Wstęp do rachunku lambda
Część I: rachunek lambda bez typów

Antoni Kościelski

1 Pojęcia wstępne

1.1 Aplikacja

Aplikacją będziemy nazywać działanie, które funkcji f i argumentowi x przypisuje – w przypadku funkcji jednej zmiennej – wartość funkcji f dla argumentu x, czyli coś, co zwykle jest oznaczane symbolem $f(x)$. Działanie to czasem będziemy oznaczać kropką $\cdot$, podobnie jak mnożenie. Zachodzi więc równość $f \cdot x = f(x)$.

Dla funkcji f dwóch zmiennych wynikiem aplikacji $f \cdot x$ będzie funkcja g zdefiniowana wzorem $g(y) = f(x, y)$. W tym przypadku aplikacja ustala pierwszy argument funkcji. Mamy więc następującą równość $(f \cdot x) \cdot y = f(x, y)$. Analogicznie będziemy rozumieć aplikację w przypadku funkcji wielu zmiennych.

Aplikacja jest operacją chyba najczęściej stosowaną we wzorach matematycznych. Jeżeli piszemy $\sqrt{3 + sin^2 x}$, to oczywiście posługujemy się wieloma funkcjami, pierwiastkowaniem, dodawaniem, funkcją sinus, ale w pierwszym rzędzie posługujemy się aplikacją. Aplikujemy pierwiastkowanie do wyniku aplikowania dodawania do liczby 3 i czegoś, co otrzymujemy w wyniku aplikowania podnoszenia do kwadratu do aplikacji sinusa do wartości oznaczanej przez x.

Prawdę mówiąc, mimo że w pewnym sensie będziemy zajmować się aplikacją, sama aplikacja rozumiana jako działanie będzie nam mało potrzebna. Wystarczy symbol $\cdot$, który też będziemy nazywać aplikacją, i który będzie wykorzystywany w sposób wzorowany na własnościach aplikacji.

1.2 Zmienne i stałe

Pisząc wzory posługujemy się między innymi symbolami, które nazywamy zmiennymi. W λ-rachunku też nam będą potrzebne zmienne. Zbiór zmiennych będziemy oznaczać literą V. Zakładamy, że jest to zbiór nieskończony (przeliczalny). Chcemy mieć do dyspozycji każdą skończoną liczbę zmiennych. To, czym są zmienne nie ma większego znaczenia. Ważne jest to, że odróżniamy je od innych rzeczy. Możemy myśleć, że są to znaki, litery lub identyfikatory, z których będziemy budować dłuższe słowa albo napisy. Dla oznaczenia zmiennych będziemy zwykle używać małych liter, w razie potrzeby z jakimiś indeksami. W razie konieczności posługiwania się skończoną liczbą znaków możemy przyjąć, że zmiennymi są napisy złożone z litery v i pewnej liczby primów, na przykład v'''', lub z indeksem będącym przedstawieniem liczby, na przykład v_{113}.

Czasem oprócz zmiennych będą nam potrzebne stałe. Możemy je uważać za szczególne zmienne, o specjalnym przeznaczeniu. Bywa na przykład, że twierdzimy, że $x > 0$. W takim stwierdzeniu x jest chyba zmienną. Domyślamy się, że oznacza

jakąś liczbę, choć zwykle nie wiemy którą. Znak (cyfra) 0 jest zapewne stałą, oznacza ściśle określoną i znaną nam liczbę.

1.3 Termy, czyli wyrażenia z aplikacją

Termy to takie fragmenty wzorów matematycznych, które zwykle znajdują się między symbolami równości. Definiujemy je na kilka sposobów, mogą być drzewami bądź napisami. Najwygodniej definiuje się je jako drzewa, ale najłatwiej przekazuje się informacje o termach będących napisami. Na razie budujemy proste termy ze zmiennych, z symbolu aplikacji i ewentualnie z nawiasów.

1.4 Termy jako drzewa

Termy definiujemy jako drzewa binarne, etykietowane i spełniające dalej podane warunki. Każdy węzeł takiego drzewa ma etykietę, którą jest zmienna lub symbol aplikacji. Węzły etykietowane zmiennymi nie mają potomków. Każdy węzeł z etykietą będącą symbolem aplikacji ma dwa węzły potomne: lewy i prawy.

1.5 Termy jako napisy

Każdy term rozumiany jako drzewo może też być reprezentowany przez pewien napis (czyli słowo). Drzewo z jednym węzłem z etykietą x jest reprezentowane przez zmienną x. Drzewo z korzeniem z symbolem aplikacji i dwoma poddrzewami: lewym, reprezentowanym przez napis t_1 i prawym, reprezentowanym przez napis t_2, jest reprezentowane przez napis $t_1(t_2)$, chyba że t_2 jest zmienną. W tym drugim przypadku, interesujące nas drzewo jest reprezentowane przez napis $t_1 t_2$ (pojedyńczej zmiennej nie bierzemy w nawiasy).

Napisy reprezentujące drzewa będące termami też możemy uważać za termy. Od razu zauważmy, że napisy reprezentujące termy nie zawierają symbolu aplikacji. Symbol ten pomijamy podobnie, jak symbol mnożenia w wyrażeniach liczbowych.

Termy rozumiane jako napisy są generowane przez następującą gramatykę:

1) $\langle term \rangle \ ::= \ \langle zmienna \rangle \ | \ \langle aplikacja \rangle$,

2) $\langle aplikacja \rangle \ ::= \ \langle term \rangle \langle zmienna \rangle \ | \ \langle term \rangle (\langle aplikacja \rangle)$,

3) $\langle zmienna \rangle \ ::= \ \ldots$

1.6 O termach raz jeszcze

Termy rozumiane jako napisy definiuje się także nieco mniej precyzyjnie, naśladując sposób traktowania wyrażeń (wzorów matematycznych) znany z lekcji i z wykładów z matematyki.

Możemy przyjąć, że termy są elementami zbioru termów T, do którego należą napisy (słowa) utworzone ze zmiennych ze zbioru V i nawiasów okrągłych, i który jest najmniejszym ze względu na zawieranie zbiorem spełniającym następujące warunki:

1) zmienne z V (rozumiane jako jednoliterowe słowa) należą do T (czyli $V \subseteq T$),

2) jeżeli $M_1, M_2 \in T$, to $(M_1 M_2) \in T$.

W powyższej definicji (M_1M_2) oznacza konkatenację czterech napisów: jednoliterowego słowa (, napisów M_1 i M_2 oraz jednoliterowego słowa).

Zgodnie z tą definicją, tworząc termy za pomocą nawiasów jednoznacznie ustalamy kolejność wykonywania wszystkich aplikacji, używając wszystkich potrzebnych. Zwykle uzupełniamy ją o zasady opuszczania niektórych nawiasów, które nie tyle obowiązują, ile mają ułatwić nam zapisywanie termów. Zasady te pozwalają term postaci $(M_1M_2)M_3$ skrócić do $M_1M_2M_3$ (o ile została uwidoczniona para odpowiadających sobie nawiasów i M_1, M_2, M_3 są termami), a także pominąć najbardziej zewnętrzne nawiasy termu.

Dokładniej, każdy fragment termu postaci $((M_1M_2)M_3$, w którym nawiasy obejmujące M_1M_2 odpowiadają sobie, możemy zastąpić przez $(M_1M_2M_3$, a na koniec, jeżeli cały interesujący nas term jest w nawiasach, to mamy prawo pominąć nawiasy najbardziej zewnętrzne. Tak więc

$$((((xy)z)((xz)t))(ty)) = (((xyz)((xz)t))(ty)) = ((xyz((xz)t))(ty)) =$$
$$= (xyz((xz)t)(ty)) = (xyz(xzt)(ty)) = xyz(xzt)(ty).$$

2 Algebry kombinatoryjne

2.1 Algebry aplikacyjne

Algebrą aplikacyjną nazywamy zbiór z działaniem binarnym. Matematycy takie algebry czasem nazywają grupoidami (a więc algebrami, które przypominają grupy). Będziemy też dopuszczać w takich algebrach stałe, a czasem będziemy wymagać, aby miały przynajmniej dwa elementy.

Działanie w algebrze aplikacyjnej będziemy nazywać aplikacją. Zauważmy też, że w algebrze aplikacyjnej, mając dwa dowolne elementy a i b, możemy wyliczyć aplikację $a \cdot b$, ale także aplikacje $b \cdot a$ oraz $a \cdot a$.

2.2 Algebry kombinatoryjne

Algebrą kombinatoryjną będziemy nazywać algebrę aplikacyjną z dwoma wyróżnionymi elementami (stałymi) S i K, w której spełnione są równości

$$Sxyz = xz(yz) \quad \text{oraz} \quad Kxy = x.$$

2.3 Wartość termu w algebrze aplikacyjnej

Przypuśćmy, że mamy daną algebrę aplikacyjną, a więc mamy pewien zbiór A, binarne działanie w tym zbiorze $\cdot$ i dodatkowo w tym zbiorze dwa elementy S i K. Niech M będzie termem zbudowanym ze zmiennych ze zbioru V i oczywiście nawiasów, a także dwóch stałych S i K.

Będziemy zakładać, że stała S oznacza element $S \in A$, i to samo dotyczy stałej K. Z podobną sytuacją mamy do czynienia w elementarnej matematyce: 0 to pewien znak (cyfra), także przedstawienie pewnej liczby, ale również myślimy, że jest to liczba zero. Na ogół nie odróżniamy liczb od ich przedstawień i stałych od tego, co oznaczają.

Jeżeli umówimy się, że zmienne z V oznaczają pewne elementy z algebry A, to będziemy mogli określić, co oznaczają poszczególne termy. Przyjmijmy więc, że mamy wartościowanie val zmiennych w algebrze A, czyli pewną funkcję $val : V \to A$.

Dla zmiennej $x \in V$ element $val(x)$ będziemy nazywać wartością zmiennej x przy wartościowaniu val. Funkcję val (określoną na zbiorze V) możemy rekurencyjnie rozszerzyć do funkcji określonej na zbiorze termów przyjmując, że spełnia następujące równości:

1) $val(S) = S$ oraz $val(K) = K$,

2) $val(Mx) = val(M) \cdot val(x)$ dla wszystkich termów M i dowolnej zmiennej x,

3) $val(M_1(M_2)) = val(M_1) \cdot val(M_2)$ dla dowolnego termu M_1 i termu M_2 nie będącego zmienną.

Zdefiniowany w ten sposób element $val(M)$ będziemy nazywać wartością termu M przy wartościowaniu val.

Pojęcie wartości termu pozwala wyjaśnić, kiedy w algebrze jest spełniona (zachodzi) równość dwóch termów $M_1 = M_2$. Przyjmujemy, że jest ona spełniona, jeżeli dla dowolnego wartościowania zmiennych val elementy $val(M_1)$ i $val(M_2)$ są identyczne.

Definiując pojęcie wartości termów w algebrach aplikacyjnych określiliśmy przy okazji kolejność działań, gdy nie wynika ona z rozmieszczenia nawiasów. Zauważmy, że licząc wartość termu $x \cdot y \cdot z$ najpierw obliczamy wartość $val(x \cdot y)$, a następnie wynik obliczeń aplikujemy do $val(z)$. Oznacza to, że jeżeli o kolejności działań nie rozstrzygają nawiasy, to działania są wykonywane od lewej do prawej, i ostatnim wykonywanym działaniem jest to znajdujące się najbardziej po prawej stronie.

2.4 Kombinatoryjna zupełność

Dzisiaj zwykle posługujemy się teoriomnogościowym pojęciem funkcji, dawniej raczej utożsamiało się funkcje z wzorami definiującymi. Wzory te wymagały użycia symboli oznaczających argumenty i nazywanych zmiennymi niezależnymi. Stąd być może wynika kariera w logice terminu *zmienna*. W logice symbole te występują w roli tzw. zmiennych wolnych. Wielu badaczy uważało, że korzystanie z tych zmiennych powoduje w matematyce wiele niejasności, a przynajmniej wymaga zbadania. Pierwszy krok w tym kierunku wykonał Mojżesz Schönfinkel, który zaproponował użycie w celu eliminowania zmiennych wolnych tzw. kombinatorów, takich jak S i K.

Zauważmy, że jeżeli S zostanie zaaplikowane do dwóch funkcji: dwuargumentowej f i jednoargumentowej g, oraz do argumentu z, to otrzymamy

$$S(f, g, z) = Sfgz = fz(gz) = f(z, g(z)).$$

Tak więc operacja S odpowiada specyficznemu składaniu argumentów. Podobnie, operacja K pozwala na definiowanie funkcji stałych, napis Kx może być rozumiany jako funkcja stale równa x.

Prawo wyłączonego środka zwykle zapisujemy w postaci $p \vee \neg p$. Może być rozumiane jako wzór definiujący pewną funkcję zmiennej p. Funkcję tę można otrzymać składając odpowiednio alternatywę i negację: $p \vee \neg p = \vee(p, \neg(p)) = S \vee \neg p$. Przypuśćmy, że w algebrze kombinatoryjnej mamy oprócz S i K także alternatywę $\vee$, negację $\neg$ i prawdę t. W tej algebrze funkcja z prawa wyłączonego środka może być reprezentowana przez element tej algebry dany wzorem $S \vee \neg$, zdefiniowany, jak widać, bez użycia zmiennych wolnych. Fakt, że prawo wyłączonego środka jest tautologią może dać się wyrazić wzorem $S \vee \neg = Kt$.

Mojżesz Schönfinkel zauważył, że mając kombinatory S i K z wszystkich wzorów można w podobny sposób eleliminować zmienne wolne. Mamy bowiem

Twierdzenie 2.1 *Algebry kombinatoryjne są kombinatoryjnie zupełne, a więc dla dowolnego termu M ze zmiennymi wymienionymi w ciągu $x_1, x_2, \ldots, x_n$ istnieje term stały T (bez zmiennych, zbudowany tylko ze stałych S i K) taki, że równość*

$$Tx_1x_2 \ldots x_n = M$$

jest spełniona we wszystkich algebrach kombinatoryjnych.

Dowód. Przypuśćmy, że x jest zmienną. Najpierw zdefiniujemy operację na termach φ_x taką, że

1) w termie $\varphi_x(M)$ występują jedynie zmienne występujące w M i różne od x,

2) spełniona jest równość $\varphi_x(M)x = M$ (czyli term $\varphi_x(M)$ reprezentuje funkcję zmiennej x definiowaną wzorem M).

Operację φ_x definiujemy przez rekursję następującymi wzorami

1) $\varphi_x(x) = SKK$,

2) $\varphi_x(y) = Ky$ dla zmiennej y różnej od x,

3) $\varphi_x(M_1M_2) = S(\varphi_x(M_1))(\varphi_x(M_2))$.

Mając już operacje φ_x dla wszystkich zmiennych, w szczególnym przypadku termu M ze zmiennymi x, y i z term T definiujemy jako

$$T = \varphi_x(\varphi_y(\varphi_z(M))).$$

W pozostałych przypadkach postępujemy analogicznie.

Sprawdzenie, że zdefiniowana operacja ma oczekiwane własności, jak również, że zachodzi teza jest łatwe i pozostawiam to dociekliwemu Czytelnikowi. □

3 Formalizacja λ-rachunku

3.1 Kilka uwag historycznych

Dawno temu, w czasach przed teorią mnogości, nie widziano potrzeby formalizowania pojęcia funkcji i raczej stosowano go w ograniczonym zakresie. Za to posługiwano się wzorami postaci $y = \sqrt{sin^2x + 3}$ lub $s = 1/2 \cdot gt^2$ (o pewnym znaczeniu fizycznym). Mówiono, że pierwszy z tych wzorów wiąże zmienną niezależną x i pewną zależną od niej zmienną y, drugi, opisujący swobodne spadanie, podaje drogę przebytą przez ciało w czasie pierwszych t sekund lotu i – rzecz jasna – jest stosowany dla wielu zmieniających się wartości t i nie tyle definiował zależność między t i s, co dla każdego t pozwalał wyliczyć odpowiednią wartość s. Dzisiaj takie wzory często uważamy za definicje funkcji f i s i zapisujemy w postaci $f(x) = \sqrt{3 + sin^2x}$ oraz $s(t) = 1/2 \cdot gt^2$. Alonzo Church, twórca λ-rachunku, zapragnął mieć operację, która dany wzór przekształca w funkcję definiowaną tym wzorem i zaczął ją stosować oraz badać. W szczególności przyjął, że funkcję f będzie oznaczać wyrażeniem postaci $\lambda x\ \sqrt{3 + sin^2x}$, podobnie $s = \lambda t\ 1/2 \cdot gt^2$. Operacja ta jest nazywana abstrakcją, symbol λ pojawił się chyba przypadkowo. Używanie jej powoduje konieczność posługiwania się termami z operatorem λ.

3.2 Termy λ-rachunku jako drzewa

Aby zdefiniować termy potrzebne są nam zmienne. Zbiór zmiennych, jak dotychczas, będziemy oznaczać symbolem V.

Termy λ-rachunku można uważać za drzewa binarne z węzłami etykietowanymi zmiennymi, symbolem $\cdot$ i napisami λx dla $x \in V$. W tym przypadku zbiór takich termów Λ jest najmniejszym zbiorem drzew binarnych zawierającym

1) jednoelementowe drzewa binarne z węzłem z etykietą, która jest zmienną,

2) drzewa z korzeniem z etykietą $\cdot$, z lewym poddrzewem M i prawym poddrzewem N, takimi, które są termami ($M, N \in \Lambda$),

3) drzewa z korzeniem z etykietą λx ($x \in V$), którego prawe poddrzewo jest puste, a lewe poddrzewo M jest termem, czyli należy do Λ.

Takie etykietowane drzewa mogą być reprezentowane przez napisy. Ogólna idea konstruowania napisu reprezentującego drzewo jest następująca: najpierw piszemy napis reprezentujący lewe poddrzewo, dopisujemy do niego etykietę korzenia i w końcu podajemy napis reprezentujący prawe poddrzewo. W takim napisie używamy też nawiasów pozwalających jednoznacznie odtworzyć reprezentowane drzewo. Dodatkowo pomijamy symbol aplikacji $\cdot$.

3.3 Termy jako napisy

Termy możemy też definiować jako napisy generowane przez następującą gramatykę:

1) $\langle \lambda - term \rangle \ ::= \ \langle zmienna \rangle \ | \ \langle aplikacja \rangle \ | \ \langle abstrakcja \rangle$,

2) $\langle term\ prosty \rangle \ ::= \ \langle zmienna \rangle \ | \ (\langle aplikacja \rangle) \ | \ (\langle abstrakcja \rangle)$,

3) $\langle aplikacja \rangle \ ::= \ \langle zmienna \rangle \langle term\ prosty \rangle \ | \ (\langle abstrakcja \rangle) \langle term\ prosty \rangle \ | \ | \ \langle aplikacja \rangle \langle term\ prosty \rangle$,

4) $\langle abstrakcja \rangle \ ::= \ \lambda \langle zmienna \rangle \langle \lambda - term \rangle$,

5) $\langle zmienna \rangle \ ::= \ \ldots$.

Jak widać mamy trzy rodzaje termów: zmienne, aplikacje i abstrakcje. Napisy tworzone zgodnie z powyższą gramatyką zawierają bardzo mało nawiasów. Abstrakcje będące członami aplikacji są ujęte w nawiasy. Wieloczłonowe aplikacje są interpretowane tak, jak w przypadku algebr aplikacyjnych.

3.4 Trzeci sposób definiowania λ-termów

Zbiór Λ możemy też definiować jako najmniejszy zbiór napisów (słów) taki, że

1) $V \subseteq \Lambda$,

2) jeżeli $M_1, M_2 \in \Lambda$, to $(M_1 M_2) \in \Lambda$,

3) jeżeli $x \in V$ oraz $M \in \Lambda$, to $(\lambda x\ M) \in \Lambda$.

W tak rozumianych termach używamy wszystkich możliwych nawiasów. Wobec tego taką definicję stosujemy razem pewnymi zasadami upraszczającymi. W szczególności, termy postaci $(M_1M_2)M_3$ zapisujemy w postaci $M_1M_2M_3$ (pomijamy nawiasy, podobnie jak w przypadku termów w algebrach aplikacyjnych). Wielokrotne abstrakcje takie, jak $\lambda x\lambda y\lambda z\ M$ zapisujemy z kropką w postaci $\lambda xyz.M$. Abstrakcje na ogół ujmujemy w nawiasy z wyjątkiem całych termów, które zapisujemy bez nawiasów zewnętrznych.

3.5 Wystąpienia zmiennych, wystąpienia wolne i związane

Aby określić wystąpienie zmiennej w termie musimy wskazać zmienną, która nas interesuje, podać term, w którym ta zmienna ma występować, i na przykład określić liczbę znaków poprzedzających wystąpienie w rozważanym termie.

Wystąpienia zmiennych mogą być wolne lub związane. Wystąpienie jest wolne wtedy i tylko wtedy, gdy nie jest związane. Pojęcia te są definiowane przez indukcję ze względu na budowę termu.

Każde (jedyne) wystąpienie zmiennej x w termie x jest wolne, żadna zmienna nie jest w tym termie związana. Jeżeli interesuje nas wystąpienie zmiennej x w termie M_1, to jest to także wystąpienie w termie M_1M_2 i jest ono wolne w tym termie wtedy i tylko wtedy, gdy jest ono wolne w M_1. To samo dotyczy zmiennej x w termie M_2. Żadne wystąpienie x nie jest wolne w termie λxM, a więc wszystkie są związane. Wystąpienie zmiennej x w termie λyM jest wolne wtedy i tylko wtedy, gdy jest wolnym wystąpieniem w termie M.

W termie $x(\lambda xxx)$ mamy raczej trzy, a nie cztery wystąpienia zmiennej x. Pierwsze wystąpienie jest wolne, ostatnie i przedostatnie są związane. Dobrze jest nie rozważać jako wystąpienia tej zmiennej x, która znajduje się bezpośrednio za operatorem λ. Lepiej uważać ją indeks operatora λ, dodatkową informację dookreślającą operator i jakby część tego operatora. Zmienną umieszczoną bezpośrednio za operatorem λ czasem określa się jako wiązaną tym operatorem, a o operatorze mówi się, że wiąże zmienną podaną bezpośrednio za nim. W rzeczywistości, wiąże on w podanym termie dwie ostatnie zmienne x lub dwa ostatnie wystąpienia tej zmiennej.

Zmienna x (a nie wystąpienie tej zmiennej) jest wolna w termie M, jeżeli pewne jej wystąpienie jest wolne w termie M. Analogicznie definiujemy zmienne związane. Zauważmy, że zmienna x może nie być związana w termie M mimo, że w tym termie występuje operator λ wiążący zmienną x.

Symbolem $FV(M)$ będziemy oznaczać zbiór zmiennych wolnych (występujących jako wolne) w termie M. Zbiory te są definiowane indukcyjnie równościami

1) $FV(x) = \{x\}$,

2) $FV(M_1M_2) = FV(M_1) \cup FV(M_2)$,

3) $FV(\lambda xM) = FV(M) \setminus \{x\}$.

