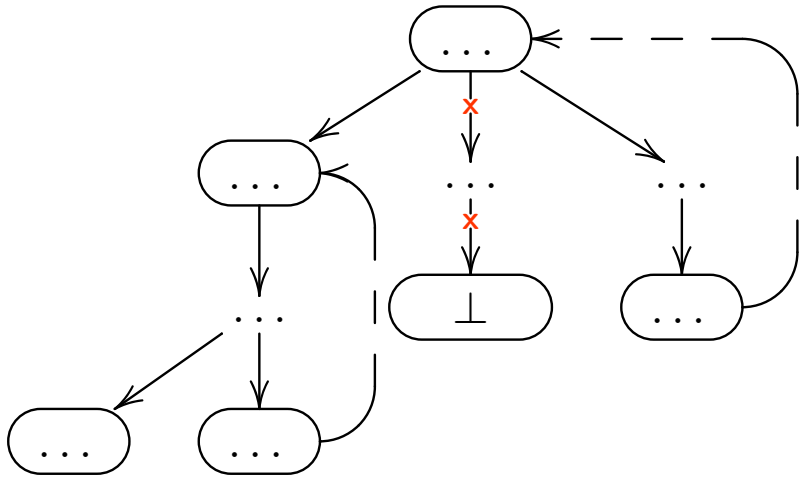
Языки образцов и анализ Рефал-программ

Антонина Непейвода

Рабочее совещание МГТУ им. Н. Э. Баумана и ИПС РАН по Рефалу
12 июня 2020



Определение

Состояние C_1 вкладывается в состояние C_2 графа вычислений программы, если все пути вычислений, порождаемые C_1, могут порождаться также и C_2.

Определение

Состояние C_g является обобщением состояний C_1 и C_2 графа вычислений программы, если все пути вычислений, порождаемые C_1 и C_2, могут порождаться также и C_g.

Наиболее точные, но семантические понятия.

Вложение и обобщение в суперкомпиляции

Определение

Состояние C_1 вкладывается в состояние C_2 графа вычислений программы, если существует подстановка σ такая, что $C_2\sigma = C_1$.

Определение

Состояние C_g является обобщением состояний C_1 и C_2 графа вычислений программы, если существуют подстановки σ_1 и σ_2 такие, что $C_g\sigma_1 = C_1$, $C_g\sigma_2 = C_2$.

- Синтаксическое
- Легко верифицируемое
- Запрещает бесконечные нисходящие цепочки для lisp-подобных данных

Суперкомпиляция Рефал-программ

Наличие ассоциативной конкатенации порождает ряд сложностей.

- Теряется однозначность наилучшего обобщения. У терма T может быть бесконечно много обобщений: $\mathtt{e.x_1}\ \mathtt{T}$, $\mathtt{e.x_1}\ \mathtt{e.x_2}\ \mathtt{T}$, ...
- Теряется свойство конечности цепочек обобщающих выражений: $\mathtt{e.x_1}\ \mathtt{e.x_1}$, $\mathtt{e.x_2}\ \mathtt{e.x_1}\ \mathtt{e.x_1}\ \mathtt{e.x_2}$, ..., $\mathtt{e.x_n}\ \ldots\ \mathtt{e.x_1}\ \mathtt{e.x_1}\ \ldots\ \mathtt{e.x_n}$,...
- Отсутствие подстановки может быть обусловлено формой записи состояния (а не множеством путей вычисления, порождаемом этим состоянием): $\mathtt{e.x\ t.y}$ и $\mathtt{t.y\ e.x}$.

Естественный подход: найти альтернативные конкретизации понятий обобщения и вложения состояний в графе вычислений, подходящие для ассоциативного типа данных.

От состояний к образцам

$\mathcal{V}_\mathcal{T}$ — множество переменных типа $\mathcal{T}$, $\mathcal{V} = \bigcup^{\mathcal{T}} \mathcal{V}_\mathcal{T}$.

Σ — *достаточно большой* алфавит констант.

Определение

Плоский образец P — *строка в алфавите* $\mathcal{V} \cup \Sigma$. *Образец* P *(плоский или нет) линеен, если кратность каждой* e*-переменной в нем равна* 1. *Подстановка* σ — *морфизм из* $(\mathcal{V} \cup \Sigma)^*$ *в* $(\mathcal{V} \cup \Sigma)^*$, *сохраняющий константы (т.е. для всех* $\mathbf{A} \in \Sigma$ $\sigma(\mathbf{A}) = \mathbf{A}$*).*

Пример

Образец $\mathrm{e.x}\,\mathbf{A}\,\mathrm{e.x}$ плоский, но не линейный.

Образец $(\mathrm{e.x_0}\ \mathrm{s.z}\ \mathrm{e.x_1})\ \mathrm{e.x_2}\ \mathrm{s.z}$ — линейный и не плоский.

