

Zastosowanie unifikacji do problemów
wyprowadzania typów

Rozprawa doktorska

Aleksy Schubert
Instytut Informatyki
Uniwersytetu Warszawskiego
alx@mimuw.edu.pl

25 stycznia 2010

Mamie

Motto:

*Gdybym kimkolwiek miał być, a nie sobą,
Tylko nim chciałbym być.*

William Shakespeare, *Koriolan*, I, 1.

Spis treści

Spis treści	3
Wprowadzenie	7
1 Podstawowe pojęcia	13
1.1 Beztypowy rachunek lambda	13
1.2 Rachunki lambda z typami	17
1.2.1 Rachunki lambda w wersji Curry’ego	17
1.2.2 Rachunki lambda w wersji Churcha	22
1.2.3 Problemy uzupełniania osądów	28
1.3 Logika pierwszego rzędu i przepisywanie termów	30
1.4 Podstawowe pojęcia związane z unifikacją	33
2 Unifikacja	37
2.1 Unifikacja z prostymi argumentami	37
2.1.1 Automaty dwulicznikowe	37
2.1.2 Nierozstrzygalność unifikacji drugiego rzędu	41
2.2 Unifikacja ze zmiennymi w pozycjach czołowych	65
2.3 Unifikacja ze zmiennymi w pozycjach czołowych — trzeci rząd	69
3 Wybrane systemy typów i unifikacja	71
3.1 Wyprowadzanie typów w rachunku $\lambda 2$ w wersji Churcha	71
3.2 Kontekstowe wyprowadzanie typów dla logiki pierwszego rzędu	82
3.2.1 Zanurzenie logiki pierwszego rzędu w λP	85
3.2.2 Nierozstrzygalność wyprowadzania typów z konteksta- mi pierwszego rzędu	94
3.2.3 Rozstrzygalność kontekstowego wyprowadzania typów w logice pierwszego rzędu	97
4 Podsumowanie	131
Podziękowania	133

Wprowadzenie

Rachunki lambda

Początki rachunku λ Rachunek λ został wprowadzony niezależnie przez Curry’ego [Cur30a, Cur30b] i Churcha [Chu33] w początkach lat trzydziestych XX w. jako propozycja formalizacji podstaw matematyki. System ten okazał się sprzeczny jak pokazali Kleene i Rosser [KR35]. W roku 1936 Church i Rosser wskazali niesprzeczny podsystem wcześniej wprowadzonego rachunku. Określone przy pomocy tego systemu pojęcie *λ -definiowalności* stanowiło jeden z kamieni milowych przy tworzeniu pojęcia *efektywnej obliczalności* (współczesny opis tych zagadnień można znaleźć w książce Barendregta [Bar84]).

Rachunki lambda z typami Wraz z rozwojem rachunku beztypowego zaczęły pojawiać się rachunki λ z typami. Pierwszą taką propozycją był rachunek lambda z typami prostymi $\lambda \rightarrow$, który został niezależnie wprowadzony przez Curry’ego [Cur34] i Churcha [Chu40]. Główną motywacją wprowadzenia tego rachunku było umożliwienie wyrażania informacji o dziedzinie i przeciwdziedzinie określonej funkcji.

System \mathcal{F} (oznaczany tutaj jako $\lambda 2$) — drugi rozważany tutaj rachunek λ z typami — został zaproponowany niezależnie przez Girarda [Gir72] (jako narzędzie do pokazywania twierdzenia o eliminacji cięcia) oraz przez Reynoldsa [Rey74] jako propozycja abstrakcji języka programowania pozwalającego wykorzystywać polimorficzne definicje programów. Ten ostatni nurt zaowocował wprowadzeniem języka programowania ML [Mil78, Mil84] oraz innych języków funkcyjnych w tym Mirandy [Tur86], Haskell’a [HF92] czy Camla [Ler97].

Z czasem zauważono ścisły związek między różnego rodzaju logikami intuicjonistycznymi a rachunkami λ z typami — sposób tworzenia termów i przypisywania im typów doskonale odpowiada konstrukcyjnej naturze formalizmów intuicjonistycznych. Na przykład rachunek λ z typami prostymi okazał się odpowiadać intuicjonistycznemu rachunkowi zdań, zaś system \mathcal{F} — zdaniowej logice drugiego rzędu. Te i wiele innych obserwacji tej natury pozwoliły na wprowadzenie pojęcia *izomorfizmu Curry’ego-Howarda* — wzajemnej odpowiedniości między dowodami w systemach naturalnej deduk-

cji a λ -termami. Związek pomiędzy niepustością typów w *logice kombinatoryj* (znajdującej się w ścisłym związku z rachunkiem lambda) a formułami intuicjonistycznej logiki zdaniowej w postaci Hilberta wykazał Curry w książce [CF58]. W pracy tej pojawiły się też uwagi o istnieniu stosownego związku bezpośrednio dla rachunku lambda z typami prostymi i logiki intuicjonistycznej w wersji z naturalną dedukcją. Obecnie funkcjonuje przekonanie podobne do słynnej tezy Turinga, że praktycznie każdy rachunek lambda z typami odpowiada jakiemuś systemowi logicznemu.

Powyższe powiązanie decyduje w ogromnym stopniu o zastosowaniu rachunków λ w teorii systemów automatycznego i wspomaganego maszynowo dowodzenia twierdzeń. Na ideach rachunków λ z typami i powiązanych z nimi logikach wyższego rzędu oparte były i są między innymi projekt AUTOMATH de Bruijna [dB80], Edinburgh Logical Framework zaproponowany przez Harpera, Honsela i Plotkina [HHP93], Isabelle zaproponowana przez Paulsona [Pau86], HOL [Gor87], Lego wprowadzone przez Harpera i Milnera [HP89], Coq [CH85], Nuprl [Con86].

W wyniku zainteresowania tematyką automatycznego dowodzenia twierdzeń pojawił się trzeci rachunek z typami, którym się tutaj będziemy zajmować — λP . Rachunek ten, opracowany przez Harpera, Honsela i Plotkina [HHP93] pozwala na wprowadzanie zależności typu od obiektu (termu). Pozwala to z jednej strony myśleć o takich typach jak o konstrukcjach umożliwiających wyrażanie zależności w rodzaju `list(n)` oznaczających na przykład listy zawierające wyłącznie elementy o wartości n . Z drugiej strony wprowadzona tutaj kwantyfikacja pozwala myśleć o typach (po zastosowaniu izomorfizmu Curry’ego-Howarda) jak o zdaniach z kwantyfikacją „pierwszego rzędu”. W związku z tym rachunek ten znajduje swoje zastosowanie w systemach hybrydowych łączących elementy programowania i dowodzenia twierdzeń, np. we wspomnianym już Edinburgh Logical Framework.

Problemy decyzyjne dla rachunków λ W związku z potrzebą automatyzacji różnych operacji (np. tworzenia dowodu w jakiejś logice, analizy statycznej programu funkcyjnego itp.) pojawił się szereg problemów decyzyjnych związanych z typami w różnych rachunkach λ . Bardziej szczegółowe ich omówienie znajduje się w sekcji 1.2.3, gdzie wprowadzamy szczegółowe ich definicje. Tutaj wspomnimy jedynie, że problemy te polegają zasadniczo na rozważaniu, czy z fragmentu typowego osądu rachunku λ postaci:

$$\Gamma \vdash M : \tau,$$

gdzie Γ jest kontekstem interpretującym zmienne wolne, M — λ -termem, a τ typem, da się stworzyć pełny osąd. Problemy te, gdy nieznanym jest typ i /lub kontekst mają silny związek z różnymi wariantami unifikacji.

Unifikacja

Unifikacja pierwszego rzędu Unifikacja rozumiana jako rozwiązywanie równań na termach logiki pierwszego rzędu (na niewiadome można podstawić termy) została wprowadzona przez Herbranda w 1930 roku [Her68]. Pierwszy pełny algorytm podany został niezależnie przez Robinsona [Rob65] oraz Guarda [Gua64].

Unifikacja pierwszego rzędu gra kluczową rolę w programowaniu w logice, gdzie stanowi podstawę mechanizmu obliczeniowego — SLD-rezolucji [AE82]. Unifikacja ta znalazła także zastosowanie w dziedzinie rachunków λ , o czym będziemy jeszcze pisać poniżej.

Unifikacja wyższego rzędu Jako uogólnienie unifikacji pierwszego rzędu pojawiło się pojęcie unifikacji wyższego rzędu. W tej unifikacji na niewiadome w porównywanych termach można podstawić obiekty funkcyjne, czyli takie, które przyjmują argumenty. Dla tego problemu podany został semialgorytm wyliczający dla danego równania wszystkie unifikatory (może ich być nieskończenie wiele) [PJ72] oraz semialgorytm sprawdzający istnienie unifikatora [Hue75]. Mniej więcej w tym samym czasie okazało się, że problem ten jest nierozstrzygalny [Hue73, Luc72]. Powyższe wyniki dotyczyły unifikacji trzeciego rzędu (niewiadome mogą mieć jako argumenty funkcje). Dla unifikacji drugiego rzędu (niewiadome mogą mieć jako argumenty tylko termy) nierozstrzygalność została udowodniona dopiero w roku 1981 przez Goldfarba [Gol81]. W dowodzie Goldfarba wykonywana była redukcja 10. problemu Hilberta do unifikacji drugiego rzędu.

Wyprowadzanie typów

Problem *wyprowadzania typów* dla danego rachunku λ polega na określaniu, czy dla danego termu M istnieje kontekst Γ i typ τ , takie że wyprowadzalny w rachunku jest osąd

$$\Gamma \vdash M : \tau.$$

Omówimy tutaj historię tego problemu dla interesujących nas tutaj rachunków $\lambda \rightarrow$, $\lambda 2$ oraz λP .

W literaturze rozważany jest także wariant powyższego problemu nazywany tutaj problemem *kontekstowego wyprowadzania typów*, w którym jako dane występują kontekst Γ i term M , a w pytaniu chodzi o istnienie typu τ , takiego że wspomniany wyżej osąd ma wyprowadzenie.

Wyprowadzanie typów w $\lambda \rightarrow$ Rozstrzygalność problemu (kontekstowego i normalnego) wyprowadzania typów dla $\lambda \rightarrow$ została niezależnie udowodniona przez Curry'ego [Cur69] oraz Hindleya [Hin69]. Szczególną rolę gra tutaj wspomniana wcześniej unifikacja pierwszego rzędu. Dana para Γ, M jest przetwarzana na układ równań takiej unifikacji. Rozwiązywalność tego

układu jest równoważna istnieniu typu τ , dla którego osąd $\Gamma \vdash M : \tau$ ma wyprowadzenie w $\lambda \rightarrow$. Typ τ można bezpośrednio odczytać z uzyskanego unifikatora.

Unifikacja pierwszego rzędu ma zastosowanie także w (kontekstowym i normalnym) wyprowadzaniu typów dla wyrażeń języka programowania ML, który stanowi rozszerzenie rachunku $\lambda \rightarrow$ o pewną ograniczoną postać polimorfizmu, mającą jednak rozstrzygalne (kontekstowe) wyprowadzanie typów. Dowód rozstrzygalności wyprowadzania typów w tym rachunku znajduje się w klasycznej pracy Milnera [Mil78].

Wyprowadzanie typów w $\lambda 2$ Kluczowy wynik dla tego problemu — nierozstrzygalność zwykłego i kontekstowego wyprowadzania typów w $\lambda 2$ -Curry udowodnił Wells w swojej pracy [Wel99]. W pracy tej wykonana została redukcja nierozstrzygalnego problemu semiunifikacji, który polega na rozwiązywaniu pewnych nierówności na termach.

Wcześniej znany był wynik częściowy opracowany przez Böhma i Pfenninga w dwóch odrębnych pracach [Boe85, Pfe88] dla tak zwanego rachunku $\lambda 2$ z wytartymi typami — przez ten system rozumiemy rachunek zdefiniowany w zasadzie tak jak $\lambda 2$ -Church z tym, że w niektórych miejscach zamiast typów mogą znajdować się zaznaczone puste miejsca, w których w intencji może się pojawić dowolny, pasujący typ. Gramatyka dla termów tego rachunku wygląda tak:

$$\Lambda'_{\mathcal{F}} = \mathbf{V} \mid \Lambda'_{\mathcal{F}} \Lambda'_{\mathcal{F}} \mid \Lambda'_{\mathcal{F}} \mathbf{T} \mid \Lambda'_{\mathcal{F}} [] \mid \lambda \mathbf{V} : \mathbf{T} \Lambda'_{\mathcal{F}} \mid \lambda \mathbf{V} : [] \Lambda'_{\mathcal{F}} \mid \Lambda \mathbf{V} \Lambda'_{\mathcal{F}}.$$

Dla systemu $\lambda 2$ -Church rozstrzygalność kontekstowego wyprowadzania typów jest znana od dosyć dawna, a jej formalny dowód w ogólniejszym przypadku — dla dowolnego PTS mającego tzw. własność silnej normalizacji — znajduje się w pracy [Ben93].

Dotąd nieznanne były wyniki dla problemu wyprowadzania typów dla $\lambda 2$ -Church. Podsumowanie znanych wyników znajduje się w tablicy 1.

	$\lambda 2$ -Curry	$\lambda 2$ -Church
$\Gamma \vdash M : ?$	nierozstrzygalne — łatwa redukcja z [Wel99]	rozstrzygalne — łatwa redukcja do [Ben93]
$? \vdash M : ?$	nierozstrzygalne — [Wel99]	

Tablica 1: Rozstrzygalność problemów decyzyjnych dla $\lambda 2$

Wyprowadzanie typów w λP Sytuacja w przypadku tego rachunku jest dosyć interesująca, gdyż problem sprawdzania typów (dane są kontekst, term i typ) jest tutaj rozstrzygalny zob. np. [SU98]. Jednak przy tym problem kontekstowego wyprowadzania typów dla λP jest nierozstrzygalny, jak udowodnił Dowek [Dow93].

Zastosowania problemów wyprowadzania Interesujący nas tutaj problem wyprowadzania typów tak kontekstowy, jak i normalny pojawia się w naturalny sposób w badaniach nad językami programowania, gdyż algorytmy wyprowadzania typów służą do uzupełniania programów o informacje o rodzaju dopuszczalnych danych wejściowych i wyników. Problem wyprowadzania typów pojawia się jednak także w obszarze czystej logiki. Problemy wyprowadzania typów są jedną z możliwych form rozważanego tam problemu znajdowania dowodu dla danego jego szkieletu. Tego typu problemami zajmował się Voronkow w pracy [Vor96] oraz innych pracach prowadzących do zamieszczonego tam wyniku. Nasze podejście jest jednak w znaczący sposób odmienne od problemu Voronkowa, gdyż w naszym przypadku zajmujemy się termami, które nie muszą być w postaci normalnej (odpowiada to używaniu szkieletów z zaznaczonymi regułami (cut)) oraz dodatkowo nasze termy w miejscu, gdzie używana jest reguła podstawienia na zmienną kwantyfikowaną ogólnie, zawierają algebraiczne termy pierwszego rzędu.

Wkład niniejszej pracy

Związek między unifikacją a wyprowadzaniem typów W niniejszej pracy przedstawione jest kilka szczególnych przypadków problemu unifikacji drugiego rzędu wraz z szeregiem konstrukcji wskazujących na związek między tymi szczególnymi przypadkami a wyprowadzaniem typów w rachunkach λ z typami. Konstrukcje te wskazują, że szczególną rolę grają tutaj instancje problemu unifikacji, gdzie argumenty zmiennej drugiego rzędu są termami bez niewiadomych (zmiennych wolnych).

Główny związek między unifikacją a wyprowadzaniem typów stanowi redukcja unifikacji z prostymi argumentami do problemu wyprowadzania typów dla wersji Churcha rachunku $\lambda 2$. Unifikacja z prostymi argumentami to właśnie unifikacja, gdzie argumenty zmiennej drugiego rzędu są termami bez niewiadomych. Nierozstrzygalność takiej unifikacji implikuje nierozstrzygalność wyprowadzania typów (podrozdział 3.1).

Drugi związek tutaj uzyskany jest odwrotnej natury — jest to redukcja kontekstowego wyprowadzania typów w podsystemie λP odpowiadającym logice pierwszego rzędu do unifikacji ze zmiennymi w pozycjach czołowych. Taka unifikacja stanowi dalsze ograniczenie zakresu dopuszczalnych niewiadomych — mogą one występować tylko jako pierwszy symbol w termie równania (argumenty oczywiście dalej nie mogą zawierać niewiadomych). Tym razem z rozstrzygalności takiej unifikacji wynika rozstrzygalność wspomnianego

nego kontekstowego wyprowadzania typów (stosowne wyniki znajdują się w podrozdziale 3.2).

Dla pełności przedstawienia umieszczona została tutaj jeszcze jedna znana wcześniej konstrukcja (w niniejszej pracy wyciągamy z niej jednak nowe wnioski), która wiąże kontekstowe wyprowadzanie typów z unifikacją o instancjach ze zmiennymi w pozycjach czołowych zmodyfikowaną w ten sposób, że dozwalamy, aby niewiadome były trzeciego rzędu. Taka unifikacja daje nierozstrzygalność (podrozdział 3.2) kontekstowego wyprowadzania dla systemu, w którym dopuszcza się kwantyfikację po symbolach drugiego rzędu (po symbolach funkcyjnych). Przy okazji dowód ten pokazuje, że jeśli rozluźnimy nasze wymagania dotyczące wyprowadzenia dla danego osądu mieszczącego się w naszym zanurzeniu logiki pierwszego rzędu i pozwolimy, aby jego wyprowadzenie było dowolnym wyprowadzeniem w λP , to problem kontekstowego wyprowadzania typów dla logiki pierwszego rzędu staje się nierozstrzygalny.

Unifikacja drugiego rzędu W niniejszej pracy znajduje się nowy dowód nierozstrzygalności unifikacji drugiego rzędu. Dowód ten jest istotnie różny od dowodu Goldfarba [Gol81], nie wykorzystuje redukcji do mającego bardzo skomplikowany dowód nierozstrzygalności 10. problemu Hilberta, ale za to bezpośrednio wskazuje na mechanizm obliczeniowy kryjący się w unifikacji drugiego rzędu. Dowód tutaj przedstawiony jest na tyle uniwersalny, że łatwo, posługując się jego ogólną ideą, przeprowadzić redukcję wielu innych problemów (zob. [Vea98]). Dowód ten przy okazji obejmuje ciekawy podproblem unifikacji drugiego rzędu — problem unifikacji z prostymi instancjami, gdzie w argumentach niewiadomych mogą występować tylko termy bez niewiadomych (rezultaty te przedstawiamy w podrozdziale 2.1).

Dodatkowo zaprezentowany jest tutaj interesujący rozstrzygalny fragment unifikacji drugiego rzędu — unifikację z symbolami w pozycjach czołowych (podrozdział 2.2). Unifikacja ta jest o tyle ciekawa, że pozwala na wyrażanie własności systemów przepisywania termów, np. czy istnieje ciąg redukcji równoległych o co najwyżej k krokach, które prowadzą od termu t_1 do termu t_2 .

Uzyskujemy tutaj także ciekawe związki między przepisywaniem termów a unifikacją. Pojawiają się one przy okazji prezentacji konstrukcji związanych z dowodem nierozstrzygalności unifikacji drugiego rzędu (sekcja 2.1.2).

Rozdział 1

Podstawowe pojęcia

Podstawowymi pojęciami używanymi tutaj są pojęcia z dziedziny rachunków lambda z typami. Opisane one są w podrozdziałach 1.1 i 1.2. W podrozdziale tym wprowadzimy definicje wzorowane na [Bar92]. Przedstawimy tutaj także intuicjonistyczną logikę pierwszego rzędu (podrozdział 1.3) oraz podstawowe pojęcia związane z unifikacją (podrozdział 1.4).

1.1 Beztypowy rachunek lambda

Nasze przedstawienie zaczniemy od definicji rachunku lambda bez typów.

Definicja 1.1.1 (beztypowy rachunek lambda)

Zbiór λ -termów, oznaczany Λ , utworzony jest na bazie nieskończonego zbioru *zmiennych przedmiotowych* $V = \{v, v', v'', \dots\}$ oraz operacji aplikacji i abstrakcji:

$$\begin{aligned}x \in V &\quad \Rightarrow \quad x \in \Lambda, \\M, N \in \Lambda &\quad \Rightarrow \quad (MN) \in \Lambda, \\M \in \Lambda, x \in V &\quad \Rightarrow \quad (\lambda x M) \in \Lambda.\end{aligned}$$

Przyjmujemy tutaj zwykłe konwencje dotyczące zapisu termów rachunku lambda: litery x, y, z, \dots oznaczają zmienne, litery M, N, L, \dots oznaczają dowolne termy, zewnętrzne nawiasy są pomijane, a także

$$FM_1 \dots M_n \text{ oznacza } (\dots ((FM_1)M_2) \dots M_n)$$

oraz

$$\lambda x_1 \dots x_n. M \text{ oznacza } (\lambda x_1 (\lambda x_2 (\dots (\lambda x_n (M)) \dots))).$$

Definicje rekurencyjne określające różne obiekty składniowe będziemy zwykle zapisywać za pomocą gramatyk bezkontekstowych. Termy z definicji 1.1.1 opisywane są za pomocą gramatyki:

$$\Lambda ::= V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda).$$

Określimy teraz pojęcie wystąpienia zmiennej związanej.

Definicja 1.1.2 (wystąpienie zmiennej związanej)

W termie M wszystkie *wystąpienia* zmiennej x znajdujące się w podtermie postaci $\lambda x.N$ są *związane*.

Zmienne wolne określone są z kolei tak:

Definicja 1.1.3 (zmienne wolne)

Zbiór *zmiennych wolnych* termu M (oznaczany przez $FV(M)$) zdefiniowany jest rekurencyjnie w sposób następujący:

$$\begin{aligned} FV(x) &= \{x\}; \\ FV(MN) &= FV(M) \cup FV(N); \\ FV(\lambda x.M) &= FV(M) - \{x\}. \end{aligned}$$

Definicja 1.1.4 (term zamknięty)

Term zamknięty to taki term M , dla którego $FV(M) = \emptyset$. Zbiór wszystkich takich termów oznaczamy przez Λ^0 .

Definicja 1.1.5 (zastąpienie zmiennej)

Operację *zastąpienia zmiennej x przez y w termie M* (ozn. $M[x := y]$) określamy rekurencyjnie tak:

$$\begin{aligned} x[x := y] &= y; \\ z[x := y] &= z, \text{ o ile } x \neq z; \\ (M_1M_2)[x := y] &= (M_1[x := y])(M_2[x := y]); \\ (\lambda z.P)[x := y] &= \lambda z(P[x := y]), \text{ gdzie } x \neq z \text{ oraz } y \neq z; \\ (\lambda y.P)[x := y] &= \lambda z(P[y := z][x := y]), \text{ gdzie } x \neq z \text{ oraz } y \neq z; \\ (\lambda x.P)[x := y] &= (\lambda x.P). \end{aligned}$$

W obecnej postaci powyższa definicja nie jest dobrze określona, gdyż w piątym punkcie dobór zmiennej z nie jest jednoznaczny. Oczywiście jest jednak, że przy nieskończonej liczbie zmiennych istnieje sposób jednoznacznego wyboru tej zmiennej w zależności od P, x oraz y , tak aby spełnione były wymienione w tym punkcie warunki. Co więcej metoda wyboru może być efektywna (np. przy ustalonym efektywnym dobrym porządku na zmiennych wybieramy pierwszą zmienną nie występującą w Pxy).

Zdefiniowane w definicji 1.1.1 λ -termy utożsamiamy, gdy znajdują się w relacji równoważności zwanej α -konwersją.

Definicja 1.1.6 (α -konwersja)

Relacja α -konwersji, oznaczana przez \equiv_α , zdefiniowana jest rekurencyjnie tak:

- $x \equiv_\alpha x$;
- jeśli $M_1 \equiv_\alpha M'_1$ oraz $M_2 \equiv_\alpha M'_2$, to także $M_1M_2 \equiv_\alpha M'_1M'_2$;

- jeśli $M_1 \equiv_\alpha M'_1$, to $\lambda x M_1 \equiv_\alpha \lambda y M'_1[x := y]$ dla dowolnego y (w szczególności dla $y = x$).

Od tej pory λ -termy będziemy traktowali jako klasy abstrakcji \equiv_α . Dodatkowo przyjmujemy konwencję używania reprezentantów klas abstrakcji, które mają rozłączne zbiory zmiennych wolnych i związanych i co więcej każda zmienna związana jest wiązana w termie tylko w jednym miejscu.

Definicja 1.1.7 (podstawienie)

W wyniku wykonania podstawienia termów N_1, \dots, N_n na odpowiednie zmienne x_1, \dots, x_n (wolne ich wystąpienia) w termie M powstaje term oznaczany $M[x_1 := N_1, \dots, x_n := N_n]$ określony rekurencyjnie tak:

$$\begin{aligned}
x_i[x_1 := N_1, \dots, x_n := N_n] &= N_i \text{ dla } i \in \{1, \dots, n\}; \\
y[x_1 := N_1, \dots, x_n := N_n] &= y, \text{ o ile } x_i \neq y \text{ dla } i \in \{1, \dots, n\}; \\
(M_1 M_2)[x_1 := N_1, \dots, x_n := N_n] &= (M_1[x_1 := N_1, \dots, x_n := N_n]) \\
&\quad (M_2[x_1 := N_1, \dots, x_n := N_n]); \\
(\lambda y.P)[x_1 := N_1, \dots, x_n := N_n] &= \lambda z.(P[y := z] \\
&\quad [x_1 := N_1, \dots, x_n := N_n]), \\
&\quad \text{o ile } x_i \neq y \text{ oraz } z \text{ nie występuje} \\
&\quad \text{w } N_i \text{ i } P \text{ dla } i \in \{1, \dots, n\}; \\
(\lambda x_i.P)[x_1 := N_1, \dots, x_n := N_n] &= (\lambda x_i.P[x_1 := N_1, \dots, \\
&\quad x_{i-1} := N_{i-1}, \\
&\quad x_{i+1} := N_{i+1}, \dots, \\
&\quad x_n := N_n]).
\end{aligned}$$

Zwykle podstawienie termów N_1, \dots, N_n na odpowiednie zmienne z zestawu x_1, \dots, x_n oznaczamy jako $[x_1 := N_1, \dots, x_n := N_n]$. Zauważmy, że tak określone podstawienia stanowią funkcje częściowe ze zbioru zmiennych V do zbioru termów Λ . Takie funkcje częściowe będziemy oznaczali za pomocą liter S, T, \dots . W związku z tym dla podstawień ma sens pojęcie dziedziny. Dziedzinę podstawienia S będziemy oznaczali jako $\text{Dom}(S)$.

Czasami stosuje się odrobinę inne podejście do rachunku λ . Zakłada się, że pewne zmienne wolne mają status stałych — obiektów, na które nie działają podstawienia. Efekt niemożności podstawienia czegokolwiek na stałe można uzyskać łatwo przy oryginalnym podejściu, pisząc na początku termu λ -abstrakcje po wszystkich symbolach oznaczających stałe. Oczywiście nie będziemy tutaj stosowali tego typu „ozdobników” notacyjnych. W tym momencie powinno być jasne, jak zdefiniować taki rachunek ze stałymi, w związku z tym nie będziemy tutaj wprowadzali formalnie tej wersji rachunku λ .

Termy powyższe stanowią dziedzinę systemu redukcyjnego. Jego relacja redukcji określona jest jak następuje

Definicja 1.1.8 (β -redukcja)

Powiemy, że $M \rightarrow_\beta N$, gdy

- $M \equiv_\alpha (\lambda x.M_1)M_2$ i $N \equiv_\alpha M_1[x := M_2]$;
- $M \equiv_\alpha M_1M_2$, $M_1 \rightarrow_\beta N_1$ i $N \equiv_\alpha N_1M_2$;
- $M \equiv_\alpha M_1M_2$, $M_2 \rightarrow_\beta N_2$ i $N \equiv_\alpha M_1N_2$;
- $M \equiv_\alpha \lambda x.M_1$, $M_1 \rightarrow_\beta N_1$ i $N \equiv_\alpha \lambda x.N_1$.

Domknięcie zwrotno-przechodnie relacji \rightarrow_β zapisujemy symbolem \rightarrow_β , zaś najmniejszą relację równoważności zawierającą \rightarrow_β oznaczamy za pomocą symbolu $=_\beta$.

Zauważmy, że istotnym elementem tej definicji jest stwierdzenie, że

$$(\lambda x.M_1)M_2 \rightarrow_\beta M_1[x := M_2],$$

natomiast korzystając z pozostałych punktów tej definicji, można wykonać redukcję termu, gdy przedstawiona przed chwilą sytuacja pojawi się gdzieś w podtermie.

Aby precyzyjnie wyrazić powyżej wyrażoną intuicję konieczne jest kilka definicji. Definicje te mają charakter ogólny i dotyczą termów dowolnego języka.

Definicja 1.1.9 (kontekst)

Powiemy, że term M ze zmienną wolną \square jest *kontekstem*, gdy \square ma dokładnie jedno wystąpienie w M .

Definicja 1.1.10 (wstawienie do kontekstu)

Wynik wstawienia termu M do kontekstu N (oznaczany $M[N]$) definiujemy rekurencyjnie tak:

- $\square[N] = N$;
- $x[N] = x$;
- $M_1M_2[N] = M_1[N]M_2[N]$;
- $(\lambda x.M)[N] = \lambda x.(M[N])$.

Zauważmy, że w powyższej definicji nie dokonujemy przemianowania zmiennych.

Definicja 1.1.11 (relacja zgodna)

Powiemy, że relacja R jest zgodna na zbiorze termów T , gdy dla każdego kontekstu M w tym zbiorze, jeśli M_1RM_2 , to także $M[M_1]RM[M_2]$.

Możemy teraz powiedzieć, że relacja \rightarrow_β jest najmniejszą relacją zgodną spełniającą warunek:

$$(\lambda x.M_1)M_2 \rightarrow_\beta M_1[x := M_2].$$

Podobnej konstrukcji będziemy jeszcze używać w przyszłości dla określenia β -redukcji dla różnych rachunków lambda.

Zdefiniowana tutaj relacja \rightarrow_β ma własność Churcha-Rossera (zob. twierdzenie 2.3.7 w [Bar92]), tzn. dla dowolnego termu M_1 oraz termów M_2, M_3 takich, że $M_1 \rightarrow_\beta M_2$ oraz $M_1 \rightarrow_\beta M_3$, istnieje term M_4 , taki że $M_2 \rightarrow_\beta M_4$ oraz $M_3 \rightarrow_\beta M_4$.

1.2 Rachunki lambda z typami

Na potrzeby niniejszej pracy zdefiniujemy trzy rachunki lambda z typami. Będą to: $\lambda \rightarrow$, $\lambda 2$ oraz λP . Podamy definicje tych rachunków w wersji Curry’ego i Churcha.

1.2.1 Rachunki lambda w wersji Curry’ego

Zacniemy od wersji Curry’ego. Termy wszystkich wspomnianych rachunków mają w tej wersji tę samą składnię i jest to opisana już składnia beztypowego rachunku lambda. Różnice pojawiają się dopiero w składni typów.

Definicja 1.2.1 (typy rachunku $\lambda \rightarrow$ -Curry)

Zbiór typów rachunku lambda z typami prostymi, oznaczany za pomocą $\text{Typy}(\lambda \rightarrow)$, definiujemy przy użyciu następującej gramatyki

$$\mathbb{T} ::= \mathbb{V} \mid \mathbb{T} \rightarrow \mathbb{T},$$

gdzie \mathbb{T} jest skrótem dla $\text{Typy}(\lambda \rightarrow)$, zaś \mathbb{V} stanowi nieskończony zbiór zmiennych typowych.

W rachunku lambda z typami prostymi typ postaci $\sigma \rightarrow \tau$ pozwala na wyrażanie zależności: „jeśli zostanie podany argument typu σ , to dostaniemy wynik typu τ ”.

Definicja 1.2.2 (typy rachunku $\lambda 2$ -Curry)

Zbiór typów polimorficznego rachunku lambda, oznaczany przez $\text{Typy}(\lambda 2)$, definiujemy za pomocą następującej gramatyki

$$\mathbb{T} ::= \mathbb{V} \mid \mathbb{T} \rightarrow \mathbb{T} \mid (\forall \mathbb{V})\mathbb{T},$$

gdzie \mathbb{T} jest skrótem dla $\text{Typy}(\lambda 2)$, zaś \mathbb{V} stanowi nieskończony zbiór zmiennych typowych.

Dla typów rachunku polimorficznego istnieje również pojęcie zmiennej związanej i wolnej, w związku z czym można, podobnie jak dla λ -termów

wprowadzić relację α -konwersji. Relacja ta podobnie jak w przypadku termów utożsamia typy różniące się tylko nazwami zmiennych związanych. Ze względu na podobieństwo tego pojęcia do pojęcia stosowanego w przypadku λ -termów (a także logiki pierwszego rzędu) nie wprowadzamy tego pojęcia *explicite*.

W polimorficznym rachunku lambda typ postaci $\forall\alpha\sigma$ pozwala na wyrażanie zależności polimorficznej: „dla każdego typu (wstawianego na α) możemy skonstruować typ na podstawie schematu σ ”. Pozwala to między innymi na traktowanie termu $\lambda x.x$ (dla podanego argumentu dajemy w wyniku właśnie ten argument) uniwersalnie — ten przepis na obliczenie działa dla dowolnych danych wejściowych.

