \wedge \\ \mathbf{m.r.} \wedge \overline{\mathbf{ground}}(\rho_{\mathbb{P}}) \subseteq \exists_{\kappa}(\overline{\rho}_{\mathbb{M}}) \subseteq \alpha^{\mathcal{X}}_{\mathbb{M}1}(\rho) \end{array} \right\} \right\}\right) \\
\text{if } x \notin \mathbb{X}^{\lambda} & \overset{(3)}{=} & \lambda \rho_{\mathbb{P}} \,.\, \overline{\mathbf{lca}}(\exists_{y'_1,\ldots y'_n}(expand_{xy_1\ldots y_n.\kappa y'_1\ldots y'_n}(\overline{\mathbf{ground}}(\rho_{\mathbb{P}})) \text{ where } \{y_1,\ldots y_n\} = \{y \in \mathbf{dom}(\rho_{\mathbb{P}}) \mid x \prec y\} \\
\text{if } x \notin \mathbb{X}^{\lambda} & \overset{(4)}{=} & \lambda \rho_{\mathbb{P}} \,.\, \exists_{y'_1,\ldots y'_n}(expand_{xy_1\ldots y_n.\kappa y'_1\ldots y'_n}(\rho_{\mathbb{P}})) \text{ where } \{y_1,\ldots y_n\} = \{y \in \mathbf{dom}(\rho_{\mathbb{P}}) \mid x \prec y\} \\
\text{if } x \in \mathbb{X}^{\lambda} & \overset{(5)}{=} & \lambda \rho_{\mathbb{P}} \,.\, \rho_{\mathbb{P}}[\kappa \mapsto \rho_{\mathbb{P}}(x)] \\
\mathcal{T}[\![\lambda x \,.\, e]\!] & \overset{(6)}{=} & \overline{\mathbf{lca}} \circ \lambda \overline{\rho}_{\mathbb{M}} \,.\, (\bigcap_{\rho \in \mathbb{X} \to \mathbb{U}_{\perp}} (\bigcup \{\rho_{\mathbb{M}}[\kappa \mapsto t] \mid t \in \alpha_{\mathbb{M}}(S\rho) \wedge S \in \mathcal{C}[\![\lambda x \,.\, e]\!] \wedge \rho_{\mathbb{M}} \in \overline{\rho}_{\mathbb{M}} \subseteq \alpha^{\mathcal{X}}_{\mathbb{M}1}(\rho)\})) \circ \overline{\mathbf{ground}} \\
 & \overset{(7)}{=} & \lambda \rho_{\mathbb{P}} \,.\, \bigcap_{\rho \in \mathbb{X} \to \mathbb{U}_{\perp}} \overline{\mathbf{lca}}\{\rho_{\mathbb{M}}[\kappa \mapsto t] \mid t \in \alpha_{\mathbb{M}}(S\rho) \wedge S \in \mathcal{C}[\![\lambda x \,.\, e]\!] \wedge \rho_{\mathbb{M}} \in \overline{\mathbf{ground}}(\rho_{\mathbb{P}}) \subseteq \alpha^{\mathcal{X}}_{\mathbb{M}1}(\rho)\} \\
 & \overset{(8)}{=} & \lambda \rho_{\mathbb{P}} \,.\, \bigcap_{\rho \in \mathbb{X} \to \mathbb{U}_{\perp}} \overline{\mathbf{lca}}\{\rho_{\mathbb{M}}[\kappa \mapsto t] \mid t \in \alpha_{\mathbb{M}}(\uparrow^{\mathbb{U}}_{\mathbb{F}} (\lambda v \,.\, S(\rho[x \mapsto v]))) \wedge S \in \mathcal{C}[\![e]\!] \wedge \rho_{\mathbb{M}} \in \overline{\mathbf{ground}}(\rho_{\mathbb{P}}) \subseteq \alpha^{\mathcal{X}}_{\mathbb{M}1}(\rho)\} \\