Языки, распознаваемые образцами

Определение

Языком $\mathscr{L}(\mathrm{P})$, *распознаваемым образцом* P *(англ. — pattern language, сокращенно PL), назовем множество элементов* $\Phi \in \Sigma^*$, *для которых существует подстановка* σ: $\sigma(\mathrm{P}) = \Phi$. *Образец* P_1 *сводится к образцу* P_2, *если* $\mathscr{L}(\mathrm{P}_1) \subseteq \mathscr{L}(\mathrm{P}_2)$. *Плоский образец* P *— краткий, если все образцы* P' *такие, что* $\mathscr{L}(\mathrm{P}) = \mathscr{L}(\mathrm{P}')$, *не короче образца* P.

Если $\sigma(\mathrm{e.x}) = \varepsilon$ допустимо — стирающий (erasing) PL (EPL). Если недопустимо — нестирающий (non-erasing) PL (NePL).

- EPL, распознаваемый образцом-строкой $\mathrm{P} \in \Sigma^*$, есть $\{\mathrm{P}\}$.
- EPL, распознаваемый образцом $\mathrm{P} = \mathrm{e.x_1\ e.x_2\ \ldots\ e.x_n}$, есть все множество Σ^*. NePL, распознаваемый этим образцом — множество слов, имеющих не меньше, чем n букв.

Языки образцов и суперкомпиляция

Рассмотрим состояние дерева вычислений как образец в языке, расширенном конструкторами вызовов функций.

Определение

Состояние C_1 вкладывается в состояние C_2 графа вычислений программы, если $\mathscr{L}(C_1) \subseteq \mathscr{L}(C_2)$.

Определение

Состояние C_g является обобщением состояний C_1 и C_2 графа вычислений программы, если $\mathscr{L}(C_1) \subseteq \mathscr{L}(C_g)$ и $\mathscr{L}(C_2) \subseteq \mathscr{L}(C_g)$.

Полностью согласуются с определением вложения и обобщения состояний в семантическом смысле.

Свойства «образцового» понятия обобщения

- Бесконечные цепочки обобщаемых выражений все еще существуют:

$$\mathscr{L}(\mathrm{e.x_1\ e.x_1}) \subset \mathscr{L}(\mathrm{e.x_2\ e.x_1\ e.x_1\ e.x_2}) \subset \ldots$$
$$\cdots \subset \mathscr{L}(\mathrm{e.x_n\ \ldots\ e.x_1\ e.x_1\ \ldots\ e.x_n})$$

- В случае линейных образцов состояний — такие цепочки невозможны, при условии, что вложение строгое. Невозможно и существование бесконечного множества обобщений состояния, если потребовать, чтобы соответвующие этим обобщениям образцы были краткими.

Отношение Хигмана-Крускала и языки образцов

Типичная ситуация при анализе двух состояний C_1 и C_2 в графе вычислений Рефал-программы: $C_1 \trianglelefteq C_2$, но $\mathscr{L}(C_1) \subseteq \mathscr{L}(C_2)$. Свидетельство того, что предпочтительно обобщать состояние C_2, а не C_1 (либо не трогать его, если образец, соответствующий C_2, линейный и краткий).

Пример

- $<\mathtt{F}\ \mathbf{A}\,\mathtt{e.x}> \trianglelefteq <\mathtt{F}\ \mathtt{e.y}_1\ \mathbf{A}\,\mathtt{e.y}_2>$. Можно спокойно продолжать развертку $<\mathtt{F}\ \mathtt{e.y}_1\ \mathbf{A}\,\mathtt{e.y}_2>$, не боясь, что она будет продолжаться бесконечно.
- $<\mathtt{F}\ \mathbf{A}\,\mathtt{e.x}> \trianglelefteq <\mathtt{F}\ \mathtt{e.y}_1\ \mathtt{e.y}_2>$ — а вот тут обобщать снизу уже имеет смысл, поскольку соответствующий образец не краткий.

А если нет разницы...

В каких случаях $\mathscr{L}(\mathrm{P}_1) \subseteq \mathscr{L}(\mathrm{P}_2)$ влечет существование подстановки σ такой, что $\mathrm{P}_2\sigma = \mathrm{P}_1$?

- Если образец P_2 плоский и не содержит t-переменных.
- Если образец P_2 линейный и не содержит «плавающих» t-переменных.

Пример

Образец $\mathtt{t.y}_1$ **A** $\mathtt{e.x}$ $\mathtt{t.y}_2$ — «плохой», переменная $\mathtt{t.y}_2$ плавающая.