Definicja 1.2.3 (typy rachunku λP -Curry)

Zbiór typów rachunku λP , oznaczany przez $\text{Typy}(\lambda P)$, definiujemy za pomocą następującej gramatyki

$$\begin{aligned} \mathbb{T} &::= \mathbb{V} \mid (\forall V : \mathbb{T})\mathbb{T} \mid (\mathbb{T}(\Lambda)), \\ \mathbb{K} &::= * \mid (\Pi V : \mathbb{T})\mathbb{K}, \end{aligned}$$

gdzie \mathbb{T} jest skrótem dla $\text{Typy}(\lambda P)$, symbol \mathbb{K} oznacza *rodzaje* rachunku λP , zaś \mathbb{V} stanowi nieskończony zbiór zmiennych typowych, a V stanowi nieskończony zbiór zmiennych przedmiotowych (zbiór V z definicji 1.1.1). Symbole Π i \forall wiążą umieszczone za sobą zmienne.

Podobnie jak w $\lambda 2$ typy rachunku λP będą utożsamiane zgodnie z regułą α -konwersji, która utożsamia termy i typy różniące się tylko nazwami zmiennych związanych.

Typy będą oznaczane za pomocą liter σ, τ, \dots , zmienne typowe oznaczane są za pomocą liter α, β, \dots , wreszcie rodzaje oznaczane są za pomocą liter κ, μ, \dots . Dla typów rachunku λP postaci $(\forall x : \sigma)\tau$, gdzie x nie występuje wolno w τ , przyjmujemy konwencję zapisu jako $\sigma \rightarrow \tau$.

Z rodzajami i typami postaci $(\Pi x : \sigma)\kappa$ lub $(\forall x : \sigma)\tau$ związana jest pewna intuicja teoriomnogościowa. Mianowicie mogą one być rozumiane jak produkt rodziny zbiorów.

Dla rodziny zbiorów $\{A_t\}_{t \in T}$ (tutaj T odpowiada σ) możemy określić produkt:

$$\prod_{t \in T} A_t = \{f \in (\bigcup_{t \in T} A_t)^T \mid f(t) \in A_t \text{ dla wszystkich } t \in T\}.$$

Elementami produktu są funkcje, których wartości dla danego $t \in T$ należą do zbioru A_t , przy tym zbiory A_t mogą być różne dla różnych t . Można więc powiedzieć, że przeciwdziedzina funkcji f należącej do produktu nie jest ustalona, ale *zależna* od argumentu t . Podobnie, jeśli M jest termem typu $(\forall x : \sigma)\tau$, to typ aplikacji Mx *zależy* od typu argumentu x . Głębsze

wyjaśnienia można znaleźć w pracy [HHP93]. Jeśli wszystkie A_t są równe, to uzyskujemy równość

$$\prod_{t \in T} A_t = A_{t_0}^T$$

dla jakiegoś ustalonego $t_0 \in T$, która usprawiedliwia przyjętą przez nas konwencję traktowania pewnych typów jak typów funkcyjnych.

Patrząc na to od strony obliczeniowej, typy te pozwalają na wprowadzanie zależności typu od termu. Zwykle w literaturze stosuje się przykład z typem $(\forall x : \mathbb{N}) \text{list}(x)$ (gdzie \mathbb{N} i list należy traktować jako symbole wprowadzone w odpowiednim kontekście λP), który oznacza typ list parametryzowanych liczbami naturalnymi. Można sobie wyobrazić oczywiście wiele takich typów, np.: listy o wartościach nie przekraczających n , listy o elementach równych zawsze n itp. Czasami, żeby podkreślić ten właśnie aspekt rachunku λP , mówi się, że podstawowym elementem tego rachunku są konstruktory typów, a typy to taka specjalna kategoria konstruktorów, które nie mogą być zaaplikowane do termów.

Przy określaniu rachunku lambda z typami konieczne jest podanie definicji kontekstu.

Definicja 1.2.4 (konteksty w $\lambda \rightarrow$ -Curry i $\lambda 2$ -Curry)

Kontekstem w $\lambda \rightarrow$ ($\lambda 2$) nazwiemy skończony zbiór par $\langle x, \tau \rangle$, gdzie x jest zmienną przedmiotową, a τ typem, przy czym w zbiorze tym żadna zmienna przedmiotowa się nie powtarza. Parę złożoną ze zmiennej x i typu σ będziemy oznaczali przez $x : \sigma$. Konteksty będziemy oznaczali literami typu Γ, Δ, \dots . Wyrażenie $\Gamma, x : \sigma$ oznacza $\Gamma' \cup \{x : \sigma\}$, gdzie $\Gamma' = \Gamma \setminus \{x : \tau \mid \tau \text{ jest typem}\}$.

$$\begin{aligned} & \text{(var)} \quad \Gamma, x : \tau \vdash_{\lambda \rightarrow} x : \tau \quad (x \notin \text{Dom}(\Gamma)); \\ & \text{(app)} \quad \frac{\Gamma \vdash_{\lambda \rightarrow} M : \sigma \rightarrow \tau \quad \Gamma \vdash_{\lambda \rightarrow} N : \sigma}{\Gamma \vdash_{\lambda \rightarrow} MN : \tau}; \quad \text{(abs)} \quad \frac{\Gamma, x : \sigma \vdash_{\lambda \rightarrow} M : \tau}{\Gamma \vdash_{\lambda \rightarrow} \lambda x.M : \sigma \rightarrow \tau}. \end{aligned}$$

Rysunek 1.1: Reguły przypisywania typów w $\lambda \rightarrow$ -Curry

Dla systemu λP konteksty są bardziej skomplikowane:

Definicja 1.2.5 (konteksty w λP -Curry)

Kontekstem w λP nazwiemy skończony ciąg składający się z par $\langle x, \tau \rangle$ lub $\langle \alpha, \kappa \rangle$, gdzie x jest zmienną przedmiotową, τ typem, α zmienną typową, a κ rodzajem. Parę złożoną ze zmiennej x i typu σ będziemy oznaczali przez $x : \sigma$, podobnie dla zmiennej typowej α i rodzaju κ mamy oznaczenie

$\alpha : \kappa$. Konteksty będziemy oznaczali literami typu Γ, Δ, \dots . Wyrażenie $\Gamma, x : \sigma$ oznacza $\Gamma \cdot (x : \sigma)$, analogicznie dla zmiennej typowej i rodzaju (znak \cdot oznacza konkatenację ciągów). Pusty kontekst oznaczać będziemy przez \emptyset .

Dla każdego kontekstu Γ zbiór rzutowań jego elementów na pierwszą współrzedną (zbiór zmiennych tam deklarowanych) oznaczamy przez $\text{Dom}(\Gamma)$.

Zasadniczą część definicji rachunku lambda z typami stanowi określenie reguł przypisywania typów.

Definicja 1.2.6 (przypisywanie typów w $\lambda \rightarrow$ -Curry)

Dla kontekstu Γ , termu M i typu σ będziemy mówić, że *w kontekście Γ term M ma typ σ* , oznaczenie $\Gamma \vdash_{\lambda \rightarrow} M : \sigma$, gdy można taki osąd wyprowadzić używając reguł z rys. 1.1.

$$\begin{array}{c}
 (\text{var}) \quad \Gamma, x : \tau \vdash_{\lambda 2} x : \tau \quad (x \notin \text{Dom}(\Gamma)); \\
 (\text{app}) \frac{\Gamma \vdash_{\lambda 2} M : \sigma \rightarrow \tau \quad \Gamma \vdash_{\lambda 2} N : \sigma}{\Gamma \vdash_{\lambda 2} MN : \tau}; \quad (\text{abs}) \frac{\Gamma, x : \sigma \vdash_{\lambda 2} M : \tau}{\Gamma \vdash_{\lambda 2} \lambda x.M : \sigma \rightarrow \tau}; \\
 (\text{inst}) \frac{\Gamma \vdash_{\lambda 2} M : (\forall \alpha. \sigma)}{\Gamma \vdash_{\lambda 2} M : (\sigma[\alpha := \tau])}; \\
 (\text{gen}) \frac{\Gamma \vdash_{\lambda 2} M : \sigma}{\Gamma \vdash_{\lambda 2} M : (\forall \alpha. \sigma)}, \quad \alpha \notin FV(\Gamma).
 \end{array}$$

Rysunek 1.2: Reguły przypisywania typów w $\lambda 2$ -Curry

Definicja 1.2.7 (przypisywanie typów w $\lambda 2$ -Curry)

Dla kontekstu Γ , termu M i typu σ będziemy mówić, że *w kontekście Γ term M ma typ σ* , oznaczenie $\Gamma \vdash_{\lambda 2} M : \sigma$, gdy można taki osąd wyprowadzić używając reguł z rys. 1.2.

Definicja 1.2.8 (przypisywanie typów w λP -Curry)

Dla kontekstu Γ , termu M i typu σ będziemy mówić, że *w kontekście Γ term M ma typ σ* , oznaczenie $\Gamma \vdash_{\lambda P} M : \sigma$, gdy można taki osąd wyprowadzić używając reguł z rys. 1.3.

Przedstawimy teraz kilka tradycyjnych własności, które mają rachunki lambda z typami. Ich dowody dla $\lambda \rightarrow$, $\lambda 2$ oraz λP można znaleźć w [Bar92] oraz [HHP93].

$$\begin{array}{c}
\text{(type)} \vdash_{\lambda P} * : \square \qquad \text{(kd-abs)} \frac{\Gamma, x : \tau \vdash_{\lambda P} \kappa : \square}{\Gamma \vdash_{\lambda P} (\Pi x : \tau) \kappa : \square} \\
\\
\text{(kd-var)} \frac{\Gamma \vdash_{\lambda P} \kappa : \square}{\Gamma, \alpha : \kappa \vdash_{\lambda P} \alpha : \kappa} (\alpha \notin \text{Dom}(\Gamma)) \\
\\
\text{(kd-app)} \frac{\Gamma \vdash_{\lambda P} \phi : (\Pi x : \tau) \kappa \quad \Gamma \vdash_{\lambda P} M : \tau}{\Gamma \vdash_{\lambda P} \phi M : \kappa[x := M]} \\
\\
\text{(typ-abs)} \frac{\Gamma, x : \tau \vdash_{\lambda P} \sigma : *}{\Gamma \vdash_{\lambda P} (\forall x : \tau) \sigma : *} \qquad \text{(var)} \frac{\Gamma \vdash_{\lambda P} \tau : *}{\Gamma, x : \tau \vdash_{\lambda P} x : \tau} (x \notin \text{Dom}(\Gamma)) \\
\\
\text{(app)} \frac{\Gamma \vdash_{\lambda P} N : (\forall x : \tau) \sigma \quad \Gamma \vdash_{\lambda P} M : \tau}{\Gamma \vdash_{\lambda P} NM : \sigma[x := M]} \qquad \text{(abs)} \frac{\Gamma, x : \tau \vdash_{\lambda P} M : \sigma}{\Gamma \vdash_{\lambda P} \lambda x. M : (\forall x : \tau) \sigma} \\
\\
\text{(trm-kd)} \frac{\Gamma \vdash_{\lambda P} \tau : * \quad \Gamma \vdash_{\lambda P} \kappa : \square}{\Gamma, x : \tau \vdash_{\lambda P} \kappa : \square} (x \notin \text{Dom}(\Gamma)) \\
\\
\text{(typ-kd)} \frac{\Gamma \vdash_{\lambda P} \kappa : \square \quad \Gamma \vdash_{\lambda P} \kappa' : \square}{\Gamma, \alpha : \kappa \vdash_{\lambda P} \kappa' : \square} (\alpha \notin \text{Dom}(\Gamma)) \\
\\
\text{(trm-ty)} \frac{\Gamma \vdash_{\lambda P} \tau : * \quad \Gamma \vdash_{\lambda P} \phi : \kappa}{\Gamma, x : \tau \vdash_{\lambda P} \phi : \kappa} (x \notin \text{Dom}(\Gamma)) \\
\\
\text{(typ-ty)} \frac{\Gamma \vdash_{\lambda P} \kappa : \square \quad \Gamma \vdash_{\lambda P} \phi : \kappa'}{\Gamma, \alpha : \kappa \vdash_{\lambda P} \phi : \kappa'} (\alpha \notin \text{Dom}(\Gamma)) \\
\\
\text{(trm-trm)} \frac{\Gamma \vdash_{\lambda P} \tau : * \quad \Gamma \vdash_{\lambda P} M : \sigma}{\Gamma, x : \tau \vdash_{\lambda P} M : \sigma} (x \notin \text{Dom}(\Gamma)) \\
\\
\text{(typ-trm)} \frac{\Gamma \vdash_{\lambda P} \kappa : \square \quad \Gamma \vdash_{\lambda P} M : \sigma}{\Gamma, \alpha : \kappa \vdash_{\lambda P} M : \sigma} (\alpha \notin \text{Dom}(\Gamma)) \\
\\
\text{(kd-conv)} \frac{\Gamma \vdash_{\lambda P} \phi : \kappa \quad \kappa =_{\beta} \kappa'}{\Gamma \vdash_{\lambda P} \phi : \kappa'} \qquad \text{(typ-conv)} \frac{\Gamma \vdash_{\lambda P} M : \sigma \quad \sigma =_{\beta} \sigma'}{\Gamma \vdash_{\lambda P} M : \sigma'}
\end{array}$$

Rysunek 1.3: Reguły przypisywania typów w λP -Curry

Fakt 1.2.9 (silna normalizacja)

Relacja β -redukcji ma dla wszystkich trzech opisanych tutaj rachunków interesującą własność silnej normalizacji (zob. twierdzenie 5.3.33 w [Bar92]), tzn. dla każdego termu M , który w jakimś kontekście jest typowalny, wszystkie ciągi redukcji są skończone.

Fakt 1.2.10 (własność Churcha-Rossera)

Relacja β -redukcji ma własność Churcha-Rossera, czyli dla każdej trójki termów M_1, M_2, M_3 , takiej że $M_1 \rightarrow_\beta M_2$ oraz $M_1 \rightarrow_\beta M_3$, istnieje term M_4 , taki że $M_2 \rightarrow_\beta M_4$ oraz $M_3 \rightarrow_\beta M_4$.

Powyższe dwie własności na mocy lematu Newmana jednoznacznie definiują postać normalną β -redukcji.

Fakt 1.2.11 (postać normalna)

Każdy term M ma jedyną postać normalną tzn. taki term M' , że $M \rightarrow_\beta M'$ oraz nie istnieje $M'' \neq M'$, dla którego $M' \rightarrow_\beta M''$.

Dla danego termu M jego postać normalną oznaczamy przez $\text{NF}(M)$.

Jeszcze jedną ciekawą własnością jest własność *zachowywania typu* (ang. *subject reduction*):

Fakt 1.2.12 (zachowywanie typu)

Wymienione tutaj rachunki λ mają własność zachowywania typu, tzn. dla każdego termu M , jeśli wyprowadzalny jest osąd $\Gamma \vdash M : \tau$ oraz $M \rightarrow_\beta M'$, to wyprowadzalny jest też osąd $\Gamma \vdash M' : \tau$.

1.2.2 Rachunki lambda w wersji Churcha

Obecnie opiszemy wspomniane już rachunki w wersji Churcha. W wersji Churcha termy mają bardziej skomplikowaną składnię, której opis musi w niektórych przypadkach być przeplatany z definicją typu i kontekstu.

Definicja 1.2.13 (język $\lambda \rightarrow$ -Church)

Zbiór *pseudotermów* $\lambda \rightarrow$ -Church opisany jest za pomocą następującej gramatyki:

$$\Lambda_{\mathbb{T}} ::= V \mid (\Lambda_{\mathbb{T}} \Lambda_{\mathbb{T}}) \mid (\lambda V : \mathbb{T} \Lambda_{\mathbb{T}}),$$

gdzie \mathbb{T} jest zbiorem typów rachunku $\lambda \rightarrow$ zdefiniowanym w definicji 1.2.1.

Definicja 1.2.14 (język $\lambda 2$ -Church)

Zbiór *pseudotermów* rachunku $\lambda 2$ -Church opisany jest za pomocą następującej gramatyki:

$$\Lambda_{\mathbb{T}} ::= V \mid (\Lambda_{\mathbb{T}} \Lambda_{\mathbb{T}}) \mid (\Lambda_{\mathbb{T}} \mathbb{T}) \mid (\lambda V : \mathbb{T} \Lambda_{\mathbb{T}}) \mid (\lambda V \Lambda_{\mathbb{T}}),$$

gdzie \mathbb{T} jest zbiorem typów rachunku $\lambda 2$ zdefiniowanym w definicji 1.2.2, zaś \mathbb{V} jest wymienionym w tej samej definicji zbiorem zmiennych typowych.

Najbardziej skomplikowana sytuacja jest w przypadku rachunku λP .

Definicja 1.2.15 (język λP -Church)

Zbiór *pseudotermów* rachunku λP -Church definiowany jest jednocześnie ze zbiorem typów i rodzajów rachunku λP -Church. Zbiory te opisane są za pomocą następującej gramatyki:

$$\begin{aligned} \mathbb{T} &::= \mathbb{V} \mid (\forall V : \mathbb{T})\mathbb{T} \mid (\mathbb{T} \ \Lambda_{\mathbb{T}}) \\ \mathbb{K} &::= * \mid (\Pi V : \mathbb{T})\mathbb{K} \\ \Lambda_{\mathbb{T}} &::= V \mid (\Lambda_{\mathbb{T}}\Lambda_{\mathbb{T}}) \mid (\lambda V : \mathbb{T}.\Lambda_{\mathbb{T}}). \end{aligned}$$

Wyrażenia wyprowadzane za pomocą reguły \mathbb{T} należą do zbioru $\text{Typy}_{\text{Ch}}(\lambda P)$ i nazywane są *pseudotypami*, zaś wyrażenia wyprowadzane za pomocą reguły \mathbb{K} należą do zbioru $\text{Rodzaje}_{\text{Ch}}(\lambda P)$ i nazywane są *pseudorodzajami*.

Czasami, żeby rozróżnić zbiory termów w $\lambda \rightarrow$, $\lambda 2$ i λP , dodajemy odpowiednie adnotacje do $\Lambda_{\mathbb{T}}$, tj. piszemy odpowiednio $\Lambda_{\mathbb{T},\lambda \rightarrow}$, $\Lambda_{\mathbb{T},\lambda 2}$ i $\Lambda_{\mathbb{T},\lambda P}$.

Definicja 1.2.16 (konteksty w systemach Churcha)

Konteksty dla rachunków $\lambda \rightarrow$ i $\lambda 2$ w wersji Churcha są zdefiniowane tak jak w definicji 1.2.4. W kontekście rachunku λP w wersji Churcha obiekty określone w definicji 1.2.5 będziemy nazywali *pseudokontekstami*.

Powyższe definicje pozwalają dopiero określić, jak wyglądają wyrażenia, które stanowią elementy osądów wyprowadzanych w rachunkach Churcha. O pseudotermie możemy powiedzieć, że jest termem rachunku Churcha, dopiero, gdy uda się dla niego wyprowadzić jakiś osąd.

Definicja 1.2.17 (reguły wyprowadzania dla $\lambda \rightarrow$ -Church)

Dla kontekstu Γ , pseudotermu M i typu σ będziemy mówić, że *w kontekście Γ term M ma typ σ* , oznaczenie $\Gamma \vdash_{\lambda \rightarrow} M : \sigma$, gdy można taki osąd wyprowadzić używając reguł z rys. 1.4.

$$\begin{aligned} &(\text{var}) \quad \Gamma, x : \tau \vdash_{\lambda \rightarrow} x : \tau \quad (x \notin \text{Dom}(\Gamma)); \\ &(\text{app}) \quad \frac{\Gamma \vdash_{\lambda \rightarrow} M : \sigma \rightarrow \tau \quad \Gamma \vdash_{\lambda \rightarrow} N : \sigma}{\Gamma \vdash_{\lambda \rightarrow} MN : \tau}; \quad (\text{abs}) \quad \frac{\Gamma, x : \sigma \vdash_{\lambda \rightarrow} M : \tau}{\Gamma \vdash_{\lambda \rightarrow} \lambda x : \sigma. M : \sigma \rightarrow \tau}. \end{aligned}$$

Rysunek 1.4: Reguły przypisywania typów w $\lambda \rightarrow$ -Church

Systemy wyprowadzania typów dla $\lambda \rightarrow$ -Church i dla $\lambda \rightarrow$ -Curry różnią się między sobą tylko w regule (abs), gdzie w wersji Churcha dodana jest adnotacja typowa do zmiennej pod λ -abstrakcją.

Definicja 1.2.18 (reguły wyprowadzania dla λ_2 -Church)

Dla kontekstu Γ , pseudoteremu M i typu σ będziemy mówić, że w *kontekście* Γ term M ma typ σ , oznaczenie $\Gamma \vdash_{\lambda_2} M : \sigma$, gdy można taki osąd wyprowadzić używając reguł z rys. 1.5.

$$\begin{array}{c}
(\text{var}) \quad \Gamma, x : \tau \vdash_{\lambda_2} x : \tau \quad (x \notin \text{Dom}(\Gamma)); \\
(\text{app}) \frac{\Gamma \vdash_{\lambda_2} M : \sigma \rightarrow \tau \quad \Gamma \vdash_{\lambda_2} N : \sigma}{\Gamma \vdash_{\lambda_2} MN : \tau}; \quad (\text{abs}) \frac{\Gamma, x : \sigma \vdash_{\lambda_2} M : \tau}{\Gamma \vdash_{\lambda_2} \lambda x : \sigma. M : \sigma \rightarrow \tau}; \\
(\text{inst}) \frac{\Gamma \vdash_{\lambda_2} M : (\forall \alpha. \sigma)}{\Gamma \vdash_{\lambda_2} M \tau : (\sigma[\alpha := \tau])}; \\
(\text{gen}) \frac{\Gamma \vdash_{\lambda_2} M : \sigma}{\Gamma \vdash_{\lambda_2} (\Lambda \alpha. M) : (\forall \alpha. \sigma)}, \quad \alpha \notin FV(\Gamma).
\end{array}$$

Rysunek 1.5: Reguły przypisywania typów w λ_2 -Church

W przypadku systemu λ_2 różnice między rachunkiem Churcha i Curry'ego są bardziej zauważalne. Oprócz różnicy w regule (abs), gdzie rachunek Churcha dodaje adnotację analogicznie jak w przypadku rachunku $\lambda \rightarrow$, dochodzą jeszcze różnice w regułach (inst) i (gen). W tej pierwszej mamy dodatkowo obecną aplikację typową, w tej drugiej zaś Λ -abstrakcję zmiennej typowej.

Definicja 1.2.19 (przypisywanie typów w λP -Church)

Dla pseudokontekstu Γ , pseudoteremu M i pseudotypu σ będziemy mówić, że w *kontekście* Γ term M ma typ σ , oznaczenie $\Gamma \vdash_{\lambda P} M : \sigma$, gdy można taki osąd wyprowadzić używając reguł z rys. 1.6.

Dla pseudokontekstu Γ , pseudotypu τ i pseudorodzaju κ będziemy mówić, że w *kontekście* Γ typ τ ma rodzaj κ , oznaczenie $\Gamma \vdash_{\lambda P} \tau : \kappa$, gdy można taki osąd wyprowadzić, używając reguł z rys. 1.6.

Zauważmy, że powyższa definicja określa, kiedy o pseudokontekście można mówić, że jest kontekstem — ma to miejsce, gdy pseudokontekst występuje w jakimś wyprowadzeniu zgodnym z regułami λP . Warto jeszcze nadmienić, że reguła (typ-conv) z rys. 1.6 korzysta z pojęcia relacji $=_{\beta}$, którą określamy dopiero w definicji 1.2.28. Przenieśliśmy tę definicję tam, by ułatwić porównanie β -redukcji w poszczególnych rachunkach.

Tutaj różnica jest pozornie bardzo nieznacząca i sprowadza się do dodania adnotacji typowej do zmiennej przedmiotowej w regule (abs). W tym rachunku występuje jednak pewna dodatkowa różnica między wersją Churcha

$$\begin{array}{c}
\text{(type)} \frac{}{\Gamma \vdash_{\lambda P} * : \square} \qquad \text{(kd-abs)} \frac{\Gamma, x : \tau \vdash_{\lambda P} \kappa : \square}{\Gamma \vdash_{\lambda P} (\Pi x : \tau) \kappa : \square} \\
\\
\text{(kd-var)} \frac{\Gamma \vdash_{\lambda P} \kappa : \square}{\Gamma, \alpha : \kappa \vdash_{\lambda P} \alpha : \kappa} (\alpha \notin \text{Dom}(\Gamma)) \\
\\
\text{(kd-app)} \frac{\Gamma \vdash_{\lambda P} \phi : (\Pi x : \tau) \kappa \quad \Gamma \vdash_{\lambda P} M : \tau}{\Gamma \vdash_{\lambda P} \phi M : \kappa[x := M]} \\
\\
\text{(typ-abs)} \frac{\Gamma, x : \tau \vdash_{\lambda P} \sigma : *}{\Gamma \vdash_{\lambda P} (\forall x : \tau) \sigma : *} \qquad \text{(var)} \frac{\Gamma \vdash_{\lambda P} \tau : *}{\Gamma, x : \tau \vdash_{\lambda P} x : \tau} (x \notin \text{Dom}(\Gamma)) \\
\\
\text{(app)} \frac{\Gamma \vdash_{\lambda P} N : (\forall x : \tau) \sigma \quad \Gamma \vdash_{\lambda P} M : \tau}{\Gamma \vdash_{\lambda P} NM : \sigma[x := M]} \qquad \text{(abs)} \frac{\Gamma, x : \tau \vdash_{\lambda P} M : \sigma}{\Gamma \vdash_{\lambda P} \lambda x : \tau. M : (\forall x : \tau) \sigma} \\
\\
\text{(trm-kd)} \frac{\Gamma \vdash_{\lambda P} \tau : * \quad \Gamma \vdash_{\lambda P} \kappa : \square}{\Gamma, x : \tau \vdash_{\lambda P} \kappa : \square} (x \notin \text{Dom}(\Gamma)) \\
\\
\text{(typ-kd)} \frac{\Gamma \vdash_{\lambda P} \kappa : \square \quad \Gamma \vdash_{\lambda P} \kappa' : \square}{\Gamma, \alpha : \kappa \vdash_{\lambda P} \kappa' : \square} (\alpha \notin \text{Dom}(\Gamma)) \\
\\
\text{(trm-typ)} \frac{\Gamma \vdash_{\lambda P} \tau : * \quad \Gamma \vdash_{\lambda P} \phi : \kappa}{\Gamma, x : \tau \vdash_{\lambda P} \phi : \kappa} (x \notin \text{Dom}(\Gamma)) \\
\\
\text{(typ-typ)} \frac{\Gamma \vdash_{\lambda P} \kappa : \square \quad \Gamma \vdash_{\lambda P} \phi : \kappa'}{\Gamma, \alpha : \kappa \vdash_{\lambda P} \phi : \kappa'} (\alpha \notin \text{Dom}(\Gamma)) \\
\\
\text{(trm-trm)} \frac{\Gamma \vdash_{\lambda P} \tau : * \quad \Gamma \vdash_{\lambda P} M : \sigma}{\Gamma, x : \tau \vdash_{\lambda P} M : \sigma} (x \notin \text{Dom}(\Gamma)) \\
\\
\text{(typ-trm)} \frac{\Gamma \vdash_{\lambda P} \kappa : \square \quad \Gamma \vdash_{\lambda P} M : \sigma}{\Gamma, \alpha : \kappa \vdash_{\lambda P} M : \sigma} (\alpha \notin \text{Dom}(\Gamma)) \\
\\
\text{(kd-conv)} \frac{\Gamma \vdash_{\lambda P} \phi : \kappa \quad \kappa =_{\beta} \kappa'}{\Gamma \vdash_{\lambda P} \phi : \kappa'} \qquad \text{(typ-conv)} \frac{\Gamma \vdash_{\lambda P} M : \sigma \quad \sigma =_{\beta} \sigma'}{\Gamma \vdash_{\lambda P} M : \sigma'}
\end{array}$$

Rysunek 1.6: Reguły przypisywania typów w λP -Church

a Curry’ego, na którą warto zwrócić uwagę, a która nie występuje w przypadku $\lambda \rightarrow$ i $\lambda 2$. Mianowicie, ponieważ typy zależą tutaj od termów, a składnia termów jest zmieniona, to inna jest też definicja samych typów.

Możemy wreszcie określić zbiory legalnych termów dla omawianych tutaj rachunków:

Definicja 1.2.20 (termy w wersji Churcha)

Zbiór termów legalnych w jakimś kontekście rachunku $\lambda \rightarrow$ (lub rachunków $\lambda 2$, λP), oznaczany w niniejszej pracy przez $\text{Termy}_{\text{Ch}}(\lambda \rightarrow)$ (lub odpowiednio przez $\text{Termy}_{\text{Ch}}(\lambda 2)$, $\text{Termy}_{\text{Ch}}(\lambda P)$), definiujemy tak:

$$\text{Termy}_{\text{Ch}}(\lambda X) = \{M \in \Lambda_{\mathbb{T}, \lambda X} \mid \text{istnieją } \Gamma \text{ i } \tau, \text{ takie że } \Gamma \vdash_{\lambda X} M : \tau\},$$

gdzie X jest równe \rightarrow (lub odpowiednio $2, P$).

Między rachunkami w wersji Churcha i Curry’ego istnieje interesujący związek. Wyrażenie jego wymaga wprowadzenia pewnej dodatkowej definicji:

Definicja 1.2.21 (wycieranie typów)

Operację $|\cdot|$ *wycierania typów* przekształcającą termy w rachunku Churcha na termy rachunku beztypowego definiujemy

- dla rachunku $\lambda \rightarrow$ jako

$$\begin{aligned} |x| &= x \\ |MN| &= |M||N| \\ |\lambda x : \sigma.M| &= \lambda x. |M| \end{aligned}$$

- dla rachunku $\lambda 2$ jako

$$\begin{aligned} |x| &= x \\ |MN| &= |M||N| \\ |M\tau| &= |M| \\ |\lambda x : \sigma.M| &= \lambda x. |M| \\ |\Lambda \alpha.M| &= |M|, \end{aligned}$$

- zaś dla rachunku λP jako

$$\begin{aligned} |x| &= x \\ |MN| &= |M||N| \\ |\lambda x : \sigma.M| &= \lambda x. |M|. \end{aligned}$$

Powyższe definicje wycierania typów rozszerzamy na konteksty i typy (i rodzaje) w ten sposób, że dla rachunków $\lambda \rightarrow$ oraz $\lambda 2$ operacja ta na kontek-

stach jest identycznościowa, zaś dla λP jest określona jak następuje:

$$\begin{aligned}
|\alpha| &= \alpha \\
|(\forall x : \sigma)\tau| &= (\forall x : |\sigma|)|\tau| \\
|\sigma M| &= |\sigma||M| \\
|*| &= * \\
|(\Pi x : \sigma)\kappa| &= (\Pi x : |\sigma|)|\kappa| \\
|\Gamma, (\alpha : \kappa)| &= |\Gamma|, (\alpha : |\kappa|) \\
|\Gamma, (x : \sigma)| &= |\Gamma|, (x : |\sigma|)
\end{aligned}$$

Twierdzenie 1.2.22 (zależność między Curry i Church)

Dla każdego z rachunków $\lambda \rightarrow$, $\lambda 2$ i λP osąd $\Gamma \vdash M : \sigma$ jest wyprowadzalny w rachunku w wersji Churcha wtedy i tylko wtedy, gdy w rachunku w wersji Curry'ego wyprowadzalny jest osąd $|\Gamma| \vdash |M| : |\sigma|$.

Dowód:

Dowód można znaleźć na przykład w [BLRU97]. \square

Będziemy się tutaj posługiwać również pojęciem rzędu. Wprowadzamy go tutaj za [Hue76]:

Definicja 1.2.23 (rzęd typy)

Rząd typu prostego σ , oznaczany $\text{ord}(\sigma)$, jest liczbą naturalną określoną następującym wzorem indukcyjnym:

$$\begin{aligned}
\text{ord}(\alpha) &= 1 \quad \text{dla } \alpha \text{ atomowego;} \\
\text{ord}(\sigma \rightarrow \tau) &= \max(\text{ord}(\sigma) + 1, \tau).
\end{aligned}$$

Rzędem termu M w osądzie $\Gamma \vdash_{\lambda \rightarrow} M : \tau$ nazwiemy rząd typu τ . Zwykle opuszczamy informację o osądzie, jeśli z kontekstu wynika, o jaki osąd chodzi.

Pojęcie to pozwala na zdefiniowanie, czym jest rachunek λ (z typami prostymi) określonego rzędu:

Definicja 1.2.24 (rachunek λ rzędu n)

Rachunek λ rzędu n (z typami prostymi) w wersji Churcha określamy przez ograniczenie zbioru termów rachunku lambda z typami prostymi (w wersji Churcha) do tych, które są zbudowane przy użyciu zmiennych o typach rzędu co najwyżej n . Zbiór termów takiego rachunku oznaczamy przez $\text{Termy}_{\text{Ch}}(\lambda \rightarrow n)$. Wszystkie dopuszczalne wyprowadzenia i konteksty muszą składać się z tak ograniczonych termów.

Rachunek n -tego rzędu w wersji Curry'ego uzyskujemy z rachunku w wersji Churcha, stosując wycieranie typów. Zbiór termów rachunku Curry'ego rzędu n oznaczamy przez $\text{Termy}(\lambda \rightarrow n)$.