 & \overset{(9)}{=} & \lambda \rho_{\mathbb{P}} \,.\, \bigcap_{\rho \in \mathbb{X} \to \mathbb{U}_{\perp}} \overline{\mathbf{lca}}\{\rho_{\mathbb{M}}[\kappa \mapsto t_1 \to t_2] \mid v_2 = S(\rho[x \mapsto v_1]) \wedge S \in \mathcal{C}[\![e]\!] \wedge v_1 \in \mathbb{U} \wedge t_i \in \alpha_{\mathbb{M}}(v_i) \wedge \rho_{\mathbb{M}} \in \overline{\mathbf{ground}}(\rho_{\mathbb{P}}) \subseteq \alpha^{\mathcal{X}}_{\mathbb{M}1}(\rho)\} \\
 & \overset{(10)}{=} & \lambda \rho_{\mathbb{P}} \,.\, \bigcap_{\rho \in \mathbb{X} \to \mathbb{U}_{\perp}} \overline{\mathbf{lca}}\{\rho_{\mathbb{M}}[\kappa \mapsto t_1 \to t_2] \mid t_2 = \mathcal{T}[\![e]\!](\rho_{\mathbb{M}}[x \mapsto t_1])\kappa \wedge t_1 \in \mathbb{M} \wedge \rho_{\mathbb{M}} \in \overline{\mathbf{ground}}(\rho_{\mathbb{P}}) \subseteq \alpha^{\mathcal{X}}_{\mathbb{M}1}(\rho)\} \\
 & \overset{(11)}{=} & \lambda \rho_{\mathbb{P}} \,.\, \overline{\mathbf{lca}}\{\rho_{\mathbb{M}}[\kappa \mapsto t_1 \to t_2] \mid t_2 = \mathcal{T}[\![e]\!](\rho_{\mathbb{M}}[x \mapsto t_1])\kappa \wedge t_1 \in \mathbb{M} \wedge \rho_{\mathbb{M}} \in \overline{\mathbf{ground}}(\rho_{\mathbb{P}})\} \\
 & \overset{(12)}{=} & \lambda \rho_{\mathbb{P}} \,.\, \begin{cases} \rho'_{\mathbb{P}}[\kappa \mapsto t_1 \to t_2] & \text{if } \perp_{\mathbb{P}} \neq \rho'_{\mathbb{P}} = \mathcal{T}[\![e]\!](\rho_{\mathbb{P}}[x \mapsto (\mathrm{a})_{=_{\mathbb{F}}}]) \text{ and } t_1 = \rho'_{\mathbb{P}}(x) \text{ and } t_2 = \rho'_{\mathbb{P}}(\kappa) \\ \perp_{\mathbb{P}} & \text{otherwise} \end{cases} \\
\mathcal{T}[\![e_1\ e_2]\!] & = & \overline{\mathbf{lca}} \circ \lambda \overline{\rho}_{\mathbb{M}} \,.\, (\bigcap_{\rho \in \mathbb{X} \to \mathbb{U}_{\perp}} (\bigcup \{\rho_{\mathbb{M}}[\kappa \mapsto t] \mid t \in \alpha_{\mathbb{M}}(S\rho) \wedge S \in \mathcal{C}[\![e_1\ e_2]\!] \wedge \rho_{\mathbb{M}} \in \overline{\rho}_{\mathbb{M}} \subseteq \alpha^{\mathcal{X}}_{\mathbb{M}1}(\rho)\}) \circ \overline{\mathbf{ground}} \\
 & \overset{(13)}{=} & \lambda \rho_{\mathbb{P}} \,.\, \overline{\mathbf{lca}}[\rho_{\mathbb{M}}[\kappa \mapsto t] \mid \downarrow^{\mathbb{U}}_{\mathbb{F}} (v_1) \in \mathbb{F} \wedge v_2 \neq \Omega \wedge t \in \alpha_{\mathbb{M}}(\downarrow^{\mathbb{U}}_{\mathbb{F}} (v_1)v_2) \wedge \mathbf{env}\} \text{ where} \\
 & & \quad \mathbf{env} \equiv v_i = S_i\rho \wedge S_i \in \mathcal{S}[\![e_i]\!] \ \wedge \rho_{\mathbb{M}} \in \overline{\rho}_{\mathbb{M}} \subseteq \alpha^{\mathcal{X}}_{\mathbb{M}1}(\rho) \\