3.6 Podtermy

Pojęcie podtermu najłatwiej zdefiniować, gdy termy uważamy za drzewa. W tym przypadku podtermem jest poddrzewo o określonym korzeniu. Tak rozumiane podtermy są właściwie ich wystąpieniami: różne węzły drzewa mogą być korzeniami tak samo zbudowanych poddrzew, ale będą wyznaczać różne podtermy.

Podtermy możemy też definiować rekurencyjnie. Każdy term ma jeden niewłaściwy podterm: jest nim on sam. Pozostałe podtermy nazywamy właściwymi. Zmienne nie mają właściwych podtermów. Właściwymi podtermami aplikacji M_1M_2 są wszelkie podtermy (właściwe i niewłaściwe) zarówno termu M_1, jak i termu M_2. Właściwymi podtermami abstrakcji λxM są wszelkie podtermy termu M.

Każdy podterm można otrzymać usuwając z termu pewne liczby znaków: z początku i z końca termu. Jest on wyznaczony przez liczbę znaków usuniętych z początku termu. Zauważmy też, że yz nie jest podtermem termu xyz. Ten term rozumiany jako drzewo ma pięć węzłów i, w konsekwencji, tylko pięć podtermów.

3.7 Konteksty

Kontekstem jest to, co zostaje z termu po usunięciu z niego pewnego podtermu.

Zgodnie z bardziej precyzyjną definicją, konteksty to λ-termy, w których występuje specjalna zmienna oznaczana symbolem [ ]. Zmienna ta w kontekście występuje dokładnie jeden raz i nie jest wiązana operatorem λ.

Pojęcie kontekstu definiujemy rekurencyjnie przyjmując, że

1) [ ] jest kontekstem,

2) jeżeli C jest kontekstem, a $M \in \Lambda$, to (CM) oraz (MC) są kontekstami,

3) jeżeli C jest kontekstem, a x – zmienną z V (różną od [ ]), to (λxC) jest kontekstem.

3.8 Podstawianie w kontekstach

W kontekstach będziemy podstawiać za zmienną [ ]. Wynik podstawiania w kontekście C termu N będziemy oznaczać symbolem $C[N]$. Operacja ta spełnia następujące równości:

1) $[\,][N] = N$,

2) $(CM)[N] = C[N]M$ oraz $(MC)[N] = MC[N]$ dla wszystkich kontekstów C i $M \in \Lambda$,

3) $(\lambda x\ C)[N] = \lambda x\ C[N]$ dla wszystkich kontekstów C i $x \in V$.

Podstawianie w kontekstach nie podlega żadnym ograniczeniom. Każdy term możemy podstawić w każdym kontekście.

Związek między kontekstami i podtermami wyraża następujący

Lemat 3.1 *Term N jest podtermem termu M wtedy i tylko wtedy, gdy $M = C[N]$ dla pewnego kontekstu C.* □

3.9 Trochę o podstawianiu

Doskonale wiadomo, co trzeba zrobić, aby wyliczyć wartość funkcji definiowanej pewnym wzorem: w pierwszym rzędzie trzeba argument podstawić we wzorze definiującym funkcję. Jeżeli na przykład chcemy wyliczyć wartość funkcji $f(x) =$

$\sqrt{3 + sin^2 x}$ dla liczby 5, to obliczamy wartość wyrażenia $\sqrt{3 + sin^2\ 5}$. Analogicznie postępujemy w rachunku lambda wykorzystując ściśle zdefiniowaną operację podstawienia.

Wzory w λ-rachunku są jednak bardziej skomplikowane od przytoczonego. Bardziej przypominają wyrażenia takie, jak $x^3 \cdot \int \sqrt{3 + sin^2 x}\ dx$, w którym prosta zamiana x na 5 nie prowadzi do niczego sensownego.

3.10 Operacja podstawiania

Dla dwóch λ-termów M i N i zmiennej x rekurencyjnie definiujemy podstawienie $M[x := N]$ termu N za zmienną x w termie M. Przyjmujemy, że

1) $x[x := N] = N$,

2) jeżeli $y \neq x$, to $y[x := N] = y$,

3) $(M_1 M_2)[x := N] = M_1[x := N]M_2[x := n]$,

4) $(\lambda x\ M_1)[x := N] = \lambda x\ M_1$,

5) jeżeli $y \neq x$, to $(\lambda y\ M_1)[x := N] = \lambda y\ M_1[x := N]$.

3.11 Kłopoty z podstawieniami

Weźmy term $F = \lambda xy.yx$. Intuicje związane z rachunkiem λ podpowiadają, że F oznacza operację dwóch zmiennych x i y, która drugi z otrzymanych argumentów aplikuje do pierwszego. W związku z tym spodziewamy się, że ten term dla dowolnych M i N spełnia równość $FMN = NM$, a w szczególności spełnia $Fyx = xy$.

Chcemy także, aby zachodziły równości $FM = (\lambda xy.yx)M = \lambda y.(yx)[x := M] = \lambda y.yM$, w szczególności, aby $Fy = \lambda y.yy$ i analogicznie, aby $Fyx = (\lambda y.yy)x = xx$. Wydaje się więc, że powinna zachodzić równość $xy = xx$.

Jeżeli na równych termach wykonamy te same operacje powinniśmy otrzymać rzeczy identyczne. Tak więc $\lambda xy.xy = \lambda xy.xx$ i dalej

$$K = IK = (\lambda xy.xy)IK = (\lambda xy.xx)IK = II = I.$$

Stąd możemy ostatecznie wywnioskować, że każde dwa termy są równe:

$$M = IKMN = KIMN = IN = N.$$

Błąd tkwi w dowodzie równość $Fy = \lambda y.yy$. Po lewej stronie równości zmienna y występuje w dwóch rolach. W termie F jest zmienną wiązaną operatorem λ, wskazuje, gdzie powinniśmy podstawić argument, gdyby przyszło nam liczyć jakąś wartość podtermu $\lambda y.xy$. Z drugiej strony, y oznacza jakąś rzecz, do której aplikujemy F. W wyniku podstawienia to y pojawia się w zasięgu operatora λ wiążącego zmienną y, przestaje oznaczać tę rzecz, staje się oznaczeniem miejsca, które być może zastąpimy jakimś argumentem podczas dalszych obliczeń i bezpodstawnie zmienia swoje znaczenie.

3.12 Podstawialność

Term N jest podstawialny za zmienną x w termie M, jeżeli w termie M żadne wolne wystąpienie zmiennej x nie znajduje się w zasięgu operatora abstrakcji λ wiążącego zmienną wolną termu N.

Lemat 3.2 *Jeżeli zmienne wiązane z termu M nie są wolne (w szczególności: nie występują) w termie N, to N jest podstawialny w termie M za dowolną zmienną.* □

3.13 Własności podstawiania

Lemat 3.3 *Zawsze zachodzi wzór*

1) $M[x := x] = M$.

2) jeżeli zmienna $x \notin FV(M)$, *to* $M[x := N] = M$,

3) jeżeli $x \neq y$ *oraz* $x, y \notin FV(L)$, *to*

$$M[x := N][y := L] = M[y := L][x := N][y := L]. \quad \square$$

Ważną własność podstawiania wyraża

Lemat 3.4 *Jeżeli* $x \neq y$ *i* $x \notin FV(L)$, *to*

$$M[x := N][y := L] = M[y := L][x := N[y := L]]. \quad \square$$

4 Rodzaje relacji

4.1 Relacje zgodne

Relacja R w zbiorze λ-termów jest zgodna (z operacjami λ-rachunku) jeżeli dla wszystkich $M, N, Z \in \Lambda$

1) warunek $M\ R\ N$ pociąga za sobą $(ZM)\ R\ (ZN)$ oraz $(MZ)\ R\ (NZ)$,

2) warunek $M\ R\ N$ implikuje, że $(\lambda x.M)\ R\ (\lambda x.N)$.

4.2 Kongruencje

Kongruencjami nazywamy zgodne relacje równoważności. Najprostszym przykładem kongruencji jest relacja równości.

4.3 Redukcje

Mamy dwa rodzaje redukcji: w jednym i w wielu krokach. Redukcja w jednym kroku najczęściej jest definiowana jako najmniejsza relacja zgodna rozszerzająca pewne proste przekształcenie. Redukcja w jednym kroku wyznacza redukcję w wielu krokach, która jest krótko nazywana redukcją.

Redukcją zwykle nazywamy zgodną relację zwrotną i przechodną.

Mając redukcję w jednym kroku $\rightarrow$ definiujemy relację $\twoheadrightarrow$ przyjmując, że jest to najmniejsza relacja spełniająca dla wszystkich $L, M, N \in \Lambda$ warunki

1) jeżeli $M = N$, to $M \twoheadrightarrow N$,

2) jeżeli $M \to N$, to $M \twoheadrightarrow N$,

3) jeżeli $M \twoheadrightarrow L$ i $L \twoheadrightarrow N$, to $M \twoheadrightarrow N$.

Krótko mówiąc, relację $\twoheadrightarrow$ definiujemy jako zwrotne i przechodnie domknięcie relacji $\to$.

Lemat 4.1 *Jeżeli relacja redukcji w jednym kroku $\to$ jest zgodna, to relacja $\twoheadrightarrow$ jest redukcją.* □

4.4 Konwersje

Mając relację redukcji $\twoheadrightarrow$ definiujemy związaną z nią relację konwersji $\cong$ przyjmując, że jest to najmniejsza relacja spełniająca dla wszystkich $L, M, N \in \Lambda$ warunki

1) jeżeli $M \twoheadrightarrow N$, to $M \cong N$,

2) jeżeli $M \cong N$, to $N \cong M$,

3) jeżeli $M \cong L$ i $L \cong N$, to $M \cong N$.

Lemat 4.2 *Jeżeli relacja $\twoheadrightarrow$ jest redukcją, to relacja $\cong$ jest kongruencją.* □

5 α-konwersja

Njpierw zauważmy, że wzory $\sqrt{3 + sin^2 x}$ oraz $\sqrt{3 + sin^2 y}$ definiują tę samą funkcję, w obu przypadkach liczenie wartości funkcji dla danego argumentu prowadzi do tych samych obliczeń. Tak więc wydaje się, że w λ-rachunku wyrażenia $\lambda x\sqrt{3 + sin^2 x}$ oraz $\lambda y\sqrt{3 + sin^2 y}$ powinny być uważane za identyczne. Aby sformalizować to spostrzeżenie, wprowadzimy relacji α-redukcji i α-konwersji.

5.1 α-redukcje, w jednym i w wielu krokach

Relację α-redukcji w jednym kroku, czyli relację $\to_\alpha$, definiujemy jako najmniejszą relację zgodną z operacjami rachunku lambda zawierającą wszystkie pary

$$\lambda x.M \to_\alpha \lambda y.M[x := y],$$

gdzie y jest zmienną nie będącą wolną w termie M ($y \notin FV(M)$) i podstawialną w M za zmienną x.

Relacja α-redukcji w jednym kroku wyznacza tak, jak to zostało wyżej opisane, relację α-redukcji $\twoheadrightarrow_\alpha$ i relację α-konwersji. Relację α-konwersji będziemy najczęściej oznaczać symbolem $=_\alpha$, a czasem może być ona oznacza także symbolem $=$.

Oczywiście, α-konwersja jest kongruencją.

Lemat 5.1 *1) Jeżeli $y \notin FV(M)$, to $x \notin FV(M[x := y])$.*

2) Jeżeli $y \notin FV(M)$, to y nie występuje w termie $M[x := y]$ jako zmienna wolna w zasięgu kwantyfikatora wiążącego x.

3) Jeżeli $y \notin FV(M)$, to zmienna x jest podstawialna za y w termie $M[x := y]$.

4) Jeżeli $y \notin FV(M)$, to $\lambda y.M[x := y] \rightarrow_\alpha M[x := y][y := x] = M$.

5) Relacja $\rightarrow_\alpha$ jest symetryczna.

6) Relacja $\twoheadrightarrow_\alpha$ jest symetryczna. □

Wniosek 5.2 *Relacja α-redukcji $\twoheadrightarrow_\alpha$ jest relacją α-konwersji $=_\alpha$.* □

5.2 α-konwersja, a podstawianie

Operacja podstawiania jest zawsze wykonalna, ale czasem prowadzi do absurdalnych wyników. W związku z tym ograniczamy jej stosowanie do sytuacji, w których podstawiamy term podstawialny. Narzucamy więc sobie ograniczenia wykonalność podstawiania. Z drugiej strony niektóre termy uważamy za różniące się nieistotnie. To daje szansę na zmianę termu, w którym nie można podstawiać tak, aby podstawianie stało się możliwe.

Lemat 5.3 *Dla każdych λ-termów M i N i dowolnej zmiennej x istnieje term M' taki, że $M =_\alpha M'$ i term N jest podstawialny za x w termie M'.*

Dowód. Weźmy term M i policzmy występujące w nim operatory λ. Jeżeli jest ich n, to wybieramy ciąg $y_1, y_2, \ldots, y_n$ parami różnych zmiennych nie występujących ani w M, ani w N, różnych od x. Teraz przekształcamu dany term M w następujący sposób:

1) $i = 1$;

2) dla kolejnych operatorów λ w termie M:

3) przedstaw M w postaci $C[\lambda y\ A]$ dla pewnego kontekstu C, termu A i zmiennej y;

4) M zastąp przez $C[\lambda y_i\ A[y := y_i]]$; $i = i + 1$

5) zwróć term M.

Niech M' będzie termem skonstruowanym przez podany algorytm. Nietrudno zauważyć, że M' powstał z M w wyniku wielokrotnej α-redukcji. Tak więc $M =_\alpha M'$. Term N jest podstawialny w M', ponieważ żadna występująca w nim zmienna nie jest związana w termie M' (w M' wiązane są wyłącznie zmienne $y_1, y_2, \ldots, y_n$). □

Lemat 5.4 *Jeżeli M_1 i M_2 są różnymi α-konwertowalnymi termami, to pierwsze znaki różniące te termy są zmiennymi wiązanymi operatorami λ.* □

Lemat 5.5 *Jeżeli M_1 i M_2 są α-konwertowalnymi termami i term N jest w nich podstawialny za zmienną x, to termy $M_1[x := N]$ i $M_2[x := N]$ też są α-konwertowalne (a więc $M_1[x := N] =_\alpha M_2[x := N]$).*

Dowód. □

5.3 Operacje na klasach abstrakcji α-konwersji

Umówmy się, że w tym rozdziale klasę abstrakcji α-konwersji wyznaczoną przez term M oznaczamy symbolem $[M]_\alpha$.

Na termach wykonujemy trzy operacje: mając dwa termy tworzymy term oznaczający aplikację, mając term tworzymy term oznaczający abstrakcję i wykonujemy bardziej skomplikowaną operację podstawiania. Te trzy operacje można rozszerzyć do operacji na klasach abstrakcji α-konwersji i robimy to w sposób standardowy. Przyjmujemy, że

1) $[M]_\alpha[N]_\alpha = [MN]_\alpha$,

2) $\lambda x\ [M]_\alpha = [\lambda x\ M]_\alpha$,

3) $[M]_\alpha[x := [N]_\alpha] = [M'[x := N]]_\alpha$, gdzie M' jest pewnym termem równym M w sensie relacji α-konwersji ($M' =_\alpha M$), w którym term N jest podstawialny za zmienną x.

Jak zwykle w takich przypadkach pojawia się problem poprawności tych definicji. Dwie pierwsze są poprawne, ponieważ relacja α-konwersji jest zgodna z działaniami wykonywanymi w λ-rachunku. Poprawność ostatniej wynika z z lematów 5.3 i 5.5.

5.4 Raz jeszcze o termach

Jest jeszcze jeden sposób definiowania λ-termów. Jak dotychczas możemy λ-termy uważać za napisy lub, jak ktoś woli, za drzewa spełniające wiadome warunki. Zwykle jednak uważa się termy za klasy abstrakcji relacji α-konwersji.

Możemy w λ-rachunku pisać λ-termy (odpowiednie napisy) i przekształcać je zgodnie z obowiązującymi zasadami. Wtedy zastąpienie termu M termem M' będącym z nim w relacji α-konwersji powinniśmy uznać za jeden z dopuszczalnych sposobów przekształcania.

Możemy też prawdziwe λ-termy definiować jako klasy abstrakcji relacji α-konwersji, czyli elementy zbioru $\Lambda/_{=_\alpha} = \{[M]_\alpha \ : \ M \in \Lambda\}$. Wtedy te klasy przekształcamy stosując odpowiednie rozszerzenie relacji β-redukcji

$$(\lambda x[M]_\alpha)[N]_\alpha \to_\beta [M]_\alpha[x := [N]_\alpha].$$

Taka definicja i rygorystycznie stosowane oznaczenia prowadzą do bardzo skomplikowanych wzorów. Wobec tego, tak rozumiejąc termy stosujemy uproszczoną notację pozwalającą pisać o klasie abstrakcji $[M]_\alpha$ podając tylko jej reprezentanta M (zamiast $[M]_\alpha$ piszemy po prostu M). Także oznaczenie $\Lambda/_{=_\alpha}$ będziemy skracać do Λ.

6 Pierwsze podejście do λ-rachunku

λ-rachunek może być rozumiany jako system dowodzenia (wyprowadzania) równości. Definiując go w stylu znanym z logiki możemy przyjąć, że jest to najmniejszy zbiór równości termów zamknięty ze względu na wnioskowanie za pomocą następujących reguł dowodzenia:

1) odpowiadających podstawowym aksjomatom równości:

$$\frac{}{M = M}, \quad \frac{M = N}{N = M}, \quad \frac{K = M,\ M = N}{K = N},$$

2) odpowiadających aksjomatom równości związanym z działaniami:

$$\frac{M_1 = M_2}{M_1 N = M_2 N}, \quad \frac{M_1 = M_2}{N M_1 = N M_2}, \quad \frac{M_1 = M_2}{\lambda x M_1 = \lambda x M_2},$$

3) i najważniejszej reguły (β-reguły)

$$\frac{}{(\lambda x\ M)N = M[x := N]}.$$

Fakt, że równość $M_1 = M_2$ należy do wyżej zdefiniowanego zbioru (czyli daje się uzasadnić za pomocą β-reguły) zapisujemy bardziej formalnie wzorem $\lambda \vdash M_1 = M_2$, ale najczęściej zapisujemy po prostu jako $M_1 = M_2$.

Przytoczona definicja wymaga, aby termy λ-rachunku były rozumiane jako klasy abstrakcji α-konwersji. Jeżeli termy rozumiemy jako napisy, to musimy ograniczyć stosowanie β-reguły do sytuacji, gdy term N jest podstawialny w M za zmienną x i dodać jeszcze regułę α-konwersji

$$\frac{}{\lambda x\ M = \lambda y\ M[x := y]},$$

która jest stosowana pod zwykłymi warunkami (patrz 5.1).

Na razie przestawione tu podejście do λ-rachunku jest nas wystarczające. Później, gdy będziemy szczegółowo analizować wyprowadzenia równości, pojawi się relacja β-redukcji $\rightarrow_\beta$, która w najprostszej wersji (dla klas abstrakcji) jest definiowana wzorem $(\lambda x\, M)N \rightarrow_\beta M[x := N]$.

7 Liczby naturalne w λ-rachunku

7.1 Numerały Churcha

Najpierw wprowadźmy operację, która z pary termów $F, M \in \Lambda$ i liczby naturalnej n robi term $F^n(M)$. Wynik tej operacji jest definiowany rekurencyjnie wzorami

$$F^0(M) = M \ \text{ oraz } \ F^{n+1}(M) = F(F^n(M)).$$

Oczywiście, numerały Churcha zostały wymyślone przez Alonzo Churcha. Tak nazywamy termy

$$c_n = \lambda f x.\ f^n(x).$$

Możemy uważać, że termy te reprezentują w rachunku λ liczby naturalne.

Zauważmy, że

1) $c_n\ f\ x\ =\ f^n(x)$, można przyjąć, że $c_n\ f\ x$ oznacza wartość n-krotnego złożenia funkcji f dla argumentu x,

2) $c_n\ f\ =\ \lambda x\ f^n(x)$, tak więc $c_n\ f$ odpowiada n-krotnemu złożeniu funkcji f,

3) w końcu samo c_n odpowiada operacji n-krotnego składania.

7.2 Pierwsze wzory

Najpierw zauważmy, że $\lambda x\,(\lambda x\,M)x \;=\; \lambda x\,M$.

Jest oczywiste, że

$$c_{n+1}\,f\,x \;=\; f^{n+1}(x) \;=\; f(f^n(x)) \;=\; f(c_n\,f\,x).$$

Wobec tego,

$$c_{n+1} \;=\; \lambda f x.\, c_{n+1}\,f\,x \;=\; \lambda f x.\, f(c_n\,f\,x) \;=\; (\lambda a f x.\, f(a\,f\,x))\,c_n.$$

Podobnie,

$$c_{m+n}\,f\,x \;=\; f^{m+n}(x) \;=\; f^n(f^m(x)) \;=\; f^n(c_m\,f\,x) \;=\; c_n\,f\,(c_m\,f\,x),$$

i w konsekwencji

$$c_{m+n} = \lambda f x.\, c_{m+n}\,f\,x \;=\; \lambda f x.\, c_n\,f\,(c_m\,f\,x) \;=\; (\lambda a b f x.\, b\,f\,(a\,f\,x))\,c_m\,c_n.$$

Odwołując się może nieco do intuicji mamy też

$$f^{mn}(x) \;=\; (f^m)^n(x) \;=\; c_n\,f^m\,x \;=\; c_n\,(c_m\,f)\,x.$$

Równość skrajnych wyrażeń w tym wzorze można łatwo dowieść przez indukcję ze względu na n. Dla $n = 0$ jest ona oczywista, a ponadto

$$\begin{aligned} f^{m(n+1)}(x) \;&=\; f^m(f^{mn}(x)) \;=\; c_m\,f\,(c_n\,(c_m\,f)\,x) \;= \\ &=\; c_m\,f\,((c_m\,f)^n(x)) \;=\; (c_m\,f)^{n+1}(x) \;=\; c_{n+1}\,(c_m\,f)\,x. \end{aligned}$$

Stąd otrzymujemy wzór $c_{mn} \;=\; (\lambda a b f x.\, b\,(a\,f)\,x)\,c_m\,c_n$, a także nieco krótszy wzór

$$\begin{aligned} (\lambda a b f.\, b\,(a\,f))\,c_m\,c_n \;&=\; \lambda f\,c_n\,(c_m\,f) \;=\; \lambda f\,\lambda x\,(c_m\,f)^n(x) \;= \\ &=\; \lambda f x.\, c_n\,(c_m\,f)\,x \;=\; \lambda f x.\, f^{mn}(x) \;=\; c_{mn}. \end{aligned}$$

7.3 Funkcje λ-reprezentowalne

Teraz będziemy rozważać funkcje jednej lub wielu zmiennych naturalnych, całkowite i przyjmujące wartości będące liczbami naturalnymi. Niech $f : N^k \to N$ będzie taką funkcją.

