1 Closure of a binary relation relative to some property, uniqueness of a closure

Определение

Примерами свойств отношений являются: рефлексивность, симметричность, транзитивность.

Дано бинарное отношение $r \subseteq A^2$ и свойство $\mathcal{P}$, назовём бинарное отношение r^* **замыканием** r относительно $\mathcal{P}$, тогда и только тогда, когда выполнены следующие условия:

- $r \subseteq r^*$
- r^* обладает свойством $\mathcal{P}$
- для любого другого r' такого, что r' обладает $\mathcal{P}$ и $r \subseteq r'$, $r^* \subseteq r'$

Предложение

Для любого бинарного отношения $r \subseteq A^2$ и свойства $\mathcal{P}$ верно следующее: если замыкание r относительно $\mathcal{P}$ существует, оно единственно и совпадает с множеством

$$cl_{\mathcal{P}}(r) = \bigcap \{r' | r \subseteq r' \text{ и } r' \text{ обладает } \mathcal{P}\}$$

Доказательство

Единственность. Предположим, что существует другое замыкание r^{**} r относительно $\mathcal{P}$. Тогда, поскольку $r \subseteq r^*$ и r^* обладает $\mathcal{P}$, по определению замыкания, $r^{**} \subseteq r^*$. С другой стороны, используя определение r^*, можно получить обратное включение: $r^* \subseteq r^{**}$. Тогда $r^{**} = r^*$. Теперь предположим, что r' существует. Чтобы доказать вторую часть, проверим два включения: $cl_{\mathcal{P}}(r) \subseteq r^*$ и $r^* \subseteq cl_{\mathcal{P}}(r)$. Первое верно, потому что r^* принадлежит пересечению, второе верно, потому что r^* минимальный элемент этого пересечения.

2 Y-combinator, it's properties

Y комбинатор

Рассмотрим комбинатор $Y = \lambda h.(\lambda x.h(xx))(\lambda x.h(xx))$: для любого x верно следующее:

$$Yx \equiv x(Yx)$$

Этот комбинатор называется комбинатором **неподвижной точки**.

Проверим, что $a(Ya)$ действительно сводится к Ya:

$$(1)\ a(Ya) = a((\lambda h.(\lambda x.h(xx))(\lambda x.h(xx)))a) \Rightarrow$$

$$a((\lambda x.a(xx))\lambda x.a(xx))$$

$$(2)\ Ya = (\lambda x.a(xx))\lambda x.a(xx) \Rightarrow$$

$$a((\lambda x.a(xx))\lambda x.a(xx))$$

Итак, оба Ya и $a(Ya)$ сводятся к одному и тому же λ-терму, следовательно, они эквивалентны.

3 Semantics for a programming language: semantics of data types and operational semantics

Математическая семантика типов данных

В общем случае, для данной системы типов, её **математическая семантика** состоит из:

- для каждого типа τ множество всех значений D_τ - **область допустимых значений** типа τ
- для каждого арифметического оператора $\star$, действующего на типы $\tau_1, \ldots, \tau_n$ и возвращающего значение типа τ_0, **интерпретация** этого оператора, а именно отображение $\star : D_{\tau_1} \times \ldots \times D_{\tau_n} \to D_{\tau_0}$

Семантика типов в SPL

В SPL существует только один тип: int, следовательно, его область определения - это просто множество целых чисел Z. Все арифметические операторы: $+, \cdot, -, /$ интерпретируются как стандартные операции над целыми числами, аналогично для равенства и отношений $<, \leq, >, \geq$. Объединяя множество и все интерпретации, получим алгебраическую структуру $\mathbb{Z} = (Z, \{+, \cdot, -, /, <\})$ в качестве семантики для типов данных в SPL.

Состояние программы

Дана SPL программа π, её **состояние** - это отображение:

$$s : V(\pi) \to D_{int} = Z$$

Если существует больше одного типа, переменные становятся типизированными, и это отображение должно учитывать их типизацию. Множество всех состояний программы π обозначается как $SP(\pi)$.

Семантика операций

Дана программа π, её **семантика** $< \pi >$ - это отображение (частичное!)

$$< \pi >: SP(\pi) \to SP(\pi)$$

Значение: начиная с состояния s_0, программа π либо не завершится, и в этом случае $< \pi > (s_0)$ не определена, либо завершится с состоянием s_1, и в этом случае $< \pi > (s_0) = s_1$.

Семантика операций в SPL

Дана программа π, её семантика $< \pi >$ определяется индукцией по построению π:

1. $< x := e > (s) = s^x_{e[s]}$
2. $< \pi; \rho > (s) =< \pi > \circ < \rho > (s) =< \rho > (< \pi > (s))$
3. $< if(cond)then\{\pi\}else\{\rho\} > (s) = \begin{cases} < \pi > (s) & , s \models cond \\ < \rho > (s) & , s \not\models cond \end{cases}$

4. $< while(cond)do\{\pi\} > (s) = \underbrace{< \pi > \circ \ldots \circ < \pi >}_{n}(s) = s'$, где n - (минимальный) такой, что $s' \not\models cond$ для всех $k < n \underbrace{< \pi > \circ \ldots \circ < \pi >}_{k}(s) \models cond$