 & \overset{(14)}{=} & \lambda \rho_{\mathbb{P}} \,.\, \bigcap_{\rho \in \mathbb{X} \to \mathbb{U}_{\perp}} \overline{\mathbf{lca}}\{\rho_{\mathbb{M}}[\kappa \mapsto t] \mid \exists t_a, t_r \in \mathbb{M} \,.\, \downarrow^{\mathbb{U}}_{\mathbb{F}} (v_1) \in \mathbb{F} \wedge v_2 \neq \Omega \wedge \uparrow^{\mathbb{U}}_{\mathbb{F}} (\downarrow^{\mathbb{U}}_{\mathbb{F}} (v_1)) \in \gamma_{\mathbb{M}}(t_a \to t_r) \wedge \\
 & & \qquad\qquad \forall v \in \gamma_{\mathbb{M}}(t_a) \,.\, \downarrow^{\mathbb{U}}_{\mathbb{F}} (v_1)v \in \gamma_{\mathbb{M}}(t_r) \setminus \perp_{\mathbb{P}} \wedge t \in \alpha_{\mathbb{M}}(\downarrow^{\mathbb{U}}_{\mathbb{F}} (v_1)v_2) \wedge \mathbf{env}\} \\
 & \overset{(15)}{=} & \lambda \rho_{\mathbb{P}} \,.\, \bigcap_{\rho \in \mathbb{X} \to \mathbb{U}_{\perp}} \overline{\mathbf{lca}}\{\rho_{\mathbb{M}}[\kappa \mapsto t] \mid \exists t_r \in \mathbb{M} \,.\, v_1 \in \gamma_{\mathbb{M}}(t_a \to t_r) \wedge v_2 \in \gamma_{\mathbb{M}}(t_a) \wedge \downarrow^{\mathbb{U}}_{\mathbb{F}} (v_1)v_2 \in \gamma_{\mathbb{M}}(t_r) \wedge t \in \alpha_{\mathbb{M}}(\downarrow^{\mathbb{U}}_{\mathbb{F}} (v_1)v_2) \wedge \mathbf{env}\} \\
 & \overset{(16)}{=} & \lambda \rho_{\mathbb{P}} \,.\, \bigcap_{\rho \in \mathbb{X} \to \mathbb{U}_{\perp}} \overline{\mathbf{lca}}\{\rho_{\mathbb{M}}[\kappa \mapsto t_r] \mid \exists t_r \in \mathbb{M} \,.\, v_1 \in \gamma_{\mathbb{M}}(t_a \to t_r) \wedge v_2 \in \gamma_{\mathbb{M}}(t_a) \wedge \mathbf{env}\} \\
 & \overset{(17)}{=} & \lambda \rho_{\mathbb{P}} \,.\, \begin{cases} \overline{\mathbf{lca}}\{\rho_{\mathbb{M}}[\kappa \mapsto t_r] \mid \exists t_r \in \mathbb{M} \,.\, t_1 = t_a \to t_r \wedge t_2 = t_a \wedge t_i = \mathcal{T}[\![e_i]\!]\{\rho_{\mathbb{M}}\}\kappa \ \wedge \ \rho_{\mathbb{M}} \in \overline{\mathbf{ground}}(\rho_{\mathbb{P}})\} & \text{if } \mathcal{T}[\![e_i]\!]\rho_{\mathbb{M}} \neq \emptyset \\ \perp_{\mathbb{P}} & \text{otherwise} \end{cases} \\
 & \overset{(18)}{=} & \lambda \rho_{\mathbb{P}} \,.\, \begin{cases} \rho'_{\mathbb{P}}[\kappa \mapsto t_r] & \text{if } \perp_{\mathbb{P}} \neq \rho'_{\mathbb{P}} = \overline{\mathbf{gci}}(\mathcal{T}[\![e_1]\!]\rho_{\mathbb{P}}, \rho_{\mathbb{P}}[\kappa \mapsto (\mathcal{T}[\![e_2]\!]\rho_{\mathbb{P}}\kappa) \to (\mathrm{a})_{=_{\mathbb{P}}}]) \text{ and } \mathcal{T}[\![e_i]\!]\rho_{\mathbb{P}} \neq \perp_{\mathbb{P}} \text{ and let } t_a \to t_r = \rho'_{\mathbb{P}}(\kappa) \\ \perp_{\mathbb{P}} & \text{otherwise} \end{cases}
\end{array}
$$