Term $F \in \Lambda$ reprezentuje w λ-rachunku funkcję f, jeżeli w λ-rachunku można wyprowadzić wszystkie równości postaci

$$F\,c_{n_1}\,c_{n_2}\,\ldots\,c_{n_k} \;=\; c_{f(n_1,n_2,\ldots,n_k)}.$$

Funkcja f jest reprezentowalna w λ-rachunku, jeżeli istnieje λ-term, który ją reprezentuje.

Pojęcie funkcji reprezentowalnej może zależeć od sposobu reprezentowania w λ-rachunku liczb naturalnych. Powyższa definicja, oczywiście, dotyczy sytuacji, gdy liczby naturalne reprezentujemy za pomocą numerałów Churcha. Będziemy przyjmować analogiczne definicje dla innych sposobów reprezentowania liczb naturalnych.

Z rachunków z rozdziału 7.2 wynika, że operacja następnika, dodawanie i mnożenie są funkcjami reprezentowalnymi. W szczególności, term $\lambda a f x.\, f(a\,f\,x)$ reprezentuje następnik, dodawanie jest reprezentowane przez term $\lambda a b f x.\, b\,f\,(a\,f\,x)$, a mnożenie – przez $\lambda a b f.\, b\,(a\,f)$.

7.4 Prawda i fałsz w λ-rachunku

Aby reprezentować w rachunku λ prawdę i fałsz wystarczy mieć dwa termy, o których nie można dowieść, że są różne. Na razie trudno nam będzie spełnić ten warunek.

Zwykle jednak prawdę i fałsz reprezentuje się odpowiednio za pomocą termów

$$\mathbf{true} \;=\; \lambda xy.\, x \;\text{ oraz }\; \mathbf{false} \;=\; \lambda xy.\, y.$$

Wtedy mamy

$$\mathbf{true}\, M\, N \;=\; M \;\text{ oraz }\; \mathbf{false}\, M\, N \;=\; N.$$

Dla takiej definicji prawdy i fałszu łatwo reprezentować spójnik **if then else** . Możemy przyjąć, że

$$\mathbf{if\ then\ else} \;=\; \lambda xyz.\, zxy \;\text{ lub ewentualnie }\; \mathbf{if\ then\ else} \;=\; \lambda xyz.\, xyz$$

zależnie od tego, w jakiej kolejności spójnik otrzymuje swoje argumenty. W szczególności

$$\mathbf{if\ true\ then}\ M\ \mathbf{else}\ N \;=\; M \;\text{ oraz }\; \mathbf{if\ false\ then}\ M\ \mathbf{else}\ N \;=\; N.$$

W logice klasycznej, mając spójnik **if then else** oraz prawdę i fałsz, można zdefiniować pozostałe spójniki logiczne. Analogicznie spójniki te reprezentujemy w λ-rachunku. Tak więc term $\lambda x.\, x\, \mathbf{false}\, \mathbf{true}$ reprezentuje negację, a termy $\lambda xy.\, x\, y\, \mathbf{false}$ i $\lambda xy.\, x\, \mathbf{true}\, y$ reprezentują odpowiednio koniunkcję i alternatywę.

7.5 Pary uporządkowane

Termy **if then else** , **true** i **false** są w rachunku λ wykorzystywane także do formalizacji pojęcia pary uporządkowanej. Term **if then else** , zwłaszcza w postaci $\lambda xyz.\, zxy$, jest kombinatorem tworzącym pary uporządkowane. Tak więc wyrażenie

$$[M, N] \;=\; (\lambda xyz.\, zxy)\, M\, N \;=\; \lambda z\, z\, M\, N$$

jest parą uporządkowaną o pierwszej współrzędnej M i drugiej – N, a raczej jest pewnym termem utworzonym z termów M i N, który w prosty sposób pozwala odtworzyć oba te termy. Mamy w szczególności

$$[M, N]\, \mathbf{true} \;=\; M \;\text{ oraz }\; [M, N]\, \mathbf{false} \;=\; N.$$

7.6 Systemy liczbowe

System liczbowy to sposób reprezentowania w λ-rachunku liczb naturalnych.

Definiując liczby naturalne zwykle musimy wskazać liczbę 0 i dla dowolnej liczby jej następnik, czyli liczbę następną, o 1 większą. O operacji tworzenia następnika zakładamy, że jest różnowartościowa i nie przejmuje wartości 0. Wymagania stawiane reprezentacjom liczb naturalnych w λ-rachunku są wzorowane na tego typu definicjach.

Systemem liczbowym (wg Barendregta: adekwatnym systemem liczbowym) nazywamy ciąg λ-termów $D_0, D_1, D_2, \ldots$ wraz w trzema termami Z, S^+ i P^- spełniającymi dla wszystkich liczb naturalnych n następujące warunki:

$$Z\, D_0 = \mathbf{true},\;\; Z\, D_{n+1} = \mathbf{false},\;\; S^+\, D_n = D_{n+1} \;\text{ oraz }\; P^-\, D_{n+1} = D_n.$$

Nie jest oczywiste, że numerały Churcha tworzą system liczbowy. Przekonamy się o tym później.

7.7 Numerały Barendregta

Barendregt zaproponował inny sposób reprezentowania liczb naturalnych, różny od numerałów Churcha. Numerały Barendregta $\lceil n \rceil$ są definiowane przez rekursję wzorami

$$\ulcorner 0 \urcorner \;=\; I \;=\; \lambda x\, x \;\text{ oraz }\; \ulcorner n+1 \urcorner \;=\; [\mathbf{false}, \ulcorner n \urcorner].$$

Numerały Barendregta razem z niżej zdefiniowanymi termami Z, S^+ i P^- tworzą system liczbowy. Wystarczy przyjąć, że

$$Z \;=\; \lambda x\, x\, \mathbf{true}, \quad S^+ \;=\; \lambda xz.\, z\, \mathbf{false}\, x \;=\; \lambda x\, [\mathbf{false}, x] \;\text{ oraz }\; P^- \;=\; \lambda x\, x\, \mathbf{false}.$$

8 Funkcje pierwotnie rekurencyjne

8.1 Definicja

Klasa funkcji pierwotnie rekurencyjnych jest najmniejszą klasą naturalnych funkcji całkowitych

1) zawierającą funkcje Z, S i U_k^n spełniające dla wszystkich możliwych naturalnych argumentów równości

$$Z(m) = 0, \quad S(m) = m+1 \quad \text{oraz} \quad U_k^n(m_1, m_2, \ldots, m_n) = U_k^n(\vec{m}) = m_k$$

2) i zamkniętą ze względu na złożenie

$$h(\vec{m}) = f(g_1(\vec{m}), \ldots, g_k(\vec{m}))$$

(jeżeli funkcje f, $g_1, \ldots, g_k$ są pierwotnie rekurencyjne, to także funkcja h jest pierwotnie rekurencyjna, tutaj $\vec{m} = (m_1, \ldots, m_n)$),

3) oraz definiowanie przez rekursję prostą

$$f(\vec{m}, 0) = g(\vec{m}) \quad \text{oraz} \quad f(\vec{m}, k+1) = h(\vec{m}, k, f(\vec{m}, k))$$

(jeżeli funkcje g i h są pierwotnie rekurencyjne, to także funkcja f jest pierwotnie rekurencyjna).

Definiowanie przez rekursję prostą obejmuje także przypadek $n = 0$. W tym przypadku definicja przyjmuje postać

$$f(0) = g \quad \text{oraz} \quad f(k+1) = h(k, f(k))$$

dla dowolnie ustalonej liczby g.

Niemal wszystkie używane na codzień funkcje naturalne są pierwotnie rekurencyjne. Wielu matematyków nie potrafi podać przykładu funkcji naturalnej, która nie jest pierwotnie rekurencyjna, gdyż nie posługują się takimi funkcjami.

8.2 Trochę historii

Pierwszą osobą, która istotnie posłużyła się funkcjami pierwotnie rekurencyjnymi, był Kurt Gödel. Użył ich w dowodzie twierdzenia o niezupełności arytmetyki (z 1932 roku) i w tamtych czasach nazywał je rekurencyjnymi. Wcześniej była znana funkcja Ackermana i było wiadomo, że nie można jej zdefiniować przez rekursję prostą.

Nie jest jasne, kiedy Kurt Gödel zauważył związek funkcji pierwotnie rekurencyjnych z obliczalnością. Podczas wykładu w Princeton wiosną 1936 roku, już na początku przedstawienie definicji tych funkcji uzupełnił uwagą, że można je obliczać za pomocą mechanicznych procedur. Stwierdził też, że wie, jak rozszerzyć zaprezentowaną definicję, aby obejmowała wszystkie tak obliczane funkcje.

8.3 λ-definiowalność złożenia

Twierdzenie 8.1 *Klasa całkowitych funkcji lambda definiowalnych jest zamknięta ze względu na złożenie.*

Dowód. Może zostać przedstawiony na przykładzie. Przypuśćmy, że funkcja h jest złożeniem całkowitych funkcji f, g_1 i g_2 takim, że

$$h(m,n) = f(g_1(m,n), g_2(m,n)).$$

Niech F, G_1 i G_2 będą termami reprezentującymi (definiującymi) odpowiednio funkcje f, g_1 i g_2. Wtedy term

$$\lambda ab.F(G_1ab)(G_2ab)$$

definiuje funkcję h. Sprawdzenie warunku z definicji reprezentowalności nie nastręcza trudności. □

8.4 λ-definiowalność funkcji określanych przez iterację

Najpierw zajmiemy się szczególnym przypadkiem rekursji prostej zwanym iteracją, i będziemy posługiwać się numerałami Churcha. Operacja ta dla danych funkcji g i h pozwala zdefiniować funkcję f spełniającą warunki

$$f(m,0) = g(m) \quad \text{oraz} \quad f(m,n+1) = h(m, f(n)). \tag{1}$$

Twierdzenie 8.2 *Klasa całkowitych funkcji λ-definiowalnych jest zamknięta ze względu na iterację, a więc, jeżeli funkcje g i h są λ-definiowalne, to funkcja f definiowana wzorami (1) też jest λ-definiowalna.*

Dowód. Jeżeli termy G i H definiują odpowiednio funkcje g i h, to term $F = \lambda xy.\, y(H\,x)\,(G\,x)$ definiuje funkcję f. Uzasadnienie tego faktu wymaga wcześniejszego wyprowadzenia wzoru $c_n\,(H\,c_m)\,(G\,c_m) = c_{f(n,m)}$ Wzór ten otrzymujemy łatwo przez indukcję ze względu na n. Mamy bowiem

$$c_0\,(H\,c_m)\,(G\,c_m) = (\lambda fx.\,x)\,(H\,c_m)\,(G\,c_m) = G\,c_m = c_{g(m)} = c_{f(0,m)}$$

oraz

$$c_{n+1}\,(H\,c_m)\,(G\,c_m) = (H\,c_m)^{n+1}(G\,c_m) = Hc_m((H\,c_m)^n(G\,c_m)) =$$

$$= H\,c_m\,(c_n\,(H\,c_m)\,(G\,c_m)) = H\,c_m\,c_{f(n,m)} = c_{h(m,f(n,m))} = c_{f(n+1,m)}.$$

Z wyprowadzonego wzoru tezę otrzymujemy w następujący sposób:

$$F\,c_m\,c_n = (\lambda xy.\,y(H\,x)\,(G\,x))\,c_m\,c_n = c_n\,(H\,c_m)\,(G\,c_m) = c_{f(n,m)}.\ \square$$

8.5 λ-definiowalność rekursji prostej

Najpierw pokażemy, że rekursja prosta sprowadza się do iteracji. W tym celu możemy posłużyć się parami liczb naturalnych lub kodowaniem par liczb naturalnych. Przyjmijmy, że $\langle x, y\rangle$ oznacza parę liczb naturalnych (lub liczbę kodującą taką parę), a $(p)_0$ i $(p)_1$ oznaczają odpowiednio pierwszą i drugą współrzędną pary p.

Weźmy funkcję f taką, że

$$f(m,0) = g(m) \;\text{ oraz }\; f(m,n+1) = h(m,n,f(m,n)) \tag{2}$$

i zdefiniujmy funkcję pomocniczą t (można myśleć, że $t : N^2 \to N^2$)

$$t(m,p) = \langle (p)_0 + 1, h(m,(p)_0,(p)_1)\rangle.$$

Jest oczywiste, że

$$t(m,\langle x,y\rangle) = \langle x+1, h(m,x,y)\rangle \;\text{ oraz }\; t(m,\langle n, f(m,n)\rangle) = \langle n+1, f(m,n+1)\rangle.$$

Iterując funkcję t możemy zdefiniować funkcję f' taką, że

$$f'(m,0) = \langle 0, g(m)\rangle \;\text{ oraz }\; f'(m,n+1) = t(m,f'(m,n)).$$

Przez indukcję ze względu na n można wykazać, że

$$f'(m,n) = \langle n, f(m,n)\rangle \;\text{ i w konsekwencji }\; f(m,n) = (f'(m,n))_1.$$

Wyżej przedstawiony sposób definiowania funkcji f wykorzystamy do napisania λ-termu reprezentującego f.

Twierdzenie 8.3 *Klasa całkowitych funkcji λ-definiowalnych jest zamknięta ze względu na definiowanie przez rekursję prostą.*

Dowód. Najpierw musimy przypomnieć sobie odpowiedni aparat. W rozdziałach 7.4 i 7.5 przyjeliśmy, że

$$[M,N] = \lambda x.xMN, \quad \mathbf{true} = \lambda xy.x \quad \text{oraz} \quad \mathbf{false} = \lambda xy.y.$$

Oczywiście, mamy wtedy $[M,N]\,\mathbf{true} = M$ oraz $[M,N]\,\mathbf{false} = N$.

Pokażemy (znowu na przykładzie) λ-definiowalność funkcji f danej wzorami (2)

$$f(m,0) = g(m) \;\text{ oraz }\; f(m,n+1) = h(m,n,f(m,n))).$$

Przyjmijmy, że G i H są termami definiującymi odpowiednio g i h.

Nawiasy kwadratowe oznaczają rodzaj funkcji pary. Niech t oznacza funkcję taką, że

$$t(m,[x,y]) = [x+1, h(m,x,y)].$$

W rachunku lambda taką funkcję reprezentuje term

$$T = \lambda uvw.\, w\,(S^+(v\,\mathbf{true}))\,(H\,u\,(v\,\mathbf{true})\,(v\,\mathbf{false})).$$

Pokażemy najpierw, że $T\,c_m\,[c_n,c_f] = [c_{n+1}, c_{h(m,n,f)}]$. Mamy

$$T\,c_m\,[c_n,c_f] = \lambda w.\, w\,(S^+([c_n,c_f]\,\mathbf{true}))\,(H\,c_m\,([c_n,c_f]\,\mathbf{true})\,([c_n,c_f]\,\mathbf{false})) =$$

$$= \lambda w.\, w\,(S^+ c_n)\,(H\, c_m c_n c_f) = [c_{n+1}, H\, c_m\, c_n\, c_f,] = [c_{k+1}, c_{h(m,n,f)}].$$

Stąd przez indukcję otrzymujemy, że

$$(T\, c_m)^n([c_0, c_{g(m)}]) = [c_n, c_{f(m,n)}]$$

Zauważmy, że $n = 0$ równość ta jest oczywista. Ponadto mamy

$$(T\, c_m)^{n+1}([c_0, c_{g(m)}]) = T\, c_m\, [c_n, c_{f(m,n)}] = [c_{n+1}, c_{h(m,n,f(m,n))}] = [c_{n+1}, c_{f(n+1,m)}].$$

Teraz zdefiniujmy term F jako

$$F = \lambda xy.\, (y\,(Tx)\,[c_0, Gx])\ \mathbf{false}.$$

Dla tego termu z wyżej udowodnionego wzoru otrzymujemy

$$F\, c_m\, c_n = c_n\,(T\, c_m)\,[c_0, Gc_m]\,\mathbf{false} = (T\, c_m)^n([c_0, c_{g(m)}])\,\mathbf{false} = [c_n, c_{f(m,n)}]\,\mathbf{false} = c_{f(m,n)}.$$

Równość ta świadczy o tym, że term F definiuje funkcję f. □

Wniosek 8.4 *Funkcja poprzednik $p : N \to N$ taka, że $p(0) = 0$ oraz $p(n+1) = n$ jest definiowalna w rachunku lambda.*

Dowód. Jest to wniosek ważny z historycznego punktu widzenia. Znalezienie dowodu tego faktu sprawiało trochę kłopotów.

Funkcję p można zdefiniować przez rekursję prostą („pustą", nie odwołującą się do wartości poprzedniej) przyjmując, że

$$p(0) = 0 \ \text{ oraz } \ p(n+1) = U_2^1(n, p(n)).$$

Wystarczy więc powtórzyć wyżej przedstawioną konstrukcję w tym przypadku. Powinniśmy otrzymać term

$$P^- = \lambda x\,(x\,(\lambda a\,(\lambda z\, z\,(S^+\,(a\,\mathbf{true}))\,(a\,\mathbf{true}))\,(\lambda z\, z\, c_0\, c_0))\,\mathbf{false}.\ \square$$

Wniosek 8.5 *Numerały Churcha można uzupełnić do systemu liczbowego.*

Dowód. Aby numerały Churcha tworzyły system liczbowy potrzebne są trzy termy Z, S^+ i P^-. Term P^- został zdefiniowany w poprzednim dowodzie. Term S^+ może być równy $\lambda a f x.\, f\,(a\, f\, x)$ (patrz rozdział 7.2). Termem Z może być $\lambda a\, a\,(\mathbf{true}\,\mathbf{false})\,\mathbf{true}$. □

8.6 Pierwsze twierdzenie o λ-reprezentowalności

Twierdzenie 8.6 *Wszystkie funkcje pierwotnie rekurencyjne są reprezentowalne w rachunku lambda.*

Dowód. Wobec twierdzeń 8.1 i 8.3, aby dowieść twierdzenie wystarczy podać termy definiujące funkcje Z, S i U_k^n (patrz rodział 8.1). Są nimi odpowiednio termy $\lambda a\,\mathbf{true}\, c_0\, a$, S^+ oraz $\lambda x_1, \ldots, x_n.x_k$. Zauważmy jeszcze, że w ten sposób dowiedliśmy twierdzenie przy założeniu, że liczby naturalne są reprezentowane za pomocą numerałów Churcha. □

9 Funkcje rekurencyjne

9.1 Operacja minimum

Operacja minimum przyporządkowuje funkcji naturalnej $g : N^{k+1} \to N$ funkcję $f : N^k \to N$ oznaczaną zwykle wzorem

$$f(\vec{m}) = \mu n \, (g(\vec{m}, n) = 0)$$

i przyjmującą dla argumentu $\vec{m} = (m_1, m_2, \ldots, m_k)$ wartość będącą najmniejszą liczbą n spełniającą warunek $g(\vec{m}, n) = 0$.

Definicję tą będziemy stosować dla różnych rodzajów funkcji g. W najbardziej ogólnym przypadku może to być dowolną naturalną funkcją częściową. Wtedy też mogą pojawić się wątpliwości związane z definicją f. Mamy jednak naturalny algorytm pozwalający na obliczanie wartości f:

```
n:= 0;
while g(m⃗,n) ≠ 0 do n := n + 1;
return(n)
```

Może on nie zakończyć pracy z dwóch powodów: gdy dla pewnych danych nie zakończą się obliczenia wartości funkcji g, lub gdy wartości $g(\vec{m}, n)$ będą różne od 0 dla wszystkich n. Przyjmujemy, że ten algorytm zawsze poprawnie oblicza wartości funkcji f. Jeżeli dla pewnych danych $\vec{m}$ nie kończy pracy, to wartość $f(\vec{m})$ nie jest określona. W przeciwnym razie zwraca poprawną wartość $f(\vec{m})$.

O operacji minimum mówimy, że jest efektywna, jeżeli stosujemy ją wyłącznie do całkowitych funkcji g takich, że

$$\forall \vec{m} \in N^k \; \exists n \in N \;\; g(\vec{m}, n) = 0.$$

Będziemy też rozważać operację minimum zawężoną do całkowitych funkcji g.

9.2 Funkcje rekurencyjne

Klasa całkowitych funkcji rekurencyjnych to najmniejsza klasa funkcji zawierającą Z, S oraz U_k^n i zamknięta ze względu na złożenie, rekursję prostą i efektywną operację minimum.

Klasa (częściowych) funkcji rekurencyjnych jest najmniejszą klasą funkcji zawierającą Z, S oraz $U_{n,k}$ i zamkniętą ze na złożenie, rekursję prostą i operację minimum.

Definicja klasy funkcji rekurencyjnych wymaga wyjaśnienia, co to jest złożenie funkcji częściowych i jak definiujemy przez rekursję prostą w przypadku takich funkcji. Te pojęcia można wyjaśnić podobnie, jak operację minimum, wskazując naturalne algorytmy i żądając zgodności definicji i obliczeń za pomocą odpowiedniego algorytmu.

Klasę funkcji rekurencyjnych można definiować na wiele sposobów. Na przykład, można w definicji tej klasy nie wspominać o rekursji prostej. Rekursja prosta jest potrzebna do zdefiniowania trzech funkcji: dodawania, mnożenia i funkcji charakterystycznej relacji nierówności. Mając te trzy funkcje, możemy za pomocą liczb naturalnych kodować ciągi liczb naturalnych. To z kolei pozwala rekursję prostą zastąpić operacją minimum.

Można też ograniczać rolę operacji minimum. Możemy stosować ją wyłącznie do funkcji całkowitych. Bardzo silny rezultat tego typu wyraża

Twierdzenie 9.1 (o postaci normalnej Kleene'ego) *Jeżeli f jest częściową funkcją rekurencyjną, to istnieją dwie pierwotnie rekurencyjne (a więc całkowite) funkcje g i h takie, że*

$$f(\vec{m}) = h(\mu n(g(\vec{m}, n) = 0)).$$

9.3 Jeszcze raz kilka uwag historycznych

Historia rachunku λ wiąże się z poszukiwaniem negatywnego rozwiązania problemu znanego jako *Entscheidungsproblem*, sformułowanego przez Dawida Hilberta. Takie rozwiązanie wymaga precyzyjnego zdefiniowania pojęcia algorytmu lub opisania wszystkich funkcji, które mogą być obliczane za pomocą jakkolwiek rozumianych algorytmów.

Wszystkie funkcje, które mogą być obliczane w sposób zalgorytmizowany, będziemy nazywać obliczalnymi. Pojęcie to będziemy rozumieć szeroko, powinno obejmować funkcje obliczane za pomocą mechanicznych procedur, komputerów i podobnych urządzeń, nawet takich, które dopiero zostaną zaprojektowane, maszyn Turinga, formalnych rachunków możliwych do przeprowadzenia na karte papieru i innych, podobnych sposobów.