Uwaga 1.2.25 W powyższych definicjach dotyczących rachunku λ w wersji Churcha używaliśmy pojęć pseudotermy, pseudokontekstu, pseudotypu czy pseudorodzaju. W dalszej części pracy będziemy trzymali się mniej rygorystycznie rozgraniczenia między tymi pojęciami a pojęciami bez przedrostka pseudo- i będziemy ten przedrostek opuszczali.

β -redukcja Każdy z powyższych rachunków w wersji Churcha ma własną definicję β -redukcji, która jest niestety różna od definicji dla beztypowego rachunku λ . Trudno nawet zdefiniować taką redukcję w terminach redukcji w rachunku beztypowym i wycierania typów, gdyż w rachunkach Churcha mamy do czynienia z pewnymi konstrukcjami, które podlegają β -redukcji, ale nie widać ich po wytarciu adnotacji typowych (np. aplikacja typowa w rachunku $\lambda 2$ -Church).

Wraz z każdym pojęciem relacji \rightarrow_β wprowadzamy także domknięcie zwrotno-przechodnie oznaczane symbolem \rightarrow_β oraz najmniejszą relację równoważności zawierającą \rightarrow_β oznaczaną za pomocą symbolu $=_\beta$. Oto definicje poszczególnych redukcji:

Definicja 1.2.26 (β redukcja w $\lambda \rightarrow$)

Relację \rightarrow_β w $\lambda \rightarrow$ -Church określamy jako najmniejszą relację zgodną, taką że

$$(\lambda x : \sigma. M_1)M_2 \rightarrow_\beta M_1[x := M_2].$$

Definicja 1.2.27 (β redukcja w $\lambda 2$)

Relację \rightarrow_β w $\lambda 2$ -Church określamy jako najmniejszą relację zgodną, taką że

$$\begin{aligned} (\lambda x : \sigma. M_1)M_2 &\rightarrow_\beta M_1[x := M_2]; \\ (\Lambda \alpha. M_1)\sigma &\rightarrow_\beta M_1[\alpha := \sigma]. \end{aligned}$$

Definicja 1.2.28 (β redukcja w λP)

Relację \rightarrow_β w λP -Church określamy jako najmniejszą relację zgodną (na sumie zbiorów termów, typów i rodzajów), taką że

$$(\lambda x : \sigma. M_1)M_2 \rightarrow_\beta M_1[x := M_2];$$

Określone tutaj redukcje mają własność Churcha-Rossera (twierdzenie 2.3.7 w [Bar92]) oraz własność silnej normalizacji (twierdzenie 5.3.33 w [Bar92]). W związku z tym ostatnim dobrze określone jest pojęcie postaci normalnej termu, oznaczanej dla termu M przez $NF(M)$.

1.2.3 Problemy uzupełniania osądów

W związku z rachunkami λ z typami rozważane są różne ciekawe problemy decyzyjne. Problemy te dotyczą osądów, czyli wyrażeń postaci $\Gamma \vdash M : \sigma$. Otrzymujemy tutaj jako dane jakiś fragment (niekoniecznie właściwy) takiego osądu. Naszym zadaniem jest odpowiedzieć, czy istnieje takie uzupełnienie tego osądu, że wynik uzupełnienia daje się wyprowadzić w określonym systemie wyprowadzania. Każdy taki problem ma swoją wersję dla rachunku Churcha i Curry'ego.

Opiszemy tutaj główne spotykane w literaturze problemy oraz przedstawimy dotychczasowe wyniki ich dotyczące.

Problem 1.2.29 (kontekstowe sprawdzanie typów)

Dane: Kontekst Γ , term M i typ σ .

Pytanie: Czy osąd $\Gamma \vdash_{\bullet} M : \sigma$ jest wyprowadzalny w systemie \bullet ?

Problem 1.2.30 (kontekstowe wyprowadzanie typów)

Dane: Kontekst Γ i term M .

Pytanie: Czy istnieje taki typ σ , że osąd $\Gamma \vdash_{\bullet} M : \sigma$ jest wyprowadzalny w systemie \bullet ?

Problem 1.2.31 (kontekstowa inhabitacja)

Dane: Kontekst Γ i typ σ .

Pytanie: Czy istnieje taki term M , że osąd $\Gamma \vdash_{\bullet} M : \sigma$ jest wyprowadzalny w systemie \bullet ?

Problemy powyższe mają swoje wersje, w których w *Danych* opuszczony jest kontekst, przy czym pytanie zmodyfikowane jest tak, że zamiast kontekstu Γ mamy kontekst pusty. Takie wersje będziemy nazywali *ograniczonymi* wersjami problemu. I tak będziemy mieli do czynienia ze: *ograniczonym problemem sprawdzania typów*, *ograniczonym problemem wyprowadzania typów* oraz *ograniczonym problemem inhabitacji*.

Istnieje jeszcze jeden zestaw wersji powyższych problemów, gdzie zadawane pytanie jest zmodyfikowane w ten sposób, że w *Danych* opuszczony jest kontekst, a zadawane pytanie zmodyfikowane jest tak, że dodatkowo pytamy się o istnienie stosownego kontekstu Γ . Przy takich problemach stosujemy określenie pozbawione przymiotnika „kontekstowy” i tak mamy: *problem sprawdzania typów*, *problem wyprowadzania typów* oraz *problem inhabitacji*.

Zainteresowanie problemami *sprawdzania typów* oraz *wyprowadzania typów* dla rachunków termów Curry’ego (w wersji ograniczonej, kontekstowej i normalnej) związane jest z faktem, iż pojawiają się one naturalnie w badaniach nad językami programowania. Funkcyjne języki programowania takie jak ML [Mil78, Mil84] czy Haskell [HF92] korzystają w sposób intensywny tak z mechanizmów wymagających zastosowania algorytmów wyprowadzania typów, jak i ich sprawdzania. Normalnie programista wpisuje w takim języku λ -term, który przechodzi następnie przez moduł wyprowadzania typu. Jeśli przejście się powiedzie, otrzymujemy w wyniku typ oraz pewność, że wprowadzony program-term pozbawiony jest niektórych błędów programistycznych. Jeśli przejście się nie powiedzie, trzeba przepisać term na nowo z poprawkami. Algorytmy sprawdzania typów znajdują tutaj także swoje zastosowanie — użytkownik języka ma możliwość ręcznego wprowadzania typów, w związku z czym konieczne jest sprawdzanie poprawności wykonania tej czynności.

Problem *sprawdzania typów* dla termów Churcha (w wersji ograniczonej i kontekstowej) ma zastosowanie w systemach wspomagających dowodzenie

twierdzeń. W oprogramowaniu oferującym taką funkcjonalność (np. Lego [LP92], Coq [Hue96]), o ile mechanizm dowodzenia opiera się na systemach typów wyższych rzędów, konstruuje się właśnie λ -term, który ma stanowić dowód dla jakiegoś twierdzenia wyrażonego jako typ. Mając założenia zewnętrzne teorii, dowód oraz samo twierdzenie, musimy stwierdzić, czy te elementy stanowią spójną pod względem logicznym całość. Korzystamy tutaj ze znanego związku określanego jako izomorfizm Curry’ego-Howarda [How80]. Termy odpowiadają dowodom w pewnej logice, zaś typy — formułom tej logiki.

W pracy osoby dowodzącej twierdzenia często spotykana jest sytuacja, gdy osoba taka ma jakąś konstrukcję względnie dowód jakiegoś twierdzenia i chce się dowiedzieć, czy jej dowód jest poprawny, a jeśli nie, to jakie dodatkowe założenia są potrzebne, aby można było tą drogą je dowieść. Nawet informacja, że taki dowód, niezależnie od dodanych założeń, nie może dowodzić podanego twierdzenia jest bardzo cenna. Automatyczne uzyskiwanie tego typu informacji we wspomnianych systemach można byłoby zapewnić stosując algorytmy *kontekstowego sprawdzania typów* dla termów Churcha lub *wyprowadzania typów* (we wszystkich trzech wersjach) dla termów Churcha.

Zainteresowanie problemami *inhabitacji* wynika ze stosowalności odpowiednich algorytmów w systemach automatycznego wyprowadzania dowodów. Mając algorytm dla problemu inhabitacji, możemy na mocy izomorfizmu Curry’ego-Howarda automatycznie znajdować dowody (w przypadku, gdy mamy do czynienia z wersją Churcha) lub schematy dowodów (w przypadku, gdy mamy do czynienia z wersją Curry’ego). Ma tutaj zastosowanie wspomniany już izomorfizm Curry’ego-Howarda.

1.3 Logika pierwszego rzędu i przepisywanie termów

Będziemy w niniejszej pracy rozważali implikacyjno-universalny fragment intuicjonistycznej logiki pierwszego rzędu. W związku z tym wprowadzimy tutaj definicję tego formalizmu.

Definicja 1.3.1 (sygnatura)

Sygnaturą języka pierwszego rzędu nazwiemy dowolną parę rodzin rozłącznych zbiorów $\langle \{\Sigma_{rn}\}_{n \in \mathbb{N}}, \{\Sigma_{fn}\}_{n \in \mathbb{N}} \rangle$. Rodzinę Σ_r nazwiemy *zbiorem symboli relacyjnych*, zaś rodzinę Σ_f *zbiorem symboli funkcyjnych*. Dla każdego symbolu s indeks zbioru, do którego należy s jest nazywany *arnością* s . Zbiór symboli relacyjnych arności n oznaczamy Σ_{rn} , zaś symboli funkcyjnych Σ_{fn} . Rodzinę $\{\Sigma_{rn}\}_{n \in \mathbb{N}}$ będziemy oznaczali przez Σ_r , zaś rodzinę $\{\Sigma_{fn}\}_{n \in \mathbb{N}}$ przez Σ_f . Parę $\langle \Sigma_r, \Sigma_f \rangle$ będziemy oznaczać przez Σ .

W powyższej definicji pozwoliliśmy sobie na utożsamienie symboli relacyjnych o arności 0 z predykatami będącymi bezpośrednimi odpowiednikami zmiennych zdaniowych logiki zdaniowej, oraz na utożsamienie symboli funkcyjnych o arności 0 ze stałymi.

Definicja 1.3.2 (termy algebraiczne)

Zbiór *termów (algebraicznych)* nad sygnaturą Σ i ze zmiennymi ze zbioru X to najmniejszy zbiór $T_\Sigma(X)$, taki że

- $X \subseteq T_\Sigma(X)$;
- jeśli $f \in \Sigma_{fn}$ i $M_1, \dots, M_n \in T_\Sigma(X)$, to $fM_1 \dots M_n \in T_\Sigma(X)$.

Zbiór $T_\Sigma(\emptyset)$ będziemy oznaczali przez T_Σ .

W powyższej definicji zastosowaliśmy odrobinę niestandardową notację — termy oznaczamy dużymi literami z okolic M oraz aplikację symbolu funkcyjnego oznaczamy beznawiasowo (normalnie oznacza się to za pomocą wyrażenia: $f(M_1, \dots, M_n)$). Zrobiliśmy tak ze względu na fakt, że w niniejszej pracy działamy jednolicie w ramach rachunku λ .

Wprowadzimy tutaj kilka niskopoziomowych operacji na termach, które posłużą nam do zapisywania i udowodniania różnych potrzebnych w całej pracy faktów.

Definicja 1.3.3 (wskazywanie podtermu)

Niech słowo ω należy do \mathbb{N}^* . Dla termów M określimy rekurencyjnie operację *wskazywania podtermu* oraz własność *bycia ścieżką w M* . Wyrażenie $\omega(M) = N$ będziemy czytali jako ω *wskazuje na N w M* . Mamy

- $\varepsilon(M) = M$ i ε jest ścieżką w M ;
- niech f będzie symbolem o arności $n \geq 0$; jeśli ω jest ścieżką w M_i , gdzie $i \in A$, to $i\omega$ jest ścieżką w $fM_1 \dots M_n$ oraz $i\omega(fM_1 \dots M_n) = \omega(M_i)$.

(Symbol ε oznacza tutaj słowo puste.) Zbiór wszystkich ścieżek w M oznaczamy symbolem $\text{Sciezki}(M)$.

O ile to nie będzie wywoływało niejednoznaczności, zapisując wskazanie do podtermu, będziemy opuszczali nawias i pisali ωM zamiast $\omega(M)$. Długość ścieżki ω będziemy oznaczali przez $|\omega|$.

Ścieżki będziemy też stosowali do λ -termów. Przyjmować tutaj będziemy zasadę, że zmienne i stałe są traktowane jak symbole o arności 0 (niezależnie od ewentualnego ich typu). Term M_1M_2 jest traktowany tak, jakby występowało tutaj użycie dwuargumentowego symbolu (aplikacji) o argumentach M_1 i M_2 , zaś term $\lambda x.M$ jest traktowany jak użycie jednoargumentowego symbolu (abstrakcji) o argumentie M .

Ponieważ używane w typach symbole \rightarrow, \forall oraz Π można potraktować jako symbole funkcyjne (odpowiednio: dwuargumentowy, jednoargumentowy, jednoargumentowy), to nasza notacja ścieżkowa rozszerza się także na typy.

Definicja 1.3.4 (porządek na ścieżkach)

Na ścieżkach określimy jeszcze porządek częściowy \preceq tak, że $\omega_1 \preceq \omega_2$, gdy ω_1 jest końcowym fragmentem ω_2 (być może równym ω_2). Porządek ten będziemy nazywali *porządkiem sufikсовym*.

Definicja 1.3.5 (zastąpienie podtermu)

Niech $\omega \in A^*$. Zdefiniujemy rekurencyjnie *zastąpienie podtermu zaczepionego w wierzchołku ω termu M przez term N* (co będziemy zapisywać jako $M[\omega \leftarrow N]$) w sposób następujący:

- $M[\varepsilon \leftarrow N] = N$;
- $(fM_1 \dots M_n)[i\omega \leftarrow N] = fM_1 \dots (M_i[\omega \leftarrow N]) \dots M_n$.

Wprowadzimy teraz pojęcie formuły logiki pierwszego rzędu.

Definicja 1.3.6 (formuły)

Zbiór *formuł* nad sygnaturą Σ i ze zmiennymi ze zbioru X to najmniejszy zbiór $F_\Sigma(X)$ taki, że

- jeśli $P \in \Sigma_{rn}$ oraz $M_1, \dots, M_n \in T_\Sigma(X)$, to $PM_1 \dots M_n \in F_\Sigma(X)$;
- jeśli $\phi_i \in F_\Sigma(X)$ dla $i = 1, 2$, to $(\phi_1 \rightarrow \phi_2) \in F_\Sigma(X)$;
- jeśli $\phi \in F_\Sigma(X \cup \{x\})$, to $(\forall x)\phi \in F_\Sigma(X)$.

Zbiór $F_\Sigma(\emptyset)$ będziemy oznaczali przez F_Σ i każdy jego element będziemy nazywali *formułą zamkniętą*.

W naszej definicji formuł logiki pierwszego rzędu zrezygnowaliśmy podobnie jak w przypadku termów z notacji $P(M_1, \dots, M_n)$. W odróżnieniu od standardowego podejścia do logiki, rozważamy tutaj tylko jej fragment z implikacją i kwantyfikatorem generalizacji. Wiąże się to z faktem, że ten fragment logiki pierwszego rzędu da się bezpośrednio zakodować w rachunku λP (por. rozdział 10 w [SU98]).

Dla pełności prezentacji przedstawimy jeszcze system dowodzenia dla rozważanego przez nas fragmentu logiki pierwszego rzędu. System ten będzie odpowiadał intuicjonistycznemu podejściu do tej logiki.

Definicja 1.3.7 (system dowodzenia)

Dla kontekstu Γ i formuły ϕ będziemy mówić, że *w kontekście Γ formuła ϕ jest wyprowadzalna*, oznaczenie $\Gamma \vdash \phi$, gdy można taki osąd wyprowadzić używając reguł z rys. 1.7.

$$\begin{array}{c}
(\text{ax}) \Gamma, \varphi \vdash_{\forall} \varphi \\
(\rightarrow \text{I}) \frac{\Gamma, \varphi \vdash_{\forall} \psi}{\Gamma \vdash_{\forall} \varphi \rightarrow \psi}; \quad (\rightarrow \text{E}) \frac{\Gamma \vdash_{\forall} \varphi \rightarrow \psi \quad \Gamma \vdash_{\forall} \varphi}{\Gamma \vdash_{\forall} \psi} \\
(\forall \text{I}) \frac{\Gamma \vdash_{\forall} \varphi}{\Gamma \vdash_{\forall} \forall x \varphi} (x \notin FV(\Gamma)); \quad (\forall \text{E}) \frac{\Gamma \vdash_{\forall} \forall x \varphi}{\Gamma \vdash_{\forall} \varphi[x := M]}.
\end{array}$$

Rysunek 1.7: System naturalnej dedukcji dla logiki pierwszego rzędu

Na termach pierwszego rzędu określa się często relację przepisywania termów.

Definicja 1.3.8 (system przepisywania termów)

Systemem przepisywania termów nazwiemy dowolny zbiór P par uporządkowanych termów nad ustaloną sygnaturą Σ . Elementy zbioru P nazywamy *regułami* systemu P . Taki system przepisywania wyznacza relację \rightarrow_P na zbiorze T_{Σ} zdefiniowaną tak:

$M_1 \rightarrow_P M_2$ wtedy i tylko wtedy, gdy istnieją $x, N, \langle M, M' \rangle \in P, x_1, \dots, x_n, N_1, \dots, N_n$, takie że

$$M_1 = N[x := M[x_1 := N_1, \dots, x_n := N_n]]$$

i

$$M_2 = N[x := M'[x_1 := N_1, \dots, x_n := N_n]].$$

Nadużywając notacji, parę $\langle M, N \rangle \in P$ będziemy zwykle zapisywać jako $M \rightarrow_P N$.

Powiemy, że system przepisywania termów P jest *staty*, gdy termy ze zbioru P należą do T_{Σ} .

W niniejszej pracy będziemy rozważali systemy przepisywania termów ze skończonym zbiorem reguł.

Powyższa definicja określała pojęcie jednokrokowej relacji przepisywania. Można oczywiście powyższą jednokrokovą relację rozszerzyć do relacji wielokrokowego przepisywania, będącej zwrotno-przechodnim domknięciem relacji \rightarrow_P . Relację taką oznaczamy \Rightarrow_P . Podobnie można określić relację równoważności $=_P$ będącą domknięciem zwrotno-symetryczno-przechodnim relacji \rightarrow_P .

1.4 Podstawowe pojęcia związane z unifikacją

Kolejnym badaniem w niniejszej pracy pojęciem jest pojęcie unifikacji. Zależne jest ono od

- języka wyrażeń formalnych \mathcal{L} ze zmiennymi;
- relacji równoważności \equiv nad powyższymi wyrażeniami;
- klasy \mathcal{S} dopuszczalnych podstawień;
- oraz częściowej operacji $-(-) : \mathcal{S} \times \mathcal{L} \rightarrow \mathcal{L}$ będącej sposobem działania podstawienia na wyrażenia (dziedzina tej operacji określa, do jakich termów można stosować jakie podstawienia).

Definicja 1.4.1 (unifikacja)

Unifikacją w języku \mathcal{L} ze zmiennymi X z dokładnością do równości \equiv przy podstawieniach \mathcal{S} działających zgodnie z $-(-)$ nazwiemy problem decyzyjny, w którym

Dane: Zbiór E par wyrażeń w \mathcal{L} ze zmiennymi X .

Pytanie: Czy istnieje podstawienie $S \in \mathcal{S}$, takie że dla każdej pary $\langle M_1, M_2 \rangle \in E$ mamy $\langle S, M_i \rangle \in \text{Dom}(-(-))$ dla $i = 1, 2$ oraz $S(M_1) \equiv S(M_2)$.

Wyrażenia ze zbioru E będziemy nazywali równaniami. Dla podkreślenia faktu, że mamy do czynienia z równaniem będziemy $\langle M_1, M_2 \rangle \in E$ oznaczali też przez $M_1 \doteq M_2$. Rozszerzymy też notację dotyczącą zmiennych wolnych. $FV(E)$ będzie oznaczało zbiór wszystkich zmiennych wolnych we wszystkich termach znajdujących się w równaniach E .

Istnieje wiele problemów unifikacyjnych odpowiadających temu schematowi, np.: unifikacja robinsonowska, unifikacja modulo przepisywanie, unifikacja wyższego rzędu itd.

W niniejszej pracy będziemy zajmowali się różnymi wariacjami problemu unifikacji drugiego rzędu.

Definicja 1.4.2 (unifikacja drugiego rzędu)

Unifikacja drugiego rzędu to problem unifikacji, gdzie

- językiem jest rachunek λ z typami prostymi rzędu 2;
- równością jest relacja $=_\beta$;
- podstawienia to skończone funkcje częściowe ze zbioru zmiennych pierwszego i drugiego rzędu do zbioru termów z typami prostymi rzędu 2, zachowujące typy zmiennych (tzn. dla zmiennej typu σ przypisujące term typu σ);
- zaś wynik działania podstawienia określony jest jak w definicji 1.1.7.

Problem ten w ogólności, jak pokazał Goldfarb w [Gol81], jest nierozstrzygalny.

W stosunku do zmiennych używanych przy unifikacji będziemy używali odmiennej notacji dla podstawień, w której zamiast postfiksowo zapisywać: $M[x_1 := N_1, \dots, x_n := N_n]$ będziemy pisać $S(M)$ (zob. definicja 1.1.7). Pozwoli nam to zachować zgodność z definicją 1.4.1 oraz z typową notacją stosowaną w kontekście problemów unifikacji. Dodatkowo zakładamy w całym tekście, że w tego typu podstawieniach wszystkie termy podstawiane (w $M[x_1 := N_1, \dots, x_n := N_n]$ termy N_1, \dots, N_n) są w postaci normalnej.

Definicja 1.4.3 (stałe podstawienie)

Będziemy mówić, że podstawienie S jest *stałe* jeśli dla dowolnej zmiennej x term $S(x)$ nie ma zmiennych wolnych (choć być może zawiera pewne stałe).

W rozdziale 2 pokazujemy nierozstrzygalność pewnego interesującego szczególnego przypadku unifikacji drugiego rzędu — unifikacji z prostymi argumentami. Aby określić taką unifikację musimy zdefiniować dwa rodzaje zmiennych: zmienne bezpieczne i zmienne niebezpieczne.

Definicja 1.4.4 (zmienne bezpieczne i niebezpieczne)

Zmienną bezpieczną w termie M nazwiemy zmienną wolną x , taką że w każdym podtermie postaci $xM_1 \dots M_n$ termy $M_1 \dots M_n$ są zamknięte. *Zmienną niebezpieczną* nazwiemy każdą inną zmienną wolną.

Definicja 1.4.5 (unifikacja z prostymi argumentami)

Term z prostymi argumentami to term, którego wszystkie zmienne wolne są bezpieczne.

Unifikacja z prostymi argumentami to unifikacja, w której instancje są ograniczone w ten sposób, że mogą w nich występować tylko termy z prostymi argumentami.

Unifikacja drugiego rzędu z prostymi argumentami to unifikacja z prostymi argumentami, przy czym język termów jest ograniczony do rachunku λ z typami prostymi rzędu 2.

Kolejnym problemem unifikacyjnym, którym się tutaj będziemy zajmować jest *problem unifikacji ze zmiennymi w pozycjach czołowych*. Problem ten definiujemy tak:

Definicja 1.4.6 (unifikacja ze zmiennymi w pozycjach czołowych)

Unifikacja ze zmiennymi w pozycjach czołowych zdefiniowana jest podobnie, jak unifikacja z prostymi argumentami. Dokonujemy dalszego ograniczenia na postać instancji. Mianowicie termy tutaj występujące, jeśli mają zmienną wolną, to zmienna ta występuje na początku termu (*w pozycji czołowej*), tj. termy ze zmienną wolną mają tutaj zawsze postać $FM_1 \dots M_n$, gdzie F jest zmienną wolną, a podtermy M_i dla $i = 1, \dots, n$ nie mają już wystąpień zmiennych wolnych.

Także w przypadku tego problemu mamy tutaj również do czynienia z całą klasą ograniczeń związanych z rzędem używanych termów. Przede wszystkim będzie nas tutaj interesował powyższy problem przy ograniczeniu do termów drugiego rzędu. Powoduje ono, że w unifikacji ze zmiennymi w pozycjach czołowych mamy trzy możliwe postacie równań:

$$Ft_1 \dots t_n \doteq F's_1 \dots s_m \quad (1.1)$$

$$Ft_1 \dots t_n \doteq f'(s_1, \dots, s_m) \quad (1.2)$$

$$f(t_1, \dots, t_n) \doteq f'(s_1, \dots, s_m), \quad (1.3)$$

gdzie F, F' są zmiennymi wolnymi drugiego rzędu, f, f' są symbolami stałych, $t_1, \dots, t_n, s_1, \dots, s_m$ są termami pierwszego rzędu.

Warto może zwrócić uwagę, na czym polega zasadnicza różnica między wyrażeniami dla unifikacji z prostymi argumentami i dla unifikacji czołowej. Otóż w tej pierwszej dopuszczalne jest równanie

$$Fc_1c_2c_2 \doteq f(Fc_1c_2c_3)c_4,$$

które nie może się pojawić w przypadku unifikacji czołowej — term z prawej strony ma wystąpienie zmiennej wolnej (F), ale w pozycji czołowej występuje symbol funkcyjny f , a nie ta zmienna.

Z drugiej jednak strony każde równanie unifikacji czołowej jest jednocześnie równaniem unifikacji z prostymi argumentami.

W zasadzie w całej pracy będziemy się zajmowali unifikacją drugiego rzędu w związku z tym zwykle będziemy, o ile to nie będzie powodowało niejasności, pomijali tę informację w określeniu problemu, z którym mamy do czynienia.

W rozdziale 2 będziemy się zajmowali jeszcze *unifikacją ze zmiennymi w pozycjach czołowych* ograniczoną do termów trzeciego rzędu. Na określenie tego problemu będziemy używali trochę innego, zgrabniejszego, określenia: *unifikacja ze zmiennymi trzeciego rzędu w pozycjach czołowych*.

Rozdział 2

Unifikacja

W niniejszym rozdziale zajmiemy się pokazywaniem wyników dotyczących unifikacji. Pokażemy tutaj oryginalne wyniki dotyczące unifikacji drugiego rzędu oraz przedstawimy dowód G. Hueta dotyczący unifikacji trzeciego rzędu. Udowodnimy tutaj, że unifikacja z prostymi argumentami jest nierozstrzygalna, zaś unifikacja ze zmiennymi w pozycjach czołowych jest rozstrzygalna. Dodatkowo wskażemy, że instancje unifikacji użyte w dowodzie G. Hueta to instancje unifikacji ze zmiennymi w pozycjach czołowych, ale z rozluźnionym warunkiem na rząd zmiennych — dozwolone są tutaj zmienne trzeciego rzędu.

2.1 Unifikacja z prostymi argumentami

W obecnym podrozdziale pokażemy nierozstrzygalność unifikacji drugiego rzędu. Dowód będzie polegał na redukcji problemu stopu dla automatów dwulicznikowych.

2.1.1 Automaty dwulicznikowe

Automaty dwulicznikowe zostały wprowadzone w latach sześćdziesiątych przez Mińskiego (zob. [Min61]) jako prosty model obliczeniowy, mający pełną siłę obliczeniową maszyny Turinga. Model ten był wielokrotnie wykorzystywany jako narzędzie do pokazywania nierozstrzygalności różnych problemów np. [Laf96, Urz94, Hoo66].

Wprowadzimy tutaj oznaczenia na zbiór możliwych operacji oraz testów wykonywanych na licznikach $\mathcal{O} = \{\text{NOP}, \text{INC}, \text{DEC}\}$ oraz $\mathcal{T} = \{\text{Z}, \text{NZ}\}$. Elementy zbioru \mathcal{O} będą służyły do opisu operacji na licznikach:

- NOP — brak operacji;
- INC — zwiększ licznik;
- DEC — zmniejsz licznik.

Elementy zbioru \mathcal{T} będą służyły do określania stanu liczników

- Z — licznik jest równy zero;
- NZ — licznik jest niezerowy.

Definicja 2.1.1 (automat dwulicznikowy)

Automat dwulicznikowy \mathcal{A} to czwórka

$$\langle Q, q_s, q_k, \delta \rangle,$$

gdzie

- Q stanowi zbiór stanów automatu,
- q_s — jest wyróżnionym stanem początkowym,
- q_k — jest wyróżnionym stanem końcowym,
- $\delta : \mathcal{T}^2 \times Q \setminus \{Z, Z, q_k\} \rightarrow \mathcal{O}^2 \times Q$ — jest funkcją (częściową) „następnego kroku”.

Będziemy tutaj stosować pewną notację dla funkcji δ . Piszemy:

$$\text{WL1, WL2, } q_1 \rightarrow_{\delta} \text{O1, O2, } q_2, \quad (2.1)$$

gdzie $\text{WL1, WL2} \in \mathcal{T}$, $q_1, q_2 \in Q$, zaś $\text{O1, O2} \in \mathcal{O}$, gdy

$$\delta(\text{WL1, WL2, } q_1) = (\text{O1, O2, } q_2).$$

Elementy funkcji δ będziemy także nazywali *regułami*. Automaty takie wykonują obliczenia zgodnie z powszechnie znanym wzorem. Mamy tutaj do dyspozycji dwa liczniki, które mogą przyjmować w dowolnym momencie wartości będące liczbami naturalnymi. Każde obliczenie zaczyna się w stanie startowym q_s , przy zerowej wartości liczników. Obliczenie kończy się pomyślnie, gdy automat znajdzie się w stanie q_k . Kroki obliczenia wykonywane są zgodnie z funkcją δ . Dla przejścia (2.1) wykonywana są następujące operacje: jeśli automat \mathcal{A} , będąc w stanie q_1 , stwierdzi, że pierwszy licznik spełnia warunek WL1, zaś drugi licznik spełnia warunek WL2, to może wykonać operację O1 na pierwszym liczniku, operację O2 na drugim i przejść do stanu q_2 .

Definicja 2.1.2 (konfiguracja)

Konfiguracja automatu to trójka

$$\langle q, n_1, n_2 \rangle,$$

gdzie q jest stanem, zaś n_1 i n_2 wartościami liczników.

Funkcję δ rozszerzamy do konfiguracji:

Definicja 2.1.3 (funkcja \rightarrow_δ na konfiguracjach)

Przyjmujemy, że

$$\langle q_1, n_1, n_2 \rangle \rightarrow_\delta \langle q_2, n'_1, n'_2 \rangle$$

wtedy i tylko wtedy, gdy

$$\text{WL1, WL2, } q_1 \rightarrow_\delta \text{O1, O2, } q_2,$$

przy czym warunek WL1 jest spełniony dla n_1 , warunek WL2 jest spełniony dla n_2 oraz $n'_1 = \text{O1}(n_1)$ i $n'_2 = \text{O2}(n_2)$.

Definicja 2.1.4 (obliczenie)

Obliczeniem nazwiemy dowolny ciąg konfiguracji a_1, \dots, a_n, \dots , taki że dla dowolnych elementów a_i, a_{i+1} mamy $a_i \rightarrow_\delta a_{i+1}$.

Definicja 2.1.5 (zatrzymywanie się automatu)

Mówimy, że *automat się zatrzymuje*, gdy istnieje jego skończone obliczenie zaczynające się od konfiguracji $\langle q_s, 0, 0 \rangle$, którego ostatnia konfiguracja

1. albo nie należy do dziedziny \rightarrow_δ ,
2. albo jest równa $\langle q_k, 0, 0 \rangle$.

Obliczenie spełniające powyższe warunki nazywamy *obliczeniem zakończonym*.

Dla automatów dwulicznikowych, podobnie jak dla wszystkich innych rodzajów automatów rozważany jest następujący problem:

Problem 2.1.6 (problem stopu)

Dane: *Automat* \mathcal{A} .

Pytanie: *Czy automat* \mathcal{A} *zatrzymuje się?*

Mamy następujące twierdzenie:

Twierdzenie 2.1.7 (nierozstrzygalność problemu stopu)

Problem stopu dla automatów dwulicznikowych jest nierozstrzygalny.

Dowód:

Zob. [HU79]. □

Bezpośrednie zastosowanie powyższego twierdzenia nie jest najwygodniejsze. W związku z tym nałożymy pewne dodatkowe warunki na automaty dwulicznikowe.

Definicja 2.1.8 (dodatkowe warunki dla automatów dwulicznikowych)

Dodatkowe warunki dla automatów dwulicznikowych:

1. dla każdej pary stanów q_1, q_2 zbiór $\delta \cap \mathcal{T}^2 \times \{q_1\} \times \mathcal{O}^2 \times \{q_2\}$ ma moc nie większą niż 1;
2. dla każdego stanu q mamy $\delta \cap \mathcal{T}^2 \times \{q\} \times \mathcal{O}^2 \times \{q\} = \emptyset$.

Dla tak ograniczonych automatów dwulicznikowych mamy

Twierdzenie 2.1.9 (ograniczenia nie szkodzą)

Ograniczenia wprowadzone w definicji 2.1.8 dają klasę automatów, która ma nierozstrzygalny problem stopu.

Dowód:

Ze względu na rutynowość stosowanych technik szkicujemy tutaj tylko ten dowód, uzupełnienie szczegółów zostawiając czytelnikom. W każdym przypadku przekształcamy dowolny automat na automat, który spełnia konkretny warunek.