Алгоритм анализа

- Берём AST и для каждого выражения строим уравнение
  - Все уравнения являются *однозначными равенствами*
  - Набор равенств легко решается алгоритмами унификации
- Вводим переменные $[\![E]\!]$ для каждого выражения E
- Здесь *выражение* — это не кусок текста, а нода в AST

Генерация уравнений

I	$[\![I]\!] = \texttt{int}$
E_1 `==` E_2	$[\![E_1]\!] = [\![E_2]\!] \wedge [\![E_1 \; op \; E_2]\!] = \texttt{int}$
$E_1 \; op \; E_2$	$[\![E_1]\!] = [\![E_2]\!] = [\![E_1 \; op \; E_2]\!] = \texttt{int}$
`input`	$[\![\texttt{input}]\!] = \texttt{int}$
X `=` E	$[\![X]\!] = [\![E]\!]$
`output` E	$[\![E]\!] = \texttt{int}$
`if(`E`) {`S`}`	$[\![E]\!] = \texttt{int}$
`if(`E`) {`S_1`} else {`S_2`}`	$[\![E]\!] = \texttt{int}$
`while(`E`) {`S`}`	$[\![E]\!] = \texttt{int}$

Генерация уравнений

`f(`X_1`,...,`X_N`) {... return` E `;}`	$[\![f]\!] = ([\![X_1]\!],\ldots,[\![X_N]\!])$ -> $[\![E]\!]$
`(`E`)(`E_1`,...,`E_N`)`	$[\![E]\!] = ([\![X_1]\!],\ldots,[\![X_N]\!])$ -> $[\![(E)(E_1,\ldots,E_N)]\!]$
`&` E	$[\![\&E]\!] = \&[\![E]\!]$
`alloc`	$[\![\texttt{alloc}]\!] = \&\alpha$
`null`	$[\![\texttt{null}]\!] = \&\alpha$
*E	$[\![E]\!] = \&[\![*E]\!]$
*X = E	$[\![X]\!] = \&[\![E]\!]$

Генерация уравнений

```
main() {
    var x,y,z;
    x = input;      ⟦x⟧ = int
    y = alloc;      ⟦y⟧ = &α
    *y = x;         ⟦y⟧ = &⟦x⟧
    z = *y;         ⟦z⟧ = ⟦*y⟧ = &⟦y⟧
    return x;       ⟦main⟧ = () -> ⟦x⟧
}
```

Генерация уравнений

`main() {`	
`  var x,y,z;`	
`  x = input;`	$[\![\texttt{x}]\!] = \texttt{int}$
`  y = alloc;`	$[\![\texttt{y}]\!] = \&\alpha$
`  *y = x;`	$[\![\texttt{y}]\!] = \&[\![\texttt{x}]\!]$
`  z = *y;`	$[\![\texttt{z}]\!] = [\![*\texttt{y}]\!] = \&[\![\texttt{y}]\!]$
`  return x;`	$[\![\texttt{main}]\!] = () \to [\![\texttt{x}]\!]$
`}`	

$$[\![\texttt{x}]\!] = \texttt{int}$$

$$[\![\texttt{y}]\!] = \&\texttt{int}$$

$$[\![\texttt{z}]\!] = \&\&\texttt{int}$$

$$[\![\texttt{main}]\!] = () \to \texttt{int}$$

Решение уравнений

Отлично решили. А как это сделать автоматически?

