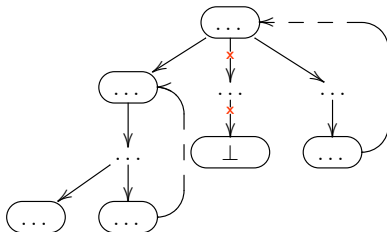


Языки образцов и анализ Рефал-программ

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 $C_1\sigma = C_2$.

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 термина T может быть бесконечно много обобщений: $e.x_1 T$, $e.x_1 e.x_2 T$, \dots
- Теряется свойство конечности нисходящих цепочек по обобщению: $e.x_1 e.x_1$, $e.x_2 e.x_1 e.x_1 e.x_2$, \dots , $e.x_n \dots e.x_1 e.x_1 \dots e.x_n, \dots$
- Отсутствие подстановки может быть обусловлено формой записи состояния (а не множеством путей вычисления, порождаемом этим состоянием): $e.x t.y$ и $t.y e.x$.

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

От состояний к образцам

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

Σ — достаточно большой алфавит констант.

Определение

Плоский образец P — строка в алфавите $\mathcal{V} \cup \Sigma$. Образец P (плоский или нет) линейен, 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 $A \in \Sigma$ $\sigma(A) = A$).

Пример

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

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

Языки, распознаваемые образцами

Определение

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

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

- EPL, распознаваемый образцом-строкой $P \in \Sigma^*$, есть $\{P\}$.
- EPL, распознаваемый образцом $P = e.x_1 e.x_2 \dots e.x_n$, 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 $\mathcal{L}(C_1) \subseteq \mathcal{L}(C_2)$.

Определение

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

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

Свойства «образцового» понятия обобщения

- Бесконечные нисходящие цепочки все еще существуют:

$$\mathcal{L}(e.x_1 e.x_1) \subset \mathcal{L}(e.x_2 e.x_1 e.x_1 e.x_2) \subset \dots \\ \dots \subset \mathcal{L}(e.x_n \dots e.x_1 e.x_1 \dots e.x_n)$$

- В случае линейных образцов состояний — такие цепочки невозможны, при условии, что вложение строгое. Невозможно и существование бесконечного множества обобщений состояния, если потребовать, чтобы соответствующие 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: $C_1 \sqsubseteq C_2$, но $\mathcal{L}(C_1) \subseteq \mathcal{L}(C_2)$.

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

- $\langle F \mathbf{A} e.x \rangle \sqsubseteq \langle F e.y_1 \mathbf{A} e.y_2 \rangle$. Можно спокойно продолжать развертку $\langle F e.y_1 \mathbf{A} e.y_2 \rangle$, не боясь, что она будет продолжаться бесконечно.
- $\langle F \mathbf{A} e.x \rangle \sqsubseteq \langle F e.y_1 e.y_2 \rangle$ — а вот тут обобщать снизу уже имеет смысл, поскольку соответствующий образец не краткий.

А если нет разницы...

В каких случаях $\mathcal{L}(P_1) \subseteq \mathcal{L}(P_2)$ влечет существование подстановки σ такой, что $P_2\sigma = P_1$?

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

Пример

Образец $t.y_1 \mathbf{A} e.x t.y_2$ — «плохой», переменная $t.y_2$ плавающая.

Образец $e.x_1 t.y t.y t.z_1 t.z_2 t.z_3 e.x_2 t.y e.x_3$ — совсем «плохой», переменные $t.z_i$ плавающие и разбивают разные вхождения переменной $t.y$.

Плавающие t -переменные

Определение

Назовем переменную $t.x$ в плоском линейном образце P якорной, если

- $t.x$ имеет кратность, не меньшую 2;
- или в P существует подслово α , не содержащее e -переменных, такое, что $\alpha = \alpha_1 t.x \alpha_2$, 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 $t.x$ плавающей.

Пример

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

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 $\mathcal{L}(P)$, 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 : $\sigma(P_i) = \Phi$.

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

Пример MPL в стиле Рефал

Пусть $P_1 = e.x_1 AAC e.x_2 CAB e.x_3 BBC$,

$P_2 = e.z_1 \text{ t.n t.n e.z}_2 \text{ t.m}_1 \text{ e.z}_3 \text{ t.m}_2 \text{ e.z}_4 \text{ t.m}_3 \text{ e.z}_5 \text{ t.n e.z}_6$.

Множество образцов NePL, порождающих P_2 :

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

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

AACCBVBBC	$P_2^3 \sigma_1, t.n \sigma_1 = C$
AACCAV_{v.z₃}BBC	$P_2^3 \sigma_2, t.n \sigma_2 = C$
AAC_{v.z₂}CABVBBC	$P_2^2 \sigma_3, t.n \sigma_3 = A$
AAC_{v.z₂}CAB_{v.z₃}BBC	$P_2^2 \sigma_4, t.n \sigma_4 = A$
v.z₁ AACCBVBBC	$P_2^3 \sigma_5, t.n \sigma_5 = C$
v.z₁ AACCAV_{v.z₃}BBC	$P_2^3 \sigma_6, t.n \sigma_6 = C$
v.z₁ AAC_{v.z₂}CABVBBC	$P_2^4 \sigma_7, t.n \sigma_7 = A$
v.z₁ AAC_{v.z₂}CAB_{v.z₃}BBC	$P_2^4 \sigma_8, t.n \sigma_8 = A$

Неплоские образцы и размер алфавита

Все хорошие свойства образцов, позволяющие работать с ними обычными методами суперкомпиляции (поиск подстановки) — следствие того, что мы подразумеваем $|\Sigma| = O(|\Sigma_{\text{Prog}}|)$, где Σ — алфавит входных данных, Σ_{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 $e.x$, истинна одна из альтернатив: $e.x \rightarrow s.n e.x'$, $e.x \rightarrow (e.x'_1)e.x'_2$, $e.x \rightarrow \varepsilon$. Так неявно задается алфавит входных данных.

Сравнение языков неплюских образцов

Определим класс образцов P_t следующим образом.

- $\varepsilon \in P_t$, $e.x \in P_t$, $s.x \in P_t$, $(e.x) \in P_t$;
- если $P_1 \in P_t$, $P_2 \in P_t$, то $P_1 P_2 \in P_t$.

В классе P_t могут быть нелинейные образцы.

Предложение (Следствие теоремы Jiang-а для EPL)

Для образцов из 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 ($\mathcal{L}(P_1) \subseteq \mathcal{L}(P_2)$) для них разрешима.

Здесь перебор всех подобразцов для поиска подстановок уже не помогает.

Пример

Вложение $\mathcal{L}((e.w_1) e.w_2 s.z e.w_3) \subseteq \mathcal{L}(e.x_1 (e.x_2) s.z e.x_3)$ выполнено, но построить соответствующие этому вложению подстановки можно, только если доказать следующее тождество для $e.w_2$:

$$e.w_2 = s.n e.y \vee e.w_2 = e.y_1 (e.y_2) \vee e.w_2 = \varepsilon \vee e.w_2 = e.y_1 (e.y_2) s.n e.y_3.$$

Finite subword characterization

Если линейный образец MPL можно представить как объединение образцов вида $e.xTe.y$, где 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.

- 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 е-переменных, открыт большой простор для исследований и импровизации.