Z teoretycznegu punktu widzenia, badając funkcje obliczalne możemy ograniczyć się (i zwykle to robimy) do funkcji naturalnych, określonych na zbiorze liczb naturalnych i przyjmujących wartości będące liczbami naturalnymi. Argumentem może być to, że dane dla komputerów i wyniki ich pracy są zapisywane jako ciągi zer i jedynek, czyli przedstawienia dwójkowe liczb naturalnych, a same obliczenia komputerowe polegają na manipulacjach takimi ciągami.

Definicje rekurencyjne są rozumiane na dwa sposoby. Starszy jest bardziej istotny, wydaje się bardziej powszechny, zwłaszcza na początku XX wieku. Polega na traktowania definicji jako dodatkowego zbioru aksjomatów. Aksjomaty są równościami i pozwają wyprowadzać inne równości (równość $f(m, 0) = g(m)$ oznacza również, że wyliczenie $f(m, 0)$ wymaga wyliczenia $g(m)$ i obie te wartości są równe). Takie podejście odsuwa na dalszy plan kwestię niesprzeczności aksjomatów. Zauważmy, że podobnie, podając odpowiednią równość, definiujemy złożenie funkcji. Tak mógł myśleć Kurt Gödel podczas swojego wykładu w Princenton.

Kurt Gödel obiecał, że rozszerzy definicję klasy funkcji (pierwotnie) rekurencyjnych tak, aby obejmowała wszystkie funkcje obliczalne. Wywiązał się z obietnicy podczas jednego z ostatnich wykładów i zaproponował, aby rekurencyjnymi nazywać wszystkie funkcje definiowane równościami, a więc definiowane bardziej skomplikowanymi schematami rekursji (np. definiującym funkcję Ackermanna) i wszelkimi innymi sensownymi układami. Dzisiaj tak definiujemy funkcje nazywane rekurencyjnymi według Herbranda i Gödla (Herbrand miał sugerować takie rozszerzenie w liście do Gödla, a list miał zaginąć podczas II wojny światowej. List zachował się jednak i nie zawiera takich sugestii). Przytoczona wyżej definicja funkcji rekurencyjnych, pomijając szczegóły, jest opracowana przez Stephena Kleene'ego. Została wykorzystana w argumentacji uzasadniającej tezę Churcha. Obie definicje, według Herbranda i Gödla oraz Kleene'ego, są równoważne.

Zauważmy jeszcze, że funkcja $f(m) = \mu n\,(g(m, n) = 0)$ można zdefiniować następującym układem równości, uzupełnionym o równościową definicję funkcji g:

$$f(m) = h(m, 0), \;\; h(m, n) = k(m, n, g(m, n)),$$

$$k(m, n, 0) = n, \;\; k(m, n, k + 1) = h(m, n + 1).$$

9.4 Operacja minimum i λ-definiowalność

Przypuśćmy, że $g : N^2 \to N$ jest całkowitą funkcją spełniającą warunek efektywności ($\forall m \in N\ \exists n \in N\ g(m,n) = 0$) i niech $f(m) = \mu n\ (g(m,n) = 0)$. Załóżmy też, że term G reprezentuje w λ-rachunku funkcję g.

Obliczanie wartości $f(m)$ wymaga wykonania algorytmu następującej postaci;

1) przypisz 0 zmiennej n,

2) dla danej wartości n sprawdź, czy $g(m,n) = 0$ i

   jeżeli tak jest, to zakończ obliczenia i zwróć wartość n,

   w przeciwnym razie zwiększ wartość n o 1

3) powtórz czynności z poprzedniego punktu.

W λ-rachunku aplikacja MN odpowiada jakby wykonaniu procedury M z parametrem N. Term opisujący powyższy algorytm na pewno powinien otrzymać w charakterze parametru term reprezentujący wartość argumentu m, a być może również term reprezentujący aktualną wartość zmiennej n. Pewną trudnością jest wyrażenie w λ-rachunku ostatniego punktu naszego algorytmu, gdyż prowadzenie obliczeń modyfikuje, może nawet niszczy term określający te obliczenia. Rozwiązaniem może się okazać przekazanie termowi jego samego jako jednego z parametrów i wykorzystanie tego termu do zdefiniowania dalszych obliczeń. Wobec tego term wyrażający takie obliczenia powinien mieć postać $W\,W\,c_m\,c_0$, a term reprezentujący funkcję f to chyba

$$F \;=\; \lambda a\, W\, W\, a\, c_0.$$

Pozostaje więc zdefiniować term W. Term W powinien jakoś wyrażać podany algorytm, a wiec powinien mieć postać

$$W = \lambda wxy.\ \textbf{if}\ Z\,(G\,x\,y)\ \textbf{then}\ y\ \textbf{else}\ w\,w\,x\,(S^+\,y) = \lambda wxy.\ Z\,(G\,x\,y)\,y\,(w\,w\,x\,(S^+\,y)),$$

gdzie Z to term z definicji systemu liczbowego sprawdzający, czy dane przedstawienie liczby jest przedstawieniem zera.

Twierdzenie 9.2 *Niech $g : N^2 \to N$ będzie całkowitą funkcją definiowaną w λ-rachunku i spełniającą warunek efektywności ($\forall m \in N \exists n \in N\ g(m,n) = 0$). Wtedy funkcja $f : N \to N$ taka, że $f(m) = \mu n\,(g(m,n) = 0)$ jest λ-definiowalna, a dokładniej, jest ona definiowana wyżej podanym termem F.*

Dowód. Wystarczy sprawdzić nasze intuicje. Mamy

$$W\,W\,c_m\,c_n \;=\; Z\,(G\,c_m\,c_n)\,c_n\,(W\,W\,c_m\,(S^+c_n)) \;=\; Zc_{g(m,n)}\,c_n\,(W\,W\,c_m\,c_{n+1}) \;=$$

$$= \begin{cases} c_n & \text{jeżeli } g(m,n) = 0, \\ W\,W\,c_m\,c_{n+1} & \text{w przeciwnym razie.} \end{cases}$$

Z udowodnionego wzoru wynika, że jeżeli $g(m,k) \neq 0$ dla wszystkich $k < n$, to

$$F\,c_m \;=\; W\,W\,c_m\,c_n.$$

Jeżeli więc teraz za n weźmiemy najmniejszą liczbę taką, że $g(m,n) = 0$, czyli $n = f(m)$, to mamy

$$F\,c_m \;=\; W\,W\,c_m\,c_n \;=\; c_n \;=\; c_{f(m)}.\ \square$$

9.5 Drugie twierdzenie o reprezentowalności

Twierdzenie 9.3 *Wszystkie całkowite funkcje rekurencyjne są reprezentowalne (definiowalne) w rachunku lambda.*

Dowód. Jest niemal identyczny, jak dowód twierdzenia 8.6. Dodatkowo musimy skorzystać z twierdzenia 9.2, a właściwie z jego uogólnienia. □

Prawdziwe jest także twierdzenie odwrotne.

Twierdzenie 9.4 *Każda (całkowita) funkcja naturalna, reprezentowalna w rachunku lambda jest rekurencyjna.* □

Twierdzenia tego nie dowiedziemy w tej chwili z dwóch powodów. Po pierwsze dlatego, że nie zachodzi, jeżeli rachunek λ jest sprzeczny. Musimy więc najpierw zbadać niesprzeczność λ-rachunku. Także dlatego, że dowód jest żmudny, wymaga rozbudowanej teorii funkcji rekurencyjnych i niewiele wnosi do rachunku λ.

Równoważność rekurencyjności i definiowalności w λ-rachunku została wykorzystana do uzasadnienia tezy Churcha. Dzisiaj nam mówi, że w rachunku λ można obliczać wszystkie funkcje rekurencyjne i tylko takie, czyli dokładnie te funkcje naturalne, które można obliczać w dowolnym języku programowania.

9.6 Teza Churcha

David Hilbert przyjmował, że *Entscheidungsproblem zostanie rozwiązany, gdy będzie znana procedura, za pomocą której będzie można decydować w skończonej liczbie operacji, czy dane wyrażenie logiczne jest ogólnie prawdziwe, bądź czy jest spełnialne.* Negatywne rozwiązanie tego problemu wymaga precyzyjnego ustalenia, jakie zadania mogą być rozwiązywane za pomocą takich procedur i matematycznego sformalizowania pojęcia problemu obliczalnego w sensie intuicyjnym. Zamiast problemów można też sformalizować funkcje obliczalne.

Alonzo Churchowi wydawało się, że do sformalizowania pojęcia funkcji obliczalnej można wykorzystać rachunek lambda. Musiał jednak znaleźć przekonywującą argumentację świadczącą o słuszności swoich intuicji. Taką argumentację podpowiedział mu Kurt Gödel, który uważał, że obliczalnymi mogą być funkcje naturalne definiowane rekurencyjnie na wszelkie możliwe sposoby, a może nawet funkcje definiowane wszelkimi układami równań. Propozycja Kurta Gödla została przeanalizowana w zespole Alonzo Churcha (głównie przez Stephena Klenee'ego), doprowadziła stworzenia do współcześnie używanej definicji funkcji rekurencyjnych i udowodnienia twierdzenia o równoważności rekurencyjności i λ-definiowalności.

W kwietniu 1935 roku Alonzo Church na posiedzeniu American Mathematical Society zaproponował uznanie klasy całkowitych funkcji rekurencyjnych za dobrą formalizację pojęcia całkowitej funkcji obliczalnej. Stwierdzenie to zyskało miano tezy Churcha. Głównym argumentem przemawiającym za tą tezą było wspomniane twierdzenie o równoważności rekurencyjności i λ-definiowalności. Church uważał, że zapewne nie jest przypadkiem, że dwie niezależne próby formalizacji obliczalności doprowadziły do tego samego rezultatu. Mniej więcej w tym samym czasie została ogłoszona analogiczna teza Turinga utożsamiająca pojęcie obliczalności z obliczalnością na maszynach Turinga. Nieco poźniej Alan Turing udowodnił też równoważność obliczalności na maszynach Turinga i rekurencyjności.

Mając już stosowny aparat pojęciowy Alonzo Church dowiódł twierdzenie o nierozstrzygalności arytmetyki i twierdzenie o nierozstrzygalności rachunku kwantyfikatorów, i tym samym rozwiązał negatywnie *Entscheidungsproblem*.

Do dnia dzisiejszego teza Churcha nie została podważona.

Informatycy o obliczalności mówią na cztery, równoważne sposoby: rozważają zbiory rozstrzygalne i semi-rozstrzygalne, oraz całkowite i częściowe funkcje obliczalne. Poniższa tabela podaje wynikającą z tezy Churcha zależność między obliczalnością i rekurencyjnością (przy założeniu, że obliczalność ograniczamy do zbiorów liczb naturalnych i do funkcji naturalnych).

całkowite funkcje obliczalne	całkowite funkcje rekurencyjne
częściowe funkcje obliczalne (obliczane za pomocą algorytmu, który może się zapętlić)	częściowe funkcje rekurencyjne
zbiory rozstrzygalne	zbiory rekurencyjne (mają całkowitą i rekurencyjną funkcję charakterystyczną)
zbiory semirozstrzygalne	zbiory rekurencyjnie przeliczalne (rzuty zbiorów rekurencyjnych lub zbiory wartości całkowitych funkcji rekurencyjnych)

10 Punkty stałe

10.1 Podstawowe definicje

Zacznijmy od definicji. Term F nazywamy punktem stałym termu (aplikacji, kombinatora) M, jeżeli $F = MF$.

Term C nazywamy kombinatorem punktu stałego, jeżeli dla wszystkich termów M mamy $CM = M(CM)$.

10.2 Po co są punkty stałe?

Definiowalność w λ-rachunku funkcji określanych przez rekursję została już wykazana, ale dowód wymagał reprazentowania liczb naturalnych za pomocą numerałów Churcha. Spróbujemy przeprowadzić go jakoś inaczej, bez numerałów Churcha.

Przyjmijmy, że

$$f(m,0) = g(m) \;\text{ oraz }\; f(m,n+1) = h(m,n,f(m,n)) \tag{3}$$

dla wszystkich $m, n \in N$, a także, że w definicji f zostały użyte funkcje g i h reprezentowane w λ-rachunku odpowiednio za pomocą termów G i H. Umówmy się też, że liczbę naturalną n reprezentujemy za pomocą termu c_n, ale niekoniecznie jest to numerał Churcha.

Definicję funkcji f można uznać też za warunkową i zapisać jakoś tak:

$$f(m,n) \;=\; \textbf{if } n = 0 \textbf{ then } g(m) \textbf{ else } h(m, n-1, f(m,n-1)).$$

Taka równość daje się wyrazić w λ-rachunku w postaci

$$Fc_mc_n = \textbf{if } Zc_n \textbf{ then } Gc_m \textbf{ else } Hc_m(P^-c_n)(Fc_m(P^-c_n)) = Zc_n(Gc_m)(Hc_m(P^-c_n)(Fc_m(P^-c_n))).$$

Bardzo łatwo przekonać się, że jeżeli term F spełnia powyższe równości dla wszystkich $m, n \in N$, to reprezentuje w λ-rachunku funkcję f.

Jeżeli teraz przyjmiemy, że

$$M = Zy(Gx)(Hx(P^{-}y)(fx(P^{-}y))),$$

to powyższą równość zapiszemy również w postaci $(x, y \notin FVM)$

$$Fc_m c_n = M[f := F])[x := c_m][y := c_n].$$

Wystarczy, aby zachodziła dla wszystkich m i n, ale być może mogłaby zachodzić także dla innych termów, także dla zmiennych x i y. Wtedy

$$Fxy = M[f := F].$$

Gdyby jeszcze term F okazał się abstrakcją $\lambda xy. \ldots$, to mielibyśmy

$$F = \lambda xy.Fxy = \lambda xy.M[f := F] = (\lambda fxy.M)F.$$

Tak więc mając punkt stały termu $\lambda fxy.M$ (spełniający jeszcze kilka drobnych warunków) moglibyśmy w λ-rachunku reprezentować funkcję f.

10.3 Pierwsza próba konstrukcji punktu stałego

Funkcje definiowane przez rekursję można programować używając pętli while i podobnych. Spróbujemy więc przeprowadzić (raz jeszcze) rozumowanie wykorzystane do wyrażenia operacji minimum.

Nasz cel w rachunku lambda można sobie wyobrazić jako stworzenie termu $M(M(M \ldots (ME) \ldots))$, gdzie E odpowiada jakimś czynnościom kończącym budowanie termu lub wykonywanych na początku podczas obliczania wartości termu. Przypuśćmy, że chcemy w tym celu naśladować wykonanie instrukcji *while* C *do* M (dopóki spełniony jest warunek C wykonuj instrukcję M). Wykonanie instrukcji *while* polega na sprawdzeniu, czy jest spełniony warunek C i w zależności od wyniku testu, albo zakończenie wykonywania (ewentualnie wykonujemy jakieś E), albo też ponowne wykonanie instrukcji *while* C *do* M. Przyjmując, że term W odpowiada instrukcji *while*, powinien on spełniać równość

$$WCM = \textbf{if } C \textbf{ then } M(WCM) \textbf{ else } E) = C(M(WCM))E.$$

Term W nie może być częścią termu W, jednak obliczanie termu W wymaga na ogół ponownego wyliczania W. Aby umożliwić to ponowne wyliczanie, term W możemy dodatkowo przekazać jako jeden z parametrów i wykorzystać do określenia sposobu kontynuacji obliczeń. Tak więc powyższa równość powinna mieć raczej postać

$$WWCM = \textbf{if } C \textbf{ then } M(WWCM) \textbf{ else } E) = C(M(WWCM))E, \qquad (4)$$

a term W powinien być definiowany wyrażeniem

$$W\, w\, c\, m \;=\; c\,(m(w\, w\, c\, m))\, E.$$

Abstrakcja łatwo pozwala zdefiniować taki term w λ-rachunku. Przyjmijmy więc, że

$$W \;=\; \lambda wcm.\, c\,(m(w\, w\, c\, m))\, E.$$

Zauważmy, że taki term W spełnia równość (4), a term WW jest pewnego rodzaju punktem stałym.

10.4 O rekursji prostej raz jeszcze

Spróbujemy teraz wykorzystać luźne rozważania z poprzedniego rozdziału 10.3 do zdefiniowania termu reprezentującego funkcję f daną wzorami (3), określoną przez rekursję prostą. Najpierw w tych rozważaniach wprowadzimy pewne zmiany. Po pierwsze, zamiast C w definicji W użyjemy negacji C (wystarczy zamiast $C(M(WCM))E$ napisać $CE(M(WCM))$, negację C łatwiej potrafimy wyrazić). Po drugie, ponieważ C i M są konkretnymi termami, wpiszemy je do termu W i zamiast WCM będziemy pisać W. W końcu dodamy dwa parametry x i y odpowiadające argumentom funkcji f. W wyniku tych zmian, przyjmując $C = Zy$ (Z to test 0), $E = Gx$ i $M = Hx(P^-y)$ (P^- reprezentuje poprzednik), otrzymujemy term

$$W = \lambda wxy.\, Z\, y\, (G\, x)(H\, x\, (P^-y)\, (w\, w\, x\, (P^-y)).$$

Mając term W wystarczy pokazać

Twierdzenie 10.1 *Dla wszystkich liczb naturalnych m i n zachodzi wzór*

$$W\, W\, c_m\, c_n \;=\; c_{f(m,n)},$$

gdzie funkcja f jest definiowana wzorem (3). Oznacza to, że term WW reprezentuje w λ-rachunku funkcję f.

Dowód. Równość z tezy dowodzimy przez indukcję ze względu na n. Dla $n = 0$ mamy

$$WWc_mc_0 = Zc_0(Gc_m)(Hc_m(P^-c_0)(WWc_m(P^-c_0)) = Gc_m = c_{g(m)} = c_{f(m,0)}.$$

Drugi krok indukcyjny wynika z równości

$$WWc_mc_{n+1} = Hc_mc_n(WWc_mc_n) = Hc_mc_nc_{f(m,n)} = c_{h(m,n,f(m,n))} = c_{f(m,n+1)}. \quad \square$$

Nietrudno zauważyć, że rozumowanie z powyższego dowodu można przeprowadzić dla dowolnego systemu liczbowego. Wynika stąd, że także twierdzenia 8.6 i 9.3 zachodzą dla wszystkich systemów liczbowych.

10.5 Twierdzenie o punkcie stałym

Punkt stały F termu M, czyli $F = MF = M(M(M\ldots))$ chyba możemy wyobrażać sobie jako efekt wykonania instrukcji *while* **true** *do* M. Przeglądając raz jeszcze rozważania z rozdziału 10.3 dostosujmy do niej definicję termu W. Warunek C powinniśmy zastąpić przez **true**, a właściwie możemy usunąć C razem z E. W ten sposób otrzymujemy term

$$W \;=\; \lambda wm.\, m\, (w\, w\, m).$$

Zauważmy, że spełnia on równości

$$WWM \;=\; (\lambda wm.\, m\, (w\, w\, m))WM = M(WWM).$$

W ten sposób dowiedliśmy

Twierdzenie 10.2 (o punkcie stałym) *Dla każdego λ-termu M istnieje term F taki, że $F = MF$. Co więcej, istnieje też kombinator punktu stałego, a więc taki term, który zaaplikowany do termu M daje punkt stały M.*

Dowód. Z udowodnionych wyżej równości wynika, że term WW jest kombinatorem punktu stałego. □

Kombinator punktu stałego WW jest nazywany kombinatorem Turinga i oznaczamy symbolem $\mathbf{\Theta}$. Mamy więc

$$\mathbf{\Theta} \;=\; (\lambda wm.\, m\,(w\,w\,m))(\lambda wm.\, m\,(w\,w\,m)).$$

Term W możemy jeszcze bardziej uprościć, jeżeli będziemy konstruować punkt stały wyłącznie termu M. W takiej sytuacja, zamiast uważać M za jeden z argumentów W możemy M uznać za część W. Prowadzi to do następującej definicji:

$$W' \;=\; \lambda w\, M(w\,w).$$

Mamy oczywiście

$$W'W' \;=\; (\lambda w\, M(w\,w))\, W' \;=\; M(W'W').$$

Oznacza to, że term $W'W'$ jest punktem stałym M. Możemy też przekształcić go w kombinator punktu stałego biorąc

$$\mathbf{Y} \;=\; \lambda m\,(\lambda w\, m(w\,w))(\lambda w\, m(w\,w)).$$

Wtedy $\mathbf{Y}M$ jest oczywiście punktem stałym M. Term $\mathbf{Y}$ nazywamy kombinatorem paradoksalnym Curry'ego.

10.6 Inne myślenie o punktach stałych

Jednym z najważniejszych osiągnięć twórcy teorii mnogości Georga Cantora jest twierdzenie o zbiorze potęgowym, które (w podstawowej postaci) mówi, że nie istnieje funkcja $f : N \to \mathcal{P}(N)$, której wartościami są wszystkie możliwe zbiory liczb liczb naturalnych. Dowód tego twierdzenia wymaga metody przekątniowej. Mając funkcję $f : N \to \mathcal{P}(N)$ tworzymy nieskończoną tablicę, której wiersze i kolumny są indeksowane liczbami naturalnymi. W n-tym wierszu tej tablicy umieszczamy informacje, czy kolejne liczby naturalne należą do zbioru $f(n)$, a więc, czy

$$0 \in f(n),\; 1 \in f(n),\; 2 \in f(n)\;\ldots$$

Wobec tego, na przekątnej tej tablicy znajdują się informacje, czy

$$0 \in f(0),\; 1 \in f(1),\; 2 \in f(2),\;\ldots$$

Zanegujmy te informacje i weźmy zbiór $A = \{n \in N :\; n \notin f(n)\}$. Zbiór ten nie jest wartością funkcji f, zbiory $f(n)$ i A różnią się liczbą n, n do jednego z tych zbiorów należy, a do drugiego – nie.

Bertrand Russel uogólnił rozumowanie Cantora i otrzymał antynomię Russela. Wydaje się, że zamiast nieskończonej tablicy indeksowanej liczbami naturalnymi rozważał olbrzymią tablicę indeksowaną wszystkimi możliwymi zbiorami. Analogicznie zdefiniował "zbiór" $A = \{x :\; x \notin x\}$, który jest różny od wszystkich istniejących zbiorów. Spowodowało to doprecyzowanie aksjomatów teorii mnogości i dzisiaj definicje postaci $\{x :\; \varphi(x)\}$ stosujemy tylko dla kilku konkretnych formuł $\varphi(x)$ (np. poprawną definicją jest $\{x :\; x = R\}$, gdzie R to zbiór liczb rzeczywistych), oraz dla formuł spełniających dodatkowe warunki (np. formuł postaci takich, jak w definicji $\{x \in X :\; \varphi(x)\} = \{x :\; x \in X \wedge \varphi(x)\}$).