Унификация

Есть набор равенств вида:

$\mathsf{k}(\alpha,\mathsf{b},\beta) = \mathsf{k}(\mathsf{f}(\beta,\gamma),\gamma,\mathsf{d}(\gamma))$

Решение — набор значений переменных (**подстановка**), таких, что они делают эти равенства верными:

$\alpha = f(d(b), b)$

$\beta = d(b)$

$\gamma = b$

Решением этой задачи занимаются **унифицирующие солверы**

Унификация

Линейный алгоритм, Патерсон и Вегман, 1978 год

Повторяем до готовности набор операций:

- **delete**: удаляем уравнения вида $t = t$
- **decompose**: $c(a, b, c) = c(x, y, z)$ => $a = x; b = y; c = z$
- **conflict**: $c(x_0, .., x_N) = d(y_0, .., y_N)$ => ошибка
- **swap**: $c(x_0, .., x_N) = z$ => $z = c(x_0, .., x_N)$
- **eliminate**: $x = t$ если x не содержится в t, но содержится в других уравнениях

  => подставляем t вместо x во всех остальных уравнениях
- **check**: $t = f(x_0, .., x_N)$ и t содержится в x_0,..,x_N => ошибка

$O(N)$, но рекурсивные уравнения **запрещены**

Сведение проверки типов к унификации

`int`	int
$[\![\mathsf{V}]\!]$	α
$\&\tau$	$\mathrm{ptr}(\tau)$
$(x_0,..,x_N)$ -> y	$\mathrm{fun}N(x_0,..,x_N,y)$

Упражнение:

попробуйте применить к программе выше линейный алгоритм

Рекурсивные структуры данных

```
var p;
p = alloc;
*p = p;
```

$[\![p]\!] = \&\alpha$

$[\![p]\!] = \&[\![p]\!]$ OOPS!

Решение:

$[\![p]\!] = \alpha$ where $\alpha = \&\alpha$

Рекурсивные структуры данных

```
var p;
p = alloc;
*p = p;
```

$[\![\texttt{p}]\!] = \&\alpha$

$[\![\texttt{p}]\!] = \&[\![\texttt{p}]\!]$ OOPS!

Решение:

$[\![\texttt{p}]\!] = \alpha$ where $\alpha = \&\alpha$

Есть специальная нотация для таких типов:

$[\![\texttt{p}]\!] = \mu\alpha.\&\alpha$

Регулярная унификация

- Позволяет *регулярные* рекурсивные термы
  - Как раз те, которые можно записать через μ-нотацию
- Хьюет, 1976
- $O(N * A(N))$, где $A(N)$ это обратная функция Аккермана
  - $A(n)$ это такое наименьшее k, что $Ack(k, k) = n$
  - $Ack(2, 4) = 2^{65536} - 3$, так что переживать не о чем
- На основе Union-Find

Кто знает, что такое Union-Find?

Union-Find

- Галлер и Фишер, 1964
- Система непересекающихся подмножеств
- Эффективные (за амортизированные $O(A(N))$)
    - Поиск подмножества по элементу
    - Объединения двух подмножеств
- Очень проста в реализации

Примерчик посложнее

```
foo(p,x) {
  var f,q;
  if(*p==0) {
    f=1;
  } else {
    q = alloc;
    *q = (*p)-1;
    f = (*p)*((x)(q,x));
  }
  return f;
}
```

```
main() {
  var n;
  n = input;
  return foo(&n,foo);
}
```

Уравнения

$[\![*\mathtt{p} == 0]\!] = \mathtt{int}$

$[\![\mathtt{p}]\!] = \&[\![*\mathtt{p}]\!]$

$[\![0]\!] = \mathtt{int}$

$[\![\mathtt{f}]\!] = [\![1]\!]$

$[\![1]\!] = \mathtt{int}$

$[\![\mathtt{q}]\!] = \&\alpha_1$

$[\![\mathtt{q}]\!] = \&[\![*\mathtt{p} - 1]\!]$

$[\![*\mathtt{p} - 1]\!] = [\![*\mathtt{p}]\!] = [\![1]\!]$

$[\![\mathtt{f}]\!] = [\![*\mathtt{p} * ((\mathtt{x})(\mathtt{q}, \mathtt{x}))]\!]$

$[\![*\mathtt{p} * ((\mathtt{x})(\mathtt{q}, \mathtt{x}))]\!] = [\![*\mathtt{p}]\!] = [\![((\mathtt{x})(\mathtt{q}, \mathtt{x}))]\!]$

$[\![\mathtt{x}]\!] = ([\![\mathtt{q}]\!],[\![\mathtt{x}]\!]) \to [\![(\mathtt{x})(\mathtt{q}, \mathtt{x})]\!]$

