1 Пояснения к заданиям

Если будут сложности, тут всякий вспомогательный текст и примеры.

1.1 Исчисление секвенций

Секвенциальное исчисление предикатов, в отличие от обычного исчисления секвенций, не позволяет автоматически осуществить вывод любой тавтологии, хотя исчисление и полное. Из-за этого иногда при выводе в секвенциальном исчислении предикатов приходится думать.

1.1.1 Новые правила

Относительно обычного исчисления секвенций добавились только несколько правил вывода. Разберём их.

$$\frac{A[x := y], \Gamma \vdash \Delta}{\exists x A, \Gamma \vdash \Delta}\ (\exists \vdash)$$

Это правило говорит, что для доказательства справедливости Δ в предположении контекста $\exists x A, \Gamma \vdash \Delta$ достаточно показать, как доказать $A[x := y], \Gamma \vdash \Delta$ для произвольной (возможно, выбранной нашим врагом) переменной y. Идея заключается в том, что если нам дано только то, что A выполняется на *каком-то* x, а мы не знаем, на каком именно, но всё равно смогли доказать Δ, то мы можем разобрать любое $\exists x A$, какое бы нам ни подали, и предоставить доказательство. Это правило идейно совпадает с правилом (В$\exists$) из исчисления предикатов.

$$\frac{\Gamma \vdash \exists x A, \Delta}{\Gamma \vdash A[x := \tau], \Delta}\ (\vdash \exists)$$

Это правило совпадает с аксиомой 13 из исчисления предикатов: чтобы показать, что какой-то x существует, достаточно выбрать конкретный терм, который это демонстрирует.

$$\frac{\forall x A, \Gamma \vdash \Delta}{A[x := \tau], \Gamma \vdash \Delta}\ (\forall \vdash)$$

Здесь нам говорится, что если мы смогли доказать Δ в предположении какого-то $A[x := \tau]$, то уж и подавно нам хватит предположения, что $\forall x A$.

$$\frac{\Gamma \vdash \forall x A, \Delta}{\Gamma \vdash A[x := y], \Delta}\ (\vdash \forall)$$

Это аналог правила (В$\forall$) и говорит ровно то же самое: если у нас получилось доказать $A[x := y]$ для какой-то y, о которой мы ничего не знаем, то, получается, мы можем доказать A для любого x.

1.1.2 Вывод

Творческий процесс в использовании этого исчисления заключается только в понимании, какие экземпляры τ выбрать. Дополнительно надо не забывать делать contraction, когда какое-то высказывание нужно несколько раз.

Несколько примеров для разминки:

1. `http://logitext.mit.edu/proving/.28forall+x.2C+Q.28x.2C+x.29.29+.2D.3E+forall+x.2C+exists+y.2C+Q.28x.2C+y.29`
2. `http://logitext.mit.edu/proving/.28exists+x.2C+P.28x.29+.5C.2F+Q.28x.29.29+.2D.3E+.28exists+x.2C+P.28x.29.29+.5C.2F+.28exists+y.2C+Q.28y.29.29`
3. `http://logitext.mit.edu/proving/.28forall+x.2C+.28P.28g.28x.29.29+.2D.3E+P.28x.29.29.29+.2D.3E+forall+x.2C+.28P.28g.28g.28x.29.29.29+.2D.3E+P.28x.29.29`

Пояснение к последнему такое: надо просто понять, почему оно работает. А именно, потому, что если нам дано $P(g(g(x)))$, то мы можем воспользоваться одним шагом "снятия g" и прийти к $P(g(x))$, а потом ещё раз "снять g" и прийти к $P(x)$.

1.2 Невыразимые предикаты

На лекции вам показали, как можно придумывать формулы логики предикатов для выражения реальных предикатов в заранее выбранных наборах функциональных и предикатных символов (см. раздел про стандартную интерпретацию языка элементарной арифметики). Это полезно, потому что не хочется на каждый мелкий предикат вводить отдельный символ: тогда было бы сложнее доказывать что-то про связь этих предикатов, поскольку требовалось бы вводить на каждую пару предикатов какие-то ограничения на их взаимную выполнимость или невыполнимость.

Но иногда в своём желании минимизировать число символов можно зайти далеко и пытаться выразить невыразимое. В общем случае это сложно понять. В таких случаях надо остановиться и попытаться, напротив, опровергнуть возможность выразить предикат.

Один из способов это сделать — это показать, что мы могли бы каким-то образом *что-то изменить* так, что все имеющиеся свойства продолжили бы сохраняться, а предикат, который мы пытались выразить, сообщил бы какой-то другой результат. Довольно убедительно, что в таком случае предикат действительно невыразимый: если его можно выразить через другие символы, но результат на них не изменился, то почему он сам работает иначе после этого изменения?

