Домашние задания

Курс «Теории типов»

1. Лекция 1. Языки и алгебраические теории

2. Лекция 2. System T, NbE и «Диалектика»

2.1. Общие соглашения

В части I выпишите типизированные термы System T; при желании проверьте их в OCaml. В части II закодируйте примеры конструкторами `tm` и запустите нормализатор на OCaml; вместо этого можно использовать индексированные термы Rocq. Часть III выполните в Rocq.

Используется вариант System T из лекции:

$$T ::= \mathbb{N} \mid \mathbb{B} \mid \mathbb{1} \mid T \to T \mid T \times T.$$

Рекурсор имеет тип

$$\mathsf{iterate}(n)\,\mathrm{step}\,\mathrm{base} : T, \qquad \mathrm{step} : \mathbb{N} \to T \to T,$$

и вычисляется по правилам

$$\begin{aligned} \mathsf{iterate}(0)\; s\; z &\rightsquigarrow z, \\ \mathsf{iterate}(\mathrm{S}\,n)\; s\; z &\rightsquigarrow s\; n\; (\mathsf{iterate}(n)\; s\; z). \end{aligned}$$

Аргументы всюду каррированы, а стрелка ассоциируется вправо. Число 2 служит сокращением для $\mathrm{S}(\mathrm{S}\,0)$.

Сложение зафиксируем в том же порядке аргументов, что и в лекции:

$$\mathrm{add}\,m\,n := \mathsf{iterate}(n)(\lambda k\, r.\, \mathrm{S}\,r)\,m.$$

2.2. Часть I. Программирование в System T

Определения ниже должны быть термами System T и комбинировать λ-абстракцию, применение, пары, проекции, булевы значения, if и iterate. Для каждого определения напишите сам терм и его тип. Для исполнимой проверки используйте готовые типы `ty`, `tm`, `nf` и функцию `norm` из `reference/nbe_native.ml`.[1] Можно создать отдельный файл `homework.ml` в корне клона репозитория и загрузить эталонную реализацию:

```
#mod_use "reference/nbe_native.ml";;
open Nbe_native
```

Если выполняете OCaml-проверку, используйте следующее соответствие между

[1] https://github.com/clayrat/nbe-system-t/blob/main/reference/nbe_native.ml

записью лекции и конструкторами эталонного файла:

System T	OCaml	смысл
$\mathbb{N}$, $\mathbb{1}$, $\mathbb{B}$	`TN`, `TUnit`, `TBool`	базовые типы
$A \to B$, $A \times B$	`Tarr(A,B)`, `Tprod(A,B)`	составные типы
0, $\mathrm{S}\,t$	`Tzero`, `Tsuc t`	натуральные числа
x, $\lambda x.\,t$, $f\,a$	`Tvar(T,i)`, `Tlam t`, `Tapp(f,a)`	переменная, функция, применение
$\langle a, b\rangle$, p_{fst}, p_{snd}	`Tpair(a,b)`, `Tfst p`, `Tsnd p`	пары
$\mathsf{if}(c)\,t\,f$	`Tif(T,c,t,f)`	условие с результатом типа T
$\mathsf{iterate}(n)\,s\,z$	`Trec(T,z,s,n)`	рекурсор с результатом типа T

Переменные представлены индексами де Брёйна: `0` обозначает ближайшую переменную, `1` - следующую внешнюю и так далее. Обратите внимание на порядок аргументов `Trec`: после типа результата идут база, шаг и только затем разбираемое число, тогда как в математической записи листка стоят число, шаг и база.

Контексты в заданиях перечисляются от ближайшей переменной к более внешним. Например, в контексте $[f, h]$ переменная f имеет индекс `0`, а h - индекс `1`.

Файл `nbe_native.ml` не содержит типизатора. Вызов `norm len type term` доверяет длине контекста, типу результата, аннотациям `Tvar`/`Trec`/`Tif` и общей корректности дерева.

Определение `tadd` в конце эталонного файла можно использовать как пример порядка конструкторов.