$[\![\mathtt{foo}]\!] = ([\![\mathtt{p}]\!],[\![\mathtt{x}]\!]) \to [\![\mathtt{f}]\!]$

$[\![\mathtt{n}]\!] = \alpha_2$

$[\![\&\mathtt{n}]\!] = \&[\![\mathtt{n}]\!]$

$[\![\mathtt{foo}]\!] = ([\![\&\mathtt{n}]\!],[\![\mathtt{foo}]\!]) \to [\![\mathtt{foo}(\&\mathtt{n}, \mathtt{foo})]\!]$

$[\![\mathtt{main}]\!] = () \to [\![\mathtt{foo}(\&\mathtt{n}, \mathtt{foo})]\!]$

Решение

- Отображение выражений в их типы
- Типы храним в UF
- Для каждого уравнения унифицируем
- При унификации подтягиваем вверх нетривиальные типы
- Осторожнее с рекурсией при извлечении результатов!

$[\![\texttt{p}]\!]$ = `&int`

$[\![\texttt{q}]\!]$ = `&int`

$[\![\texttt{alloc}]\!]$ = `&int`

$[\![\texttt{x}]\!]$ = $\mu\psi.(\texttt{\&int}, \psi)$ `-> int`

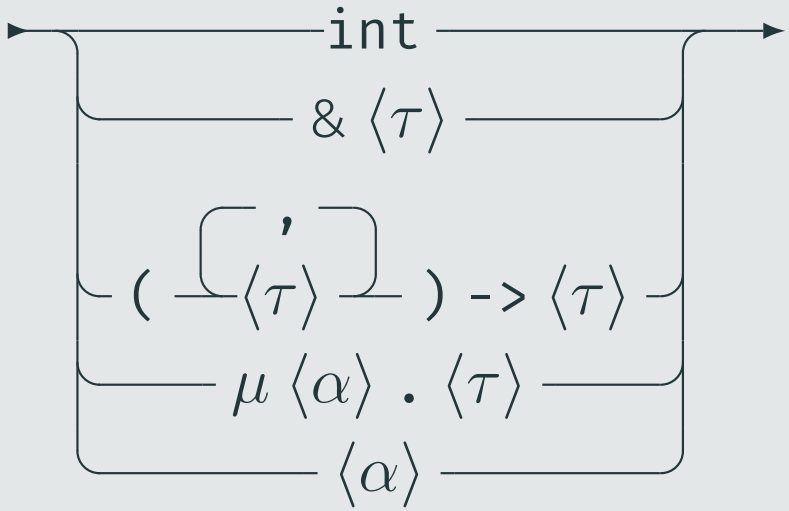
$[\![\texttt{foo}]\!]$ = $\mu\psi.(\texttt{\&int}, \psi)$ `-> int`

$[\![\texttt{main}]\!]$ = () `-> int`

Всё остальное — `int`

К чему в итоге пришли?

$\langle \tau \rangle$



где $\langle \alpha \rangle$ — это *типовая переменная*

Beyond simple type systems

Куда можно развивать простую систему типов

- Subtyping
- Generic types
- Дальше — тёмный лес
    - Dependent types
    - Liquid types
    - HoTT

Хотите знать больше?

Benjamin C. Pierce, **Types in programming languages**

Сегодня мы многое поняли

- Анализ типов — это статический анализ
  - Вернее, просто формализм для статического анализа
- Рассмотрели несколько методов унификации
  - Унификация применяется не только в анализе типов

Далеко не все ошибки являются ошибками типов:

- Разыменование нулей
- Чтение неинициализированных переменных
- Утечки памяти

Другие виды статического анализа умеют с этим работать

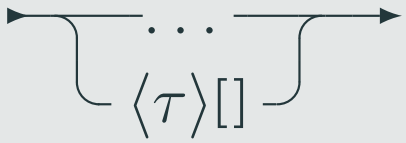
На подумать

- Что будет, если в нашу систему ввести тип Bool?
  - Попробуйте переписать все правила подходящим образом
  - Будет ли анализ
    - более полным?
    - более точным?

На подумать

- Что будет, если в нашу систему ввести тип Array?