1. Określimy nowy automat $\mathcal{A}' = \langle Q', q'_s, q'_k, \delta' \rangle$, w którym

- $Q' = \mathcal{T}^2 \times Q \cup \{q'_k\}$;
- $q'_s = \langle 0, q_s \rangle$
- $\delta'(\text{WL1}_2, \text{WL2}_2, \langle \text{WL1}_1, \text{WL2}_1, q \rangle) = (\text{O1}, \text{O2}, \langle \text{WL1}_2, \text{WL2}_2, q' \rangle)$,
przy czym $\delta(\text{WL1}_2, \text{WL2}_2, q) = (\text{O1}, \text{O2}, q')$,
- $\delta'(\text{Z}, \text{Z}, \langle \text{WL1}_1, \text{WL2}_1, q_k \rangle) = (\text{NOP}, \text{NOP}, q'_k)$.

Zauważmy, że ostatni punkt naszego określenia δ' nie powoduje, że relacja ta przestaje być funkcją częściową. Wynika to z faktu, że $\langle \text{Z}, \text{Z}, q_k \rangle$ nie należy do dziedziny δ i w związku z tym interesująca nas wartość nie mogła być określona w przedostatnim punkcie.

Oczywiste jest przy tej definicji, że jeśli dla automatu \mathcal{A} istnieje obliczenie skończone, to dla automatu \mathcal{A}' też. Z kolei, jeśli istnieje zakończone obliczenie dla automatu \mathcal{A}' , to można z tego obliczenia odczytać obliczenie zakończone \mathcal{A} , rzutując wszystkie stany z obliczenia \mathcal{A}' na trzecią współrzędną, o ile stan należy do $\mathcal{T}^2 \times Q$. Jeśli zaś stanem jest q'_k , to usuwamy ten stan i z poprzedzającego stanu wykonujemy przejście takie, jakie było wykonane ze stanu q'_k . Szczegóły dowodu oraz sprawdzenie, że warunek (1) jest spełniony zostawiamy Czytelnikowi.

2. Tutaj postępowanie jest podobne jak przy warunku (1). Określamy automat $\mathcal{A}' \langle Q', q'_s, q'_k, \delta' \rangle$, w którym

- $Q' = \mathcal{T}^2 \times Q \cup \{q'_k\}$;
- $q'_s = \langle \text{Z}, \text{Z}, q_s \rangle$
- $\delta'(\text{WL1}, \text{WL2}, \langle i, q \rangle) = (\text{O1}, \text{O2}, \langle i, q' \rangle)$, o ile spełniony jest warunek $\delta(\text{WL1}, \text{WL2}, q) = (\text{O1}, \text{O2}, q')$ oraz $q \neq q'$,

- $\delta'(\text{WL1}, \text{WL2}, \langle i, q \rangle) = (\text{O1}, \text{O2}, \langle (i+1) \bmod 2, q \rangle)$, o ile spełniony jest warunek $\delta(\text{WL1}, \text{WL2}, q) = (\text{O1}, \text{O2}, q)$,
- $\delta'(\text{Z}, \text{Z}, \langle i, q_k \rangle) = q'_k$ dla $i = 1, 2$.

Relacja δ' jest funkcją z tych samych powodów, które powodowały funkcyjność δ' w konstrukcji dla poprzedniego warunku.

Oczywiste jest przy tej definicji, że jeśli dla automatu \mathcal{A} istnieje obliczenie zakończone, to dla automatu \mathcal{A}' też. Z kolei, jeśli istnieje obliczenie zakończone dla automatu \mathcal{A}' , to można z tego obliczenia odczytać obliczenie zakończone \mathcal{A} rzutując wszystkie stany z obliczenia \mathcal{A}' na drugą współrzędną, o ile stan należy do $\{0, 1\} \times Q$. Jeśli zaś stanem jest q'_k , to usuwamy ten stan. Szczegóły dowodu oraz sprawdzenie, że warunek (2) jest spełniony zostawiamy Czytelnikowi.

□

2.1.2 Nierozstrzygalność unifikacji drugiego rzędu

W niniejszej części pracy będziemy pracowali z termami, w których, zgodnie z konwencją notacyjną wprowadzoną w podrozdziale 1.4, występuje symbol stałej \rightarrow , która ma typ $\alpha \rightarrow \alpha \rightarrow \alpha$. Dodatkowo będziemy się tutaj posługiwali pewną liczbą stałych typu α . Proszę zauważyć, że wszystkie wprowadzone stałe mają rząd 2. W związku z tym, że \rightarrow jest symbolem dwuargumentowym, a kojarzy się z implikacją, będziemy stosowali wobec niego notację infiksową (np. $\rightarrow MN$ będziemy zapisywali jako $M \rightarrow N$).

Przepisywanie a równania unifikacyjne

Na proces obliczania można patrzeć jako na proces swego rodzaju przepisywania jednych konfiguracji w drugie. Tę intuicję wykorzystamy w naszej konstrukcji unifikacyjnej. Pokażemy tutaj związek między unifikacją z prostymi argumentami a przepisywaniem termów.

W poniższych akapitach będziemy pokazywali związek między systemami przepisywania termów (zob. definicja 1.3.8) a równaniami unifikacyjnymi.

Definicja 2.1.10 (równanie redukcyjne dla systemu P)

Niech system redukcyjny P będzie dany jako $\{M_1^1 \rightarrow_P M_1^2, \dots, M_n^1 \rightarrow_P M_n^2\}$. *Równaniem redukcyjnym dla stałego systemu redukcyjnego P zaczynającym od termu M_0 , a kończącym na M_k* nazwiemy dowolne równanie unifikacji drugiego rzędu postaci:

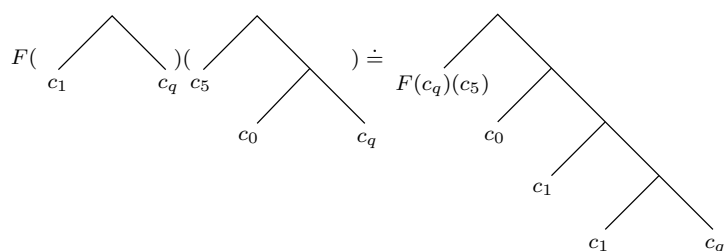
$$FM_1^1 \dots M_n^1(c \rightarrow (d \rightarrow M_k)) \doteq (FM_1^2 \dots M_n^2 c) \rightarrow (d \rightarrow M_0)$$

(c, d są wyróżnionymi stałymi pierwszego rzędu nie występującymi w termach P).

Przyjrzyjmy się teraz przykładowi, który da nam wyobrażenie, jak wygląda związek pomiędzy właśnie opisanymi równaniami a przepisywaniem (uwaga! przykład ten jest uproszczony w stosunku do definicji 2.1.10 i nie uwzględnia roli stałej d). Weźmy pod uwagę system redukcyjny \rightarrow_P określony przez zbiór

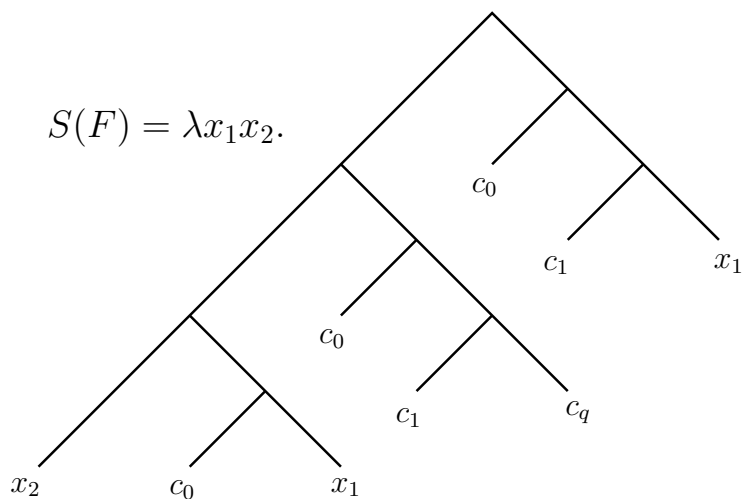
$$P = \{ (c_1 \rightarrow c_q) \rightarrow_R c_q \}.$$

Odpowiada mu równanie przedstawione na rysunku 2.1. Rozważmy pewne



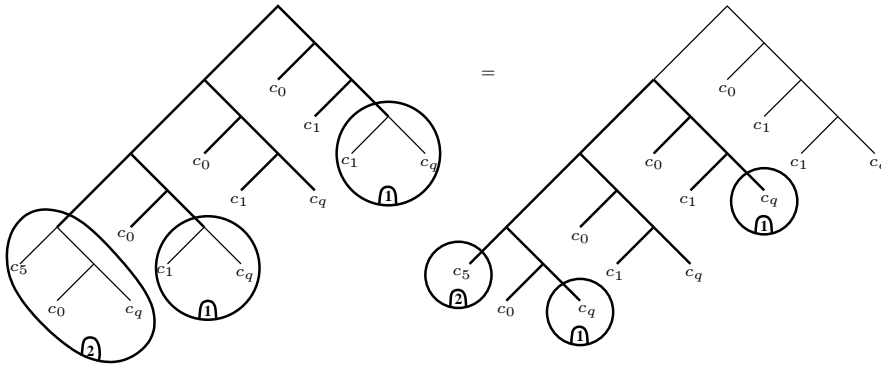
Rysunek 2.1: Przykład równania dla przepisywania

podstawienie S , które jak się okaże jest rozwiązaniem równania przedstawionego na rysunku 2.1. Podstawienie to ma postać taką jak na rysunku 2.2. Sposób jego działania widać na rysunku 2.3 (każde rozgałęzienie drzewa



Rysunek 2.2: Rozwiązanie równania dla przepisywania

oznacza obecność symbolu \rightarrow). Widoczne na tym rysunku elementy zaznaczone grubszą linią oznaczają fragmenty termu pochodzące z podstawienia, elementy zaznaczone cieńszą linią oznaczają fragmenty termu pochodzące z równania. Porównanie rysunków 2.1–2.3 pozwala natychmiast stwierdzić, że S jest rozwiązaniem rozważanego tutaj równania.



Rysunek 2.3: Rozwiązanie po dokonaniu podstawienia i redukcji

Warto zwrócić uwagę na pracę następującego mechanizmu. Term, który znajduje się w prawej gałęzi termu po prawej stronie równania, musi zostać odtworzony po lewej stronie równania. Ponieważ żaden argument nie zawiera w sobie termu z prawej strony, to na F musi zostać podstawione coś, co zaczyna się od \rightarrow . Stąd mamy prawą i lewą gałąź rozwiązania. Jeśli będziemy powtarzali podobne rozumowanie, to w końcu dojdziemy do tego, że prawe poddrzewo termu podstawianego na F ma postać $c_0 \rightarrow c_1 \rightarrow G$, przy czym na G można już podstawić jakiś argument F . Spróbujmy skorzystać z tej sposobności i wstawić na G argument $c_1 \rightarrow c_q$. Dostaniemy wtedy prawą gałąź termu podstawianego na F taką jak w rozwiązaniu z rysunku 2.2. W związku z tym, że po prawej stronie równania występuje również zmienna F , to tam także musi się pojawić prawa gałąź termu podstawianego na F . Z tym, że zmienił nam się argument, który miał trafić na miejsce G . Jest nim c_q . W ten sposób doszło do zamienienia w prawej podgałęzi termu $c_1 \rightarrow c_q$ na term c_q . Odpowiada to bezpośrednio wykonaniu przepisania termu $c_1 \rightarrow c_1 \rightarrow c_q$ na term $c_1 \rightarrow c_q$ zgodnie z regułą systemu P : $(c_1 \rightarrow c_q) \rightarrow_P c_q$.

Prawa gałąź termu podstawianego za pomocą S wraz z podstawionym na jej końcu argumentem c_q zmiennej F znajduje się teraz po prawej stronie równania i musi zostać skopiowana po lewej stronie równania. Znowu można uznać, że do końcówki rozważanej przez nas w tym momencie gałęzi pasuje któryś z argumentów. Można jednak nie wykonywać takiego przypasowania. Wtedy nasza gałąź, skopiowana za pomocą F , musi się znaleźć po prawej stronie równania. Dokładnie taka sama gałąź pojawiła się w poprzedniej turze naszej analizy (na rysunku 2.3 druga i trzecia gałąź termu po prawej stronie równania). Cała historia się zatem powtarza — nasza gałąź musi zostać skopiowana po lewej stronie, tam może się dopasować jakiś argument, może też się nie dopasować. Opisaną tutaj sytuację, gdy nie następuje dopasowanie się argumentu, da się doskonale odnieść do przepisywania w ten spo-

sób, że odpowiada ona zapisowi $t \rightarrow_P t$ — domknięcie zwrotno-przechodnie relacji jest przecież zwrotne i może oznaczać wykonanie zera kroków redukcji.

Zatrzymaliśmy naszą analizę sposobu rozwiązywania równania na trzeciej prawostronnej gałęzi po prawej stronie rozwiązania (rys. 2.2). Gałąź ta musi się znaleźć, jak i w przypadku poprzednich gałęzi, po lewej stronie równania. Tam dopasowuje się do niej pierwszy argument F , co po przejściu znowu na prawą stronę dają odpowiednik redukcji: $c_1 \rightarrow c_q \rightarrow_R c_q$ za pomocą reguły $c_1 \rightarrow c_q \rightarrow_R c_q$.

W tym momencie po lewej stronie pojawia się gałąź, do której może się dopasować drugi argument F , co też ma miejsce. Zauważmy, że argument ten dopasowuje się jednocześnie do lewego rozgałęzienia odbiegającego od naszej gałęzi, wstawiając tam stałą c_5 . Odgałęzienie to do tej pory nie było dopasowywane do żadnych fragmentów argumentów F i służyło za miejsce, gdzie realizował się będzie dalszy ciąg naszej redukcji. Teraz miejsce to zostaje „zasklepione” i po prawej stronie równania musi się pojawić coś, co zapewni końcowe wyrównanie termów. Tym czymś jest drugi argument F , który ma tutaj wartość c_5 .

Reasumując, powyższy opis pozwolił powiązać rozwiązanie równania z rysunku 2.1 z następującym ciągiem redukcji w systemie P :

$$(c_1 \rightarrow c_1 \rightarrow c_q) \rightarrow_P (c_1 \rightarrow c_q) \rightarrow_P (c_1 \rightarrow c_q) \rightarrow_P c_q$$

Zauważmy jeszcze, że powyższą redukcję można interpretować jako obliczenie automatu jednolicznikowego z jednym stanem q . Automat ten wykonuje w każdym kroku operację zmniejszania licznika, kończy zaś działanie po jego wyzerowaniu. Wstawiając za prawe rozgałęzienie prawej strony równania z rysunku 2.1 termy postaci

$$c_0 \rightarrow c_1 \rightarrow c_1 \rightarrow \cdots \rightarrow c_1 \rightarrow c_q$$

(o większej liczbie gałęzi wiodących do stałej c_1), dostajemy rozwiązania odpowiadające obliczeniom, które wykonują więcej zmniejszeń tego licznika.

Powyższy opis można uogólnić do następującego twierdzenia.

Twierdzenie 2.1.11 (równanie redukcyjne i redukcja)

Równanie redukcyjne dla stałego systemu redukcyjnego P , które zaczyna od termu M_0 , a kończy na M_k ma rozwiązanie wtedy i tylko wtedy, gdy istnieje ciąg redukcji

$$M_0 \rightarrow_P M_1 \rightarrow_P \cdots \rightarrow_P M_k.$$

Co więcej, jeśli S jest rozwiązaniem równania redukcyjnego, to mamy ciąg redukcji (zgodnie z oznaczeniami definicji 2.1.10):

$$1^0 22S(N) \rightarrow_P 1^1 22S(N) \rightarrow_P \cdots \rightarrow_P 1^i 22S(N) \rightarrow_P \cdots \rightarrow_P 1^n 22S(N),$$

gdzie $N = FM_1^1 \dots M_n^1(c \rightarrow (d \rightarrow M_k))$ oraz $1^{n+1}S(N) = c$.

Dowód:

(\Rightarrow) Niech równanie redukcyjne dla stałego systemu redukcyjnego P zaczynające od termu M_0 , a kończące na M_k ma rozwiązanie S . Niech $S(F) = \lambda x_1 \dots x_{n+1}.M$. Dowód przeprowadzimy przez indukcję ze względu na liczbę słów postaci 1^*2 , które są ścieżkami w M .

Jeśli liczba takich ścieżek jest zerowa, to głównym symbolem M nie jest symbol \rightarrow . Symbolem tym może być

- inna stała; wtedy jednak nie można zrównać lewej strony równania z prawą, bo po prawej stronie głównym symbolem jest \rightarrow ;
- symbol x_i ; ponieważ po prawej stronie w równaniu występuje stała d , a jedynym argumentem F po lewej stronie, który zawiera d jest argument $n+1$, to $i = n+1$, to jednak natychmiast oznacza, że $M_k = M_0$, a więc mamy ciąg redukcji długości zero.

Jeśli liczba ścieżek postaci 1^*2 jest niezerowa, to głównym symbolem termu M jest \rightarrow . W związku z tym rozwiązywalność naszego równania jest równoważna rozwiązywalności równania:

$$(F_1 M_1^1 \dots M_n^1 (c \rightarrow (d \rightarrow M_k))) \rightarrow N_L \doteq ((F_1 M_1^2 \dots M_n^2 c) \rightarrow N_P) \rightarrow (d \rightarrow M_0), \quad (2.2)$$

gdzie $N_L = F_2 M_1^1 \dots M_n^1 (c \rightarrow (d \rightarrow M_k))$ oraz $N_P = F_2 M_1^2 \dots M_n^2 c$. Niech $S'(F_j) = \lambda x_1 \dots x_{n+1}.j(S(F)x_1 \dots x_{n+1})$ dla $j = 1, 2$. Ponieważ S było rozwiązaniem pierwotnego równania, S' jest rozwiązaniem równania (2.2). Term $S'(F_2)x_1 \dots x_{n+1}$ nie zawiera wystąpienia zmiennej x_{n+1} . Wynika to z tego, że term $d \rightarrow M_0$ nie zawiera stałej c , a taka stała występuje w $n+1$. argumentem F (czyli w $c \rightarrow (d \rightarrow M_k)$). Oznacza to, że dla $N'_L = S'(F_2)M_1^1 \dots M_n^1 x_{n+1}$ i $N'_P = S'(F_2)M_1^2 \dots M_n^2 x_{n+1}$ mamy $2(N'_L) \rightarrow_R 2(N'_P)$. Rzeczywiście, stosowny ciąg redukcji można zdefiniować tutaj tak: niech $A = \{\omega_0, \dots, \omega_l\}$ będzie zbiorem pozycji w N , na których znajdują się zmienne x_1, \dots, x_n . Definiujemy $P_0 = 2(N'_L)$ oraz $P_{i+1} = P_i[\omega_i \leftarrow M_j^2]$, przy czym $\omega_i N = x_j$. Ta definicja jest poprawna, ponieważ zastąpienia następują w miejscach, gdzie w N były zmienne x_1, \dots, x_n . Z tego samego powodu mamy własność $\omega_i P_{i-1} = M_j^1$, co daje natychmiast $P_{i-1} \rightarrow_P P_i$. Mamy zatem

$$2(N'_L) \rightarrow_P 2(N'_P). \quad (2.3)$$

Z równania (2.2) można wyłuskać lewe argumenty głównego wystąpienia stałej \rightarrow , które dadzą równanie:

$$F_1 M_1^1 \dots M_n^1 (c \rightarrow (d \rightarrow M_k)) \doteq (F_1 M_1^2 \dots M_n^2 c) \rightarrow N'_P, \quad (2.4)$$

przy czym term N'_P ma postać $d \rightarrow N''_P$ (to ostatnie wynika z faktu, że $2S'(N_L) = d \rightarrow M_0$ i żaden z argumentów M zmiennej F_2 nie ma wystąpienia termu d w $1(M)$). To powoduje, że równanie (2.4) jest równaniem

redukcyjnym dla systemu P zaczynającym od N''_P , a kończącym na M_k . Tutaj rozwiązanie określone jest przez podstawienie S' , które na F_1 podstawia zgodnie ze swoją definicją term o jedno odgałęzienie w lewo krótszy niż term podstawiany przez S na F . Pozwala to na skorzystanie z założenia indukcyjnego, co daje $N''_P \rightarrow_R M_k$. To zaś razem z (2.3) daje ciąg redukcji od M_0 do M_k postaci $M_0 \rightarrow_R N''_P \rightarrow_R M_k$.

(\Leftarrow) Dowód przez indukcję ze względu na długość ciągu redukcji.

Jeśli ciąg redukcji ma zero kroków, to $M_0 = M_k$ i rozwiązaniem jest podstawienie $S(F) = \lambda x_1, \dots, x_n, x_{n+1}.x_{n+1}$. Podstawienie to zaaplikowane do lewej strony daje $n+1$. argument, czyli $c \rightarrow (d \rightarrow M_k)$, zaś zaaplikowane do prawej strony — term postaci $M \rightarrow (d \rightarrow M_0)$, przy czym $M = c$, gdyż taki jest tutaj $n+1$. argument F . To zaś implikuje, że obie strony równania po wykonaniu podstawienia są równe.

Jeśli ciąg redukcji ma $m+1$ kroków, to wygląda on tak: $M_0 \rightarrow_P M_1 \rightarrow_P \dots \rightarrow_P M_k$. Z założenia indukcyjnego mamy rozwiązanie S' dla równania redukcyjnego dla P zaczynającego od termu M_1 , a kończącego na M_k :

$$F_1 M_1^1 \dots M_n^1 (c \rightarrow (d \rightarrow M_k)) \doteq (F_1 M_1^2 \dots M_n^2 c) \rightarrow (d \rightarrow M_1).$$

Zdefiniujemy rozwiązanie równania redukcyjnego dla P zaczynającego od termu M_0 , a kończącego na M_k , jako

$$S(F) = \lambda x_1 \dots x_{n+1}.(S'(F_1)x_1 \dots x_{n+1}) \rightarrow (d \rightarrow (M_0[\omega \leftarrow x_i])),$$

przy czym ω jest pozycją redeksu, który został użyty w pierwszym kroku wykonywanej tutaj redukcji, a i jest argumentem, w którym zakodowana została para z P użyta w pierwszym kroku redukcji.

Pokażemy, że

$$S(FM_1^1 \dots M_n^1 (c \rightarrow (d \rightarrow M_k))) = S((FM_1^2 \dots M_n^2 c) \rightarrow (d \rightarrow M_0)).$$

Oznaczmy $FM_1^1 \dots M_n^1 (c \rightarrow (d \rightarrow M_k))$ jako M_L , zaś $(FM_1^2 \dots M_n^2 c) \rightarrow (d \rightarrow M_0)$ jako M_P . Zgodnie z definicją S mamy, że

$$S(M_L) = (S'(F_1)M_1^1 \dots M_n^1 (c \rightarrow (d \rightarrow M_k))) \rightarrow (d \rightarrow M_0[\omega \leftarrow M_i^1]).$$

Zauważmy jednak, że $M_0[\omega \leftarrow M_i^1] = M_0$, a zatem $2(S(M_L)) = 2(S(M_P))$.

Z definicji S mamy, że $1(S(M_L)) = S'(F_1)M_1^1 \dots M_n^1 (c \rightarrow (d \rightarrow M_k))$. Z drugiej strony z homomorficzności S mamy $1(S(M_P)) = S(FM_1^2 \dots M_n^2 c)$ i dalej z definicji S uzyskujemy $1(S(M_P)) = S'(F_1)M_1^2 \dots M_n^2 c \rightarrow (d \rightarrow M_0[\omega \leftarrow M_i^2])$, ale z definicji ciągu redukcji $M_0[\omega \leftarrow M_i^2] = M_1$. W związku z tym wystarczy, że pokażemy równość:

$$S'(F_1)M_1^1 \dots M_n^1 (c \rightarrow (d \rightarrow M_k)) = S'(F_1)M_1^2 \dots M_n^2 c \rightarrow (d \rightarrow M_1).$$

To zaś natychmiast wynika z założenia indukcyjnego.

Część „Co więcej” pozostawiamy do udowodnienia czytelnikowi. \square

Powyższe twierdzenie można jeszcze dodatkowo uogólnić stosując następujący lemat:

Lemat 2.1.12 (nieszkodliwy unifikator)

Niech S będzie statym unifikatorem (zob. definicja 1.4.3), który unifikuje $M_1 \doteq M_2[\omega \leftarrow S(N)]$, gdzie $\omega \in \{1, 2\}^*$, zaś M_1, M_2, N są pewnymi termami. Jeśli $FV(N) \cap (FV(M_1) \cup FV(M_2[\omega \leftarrow c_0])) = \emptyset$, gdzie c_0 jest pewną stałą pierwszego rzędu, oraz żaden prefiks ω nie wskazuje w M_2 na podterm zaczynający się od zmiennej, to S unifikuje $M_1 \doteq M_2[\omega \leftarrow N]$.

Dowód:

Indukcja ze względu na długość ścieżki ω .

Jeśli ścieżka ta ma długość zero, to założenie redukuje się do informacji, że S unifikuje $M_1 \doteq S(N)$. Ponieważ $S(N)$ nie ma zmiennych wolnych, mamy $S(M_1) = S(N)$, stąd S unifikuje $M_1 \doteq N$, czyli $M_1 \doteq M_2[\omega \leftarrow N]$.

Jeśli ścieżka ω ma długość większą od zera, to głównym symbolem M_2 musi być \rightarrow (symbol stały, bo żaden prefiks ω nie wskazuje na term zaczynający się od zmiennej). Dodatkowo $\omega = A\omega'$ dla $A = 1$ lub 2 . Załóżmy, że $A = 1$, dowód dla $A = 2$ przebiega analogicznie. Rozważmy dwa przypadki:

- Gdy głównym symbolem M_1 jest też \rightarrow , to możemy zaaplikować 1 do obu stron równania i z definicji zastąpienia podtermu dostać, że S unifikuje $1(M_1) \doteq 1(M_2)[\omega' \leftarrow S(N)]$. Założenie indukcyjne daje nam, że S unifikuje

$$1(M_1) \doteq 1(M_2)[\omega' \leftarrow N]. \quad (2.5)$$

Z założenia, że S unifikuje $M_1 \doteq M_2[\omega \leftarrow S(N)]$ oraz z faktu, że dla ω zaczynającego się od 1 prawe termy nie ulegają zmianie przy zastępowaniu, wiemy, iż S unifikuje $2(M_1) \doteq 2(M_2)$. To wraz z (2.5) daje unifikowanie $M_1 \doteq M_2[\omega \leftarrow N]$.

- Gdy głównym symbolem M_1 nie jest \rightarrow , to symbolem tym musi być zmienna wolna (w przeciwnym wypadku po lewej i prawej stronie mielibyśmy różne stałe). Niech zmienną tą będzie F .

- Jeśli jest to zmienna pierwszego rzędu, to rozwiązanie dla $M_1 \doteq M_2[\omega \leftarrow S(N)]$ istnieje wtedy i tylko wtedy, gdy istnieje rozwiązanie dla równania $F_1 \rightarrow F_2 \doteq M_2[\omega \leftarrow S(N)]$. Dalej zaś można postępować tak, jak w przypadku, gdy głównym symbolem po lewej stronie równania było \rightarrow .
- Jeśli jest to zmienna drugiego rzędu, to $M_1 = FN_1 \dots N_n$ i rozwiązanie dla $M_1 \doteq M_2[\omega \leftarrow S(N)]$ istnieje wtedy i tylko wtedy, gdy istnieje rozwiązanie dla równania

$$F_1 N_1 \dots N_n \rightarrow F_2 N_1 \dots N_n \doteq M_2[\omega \leftarrow S(N)].$$

Dalej zaś można postępować tak, jak w przypadku, gdy głównym symbolem po lewej stronie równania było \rightarrow .

□

Powyższy lemat ma zastosowanie w sytuacji, gdy przepisywanie musimy wykonać niekoniecznie zaczynając od termu stałego. Uogólnienie, z którego będziemy korzystać w dalszej części tej pracy, wygląda tak:

Twierdzenie 2.1.13 (uogólnione równanie redukcyjne i redukcja)

Niech S będzie stałym podstawieniem i dalej niech $N_1 \doteq N_2[22 \leftarrow S(N)]$ będzie równaniem redukcyjnym dla stałego systemu redukcyjnego P zaczynającym od termu $S(N)$, a kończącym na pewnym termie M_k . Niech $FV(N) \cap (FV(N_1) \cup FV(N_2[22 \leftarrow c_0])) = \emptyset$, gdzie c_0 jest pewną stałą pierwszego rzędu.

Jeśli S unifikuje $N_1 \doteq N_2[\omega \leftarrow N]$, to istnieje ciąg redukcji

$$S(N) \rightarrow_P M_1 \rightarrow_P \cdots \rightarrow_P M_k.$$

Jeśli istnieje ciąg redukcji

$$S(N) \rightarrow_P M_1 \rightarrow_P \cdots \rightarrow_P M_k,$$

to istnieje unifikator dla $N_1 \doteq N_2[22 \leftarrow N]$.

Dowód:

Jeśli S unifikuje $N_1 \doteq N_2[\omega \leftarrow N]$, to także unifikuje $N_1 \doteq N_2[\omega \leftarrow S(N)]$, gdyż S jest stałym podstawieniem. Ponieważ to ostatnie jest równaniem redukcyjnym dla stałego systemu redukcyjnego P zaczynającym od termu $S(N)$, z twierdzenia 2.1.11 mamy ciąg redukcji

$$S(N) \rightarrow_P M_1 \rightarrow_P \cdots \rightarrow_P M_k.$$

Z kolei, jeśli mamy ciąg redukcji, jak powyżej, to z twierdzenia 2.1.11 istnieje rozwiązanie S' równania $N_1 \doteq N_2[22 \leftarrow S(N)]$. Zdefiniujemy

$$S''(x) = \begin{cases} S(x) & \text{dla } x \in FV(N) \\ S'(x) & \text{dla } x \notin FV(N). \end{cases}$$

Podstawienie S'' unifikuje $N_1 \doteq N_2[22 \leftarrow N]$, gdyż zmienne z N są rozłączne ze zmiennymi z N_1 i N_2 . □

Przedstawione uogólnienie będzie nam potrzebne jako część konstrukcji pokazującej nierozstrzygalność — jedno z utworzonych równań będzie miało jako term początkowy wynik podstawienia na pewien term.

Kodowanie

Będziemy teraz definiować po kolei poszczególne elementy kodowania całego procesu obliczania za pomocą automatu dwulicznikowego. Zaczniemy od kodowania licznika. Będziemy tutaj posługiwali się dodatkową stałą pierwszego rzędu c_1 .

Definicja 2.1.14 (kod pseudolicznika)

Kod pseudolicznika $\text{plicznik}(n, M)$, gdzie n jest liczbą naturalną (zawartością licznika) zaś M jest termem, definiujemy rekurencyjnie jako

- $\text{plicznik}(0, M) = M$;
- $\text{plicznik}(n + 1, M) = c_1 \rightarrow \text{plicznik}(n, M)$.

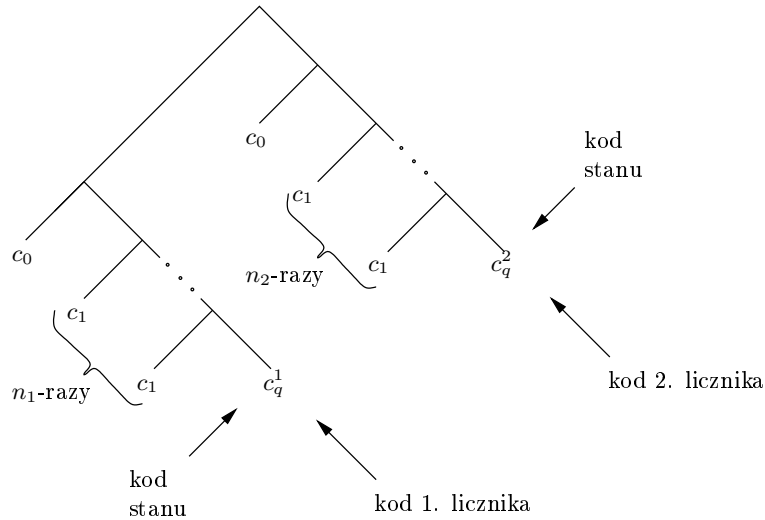
Definicja 2.1.15 (kod konfiguracji i kod pseudokonfiguracji)

Kodem pseudokonfiguracji nazwiemy term postaci $M_1 \rightarrow M_2$, w którym $M_i = c_0 \rightarrow \text{plicznik}(n_i, c_{q_i}^i)$ dla pewnych liczb naturalnych n_i i stanów q_i , gdzie $i \in \{1, 2\}$.

Dla danej konfiguracji $\langle q, n_1, n_2 \rangle$ określamy kod konfiguracji $\langle q, n_1, n_2 \rangle$ jako

$$M_{\langle q, n_1, n_2 \rangle} = (c_0 \rightarrow \text{plicznik}(n_1, c_q^1)) \rightarrow (c_0 \rightarrow \text{plicznik}(n_2, c_q^2)).$$

Na rys. 2.1.2 uwidoczniono wygląd takiego kodowania. Liczba n_1 oznacza wartość pierwszego licznika, zaś liczba n_2 oznacza wartość drugiego licznika. Stałe c_q^1, c_q^2 informują, że automat znajduje się w stanie q .



Rysunek 2.4: Wygląd konfiguracji

Wprowadzimy teraz system redukcyjny, który będzie symulował obliczenie danego automatu dwulicznikowego \mathcal{A} . Za poniższą definicją znajduje się szersze omówienie opisanych w niej elementów konstrukcji.