1. Вычитание с усечением

Определите

$$\mathrm{monus} : \mathbb{N} \to \mathbb{N} \to \mathbb{N}, \qquad \mathrm{monus}\, m\, n = \max(m - n, 0).$$

Подсказка: шаг рекурсора получает предшественника счётчика k. Поэтому сначала определите $\mathrm{pred} : \mathbb{N} \to \mathbb{N}$, возвращая k на каждом ненулевом шаге, а затем повторите `pred` нужное число раз. Парный трюк для этого варианта `pred` не нужен.

2. Порядок и минимум

Определите

$$\mathrm{leq} : \mathbb{N} \to \mathbb{N} \to \mathbb{B}, \quad \mathrm{leq}\, m\, n = \mathtt{tt} \quad \text{тогда и только тогда, когда } m \le n,$$
$$\mathrm{min} : \mathbb{N} \to \mathbb{N} \to \mathbb{N}, \qquad \mathrm{min}\, m\, n = \min(m, n).$$

Подсказка: решите, выгоднее ли определять `leq` собственной рекурсией или выразить через уже построенные функции.

Проверьте порядок аргументов на следующих примерах:

$$\mathrm{monus}\,5\,2 = 3, \quad \mathrm{monus}\,2\,5 = 0,$$
$$\mathrm{leq}\,2\,5 = \mathtt{tt}, \quad \mathrm{leq}\,5\,2 = \mathtt{ff},$$
$$\mathrm{min}\,2\,5 = 2, \quad \mathrm{min}\,5\,2 = 2.$$

3. Числа Фибоначчи

Определите $\mathrm{fib} : \mathbb{N} \to \mathbb{N}$ при соглашении $\mathrm{fib}(0) = 0$, $\mathrm{fib}(1) = 1$.

Подсказка: не вычисляйте два предыдущих значения двумя отдельными рекурсивными вызовами. После k шагов храните пару соседних значений $\langle \mathrm{fib}(k), \mathrm{fib}(k+1) \rangle$.

Для OCaml-проверки примените AST вашего замкнутого терма `fib` к `numeral 0`, …, `numeral 10` и сравните нормальные формы с нормализациями соответствующих эталонных числительных. OCaml-код здесь только строит `Tapp` и вызывает `norm`. В Rocq-версии дополнительным результатом может быть лемма о найденном инварианте, доказанная индукцией по n.

4. Функция Аккермана

Используйте соглашение

$$\begin{aligned} A(0, n) &= \mathrm{S}\, n, \\ A(\mathrm{S}\, m, 0) &= A(m, 1), \\ A(\mathrm{S}\, m, \mathrm{S}\, n) &= A(m, A(\mathrm{S}\, m, n)). \end{aligned}$$

Подсказка: для фиксированного m используйте функцию $A_m := \lambda n.\, A(m, n)$ как состояние внешнего рекурсора типа $\mathbb{N} \to \mathbb{N}$. Выразите A_{m+1} через повторное применение A_m.

Проверьте $A(0, 4) = 5$, $A(1, 3) = 5$, $A(2, 2) = 7$ и $A(3, 2) = 29$. Не проверяйте большие аргументы: нормализация быстро становится дорогой.

2.3. Часть II. Полная нормализация и η-длинные формы

Для всех задач на NbE действует равенство по определению из лекции: β-правило, вычисление проекций, if и iterate, а также η-правила для функций и пар. Сначала **не** добавляйте η-правила для базовых типов $\mathbb{N}$, $\mathbb{B}$ и $\mathbb{1}$.

Как считать нормальную форму

Для каждого примера действуйте в следующем порядке.

1. Выпишите контекст Γ и проверьте тип терма.
2. По типу результата предскажите внешний конструктор η-длинной формы: функция должна начинаться с λ, пара --- с $\langle \_, \_ \rangle$.
3. Выполните β-редукцию, проекции и известные шаги if/iterate, в том числе под λ.
4. Если вычисление упёрлось в свободную переменную базового типа, сохраните всю заблокированную элиминацию как нейтральный терм.
5. η-разверните оставшиеся нейтрали функционального и парного типа.

Ответ должен быть β-нормальным и η-длинным. Нейтрали функционального и парного типа η-разворачиваются; заблокированные элиминации над переменными базового типа сохраняются.

5. Что изменит η-правило для $()$?