W podobny sposób, mając ciąg $f_0, f_1, f_2, \ldots$ funkcji $f_i : N \to N$ możemy zdefiniować funkcję naturalną g, która w tym ciągu nie występuje. Wystarczy zagwarantować jakoś, że liczby $g(n)$ i $f_n(n)$ są różne, na przykład możemy przyjąć, że $g(n) = f_n(n) + 1$ dla wszystkich $n \in N$.

Analogiczne rozumowanie spróbujemy przeprowadzić w algebrach kombinatoryjnych. Weźmy wyrażenie $z(xx)$: chcemy xx zamienić na coś innego, więc do xx aplikujemy odpowiednią funkcję z. W algebrze kombinatoryjnej dla dowolnego wyrażenia (ze zmienną x), w szczególności dla $z(xx)$, możemy znaleźć taki element g, że $gx = z(xx)$ dla wszystkich x. Jeżeli teraz przyjmiemy, że $x = g$, to otrzymamy równość

$$gg = z(gg).$$

Tak więc gg okazało się punktem stałym z, a gdyby istniało z takie, że $zx \neq x$ dla wszystkich x, to uzyskalibyśmy sprzeczność. W ten sposób dowiedliśmy twierdzenie o punkcie stałym dla algebr kombinatoryjnych. Analogiczne rozumowanie można przeprowadzić w λ-rachunku.

W rachunku lambda rolę funkcji g wyznaczonej przez term M przejmuje abstrakcja $\lambda x\, M(x\, x)$. Redukując $(\lambda x\, M(x\, x))(\lambda x\, M(x\, x))$ łatwo sprawdzić, że jest to punkt stały M. Term będący punktem stałym M podpowiada też metodę konstrukcji punktu stałego: aby skonstruować punkt stały M należy do M zaaplikować term $\lambda z\, (\lambda x\, z(x\, x))(\lambda x\, z(x\, x))$, który – jak widać – jest znanym kombinatorem paradoksalnym Curry'ego $\mathbf{Y}$.

11 β-redukcja i drugie podejście do λ-rachunku

Jeżeli λ-rachunek uważamy za teorię równościową, to raczej nie zastanawiamy się, jak dowieść konkretną równość. Udowodnimy ją, jeżeli wykażemy się dostateczną pomysłowością. Teraz zaczniemy zajmować się analizą metod dowodzenia równości.

11.1 β-redukcja w jednym kroku

W λ-rachunku, jeżeli mamy redeks, czyli term postaci $(\lambda x\, M)N$, to możemy go przekształcić w redukt, czyli w term postaci $M[x := N]$. To przekształcenie nazywamy β-redukcją i zapisujemy w formie relacji

$$(\lambda x\, M)N \to_\beta M[x := N].$$

Symbol $\to_\beta$ ma jednak nieco szersze znaczenie i oznacza coś więcej, a mianowicie rozszerzenie powyższej relacji do zgodniej z działaniami. W ogólności mamy więc

$$C[(\lambda x\, M)N] \to_\beta C[M[x := N]]$$

dla dowolnego kontekstu C i dowolnego redeksu $(\lambda x\, M)N$ (termy oczywiście rozumiemy jako klasy abstrakcji relacji α-konwersji). Tak zdefiniowaną relację $\to_\beta$ nazywamy β-redukcją w jednym kroku.

β-redukcję w jednym kroku możemy też zdefiniować jako najmniejszą relację $\to_\beta$ spełniającą dla dowolnych termów K, M i N następujące warunki:

1) $(\lambda x\, M)N \to_\beta M[x := N]$,

2) jeżeli $M \to_\beta N$, to $KM \to_\beta KN$ oraz $MK \to_\beta NK$,

3) jeżeli $M \to_\beta N$, to $\lambda x\, M \to_\beta \lambda x\, N$.

11.2 β-redukcja

Jak już wiadomo, β-redukcją nazywamy zwrotne i przechodnie domknięcie β-redukcji w jednym kroku. β-redukcję oznaczamy symbolem $\twoheadrightarrow_\beta$. Jest więc to najmniejsza relacja spełniająca dla wszystkich λ-termów K, M i N następujące warunki:

1) $M \twoheadrightarrow_\beta M$,

2) jeżeli $M \rightarrow_\beta N$, to $M \twoheadrightarrow_\beta N$,

3) jeżeli $K \twoheadrightarrow_\beta M$ i $M \twoheadrightarrow_\beta N$, to $K \twoheadrightarrow_\beta N$.

11.3 Lemat o podstawianiu

Dla relacji zgodnych z działaniami zachodzi następujący:

Lemat 11.1 *Jeżeli $\rightarrow$ jest zgodna z działaniami, to warunek $N_1 \rightarrow N_2$ implikuje, że $M[x := N_1] \rightarrow M[x := N_2]$.* $\square$

Relacja $R \subseteq \Lambda^2$ jest zgodna z podstawianiem, jeżeli dla dowolnych termów M_1, M_2 i N warunek $M_1 \, R \, M_2$ implikuje, że $M_1[x := N] \, R \, M_2[x := N]$.

Lemat 11.2 *Jeżeli relacja $\rightarrow$ jest zgodna z działaniami i z podstawianiem, to warunki $M_1 \rightarrow M_2$ i $N_1 \rightarrow N_2$ implikują, że $M_1[x := N_1] \rightarrow M_2[x := N_2]$.* $\square$

Lemat 11.3 *Jeżeli relacja jest zgodna z podstawianiem, to jej rozszerzenie do relacji zgodnej z działaniami też jest zgodne z podstawianiem. Jeżeli redukcja w jednym kroku $\rightarrow$ jest zgodna z podstawianiem, to zgodne z podstawianiem są reducja (w wielu krokach) $\twoheadrightarrow$ i odpowiadająca jej konwersja.* $\square$

Nietrudno zauważyć, że

$$((\lambda y\, P)Q)[x := N] = (\lambda y\, P[x := N])(Q[x := N]) \rightarrow_\beta$$
$$\rightarrow_\beta P[x := N][y := Q[x := N]] = P[y := Q][x := N]$$

na mocy lematu 3.4. Stąd

Lemat 11.4 (o podstawianiu dla β-redukcji) *Zachodzi implikacja*

$$M_1 \twoheadrightarrow_\beta M_2 \;\; \textit{oraz} \;\; N_1 \twoheadrightarrow_\beta N_2 \quad \Rightarrow \quad M_1[x := N_1] \twoheadrightarrow_\beta M_2[x := N_2],$$

a także

$$M_1 =_\beta M_2 \;\; \textit{orax} \;\; N_1 =_\beta N_2 \quad \Rightarrow \quad M_1[x := N_1] =_\beta M_2[x := N_2].$$ $\square$

11.4 β-konwersja i λ-rachunek

β-konwersja jest przechodnim i symetrycznym domknięciem β-redukcji, a więc jest najmniejszą relacją $=_\beta$ spełniającą

1) jeżeli $M \rightarrow_\beta N$, to $M =_\beta N$,

2) jeżeli $M =_\beta N$, to $N =_\beta M$,

3) jeżeli $K =_\beta M$ oraz $M =_\beta N$, to $K \rightarrow_\beta N$.

Główną własność β-konwersji wyraża

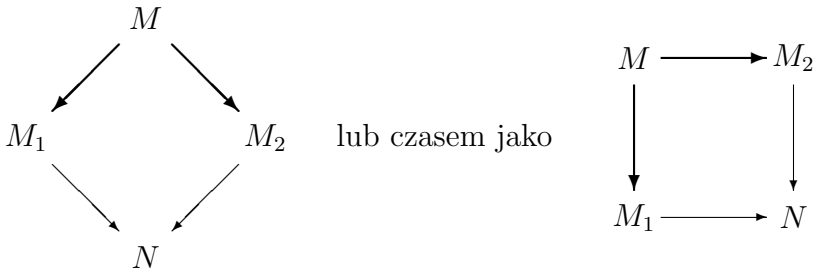
Lemat 11.5 *Dla dowolnych dwóch λ-termów M i N warunki $\lambda \vdash M = N$ oraz $M =_\beta N$ są równoważne.* $\square$

12 Twierdzenie Churcha-Rossera

12.1 Własność Churcha-Rossera

Symbol $\rightarrow$ będzie teraz oznaczać dowolną relację, choć w zastosowanich zwykle będzie jakąś redukcją, a podstawowe rezultaty będą dotyczyć β-redukcji i ewentualnie β-redukcji w jednym kroku.

Relacja $\rightarrow$ ma własność Churcha -Rossera (własność CR, własność rombu lub własność $\diamond$), jeżeli dla dowolnych elementów M, M_1 i M_2 z jej dziedziny (zwykle λ-termów) takich, że $M \rightarrow M_1$ i $M \rightarrow M_2$ istnieje N taki, że $M_1 \rightarrow N$ oraz $M_2 \rightarrow N$. Definicję tę przedstawia się zwykle w formie następującego diagramu:

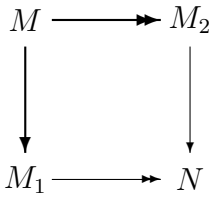


Na powyższych diagramach obowiązuje zasada, że pogrubione strzałki oznaczają relacje, o których wiemy, że zachodzą, zwykłe – takie, których istnienie stwierdzamy.

Relacja β-redukcji $\rightarrow_\beta$ nie ma własności Churcha-Rossera. Świadczy o tym przykład $(\lambda x\, xx)((\lambda x\, xx)M)$.

Lemat 12.1 *Jeżeli $\rightarrow$ ma własność Churcha-Rossera i $\twoheadrightarrow$ oznacza zwrotne i przechodnie domknięcie $\rightarrow$, to dla wszystkich M, M_1 i M_2 takich, że $M \rightarrow M_1$ i $M \twoheadrightarrow M_2$ istnieje N takie, że $M_1 \twoheadrightarrow N$ i $M_2 \rightarrow N$.*

Dowód. Na diagramie tezę lematu można przedstawić w następujący sposób:



Aby dowieść lemat weźmy relację

$$R = \{(M, M') :\ \forall M_1\ (M \rightarrow M_1 \Rightarrow \exists N\ M_1 \twoheadrightarrow N \wedge M' \rightarrow N)\}.$$

Jest oczywiste, że

1) jeżeli $M \rightarrow M'$, to $M\, R\, M'$ (na mocy założenia),

2) $M\, R\, M$ (w dowodzie wystarczy za N wziąć M_1),

3) jeżeli $M\, R\, M'$ i $M'\, R\, M''$, to także $M\, R\, M''$ (ma naturalne uzasadnienie).

Z tych trzech zasygnalizowanych stwierdzeń wynika, że relacja R ma własności wymagane od relacji $\twoheadrightarrow$. Stąd otrzymujemy, że relacja $\twoheadrightarrow$, jako najmniejsza taka relacja, jest zawarta w relacji R, czyli otrzymujemy, że warunek $M \twoheadrightarrow M_2$ implikuje, że $M\,R\,M_2$. To ostatnie stwierdzenie jest tezą naszego lematu. □

Lemat 12.2 *Jeżeli $\twoheadrightarrow$ oznacza zwrotne i przechodnie domknięcie $\rightarrow$ i dla wszystkich M, M_1 i M_2 takich, że $M \rightarrow M_1$ i $M \twoheadrightarrow M_2$ istnieje N takie, że $M_1 \twoheadrightarrow N$ i $M_2 \twoheadrightarrow N$ (np. gdy jest spełniona teza lematu 12.1), to relacja $\twoheadrightarrow$ ma własność Churcha-Rossera.*

Dowód. Analogiczny do poprzedniego. □

Przekonamy się nieco później, że z udowodnione teraz lematy mimo wszystko można wykorzystać w dowodzie twierdzenia Churcha-Rossera.

12.2 Twierdzenie Churcha-Rossera

Twierdzenie Churcha-Rossera ma dwie równoważne postaci. Pozornie słabsze jest

Twierdzenie 12.3 (Churcha-Rossera) *Relacja β-redukcji $\twoheadrightarrow_\beta$ ma własność Churcha-Rossera.*

Twierdzenie to dowiedziemy później (patrz rozdziały 12.5 i 13.4). Jego nieco mocniejszą wersją jest

Twierdzenie 12.4 (Churcha-Rossera) *Jeżeli $M_1 =_\beta M_2$, to dla pewnego termu N mamy $M_1 \twoheadrightarrow_\beta N$ i $M_2 \twoheadrightarrow_\beta N$.*

Dowód. Indukcyjny, korzystający z poprzedniego twierdzenia. Relacja β-konwersji $=_\beta$ jest najmniejszą symetryczną i przechodnią relacją zawierającą β-redukcję. Weźmy relację

$$R = \{(M_1, M_2) : \exists N\, M_1 \twoheadrightarrow_\beta N \wedge M_2 \twoheadrightarrow_\beta N\}.$$

Jest oczywiste, że relacja R jest symetryczna i zawiera β-redukcję. Z poprzedniego twierdzenia można wyprowadzić, że jest to relacja przechodnia. Stąd wynika, że R zawiera najmniejszą relację o tych trzech własnościach, czyli β-konwersję, a to zawieranie jest równoważne tezie naszego twierdzenia. □

12.3 Postać normalna termu

Postać normalną termu definiujemy dla ustalonej relacji redukcji. Najczęściej będzie to zwykła β-redukcja, ale będziemy rozważać postacie normalne także dla innych redukcji. Załóżmy, że mamy redukcję w jednym kroku $\rightarrow$, rozszerzyliśmy ją do redukcji w wielu krokach $\twoheadrightarrow$ i do relacji konwersji $=$.

Term M jest w postaci normalnej, jeżeli nie istnieje term N taki, że $M \rightarrow N$.

Term N jest postacią normalną termu M, jeżeli N jest w postaci normalnej i $M \twoheadrightarrow N$.

Term M ma postać normalną, jeżeli istnieje term N, który jest postacią normalną termu M.

Mamy też oczywisty fakt:

Fakt 12.5 Jeżeli term M jest w postaci normalnej i $M \twoheadrightarrow N$, to termy M i N są identyczne. □

12.4 Konsekwencje twierdzenia Churcha-Rossera

Lemat 12.6 *Każdy term ma najwyżej jedną postać normalną.*

Dowód. Przypuśćmy, że term M ma dwie postacie normalne M_1 i M_2. Na mocy twierdzenia Churcha-Rossera 12.3 istnieje term N taki, że $M_1 \twoheadrightarrow N$ i $M_2 \twoheadrightarrow N$. Z faktu 12.5 wynika, że termy M_1, N i M_2 są identyczne. □

Lemat 12.7 *Jeżeli N jest postacią normalną M_1 i $M_1 = M_2$ (M_1 jest konwertowalne do M_2), to N jest postacią normalną M_2.* □

Lemat 12.8 *Jeżeli M i N są różne (nieidentyczne) i są w postaci normalnej, to $M \neq N$ (M nie jest konwertowalny do N).* □

Z ostatniego lematu, jeżeli tylko udowodnimy twierdzenie Churcha-Rossera dla β-redukcji, będzie wynikać niesprzeczność rachunku λ.

12.5 Relacja równoległej β-redukcji

Relacja równoległej β-redukcji (w jednym kroku) $\rightarrow_{r\beta}$ jest najmniejszą relacją w zbiorze λ-termów spełniającą

1) $M \rightarrow_{r\beta} M$,

2) jeżeli $M \rightarrow_{r\beta} M_1$ i $N \rightarrow_{r\beta} N_1$, to $(\lambda x M)N \rightarrow_{r\beta} M_1[x := N_1]$,

3) jeżeli $M \rightarrow_{r\beta} M_1$ oraz $N \rightarrow_{r\beta} N_1$, to $MN \rightarrow_{r\beta} M_1 N_1$,

4) jeżeli $M \rightarrow_{r\beta} M_1$, to $\lambda x M \rightarrow_{r\beta} \lambda x M_1$.

Relacja równoległej redukcji odpowiada sytuacji, w której mając dany term wybieramy w nim kilka redeksów (także 0 redeksów), i redukujemy jednocześnie wszystkie wybrane.

Lemat 12.9 *Relacja $\rightarrow_\beta$ jest częścią równoległej β-redukcji, a więc zachodzi zawieranie $\rightarrow_\beta \subseteq \rightarrow_{r\beta}$.* □

Lemat 12.10 *Relacja równoległej β-redukcji $\rightarrow_{r\beta}$ jest częścią β-redukcji, a więc mamy zawieranie $\rightarrow_{r\beta} \subseteq \twoheadrightarrow_\beta$.*

Dowód. Relacja $\twoheadrightarrow_\beta$ ma własności wymagane od relacji $\rightarrow_{r\beta}$. Na przykład, ma własność 2), czyli

jeżeli $M \twoheadrightarrow_\beta M_1$ i $N \twoheadrightarrow_\beta N_1$, to $(\lambda x M)N \twoheadrightarrow_\beta M_1[x := N_1]$.

Dowód nie nastręcza większych trudności. Jeżeli $M \twoheadrightarrow_\beta M_1$, to ze zgodności $\twoheadrightarrow_\beta$ także mamy $\lambda x M \twoheadrightarrow_\beta \lambda x M_1$ oraz $(\lambda x M)N \twoheadrightarrow_\beta (\lambda x M_1)N$. Podobnie, $(\lambda x M_1)N \twoheadrightarrow_\beta (\lambda x M_1)N_1$. W końcu, $(\lambda x M_1)N_1 \rightarrow_\beta M_1'[x := N_1]$. Korzystając z przechodniości relacji $\twoheadrightarrow_\beta$ otrzymujemy, że $(\lambda x M)N \twoheadrightarrow_\beta M_1'[x := N_1]$.

Pozostałe, wymagane własności relacji $\twoheadrightarrow_\beta$ dowodzimy analogicznie. Relacja $\rightarrow_{r\beta}$, jako najmniejsza relacja o tych własnościach, spełnia zawieranie $\rightarrow_{r\beta} \subseteq \twoheadrightarrow_\beta$. □

Z udowodnionych lematów jako oczywisty wniosek otrzymujemy

Twierdzenie 12.11 *Relacje $\twoheadrightarrow_\beta$ oraz $\twoheadrightarrow_{r\beta}$ są identyczne.* □

O roli redukcji równoległej najlepiej świadczy następujące

Twierdzenie 12.12 *Relacja równoległej β-redukcji ma własność Churcha-Rossera.*

Dowód. Znajduje się w dodatku. □

Stąd już jako oczywisty wniosek dostajemy twierdzenie Churcha-Rossera 12.3.

13 Termy z kolorowymi redeksami

Będziemy teraz rozważać termy z wieloma rodzajami operatorów λ. Będziemy je nazywać kolorowymi. Kolorowe operatory będziemy oznaczać wspólnym symbolem $\underline{\lambda}$ pamiętając, że może być ich kilka rodzajów. Można by rzeczywiście zapisywać je używając kolorowych znaków. Oprócz kolorowych zawsze będzie też zwykły operator, niekolorowy, można by uważać go za bezbarwnego, albo wręcz przeciwnie, za czarnego. Będzie oznaczany symbolem λ. Kolorowych operatorów nie będzie można dowolnie używać, będziemy zakładać, że kolorowe mogą być tylko pierwsze operatory λ w redeksach.

Zbiór termów z kolorowymi redeksami $\underline{\Lambda}$ definiujemy jako najmniejszy zbior taki, że

1) zmienne należą do $\underline{\Lambda}$ (są termami z kolorowymi redeksami),

2) jeżeli $M, N \in \underline{\Lambda}$, to $MN \in \underline{\Lambda}$,

3) jeżeli x jest zmienną i $M \in \underline{\Lambda}$, to $\lambda x\, M \in \underline{\Lambda}$,

4) jeżeli x jest zmienną oraz $M, N \in \underline{\Lambda}$, to $(\underline{\lambda} x\, M)N \in \underline{\Lambda}$.

Pojęcia α-konwersji, podstawiania i podstawialności dla termów z kolorowymi redeksami definiujemy tak, jak dla zwykłych termów, nie zwracając uwagi na kolory λ. W formalnych definicjach takie podejście wymaga dopisania do własności zwykłych operatorów analogicznych własności kolorowych λ.

Lemat 13.1 *Jeżeli M i N są termami z kolorowymi redeksami, to term $M[x := N]$ też jest termem z kolorowymi redeksami. Jeżeli w termie z kolorowymi redeksami β-redukujemy dowolny redeks (kolorowy lub nie), to otrzymujemy term z kolorowymi redeksami.* □

Wielość operatorów λ pozwala ograniczać β-redukcję do części redeksów i w konsekwencji rozważać w zbiorze $\underline{\Lambda}$ wiele rodzajów redukcji. Możemy umówić się, że redukujemy wyłącznie redeksy określonego koloru, albo tylko dowolne redeksy kolorowe. Możemy też redukować dowolne redeksy nie zwracając uwagi ani na ich kolory, ani na ich brak. Nie wprowadzamy oznaczeń na poszczególne rodzaje β-redukcji. To, jaką redukcję stosujemy, zawsze powinno jakoś wynikać z kontekstu.

Bardzo naturalnym sposobem redukcji w zbiorze $\underline{\Lambda}$ jest redukcja ograniczona wyłącznie do kolorowych redeksów. Tak rozumianą relację β-redukcji w jednym kroku dla termów z kolorowymi redeksami definiujemy jako najmniejszą relację zgodną z operacjami λ-rachunku zawierającą pary

$$(\underline{\lambda} x\, M)N \to_\beta M[x := N]$$

(nie redukujemy w tym sensie redeksów z bezbarwnymi operatorami).

Kolorowanie redeksów wprowadza do λ-rachunku istotne ograniczenie. W zwykłym rachunku redukcja redeksu może zwiększać liczbę redeksów na dwa sposoby: poprzez kopiowanie i poprzez tworzenie nowych redeksów. Na przykład, redukując term $(\lambda x\, xxx)M$ do MMM, trzykrotnemu skopiowaniu ulegają redeksy występujące w termie M. Jednocześnie może powstać nowy redeks: wystarczy, aby term M był abstrakcją, powstaje wtedy redeks MM. Jeżeli $(\lambda x\, xxx)M \in \underline{\Lambda}$ i M jest abstrakcją, to M rozpoczyna się zwykłym operatorem λ, a redeks MM nie jest kolorowym redeksem. Jeżeli więc redukcję ograniczamy do kolorowych redeksów, to zwiększenie liczby kolorowych redeksów powodowane jest wyłącznie kopiowaniem i nie powstają nowe kolorowe redeksy. Nowy redeks co prawda może powstać, ale nie może być on kolorowy i nie można go zredukować stosując tylko redukcję kolorowych redeksów.

13.1 Kolorowanie a β-redukcja

Jest oczywiste, że jeżeli mamy term (z kolorowymi redeksami lub nie), i w nim redeks, który nie jest kolorowy, to możemy spowodować, aby stał się on kolorowy, i w wyniku tego kolorowania otrzymujemy term z kolorowymi redeksami, czyli ze zbioru $\underline{\Lambda}$.