Definicja 2.1.16 (system redukcyjny obliczania)

System redukcyjny obliczania Sym zdefiniowany jest jako zestaw reguł przepisywania

$$B_i^j \rightarrow_{Sym} B_i'^j,$$

gdzie $i \in \{1, \dots, |\delta|\}$, $j \in \{1, 2\}$, zaś termny $B_i^j, B_i'^j$ określone są w zależności od i -tej reguły funkcji δ . Niech i -ta reguła ma postać

$$\text{WL1, WL2, } q_1 \rightarrow_{\delta} \text{O1, O2, } q_2.$$

Kładziemy

$$B_i^j = c_a \rightarrow c_{q_1}^j, \quad (2.6)$$

gdzie $a = 0$, jeśli $\text{WL}j = \text{Z}$, lub $a = 1$, jeśli $\text{WL}j = \text{NZ}$. Następnie

$$B_i'^j = 1(B_i^j) \rightarrow (c_1 \rightarrow c_{q_2}^j), \quad \text{o ile } \text{O}j = \text{INC}. \quad (2.7)$$

$$B_i'^j = 1(B_i^j) \rightarrow c_{q_2}^j, \quad \text{o ile } \text{O}j = \text{NOP}; \quad (2.8)$$

$$B_i'^j = c_{q_2}^j, \quad \text{o ile } \text{O}j = \text{DEC}; \quad (2.9)$$

Spróbujemy przyrzeć się działaniu powyższej definicji na przykładzie. Założmy, że chcemy zapisać w naszym systemie redukcyjnym reguły dla przejścia:

$$\text{Z, NZ, } p \rightarrow_{\delta} \text{INC, DEC, } q.$$

Zgodnie z definicją tworzymy dla takiego przejścia dwie reguły:

$$\begin{aligned} B^1 &\rightarrow_{\text{Sym}} B'^1, \\ B^2 &\rightarrow_{\text{Sym}} B'^2. \end{aligned}$$

Pierwsza z nich odpowiada temu, co się dzieje w pierwszym liczniku — zgodnie z definicją stałe kodujące stan występują tutaj z górnym indeksem równym 1. Analogicznie druga reguła będzie odpowiadała drugiemu licznikowi i stałe kodujące stan występują tutaj z górnym indeksem równym 2. Termny B^1, B^2 odpowiadają za testowanie liczników i stanu wejściowego, zaś termny B'^1, B'^2 odpowiadają wykonywanym operacjom na licznikach.

Term B^1 będzie odpowiadał testowaniu zerowości pierwszego licznika i zgodnie z (2.1.16) jest równy $c_0 \rightarrow c_p^1$. Zauważmy, że zgodnie z rysunkiem 2.1.2 taki term przypasować się może tylko do końca lewej odnogi kodu konfiguracji i to tylko w takiej sytuacji, gdy nie ma w tej odnodze żadnego wystąpienia stałej c_1 , co odpowiada konfiguracji z $n_1 = 0$ — czyli sytuacji, gdy licznik ten jest zerowy.

Term B^2 będzie odpowiadał testowaniu niezerowości drugiego licznika i zgodnie z (2.1.16) jest równy $c_1 \rightarrow c_p^2$. Podobnie jak poprzednio zgodnie z rysunkiem 2.1.2 taki term przypasować się może tylko do końca prawej odnogi kodu konfiguracji i to tylko w takiej sytuacji, gdy w tej końcówce występuje c_1 nie ma w tej odnodze żadnego wystąpienia stałej c_1 , co odpowiada konfiguracji z $n_2 > 0$ — czyli sytuacji, gdy licznik ten jest niezerowy.

Operacja zwiększania wartości pierwszego licznika realizowana jest dzięki postaci termu B'^1 . Zgodnie z definicją 2.1.16 ma on postać $c_0 \rightarrow c_1 \rightarrow c_q^1$. Zauważmy, że wykonanie redukcji $B^1 \rightarrow_{\text{Sym}} B'^1$ powoduje zastąpienie $c_0 \rightarrow c_p^1$ przez $c_0 \rightarrow c_1 \rightarrow c_q^1$. Ten ostatni term zgodnie z definicją kodu

konfiguracji odpowiada wartości licznika 1 — czyli w naszej sytuacji zastanej wartości powiększonej o 1. Dodatkowo zamiast stałej c_p^1 pojawia się na końcu prawej gałęzi termu kodującego licznik stała c_q^2 , co oznacza, że wynikowy term odpowiada konfiguracji, w której automat znajduje się w stanie q .

Podobnie operacja zmniejszania drugiego licznika realizowana jest dzięki postaci termu B'^2 . Zgodnie z definicją 2.1.16 ma on postać c_q^2 . Wykonanie redukcji $B^2 \rightarrow_{Sym} B'^2$ powoduje zastąpienie $c_1 \rightarrow c_p^2$ przez c_q^2 . Oznacza to, że liczba c_1 w zredukowanym termie się zmniejszyła o 1, co oznacza dokładnie zmniejszenie licznika o 1. Jednocześnie następuje, podobnie jak w przypadku pierwszej redukcji, przejście od stanu p do q .

Zauważmy, że w powyższym opisie zakładaliśmy, że wyjściowy term ma postać taką, że na końcu jego lewej odnogi znajduje się c_p^1 , zaś na końcu prawej — c_p^2 . Wymuszenie, aby podczas całej redukcji zachodziła taka własność, jest w przypadku zwykłego przepisywania niemożliwe. Jednak okaże się, że unifikacja pozwala na nałożenie stosownych więzów na redukcję.

Tymczasem pokażemy związek pomiędzy zdefiniowanym systemem przepisywania a obliczeniami automatu. Na początku przedstawimy kilka technicznych lematów opisujących zależności między termami reprezentującymi konfiguracje a samymi konfiguracjami.

Lemat 2.1.17 (konfiguracja a testy liczników)

Dla dowolnej konfiguracji $\langle q, n, n' \rangle$:

1. jeśli $n = 0$, to $1(M_{\langle q, n, n' \rangle}) = c_0 \rightarrow c_q^1$;
2. jeśli $n' = 0$, to $2(M_{\langle q, n, n' \rangle}) = c_0 \rightarrow c_q^2$;
3. jeśli $n \neq 0$, to $12^n(M_{\langle q, n, n' \rangle}) = c_1 \rightarrow c_q^1$;
4. jeśli $n' \neq 0$, to $22^{n'}(M_{\langle q, n, n' \rangle}) = c_1 \rightarrow c_q^2$.

Dowód:

Po kolei dowodzimy poszczególne punkty.

1. Zgodnie z definicją $\text{plicznik}(0, c_q^1) = c_q^1$, a zatem $M_{\langle q, n, n' \rangle} = (c_0 \rightarrow c_q^1) \rightarrow M'$, co natychmiast daje nasze stwierdzenie.
2. Dowód analogiczny do poprzedniego przypadku zostawiamy dla Czytelnika.
3. Indukcja ze względu na n . Jeśli $n = 1$, to zgodnie z definicją mamy, że $\text{plicznik}(1, c_q^1) = c_1 \rightarrow c_q^1$, a zatem $M_{\langle q, n, n' \rangle} = (c_0 \rightarrow c_1 \rightarrow c_q^1) \rightarrow M'$, co natychmiast daje nasze stwierdzenie.

Jeśli $n > 1$, to z założenia indukcyjnego mamy, że $12^{n-1}(M_{\langle q, n-1, n' \rangle}) = 12^{n-1}((c_0 \rightarrow \text{plicznik}(n-1, c_q^1)) \rightarrow M')$ jest równe $c_1 \rightarrow c_q^1$, czyli

$$2^{n-2}(\text{plicznik}(n-1, c_q^1)) = c_1 \rightarrow c_q^1. \quad (2.10)$$

Ale $12^n(M_{\langle q,n,n' \rangle}) = 12^n((c_0 \rightarrow \text{plicznik}(n, c_q^1)) \rightarrow M') = 12^n((c_0 \rightarrow c_1 \rightarrow \text{plicznik}(n-1, c_q^1)) \rightarrow M')$, co korzystając z (2.10), daje

$$12^n(M_{\langle q,n,n' \rangle}) = c_1 \rightarrow c_q^1.$$

4. Dowód analogiczny do poprzedniego przypadku zostawiamy dla Czytelnika.

□

Lemat 2.1.18 (konfiguracja a operacje na licznikach)

Jeśli dla termu $c_0 \rightarrow \text{plicznik}(n, c_q^j)$ przy pewnym $j \in \{1, 2\}$ i stanie q zostanie wykonana redukcja zgodnie z regułą postaci:

1. $B_l^j \rightarrow_{\text{sym}} 1(B_l^j) \rightarrow c_1 \rightarrow c_p^j$, to wynikowy term ma postać $c_0 \rightarrow \text{plicznik}(n+1, c_p^j)$;
2. $B_l^j \rightarrow_{\text{sym}} c_1 \rightarrow c_p^j$, to wynikowy term ma postać $c_0 \rightarrow \text{plicznik}(n, c_p^j)$;
3. $B_l^j \rightarrow_{\text{sym}} c_p^j$, to wynikowy term ma postać $c_0 \rightarrow \text{plicznik}(n-1, c_p^j)$.

Dowód:

Pokażemy najpierw przez indukcję ze względu n , że jeśli term $\text{plicznik}(n, c_q^j)$ dla pewnego $j \in \{1, 2\}$ i stanu q zostanie wyredukowany regułami postaci:

1. $B_l^j \rightarrow_{\text{sym}} 1(B_l^j) \rightarrow c_1 \rightarrow c_p^j$, to wynikowy term ma postać $\text{plicznik}(n+1, c_p^j)$;
2. $B_l^j \rightarrow_{\text{sym}} c_1 \rightarrow c_p^j$, to wynikowy term ma postać $\text{plicznik}(n, c_p^j)$;
3. $B_l^j \rightarrow_{\text{sym}} c_p^j$, to wynikowy term ma postać $\text{plicznik}(n-1, c_p^j)$.

Jeśli $n = 1$, to $\text{plicznik}(1, c_q^j) = c_1 \rightarrow c_q^j$. Jediną możliwą postacią B_l^j jest $c_1 \rightarrow c_q^j$ (ponieważ c_0 nie występuje w $\text{plicznik}(1, c_q^j)$), czyli zachodzi dopasowanie się B_l^j do całego termu, stąd wynikiem redukcji jest odpowiednio

1. $c_1 \rightarrow c_1 \rightarrow c_p^j$ i wynikowy term ma postać $\text{plicznik}(2, c_p^j) = \text{plicznik}(n+1, c_p^j)$;
2. $c_1 \rightarrow c_p^j$ i wynikowy term ma postać $\text{plicznik}(1, c_p^j) = \text{plicznik}(n, c_p^j)$;
3. c_p^j i wynikowy term ma postać $\text{plicznik}(0, c_p^j) = \text{plicznik}(n-1, c_p^j)$.

Jeśli $n > 1$, to znowu jedyną możliwą postacią B_l^j jest $c_1 \rightarrow c_q^j$. Co więcej, miejsce dopasowania B_l^j do $\text{plicznik}(n, c_q^j)$ znajduje się w podtermie $2(\text{plicznik}(n, c_q^j))$, gdyż $1(\text{plicznik}(n, c_q^j))$ ma maksymalną długość ścieżki równą 0, a samo $\text{plicznik}(n, c_q^j)$ ma maksymalną długość ścieżki równą n , czyli większą od 1. Ponieważ $2(\text{plicznik}(n, c_q^j)) = \text{plicznik}(n-1, c_q^j)$, to korzystając z założenia indukcyjnego, dostajemy, że wynikiem redukcji termu $\text{plicznik}(n-1, c_q^j)$ jest

1. $\text{plicznik}(n, c_p^j)$, co daje z definicji $\text{plicznik}(\cdot)$, że wynikiem redukcji $\text{plicznik}(n, c_q^j)$ jest $\text{plicznik}(n+1, c_p^j)$;
2. $\text{plicznik}(n-1, c_p^j)$, co daje z definicji $\text{plicznik}(\cdot)$, że wynikiem redukcji $\text{plicznik}(n, c_q^j)$ jest $\text{plicznik}(n, c_p^j)$;
3. $\text{plicznik}(n-2, c_p^j)$, co daje z definicji $\text{plicznik}(\cdot)$, że wynikiem redukcji $\text{plicznik}(n, c_q^j)$ jest $\text{plicznik}(n-1, c_p^j)$.

Wracając do dowodu głównej naszej tezy, możemy założyć, że $n = 0$ lub, że $n > 0$. W tej pierwszej sytuacji badany przez nas term początkowy ma postać $c_0 \rightarrow c_q^j$. W tej sytuacji $B_l^j = c_0 \rightarrow c_q^j$, gdyż tylko taki term spośród możliwych dla B_l^j może się dopasować do naszego termu początkowego. I dalej, w zależności od drugiego termu l -tej reguły δ mamy, że wynikiem redukcji jest

1. $c_0 \rightarrow c_1 \rightarrow c_p^j$ i wynikowy term ma postać $c_0 \rightarrow \text{plicznik}(1, c_p^j) = c_0 \rightarrow \text{plicznik}(n+1, c_p^j)$;
2. $c_0 \rightarrow c_p^j$ i wynikowy term ma postać

$$c_0 \rightarrow \text{plicznik}(0, c_p^j) = \text{plicznik}(n, c_p^j);$$

3. ta sytuacja jest niemożliwa, bo operacja DEC jest niemożliwa przy teście Z.

Dla $n > 0$ wiadomo, że miejsce dopasowania B_l^j do $c_0 \rightarrow \text{plicznik}(n, c_q^j)$ znajduje się w podtermie $2(c_0 \rightarrow \text{plicznik}(n, c_q^j)) = \text{plicznik}(n, c_q^j)$, gdyż $1(c_0 \rightarrow \text{plicznik}(n, c_q^j))$ ma maksymalną długość ścieżki równą 0, a samo $c_0 \rightarrow \text{plicznik}(n, c_q^j)$ ma maksymalną długość ścieżki równą $n+1$, czyli jest większe od 1. Korzystając z udowodnionego na początku obecnego dowodu faktu, otrzymujemy, że

1. $\text{plicznik}(n, c_q^j)$ redukuje się do $\text{plicznik}(n+1, c_p^j)$, co daje, że $c_0 \rightarrow \text{plicznik}(n, c_q^j)$ redukuje się do $c_0 \rightarrow \text{plicznik}(n+1, c_p^j)$;
2. $\text{plicznik}(n, c_q^j)$ redukuje się do $\text{plicznik}(n, c_p^j)$, co daje, że term $c_0 \rightarrow \text{plicznik}(n, c_q^j)$ redukuje się do $c_0 \rightarrow \text{plicznik}(n, c_p^j)$;
3. $\text{plicznik}(n, c_q^j)$ redukuje się do $\text{plicznik}(n-1, c_p^j)$, co daje, że $c_0 \rightarrow \text{plicznik}(n, c_q^j)$ redukuje się do $c_0 \rightarrow \text{plicznik}(n-1, c_p^j)$.

□

Potrzebna jest jeszcze jedna definicja określająca, jakie ciągi termów będą odpowiadały obliczeniom.

Definicja 2.1.19 (symetryczny ciąg redukcyjny)

Powiemy, że ciąg termów M_0, M_1, \dots, M_n jest *symetrycznym ciągiem redukcyjnym*, jeśli

- dla każdego i mamy $M_i \rightarrow_{Sym} M_{i+1}$;
- istnieją takie n_1 i n_2 , że dla $j = 1, 2$ mamy $j2^{n_j}(M_0) = c_q^j$ dla pewnego q , oraz
- jeśli dla jakiegoś i mamy $M_i \rightarrow_{Sym} M_{i+1}$ zgodnie z regułą $B_l^j \rightarrow_{Sym} B_l^{j'}$, to albo redukcja $M_{i-1} \rightarrow_{Sym} M_i$ albo redukcja $M_{i+1} \rightarrow_{Sym} M_{i+2}$ odbywa się zgodnie z regułą $B_l^{j'} \rightarrow_{Sym} B_l^{j''}$, gdzie $j' = 3 - j$, a l jest numerem pewnej reguły w δ .

Twierdzenie 2.1.20 (ciągi redukcyjne a obliczenia)

Automat \mathcal{A} zatrzymuje się wtedy i tylko wtedy, gdy istnieje symetryczny ciąg redukcyjny

$$M_{\langle q_s, 0, 0 \rangle} = M_0, M_1, \dots, M_n = M_{\langle q_f, 0, 0 \rangle}.$$

Dowód:

(\Rightarrow) Załóżmy, że automat \mathcal{A} się zatrzymuje. Mamy wtedy skończony ciąg konfiguracji:

$$\langle q_s, 0, 0 \rangle = \langle q_0, n_0^1, n_0^2 \rangle \rightarrow_{\delta} \langle q_1, n_1^1, n_1^2 \rangle \rightarrow_{\delta} \dots \rightarrow_{\delta} \langle q_n, n_n^1, n_n^2 \rangle = \langle q_f, 0, 0 \rangle.$$

Zdefiniujemy teraz symetryczny ciąg redukcyjny dla \rightarrow_{Sym} , który będzie się zaczynał od $M_{\langle q_s, 0, 0 \rangle}$ i kończył na $M_{\langle q_f, 0, 0 \rangle}$. Załóżmy, że przejście od $\langle q_i, n_i^1, n_i^2 \rangle$ do $\langle q_{i+1}, n_{i+1}^1, n_{i+1}^2 \rangle$ odbywa się dzięki l -tej regule funkcji δ . Określimy term $M_{2i} = M_{\langle q_i, n_i^1, n_i^2 \rangle}$ i $M_{2i+1} = M_{\langle q_i, n_i^1, n_i^2 \rangle} [12^{n_i^1} \leftarrow B_l^1]$.

Pokażemy teraz, że $M_{2i} \rightarrow_{Sym} M_{2i+1}$ oraz $M_{2i+1} \rightarrow_{Sym} M_{2i+2}$. Ta pierwsza redukcja zachodzi zgodnie z regułą $B_l^1 \rightarrow_{Sym} B_l^1$ i niemalże bezpośrednio wynika z definicji M_{2i} i M_{2i+1} . Wystarczy tutaj pokazać, że $12^{n_i^1}(M_{2i}) = B_l^1$. Gdy $n_i^1 = 0$, wynika to z lematu 2.1.17(1). Gdy $n_i^1 > 0$, wynika to z lematu 2.1.17(3).

Dla dowodu drugiej redukcji wystarczy, że pokażemy równość $1M_{2i+1} = 1M_{2i+2}$, oraz że $2M_{2i+1} \rightarrow_{Sym} 2M_{2i+2}$ dzięki regule $B_l^2 \rightarrow_{Sym} B_l^2$ systemu redukcyjnego Sym . Przyjmijemy jeszcze, że jeśli l -ta reguła przejścia na j -tym liczniku wykonywała operację

- INC, to $\Delta^j = 1$;
- NOP, to $\Delta^j = 0$;
- DEC, to $\Delta^j = -1$.

Dla dowodu $1M_{2i+1} = 1M_{2i+2}$ zauważymy, że

$$1M_{2i} = c_0 \rightarrow \text{plicznik}(n_i^1, c_{q_i}^1).$$

Przy redukcji $M_{2i} \rightarrow_{Sym} M_{2i+1}$ zaś brała udział, jak już pokazaliśmy, reguła $B_l^1 \rightarrow_{Sym} B_l'^1$, przy czym ten ostatni term w zależności od wykonywanej w l -tej regule operacji na pierwszym liczniku jest równy: $1(B_l^1) \rightarrow c_1 \rightarrow c_{q_{i+1}}^1$ (dla INC), $c_1 \rightarrow c_{q_{i+1}}^1$ (dla NOP) i $c_{q_{i+1}}^1$ (dla DEC). W związku z tym zgodnie z lematem 2.1.18 term $1M_{2i+1} = c_0 \rightarrow$ plicznik($n_i^1 + \Delta^1, c_{q_{i+1}}^1$). Jednocześnie $n_{i+1}^1 = n_i^1 + \Delta^1$, stąd $1M_{2i+1} = c_0 \rightarrow$ plicznik($n_{i+1}^1, c_{q_{i+1}}^1$), ale zgodnie z definicją $M_{\langle q_{i+1}, n_{i+1}^1, n_{i+1}^2 \rangle}$, mamy $1M_{\langle q_{i+1}, n_{i+1}^1, n_{i+1}^2 \rangle} = c_0 \rightarrow$ plicznik($n_{i+1}^1, c_{q_{i+1}}^1$), co kończy dowód w tym przypadku.

Dla dowodu, że $2M_{2i+1} \rightarrow_{Sym} 2M_{2i+2}$ dzięki regule $B_l^2 \rightarrow_{Sym} B_l'^2$ systemu redukcyjnego Sym , wystarczy wykazać, że istnieje takie m , że zachodzą równości $22^m M_{2i+1} = B_l^2$ oraz $22^m M_{2i+2} = B_l'^2$. Pokażemy, że $m = n_i^2$ ma właśnie taką własność.

Dla dowodu pierwszej z tych własności zauważmy przede wszystkim, że $2M_{2i+1} = 2M_{2i}$, gdyż zgodnie z definicją ta gałąź termu M_{2i} nie jest zmieniana. W związku z tym będziemy się teraz zajmować pokazywaniem naszej własności dla $2M_{2i}$. Przy przechodzeniu od konfiguracji $\langle q_i, n_i^1, n_i^2 \rangle$ do konfiguracji $\langle q_{i+1}, n_{i+1}^1, n_{i+1}^2 \rangle$ użyliśmy sprawdzania, czy drugi licznik jest zerowy, lub niezerowy. W tym pierwszym wypadku zgodnie z definicją $B_l^2 = c_0 \rightarrow c_{q_i}^2$, zaś w tym drugim $B_l^2 = c_1 \rightarrow c_{q_i}^2$. Zgodnie z lematem 2.1.17(2) w pierwszym przypadku mamy $22^{n_i^2} M_{2i} = c_0 \rightarrow c_{q_i}^2 = B_l^2$, zaś w drugim zgodnie z lematem 2.1.17(4) — $22^{n_i^2} M_{2i} = c_1 \rightarrow c_{q_i}^2 = B_l^2$.

Dla dowodu $22^{n_i^2} M_{2i+2} = B_l'^2$ zauważymy, że $n_{i+1}^2 = n_{i+1}^2 + \Delta^2$ oraz zgodnie z definicją plicznik(\cdot) mamy $2(M_{2i+2}) = c_0 \rightarrow$ plicznik($n_i^2 + \Delta^2, c_{q_2}^j$). W związku z tym w zależności od wartości Δ^2 , a co za tym idzie wartości operacji na drugim liczniku:

1. dla INC — $22^{n_i^2}(M_{2i+2}) = B \rightarrow c_1 \rightarrow c_{q_{i+1}}^2$, przy czym dla wykonywanego na drugim liczniku testu Z mamy $B = c_0$, zaś dla testu NZ — $B = c_1$;
2. dla NOP — $22^{n_i^2}(M_{2i+2}) = c_1 \rightarrow c_{q_{i+1}}^2$;
3. dla DEC — $22^{n_i^2}(M_{2i+2}) = c_{q_{i+1}}^2$.

Zauważmy, że odpowiada to stosownie przypadkom (2.7), (2.8), (2.9) w definicji 2.1.16. Stąd dostajemy $22^{n_i^2}(M_{2i+2}) = B_l'^2$. To kończy dowód ostatniego przypadku dla (\Rightarrow).

(\Leftarrow) Załóżmy, że istnieje symetryczny ciąg redukcyjny

$$M_{\langle q_s, 0, 0 \rangle} = M_0, M_1, \dots, M_n = M_{\langle q_f, 0, 0 \rangle}.$$

Przez indukcję ze względu na i od 0 do $\lfloor n/2 \rfloor$ ($\lfloor \cdot \rfloor$ oznacza liczbę zaokrągloną do największej liczby całkowitej mniejszej od podanej w argumencie) pokażemy, że

- każde M_{2i} jest kodem pewnej konfiguracji $\langle q_i, n_i^1, n_i^2 \rangle$;
- jeśli $i > 0$, to redukcje $M_{2i-2} \rightarrow_{Sym} M_{2i-1}$ i $M_{2i-1} \rightarrow_{Sym} M_{2i}$ odbywają się dzięki regułom $B_l^1 \rightarrow_{Sym} B_l'^1$ i $B_l^2 \rightarrow_{Sym} B_l'^2$ dla pewnego l (niekoniecznie odpowiednio);
- jeśli $i > 0$, to dla tak zdefiniowanych konfiguracji mamy

$$\langle q_{i-1}, n_{i-1}^1, n_{i-1}^2 \rangle \rightarrow_{\delta} \langle q_i, n_i^1, n_i^2 \rangle.$$

Z tego zaś natychmiast wynika nasza teza.

Dla $i = 0$ mamy z postaci ciągu redukcji, że M_0 jest kodem $\langle q_s, 0, 0 \rangle$, czyli $q_0 = q_s, n_0^1 = 0$ i $n_0^2 = 0$. Dodatkowo do naszej tezy indukcyjnej dołożymy jeszcze, że dla dowolnego i jeśli dla $M_{2i-4} \rightarrow_{Sym} M_{2i-3}$ redukcja odbywa się zgodnie z regułą $B_l^j \rightarrow_{Sym} B_l'^j$, to redukcja $M_{2i-3} \rightarrow_{Sym} M_{2i-2}$ odbywa się zgodnie z regułą $B_l^{j'} \rightarrow_{Sym} B_l'^{j'}$, gdzie $j' = 3 - j$, a l jest numerem pewnej reguły w δ .

Założmy, że $i > 0$. Z założenia indukcyjnego mamy zdefiniowane konfiguracje dla $i - 1$. Wiemy w związku z tym, że $M_{2i-2} = M_{\langle q_i, n_i^1, n_i^2 \rangle}$. Z definicji symetrycznego ciągu redukcyjnego wiemy, że $M_{2i-2} \rightarrow_{Sym} M_{2i-1}$. Ponieważ każda reguła redukcji *Sym* odpowiada jakiejś regule δ , to możemy przyjąć, że wspomniana przed chwilą redukcja odpowiada pewnej regule l z δ . Mamy dwa przypadki:

- $i = 1$; tutaj redukcja $M_{2i-2} \rightarrow_{Sym} M_{2i-1}$ jest pierwszą redukcją w ciągu. Oznacza to jednak w świetle ostatniego warunku określającego symetryczny ciąg redukcyjny, że $M_{2i-1} \rightarrow_{Sym} M_{2i}$ odbywa się także według l -tej reguły δ .
- $i > 1$; z założenia indukcyjnego wiadomo, że redukcja $M_{2i-4} \rightarrow_{Sym} M_{2i-3}$ i redukcja $M_{2i-3} \rightarrow_{Sym} M_{2i-2}$ odbywają się zgodnie z l' -tą regułą δ . Przy czym l' jest różne od l (gdyż każda reguła przepisywania $B_l^j \rightarrow_{Sym} B_l'^j$ w *Sym* zmienia stałą z $2(B_l^j)$; ta zaś ostatnia własność wynika z umieszczonego w definicji 2.1.8 warunku (2) ograniczającego dopuszczalne maszyny dwulicznikowe). To ostatnie łącznie z ostatnim warunkiem określającym symetryczny ciąg redukcyjny oznacza, iż redukcja $M_{2i-1} \rightarrow_{Sym} M_{2i}$ odbywa się według l -tej reguły δ .

Wiemy zatem, że redukcje $M_{2i-2} \rightarrow_{Sym} M_{2i-1}$ i $M_{2i-1} \rightarrow_{Sym} M_{2i}$ odbywają się zgodnie z l -tą regułą funkcji δ . Dalszą część dowodu poprowadzimy dla sytuacji, gdy najpierw wykonywana jest redukcja $B_l^1 \rightarrow_{Sym} B_l'^1$, a potem redukcja $B_l^2 \rightarrow_{Sym} B_l'^2$, dowód dla przypadku gdy redukcje te są wykonywane w odwrotnej kolejności jest analogiczny.

W związku z tym, że term $2(M_{2i-2})$ nie zawiera wystąpienia żadnej stałej postaci c_q^1 , to redukcja zgodna z regułą $B_l^1 \rightarrow_{Sym} B_l'^1$ musi zachodzić w podtermie $1(M_{2i-2})$, który jest równy $c_0 \rightarrow$ plicznik($n_{i-1}^1, c_{q_{i-1}}^1$). W związku

z tym zgodnie z lematem 2.1.18 po otoczeniu $c_0 \rightarrow \text{plicznik}(n_{i-1}^1 + \Delta^1, c_{q_i}^1)$ kontekstem, takim jak miał $c_0 \rightarrow \text{plicznik}(n_{i-1}^1, c_{q_{i-1}}^1)$ w M_{2i-2} , dostajemy

$$M_{2i-1} = (c_0 \rightarrow \text{plicznik}(n_{i-1}^1 + \Delta^1, c_{q_i}^1)) \rightarrow 2(M_{2i-2})$$

($\Delta^1 = 1$, gdy w l -tej regule operacją na pierwszym liczniku jest INC; $\Delta^1 = 0$, gdy w l -tej regule operacją na pierwszym liczniku jest NOP; $\Delta^1 = -1$, gdy w l -tej regule operacją na pierwszym liczniku jest DEC).

W związku z tym, że term $1(M_{2i-1})$ ($= 1(M_{2i-2})$) nie zawiera wystąpienia żadnej stałej postaci c_q^2 , to redukcja zgodna z regułą $B_l^2 \rightarrow_{\text{sym}} B_l'^2$ musi zachodzić w podtermie $2(M_{2i-1})$, który jest równy $c_0 \rightarrow \text{plicznik}(n_{i-1}^2, c_{q_{i-1}}^2)$. W związku z tym zgodnie z lematem 2.1.18 po otoczeniu $c_0 \rightarrow \text{plicznik}(n_{i-1}^2 + \Delta^2, c_{q_i}^2)$ kontekstem takim, jak miał $c_0 \rightarrow \text{plicznik}(n_{i-1}^2, c_{q_{i-1}}^2)$ w M_{2i-1} , dostajemy

$$M_{2i} = (c_0 \rightarrow \text{plicznik}(n_{i-1}^1 + \Delta^1, c_{q_i}^1)) \rightarrow (c_0 \rightarrow \text{plicznik}(n_{i-1}^2 + \Delta^2, c_{q_i}^2))$$

($\Delta^2 = 1$, gdy w l -tej regule operacją na drugim liczniku jest INC; $\Delta^2 = 0$, gdy w l -tej regule operacją na drugim liczniku jest NOP; $\Delta^2 = -1$, gdy w l -tej regule operacją na drugim liczniku jest DEC).

Pokazaliśmy w takim razie, że M_{2i} jest kodem konfiguracji $\langle q_i, n_{i-1}^1 + \Delta^1, n_{i-1}^2 + \Delta^2 \rangle$ oraz że redukcje $M_{2i-2} \rightarrow_{\text{sym}} M_{2i-1}$ i $M_{2i-1} \rightarrow_{\text{sym}} M_{2i}$ odbywają się dzięki regułom $B_l^1 \rightarrow_{\text{sym}} B_l'^1$ i $B_l^2 \rightarrow_{\text{sym}} B_l'^2$ dla pewnego l . Pozostaje jeszcze przez chwilę zatrzymać się nad pokazaniem, że

$$\langle q_{i-1}, n_{i-1}^1, n_{i-1}^2 \rangle \rightarrow_{\delta} \langle q_i, n_{i-1}^1 + \Delta^1, n_{i-1}^2 + \Delta^2 \rangle,$$

co pozwoli nam zdefiniować $n_i^1 = n_{i-1}^1 + \Delta^1$ i $n_i^2 = n_{i-1}^2 + \Delta^2$. Oczywiście pokażemy, że przejście to odbywa się zgodnie z l -tą regułą δ . Termy $1(B_l^j)$ to stałe c_0 lub c_1 w zależności odpowiednio od tego, czy testujemy Z, czy NZ. Taki term dopasowuje się do $c_0 \rightarrow \text{plicznik}(n_{i-1}^j, c_{q_{i-1}}^j)$ tylko, gdy n_{i-1}^j jest odpowiednio równe zeru lub różne od zera. Następnie na liczniku wykonywana jest operacja. Zauważmy jednak, że wartości Δ^j są zdefiniowane zgodnie z operacjami wykonywanymi na licznikach. Stąd $n_{i-1}^j + \Delta^j$ jest rzeczywiście wynikiem wykonania wskazanej w l operacji na j -tym liczniku. \square

Warto zwrócić uwagę na fakt, że w dowodzie (\Rightarrow) korzystaliśmy w istotny sposób z tego, że maszyny dwulicznikowe, które redukujemy mają wspomnianą w definicji 2.1.8 własność (2).

Koordinacja

Jak do tej pory przedstawiliśmy konstrukcję tłumaczącą przepisywanie termów na unifikację oraz tłumaczącą obliczenia maszyny dwulicznikowej na przepisywanie w pewnym systemie przepisywania termów. Te elementy nie

wystarczają jednak jeszcze do pokazania nierozstrzygalności unifikacji — nie mamy na razie mechanizmu pozwalającego stwierdzić, że uzyskane w wyniku unifikacji przepisywanie rzeczywiście koduje obliczenie automatu dwulicznikowego. W dodatku samo pytanie, czy dane dwa termy są osiągalne jeden z drugiego przy stałym przepisywaniu jest rozstrzygalne ([BN98]). Do uzupełnienia całego zestawu konstrukcji brakuje jeszcze jednego elementu — zapewnienia, że ciąg redukcji, który uzyskamy w wyniku unifikacji i który ma za zadanie symulować obliczenia automatu dwulicznikowego, będzie symetrycznym ciągiem redukcyjnym — to wystarczy do zapewnienia tłumaczenia wyniku unifikacji na obliczenie. Niniejsza sekcja poświęcona jest opisowi pewnego dodatkowego systemu redukcyjnego, którego użycie zapewni nam, że uzyskany w dotychczasowych konstrukcjach ciąg redukcyjny będzie symetryczny.