В лекции $\mathbb{1}$ считается базовым типом, и нейтраль $u : \mathbb{1}$ разрешён как нормальная форма. Исследуйте вариант равенства по определению с правилом уникальности единичного типа.

1. сформулируйте η-правило как суждение из предпосылки $\Gamma \vdash u : \mathbb{1}$;
2. сравните типизированные фрагменты грамматики до и после правила:

$$\mathrm{nf}_\Gamma(\mathbb{1}) ::= () \mid \mathrm{ne}_\Gamma(\mathbb{1}) \qquad \text{и} \qquad \mathrm{nf}_\Gamma(\mathbb{1}) ::= ();$$

   объясните, почему нейтраль больше не входит в нормальную форму этого типа;
3. опишите ветви $\mathrm{reify}_{\mathbb{1}}$ и $\mathrm{reflect}_{\mathbb{1}}$ в NbE с таким правилом;
4. объясните, почему после добавления правила $\mathbb{1}$ ведёт себя как терминальный объект и что происходит с нейтральными термами этого типа.

Для проверки ответа сравните нормальные формы свободной переменной $u : \mathbb{1}$ и терма $g\,0 : \mathbb{1}$ при $g : \mathbb{N} \to \mathbb{1}$ до и после добавления η-правила.

6. Нормальные формы конкретных термов

Посчитайте нормальные формы ниже. Используйте систему лекции **без** η для базовых типов.

1. `tiszero` x в контексте $[x : \mathbb{N}]$, где

$$\texttt{tiszero} := \lambda n.\, \mathsf{iterate}(n)(\lambda k\, r.\, \texttt{ff})\, \texttt{tt}.$$

2. $\lambda x : \mathbb{N}.\, (\lambda y : \mathbb{N}.\, y)\, x$ в пустом контексте.
3. $(\lambda x : \mathbb{N}.\, \lambda z : \mathbb{N}.\, x)\, y$ в контексте $[y : \mathbb{N}]$.
4. Свободная переменная $F : (\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$. Обратите внимание, что η-длинным должен стать не только внешний функциональный терм, но и функциональный аргумент, с которым нейтраль F встретится при обратном чтении.
5. $\mathsf{if}(b)(\lambda x : \mathbb{N}.\, 0)(\lambda x : \mathbb{N}.\, \mathrm{S}\, x)$ в контексте $[b : \mathbb{B}]$. Условие заблокировано, но тип всего условного выражения функциональный.
6. $h\, f$ в контексте $[f : \mathbb{N} \to \mathbb{N},\ h : (\mathbb{N} \to \mathbb{N}) \to \mathbb{N}]$. Результат имеет базовый тип, однако перед построением нейтрального применения NbE должен реифицировать аргумент.
7. Выражение

$$\mathsf{iterate}(2)\, (\lambda \_ : \mathbb{N}.\, \lambda r.\, r)\, (\lambda k : \mathbb{N}.\, \lambda f : \mathbb{N} \to \mathbb{N}.\, \lambda x : \mathbb{N}.\, \mathrm{S}(f\, x)).$$

   Найдите тип состояния T, проверьте, что шаг имеет требуемый тип $\mathbb{N} \to T \to T$, и нормализуйте терм. Укажите полный тип T явно.

Закодируйте каждый пример конструкторами `tm` из `reference/nbe_native.ml` и вызывайте уже предоставленную функцию `norm`. Для замкнутого терма используйте длину контекста `0`; для открытого --- число перечисленных свободных переменных. Отдельно передайте тип всего выражения, например `Tarr (TN, TN)` или `TN`. Сравните полученный `nf` с ожидаемым синтаксическим деревом и распечатайте его через `pp_nf`.

В продвинутом варианте на Rocq постройте те же примеры как элементы индексированного семейства `tm` из `theories/Syntax.v`[2] и вычислите их функцией `norm` из `theories/Model.v`.[3] Исследовательский бонус: реализуйте η-правило для $\mathbb{1}$ и восстановите доказательства здравости и полноты.

