Специализация программ с интерпретативным поведением и искусственный интеллект

Виталий Худобахшов

Факультет свободных искусств и наук
Санкт-Петербургский Государственный Университет
vitaly.khudobakhshov@gmail.com

1 июля 2013

Метавычисления

«Метавычисления – это раздел теории и практики программирования, посвященный разработке методов анализа и преобразования программ за счет реализации конструктивных метасистем (метапрограмм) над программами. В метавычисления в первую очередь включают теорию суперкомпиляции и близкие методы и средства».

— С. М. Абрамов «Метавычисления и их применение»

Специализация

Пусть дана программа, принимающая два параметра $p(x, y)$, такая что результатом ее выполнения является r. В этом случае будем писать $r = p(x, y)$. Тогда программа специализатор *spec* определяется соотношением:

$$p(x, y) = spec(p, x)(y),$$

где $p|_x = spec(p, x)$ – это специализированный по первому аргументу вариант программы p, так что $p(x, y) = p|_x(y)$ для всех y.

Оптимизация

Специализатор *spec* производит глубинные преобразования программы p при фиксированном x, так что

$$t_{p|_x}(y) < t_{p(x,y)}$$

Проекции Футамуры

- Интепретатор языка L (написанный на R) – это программа, для которой выполняется следующее соотношение:

$$r = p_L(d) = intL(p_L, d)$$

- Применяя специализатор к интерпретатору первый раз получаем

$$intL(p_L, d) = spec(intL, p_L)(d),$$

здесь $spec(intL, p_L)$ – результат компиляции программы из L в R

- Компилятор из L в R получается самоприменением специализатора:

$$spec(intL, p_L)(d) = spec(spec, intL)(p_L)(d)$$

- Также можно получить генератор компиляторов

$$spec(spec, intL)(p_L)(d) = spec(spec, spec)(intL)(p_L)(d).$$

Самоприменение специализатора

Если у нас есть специализатор языка R написанный на языке L, то можно построить самоприменимый специализатор для языка R следующим образом

$$specR_R = specR_L(intL_R, specR_L)$$

Предиктор и модель

Рассмотрим программу $pred : E \times W \to W$, которая принимает в качестве первого аргумента эмпирические знания о мире, т.е. таблицу, которая сопоставляет состоянию мира $w \in W$ следующее состояние $w' \in W$, и текущие состояние мира в качестве второго аргумента. Применяя специализацию получим

$$pred(e, w) = spec(pred, e)(w),$$

где $spec(pred, e)$ представляет собой модель мира. Вторая проекция дает генератор моделей:

$$spec(spec, pred).$$

Более интересно, что генератор генераторов моделей полностью совпадает с генератором компиляторов:

$$spec(spec, spec)$$

Пример

Рассмотрим программу *imp*, которая улучшает программу m – модель мира согласно данным о двух состояниях (таким образом предыдущий опыт аккумулируется в модели):

$$imp(m, w, w') = \begin{cases} m, & m(w) = w' \\ m' : m'(w) = w', & m(w) \neq w' \end{cases}$$

С одной стороны *imp* – это метапрограмма, так как она работает с программой m, с другой стороны неясно можно ли применить *imp* к себе?

Инверсия моделей

Если язык, на котором написана программа m, снабжен инверсной семантикой, т.е. программы можно обращать, то задача рационального поведения сводится к нахождению $m^{-1}(w_r)$ для искомого состояния w_r.

Самоприменение *imp*

Сделаем следующее обобщение для того чтобы можно было добиться самоприменения *imp*:

$$imp(y, x, x') = \begin{cases} y, & y(x) = x' \\ y' : y'(x) = x', & y(x) \neq x' \end{cases},$$

где x, x' и y – это программы.
Специализируя *imp* по w и w' получим:

$$imp|_{w,w'}(m) = spec(spec(imp, w), w')(m),$$

затем применяя *imp* к частично вычисленной программе $imp|_{w,w'}$ получим:

$$imp' = imp(imp|_{w,w'}, m_1, m_2).$$

Принцип Пифагора для программы *imp*

Последнее уравнение можно трактовать как фундаментальное ограничение процесса «обучения». Прожим, что у нас есть две системы $\mathcal{S}$ и $\mathcal{T}$, тогда

$$imp'_{\mathcal{S}} = imp_{\mathcal{T}}(imp_{\mathcal{S}}|_{w,w'}, m_{\mathcal{S}}, m_{\mathcal{T}})$$

соответствует известному высказыванию Пифагора «*Вы не можете научить кого угодно чему угодно*».

Общий и узкоспециальный ИИ

Рассмотрим следующее уравнение

$$nAI = spec(AGI, pd),$$

где AGI – это общий ИИ (к примеру когнитивная архитектура), а nAI – это узкоспециализированная программа для решения задачи pd. Если выполняется соотношение

$$t_{spec}(AGI, pd) + t_{nAI} < t_{AGI}(pd),$$

то мы будем иметь преимущество в скорости принятия решений (особенно важно для массовых задач).

Спасибо за внимание

Вопросы пожалуйста