Najpierw opiszemy termy, które będą reprezentowały ciągi kodów konfiguracji.

Definicja 2.1.21 (gąsienica (pseudo)konfiguracji)

Zbiór \mathfrak{G}^p , *gąsienic pseudokonfiguracji*, określamy rekurencyjnie tak:

- $c_3 \in \mathfrak{G}^p$;
- jeśli $M \in \mathfrak{G}^p$, to także $M \rightarrow c_2 \rightarrow N \in \mathfrak{G}^p$ dla dowolnego kodu pseudokonfiguracji N ,

przy czym stałe c_2 i c_3 są świeżymi stałymi nie występującymi w żadnych pseudokonfiguracjach.

Gąsienicę pseudokonfiguracji M , w której dla pewnego ciągu kodów pseudokonfiguracji N_1, \dots, N_n mamy $1^{i22}(M) = N_i$, oznaczamy w poniższych rozważaniach przez $[N_1, \dots, N_n]$.

Zbiór *gąsienic konfiguracji* (ozn. \mathfrak{G}) to podzbiór \mathfrak{G}^p , do którego należą tylko te gąsienice pseudokonfiguracji, których każdy podterm będący kodem pseudokonfiguracji jest kodem konfiguracji.

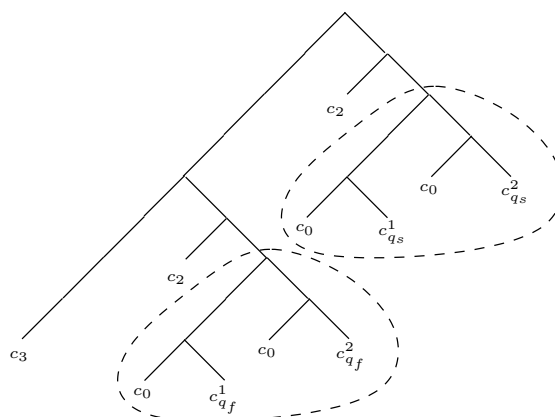
Zauważmy, że powyższa definicja odpowiada dokładnie temu, co uzyskujemy jako wynik zastosowania unifikatora uzyskanego w twierdzeniu 2.1.11 do równania z definicji 2.1.10. Stałym c i d z definicji 2.1.10 odpowiadają tutaj odpowiednio stałe c_3 i c_4 , termowi M_0 term odpowiadający kodowaniu konfiguracji początkowej, czyli $M_{\langle q_s, 0, 0 \rangle}$, zaś termowi M_k — term $M_{\langle q_k, 0, 0 \rangle}$ odpowiadający konfiguracji końcowej automatu dwulicznikowego.

Kolejną rzeczą, którą opiszemy jest stały system redukcyjny, który będzie sprawdzał, czy dana gąsienica pseudokonfiguracji jest gąsienicą konfiguracji.

Definicja 2.1.22 (system redukcyjny koordynacji)

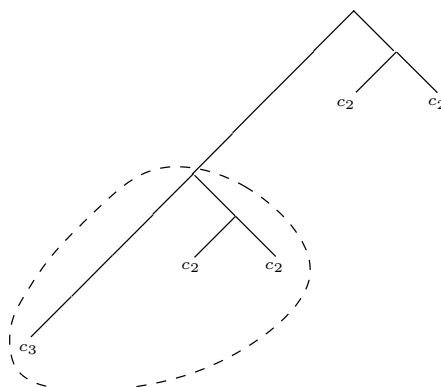
System redukcyjny koordynacji Ko zdefiniowany jest jako zestaw reguł przepisywania

$$\begin{aligned} C_i^j &\rightarrow_{Ko} C_i'^j, \\ D_i &\rightarrow_{Ko} D_i', \\ E &\rightarrow_{Ko} E', \end{aligned}$$



Rysunek 2.6: Po wyzerowaniu liczników

- Wreszcie zwróćmy uwagę, że mamy tu możliwość zmniejszania liczników niezależnie od tego, czy stany kodowane w stałych c_q^i dla różnych i , ale w tym samym kodzie pseudokonfiguracji, są takie same, czy różne.

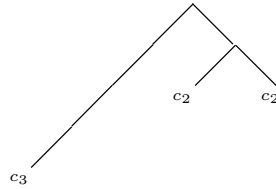


Rysunek 2.7: Po sprawdzeniu, że kody stanów na końcach liczników się zgadzają

Przyjrzyjmy się teraz sytuacji, jaką uzyskaliśmy w wyniku zastosowania pierwszego zestawu redukcji. Wygląd termu z naszego konkretnego przykładu uwidoczniony został na rysunku 2.6. Zauważmy, że otoczone przerywaną linią termy odpowiadają regułom redukcji zdefiniowanym w punkcie (2.12) definicji 2.1.22. Warto tutaj nadmienić, że wspomniane reguły dadzą się zastosować tylko, gdy stałe identyfikujące stan dla pierwszego i drugiego licznika reprezentują ten sam stan. To właśnie jest kluczowy moment rozpoznawania, czy pseudoopis jest rzeczywistym opisem. Jeśli stałe się nie

zgadzają, to w zredukowanym przez nas termie pozostanie nieredukowalny podterm postaci $(c_0 \rightarrow c_{q_1}^1) \rightarrow (c_0 \rightarrow c_{q_2}^2)$.

Wynikiem redukcji termu z rysunku 2.6 jest term uwidoczony na rysunku 2.7. Tutaj wszystkie miejsca, gdzie zgadzały się stany w kodach poszczególnych liczników, zamienione zostały na stałą c_2 . Zauważmy, że lewa dolna część tego ostatniego termu (otoczona linią przerywaną), pozwala teraz zastosować regułę (2.13). Wynikiem zastosowania tej reguły jest term z rysunku 2.8. Ten ostatni term można znowu zredukować korzystając z tej samej reguły. W końcu uzyskujemy samą stałą c_3 . W ten sposób można wyredukować dowolnej wielkości term, jednak cały proces się nie zablokuje tylko, jeśli wszystkie kody pseudokonfiguracji, które były umieszczone w gąsienicy, mogły zostać wyredukowane do c_2 , czyli tylko wtedy, gdy w rzeczywistości były kodami konfiguracji.



Rysunek 2.8: Po jednym kroku zmniejszania długości termu będącego wynikiem sprawdzania kodów stanów

Rozważania związane z powyższym przykładem możemy sformalizować w ramach następującego twierdzenia:

Twierdzenie 2.1.23 (koordynacja dla gąsienic pseudoopisów)

Niech M_0 będzie gąsienicą pseudokonfiguracji dla pewnego automatu. Term M_0 jest gąsienicą konfiguracji wtedy i tylko wtedy, gdy istnieje ciąg redukcji

$$M_0 \rightarrow_{Ko} M_1 \rightarrow_{Ko} \cdots \rightarrow_{Ko} M_n = c_3.$$

Dla dowodu powyższego twierdzenia potrzebne będą nam dwa lematy. Pierwszy z nich opisuje wykonywane w systemie Ko redukcje kodów pseudokonfiguracji.

Lemat 2.1.24 (koordynacja dla kodów pseudokonfiguracji)

Każdy term

$$M_0 = (c_0 \rightarrow \text{plicznik}(n_1, c_{q_1}^1)) \rightarrow (c_0 \rightarrow \text{plicznik}(n_2, c_{q_2}^2)),$$

gdzie q_1, q_2 są pewnymi stanami automatu, redukuje się w systemie redukcyjnym Ko do termu

$$(c_0 \rightarrow c_{q_1}^1) \rightarrow (c_0 \rightarrow c_{q_2}^2).$$

Dowód:

Najpierw przez indukcję ze względu na liczbę n pokażemy, że plicznik(n, c_q^i) redukuje się do c_q^i dla $i \in \{1, 2\}$.

Dla $n = 0$ z definicji plicznik(n, c_q^i) = c_q^i .

Dla $n > 0$, korzystając z założenia indukcyjnego, dostajemy redukcję plicznik($n - 1, c_q^i$) \rightarrow_{Ko} c_q^i . Ponieważ, plicznik(n, c_q^i) = $c_1 \rightarrow$ plicznik($n - 1, c_q^i$), otrzymujemy

$$\text{plicznik}(n, c_q^i) \rightarrow_{Ko} c_1 \rightarrow c_q^i.$$

Stosując odpowiednią regułę z zestawu (2.11) z definicji 2.1.22 dostajemy natychmiast

$$\text{plicznik}(n, c_q^i) \rightarrow_{Ko} c_1 \rightarrow c_q^i \rightarrow_{Ko} c_q^i.$$

Korzystając z powyższego faktu wykonujemy następujący ciąg redukcji:

$$\begin{aligned} M_0 &= (c_0 \rightarrow \text{plicznik}(n_1, c_{q_1}^1)) \rightarrow (c_0 \rightarrow \text{plicznik}(n_2, c_{q_2}^2)) \\ &\rightarrow_{Ko} (c_0 \rightarrow c_{q_1}^1) \rightarrow (c_0 \rightarrow \text{plicznik}(n_2, c_{q_2}^2)) \\ &\rightarrow_{Ko} (c_0 \rightarrow c_{q_1}^1) \rightarrow (c_0 \rightarrow c_{q_2}^2), \end{aligned}$$

który daje nam oczekiwaną tezę. \square

Drugi lemat mówi, że gdy potrafimy kod pseudokonfiguracji zredukować do stałej c_2 , to ten kod jest kodem konfiguracji.

Lemat 2.1.25 (pseudokonfiguracje, które są konfiguracjami)

Jeśli term

$$M_0 = (c_0 \rightarrow \text{plicznik}(n_1, c_{q_1}^1)) \rightarrow (c_0 \rightarrow \text{plicznik}(n_2, c_{q_2}^2))$$

redukuje się w systemie redukcyjnym Ko do stałej c_2 , to $q_1 = q_2$.

Dowód:

Zauważmy, na mocy lematu 2.1.24, że $M_0 \rightarrow_{Ko} (c_0 \rightarrow c_{q_1}^1) \rightarrow (c_0 \rightarrow c_{q_2}^2)$. Ten ostatni term redukuje się, zgodnie z definicją 2.1.22 do c_2 tylko, gdy stany q_1 i q_2 są równe. \square

Możemy teraz przystąpić do właściwego dowodu.

Dowód twierdzenia 2.1.23:

(\Rightarrow) Załóżmy, że M_0 jest gąsienicą konfiguracji dla pewnego automatu. Dowód przeprowadzimy przez indukcję ze względu na liczbę kodów pseudokonfiguracji w M_0 .

Jeśli w M_0 jest zero kodów konfiguracji, to $M_0 = c_3$ i w tej sytuacji ciąg redukcji składa się z jednego termu, właśnie z M_0 .

Jeśli w M_0 znajduje się $n > 0$ kodów konfiguracji, to korzystając z założenia indukcyjnego otrzymujemy ciąg redukcji

$$1(M_0) = M'_0 \rightarrow_{Ko} \cdots \rightarrow_{Ko} M'_{n'} = c_3. \quad (2.14)$$

Następnie, korzystając z lematu 2.1.24, otrzymujemy

$$22(M_0) = M_0'' \rightarrow_{K_0} \cdots \rightarrow_{K_0} M_{n''}'' = (c_0 \rightarrow c_{q_1}^1) \rightarrow (c_0 \rightarrow c_{q_2}^2) \quad (2.15)$$

Jednak $q_1 = q_2$, ponieważ M_0 jest gąsienicą opisów (a nie pseudoopisów). Możemy zatem wykonać redukcję

$$(c_0 \rightarrow c_{q_1}^1) \rightarrow (c_0 \rightarrow c_{q_2}^2) \rightarrow_{K_0} c_2 \quad (2.16)$$

zgodnie z regułami (2.11). Łącząc redukcje (2.14) z redukcjami (2.15), dostajemy:

$$M_0 = 1(M_0) \rightarrow 2(M_0) \rightarrow_{K_0} (c_3 \rightarrow c_2 \rightarrow 22(M_0)) \rightarrow_{K_0} (c_3 \rightarrow c_2 \rightarrow c_2).$$

Ten ostatni term może zostać zredukowany zgodnie z regułą (2.13) do c_3 , co daje żądaną tezę.

(\Leftarrow) Załóżmy, że istnieje ciąg redukcji

$$M_0 \rightarrow_{K_0} M_1 \rightarrow_{K_0} \cdots \rightarrow_{K_0} M_n = c_3.$$

Naszą tezę pokażemy przez indukcję ze względu na liczbę m kodów pseudo-konfiguracji, które znajdują się w gąsienicy M_0 .

Jeżeli $m = 0$, to $M_0 = c_3$, a zatem M_0 jest zgodnie z pierwszym punktem definicji 2.1.21 gąsienicą konfiguracji.

Jeżeli $m > 0$, to $M_{n-1} = c_3 \rightarrow c_2 \rightarrow c_2$, gdyż tylko reguła $c_3 \rightarrow c_2 \rightarrow c_2 \rightarrow_{K_0} c_3$ daje w wyniku stałą c_3 . Co więcej, w całej redukcji, oprócz ostatniego kroku, nie był wykonywany redeks wskazywany przez ε . Gdyby tak było, to któryś z termów M_i byłby równy c_2 lub $c_{q_i}^j$ dla pewnego q_i oraz j , a z takiego termu nie ma żadnego ciągu redukcji, który prowadziłby do $c_3 \rightarrow c_2 \rightarrow c_2$, gdyż każdy krok redukcji zmniejsza rozmiar termu. W związku z powyższym wszystkie redukcje w ciągu $M_0 \rightarrow_{K_0} \cdots \rightarrow_{K_0} M_{n-1}$ odbywają się albo w $1(M_i)$, albo w $2(M_i)$ dla $i = 1, \dots, n-1$. Z redukcji w termach $1(M_i)$ można utworzyć ciąg redukcji

$$1(M_0) \rightarrow_{K_0} 1(M_1) \rightarrow_{K_0} \cdots \rightarrow_{K_0} 1(M_{n-1}).$$

Ponieważ $1(M_{n-1}) = c_3$, możemy skorzystać z założenia indukcyjnego i użyć, że $1(M_0)$ jest gąsienicą konfiguracji. Aby dokończyć dowód, że M_0 jest gąsienicą konfiguracji, wystarczy pokazać, że $22(M_0)$ jest kodem konfiguracji.

Na mocy przeprowadzonych już wcześniej rozważań wiemy, że mamy redukcję

$$2(M_0) \rightarrow_{K_0} 2(M_1) \rightarrow_{K_0} \cdots \rightarrow_{K_0} 2(M_{n-1}) = c_2 \rightarrow c_2.$$

Ponieważ, $12(M_0) = c_2$, wszystkie ewentualne redukcje odbywają się w termach $22(M_i)$. Mamy zatem ciąg redukcji:

$$22(M_0) \rightarrow_{K_0} 22(M_1) \rightarrow_{K_0} \cdots \rightarrow_{K_0} 22(M_{n-1}) = c_2,$$

co na mocy lematu 2.1.25 daje, że $22(M_0)$ jest kodem konfiguracji. \square

Nierozstrzygalność problemu unifikacji

Możemy wreszcie podsumować nasze dotychczasowe rozważania i udowodnić najważniejszy wynik tej części pracy

Twierdzenie 2.1.26 (unifikacja z prostymi argumentami)

Problem unifikacji z prostymi argumentami jest nierozstrzygalny.

Dowód:

Pokażemy, że dla dowolnego automatu dwulicznikowego $\mathcal{A} = \langle Q, q_s, q_k, \delta \rangle$ potrafimy efektywnie skonstruować układ równań unifikacji drugiego rzędu z prostymi argumentami, który ma rozwiązanie wtedy i tylko wtedy, gdy \mathcal{A} się zatrzymuje. Na mocy twierdzenia 2.1.7 wyniknie z tego nierozstrzygalność tego rodzaju unifikacji. Ten układ to

$$\begin{aligned} FB_1^1 \dots B_{|\delta|}^1 B_1^2 \dots B_{|\delta|}^2 (c_3 \rightarrow c_2 \rightarrow M_{(q_k, 0, 0)}) &\doteq \\ &\doteq (FB_1^1 \dots B_{|\delta|}^1 B_1^2 \dots B_{|\delta|}^2 c_3) \rightarrow (c_2 \rightarrow M_{(q_s, 0, 0)}), \end{aligned} \quad (\mathcal{E}_1)$$

$$\begin{aligned} GC_1^1 \dots C_{|Q|}^1 C_1^2 \dots C_{|Q|}^2 D_1 \dots D_{|Q|} E (c_5 \rightarrow c_4 \rightarrow c_3) &\doteq \\ &\doteq (GC_1^1 \dots C_{|Q|}^1 C_1^2 \dots C_{|Q|}^2 D_1 \dots D_{|Q|} E' c_5) \rightarrow (c_4 \rightarrow N), \end{aligned} \quad (\mathcal{E}_2)$$

przy czym

- F i G są niewiadomymi;
- $N = FB_1^1 \dots B_{|\delta|}^1 B_1^2 \dots B_{|\delta|}^2 (c_3 \rightarrow c_2 \rightarrow M_{(q_k, 0, 0)})$;
- B_j^i wraz z B_j^i dla $i = 1, 2$ oraz $j = 1, \dots, |\delta|$ są określone jak w definicji 2.1.16;
- $C_j^i, C_j^i, D_j, D_j', E, E'$ dla $i = 1, 2$ oraz $j = 1, \dots, |Q|$ są określone jak w definicji 2.1.22;

Zauważmy tutaj, że równanie \mathcal{E}_1 jest równaniem redukcyjnym dla systemu *Sym* zaczynającym od $M_{(q_s, 0, 0)}$, a kończącym na $M_{(q_k, 0, 0)}$. Co więcej dla dowolnego stałego podstawienia S , takiego że $G \notin \text{dom}(S)$ wiemy, iż $S(\mathcal{E}_2)$ jest równaniem redukcyjnym dla systemu *Ko* zaczynającym od $S(N)$, a kończącym na c_3 . Możemy teraz przystąpić do właściwego dowodu.

(\Leftarrow) Załóżmy, że automat \mathcal{A} zatrzymuje się. Na mocy twierdzenia 2.1.20 istnieje wtedy symetryczny ciąg redukcyjny

$$M_{(q_s, 0, 0)} = M_0 \rightarrow_{Sym} M_1 \rightarrow_{Sym} \dots \rightarrow_{Sym} M_n = M_{(q_f, 0, 0)}. \quad (2.17)$$

Z powyższego ciągu można utworzyć gąsienicę $[M_0, \dots, M_n] = N_0$ (zob. notacja określona w definicji 2.1.21). Ponieważ wszystkie M_i są kodami konfiguracji, to na mocy twierdzenia 2.1.23 istnieje ciąg redukcyjny

$$N_0 \rightarrow_{Ko} N_1 \rightarrow_{Ko} \dots \rightarrow_{Ko} N_{n'} = c_3. \quad (2.18)$$

Na mocy twierdzenia 2.1.11, skoro istnieje ciąg (2.17), to równanie \mathcal{E}_1 ma unifikator S . Bez utraty ogólności możemy założyć, że $\text{dom}(S) = \{F\}$. Pozwala to na zastosowanie twierdzenia 2.1.13 do (2.18) i uzyskanie rozwiązania dla równania \mathcal{E}_2 .

(\Rightarrow) Załóżmy, że równania $\mathcal{E}_1, \mathcal{E}_2$ mają rozwiązanie S . Na mocy twierdzenia 2.1.11 mamy ciąg redukcji:

$$M_{\langle qs, 0, 0 \rangle} = M_0 \rightarrow_{Sym} M_1 \rightarrow_{Sym} \cdots \rightarrow_{Sym} M_n = M_{\langle qf, 0, 0 \rangle} \quad (2.19)$$

oraz na mocy twierdzenia 2.1.13 ciąg redukcji:

$$N_0 \rightarrow_{Ko} N_1 \rightarrow_{Ko} \cdots \rightarrow_{Ko} N_{n'} = c_3, \quad (2.20)$$

gdzie $N_0 = S(FB_1^1 \dots B_{|\delta|}^1 B_1^2 \dots B_{|\delta|}^2 (c_3 \rightarrow c_2 \rightarrow M_{\langle qk, 0, 0 \rangle}))$. Na mocy twierdzenia 2.1.11 term N_0 jest gąsienicą $[M'_0, \dots, M'_{n'}]$, przy czym $M'_i = 1^i 2 N_0$. W związku z tym z twierdzenia 2.1.23 zastosowanego do ciągu (2.20) wynika, iż $M'_0, \dots, M'_{n'}$ stanowią parzyste elementy pewnego symetrycznego ciągu redukcyjnego, a zatem na mocy twierdzenia 2.1.20 automat \mathcal{A} zatrzymuje się. \square

2.2 Unifikacja ze zmiennymi w pozycjach czołowych

Przeanalizujemy teraz inny ciekawy szczególny przypadek unifikacji drugiego rzędu — zdefiniowaną w końcu rozdziału 1, unifikację ze zmiennymi w pozycjach czołowych. Problem ten pozwoli nam stwierdzić rozstrzygalność kontekstowego problemu wyprowadzania typów dla pewnego podsystemu λP , który bezpośrednio odpowiada logice pierwszego rzędu. Systemem tym zajmujemy się bardziej szczegółowo w rozdziale 3.

Udowodnimy tutaj następujące twierdzenie

Twierdzenie 2.2.1 (rozstrzygalność unifikacji ze zmiennymi w pozycjach czołowych)

Istnieje niedeterministyczny algorytm rozwiązujący problem unifikacji ze zmiennymi w pozycjach czołowych (dla sygnatur z co najmniej jedną stałą) w czasie $2^{O(n \log n)}$.

Twierdzenie to udowodnimy, korzystając z serii faktów szacujących rozmiary rozwiązań istniejących tutaj równań.

Fakt 2.2.2 (równanie dopasowaniowe)

Weźmy pod uwagę równanie postaci $FM_1 \dots M_n \doteq M$, gdzie F jest zmienną wolną, zaś M_1, \dots, M_n, M są termami zamkniętymi drugiego rzędu. Jeśli S jest unifikatorem tego równania, to $S(F) = \lambda x_1 \dots x_n. N$ oraz każda ścieżka ω w N jest ścieżką w M .

Dowód:

Indukcja ze względu na k — maksymalną długość ścieżki w M .

Jeśli $k = 0$, to $M = c$ dla pewnej stałej c . W tej sytuacji $S(F) = \lambda \vec{x}.c$ lub $S(F) = \lambda x_1 \dots x_n.x_i$. W obydwu przypadkach teza naszego faktu zachodzi.

Jeśli $k > 0$, to $M = fN_1 \dots N_{n'}$ dla pewnego n' oraz termów zamkniętych drugiego rzędu $N_1, \dots, N_{n'}$. Mamy teraz możliwe dwie sytuacje:

1. $S(F) = \lambda x_1 \dots x_n.x_i$,
2. $S(F) = \lambda x_1 \dots x_n.fN'_1 \dots N'_{n'}$.

W pierwszym przypadku teza jest oczywista, bo w x_i istnieje tylko ścieżka ε .

W drugim przypadku założymy, że w $fN'_1 \dots N'_{n'}$ mamy ścieżkę $\omega \neq \varepsilon$. W takim razie $\omega = i\omega'$, gdzie $i \in \{1, \dots, n'\}$ oraz ω' jest ścieżką w N'_i . Określmy teraz równanie

$$F'M_1 \dots M_n \doteq N_i.$$

Równanie to ma rozwiązanie określone jak następuje $S'(F') = \lambda x_1 \dots x_n.N'_i$. Z założenia indukcyjnego, ponieważ ω' jest ścieżką w N'_i , to ω' jest ścieżką w N_i , a co za tym idzie $i\omega' = \omega$ jest ścieżką w $fN_1 \dots N_{n'} = N$. \square

Teraz wprowadzimy pewne pojęcie, które pomoże nam sformułować następną własność, która opisuje bardziej ogólną sytuację, z jaką mamy do czynienia w naszej unifikacji.

Definicja 2.2.3 (warstwy układu równań)

Dla danego układu równań E unifikacji ze zmiennymi w pozycjach czołowych określamy zbiór

$$W_0^E = \{M \doteq N \mid (M \doteq N) \in E \text{ oraz } N \text{ jest termem zamkniętym}\}$$

oraz rekurencyjnie na podstawie istniejącego już zbioru W_i^E zbiór

$$W_{i+1}^E = \{(FM_1 \dots M_m \doteq GN_1 \dots N_n) \in E \mid F \text{ nie występuje w } W_i^E, \\ G \text{ występuje w } W_i^E \text{ oraz } (FM_1 \dots M_m \doteq GN_1 \dots N_n) \notin \bigcup_{j \leq i} W_j^E\}.$$

O równaniach z W_i^E będziemy mówić, że należą do i -tej warstwy E . Przez W^E będziemy oznaczali $\bigcup_{i \in \mathbb{N}} W_i^E$.

Warto zauważyć, że $\bigcup_{i \in \mathbb{N}} W_i^E$ nie musi być równe E .

Fakt 2.2.4 (rozmiary rozwiązań w warstwach)

Niech F będzie zmienną występującą w i -tej warstwie układu równań E ,

a S unifikatorem E . Jeśli $S(F) = \lambda x_1 \dots x_n.M$, gdzie n jest arnością F , to liczba ścieżek w M jest ograniczona przez

$$\sum_{j=0}^i 2^j \prod_{k=0}^j m_k, \quad (2.21)$$

gdzie m_0 to maksymalna liczba ścieżek w termach ze zbioru A równego

$$\{N \mid N \text{ jest termem bez zmiennych oraz w } E \text{ istnieje równanie } N' \doteq N\},$$

zaś m_j dla $j > 0$ to maksymalna liczba ścieżek w termach ze zbioru

$$\{N_k \mid \text{istnieje } (M \doteq GN_1 \dots N_n) \in W_j^E \text{ oraz } k = 1, \dots, n\}.$$

Dowód:

Na początku zauważmy, że z istnienia zmiennej F w i -tej warstwie wynika, że zbiór A jest niepusty (jego pustość implikowałaby pustość warstwy 0).

Dalej dowód przeprowadzimy przez indukcję ze względu na i — numer warstwy.

Jeśli $i = 0$, to wzór (2.21) sprowadza się do m_0 . Zauważmy jednak, że na mocy faktu 2.2.2 dla każdego równania $FN_1 \dots N_k \doteq N$ z zerowej warstwy liczba ścieżek w M (określonym zgodnie ze wzorem $S(F) = \lambda x_1 \dots x_k.M$) jest ograniczona przez liczbę ścieżek w N , a co za tym idzie przez m_0 , co kończy dowód w tym przypadku.

Jeśli $i > 0$, to wykonujemy podstawienie S na zmiennych występujących w warstwie 0 i dostajemy nowy układ E' , który ma warstwy od 1 do $i - 1$ ma takie same jak E warstwy odpowiednio od 2 do i , warstwa zaś zerowa to warstwa 1 układu E ze zmiennymi z warstwy zerowej zastąpionymi zgodnie z podstawieniem S . Z założenia indukcyjnego mamy zatem

$$\sum_{j=0}^{i-1} 2^j \prod_{k=0}^j m'_k,$$

gdzie $m'_j = m_{j+1}$ dla $j > 0$ oraz $m'_0 \leq m_0 + m_0 m_1$. To ostatnie wynika z faktu, że po zastąpieniu zmiennej F z warstwy zerowej w E przez jej rozwiązanie z S dostajemy w równaniach pierwszej warstwy termy, w których liczba ścieżek ogranicza się przez liczbę ścieżek w F plus wszystkie ścieżki, które pochodzą z argumentów F . Tych pierwszych jest co najwyżej m_0 , bo rozwiązanie podstawiane na F (fakt 2.2.2). Tych drugich jest co najwyżej $m_0 m_1$, bo wszystkie ścieżki pochodzące z argumentów są sufiksami ścieżek

z rozwiązania dla F . Możemy zatem wykonać następujące przekształcenia:

$$\begin{aligned}
\sum_{j=0}^{i-1} 2^j \prod_{k=0}^j m'_k &= m'_0 + m'_0 \sum_{j=1}^{i-1} 2^j \prod_{k=1}^j m'_k \leq \\
&\leq m_0 + m_0 m_1 + m_0 \sum_{j=1}^{i-1} 2^j \prod_{k=1}^j m_{k+1} \\
&\quad + m_0 m_1 \sum_{j=1}^{i-1} 2^j \prod_{k=1}^j m'_k \leq \\
&\leq m_0 + m_0 m_1 + m_0 m_1 \sum_{j=1}^{i-1} 2^j \prod_{k=1}^j m_{k+1} \\
&\quad + m_0 m_1 \sum_{j=1}^{i-1} 2^j \prod_{k=1}^j m_{k+1} = \\
&= m_0 + m_0 m_1 + 2m_0 m_1 \sum_{j=1}^{i-1} 2^j \prod_{k=1}^j m_{k+1} \leq \\
&\leq m_0 + 2m_0 m_1 + m_0 m_1 \sum_{j=1}^{i-1} 2^{j+1} \prod_{k=1}^j m_{k+1} = \\
&= m_0 + 2m_0 m_1 + \sum_{j=1}^{i-1} 2^{j+1} \prod_{k=0}^{j+1} m_k = \\
&= m_0 + 2m_0 m_1 + \sum_{j=2}^i 2^j \prod_{k=0}^j m_k = \\
&= \sum_{j=0}^i 2^j \prod_{k=0}^j m_k.
\end{aligned}$$

□

Powyższy fakt pozwala natychmiast udowodnić nasze główne twierdzenie

Dowód Twierdzenia 2.2.1:

Na mocy faktu 2.2.4 dowolny unifikator S podstawia na zmienne z W^E dla zbioru równań E termy o liczbie ścieżek nie przekraczającej

$$m \frac{(2m)^n - 1}{2m - 1},$$

gdzie m jest maksymalną liczbą ścieżek w niezawierającym zmiennych wolnych podtermie naszego układu równań. Wynika to z faktu, że wszystkie m_i z faktu 2.2.4 majoryzują się przez m , a warstw nie może być więcej niż równań, czyli ich wielkość jest mniejsza od n będącego wielkością zadania. Ponieważ m też się majoryzuje przez n , więc liczba ta należy do klasy $2^{O(n \log n)}$. Z kolei na każdą zmienną F występującą w $E \setminus W^E$ wystarczy podstawić term postaci $\lambda x_1 \dots, x_n.c$, gdzie n jest arnością F , a c z góry ustaloną stałą w sygnaturze.

W związku z powyższym wystarczy, że nasz algorytm w niedeterministyczny sposób dla każdej zmiennej będzie generował term o liczbie pozycji nie przekraczającej wartości

$$n \frac{(2n)^n - 1}{2n - 1},$$

a następnie sprawdzał, czy tak uzyskane podstawienie jest unifikatorem. Ta pierwsza czynność może być wykonana w czasie $2^{O(n \log n)}$, gdyż zgadnięcie każdego symbolu wymaga stałego czasu, a liczba zmiennych jest ograniczona przez n . Druga czynność sprowadza się do wstawienia rozwiązań do równań i wykonania redukcji oraz porównania. Wykonanie redukcji wymaga tylko przejrzenia rozwiązania i wstawienia w odpowiednie miejsce wyniku, porównanie wymaga tylko przejrzenia obu stron równania. Wszystkie te czynności można wykonać w czasie $2^{O(n \log n)}$, co łącznie daje złożoność $2^{O(n \log n)}$. □

2.3 Unifikacja ze zmiennymi w pozycjach czołowych — trzeci rząd

Pokażemy tutaj redukcję problemu odpowiedności Posta do problemu unifikacji ze zmiennymi trzeciego rzędu w pozycjach czołowych. Problem ten jest o tyle interesujący, że wskazuje na granicę między problemami rozstrzygalnymi a nierozstrzygalnymi.

Problem 2.3.1 (problem odpowiedności Posta)

Dane: Dwa ciągi słów nad alfabetem $\{0, 1\}$

$$\begin{aligned} \omega_1, \dots, \omega_n, \\ \rho_1, \dots, \rho_n. \end{aligned}$$

Pytanie: Czy istnieje taki ciąg indeksów i_1, i_2, \dots, i_m ($m > 0$), że

$$\omega_{i_1}\omega_{i_2}\cdots\omega_{i_m} = \rho_{i_1}\rho_{i_2}\cdots\rho_{i_m}?$$

Wiadomo, że

Twierdzenie 2.3.2 (nierozstrzygalność problemu odpowiedności Posta)

Problem odpowiedności Posta jest nierozstrzygalny.

Dowód:

Zobacz [HU79]. □

Skonstruujemy na podstawie danych dwóch ciągów słów układ równań unifikacji ze zmiennymi trzeciego rzędu w pozycjach czołowych, którego rozwiązywalność jest równoważna znalezieniu ciągu indeksów Posta. Najpierw pokażemy translację słów.