[2] https://github.com/clayrat/nbe-system-t/blob/main/theories/Syntax.v
[3] https://github.com/clayrat/nbe-system-t/blob/main/theories/Model.v

2.4. Часть III. Диалектика

Эту часть нужно реализовать в Rocq; OCaml-варианта для неё нет. Используйте упомянутый в лекции репозиторий `dialectica-system-t`[4] и его подмодуль `nbe-system-t`.[5] Первый уже содержит индексированные типы формул и доказательств, трансляцию W/C, матрицу `diaT` и компилятор `wit`; второй предоставляет нормализацию System T.

Для справки, используйте таблицу трансляции из лекции:

A	$W(A)$	$C(A)$
атом	$\mathbb{1}$	$\mathbb{1}$
$X \wedge Y$	$W(X) \times W(Y)$	$C(X) \times C(Y)$
$X \vee Y$	$\mathbb{B} \times (W(X) \times W(Y))$	$C(X) \times C(Y)$
$X \supset Y$	$(W(X) \to W(Y)) \times (W(X) \to C(Y) \to C(X))$	$W(X) \times C(Y)$
$\forall x.\, X$	$\mathbb{N} \to W(X)$	$\mathbb{N} \times C(X)$
$\exists x.\, X$	$\mathbb{N} \times W(X)$	$C(X)$

При дизъюнкции `tt` выбирает левую ветвь, а `ff` - правую. Для импликации матрица сначала запускает обратную компоненту свидетеля, чтобы перенести контрпример к заключению назад к посылке, и затем проверяет

$$|X \supset Y|(\langle f, f^{\leftarrow}\rangle, \langle w, c\rangle) = |X|(w, f^{\leftarrow}\, w\, c) \supset |Y|(f\, w, c).$$

Здесь последняя импликация - разрешимая булева операция над результатами двух матриц.

Что сделать

Для каждой формулы A покажите:

1. формулу `A` и доказательство `d : proof [] A`;
2. типы $W(A)$ и $C(A)$, проверенные равенствами с `reflexivity`;
3. тип и упрощённую запись матрицы `diaT` $A : W(A) \to C(A) \to \mathbb{B}$;
4. терм `wit`$(d) : W(A)$; для импликации выделите прямую и обратную компоненты;
5. `realizer d`: точное синтаксическое дерево либо, для задания 8, читаемую η-длинную схему;
6. применение `soundness_syntactic` и вычисление матрицы на одном замкнутом контрпримере.

Для отрицания используйте $\neg A := A \supset \mathsf{False}$, где False - атом с матрицей `ff`.

7. Два доказательства одного существования

Дважды докажите и реализуйте

$$\exists x.\, (x = 0 \vee x = 1).$$

В первом доказательстве выберите $x = 0$ и левую ветвь, во втором - $x = 1$ и правую. Кодировка дизъюнкции не является типом суммы: её свидетель содержит булев флаг и заполнители для обеих ветвей. Поэтому явно укажите:

- какой флаг выбирает каждую ветвь;
- чем заполнена неиспользуемая единичная компонента;
- почему оба реализатора проходят одну и ту же матрицу;
- являются ли полученные нормальные формы равными по определению.

[4] https://github.com/clayrat/dialectica-system-t/

[5] https://github.com/clayrat/nbe-system-t/

8. Сжатие квантифицированной формулы

Докажите и реализуйте

$$(\forall y.\, y = 0) \supset (\forall y.\, y = 0) \wedge (\forall y.\, y = 0).$$

Подсказка: это частный случай схемы `ax_and_contr`. Сначала вычислите $W(P)$, $C(P)$ и матрицу для $P := \forall y.\, y = 0$, затем разберите прямую и обратную компоненты реализатора сжатия.

Продвинутый бонус: опишите, как η-правило для $\mathbb{1}$ изменит реализатор и какие числовые данные в нём сохранятся.

9. Контрпример универсальному равенству

Докажите и реализуйте

$$\exists x. \neg(\forall y.\, \mathrm{S}(y + y) = x).$$

Подсказка: выберите такое замкнутое x, чтобы универсальное равенство опровергалось одним конкретным y. В реализаторе выбранный y появится в обратной компоненте отрицания как контрпример универсальной формуле. При кодировании тела внутреннего квантора учитывайте порядок де Брёйна: под $\forall y$ переменная y имеет индекс `v0`, а связанная внешним квантором переменная x - индекс `v1`.