Zdefiniujmy teraz operację odwrotną $|\ \ | : \underline{\Lambda} \to \Lambda$ przyjmując, że

1) $| x | = x$ dla zmiennej x,

2) $| MN | = | M || N |$,

3) $| \lambda x\, M | = \lambda x\, | M |$ oraz

4) $| (\underline{\lambda} x\, M)N | = (\lambda x\, | M |)| N |$.

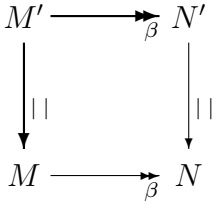
Term $| M |$ to term M, w którym operatory λ zostały pozbawione kolorów.

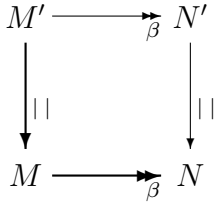
W dwóch kolejnych, raczej oczywistych lematach w zbiorze $\underline{\Lambda}$ rozważamy β-redukcję pozwalającą redukować dowolne redeksy, zarówno kolorowe, jak i niepokolorowane. Kolorowanie i pozbawianie kolorów komutuje z β-redukcją, a dokładniej

Lemat 13.2 *Przypuśćmy że mamy termy $M' \in \underline{\Lambda}$ i $M \in \Lambda$ takie, że $| M' | = M$. Jeżeli $M' \twoheadrightarrow_\beta N' \in \underline{\Lambda}$ i $| N' | = N$, to $M \twoheadrightarrow_\beta N$.* □

Lemat 13.3 *Przypuśćmy że mamy termy $M' \in \underline{\Lambda}$ i $M \in \Lambda$ takie, że $| M' | = M$. Jeżeli $M \twoheadrightarrow_\beta N$, to istnieje term $N' \in \underline{\Lambda}$ taki, że $| N' | = N$ oraz $M' \twoheadrightarrow_\beta N'$.* □

Treść tych lematów jest przedstawiona na poniższych diagramach.





13.2 Twierdzenie o normalizacji

Będziemy rozważać termy z kolorowymi redeksami i β-redukcję takich termów ograniczoną do kolorowych redeksów. Niekolorowych redeksów teraz nie będzie wolno redukować.

Dość łatwo zauważyć, że mając term z n kolorowymi redeksami można w nim w n krokach zredukować wszystkie kolorowe redeksy. Wystarczy redekować redeksy po kolei, od końca, od prawej do lewej strony termu. Jeżeli term uważamy za drzewo, to podobny efekt możemy uzyskać redukując redeksy w kolejności od liści do korzenia.

Zdefiniujmy teraz przez rekursję funkcję $\varphi : \underline{\Lambda} \to \Lambda$ przyjmując

1) $\varphi(x) = x$ dla dowolnej zmiennej x,

2) $\varphi(\lambda x\, M) = \lambda x\, \varphi(M)$,

3) $\varphi((\underline{\lambda} x\, M)N) = \varphi(M)[x := \varphi(N)]$ oraz

4) $\varphi(MN) = \varphi(M)\varphi(N)$ w przypadku aplikacji innych postaci.

Mam nadzieję, że poprawność definicji oraz rodzaj zdefiniowanej funkcji nie budzą wątpliwości. Najważniejszą własność tej funkcji wyraża

Lemat 13.4 *Każdy term z kolorowymi redeksami M można zredukować do $\varphi(M)$ stosując wyłącznie β-redukcję kolorowych redeksów (czyli β_0-redukcję).*

Dowód. Indukcyjny jest łatwy do przeprowadzenia. W przypadku kolorowego redeksu mamy

$$(\underline{\lambda} x\, M)N \twoheadrightarrow_{\beta_0} (\underline{\lambda} x\, \varphi(M))N \twoheadrightarrow_{\beta_0} (\underline{\lambda} x\, \varphi(M))\varphi(N) \to_{\beta_0} \varphi(M)[x := \varphi(N)] = \varphi((\underline{\lambda} x\, M)N).$$

Dwa pierwsze przekształcenia wynikają z założenia indukcyjnego $M \twoheadrightarrow_{\beta_0} \varphi(M)$ i $N \twoheadrightarrow_{\beta_0} \varphi(N)$. □

Wniosek 13.5 *Każdy term z kolorowymi redeksami ma postać normalną (w sensie β-redukcji kolorowych redeksów, czyli β_0-redukcji).*

Dowód. Postacią normalną termu M w rozważym sensie jest term $\varphi(M)$. Term ten należy do Λ, a więc nie zawiera kolorowych redeksów (nie można już go przekształcić). Ponadto, jak wynika z lematu, można go otrzymać w wyniku kolorowej β-redukcji termu M. □

Lemat 13.6 *Przypuśćmy, że M i N są termami z kolorowymi redeksami i $M \to_\beta N$. Jeżeli przekształcenie M w N polegało na redukcji kolorowego redeksu, to $\varphi(M)$ jest identyczne z $\varphi(N)$. W przeciwnym razie, $\varphi(M) \twoheadrightarrow_\beta \varphi(N)$.*

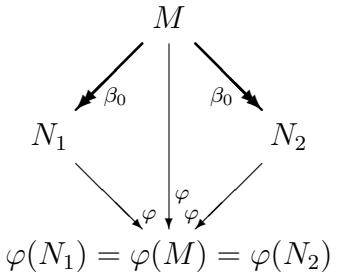
Treść powyższego lematu przedstawia następujący rysunek:

Wniosek 13.7 *Przypuśćmy, że M i N są termami z kolorowymi redeksami i $M \twoheadrightarrow_\beta N$. Jeżeli przekształcenie M w N polegało na redukcji tylko kolorowych redeksów, to $\varphi(M)$ jest identyczne z $\varphi(N)$. W przeciwnym razie, $\varphi(M) \twoheadrightarrow_\beta \varphi(N)$.*

Tak oto przedstawiamy teraz treść wniosku:

Wniosek 13.8 *β-redukcja kolorowych redeksów ma własność Churcha-Rossera.*

Dowód. Najprościej jest go przedstawić na rysunku:



Wniosek 13.9 *Każdy term z kolorowymi redeksami ma postać normalną w sensie β-redukcji kolorowych redeksów i to dokładnie jedną.* □

13.3 Twierdzenie o silnej normalizowalności

13.3.1 Zmienne o różnych wagach

W dowodzie twierdzenia o silnej normalizowalności przypiszemy zmiennym dodatkowy atrybut, wagę. Będzie to dodatnia liczba naturalna. Różne wystąpienia zmiennej zmiennej będą mogły mieć różne wagi. Wag nie przypisujemy zmiennym wiązanym, występującym bezpośrednio za operatorem λ.

Wagi zmiennych nie wpływają na podstawianie i redukcję. Podstawienie za zmienną powoduje usunięcie zmiennej razem z jej wagą. Z kolei zmienna podstawiana zachowuje swoją wagę. W jakimś stopniu waga będzie pamiętać, gdzie zmienna występowała przed rozpoczęciem redukcji.

Wagą termu M będzie suma wag występujących w nim zmiennych. Będzie ona oznaczana symbolem $\mid M \mid$.

13.3.2 Termy z malejącymi wagami

Zmienne z wagami wykorzystamy do analizy procesu redukcji termów z kolorowymi redeksami. Będziemy rozważać taką redukcję, która dopuszcza wyłącznie redukowanie kolorowych redeksów.

Term M z kolorowymi redeksami (i zmiennymi z wagami) nazywamy termem z malejącymi wagami zmiennych, jeżeli dla każdego kolorowego redeksu $(\lambda x\, P)Q$ w tym termie, wagi wszystkich wystąpień zmiennej x w termie P przekraczają wagę termu Q.

Lemat 13.10 *W dowolnym termie z kolorowymi redeksami zmiennym można nadać wagi tak, aby stał się on termem z malejącymi wagami zmiennych.*

Dowód. Wystarczy kolejnym zmiennym od tyłu (od prawej do lewej strony termu) nadać wagi będące kolejnymi potęgami liczby 2. □

13.3.3 β-redukcja i wagi termów

Przypuśćmy, że mamy term M z kolorowym redeksem $(\lambda x\, P)Q$ i z malejącymi wagami zmiennych, i redukujemy go do termu M', a dokładniej

$$M = C[(\lambda x\, P)Q] \;\rightarrow_\beta\; C[P[x := Q]] = M'.$$

Wtedy, po pierwsze, mamy

Lemat 13.11 *Waga termu M' jest mniejsza niż waga termu M.*

Dowód. Musimy rozważyć dwa przypadki. Jeżeli zmienna x nie jest wolna w termie P, to redukcja powoduje po prostu zastąpienie redeksu termem P, a to oznacza zmniejszenie wagi M o (dodatnią) wagę termu Q.

Przyjmijmy więc, że zmienna x występuje jako wolna w termie P. Ponieważ term M ma malejące wagi zmiennych, a $(\lambda x\, P)Q$ jest kolorowym redeksem, więc wagi wszystkich wolnych wystąpień zmiennej x w termie P przekraczają wagę termu Q. W konsekwencji zastąpienie w termie P zmiennej x przez Q powoduje zmniejszenie wagi termu M. □

W wyżej opisanej sytuacji mamy też następujący

Lemat 13.12 *Jeżeli M jest termem z malejącymi wagami, to term M' też jest takim termem.*

Dowód. Aby się o tym przekonać, musimy zbadać wszystkie kolorowe redeksy w termie M'. Weźmy więc dowolny kolorowy redeks $N = (\lambda y\, K)L$ zawarty w M' (może teraz warto spojrzeć na rysunek ze strony 42). W M' są dwa rodzaje redeksów: takie, które powstały w wyniku jakiegoś przekształcenia redeksu z termu M i takie, które pojawiły się po wykonywanej redukcji. Redeksy drugiego rodzaju mogą powstać na styku kontekstu C i redukowanego redeksu, oraz wyniku zastąpienia x termem Q. Analizę redeksów zaczynamy od tych drugiego rodzaju.

Przypadek 1: Term P jest abstrakcją. Taką, która w wyniku redukcji staje się pierwszym członem redeksu. Taki redeks nie może być kolorowy, gdyż drugi operator λ w termach postaci $\lambda x\, \lambda y\, K$ nie może być pokolorowany.

Przypadek 2: Term Q jest abstrakcją. I to taką, tóra w wyniku redukcji staje się pierwszym członem redeksu. Otrzymany tak redeks nie może być kolorowy, gdyż operator λ rozpoczynający drugi człon aplikacji nie może być pokolorowany.

Redeksy tworzone podczas redukcji nie mogą więc być kolorowe. Zajmijmy się teraz redeksami przekształcanymi.

Przypadek 3: Redeks $(\lambda y\, K)L$ jest rozłączny z podtermem $P[x := Q]$. Taki redeks jest też podtermem M (jest podtermem kontekstu C) i nie ulega zmiania podczas redukcji. Nie zmieniają się w nim także wagi.

Przypadek 4: Redeks $(\lambda y\, K)L$ obejmuje $P[x := Q]$.

Przypadek 4.1: Term K obejmuje $P[x := Q]$. Redeks $(\lambda y\, K)L$ powstał w wyniku redukcji pewnego redeksu postaci $(\lambda y\, K')L$ zawartego w termie M (redukowany był właściwie tylko term K'). Warunek z definicji termu z malejącymi wagami zmiennych zachodzi, gdyż zmienne y z termu K występują z tymi samymi wagami w termie K', i wagi te są większe niż waga termu L.

Przypadek 4.2: Term L obejmuje $P[x := Q]$. Redeks $(\lambda y\, K)L$ powstał w wyniku redukcji redeksu $(\lambda y\, K)L'$ w termie M (redukowany był właściwie tylko term L'). Wymagane nierówności zachodzą, gdyż wagi zmiennych y z termu K nie uległy zmianie, a waga termu L jest mniejsza od wagi L' na mocy poprzedniego lematu.

Przypadek 5: Redeks $(\lambda y\, K)L$ jest częścią $P[x := Q]$.

Przypadek 5.1: Redeks $(\lambda y\, K)L$ jest częścią P. Redeks $(\lambda y\, K)L$ postał z pewnego redeksu $(\lambda y\, K')L'$ zawartego w M w wyniku podstawiania za x termu Q. Tak więc $K = K'[x := Q]$ oraz $L = L'[x := Q]$. Wszystkie wystąpienia zmiennej y w K występują także z takimi samymi wagami w K' (y nie występuje w Q, gdyż wymagamy podstawialności Q). Ponieważ M jest termem z malejącymi wagami, więc wagi zmiennych y z K' przekraczają wagę termu L', a ta z kolei jest większa od wagi $L'[x := Q]$ (jeżeli x jest wolna w L') lub jest jej równa (w przeciwnym razie).

Przypadek 5.2: Redeks $(\lambda y\, K)L$ jest częścią pewnej kopii Q. Taki redeks jest także zawarty w M, tam spełnia wymane nierówności. Redukcja nie zmienia go i zachowuje własności związane z wagami. □

13.3.4 Dowód twierdzenia

Twierdzenie 13.13 (o silnej normalizacji dla redukcji kolorowych redeksów) *Jeżeli M jest termem z kolorowymi redeksami i redukujemy wyłącznie kolorowe redeksy (stosujemy β_0-redukcję), to każdy sposób redukcji termu M jest skończony.*

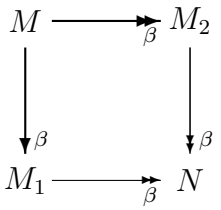
Dowód. Niech M będzie termem z kolorowymi redeksami. Korzystając z lematu 13.10 zmiennym tego termu przypisujemy wagi wagi tak, aby stał się on termem z malejącymi wagami. Weźmy dowolną redukcję termu M, a więc ciąg termów $M = M_0, M_1, M_2, \ldots$ taki, że $M_0 \to_{\beta_0} M_1 \to_{\beta_0} M_2 \to_{\beta_0} \ldots$. Zgodnie z lematem 13.12, wszystkie termy w tym ciągu są termami z malejącymi wagami. Z kolei z lematu 13.11 otrzymujemy, że wagi termów z tego ciągu tworzą malejący ciąg liczb naturalnych Taki ciąg jest skończony. □

13.4 Dowód twierdzenia Churcha-Rossera

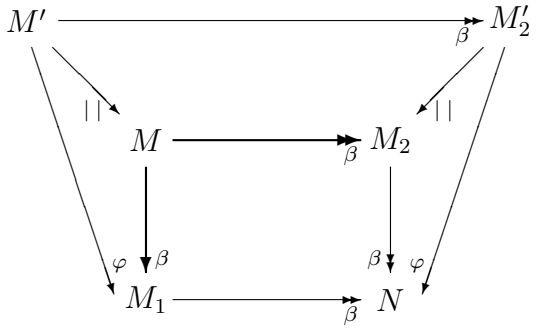
Udowodnimy teraz twierdzenie 12.3 Churcha-Rossera, czyli

Twierdzenie 13.14 *Relacja β-redukcji $\twoheadrightarrow_\beta$ ma własność Churcha-Rossera.*

Dowód. Skorzystamy z lematu 12.2. Wystarczy więc wykazać własność β-redukcji przedstawioną na diagramie



Dowód tej własności został zobrazowany na kolejnym diagramie. Mamy dane termy M, M_1 i M_2, a także redukcje M do M_1 (w jednym kroku) i M do M_2. W termie M kolorujemy redeks pozwalający zredukować M do M_1. W ten sposób otrzymujemy term M'. Usuwając w nim kolor redeksu otrzymujemy więc M, redukując w nim kolorowy redeks (na przykład obliczając φ) – otrzymujemy M_1.



Teraz korzystamy z lematu 13.3 i redukcję M do M_2 przekształcamy w redukcję termów z kolorowymi redeksami. W ten sposób znajdujemy term M_2' i redukcję M' do M_2'. W termie M_2' mogą znajdować się kolorowe redeksy. Redukujemy je wszystkie na przykład wyliczając wartość przekształcenia φ. W ten sposób obliczamy term N, który nie ma kolorowych redeksów. W ten sam sposób możemy przekształcać term M_2. Zgodnie z lematem 13.2, uwzględniając równość $|N| = N$, znowu otrzymujemy term N. W końcu, na podstawie wniosku 13.7 stwierdzamy, że term N można otrzymać także redukując M_1. To kończy dowód. □

14 Redukcje, czyli sposoby redukowania termów

Będziemy teraz szukali odpowiedzi na pytanie, jak redukować term, aby coś uzyskać, na przykład, jak znaleźć postać normalną termu.

14.1 Prefiks i sufiks podtermu

Potrzebny jest nam aparat, który dokładniej pozwoli nam identyfikować podtermy.

Przypuśćmy, że mamy term M i jego podterm N. Jest chyba oczywiste, że jeżeli termy uważamy za napisy, czyli słowa, to podterm jest podsłowem, czyli napisem, który powstaje przez usunięcie z termu pewnego prefiksu i opuszczenie pewnego sufiksu. Niech $pref_M(N)$ i $suff_M(N)$ oznaczają odpowiednio prefiks i sufiks termu M, leżące przed i za podtermem N. Mamy więc oczywisty związek

$$M = pref_M(N)\, N\, suff_M(N),$$

w nim po prawej stronie równości mamy konkatenację trzech napisów. Najbardziej istotną cechą prefiksu i sufiksu podtermu jest liczba występujących w nich

zmiennych i operatorów abstrakcji. Moglibyśmy więc dodatkowo z prefiksu i sufiksu usuwać wszystkie lub niektóre nawiasy.

Bardziej formalna definicja może być następująca:

1) $pref_M(M)$ i $suff_M(M)$ są słowami pustymi,

2) jeżeli $M = M_1M_2$ i N jest też podtermem M_1, to $pref_M(N) = pref_{M_1}(N)$ i $suff_M(N) = suff_{M_1}(N)M_2$,

3) jeżeli $M = M_1M_2$ i N jest też podtermem M_2, to $pref_M(N) = M_1 pref_{M_2}(N)$ i $suff_M(N) = suff_{M_1}(N)$,

4) jeżeli $M = \lambda x\, M_1$ i N jest też podtermem M_1, to $pref_M(N) = \lambda x\, pref_{M_1}(N)$ i $suff_M(N) = suff_{M_1}(N)$.

Dwa podtermy mogą mieć identyczne prefiksy. Tak jest w przypadku podtermów postaci N_1 i N_1N_2. Za to różne abstrakcje mają różne prefiksy, różniące się przynajmniej operatorem λ i zmienną wiązaną „pierwszej" abstrakcji. Podobnie jest dla redeksów: redeksy są wyznaczone przez rozpoczynające je abstrakcje i mają takie same prefiksy, jak te abstrakcje.

14.2 Kolejność podtermów

Przypuśćmy, że N_1 i N_2 są podtermami termu M. Mówimy, że podterm N_1 poprzedza term N_2 (leży na lewo od termu N_2) w dwóch przypadkach:

1) gdy $pref_M(N_1)$ jest krótszy niż $pref_M(N_2)$ oraz

2) gdy $pref_M(N_1) = pref_M(N_2)$ i $suff_M(N_1)$ jest krótszy niż $suff_M(N_2)$.

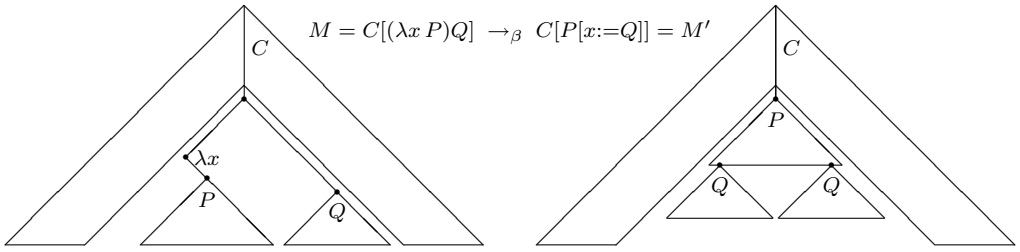
Kolejność podtermów jest oczywista, gdy są one rozłączne. Jeżeli mamy dwa podtermy N_1 i N_2, i N_2 jest też podtermem N_1, to z tych dwóch bardziej na lewo leży (lub jest wcześniejszy) podterm N_1.

14.3 Analiza pojedyńczego kroku β-redukcji

Przyjrzymy się teraz co dzieje się z podtermami podczas redukcji termu.

Przypuśćmy, że mamy term M, który chcemy zredukować. Wtedy w pierwszym rzędzie wybieramy w nim redeks i przedstawiamy go w postaci $M = C[(\lambda x\, P)Q]]$ dla pewnego kontekstu C. Term M, rozumiany jako drzewo, został przedstawiony na widocznym rysunku. Tam też jest narysowany term M' otrzymany w wyniku redukcji wybranego redeksu.

Nietrudno zauważyć, że podtermy termu M' nie biorą się znikąd, są odpowiednio rozmieszczonymi podtermami M. Będziemy mówić, że mają swojego przodka w M. Przodkiem podtermu dowolnej kopii termu Q, zawartej w M', jest odpowiadający mu podterm termu Q zawartego w M. Analogicznie definiujemy przodków pozostałych podtermów termu P i kontekstu C. Przodka podtermu N z termu M' będziemy oznaczać symbolem $p(N)$, albo nawet symbolem $p_{M'M}(N)$. Te same oznaczenia zastosujemy, jeżeli będziemy rozważać co się dzieje z podtermami podczas redukcji wymagającej wielu kroków.



Szczególnie będą nas interesowały podtermy leżące w M na lewo od redeksu i w M' na lewo od reduktu M' (a nawet leżące istotnie na lewo, mające krótsze prefiksy). Są dwa rodzaje takich podtermów: część znajduje się na ścieżce łączącej redeks z korzeniem drzewa M, pozostałe – w lewej części kontekstu. Istotne jest to, że dla tych podtermów wszystko zależy od kontekstu C, który nie ulega zmianie podczas redukcji.

Lemat 14.1 *Przypuśćmy, że $M \xrightarrow{R}_\beta M'$ i $R = (\lambda x\, P)Q$. Wtedy prefiksy redeksu i reduktu są identyczne, czyli $pref_M((\lambda x\, P)Q) = pref_{M'}(P[x := Q])$.* □

Lemat 14.2 *Przypuśćmy, że $M \xrightarrow{R}_\beta M'$ i $R = (\lambda x\, P)Q$. Operacja p bycia przodkiem przekształca wzajemnie jednoznacznie zbiór (podtermów termu M')*

$$\{N : pref_{M'}(N) < pref_{M'}(P[x := Q])\} \;\; na \;\; \{N : pref_M(N) < pref_M((\lambda x\, P)Q)\}$$

(czyli na pewien zbiór podtermów termu M). Operacja ta zachowuje typ termu (bycie aplikacją lub abstrakcją), a także fakt bycia redeksem, oraz prefiks. Tak więc mamy $pref_M(p(N)) = pref_{M'}(N)$ dla wszystkich podtermów N takich, że $pref_{M'}(N) < pref_{M'}(P[x := Q])$. □

Przykład 14.3 Przyjrzyjmy się termowi $M = II((\lambda x\, (\lambda y\, yy)x)I)$. W tym termie są trzy redeksy II, $(\lambda x\, (\lambda y\, yy)x)I$ i $(\lambda y\, yy)x$. Zostały wymienione w naturalnym porządku, od najbardziej lewego. Ich prefiksami z dokładnością do nawiasów są odpowiednio słowo puste, II, $II((\lambda x.$ Przypuśćmy, że chcemy przekształcić ostatni z redeksów. Znajduje się on w kontekście $C = II((\lambda x\, [\ ])I)$. W wyniku redukcji otrzymujemy $M' = II((\lambda x\, xx)I)$, prefiksem reduktu xx jest $II((\lambda x.$ W termie M' mamy też dwa redeksy o prefiksach krótszych: II z pustym prefiksem i $(\lambda x\, xx)I$ z prefiksem II. Przodkami tych termów są $p(II) = II$ oraz $p((\lambda x\, xx)I) = (\lambda x\, (\lambda y\, yy)x)I$. Jak widać, oba redeksy w termie M, poprzedzające redeks redukowany, zostały zachowane, a drugi z nich został jakoś przekształcony.