Definicja 2.3.3 (translacja problemu Posta dla słów)

W naszej translacji będziemy używać dwóch symboli funkcyjnych f_0 i f_1 typu $\alpha \rightarrow \alpha$ dla jakiegoś ustalonego typu bazowego α . Słowo $\omega = w_1 \cdots w_n$ kodujemy jako

$$\hat{\omega} = \lambda x.f_{w_1}(\cdots(f_{w_n}(x))\cdots).$$

Teraz przedstawimy właściwy układ równań:

Definicja 2.3.4 (układ równań kodujący problem Posta)

Niech będą dane dwa ciągi słów: $\omega_1, \dots, \omega_n$ oraz ρ_1, \dots, ρ_n . Określimy układ równań $E_{\omega, \rho}$ jako

$$\begin{aligned} F\hat{\omega}_1 \dots \hat{\omega}_n &\doteq F\hat{\rho}_1 \dots \hat{\rho}_n, \\ F(\lambda x.x) \dots (\lambda x.x) &\doteq c, \\ F(\lambda x.d) \dots (\lambda x.d) &\doteq d. \end{aligned}$$

Uwaga: zakładamy tutaj obecność dwóch dodatkowych stałych c i d typu α .

Pokażemy teraz zasadniczą własność tej translacji:

Lemat 2.3.5 (translacja problemu Posta zapewnia równoważność)

Dla dowolnej instancji problemu Posta w postaci ciągów $\omega_1, \dots, \omega_n$ oraz ρ_1, \dots, ρ_n instancja ta ma rozwiązanie wtedy i tylko wtedy, gdy układ równań $E_{\omega, \rho}$ ma rozwiązanie.

Dowód:

(\Rightarrow) Załóżmy, że instancja problemu Posta złożona z ciągów $\omega_1, \dots, \omega_n$ oraz ρ_1, \dots, ρ_n ma rozwiązanie w postaci ciągu indeksów i_1, \dots, i_m . Określmy podstawienie S jako

$$S(F) = \lambda x_1 \dots x_n \cdot x_{i_1} (\dots (x_{i_n} c) \dots).$$

Pierwsze równanie $E_{\omega, \rho}$ jest spełnione przy tym podstawieniu, bo ciąg indeksów jest rozwiązaniem problemu Posta. Równanie drugie jest spełnione, gdyż po podstawieniu termów $\lambda x \cdot x$ na wszystkie zmienne i wyredukowaniu dostaniemy stałą c . Trzecie równanie jest spełnione, bo po podstawieniu na x_{i_1} termu $\lambda x \cdot d$ redukcja daje d .

(\Leftarrow) Załóżmy, że $E_{\omega, \rho}$ ma unifikator S . Taki unifikator daje $S(F) = \lambda x_1 \dots x_n \cdot M$. Term M nie jest termem stałym (bez zmiennych x_1, \dots, x_n), gdyż dla różnych argumentów daje różne wyniki (równanie 2. i 3.). Można założyć zatem, że ma on postać $\lambda x_1, \dots, x_n \cdot x_{i_1} (\dots (x_{i_m} (M')) \dots)$, gdzie M' nie zaczyna się od zmiennej. Ponieważ w drugim równaniu wynikiem jest c , term M' jest równy c . Pierwsze równanie implikuje w związku z tym, że $\omega_{i_1} \dots \omega_{i_m} = \rho_{i_1} \dots \rho_{i_m}$. \square

Twierdzenie 2.3.6 (nierozstrzygalność unifikacji trzeciego rzędu)

Unifikacja ze zmiennymi trzeciego rzędu w pozycjach czołowych jest nierozstrzygalna.

Dowód:

Na mocy lematu 2.3.5 można przetłumaczyć instancje problemu Posta na instancje problemu unifikacji ze zmiennymi trzeciego rzędu w pozycjach czołowych w taki sposób, że te ostatnie są rozwiązywalne wtedy i tylko wtedy, gdy istnieje rozwiązanie instancji problemu Posta. W związku z tym unifikacja nie może być rozstrzygalna, gdyż jej rozstrzygalność implikowałaby rozstrzygalność problemu Posta, co jest sprzeczne z twierdzeniem 2.3.2. \square

Rozdział 3

Wybrane systemy typów i unifikacja

W niniejszym rozdziale zajmiemy się zastosowaniem uzyskanej wiedzy na temat różnych podproblemów unifikacji drugiego rzędu do rozwiązywania zagadnień związanych z typami. Pokażemy tutaj, że wyprowadzanie typów w rachunku $\lambda 2$ w wersji Churcha jest nierozstrzygalne oraz, że kontekstowe wyprowadzanie typów w podsystemie rachunku λP odpowiadającym logice pierwszego rzędu jest rozstrzygalne.

3.1 Wyprowadzanie typów w rachunku $\lambda 2$ w wersji Churcha

W obecnym podrozdziale przedstawimy dowód nierozstrzygalności problemu wyprowadzania typów w rachunku $\lambda 2$ w wersji Churcha. Dowód przeprowadzimy przez sprowadzenie problemu unifikacji 2. rzędu z prostymi argumentami do problemu wyprowadzania typów. Redukcja ta będzie działać dla przypadku, gdy mamy dostępny przynajmniej jeden symbol stałej typu $\alpha \rightarrow \alpha \rightarrow \alpha$ oraz przynajmniej jeden symbol stałej typu α . Ten pierwszy symbol będziemy oznaczali tutaj przez znane nam już \rightarrow , zaś zbiór symboli stałych typu α przez Const_α . Translację redukującą określimy dla układów równań unifikacji 2. rzędu z prostymi argumentami, które składają się z pojedynczego równania. Nie zmniejsza to ogólności naszych rozważań, gdyż w obecności symboli wieloargumentowych można wiele równań złożyć w jedno kładąc termy równań w odpowiednie argumenty takich symboli. Dla danego układu równań E wynikiem kodowania będzie term Churcha $\text{kod}(E)$ o takiej własności, że E ma rozwiązanie wtedy i tylko wtedy, gdy dla pewnych Γ i τ osąd

$$\Gamma \vdash_{\lambda 2} \text{kod}(E) : \tau$$

można wyprowadzić w $\lambda 2$.

Opiszemy teraz kilka intuicji związanych z zastosowanym tutaj kodowaniem. Przede wszystkim nasze terminy będziemy kodować jako typy w rachunku $\lambda 2$. Symbol stałej \rightarrow będzie bezpośrednio kodowany jako strzałka typowa \rightarrow . Nasze kodowanie instancji będzie dobrane tak, że typy pewnych zmiennych przedmiotowych będą odpowiadały podterminom naszego zadania unifikacyjnego (w poniższej definicji zmienne z indeksem „spr”). Równość typów odpowiadających terminom po różnych stronach równania sprawdzana będzie dzięki następującemu mechanizmowi. Weźmy pod uwagę term

$$v_1(v_2M_1)(v_2M_2).$$

Działamy w rachunku w wersji Churcha, w związku z tym, skoro v_2 nie jest zaaplikowane do żadnego typu, to typ argumentu przy pierwszym wystąpieniu v_2 musi być taki sam, jak typ argumentu v_2 przy drugim wystąpieniu, czyli typ M_1 musi być równy typowi M_2 (w poniższej definicji rolę v_2 będą pełnił zmienne z indeksem „rwn”).

Kodowanie terminu z symbolem \rightarrow będzie się odbywało z wykorzystaniem mechanizmu widocznego w terminie:

$$v_1(v_2(v_3M_1))(v_2M_2).$$

Tutaj zmienna v_3 będzie miała typ $\tau_{M_1} \rightarrow \tau_{M_2}$, gdzie τ_{M_i} jest typem M_i dla $i = 1, 2$. Wynika to z tego, że w pierwszym argumentcie v_1 (terminie $v_2(v_3M_1)$) wymuszane jest, że argument zmiennej v_3 ma typ taki jak typ M_1 , zaś wynik taki, jak typ argumentu v_2 . W drugim argumentcie v_1 (terminie v_2M_2) wymuszane jest zaś, że typ argumentu v_2 jest taki, jak typ M_2 .

Kodowanie niewiadomej 2. rzędu będzie się odbywało za pomocą wolnej zmiennej przedmiotowej. W osądzie rachunku λ typ dla takiej zmiennej podawany jest w kontekście. W problemie wyprowadzania typu kontekst jest nieznan, co doskonale odpowiada sytuacji w równaniu unifikacyjnym — tam postać terminu podstawianego przez rozwiązanie na niewiadomą drugiego rzędu jest początkowo również nieznaną. Argumenty zmiennej drugiego rzędu są zakodowane jako typy, do których zaaplikowana jest zmienna przedmiotowa. Jeśli v_F koduje zmienną drugiego rzędu F , a argumenty tej zmiennej są bezpośrednio kodowane przez typy τ_1, \dots, τ_n , to $v_F\tau_1 \cdots \tau_n$ jest podterminem, którego typ (o ile jest wyprowadzalny) jest bezpośrednim tłumaczeniem wyniku zastosowania rozwiązania do terminu $F\tau_1 \cdots \tau_n$ (uwaga na drobne nadużycie notacji — typy τ_i utożsamiliśmy z ich odpowiednikami z symbolem \rightarrow , zob. `kodT()` w definicji 3.1.2).

Powyższe uwagi właściwie opisują główne idee związane z użytym tutaj kodowaniem. Sama definicja ma jeszcze pewne dodatkowe elementy, mają one jednak mniejsze znaczenie i są związane z koniecznością przeprowadzenia definicji rekurencyjnej po całym terminie.

Wprowadzimy najpierw pomocniczą definicję miłego podterminu.

Definicja 3.1.1 (miłe wystąpienie)

Powiemy, że *wystąpienie podtermu M termu N jest miłe w N* , jeśli M nie jest zmienną oraz na ścieżce wiodącej do tego wystąpienia nigdzie nie występuje zmienna 2. rzędu.

Oto zaś samo kodowanie:

Definicja 3.1.2 (kodowanie równania)

Niech będzie dane równanie drugiego rzędu z prostymi argumentami $M_1 \doteq M_2$ (oznaczać je będziemy także przez E). Wprowadźmy dwa rozłączne zbiory zmiennych typowych

$$G_L = \{\gamma_c \mid c \in \text{Const}_\alpha\}$$

oraz

$$A_L = \{\alpha_i \mid i \in \mathbb{N}\}.$$

Określmy kodowanie termów drugiego rzędu bez zmiennych oraz ze stałymi ze zbioru $\{\rightarrow\} \cup \text{Const}_\alpha$ jako:

- $\text{kodT}^X(N_1 \rightarrow N_2) = \text{kodT}^X(N_1) \rightarrow \text{kodT}^X(N_2)$;
- $\text{kodT}^X(c) = \gamma_c$ dla $c \in X$.

Obierzmy dwa zbiory parami różnych zmiennych przedmiotowych:

$$V_{\text{rwn}}^E = \{v_{\text{rwn}}^M \mid M \text{ jest miłym wystąpieniem w którymś z } M_i \text{ dla } i = 1, 2 \text{ lub zmienną}\},$$

$$V_{\text{spr}}^E = \{v_{\text{spr}}^M \mid M \text{ jest miłym wystąpieniem podtermu w którymś z } M_i \text{ dla } i = 1, 2\}.$$

Dodatkowo wprowadzimy jeszcze trzy pomocnicze zmienne przedmiotowe: z , v_{gl} i v_{para} . Kodowanie dla miłych wystąpień określamy jako:

- $\text{kodM}^E(FN_1 \dots N_n, v) = v_{\text{para}}(v_{\text{rwn}}^{FN_1 \dots N_n}(v_{\text{rwn}}^F \text{kodT}^{\text{Const}_\alpha}(N_1) \dots \text{kodT}^{\text{Const}_\alpha}(N_n))z)(v_{\text{rwn}}^{FN_1 \dots N_n}vz)$,

gdzie F jest zmienną drugiego rzędu o arności n ;

- $\text{kodM}^E(c, v) = v_{\text{para}}(v_{\text{rwn}}^c v_{\text{spr}}^c z)(v_{\text{rwn}}^c vz)$ gdzie $c \in \text{Const}_\alpha$;

- $\text{kodM}^E(N_1 \rightarrow N_2, v) =$

$$v_{\text{para}}(v_{\text{rwn}}^{N_1 \rightarrow N_2}(v_{\text{spr}}^{N_1})\text{kodM}^E(N_1, v_{\text{spr}}^{N_1}))(v_{\text{rwn}}^{N_1 \rightarrow N_2}v_{\text{spr}}^{N_2}\text{kodM}^E(N_2, v_{\text{spr}}^{N_2})).$$

Wreszcie określimy kod dla naszego równania E

$$\text{kod}(E) = \Lambda\alpha\gamma_{c_1} \dots \gamma_{c_m}.\lambda v_{\text{para}}:\alpha \rightarrow \alpha \rightarrow \alpha.\lambda z:\alpha.\lambda v_{\text{spr}}^{c_1}:\gamma_{c_1} \dots v_{\text{spr}}^{c_m}:\gamma_{c_m}.\lambda v_{\text{gl}}v_{\text{spr}}^{M_1}\text{kodM}^E(M_1, v_{\text{spr}}^{M_1})(v_{\text{gl}}v_{\text{spr}}^{M_2}\text{kodM}^E(M_2, v_{\text{spr}}^{M_2})),$$

gdzie $\{c_1, \dots, c_m\} = \text{Const}_\alpha$.

W poniższym tekście będziemy chcieli pokazać, że jeżeli zdefiniowany powyżej term $\text{kod}(E)$ ma typ, to jest nim

$$\forall \alpha \gamma_{c_1} \dots \gamma_{c_m}. (\alpha \rightarrow \alpha \rightarrow \alpha) \rightarrow \alpha \rightarrow \gamma_{c_1} \rightarrow \dots \rightarrow \gamma_{c_m} \rightarrow \alpha.$$

Powyższy typ będziemy w naszych rozważaniach oznaczali przez τ_Σ .

Przedstawimy teraz serię pomocniczych faktów, które pozwolą nam pokazać, że istnienie podstawienia dla instancji $E = \{M_1 \doteq M_2\}$ pozwala na określenie kontekstu, w którym $\text{kod}(E)$ będzie się typować.

Opis kontekstu wyznaczonego przez podstawienie wygląda tak:

Definicja 3.1.3 (kodowanie podstawienia)

Niech S będzie podstawieniem o własności takiej, że $FV(E) \subseteq \text{Dom}(S)$. Określimy

$$\begin{aligned} \Gamma'_S = & \{v_{\text{rwn}}^{N_1 \rightarrow N_2} : \text{kodT}^{\text{Const}_\alpha}(S(N_2)) \rightarrow \alpha \rightarrow \alpha \mid N_1 \rightarrow N_2 \text{ ma miłe} \\ & \text{wystąpienie w } M_1 \text{ lub } M_2\} \cup \\ & \cup \{v_{\text{rwn}}^{FN_1 \dots N_n} : \text{kodT}^{\text{Const}_\alpha}(S(FN_1 \dots N_n)) \rightarrow \alpha \rightarrow \alpha \mid F \in \text{Dom}(S) \text{ i} \\ & \text{ma arność } n \text{ oraz } FN_1 \dots N_n \text{ ma miłe wystąpienie w } M_1 \text{ lub } M_2\} \cup \\ & \cup \{v_{\text{rwn}}^F : \forall \gamma_{x_1} \dots \gamma_{x_n}. \text{kodT}^{\text{Const}_\alpha \cup X}(M) \mid S(F) = \lambda x_1 \dots x_n. M \text{ oraz} \\ & X = \{x_1, \dots, x_n\}\} \cup \\ & \cup \{v_{\text{rwn}}^c : \text{kodT}^{\text{Const}_\alpha}(c) \rightarrow \alpha \rightarrow \alpha \mid c \in \text{Const}_\alpha\} \cup \\ & \cup \{v_{\text{spr}}^N : \text{kodT}^{\text{Const}_\alpha}(S(N)) \mid N \text{ ma miłe wstąpienie w } M_1 \text{ lub } M_2\} \\ & \cup \{v_{\text{gl}} : \text{kodT}^{\text{Const}_\alpha}(S(M_1)) \rightarrow \alpha \rightarrow \alpha\}. \end{aligned}$$

Określimy też

$$\Gamma_S = \Gamma'_S \cup \{v_{\text{para}} : \alpha \rightarrow \alpha \rightarrow \alpha, z : \alpha\}.$$

Zauważmy, że dla każdych dwóch podstawień S_1, S_2 , które spełniają warunek wstępny niniejszej definicji mamy $\text{Dom}(\Gamma_{S_1}) = \text{Dom}(\Gamma_{S_2})$. Pozwala to na określenie $\text{Dom}_{\text{podst}} = \text{Dom}(\Gamma_{S_1})$.

Musimy udowodnić w tym momencie pewien techniczny fakt:

Fakt 3.1.4 (podstawienia w typach)

Dla $M = \lambda x_1 \dots x_n. M'$, gdzie M' jest termem 2. rzędu ze stałymi z Const_α nie zawierającym λ oraz $FV(M) = \emptyset$, mamy

$$\text{kodT}^{\text{Const}_\alpha}(\text{NF}(MN_1 \dots N_n)) = \text{kodT}^{\text{Const}_\alpha \cup Y}(M')[\vec{\gamma}_{x_i} := \vec{\tau}_i]$$

przy czym $\tau_i = \text{kodT}^{\text{Const}_\alpha}(N_i)$ i $Y = \{x_1, \dots, x_n\}$.

Dowód:

Indukcja po budowie termu M' .

- $M' = c$, gdzie $c \in \text{Const}_\alpha$. W tym przypadku

$$\begin{aligned}
\text{kodT}^{\text{Const}_\alpha}(\text{NF}(MN_1 \dots N_n)) &= \text{kodT}^{\text{Const}_\alpha}(c) \\
&\quad \text{(z definicji } M') \\
&= \gamma_c \\
&\quad \text{(z definicji } \text{kodT}(\cdot)) \\
&= \gamma_c[\vec{\gamma}_{x_i} := \vec{\tau}_i] \\
&\quad (\gamma_c \neq \gamma_{x_i}) \\
&= \text{kodT}^{\text{Const}_\alpha \cup Y}(M')[\vec{\gamma}_{x_i} := \vec{\tau}_i] \\
&\quad \text{(z definicji } M').
\end{aligned}$$

- $M' = x_j$, gdzie $x_j \in Y$. W tym przypadku

$$\begin{aligned}
\text{kodT}^{\text{Const}_\alpha}(\text{NF}(MN_1 \dots N_n)) &= \text{kodT}^{\text{Const}_\alpha}(\text{NF}(\lambda x_1 \dots x_n. x_i)N_1 \dots N_n) \\
&\quad \text{(z definicji } M') \\
&= \text{kodT}^{\text{Const}_\alpha}(N_i) \\
&\quad (\beta\text{-redukcja}) \\
&= \gamma_{x_j}[\vec{\gamma}_{x_i} := \vec{\tau}_i] \\
&\quad (\tau_j = \text{kodT}^{\text{Const}_\alpha}(N_j)) \\
&= \text{kodT}^{\text{Const}_\alpha \cup Y}(M')[\vec{\gamma}_{x_i} := \vec{\tau}_i] \\
&\quad \text{(z definicji } M').
\end{aligned}$$

- $M' = M'_1 \rightarrow M'_2$. Z prostej własności β -redukcji mamy

$$\begin{aligned}
\text{kodT}^{\text{Const}_\alpha}(\text{NF}(MN_1 \dots N_n)) &= \\
&\quad \text{kodT}^{\text{Const}_\alpha}(\text{NF}(M_1N_1 \dots N_n) \rightarrow \text{NF}(M_2N_1 \dots N_n)) \quad (3.1)
\end{aligned}$$

przy czym $M_j = \lambda x_1 \dots x_n. M'_j$. Z założenia indukcyjnego mamy

$$\text{kodT}^{\text{Const}_\alpha}(\text{NF}(M_jN_1 \dots N_n)) = \text{kodT}^{\text{Const}_\alpha \cup Y}(M'_j)[\vec{\gamma}_{x_i} := \vec{\tau}_i]$$

dla $j = 1, 2$. To zaś po zaaplikowaniu definicji $\text{kodT}(\cdot)$ do (3.1) daje

$$\begin{aligned}
\text{kodT}^{\text{Const}_\alpha}(\text{NF}(MN_1 \dots N_n)) &= \\
&= \text{kodT}^{\text{Const}_\alpha \cup Y}(M'_1)[\vec{\gamma}_{x_i} := \vec{\tau}_i] \rightarrow \text{kodT}^{\text{Const}_\alpha \cup Y}(M'_2)[\vec{\gamma}_{x_i} := \vec{\tau}_i],
\end{aligned}$$

co po zastosowaniu własności podstawienia i wspomnianej już własności β -redukcji daje natychmiast oczekiwane:

$$\begin{aligned}
\text{kodT}^{\text{Const}_\alpha}(\text{NF}(MN_1 \dots N_n)) &= \\
&= (\text{kodT}^{\text{Const}_\alpha \cup Y}(M'_1) \rightarrow \text{kodT}^{\text{Const}_\alpha \cup Y}(M'_2))[\vec{\gamma}_{x_i} := \vec{\tau}_i] = \\
&= \text{kodT}^{\text{Const}_\alpha \cup Y}(M')[\vec{\gamma}_{x_i} := \vec{\tau}_i].
\end{aligned}$$

□

W tym momencie możemy udowodnić lemat pozwalający określić, w jaki sposób unifikator pozwala na skonstruowanie typowania.

Lemat 3.1.5 (własność kodowania podstawienia)

Niech S będzie podstawieniem, takim że $FV(E) \subseteq \text{Dom}(S)$. Dla każdego podtermu N występującego w E mamy, że

$$\Gamma_S \vdash \text{kodM}^E(N, v_{\text{spr}}^N) : \alpha$$

jest wyprowadzalne w systemie $\lambda 2$ w wersji Churcha.

Dowód:

Indukcja ze względu na strukturę termu N .

- $N = c$ dla pewnego $c \in \text{Const}_\alpha$. W tym przypadku z definicji mamy $\text{kodM}^E(c, v_{\text{spr}}^c) = v_{\text{para}}(v_{\text{rwn}}^c v_{\text{spr}}^c z)(v_{\text{rwn}}^c v_{\text{spr}}^c z)$. Z definicji kontekstu kodowania mamy $\Gamma_S(v_{\text{spr}}^c) = \gamma_c$ oraz $\Gamma_S(z) = \alpha$. Ponieważ $\Gamma_S(v_{\text{rwn}}^c) = \gamma_c \rightarrow \alpha \rightarrow \alpha$, otrzymujemy

$$\Gamma_S \vdash v_{\text{rwn}}^c v_{\text{spr}}^c z : \alpha,$$

co, ze względu na fakt, że $\Gamma_S(v_{\text{para}}) = \alpha \rightarrow \alpha \rightarrow \alpha$, implikuje

$$\Gamma_S \vdash v_{\text{para}}(v_{\text{rwn}}^c v_{\text{spr}}^c z)(v_{\text{rwn}}^c v_{\text{spr}}^c z) : \alpha.$$

- $N = FN_1 \dots N_n$ dla pewnej zmiennej 2. rzędu F o arności n oraz dla N_1, \dots, N_n będących termami bez zmiennych. W tym przypadku

$$\text{kodM}^E(FN_1 \dots N_n, v_{\text{spr}}^{FN_1 \dots N_n}) = v_{\text{para}}(v_{\text{rwn}}^{FN_1 \dots N_n}(v_{\text{rwn}}^F \tau_1 \dots \tau_n)z) \\ (v_{\text{rwn}}^{FN_1 \dots N_n} v_{\text{spr}}^{FN_1 \dots N_n} z),$$

gdzie $\tau_i = \text{kodT}^{\text{Const}_\alpha}(N_i)$ dla $i = 1, \dots, n$. Postaramy się teraz uzasadnić, że ta całość typuje się do α . Zacznijmy analizę od pierwszego argumentu v_{para} . Z definicji kontekstu mamy

$$\Gamma_S(v_{\text{rwn}}^F) = \forall \gamma_{x_1} \dots \gamma_{x_n} . \text{kodT}^{\text{Const}_\alpha \cup X}(M),$$

przy czym $\lambda x_1 \dots x_n . M = S(F)$ oraz $X = \{x_1, \dots, x_n\}$. Oznacza to, że

$$\Gamma_S \vdash v_{\text{rwn}}^F \tau_1 \dots \tau_n : \text{kodT}^{\text{Const}_\alpha \cup X}(M)[\vec{\gamma}_{x_i} := \tau_i].$$

Pierwszy argument $v_{\text{rwn}}^{FN_1 \dots N_n}$ ma w kontekście Γ_S typ równy

$$\text{kodT}^{\text{Const}_\alpha}(S(FN_1 \dots N_n)) = \text{kodT}^{\text{Const}_\alpha}(S(F)N_1 \dots N_n) \\ (N_i \text{ nie mają zmiennych}) \\ = \text{kodT}^{\text{Const}_\alpha \cup X}(M)[\vec{\gamma}_{x_i} := \vec{\tau}_i] \\ (\text{z Faktu 3.1.4}).$$

Drugi argument $v_{\text{rwn}}^{FN_1 \dots N_n}$ to α . Ponieważ zaś z ma typ α , to aplikując omawiane tutaj termy do $v_{\text{rwn}}^{FN_1 \dots N_n}$, uzyskujemy

$$\Gamma_S \vdash v_{\text{rwn}}^{FN_1 \dots N_n}(v_{\text{rwn}}^F \tau_1 \dots \tau_n)z : \alpha.$$

Uzyskaliśmy w ten sposób typowanie dla pierwszego argumentu v_{para} . Przechodząc do typowania drugiego argumentu, zauważmy, że z definicji $\Gamma_S(v_{\text{spr}}^{FN_1 \dots N_n}) = \text{kodT}^{\text{Const}_\alpha}(S(FN_1 \dots N_n))$. Jak już jednak zauważyliśmy to ostatnie jest równe $\text{kodT}^{\text{Const}_\alpha \cup X}(M)[\vec{\gamma}_{x_i} := \vec{\tau}_i]$, w związku z czym natychmiast dostajemy

$$\Gamma_S \vdash v_{\text{rwn}}^{FN_1 \dots N_n} v_{\text{spr}}^{FN_1 \dots N_n} z : \alpha.$$

To kończy analizę typowania dla drugiego argumentu v_{para} . Oba argumenty typują się w Γ_S do α . Ponieważ z definicji $\Gamma_S(v_{\text{para}}) = \alpha \rightarrow \alpha$, dostajemy

$$\Gamma_S \vdash v_{\text{para}}(v_{\text{rwn}}^{FN_1 \dots N_n}(v_{\text{rwn}}^F \tau_1 \dots \tau_n)z)(v_{\text{rwn}}^{FN_1 \dots N_n} v_{\text{spr}}^{FN_1 \dots N_n} z) : \alpha.$$

- $N = N_1 \rightarrow N_2$ dla pewnych termów N_1, N_2 . W tym przypadku definicja określa

$$\text{kodM}^E(N_1 \rightarrow N_2, v_{\text{spr}}^{N_1 \rightarrow N_2}) = v_{\text{para}}(v_{\text{rwn}}^{N_1 \rightarrow N_2}(v_{\text{spr}}^{N_1 \rightarrow N_2} v_{\text{spr}}^{N_1})\text{kodM}^E(N_1, v_{\text{spr}}^{N_1})) \\ (v_{\text{rwn}}^{N_1 \rightarrow N_2} v_{\text{spr}}^{N_2}\text{kodM}^E(N_2, v_{\text{spr}}^{N_2})).$$

Z założenia indukcyjnego mamy

$$\Gamma_S \vdash \text{kodM}^E(N_i, v_{\text{spr}}^{N_i}) : \alpha$$

dla $i \in \{1, 2\}$. Z określenia typów $v_{\text{spr}}^{N_1 \rightarrow N_2}, v_{\text{spr}}^{N_1}$ w Γ_S uzyskujemy $\Gamma_S \vdash v_{\text{spr}}^{N_1 \rightarrow N_2} v_{\text{spr}}^{N_1} : \text{kodT}^{\text{Const}_\alpha}(S(N_2))$. Ponieważ w Γ_S typem $v_{\text{rwn}}^{N_1 \rightarrow N_2}$ jest $\text{kodT}^{\text{Const}_\alpha}(S(N_2)) \rightarrow \alpha \rightarrow \alpha$, dostajemy

$$\Gamma_S \vdash v_{\text{rwn}}^{N_1 \rightarrow N_2}(v_{\text{spr}}^{N_1 \rightarrow N_2} v_{\text{spr}}^{N_1})\text{kodM}^E(N_1, v_{\text{spr}}^{N_1}) : \alpha,$$

co stanowi typowanie dla pierwszego argumentu v_{para} . Dla drugiego argumentu v_{para} , korzystając z typowania dla $v_{\text{spr}}^{N_2}$ w Γ_S oraz ze wspomnianego już założenia indukcyjnego, dostajemy

$$\Gamma_S \vdash v_{\text{rwn}}^{N_1 \rightarrow N_2} v_{\text{spr}}^{N_2}\text{kodM}^E(N_2, v_{\text{spr}}^{N_2}) : \alpha.$$

Połączenie typowań dla argumentów v_{para} daje

$$\Gamma_S \vdash v_{\text{para}}(v_{\text{rwn}}^{N_1 \rightarrow N_2}(v_{\text{spr}}^{N_1 \rightarrow N_2} v_{\text{spr}}^{N_1})\text{kodM}^E(N_1, v_{\text{spr}}^{N_1})) \\ (v_{\text{rwn}}^{N_1 \rightarrow N_2} v_{\text{spr}}^{N_2}\text{kodM}^E(N_2, v_{\text{spr}}^{N_2})) : \alpha.$$

□

Jako wniosek dostajemy

Wniosek 3.1.6 (unifikator pozwala na typowanie)

Jeśli S jest unifikatorem E , to $\Gamma'_S \vdash \text{kod}(E) : \tau_\Sigma$ da się wyprowadzić w systemie $\lambda 2$.

Dowód:

Założmy, że $E = \{M_1 \doteq M_2\}$. Lemat 3.1.5 mówi, że

$$\Gamma'_S \vdash \text{kodM}^E(M_i, v_{\text{spr}}^{M_i}) : \alpha$$

dla $i = 1, 2$. Z definicji Γ_S typy dla v_{gl} i $v_{\text{spr}}^{M_i}$ pozwalają wyprowadzić

$$\Gamma_S \vdash v_{\text{gl}} v_{\text{spr}}^{M_i}\text{kodM}^E(M_i, v_{\text{spr}}^{M_i}) : \alpha.$$

To zaś implikuje

$$\Gamma_S \vdash v_{\text{para}}(v_{\text{gl}}v_{\text{spr}}^{M_1}\text{kodM}^E(M_1, v_{\text{spr}}^{M_1}))(v_{\text{gl}}v_{\text{spr}}^{M_2}\text{kodM}^E(M_2, v_{\text{spr}}^{M_2})) : \alpha,$$

co po dołożeniu lambda z definicji $\text{kod}(\cdot)$ daje dokładnie

$$\Gamma'_S \vdash \text{kod}(E) : \tau_\Sigma.$$

□

Teraz możemy przystąpić do pokazywania, że również istnienie wyprowadzenia pozwala na określenie unifikatora. Na początku przedstawimy, w jaki sposób będziemy przekształcać typy na termy.

Definicja 3.1.7 (od typu do termu)

Niech $n \geq 0$. I niech x_1, \dots, x_n będą dowolnymi zmiennymi. Określimy operację $\text{tkod}'(\langle x_1 \dots x_n \rangle, \tau)$. Nasze kodowanie typów będzie się składało z dwóch operacji:

$$\begin{aligned} \text{tkod}'(\langle x_1 \dots x_n \rangle, \gamma_c) &= c, \text{ gdy } c \in \text{Const}_\alpha; \\ \text{tkod}'(\langle x_1 \dots x_n \rangle, \alpha_i) &= x_i \text{ dla } i \in \{1, \dots, n\}; \\ \text{tkod}'(\langle x_1 \dots x_n \rangle, \forall \alpha. \tau) &= c_0 \text{ przy czym } c_0 \text{ jest wyróżnioną} \\ &\text{stałą}; \\ \text{tkod}'(\langle x_1 \dots x_n \rangle, \tau_1 \rightarrow \tau_2) &= \text{tkod}'(\langle x_1 \dots x_n \rangle, \tau_1) \rightarrow \\ &\text{tkod}'(\langle x_1 \dots x_n \rangle, \tau_2). \end{aligned}$$

Teraz możemy zdefiniować

$$\begin{aligned} \text{tkod}(\forall \alpha_1 \dots \alpha_n. \tau) &= \lambda x_1 \dots x_n. \text{tkod}'(\langle x_1 \dots x_n \rangle, \tau) \\ &\text{przy czym } \tau \text{ nie zaczyna się od} \\ &\text{kwantyfikatora, a } x_i \text{ dla} \\ &i = 1, \dots, n \text{ są świeżymi} \\ &\text{zmiennymi.} \end{aligned}$$

Podamy teraz określenie podstawienia, które będzie budowane na podstawie środowiska typowego.

Definicja 3.1.8 (konstrukcja podstawienia)

Dla danego środowiska Γ , o własności $\text{Dom}(\Gamma) \subseteq \text{Dom}_{\text{podst}}$, określimy podstawienie S_Γ jako

$$S_\Gamma(F) = \text{tkod}(\Gamma(v_{\text{rwn}}^F)),$$

o ile $v_{\text{rwn}}^F \in \text{Dom}(\Gamma)$.

Określimy jeszcze pomocniczy kontekst

$$\Delta = \{v_{\text{para}} : \alpha \rightarrow \alpha \rightarrow \alpha, z : \alpha\} \cup \{v_{\text{rwn}}^c : \gamma_c \rightarrow \alpha \rightarrow \alpha \mid c \in \text{Const}_\alpha\}.$$

Potrzebny nam będzie fakt określający zależność między translacjami tkod i kodT .