Образец $\mathtt{e.x}_1$ $\mathtt{t.y}$ $\mathtt{t.y}$ $\mathtt{t.z}_1$ $\mathtt{t.z}_2$ $\mathtt{t.z}_3$ $\mathtt{e.x}_2$ $\mathtt{t.y}$ $\mathtt{e.x}_3$ — совсем «плохой», переменные $\mathtt{t.z}_\mathtt{i}$ плавающие и разбивают разные вхождения переменной $\mathtt{t.y}$.

Плавающие t-переменные

Определение

Назовем переменную $\mathtt{t.x}$ *в плоском линейном образце* P *якорной, если*

- $\mathtt{t.x}$ *имеет кратность, не меньшую* 2*;*
- *или в* P *существует подслово* α*, не содержащее* e*-переменных, такое, что* $\alpha = \alpha_1\,\mathtt{t.x}\,\alpha_2$*, причем* α_1 *и* α_2 *оба содержат хотя бы один символ,* s*-переменную, или* t*-переменную, имеющую кратность не меньше* 2*.*

В противном случае назовем $\mathtt{t.x}$ *плавающей.*

Пример

Рассмотрим образец $\mathtt{t.y_1\ t.y_2\ e.x_1\ t.y_3\ t.y_4\ t.y_2\ e.x_2\ t.y_5}$. Якорными переменными являются $\mathtt{t.y_2}$ и $\mathtt{t.y_1}$.

Плавающие переменные и языки образцов

Плавающая переменная в образце — указатель на то, что в соответствующий фрагмент образца нельзя подставить пустое слово. Аналог v-переменных Рефала-4.

Плавающий сегмент образца P — подслово P, содержащее только плавающие t-переменные и e-переменные.

Образец, в котором все e-переменные входят в плавающие сегменты — аналог образца, определяющего NePL. Достаточно заменить e-переменные на v-переменные, и можно искать подстановки. Если это верно только для некоторых e-переменных — получаем образец, определяющий смешанный язык — не NePL, но и не EPL.

Насколько это плохо?

Multi-pattern languages (Kari, Salomaa)

Определение

Языком $\mathscr{L}(\mathrm{P})$, *распознаваемым множеством образцов* P_i *(англ. — multi-pattern language, сокращенно MPL), назовем множество элементов* $\Phi \in \Sigma^*$, *для которых существует* $i \in \mathbb{N}$ *и подстановка* σ: $\sigma(\mathrm{P}_i) = \Phi$.

Множество MPL–объединений EPL совпадает с множеством MPL–объединений NePL. Образец с плавающими `t`-переменными тоже определяет MPL.

Пример MPL в стиле Рефал

Пусть $P_1 = \mathtt{e}.x_1\ \mathtt{AAC}\ \mathtt{e}.x_2\ \mathtt{CAB}\ \mathtt{e}.x_3\ \mathtt{BBC}$,
$P_2 = \mathtt{e}.z_1\ \mathtt{t.n}\ \mathtt{t.n}\ \mathtt{e}.z_2\ \mathtt{t}.m_1\ \mathtt{e}.z_3\ \mathtt{t}.m_2\ \mathtt{e}.z_4\ \mathtt{t}.m_3\ \mathtt{e}.z_5\ \mathtt{t.n}\ \mathtt{e}.z_6$.

Множество образцов NePL, порождающих P_2:

P_2^1	$\mathtt{t.n}\ \mathtt{t.n}\ \mathtt{v}.x_1\ \mathtt{v}.x_2\ \mathtt{v}.x_3\ \mathtt{t.n}$
P_2^2	$\mathtt{t.n}\ \mathtt{t.n}\ \mathtt{v}.x_1\ \mathtt{v}.x_2\ \mathtt{v}.x_3\ \mathtt{t.n}\ \mathtt{v}.x_4$
P_2^3	$\mathtt{v}.x_0\ \mathtt{t.n}\ \mathtt{t.n}\ \mathtt{v}.x_1\ \mathtt{v}.x_2\ \mathtt{v}.x_3\ \mathtt{t.n}$
P_2^4	$\mathtt{v}.x_0\ \mathtt{t.n}\ \mathtt{t.n}\ \mathtt{v}.x_1\ \mathtt{v}.x_2\ \mathtt{v}.x_3\ \mathtt{t.n}\ \mathtt{v}.x_4$

Множество образцов NePL, порождающих P_1, и обобщающие их подобразцы из P_2:

A A C C A B B B C	$P_2^3\sigma_1,\ \mathtt{t.n}\sigma_1 = \mathbf{C}$
A A C C A B $\mathtt{v}.z_3$ **B B C**	$P_2^3\sigma_2,\ \mathtt{t.n}\sigma_2 = \mathbf{C}$
A A C $\mathtt{v}.z_2$ **C A B B B C**	$P_2^2\sigma_3,\ \mathtt{t.n}\sigma_3 = \mathbf{A}$
A A C $\mathtt{v}.z_2$ **C A B** $\mathtt{v}.z_3$ **B B C**	$P_2^2\sigma_4,\ \mathtt{t.n}\sigma_4 = \mathbf{A}$
$\mathtt{v}.z_1$ **A A C C A B B B C**	$P_2^3\sigma_5,\ \mathtt{t.n}\sigma_5 = \mathbf{C}$
$\mathtt{v}.z_1$ **A A C C A B** $\mathtt{v}.z_3$ **B B C**	$P_2^3\sigma_6,\ \mathtt{t.n}\sigma_6 = \mathbf{C}$
$\mathtt{v}.z_1$ **A A C** $\mathtt{v}.z_2$ **C A B B B C**	$P_2^4\sigma_7,\ \mathtt{t.n}\sigma_7 = \mathbf{A}$
$\mathtt{v}.z_1$ **A A C** $\mathtt{v}.z_2$ **C A B** $\mathtt{v}.z_3$ **B B C**	$P_2^4\sigma_8,\ \mathtt{t.n}\sigma_8 = \mathbf{A}$

Неплоские образцы и размер алфавита

Все хорошие свойства образцов, позволяющие работать с ними обычными методами суперкомпиляции (поиск подстановки) — следствие того, что мы подразумеваем $|\Sigma| = O(|\Sigma_{\texttt{Prog}}|^2)$, где Σ — алфавит входных данных, $\Sigma_{\texttt{Prog}}$ — множество символов, явно входящих в левые части Рефал-программы.

Если разрешить вхождения структурных скобок в образцы — это допущение пропадает. Для любого значения, подставляемого в $\texttt{e.x}$, истинна одна из альтернатив: $\texttt{e.x} \to \texttt{s.n e.x}'$, $\texttt{e.x} \to (\texttt{e.x}'_1)\texttt{e.x}'_2$, $\texttt{e.x} \to \varepsilon$. Так неявно задается алфавит входных данных.

Сравнение языков неплоских образцов

Определим класс образцов $\mathbf{P}_t$ следующим образом.

- $\varepsilon \in \mathbf{P}_t$, $\mathtt{e.x} \in \mathbf{P}_\mathtt{t}$, $\mathtt{s.x} \in \mathbf{P}_\mathtt{t}$, $(\mathtt{e.x}) \in \mathbf{P}_\mathtt{t}$;
- если $\mathrm{P}_1 \in \mathbf{P}_t$, $\mathrm{P}_2 \in \mathbf{P}_t$, то $\mathrm{P}_1\,\mathrm{P}_2 \in \mathbf{P}_t$.

В классе $\mathbf{P}_t$ могут быть нелинейные образцы.

Предложение (Следствие теоремы Jiang-a для EPL)

Для образцов из $\mathbf{P}_t$ задача сводимости неразрешима.

Неплоские линейные образцы

Предложение (следствие теоремы Kari–Salomaa для линейных MPL)

Если образцы P_1 *и* P_2 *— линейные и не содержат повторных* `t`*-переменных, задача сводимости* ($\mathscr{L}(\mathrm{P}_1) \subseteq \mathscr{L}(\mathrm{P}_2)$) *для них разрешима.*

Здесь перебор всех подобразцов для поиска подстановок уже не помогает.

Пример

Вложение $\mathscr{L}((\mathtt{e.w}_1)\,\mathtt{e.w}_2\,\mathtt{s.z}\,\mathtt{e.w}_3) \subset \mathscr{L}(\mathtt{e.x}_1\,(\mathtt{e.x}_2)\,\mathtt{s.z}\,\mathtt{e.x}_3)$ выполнено, но построить соответствующие этому вложению подстановки можно, только если доказать следующее тождество для $\mathtt{e.w}_2$:

$$\mathtt{e.w}_2 = \mathtt{s.n}\,\mathtt{e.y} \vee \mathtt{e.w}_2 = \mathtt{e.y}_1\,(\mathtt{e.y}_2) \vee \mathtt{e.w}_2 = \varepsilon \vee \mathtt{e.w}_2 = \mathtt{e.y}_1\,(\mathtt{e.y}_2)\,\mathtt{s.n}\,\mathtt{e.y}_3.$$

Finite subword characterization

Если линейный образец MPL можно представить как объединение образцов вида $e.x\,\mathtt{T}\,e.y$, где $\mathtt{T}$ не содержит e-переменных на внешнем скобочном уровне, скажем, что такой образец имеет finite subword characterization (FSC).

Предложение

Задача поиска FSC для линейных образцов из MPL разрешима.

Выводы

- Многие частные (и часто используемые) случаи анализа Рефал-программ поддаются анализу на вложение языков образцов.
- Вложение состояний можно осуществлять не только по одной, а по нескольким подстановкам (возможно, с порождением вспомогательной функции в остаточной программе).
- При построении обобщений, включающих повторные вхождения e-переменных, открыт большой простор для исследований и импровизации.