14.4 Rodzaje redukcji

Przypuśćmy, że dostajemy term M i zaczynamy go redukować (używając β-redukcji). Teraz przez redukcję będziemy rozumieć ciąg termów

$$M = M_0 \to_\beta M_1 \to_\beta M_2 \to_\beta \ldots \to_\beta M_n \to_\beta \ldots$$

skończony lub nie, otrzymamy w wyniku wielokrotnego redukowania. Zapisując taką redukcję będziemy często nad symbolem $\to_\beta$ podawać redukowany redeks. Wtedy jest przedstawienie może przyjąć taką oto postać

$$M = M_0 \xrightarrow{R_0}_\beta M_1 \xrightarrow{R_1}_\beta M_2 \xrightarrow{R_2}_\beta \ldots \xrightarrow{R_{n-1}}_\beta M_n \xrightarrow{R_n}_\beta \ldots \qquad (5)$$

Redukcję (5) (skończoną lub nie) nazywamy lewostronną, jeżeli w każdym momencie redukujemy najbardziej lewy (pierwszy) redeks (dla wszystkich możliwych i redeks R_i jest pierwszym redeksem w termie M_i).

Redukcję (5) nazywamy quasi-lewostronną, jeżeli jest nieskończona i podczas jej tworzenia nieskończenie wiele razy był redukowany pierwszy redeks.

Redukcję (5) (skończoną lub nie) nazywamy standardową, jeżeli podczas jej tworzenia prefiksy redukowanych redeksów nie ulegają skróceniu, a więc dla wszystkich możliwych i mamy $pref_{M_i}(R_i) \leqslant pref_{M_{i+1}}(R_{i+1})$.

14.5 Pierwsze wnioski

Wniosek 14.4 *Przypuśćmy, że* $M \xrightarrow{R}_\beta M'$ *i redeks* R_0 *poprzedza w termie* M *redeks* R*, czyli* $pref_M(R_0) < pref_M(R)$*. Wtedy w termie* M' *istnieje potomek* R'_0 *redeksu* R_0*, czyli redeks* R'_0*, taki że* $p(R'_0) = R_0$*.*

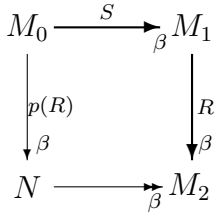
Dowód. Jest to oczywisty wniosek z lematu 14.2. □

Wniosek 14.5 *Przypuśćmy, że redukcja* $M_0 \xrightarrow{R_0}_\beta M_1 \xrightarrow{R_1}_\beta M_2 \xrightarrow{R_2}_\beta \ldots \xrightarrow{R_{n-1}}_\beta M_n$ *jest standardowa i* R_0 *nie jest pierwszym redeksem w termie* M_0*. Wtedy term* M_n *nie jest w postaci normalnej, zawiera pewien redeks, na przykład potomka pierwszego redeksu w termie* M_0*.* □

Wniosek 14.6 *Redukcja standardowa zakończona termem w postaci normalnej jest lewostronna.* □

Lemat 14.7 *Przypuśćmy, że* $M_0 \xrightarrow{S}_\beta M_1 \xrightarrow{R}_\beta M_2$ *i nie jest to redukcja standardowa (a więc* $pref_{M_0}(S) > pref_{M_1}(R)$*). Wtedy istnieje standardowa redukcja* $M_0 \xrightarrow{R'}_\beta N \twoheadrightarrow_\beta M_2$*, gdzie* $R' = p_{M_1 M_0}(R)$*.*

Dowód. Niżej mamy narysowaną treść lematu, bez kilku istotnych szczegółów



Zgodnie z lematem 14.2, redeks R z termu M_1 ma przodka $p(R) = R'$ w termie M_0 o takim samym prefiksie, a więc zachodzi $pref_{M_0}(R') < pref_{M_0}(S)$. Pomalujmy teraz redeks R' (a właściwie operator λ tego redeksu) na czerwono i redeks S na zielono. W ten sposób term M_0 stał się termem z kolorowymi redeksami.

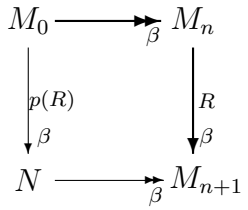
Jeżeli w termie M_0 zredukujemy redeks S, to otrzymujemy term M_1, w którym już nie ma termów zielonych. Jest w nim natomiast czerwony redeks R, jedyny kolorowy. Redukcja tego redeksu daje term M_2, bez kolorowych redeksów. Tak więc term M_2 jest postacią normalną termu M_0 w sensie redukcji kolorowych redeksów.

Jeżeli teraz w termie M_0 zredukujemy redeks R', to zniknie z M_0 redeks czerwony, redeks zielony najprawdopodobniej zostanie skopiowany (może też zniknąć), zgodnie z lematem 14.2 nie pojawi się redeks zielony z prefiksem krótszym niż

$pref_{M_0}(R')$. Przyjmijmy, że w ten sposób otrzymamy term N. Nie ma w nim redeksu czerwonego, a wszystkie redeksy zielone będą miały prefiksy przynajmniej tak długie, jak $pref_{M_0}(R')$, i będą mogły zostać zredukowane podczas tworzenia redukcji standardowej termu M_0. Dalej przekształcamy term N redukując w nim po kolei, od najbardziej lewego, wszystkie redeksy zielone, także te, które być może pojawią się podczas redukcji. Z twierdzenia o silnej normalizowalności dla redukcji kolorowych redeksów ten proces redukowania zakończy się postacią normalną termu M_0. Zgodnie z wnioskiem 13.9 otrzymamy wtedy term M_2. □

Wniosek 14.8 *Załóżmy, że wykonując n krokach redukcję w $M_0 \to_\beta M_n$ w żadnym nie zredukowaliśmy pierwszego redeksu, R jest pierwszym redeksem termu M_n i $M_n \xrightarrow{R}_\beta M_{n+1}$. Wtedy dla $R' = p_{M_n M_0}(R)$ istnieje redukcja $M_0 \xrightarrow{R'}_\beta N \twoheadrightarrow_\beta M_{n+1}$.*

Dowód. Przez indukcję ze względu na n, korzystając z poprzedniego lematu. Zgodnie w wnioskiem, jeżeli redukujemy term M_0 i w pierwszym kroku nie zredukujemy pierwszego redeksu, to redeks ten przechodzi do kolejnego termu, jest w nim pierwszym redeksem i w końcu po pewnym czasie może zostać zredukowany. Do otrzymanego w ten sposób termu dojdziemy też redukując najpierw pierwszy redeks termu M_0 i dalej wykonując odpowiednie przekształcenia. Niżej znowu mamy treść lematu przedstawioną na rysunku, ale bez istotnych szczegółów. □



Wniosek 14.9 *Jeżeli term ma redukcję quasi-lewostronną, to ma też nieskończoną redukcję lewostronną.*

Dowód. Wystarczy wykazać, że jeżeli term ma redukcję quasi-lewostronną, to dla każdej liczby naturalnej n ma też redukcję quasi-lewostronną, która rozpoczyna się redukcją lewostronną długości n. Oczywiście, dowodzimy to przez indukcję, korzystając z poprzedniego wniosku. □

14.6 Twierdzenie o standaryzacji z wnioskami

Twierdzenie 14.10 (Twierdzenie o standaryzacji) *Jeżeli $M \twoheadrightarrow_\beta N$, to istnieje standardowa redukcja M do N.*

Wniosek 14.11 *Jeżeli term M ma postać normalną, to można go zredukować do postaci normalnej za pomocą redukcji lewostronnej.*

Dowód. Niech N będzie postacią normalną termu M. Z definicji postaci normalnej wynika, że $M \twoheadrightarrow_\beta N$. Z twierdzenia 14.10 o standaryzacji term M można standardowo zredukować do N. Wniosek 14.6 stwierdza, że jest to także redukcja lewostronna. □

Wniosek 14.12 *Jeżeli term ma nieskończoną redukcję lewostronną, to nie ma postaci normalnej.*

Dowód. W przeciwnym razie musiałyby istnieć dwie różne redukcje lewostronne. Jest to niemożliwe, ponieważ w redukcji lewostronnej każdy krok jest jednoznacznie wyznaczony przez redukowany term i krótsza redukcja lewostronna jest fragmentem dłuższej. □

Wniosek 14.13 *Jeżeli term ma redukcję quasi-lewostronną, to nie ma postaci normalnej.*

Dowód. Jest to konsekwencja wniosku 14.9. □

Wniosek 14.14 *Zbiór termów, które mają postać normalną, jest rekurencyjnie przeliczalny (przynajmniej w sensie intuicyjnym).*

Dowód. Semirozstrzygający algorytm badający istnienie postaci normalnej wykonuje (dopóki jest to możliwe) redukcję najbardziej lewego redeksu danego termu, kończy działanie wtedy i tylko wtedy, gdy dany term ma postać normalną. □

14.6.1 Główny lemat

Lemat 14.15 *Przypuśćmy, że $A_0 \to_\beta B_0 \twoheadrightarrow_\beta B$ i redukcja B_0 do B jest standardowa. Wtedy A_0 też można standardowo zredukować do B.*

Dowód. Będziemy tworzyć i stopniowo rozbudowywać następujący diagram

$$
\begin{array}{ccccccccccc}
A_0 & & & & \xrightarrow{S}_\beta & & & & & & B_0 \\
{\scriptstyle R_0}\downarrow_\beta & & & & & & & & & & {\scriptstyle Q_0}\downarrow_\beta \\
A_1 & \to_\beta & & & \dots & & & & & \to_\beta & B_1 \\
{\scriptstyle R_1}\downarrow_\beta & & & & & & & & & & {\scriptstyle Q_1}\downarrow_\beta \\
\vdots & & & & & & & & & & \vdots \\
A_{i-1} & \to_\beta & C'_1 & \to_\beta & C'_2 & \to_\beta & \dots & \to_\beta & C'_n & \to_\beta & B_{i-1} \\
{\scriptstyle R_{i-1}}\downarrow_\beta & & & & & & & & & & {\scriptstyle Q_{i-1}}\downarrow_\beta \\
A_i & \xrightarrow{Z_0}_\beta & C_1 & \xrightarrow{Z_1}_\beta & C_2 & \xrightarrow{Z_2}_\beta & \dots & \xrightarrow{Z_{m-1}}_\beta & C_m & \xrightarrow{Z_m}_\beta & B_i \\
 & & & & {\scriptstyle R_i}\downarrow_\beta & & & & & & {\scriptstyle Q_i}\downarrow_\beta \\
 & & & & A_{i+1} & \twoheadrightarrow_\beta & & & & \twoheadrightarrow_\beta & B_{i+1} \\
 & & & & & & & & & & {\scriptstyle Q_{i+1}}\downarrow_\beta
\end{array}
$$

Na tym diagramie obok symboli redukcji są podane nazwy przekształcanych redeksów. Jest przedstawiony fragment danej redukcji standardowej przekształcającej $B_0 \to_\beta B_1 \to_\beta$ aż do B i redukcja A_0 do B_0.

Będziemy mieć trochę kłopotów z termami A_j. Każdy z tych termów zostanie zdefiniowany, a następnie przyjęta definicja zostanie zmieniona przed określeniem kolejnego. Będziemy więc mówić o pierwotnym termie A_j i ostatecznym termie A_j. Ostateczny term A_j otrzymamy w wyniku redukcji pierwotnego termu A_j. Relacja $A_{j-1} \overset{R_{j-1}}{\to}_\beta A_j$ będzie zachodzić zwykle dla ostatecznej wersji A_{j-1} i pierwotnej wersji A_j. Na diagramie widać ostateczne wersje termów $A_0, \dots, A_{i-1}$ i pierwotne wersje termów A_i i A_{i+1}. Za ostateczną wersję termu A_i zostanie uznany C_2.

Konstrukcja termu A_1 ma sens, jeżeli redukcja A_0 do B nie jest standardowa. W tym przypadku postępujemy tak, jak w dowodzie lematu 14.7. Znajdujemy przodka

$p_{B_0A_0}(Q_0)$ redeksu Q_0, oznaczamy go symbolem R_0 i kolorujemy na czerwono. Redeks S kolorujemy na zielono. Pierwotną wersję A_1 otrzymujemy redukując w A_0 jedyny czerwony redeks R_0. Redukując po kolei zielone redeksy termu A_1 otrzymujemy B_1.

Konstrukcja reszty diagramu będzie prowadzona tak, aby zachowywać kilka niezmienników:

1) A_i jest termem z zielonymi redeksami,

2) redukcja $A_0 \to_\beta A_1 \to_\beta \ldots \to_\beta A_i$ i dalej $A_i \to_\beta C_1 \twoheadrightarrow_\beta B_i$ aż do B_i jest standardowa,

3) jej końcowy fragment od A_i do B_i jest redukcją normalną dla zielonych redeksów, (a więc prowadzącą do termu bez kolorowych redeksów),

4) $R_{i-1} = p_{B_{i-1}A_{i-1}}(Q_{i-1})$ i w szczególności $pref_{A_{i-1}}(R_{i-1}) = pref_{B_{i-1}}(Q_{i-1})$.

Oczywiście, pierwszy krok konstrukcji spełnia wymienione niezmienniki.

Załóżmy, że fragment konstrukcji zawierający definicję redukcji od $A_i = C_0$ do B_i jest już wykonany.

Przypadek 1: ostatnim wyrazem redukcji $A_0 \twoheadrightarrow_\beta A_i \twoheadrightarrow_\beta B_i$ jest ostatni wyraz redukcji $B_0 \twoheadrightarrow_\beta B$, czyli $B_i = B$. W tym przypadku teza lematu jest oczywista i wynika z niezmiennika 2).

Przypadek 2: $B_i \neq B$, czyli nie dotarliśmy jeszcze do końca redukcji. Teraz sprawdzamy, czy redukcja $C_m \to_\beta B_i \to_\beta B_{i+1}$ jest standardowa. Porównujemy więc prefiksy $pref_{C_m}(Z_m)$ oraz $pref_{B_i}(Q_i)$.

Przypadek 2.1: $pref_{C_m}(Z_m) \leqslant pref_{B_i}(Q_i)$. W tym przypadku redukcja od A_0 do A_i, dalej od A_i do B_i i jeszcze od B_i do B jest standardowa. Lemat został więc dowiedziony.

Przypadek 2.2: $pref_{C_m}(Z_m) > pref_{B_i}(Q_i)$. W tym przypadku redeks Q_i został pominięty podczas redukcji $A_0 \twoheadrightarrow_\beta A_i \twoheadrightarrow_\beta B_i$, mimo że powinien zostać zredukowany (jest przecież redukowany w termie B_i), i trzeba go zredukować wcześniej. Staramy się znaleźć term, a w nim przodka Q_i, którego możnaby zredukować zachowując standardowość redukcji. Bierzemy więc najmniejszą liczbę $k \geqslant 0$ taką, że $pref_{C_k}(Z_k) > pref_{B_i}(Q_i)$ (w razie potrzeby przyjmujemy, że $C_0 = A_i$). Dla tak zdefiniowanego k, posługując się lematem 14.2, możemy znaleźć w termie C_k przodka $p_{B_iC_k}(Q_i)$ redeksu Q_i. Oznaczmy tego przodka symbolem $R_i = p_{B_iC_k}(Q_i)$. Mamy więc $pref_{C_k}(R_i) = pref_{B_i}(Q_i)$.

W termie C_k redeks Z_k jest zielonym redeksem położonym spośród zielonych najbardziej na lewo i sam ma po lewej stronie redeks R_i. Tak więc redeks R_i ma po prawej stronie wszystkie zielone redeksy w termie C_k. Można więc zredukować R_i, a następnie jakikolwiek zielony redeks zachowując standardowość redukcji.

Na chwilę pomalujmy na czerwono redeks R_i. Powoduje to, że wszystkie termy z redukcji od C_k do B_i zawierają po jednym czerwonym redeksie, przodku termu Q_1. W termie B_i czerwonym redeksem jest Q_i. Redukując go otrzymujemy B_{i+1}, bez zielonych oraz bez czerwonych redeksów. Term B_{i+1} jest więc postacią normalną termu C_k dla redukcji kolorowych redeksów. W kolorowej redukcji do

postaci normalnej dochodzimy zawsze, redukując redeksy w jakiejkolwiek kolejności (patrz rozdz. 13.3). Możemy więc najpierw w termie C_k zredukować redeks czerwony (otrzymujemy wtedy A_{i+1}), a następnie redukować redeksy zielone zawsze biorąc pierwszy redeks od lewej. Redukując w ten sposób też otrzymamy postać normalną, czyli term B_{i+1}. W szczególności, istnieje standardowa redukcja zaczynająca się termem C_k do B_{i+1}.

Znowu są możliwe dwa przypadki:

Przypadek 2.2.1: $k > 0$. W tym przypadku redukcja prowadząca od A_0 do A_i, dalej od A_i (oznaczanego też C_0) do C_k i na koniec do A_{i+1} otrzymanego z termu C_k przez redukcję redeksu R_i jest standardowa. Wynika to z definicji k, która gwarantuje nierówność $pref_{C_{k-1}}(Z_{k-1}) \leqslant pref_{B_i}(Q_i) = pref_{C_k}(R_i)$.

Dodając do redukcji od A_0 do C_k standardową redukcję prowadzącą od C_k do B_{i+1} otrzymujemy standardową redukcję od A_0 do B_{i+1}.

Przypadek 2.2.2: $k = 0$. Teraz mamy dwie redukcje od A_0 do A_i (czyli do C_0) oraz od $A_i = C_0$ do A_{i+1} i dalej do B_{i+1}, obie standardowe, ale nie jest jasne, czy łącząc je otrzymamy redukcję standardową (być może szukając odpowiedniego przodka redeksu Q_i nie cofnęliśmy się dostatecznie daleko, a cofanie arbitralnie ograniczyliśmy do termu C_0).

Powinniśmy więc porównać prefiksy $pref_{A_{i-1}}(R_{i-1})$ oraz $pref_{A_i}(R_i)$. Zauważmy, że

$$pref_{A_{i-1}}(R_{i-1}) = pref_{B_{i-1}}(Q_{i-1})$$

na mocy zasady 4), zachowywanej podczas przeprowadzanej konstrukcji. Z drugiej strony, redeks R_i jest przodkiem Q_i i stąd

$$pref_{A_i}(R_i) = pref_{C_0}(p_{B_iC_0}(Q_i)) = pref_{B_i}(Q_i).$$

Teraz wystarczy zauważyć, że standardowość redukcji od B_0 do B także implikuje, że

$$pref_{B_{i-1}}(Q_{i-1}) \leqslant pref_{B_i}(Q_i),$$

a to oznacza, że

$$pref_{A_{i-1}}(R_{i-1}) \leqslant pref_{A_i}(R_i).$$

Skonstruowaliśmy więc standartową redukcję od A_0 do A_i, A_{i+1} i dalej do B_{i+1}. Powinniśmy jeszcze wskazać ostateczny term A_i: będzie nim term C_k. A także sprawdzić, czy przedstawiona konstrukcja zachowuje przyjęte niezmienniki. To ostatnie nie powinno sprawiać kłopotu. □

14.6.2 Dowód twierdzenia

Twierdzenie 14.16 (Twierdzenie o standaryzacji 14.10) *Jeżeli $M \twoheadrightarrow_\beta N$, to istnieje standardowa redukcja M do N.*

Dowód. Weźmy redukcję

$$M = M_0 \to_\beta M_1 \to_\beta \ldots \to_\beta M_{n-1} \to_\beta M_n = N.$$

Oczywiście, każdy term w tej redukcji też można zredukować do N w pewnej liczbie kroków, term M_i potrafimy zredukować do N w $n-1$ krokach.

Przez indukcję ze względu na liczbę kroków potrzebnych do zredukowania do N, pokazujemy, że każdy term M_i można standardowo zredukować do N. Co więcej nie ma czego dowodzić, ponieważ krok indukcyjny jest oczywistą konsekwencją lematu 14.15. □

15 Różności

15.1 Formalizacja częściowej obliczalności

Mówiąc o obliczalności często ograniczamy się do funkcji naturalnych. Tak będzie i tym razem. Do tej pory zajmowaliśmy się całkowitymi funkcjami obliczalnymi i ustaliliśmy, że są to całkowite funkcje definiowalne w λ-rachunku. Teraz chcemy ustalić związek między częściowymi funkcjami obliczalnymi i funkcjami definiowalnymi w λ-rachunku. Wymaga to w pierwszym rzędzie przyjęcia definicji częściowej funkcji definiowalnej. Pojęcie to definiuje się na dwa sposoby. Teraz zajmiemy się sposobem klasycznym, uważanym jednak za nieco mniej adekwatny.

Przypuśćmy, że f jest częściową funkcją naturalną. Dla uproszczenia będziemy zakładać, że jest to funkcja jednoargumentowa. Uogólnienie definicji na przypadek funkcji wieloargumentowych nie powinno sprawiać kłopotów. Tak więc mamy funkcję $f : N \to N$, która dla niektórych argumentów może nie być określona. Liczby naturalne w λ-rachunku, jak zwykle, reprezentujemy za pomocą termów c_n pewnego systemu liczbowego (nie muszą to być numerały Churcha, choć mogą).

Funkcja f jest definiowalna w λ-rachunku (jest λ-definiowalna lub λ-reprezentowalna), jeżeli istnieje term F taki, że dla dowolnego argumentu $n \in N$

1) jeżeli wartość $f(n)$ nie jest określona, to term Fc_n nie ma postaci normalnej,

2) jeżeli wartość $f(n)$ jest określona, to $Fc_n =_\beta c_{f(n)}$.

O częściowych funkcjach definiowalnych można dowieść następujące

Twierdzenie 15.1 *Częściowa funkcja naturalna jest definiowalna w λ-rachunku wtedy i tylko wtedy, gdy jest częściową funkcją rekurencyjną.* □

W dalszym ciągu tylko nieco uprawdopodobnimy prawdziwość powyższego twierdzenia.