$\langle \tau \rangle$



На подумать

- Придумайте правила вывода для новых операторов
- Попробуйте протипизировать программу со слайда

```
main() {
  var x,y,z,t;
  x = {2,4,8,16,32,64};
  y = x[x[3]];
  z = {{},x};
  t = z[1];
  t[2] = y;
}
```

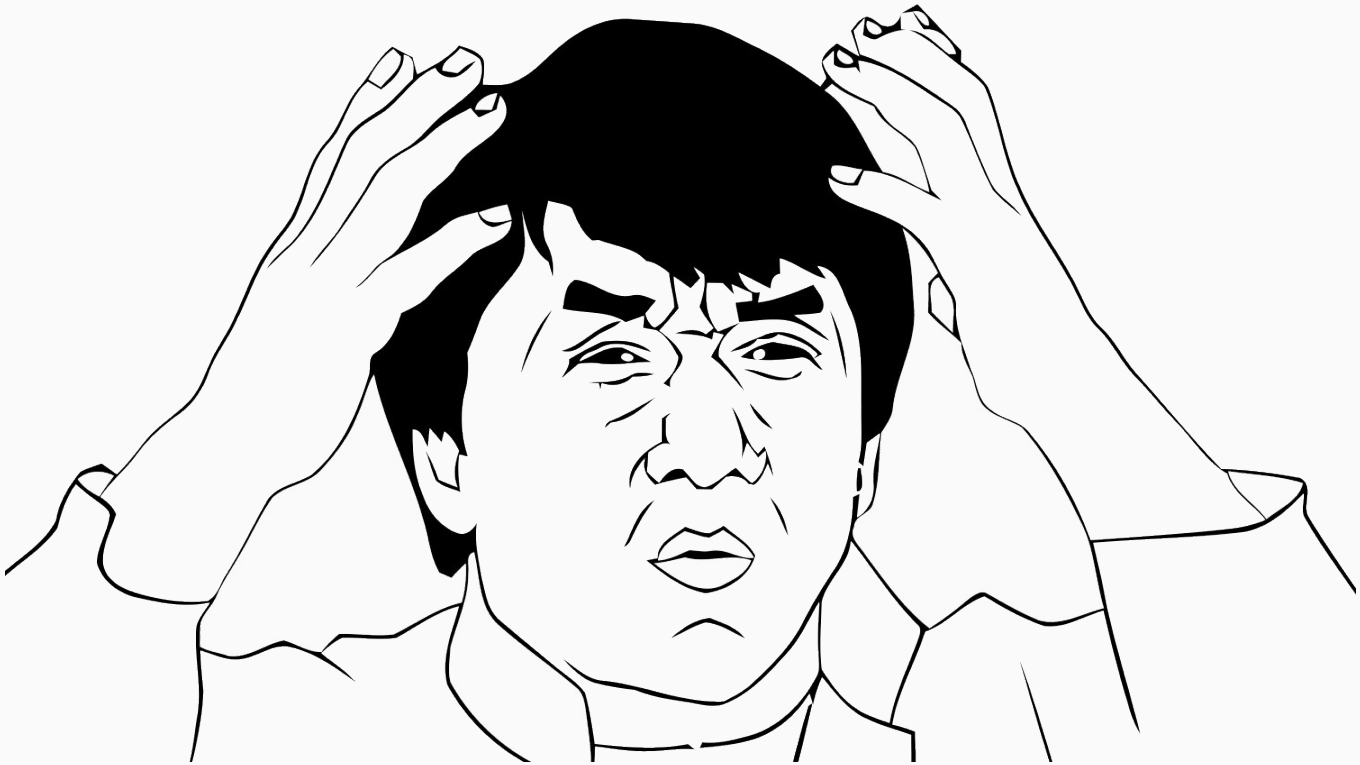
На поделать

- Дописать реализацию класса `TypeAnalysis`[1] в TIP
  (паттерн матчинг — ваш друг!)
- Подумайте, что происходит в получившейся реализации, если в программе есть рекурсивный тип?

[1] src/tip/analysis/TypeAnalysis.scala

В следующей серии

Что делать, если нам нужно не только то, **что** в выражении написано, но и **где** оно написано???



belyaev@kspt.icc.spbstu.ru

akhin@kspt.icc.spbstu.ru