Самое простое изменение — это применение автоморфизма. Говорят, что $\alpha : D \to D$ — автоморфизм, если это биекция, которая *сохраняет* все имеющиеся предикатные и функциональные символы. Под сохранением предикатного символа P^n подразумевается, что

$$[P](\alpha(x_1), \alpha(x_2), \cdots, \alpha(x_n)) \Leftrightarrow [P](x_1, x_2, \cdots, x_n)$$

и для функционального символа f^n подразумевается, что

$$[f](\alpha(x_1), \alpha(x_2), \cdots, \alpha(x_n)) = \alpha([f](x_1, x_2, \cdots, x_n))$$

Также говорят, что символ устойчив относительно α.

Здесь использовалась конструкция $[\ ]$ — она означает "интерпретация того, что внутри". Например, $+$ — это просто символ, рисунок крестика, а вот $[+]$ — это уже функция, которая принимает два элемента и возвращает их сумму.

Определение автоморфизма стоит прочитать и на каком-то уровне понять, что оно значит.

1.2.1 Примеры

Найдите автоморфизмы:

1. Сигнатура $(<^2, =^2)$, носитель $\mathbb{Z}$, интерпретация символов обычная. Предикат $x = 0$ невыразим.
2. Сигнатура $(<^2, =^2, +^2)$, носитель $\mathbb{Q}$. Предикат $x = 1$ невыразимый.

   Изменится ли что-то, если заменить носитель на $\mathbb{R}$ или $\mathbb{Z}$?
3. Сигнатура $(0^0, 1^0, <^2, =^2)$, носитель $\mathbb{R}$. Невыразимый предикат — $x = \frac{1}{2}$.

Ответы:

1. $x \mapsto x + 1$. Это действительно автоморфизм: к нему есть обратная функция $x \mapsto x - 1$, а также $x < y \Leftrightarrow x + 1 < y + 1$ и $x = y \Leftrightarrow x + 1 = y + 1$. При этом $P(x) := x = 0$ не сохраняется, поскольку $x + 1 = 0 \Leftrightarrow x = 0$ не является правдой.

2. $x \mapsto 2x$. Действительно, $2x < 2y \Leftrightarrow x < y$, $2x = 2y \Leftrightarrow x = y$ и $2x + 2y = 2(x + y)$, причём имеется обратная функция: $x \mapsto \frac{x}{2}$. Предикат $x = 1$ не сохраняется этим автоморфизмом. Так же невыразим предикат $x = C$, если $C \neq 0$. Предикат $x = 0$ выразим как $x + x = x$.

   Если заменить носитель на $\mathbb{R}$, вывод не изменится. Если же заменить его на $\mathbb{Z}$, появляется проблема: это больше не автоморфизм, так как он не сюръективен.

3. Возьмём автоморфизм $x \mapsto x^3$. Это автоморфизм, потому что кубический корень существует у любого вещественного числа. Возведение в нечётную степень — функция монотонная и сохраняет ноль и единицу. При этом $\frac{1}{2} \neq (\frac{1}{2})^3$.

2 Задания

1. Найдите вывод в секвенциальном исчислении предикатов или контрпример:
   (a) $\exists y \forall x Q(x, y) \to \exists x Q(x, x)$
   (b) $\forall x \exists y (P(x) \to P(y))$
   (c) $\exists x \forall y (P(x) \to P(y))$
   (d) $\exists x (P(y) \vee P(f(z)) \to P(x))$
   (e) $\exists x P(x) \vee \forall y (P(y) \to Q(y))$

2. Найдите автоморфизм, который не сохраняет предикат (интерпретация всех символов обычная):
   (a) Носитель $\mathbb{R}$, символы $(+^2, \leq^2)$, предикат $P(x, y) := y = x^2$. Можно ли найти такой автоморфизм, если носитель заменить на $\mathbb{Z}$?
   (b) Носитель $\mathbb{Z}$, символы $(=^2, +^2, f^2)$ $(f(x, y) \Leftrightarrow x \equiv y \pmod 3)$, предикат $P(x, y) := x + 1 = y$.

3. Приведите пример сигнатуры, интрепретации этой сигнатуры и автоморфизма этой интерпретации таких, что бинарный предикат, являющийся конгруэнцией, не устойчив, или объясните, почему такого не бывает.

4. Большинство доказательств на этом курсе конструктивны, то есть являются также алгоритмами, описывающими, как получить явное свидетельство истинности. Однако известен факт: невозможно написать программу такую, что она для произвольной формулы логики предикатов скажет, тавтологичная ли она, даже используя бесконечную память. Значит, свидетельство полноты исчисления предикатов не может быть получено конструктивно. В каком месте доказательства происходит что-то, что нельзя оформить в виде программы?