15.2 Semantyka *call_by_value*

15.2.1 Definicja semantyki

Semantyka *call_by_value* jest pojęciem z teorii języków programowania, w której oprócz analizy różnych konstrukcji stosowanych w językach programowania wyjaśnia się także, na czym polega wykonanie programu lub nawet tworzy i analizuje aparat do opisu działania programów. W przypadku rachunku lambda odpowiednikiem programu jest term, a wykonanie programu polega na wyliczeniu jego wartości.

Wyliczanie wartości aplikacji AB może być interpretowane jako wykonanie funkcji A z parametrem B. W semantyce *call_by_value* parametr jest jakby wywoływany przez wartość, a więc wymaga wyliczenia przed przystąpieniem do wykonywania funkcji. Samo wykonanie funkcji polega na wykonywaniu β-redukcji. Jedna z metod definiowania semantyki (tzw. semantyka małych kroków) polega na opisaniu pojedyńczych kroków wykonania programu. Wtedy wykonanie programu polega na iterowaniu wykonania pojedyńczych kroków. W związku z tym opiszemy relację $A \to B$, którą będziemy interpretować jako stwierdzenie, że term B otrzymujemy z A w wyniku wykonania pojedyńczego przekształcenia.

Przyjmijmy, że symbole T z indeksami będą oznaczać dowolne termy λ-rachunku, a symbole V – termy będące wartościami.

Wartościami będziemy nazywać dowolne abstrakcje, czyli termy zaczynające się symbolem λ. Oprócz wartości (potencjalnych) zdefiniujemy też pojęcie wartości termu T. Wartością termu T będzie zawsze jedna z wartości, a więc abstrakcja odpowiednio związana z termem T. Teraz tylko zasygnalizujemy, że wartością dowolnej abstrakcji jest ona sama.

Relację redukcji w jednym kroku w semantyce *call_by_value* definiujemy przez indukcję przyjmując, że spełnia następujące reguły:

$$\frac{T_1 \to T_1'}{T_1T_2 \to T_1'T_2}, \quad \frac{T_2 \to T_2'}{VT_2 \to VT_2'} \text{ oraz } \frac{}{(\lambda x\ T)V \to T[x := V]}.$$

Aby zrozumieć tę definicję zauważmy, że możemy przekształcać tylko aplikacje, za to na trzy sposoby (zależnie od postaci T_1T_2, VT_2 lub $(\lambda x\ T)V$). W żaden sposób nie przekształcamy zmiennych, abstracji, ani termów postaci xT ze zmienną x. Mając aplikację możemy przedstawić ją w postaci $T_1T_2 \ldots T_{n-1}T_n = (T_1T_2 \ldots T_{n-1})T_n$, gdzie T_1 nie jest już aplikacją, a więc jest albo zmienną, albo wartością. Dzięki pierwszej regule, przekształcanie takiego termu sprowadza się do przekształcania jego fragmentu T_1T_2. Jeżeli T_1 jest zmienną, to obliczenia kończą się bez wyliczenia wartości. Jeżeli T_1 jest wartością, to stosując drugą regułę próbujemy wyliczyć wartość termu T_2. Jeżeli zakończy się to sukcesem, to trzecia reguła pozwala zastosować β-redukcję.

Lemat 15.2 *Redukcja $\to$ jest β-redukcją z dodatkowymi ograniczeniami, a więc $\to \subseteq \to_\beta$.* $\square$

Zauważmy też, że mamy

Lemat 15.3 *Dla dowolnego T_1 jest najwyżej jeden term T_2 taki, że $T_1 \to T_2$.* $\square$

Jak zwykle, relację $\to$ rozszerzamy do redukcji w wielu krokach $\twoheadrightarrow$.

Wartością termu T nazywamy wartość V taką, że $T \twoheadrightarrow V$. Oczywiście, jeżeli wartość termu istnieje, to jest jednoznacznie wyznaczona.

Widać również, że redukowanie termu T może zakończyć się na trzy sposoby: utworzeniem nieskończonej redukcji $T = T_0 \to T_1 \to \ldots \to T_n \to \ldots$, utworzeniem skończonej redukcji zakończonej termem, który jednak nie jest wartością, lub też znalezieniem wartości termu T.

Ograniczając możliwości redukowania termów zwykle utrudniamy prowadzenie obliczeń. Dalej jednak spróbujemy się pobieżnie przekonać, że nie dotyczy to semantyki *call_by_value*.

15.2.2 Wartości boolowskie i testowanie, operacja pary

Wartości boolowskie, testy i definicje warunkowe w przypadku semantyki *call_by_value* możemy reprezentować tak, jak to zostało przedstawione w rozdziale 7.4

Na przykład wyrażenie warunkowe redukuje się w semantyce *call_by_value* w następujący sposób:

$$\textbf{if } A \textbf{ then } B \textbf{ else } C \equiv (\lambda xyz.\, xyz)ABC \to^* (\lambda xyz.\, xyz)\overline{A}BC \to$$

$$\to (\lambda yz.\, \overline{A}yz)BC \to^* (\lambda yz.\, \overline{A}yz)\overline{B}C \to^* \overline{A}\,\overline{B}\,\overline{C},$$

gdzie symbolem $\overline{A}$ oznaczamy wartość termu A.

Podobnie mamy

$$\textbf{if true then } B \textbf{ else } C \to^* \textbf{true}\,\overline{B}\,\overline{C} \to (\lambda y.\ \overline{B})\overline{C} \to \overline{B}$$

i analogicznie dla **false**.

Przeprowadzone rachunki wymagają jednak drobnego założenia o istnieniu wartości termów A, B i C.

Używając semantyki *call_by_value* możemy właściwie każdy term uważać za wartość. Na przykład, do testowania zamiast **if then else** możemy użyć bardziej skomplikowanego wyrażeniem $(\textbf{if } A \textbf{ then } \lambda t\, B \textbf{ else } \lambda t\, C)\, I$, które redukuje się w następujący sposób:

$$(\textbf{if true then } \lambda t\, B \textbf{ else } \lambda t\, C)\, I \to^* \textbf{true}\,(\lambda t\, B)\,(\lambda t\, C)\, I \equiv$$

$$\equiv (\lambda xy.\, x)\,(\lambda t\, B)\,(\lambda t\, C)\, I \to (\lambda y\,(\lambda t\, B))\,(\lambda t\, C)\, I \to (\lambda t\, B)\, I \to B$$

(zamiast termu B posługujemy się w nim wartością $\lambda t\, B$). Podobnie,

$$(\textbf{if false then } \lambda t\, B \textbf{ else } \lambda t\, C)\, I \to^* C.$$

Funkcję pary $[\cdot,\cdot]$ też definiujemy w zwykły sposób (patrz rozdział 7.5) przyjmując

$$[\cdot,\cdot] \equiv \lambda xyb.\ bxy.$$

To samo dotyczy funkcji

$$\lambda p\, p\,\textbf{true} \quad \text{oraz} \quad \lambda p\, p\,\textbf{false}$$

odczytujących współrzędne pary. Nietrudno sprawdzić, że

$$[\cdot,\cdot]\, X\, Y \to^* \lambda b\,\overline{X}\,\overline{Y} \equiv [\overline{X},\overline{Y}]$$

oraz

$$(\lambda p\, p\,\textbf{true})\,[X,Y] \to [X,Y]\,\textbf{true} \to^* \overline{X},$$
$$(\lambda p\, p\,\textbf{false})\,[X,Y] \to [X,Y]\,\textbf{false} \to^* \overline{Y}$$

przynajmniej dla X i Y mających wartość.

15.2.3 Liczby naturalne

Używając semantyki *call_by_value* liczby naturalne wygodnie jest reprezentować za pomocą numerałów Barendregta (patrz rozdział 7.7). Wtedy termy

$$Z = \lambda x\, x\,\textbf{true}, \quad S^+ = \lambda xz.\, z\,\textbf{false}\,x = \lambda x\,[\textbf{false}, x] \quad \text{oraz} \quad P^- = \lambda x\, x\,\textbf{false}$$

definiujące w λ-rachunku podstawowe operacje na tak reprezentowanych liczbach naturalnych (test zera, następnik i poprzednik odpowiednio) mają oczekiwane własności, nawet jeżeli są redukowane zgodnie z rozważaną semantyką. Tak więc dla wszystkich liczb naturalnych n mamy

$$Z\,\ulcorner 0 \urcorner \equiv (\lambda x\, x\,\textbf{true})\, I \to^* \textbf{true},$$

$$Z\,\ulcorner n{+}1 \urcorner \equiv (\lambda x\, x\,\textbf{true})\,[\textbf{false}, \ulcorner n \urcorner] \to [\textbf{false}, \ulcorner n \urcorner]\,\textbf{true} \to \textbf{true}\,\textbf{false}\,\ulcorner n \urcorner \to^* \textbf{false},$$

$$S^+\,\ulcorner n \urcorner \equiv (\lambda xz.\, z\,\textbf{false}\,x)\,\ulcorner n \urcorner \to \lambda z\, z\,\textbf{false}\,\ulcorner n \urcorner \equiv \ulcorner n+1 \urcorner,$$

$$P^-\,\ulcorner n+1 \urcorner \equiv (\lambda x\, x\,\textbf{false})\,[\textbf{false}, \ulcorner n \urcorner] \to [\textbf{false}, \ulcorner n \urcorner]\,\textbf{false} \to^* \ulcorner n \urcorner.$$

15.2.4 Punkty stałe i rekursja

Spróbujemy teraz pokazać na dostatecznie ogólnym przykładzie, że klasa funkcji obliczalnych zgodnie z semantyką *call_by_value* jest zamknięta ze względu na rekursję prostą. W tym celu posłużymy się kombinatorem punktu stałego. Jak zwykle, będzie nam też potrzebny term reprezentujący operację poprzednika.

Przypuśćmy, że $f : N \to N$ jest funkcją taką, że

$$f(0) = m \;\text{ oraz }\; f(n+1) = h(f(n), n)$$

dla wszystkich liczb naturalnych n. Przyjmijmy, że H jest termem definiującym funkcję h (w semantyce *call_by_value*, tak więc $H\,\ulcorner k\urcorner\,\ulcorner n\urcorner \to^* \ulcorner h(k,n)\urcorner$). Oczywiście, term definiujący funkcję f powinien spełniać równość

$$F\,x = \mathbf{if}\ Z\,x\ \mathbf{then}\ \ulcorner m\urcorner\ \mathbf{else}\ (H\,(F\,(P^-\,x))\,(P^-\,x)).$$

Równości tej nadamy jednak nieco inną postać:

$$F\,x = (\mathbf{if}\ Z\,x\ \mathbf{then}\ \lambda t\ulcorner m\urcorner\ \mathbf{else}\ (\lambda yt.\,H\,(F\,y)\,y)\,(P^-\,x))\,I.$$

Niech teraz

$$G = \lambda f x.\,(\mathbf{if}\ Z\,x\ \mathbf{then}\ \lambda t\ulcorner m\urcorner\ \mathbf{else}\ (\lambda yt.\,H\,(F\,y)\,y)\,(P^-\,x))\,I.$$

Term F możemy teraz zdefiniować jako $\mathbf{fix}\,G$, czyli wynik zaplikowania kombinatora punktu stałego **fix** do termu G. Powinniśmy jednak użyć specjalnego kombinatora **fix**.

Zadanie 15.1 Prześledź, co się stanie, jeżeli w roli operatora punkty stałego użyjemy operatora $\mathbf{Y}$ lub operatora Turinga $\mathbf{\Theta}$. □

Niech

$$\mathbf{fix} \equiv \lambda f\,A_f\,A_f, \quad \text{gdzie} \quad A_f \equiv \lambda x\,f(\lambda z\,xxz).$$

Zauważmy, że

$$\mathbf{fix}\,G \equiv (\lambda f\,A_f\,A_f)\,G \to A_G\,A_G \to G\,(\lambda z\,A_G\,A_G\,z).$$

Stąd w szczególności mamy (podkreślone zostały przekształcane redeksy)

$$\begin{aligned}
&\underline{\mathbf{fix}\,G}\ulcorner n+1\urcorner \to \\
&\to \underline{A_G\,A_G}\ulcorner n+1\urcorner \\
&\to \underline{G\,(\lambda x.\,A_G\,A_G\,x)}\ulcorner n+1\urcorner \\
&\to^* (\mathbf{if}\ \underline{Z\ulcorner n+1\urcorner}\ \mathbf{then}\ \lambda t\ulcorner m\urcorner\ \mathbf{else}\ ((\lambda yt.\,H\,((\lambda z\,A_G\,A_G\,z)\,y)\,y)\,(P\ulcorner n+1\urcorner))\,I \\
&\to^* (\mathbf{if}\ \underline{\mathbf{false}}\ \mathbf{then}\ \underline{\lambda t.\ulcorner m\urcorner}\ \mathbf{else}\ ((\lambda yt.\,H\,((\lambda z\,A_G\,A_G\,z)\,y)\,y)\,(P\ulcorner n+1\urcorner))\,I \\
&\to^* (\lambda z\,\mathbf{false}\,(\lambda t\ulcorner m\urcorner)\,z)\,((\lambda yt.\,H\,((\lambda z\,A_G\,A_G\,z)\,y)\,y)\,(\underline{P\ulcorner n+1\urcorner}))\,I \\
&\to^* (\lambda z\,\mathbf{false}\,(\lambda t\ulcorner m\urcorner)\,z)\,(\underline{(\lambda yt.\,H\,((\lambda z\,A_G\,A_G\,z)\,y)\,y)\ulcorner n\urcorner})\,I \\
&\to \underline{(\lambda z\,\mathbf{false}\,(\lambda t\ulcorner m\urcorner)\,z)\,(\lambda t\,H\,((\lambda z\,A_G\,A_G\,z)\ulcorner n\urcorner)\ulcorner n\urcorner)}\,I \\
&\to \underline{\mathbf{false}\,(\lambda t\ulcorner m\urcorner)\,(\lambda t\,H\,((\lambda z\,A_G\,A_G\,z)\ulcorner n\urcorner)\ulcorner n\urcorner)}\,I \\
&\to^* \underline{(\lambda t\,H\,((\lambda z\,A_G\,A_G\,z)\ulcorner n\urcorner)\ulcorner n\urcorner)\,I} \\
&\to H\,(\underline{(\lambda z\,A_G\,A_G\,z)\ulcorner n\urcorner})\ulcorner n\urcorner \to^* H\,(A_G\,A_G\ulcorner n\urcorner)\ulcorner n\urcorner.
\end{aligned}$$

Jeżeli teraz dodatkowo przyjmiemy, że $A_G\, A_G \ulcorner n \urcorner \to^* \ulcorner f(n) \urcorner$, to otrzymamy, że

$$\begin{aligned}\mathbf{fix}\, G \ulcorner n+1 \urcorner \to A_G\, A_G \ulcorner n+1 \urcorner &\to^* H\, (A_G\, A_G \ulcorner n \urcorner) \ulcorner n \urcorner \to^* \\ &\to^* H \ulcorner f(n) \urcorner \ulcorner n \urcorner \to^* \ulcorner h(f(n), n) \urcorner \equiv \ulcorner f(n+1) \urcorner.\end{aligned}$$

Łatwo także przekonać się (powtarzając przedstawione rachunki), że

$$\mathbf{fix}\, G \ulcorner 0 \urcorner \to A_G\, A_G \ulcorner 0 \urcorner \to^* (\lambda t. \ulcorner m \urcorner)\; I \to \ulcorner m \urcorner.$$

W tej sytuacji zasada indukcji matematycznej daje nam, że

$$\mathbf{fix}\, G \ulcorner n \urcorner \to A_G\, A_G \ulcorner n \urcorner \to^* \ulcorner f(n) \urcorner$$

dla wszystkich liczb naturalnych n. Oznacza to, że w semantyce *call_by_value* termy $\mathbf{fix}\, G$, $A_G\, A_G$, a także będące wartościami termy $\lambda x\, \mathbf{fix}\, G\, x$ oraz $\lambda x\, A_G\, A_G\, x$ definiują funkcję f, o ile funkcja h jest reprezentowana odpowiednim termem.

15.3 Przykład częściowej funkcji λ-definiowalnej

W tym rozdziale spróbujemy upiec dwie pieczenie na jednym ogniu.

15.4 λI-rachunek

Patrz lista zadań numer 6.

15.5 $\lambda\eta$-rachunek i $\lambda\delta$-rachunek

Patrz listy zadań o numerach 6 i 7.

16 Dodatek: własność $\diamond$ redukcji równoległej

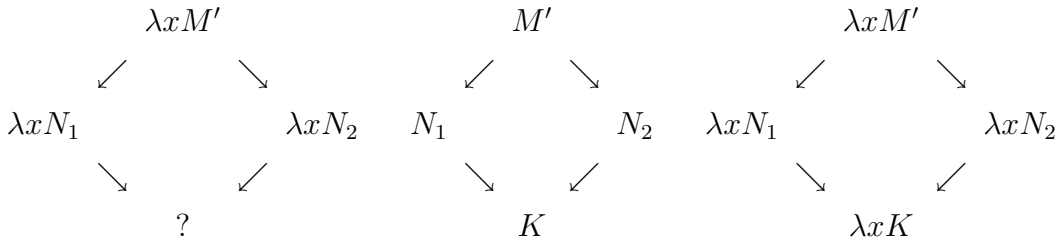
Lemat 16.1 *Relacja redukcji równoległej $\to_{r\beta}$ ma własność $\diamond$, a dokładniej.*

Dowód. Niech M oznacza term, który przekształcamy na dwa sposoby: $M \to_{r\beta} N_1$ i $M \to_{r\beta} N_2$. Lemat dowodzimy przez indukcję ze względu na budowę termu M. Dla każdego rodzaju termów będziemy rozważać wiele przypadków odpowiadających różnym dopuszczalnym sposobom przekształcania.

Przypadek 1: $N_2 = M$. Zawsze możemy korzystać ze zwrotności $\to_{r\beta}$, czyli przekształcać nic nie robiąc. Wtedy N_1 przekształcamy w $K = N_1$ oraz $N_2 = M$ też przekształcamy w $N_1 = K$. Dalej zakładamy, że wykonujemy istotne przekształcenia M.

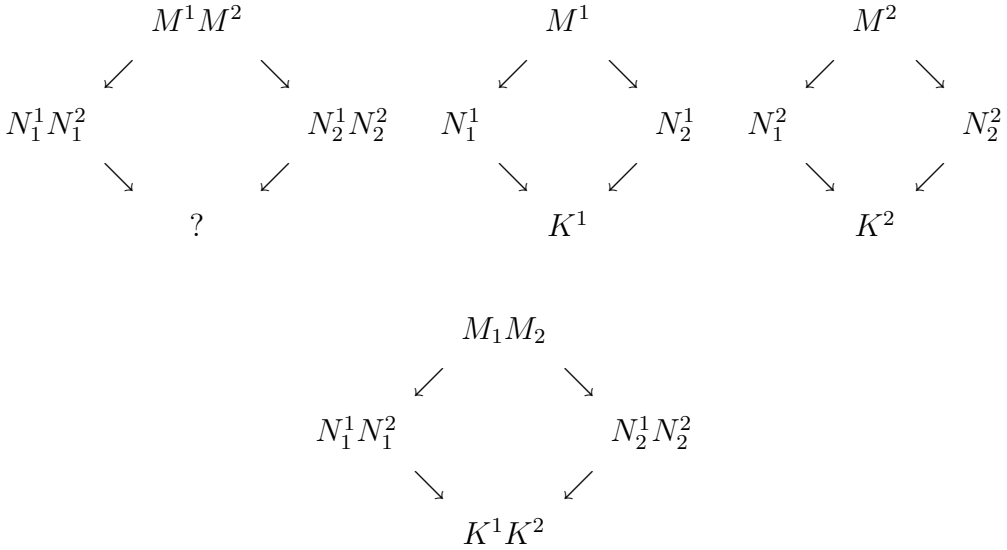
Przypadek 2: M jest zmienną. Zmiennej nie możemy redukować w istotny sposób.

Przypadek 3: $M = \lambda x M'$. Jedyny sposób redukowania abstrakcji $\lambda x M'$ to przekształcanie termu M'. Po przekształceniu na dwa sposoby dostajemy termy $\lambda x N_1$ i $\lambda x N_2$ takie, jak na pierwszej części poniższego rysunku.

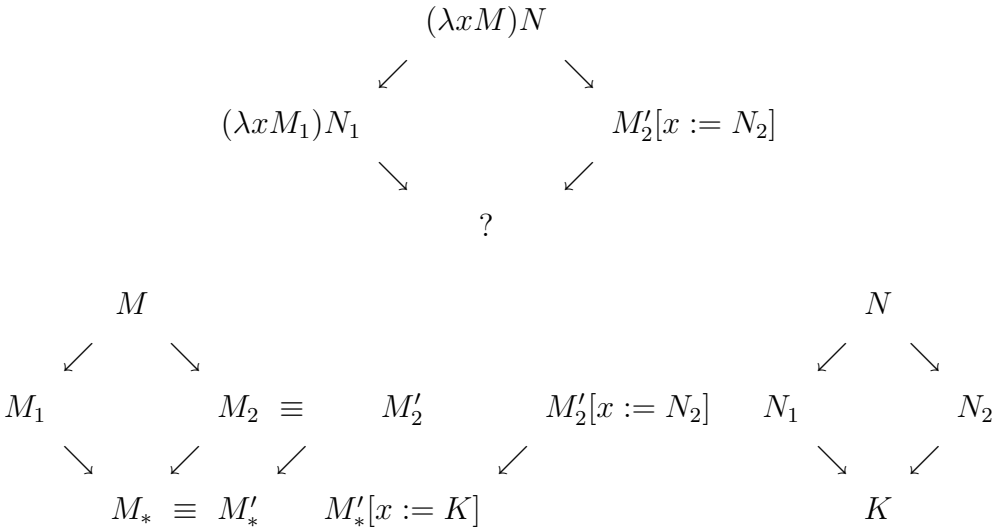


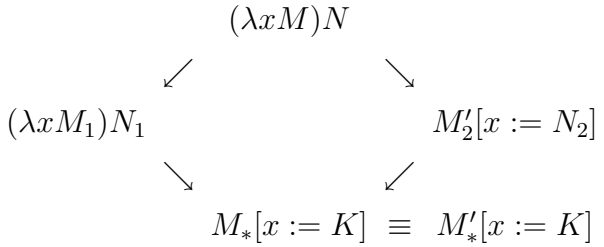
Termu N_1 i N_2 są takie, jak w środkowej części rysunku. Dla nich znajdujemy term K korzystając z założenia indukcyjnego. Term ten ma własności pokazane na ostatniej części rysunku.

Przypadek 4: M jest aplikacją, która nie jest redeksem, lub w żadnym przypadku nie jest redukowana jak redeks. Dowód jest analogiczny do dowodu z poprzedniego przypadku.



Przypadek 5: M jest redeksem redukowanym w mieszany sposób, jako redeks i jako zwykła aplikacja.





Spis treści