Fakt 3.1.9 (tkod i kodT)

Dla dowolnego termu bez zmiennych M mamy

$$\mathbf{tkod}(\mathbf{kodT}^{\mathbf{Const}_\alpha}(M)) = M.$$

Dowód:

Indukcja ze względu na budowę termu M .

- $M = c$, gdzie $c \in \mathbf{Const}_\alpha$. W tym przypadku mamy z definicji naszych transformacji $\mathbf{tkod}(\mathbf{kodT}^{\mathbf{Const}_\alpha}(c)) = \mathbf{tkod}(\gamma_c) = c$.
- $M = M_1 \rightarrow M_2$. W tym przypadku z definicji $\mathbf{kodT}(\cdot)$ mamy

$$\begin{aligned} \mathbf{tkod}(\mathbf{kodT}^{\mathbf{Const}_\alpha}(M_1 \rightarrow M_2)) &= \\ &= \mathbf{tkod}(\mathbf{kodT}^{\mathbf{Const}_\alpha}(M_1) \rightarrow \mathbf{kodT}^{\mathbf{Const}_\alpha}(M_2)). \end{aligned}$$

Z kolei z definicji $\mathbf{tkod}(\cdot)$ mamy

$$\begin{aligned} \mathbf{tkod}(\mathbf{kodT}^{\mathbf{Const}_\alpha}(M_1) \rightarrow \mathbf{kodT}^{\mathbf{Const}_\alpha}(M_2)) &= \\ &= \mathbf{kodT}^{\mathbf{Const}_\alpha}(M_1) \rightarrow \mathbf{kodT}^{\mathbf{Const}_\alpha}(M_2). \end{aligned}$$

Z założenia indukcyjnego mamy

$$\begin{aligned} \mathbf{tkod}(\mathbf{kodT}^{\mathbf{Const}_\alpha}(M_1)) &= M_1, \\ \mathbf{tkod}(\mathbf{kodT}^{\mathbf{Const}_\alpha}(M_2)) &= M_2, \end{aligned}$$

co pozwala nam zakończyć szereg równości:

$$\mathbf{kodT}^{\mathbf{Const}_\alpha}(M_1) \rightarrow \mathbf{kodT}^{\mathbf{Const}_\alpha}(M_2) = M_1 \rightarrow M_2 = M.$$

□

Udowodnimy teraz pewien techniczny fakt dotyczący podstawień.

Fakt 3.1.10 (podstawienia w typach - 2. spojrzenie)

Niech v_{rwn}^F ma typ $\forall \gamma_{x_1} \dots \gamma_{x_n} . \tau$, gdzie τ nie zaczyna się od \forall . Dla każdej ścieżki ω w τ mamy

$$\mathbf{tkod}(\omega(\tau)[\vec{\gamma}_{x_i} := \mathbf{kodT}^{\mathbf{Const}_\alpha}(N_i)]) = \mathbf{NF}(MN_1 \dots N_n),$$

gdzie $M = \lambda x_1 \dots x_n . \omega(M_0)$ i $S_\Gamma(F) = \lambda x_1 \dots x_n . M_0$.

Dowód:

Indukcja ze względu na budowę $\omega(\tau)$.

- $\omega(\tau) = \gamma_c$, gdzie $c \in \text{Const}_\alpha$. Wtedy

$$\begin{aligned}
\text{tkod}(\omega(\tau)[\vec{\gamma}_{x_i} := \text{kodT}^{\vec{\text{Const}}_\alpha}(N_i)]) &= \text{tkod}(\gamma_c) \\
&\quad \text{(definicja podstawienia)} \\
&= \text{tkod}'(\langle \rangle, \gamma_c) \\
&\quad \text{(definicja tkod}(\cdot)) \\
&= c \\
&\quad \text{(definicja tkod}'(\cdot)) \\
&= (\lambda \vec{x}_i. c) N_1 \dots N_n \\
&\quad \text{(\beta-redukcja)} \\
&= \text{NF}(S_\Gamma(F) N_1 \dots N_n) \\
&\quad \text{(definicja } S_\Gamma).
\end{aligned}$$

- $\omega(\tau) = \gamma_{x_j}$. W tym przypadku

$$\begin{aligned}
\text{tkod}(\tau[\vec{\gamma}_{x_i} := \text{kodT}^{\vec{\text{Const}}_\alpha}(N_i)]) &= \text{tkod}(\text{kodT}^{\vec{\text{Const}}_\alpha}(N_j)) \\
&\quad \text{(definicja podstawienia)} \\
&= N_j \\
&\quad \text{(fakt 3.1.9)} \\
&= (\lambda \vec{x}_i. x_j) N_1 \dots N_n \\
&\quad \text{(\beta-redukcja)} \\
&= \text{tkod}(\forall \gamma_{x_1} \dots \gamma_{x_n}. \gamma_{x_j}) N_1 \dots N_n \\
&\quad \text{(definicja tkod}(\cdot)) \\
&= \text{NF}(S_\Gamma(F) N_1 \dots N_n) \\
&\quad \text{(definicja } S_\Gamma).
\end{aligned}$$

- $\tau = \forall \alpha. \tau'$. Ten przypadek nie zachodzi, gdyż z założenia τ nie zaczyna się od \forall .
- $\tau = \tau_1 \rightarrow \tau_2$. W tym przypadku

$$\begin{aligned}
\text{tkod}(\tau[\vec{\gamma}_{x_i} := \text{kodT}^{\vec{\text{Const}}_\alpha}(N_i)]) &= \\
&\quad \text{(definicja podstawienia)} \\
&= \text{tkod}(\tau_1[\vec{\gamma}_{x_i} := \text{kodT}^{\vec{\text{Const}}_\alpha}(N_i)] \rightarrow \tau_2[\vec{\gamma}_{x_i} := \text{kodT}^{\vec{\text{Const}}_\alpha}(N_i)]) = \\
&\quad \text{(definicja tkod}(\cdot)) \\
&= \text{tkod}(\tau_1[\vec{\gamma}_{x_i} := \text{kodT}^{\vec{\text{Const}}_\alpha}(N_i)]) \rightarrow \text{tkod}(\tau_2[\vec{\gamma}_{x_i} := \text{kodT}^{\vec{\text{Const}}_\alpha}(N_i)])
\end{aligned}$$

□

Lemat 3.1.11 (własność kodowania kontekstu)

Niech M ma miłe wystąpienie w M_1 lub M_2 . Dla każdego środowiska Γ oraz typu α , takich że

$$\Gamma, \Delta \vdash \text{kodM}^E(M, v_{\text{spr}}^M) : \alpha$$

jest wyprowadzalne w systemie $\lambda 2$, mamy $\text{tkod}(\Gamma(v_{\text{spr}}^M)) = S_\Gamma(M)$.

Dowód:

Indukcja ze względu na budowę M .

- $M = c$ dla pewnej stałej $c \in \text{Const}_\alpha$. Ponieważ v_{rwn}^c ma w Δ typ $\gamma_c \rightarrow \alpha \rightarrow \alpha$ oraz ze względu na postać $\text{kodM}^E(c, v_{\text{spr}}^c)$ uzyskujemy, że v_{spr}^c ma typ γ_c w Γ . To implikuje $\text{tkod}(\Gamma(v_{\text{spr}}^c)) = c = S_\Gamma(c)$.
- $M = N_1 \rightarrow N_2$ dla pewnych termów N_1 i N_2 . Ponieważ termy $\text{kodM}^E(N_i, v_{\text{spr}}^{N_i})$ dla $i \in \{1, 2\}$ są podtermami $\text{kodM}^E(M, v_{\text{spr}}^M)$, istnieją τ_i , takie że

$$\Gamma, \Delta \vdash \text{kodM}^E(N_i, v_{\text{spr}}^{N_i}) : \tau_i$$

dla $i = 1, 2$. Ponieważ $\text{kodM}^E(P, v) = v_{\text{para}}P_1P_2$ dla dowolnego termu P oraz zmiennej v oraz odpowiednich P_1 i P_2 oraz ponieważ $\Delta(v_{\text{para}}) = \alpha \rightarrow \alpha \rightarrow \alpha$, uzyskujemy, że $\tau_i = \alpha$ dla $i = 1, 2$. Założenie indukcyjne daje nam równość $\text{tkod}(\Gamma(v_{\text{spr}}^{N_i})) = S_\Gamma(N_i)$ dla $i = 1, 2$, więc rzeczywiście w niniejszym przypadku $\text{tkod}(\Gamma(v_{\text{spr}}^M)) = S_\Gamma(M)$.

- $M = FN_1 \dots N_n$ dla pewnego n -arnego symbolu zmiennej F . Ponieważ zmienna v_{rwn}^F jest zgodnie z definicją przykładana w termie $\text{kodM}^E(M, v_{\text{spr}}^M)$ do ciągu typów $\text{kodT}^{\text{Const}_\alpha}(N_1), \dots, \text{kodT}^{\text{Const}_\alpha}(N_n)$, typ v_{rwn}^F ma postać $\forall \gamma_{x_1} \dots \gamma_{x_n} \tau$ dla pewnego typu τ oraz zmiennych typowych $\gamma_{x_1}, \dots, \gamma_{x_n}$. W związku z tym, że w pozycji pierwszego argumentu v_{rwn}^M występuje v_{spr}^M oraz term

$$v_{\text{rwn}}^F \text{kodT}^{\text{Const}_\alpha}(N_1) \dots \text{kodT}^{\text{Const}_\alpha}(N_n),$$

to typ v_{spr}^M jest równy $\tau[\vec{\gamma}_{x_i} := \text{kodT}^{\text{Const}_\alpha}(N_i)]$. Z faktu 3.1.10 wiemy, że

$$\text{tkod}(\tau[\vec{\gamma}_{x_i} := \text{kodT}^{\text{Const}_\alpha}(N_i)]) = \text{NF}(S_\Gamma(F)N_1 \dots N_n),$$

co w związku z założeniem, że N_i dla $i = 1, \dots, n$ nie zawierają zmiennych, natychmiast daje nasz wynik: $\text{tkod}(\Gamma(v_{\text{spr}}^M)) = S_\Gamma(M)$.

□

Znowu możemy podsumować powyższe rozważania użytecznym wnioskiem:

Wniosek 3.1.12 (typowalność zapewnia istnienie unifikatora)

Jeśli istnieje takie środowisko Γ oraz typ τ , że $\Gamma \vdash \text{kod}(E) : \tau$ da się wyprowadzić w systemie $\lambda 2$, to S_Γ jest unifikatorem E .

Dowód:

Postać $\text{kod}(E)$ implikuje, że typy $v_{\text{spr}}^{M_1}$ oraz $v_{\text{spr}}^{M_2}$ są równe (zmiennie te są umieszczone jako pierwsze argumenty v_{gl}). Mamy zatem

$$\begin{aligned} S_\Gamma(M_1) &= \text{tkod}(\Gamma(v_{\text{spr}}^{M_1})) = && \text{(lemat 3.1.11)} \\ &= \text{tkod}(\Gamma(v_{\text{spr}}^{M_2})) = && \text{(typy } v_{\text{spr}}^{M_1}, v_{\text{spr}}^{M_2} \text{ równe)} \\ &= S_\Gamma(M_2) && \text{(lemat 3.1.11)} \end{aligned}$$

□

Twierdzenie 3.1.13 (nierozstrzygalność wyprowadzania typów)

Problem wyprowadzania typów w systemie $\lambda 2$ w wersji Churcha jest nierozstrzygalny.

Dowód:

Wniosek 3.1.6 oraz wniosek 3.1.12 implikują, że dla danej instancji E problemu unifikacji istnieje term $\text{kod}(E)$ systemu $\lambda 2$ w wersji Churcha, taki że E ma unifikator wtedy i tylko wtedy, gdy istnieje środowisko Γ oraz typ τ , takie że osąd

$$\Gamma \vdash \text{kod}(E) : \tau$$

jest wyprowadzalny w $\lambda 2$. Istnienie takiej translacji oraz twierdzenie 2.1.26 implikują, że problem wyprowadzania typów w systemie $\lambda 2$ w wersji Churcha jest nierozstrzygalny. \square

Uwaga 3.1.14 Warto zwrócić na marginesie uwagę na fakt, że zaprezentowana tutaj konstrukcja działa także dla predykatywnej wersji systemu $\lambda 2$ wprowadzonej przez Leivanta w [Lei91]. Wszystkie typy, które są tutaj podstawiane na zmienne typowe nie mają kwantyfikatorów, co odpowiada randze 0 typów Leivanta.

3.2 Kontekstowe wyprowadzanie typów dla logiki pierwszego rzędu

W obecnym podrozdziale przedstawimy zanurzenie logiki pierwszego rzędu w rachunku λP . Następnie przedstawimy dwa sposoby rozumienia problemu kontekstowego wyprowadzania typów dla uzyskanego, odpowiadającego logice pierwszego rzędu fragmentu systemu λP . Pokażemy, że te dwa sposoby różnią się na tyle istotnie, że wyprowadzanie typów dla pierwszego z nich jest nierozstrzygalne, zaś dla drugiego da się sprowadzić do problemu unifikacji ze zmiennymi w pozycjach czołowych, o którym wiemy, że jest rozstrzygalny z twierdzenia 2.2.1.

Będziemy się tutaj posługiwać zmodyfikowaną wersją rachunku λP . Jego reguły dla wersji Curry'ego przedstawione zostały na rysunku 3.1. Rachunek ten będzie wygodniejszy przy określaniu zanurzenia logiki pierwszego rzędu w λP . Podobne przedstawienie tego rachunku znajdziemy na przykład w pracy [Dow93].

Główna różnica między tym rachunkiem a rachunkiem zdefiniowanym w podrozdziale 1.2 polega na tym, że w tamtym rachunku, były obecne reguły osłabiania, których tutaj nie ma. Z drugiej strony mamy tutaj silniejsze reguły (kd-var), (kd-abs), (typ-abs) oraz (var) oraz jawne asercje stwierdzające poprawność kontekstu.

Przedstawimy tutaj jeszcze dowód, że powyżej zdefiniowany system pozwala wyprowadzać takie same typy, co system zdefiniowany w definicji 1.2.8.

$$\begin{array}{c}
\text{(type)} \frac{\Gamma \vdash_{\lambda P} \text{poprawny}}{\Gamma \vdash_{\lambda P} * : \square} \\
\\
\text{(kd-abs)} \frac{\Gamma_1, x : \tau, \Gamma_2 \vdash_{\lambda P+} \kappa : \square \quad \Gamma_1, \Gamma_2 \vdash_{\lambda P+} \text{poprawny}}{\Gamma_1, \Gamma_2 \vdash_{\lambda P+} (\Pi x : \tau) \kappa : \square} \\
\\
\text{(kd-var)} \frac{\Gamma_1, \alpha : \kappa, \Gamma_2 \vdash_{\lambda P+} \text{poprawny}}{\Gamma_1, \alpha : \kappa, \Gamma_2 \vdash_{\lambda P+} \alpha : \kappa} \\
\\
\text{(kd-app)} \frac{\Gamma \vdash_{\lambda P+} \phi : (\Pi x : \tau) \kappa \quad \Gamma \vdash_{\lambda P+} M : \tau}{\Gamma \vdash_{\lambda P+} \phi M : \kappa[x := M]} \\
\\
\text{(typ-abs)} \frac{\Gamma_1, x : \tau, \Gamma_2 \vdash_{\lambda P+} \sigma : * \quad \Gamma_1, \Gamma_2 \vdash_{\lambda P+} \text{poprawny}}{\Gamma_1, \Gamma_2 \vdash_{\lambda P+} (\forall x : \tau) \sigma : *} \\
\\
\text{(popr-zero)} \quad \vdash_{\lambda P+} \text{poprawny} \\
\\
\text{(popr-kd)} \frac{\Gamma \vdash_{\lambda P+} \text{poprawny} \quad \Gamma \vdash_{\lambda P+} \kappa : \square}{\Gamma, \alpha : \kappa \vdash_{\lambda P+} \text{poprawny}} (\alpha \notin \text{Dom}(\Gamma)) \\
\\
\text{(popr-typ)} \frac{\Gamma \vdash_{\lambda P+} \text{poprawny} \quad \Gamma \vdash_{\lambda P+} \tau : *}{\Gamma, x : \tau \vdash_{\lambda P+} \text{poprawny}} (x \notin \text{Dom}(\Gamma)) \\
\\
\text{(var)} \frac{\Gamma_1, x : \tau, \Gamma_2 \vdash_{\lambda P+} \text{poprawny}}{\Gamma_1, x : \tau, \Gamma_2 \vdash_{\lambda P+} x : \tau} \\
\\
\text{(app)} \frac{\Gamma \vdash_{\lambda P+} N : (\forall x : \tau) \sigma \quad \Gamma \vdash_{\lambda P+} M : \tau}{\Gamma \vdash_{\lambda P+} NM : \sigma[x := M]} \\
\\
\text{(abs)} \frac{\Gamma_1, x : \tau, \Gamma_2 \vdash_{\lambda P+} M : \sigma \quad \Gamma_1, \Gamma_2 \vdash_{\lambda P+} \text{poprawny}}{\Gamma_1, \Gamma_2 \vdash_{\lambda P+} \lambda x. M : (\forall x : \tau) \sigma} \\
\\
\text{(kd-conv)} \frac{\Gamma \vdash_{\lambda P+} \phi : \kappa \quad \kappa =_{\beta} \kappa'}{\Gamma \vdash_{\lambda P+} \phi : \kappa'} \quad \text{(typ-conv)} \frac{\Gamma \vdash_{\lambda P+} M : \sigma \quad \sigma =_{\beta} \sigma'}{\Gamma \vdash_{\lambda P+} M : \sigma'}
\end{array}$$

Rysunek 3.1: Reguły zmodyfikowanego rachunku λP

Na potrzeby sformułowania i dowodu poniższego twierdzenia system zdefiniowany regułami z rysunku 3.1 będziemy w tym miejscu oznaczać λP^+ . W dalszej części tekstu na oznaczenie systemu z rysunku 3.1 będziemy używać oznaczenia λP .

Twierdzenie 3.2.1 (równoważność rachunków λP)

Dla dowolnego kontekstu Γ , termu M oraz typu τ osąd $\Gamma \vdash M : \tau$ jest wyprowadzalny w λP wtedy i tylko wtedy, gdy jest wyprowadzalny w λP^+ .

Dowód:

Do naszej tezy indukcyjnej dokładamy jeszcze dwie równoważności:

1. Dla dowolnego kontekstu Γ , typu τ osąd $\Gamma \vdash \tau : *$ jest wyprowadzalny w λP wtedy i tylko wtedy, gdy w λP^+ jest wyprowadzalny $\Gamma, x : \tau \vdash$ poprawny.
2. Dla dowolnego kontekstu Γ , termu M oraz rodzaju κ osąd $\Gamma \vdash \kappa : \square$ jest wyprowadzalny w λP wtedy i tylko wtedy, gdy w λP^+ jest wyprowadzalny osąd $\Gamma, \alpha : \kappa \vdash$ poprawny.

(\Rightarrow) Tutaj należy pokazać, jak wyprowadzenia kończące się regułami, które nie występują w λP^+ , przerobić na wyprowadzenia w λP^+ . Dowód będziemy przeprowadzać przez indukcję ze względu na długość wyprowadzenia w λP .

Dla reguły (kd-var) wystarczy, że będziemy potrafili wyprowadzić osąd $\Gamma, \alpha : \kappa \vdash$ poprawny. To zaś udowodnimy, korzystając z założenia indukcyjnego postaci 2.

Dla reguły (var) wystarczy, że będziemy potrafili wyprowadzić $\Gamma, x : \tau \vdash$ poprawny. To zaś udowodnimy, korzystając z założenia indukcyjnego postaci 1.

Dla reguły (trm-kd) możemy za pomocą założenia indukcyjnego przerobić wyprowadzenie dla $\Gamma \vdash \kappa : \square$, tak by kończyło się regułą (type) lub (kd-abs). Jeśli kończącą regułą jest (type), to możemy zmienić założenie dla tej reguły na osąd wynikający z założenia indukcyjnego przyłożonego do $\Gamma \vdash \tau : *$ i natychmiast uzyskać $\Gamma, x : \tau \vdash \kappa : \square$.

Jeśli kończącą regułą jest (kd-abs), to zastępujemy ją regułą noszącą tę samą nazwę w λP^+ . Określamy Γ_1 jako Γ oraz Γ_2 jako puste, co nam z założenia indukcyjnego pozwala natychmiast uzyskać wyprowadzenie dla lewej przesłanki reguły. Wyprowadzenie dla $\Gamma_1, \Gamma_2 \vdash$ poprawny dostajemy z założenia indukcyjnego 1, zauważając jeszcze, że jeżeli da się wyprowadzić $\Gamma_1, \Gamma_2, z : \sigma \vdash$ poprawny dla jakiegoś z i σ , to da się i wyprowadzić $\Gamma_1, \Gamma_2 \vdash$ poprawny.

Dla reguł (typ-kd), (trm-typ), (typ-typ), (trm-trm) i (typ-trm) dowód przebiega podobnie jak dla (trm-kd).

Dla reguły (kd-abs) z założenia indukcyjnego wiemy, że $\Gamma, x : \tau \vdash \kappa : \square$ ma wyprowadzenie w λP^+ . Istnienie takiego wyprowadzenia implikuje istnienie wyprowadzenia dla $\Gamma, x : \tau \vdash$ poprawny i dalej dla $\Gamma \vdash$ poprawny (bo

poprawność dla danego kontekstu oznacza poprawność dla wszystkich jego prefiksów). To zaś pozwala na zastosowanie reguły (kd-abs) w λP^+ i uzyskanie $\Gamma \vdash (\Pi x : \tau) \kappa : \square$.

Dla reguł (typ-abs) i (abs) dowód podobny do (kd-abs).

(\Leftarrow) Tutaj należy pokazać, jak wyprowadzenia kończące się regułami, które nie występują w λP , przerobić na wyprowadzenia w λP . Dowód będziemy przeprowadzać przez indukcję ze względu na długość wyprowadzenia w λP^+ .

Dla (type) w λP^+ ze względu na długość Γ pokazujemy, że można uzyskać końcowy osąd zaczynając od (type) w λP i stosując reguły osłabiania (trm-kd) i (typ-kd).

Dla (kd-abs) końcowy osąd uzyskujemy zastępując (kd-abs) z λP przez (kd-abs) z λP^+ . Musimy jednak zapewnić istnienie w λP wyprowadzenia dla $\Gamma_1, \Gamma_2, x : \tau \vdash \kappa : \square$. W tym celu najpierw pokazujemy przez indukcję ze względu na Γ_2 , że jeśli w λP^+ istnieją wyprowadzenia dla $\Gamma_1, x : \tau, \Gamma_2 \vdash \kappa : \square$ oraz $\Gamma_1, \Gamma_2 \vdash$ poprawny, to istnieje w λP^+ także nie dłuższe wyprowadzenie dla $\Gamma_1, \Gamma_2, x : \tau \vdash \kappa : \square$. Następnie zaś korzystamy z założenia indukcyjnego, co daje nam całe wyprowadzenie.

Dla (typ-abs) i (abs) dowód wygląda podobnie jak dla (kd-abs).

Dla (kd-var) mamy, że wyprowadzenie $\Gamma_1, \alpha : \kappa \vdash$ poprawny jest fragmentem wyprowadzenia $\Gamma_1, \alpha : \kappa, \Gamma_2 \vdash$ poprawny. Zatem możemy skorzystać z założenia indukcyjnego 2 i uzyskać $\Gamma_1 \vdash \kappa : \square$ w λP . To pozwala wyprowadzić $\Gamma, \alpha : \kappa \vdash \alpha : \kappa$, po czym korzystając z reguł (trm-typ) oraz (typ-typ) dołączyć kolejno wszystkie elementy Γ_2 .

Dla (var) dowód jest podobny do (kd-var).

Dla reguły (popr-kd) z założenia indukcyjnego zastosowanego do prawej przesłanki reguły mamy $\Gamma \vdash \kappa : \square$ w λP

Dla reguły (popr-typ) dowód jest podobny do (popr-kd). \square

3.2.1 Zanurzenie logiki pierwszego rzędu w λP

Zdefiniujemy zanurzenie pokazując, jak tłumaczone są poszczególne elementy logiki pierwszego rzędu na odpowiednie elementy rachunku λP . Przedstawione tutaj tłumaczenie pochodzi z [SU98].

Definicja 3.2.2 (kontekst sygnaturowy)

Kontekstem sygnaturowym pierwszego rzędu nazwiemy kontekst w λP , taki że:

1. Występuje w nim tylko jedna zmienna typowa, oznaczana tutaj przez 0 — reprezentuje ona typ wartości, po których wykonywana jest kwantyfikacja.
2. Wszystkie rodzaje mają postać $0 \Rightarrow \dots \Rightarrow 0 \Rightarrow *$;

3. W kontekście występuje skończona liczba wyróżnionych zmiennych konstruktorowych, które reprezentują symbole relacyjne początkowej sygnatury (zmiennej P przypisany jest rodzaj $\underbrace{0 \Rightarrow \dots \Rightarrow 0}_{n\text{-razy}} \Rightarrow *$, jeśli relacja P w sygnaturze miała n argumentów).
4. Symbole funkcyjne z sygnatury są reprezentowane przez wyróżnione zmienne przedmiotowe o typach postaci $0 \rightarrow \dots \rightarrow 0 \rightarrow 0$, przy czym dokładna liczba argumentów zależy od arności symbolu w sygnaturze (stałe są u nas traktowane jak symbole funkcyjne o arności 0).

Kontekst sygnaturowy odpowiadający sygnaturze Σ oznaczamy przez Γ_Σ .

Kolejnym pojęciem składowym logiki pierwszego rzędu jest pojęcie termu algebraicznego. Odpowiednikiem tego pojęcia jest:

Definicja 3.2.3 (homogeniczny term pierwszego rzędu)

Mówimy, że t jest *homogenicznym termem pierwszego rzędu* w kontekście Γ , gdy

- $t = x$, gdzie $\Gamma(x) = 0$ (tzn. x jest symbolem stałej lub zmiennej typu wartości kwantyfikowanych),
- $t = ft_1 \dots t_n$, gdzie $\Gamma(f) = \underbrace{0 \rightarrow \dots \rightarrow 0}_{n\text{-times}} \rightarrow 0$ i każdy z termów t_i jest homogenicznym termem pierwszego rzędu w Γ .

Jak w powyższej definicji termy, które odpowiadają termom algebraicznym, będziemy oznaczać przez: $t, t', t_1, \dots, s, s', s_1 \dots$

Następnym naturalnym krokiem jest zdefiniowanie odpowiednika formuł pierwszego rzędu.

Definicja 3.2.4 (typ pierwszego rzędu)

Mówimy, że typ ϕ jest *typem pierwszego rzędu* w kontekście Γ , jeśli jest zbudowany zgodnie z następującymi zasadami

- $\phi = P(t_1, \dots, t_n)$, gdzie $\Gamma(P) = \underbrace{0 \Rightarrow \dots \Rightarrow 0}_{n\text{-razy}} \Rightarrow *$, $P \neq 0$ i każde t_i jest homogenicznym termem pierwszego rzędu w Γ ,
- $\phi = (\forall x_1 : 0) \dots (\forall x_n : 0). \phi_1 \rightarrow \dots \rightarrow \phi_m$, gdzie każde ϕ_i jest typem pierwszego rzędu w $\Gamma \cup \{x_1 : 0, \dots, x_n : 0\}$.

Zauważmy, że wszystkie termy i typy pierwszego rzędu są wyrażeniami w postaci normalnej.

W naszej definicji logiki pierwszego rzędu (zob. podrozdział 1.3) w systemie naturalnej dedukcji możliwe było umieszczenie w kontekście formuły. W naszej translacji musimy umożliwić tego typu działania.

Definicja 3.2.5 (kontekst pierwszego rzędu)

Kontekstem pierwszego rzędu nad kontekstem sygnaturowym Γ_Σ nazwiemy kontekst λP postaci $\Gamma_\Sigma \cup \{x_1:\phi_1, \dots, x_n:\phi_n\}$, gdzie ϕ_i jest albo typem pierwszego rzędu w kontekście Γ_Σ , albo 0.

W powyższej definicji nadużywamy lekko notacji traktując kontekst w systemie λP jako zbiór tutaj jednak takie uproszczenie jest prawomocne, gdyż łatwo pokazać, że dla każdego uporządkowania asercji z dodawanego zbioru uzyskamy poprawny kontekst w λP .

W końcu określimy, jakie wyprowadzenia λP będziemy uważali za wyprowadzenia pierwszego rzędu. Definicję tę wprowadzimy jednak na dwa sposoby — raz przez podanie warunku ograniczającego na wyprowadzenia w λP i raz podając zestaw reguł. Następnie zaś pokażemy, że te dwie definicje są równoważne. Będą nam potrzebne dwa pojęcia, gdyż pierwsze z nich jest bardziej wygodne przy zajmowaniu się problemami decyzyjnymi, drugie zaś jest wygodniejsze przy pokazywaniu związku naszego systemu z logiką pierwszego rzędu.

Definicja 3.2.6 (wyprowadzenia pierwszego rzędu)

Mówimy, że wyprowadzenie \mathcal{P} w λP jest *wyprowadzeniem pierwszego rzędu*, gdy

każdy osąd $\Gamma \vdash M : \tau$ (gdzie Γ jest kontekstem, τ jest typem, a M termem) w \mathcal{P} ma tę własność, że Γ jest kontekstem pierwszego rzędu nad ustalonym sygnaturowym kontekstem Γ_Σ oraz τ jest typem pierwszego rzędu lub typem postaci $\underbrace{0 \rightarrow \dots \rightarrow 0}_{n\text{-times}} \rightarrow 0$,

gdzie $n \geq 0$. Przy czym jeśli zachodzi ten ostatni przypadek, to M jest podtermem jakiegoś homogenicznego termu pierwszego rzędu.

Drugie pojęcie wyprowadzenia określone jest tak:

Definicja 3.2.7 (wyprowadzenie z regułami pierwszego rzędu)

Mówimy, że wyprowadzenie \mathcal{P} jest *wyprowadzeniem z regułami pierwszego rzędu*, gdy wykonane jest zgodnie z regułami z rysunku 3.2.

Nasz system wyprowadzania będziemy oznaczali $\lambda\forall$.

Pokażemy teraz równoważność tych określeń:

Twierdzenie 3.2.8 (równoważność wyprowadzeń)

Osąd $\Gamma \vdash M : \tau$ dla typu pierwszego rzędu τ i kontekstu pierwszego rzędu Γ ma wyprowadzenie z regułami pierwszego rzędu wtedy i tylko wtedy, gdy ma wyprowadzenie pierwszego rzędu.

$$\begin{array}{c}
\text{(type)} \frac{\Gamma \vdash_{\lambda\forall} \text{poprawny}}{\Gamma \vdash_{\lambda\forall} * : \square} \\
\\
\text{(kd-abs)} \frac{\Gamma_1, x : 0, \Gamma_2 \vdash_{\lambda\forall} \kappa : \square \quad \Gamma_1, \Gamma_2 \vdash_{\lambda\forall} \text{poprawny}}{\Gamma_1, \Gamma_2 \vdash_{\lambda\forall} (\Pi x : 0) \kappa : \square} \\
\\
\text{(kd-var)} \frac{\Gamma_1, \alpha : \kappa, \Gamma_2 \vdash_{\lambda\forall} \text{poprawny}}{\Gamma_1, \alpha : \kappa, \Gamma_2 \vdash_{\lambda\forall} \alpha : \kappa} \\
\\
\text{(kd-app)} \frac{\Gamma \vdash_{\lambda\forall} \phi : (\Pi x : 0) \kappa \quad \Gamma \vdash_{\lambda\forall} M : 0}{\Gamma \vdash_{\lambda\forall} \phi M : \kappa[x := M]} \\
\\
\text{(typ-abs)} \frac{\Gamma_1, x : 0, \Gamma_2 \vdash_{\lambda\forall} \sigma : * \quad \Gamma_1, \Gamma_2 \vdash_{\lambda\forall} \text{poprawny}}{\Gamma_1, \Gamma_2 \vdash_{\lambda\forall} (\forall x : 0) \sigma : *} \\
\\
\text{(typ-arr)} \frac{\Gamma_1, x : \tau, \Gamma_2 \vdash_{\lambda\forall} \sigma : * \quad \Gamma_1, \Gamma_2 \vdash_{\lambda\forall} \text{poprawny}}{\Gamma_1, \Gamma_2 \vdash_{\lambda\forall} \tau \rightarrow \sigma : *} (x \notin FV(\sigma)) \\
\\
\text{(popr-zero)} \quad \vdash_{\lambda\forall} \text{poprawny} \quad \text{(popr-kd)} \frac{\Gamma \vdash_{\lambda\forall} \text{poprawny} \quad \Gamma \vdash_{\lambda\forall} \kappa : \square}{\Gamma, \alpha : \kappa \vdash_{\lambda\forall} \text{poprawny}} \\
\\
\text{(popr-typ)} \frac{\Gamma \vdash_{\lambda\forall} \text{poprawny} \quad \Gamma \vdash_{\lambda\forall} \tau : *}{\Gamma, x : \tau \vdash_{\lambda\forall} \text{poprawny}} \\
\\
\text{(var)} \frac{\Gamma_1, x : \tau, \Gamma_2 \vdash_{\lambda\forall} \text{poprawny}}{\Gamma_1, x : \tau, \Gamma_2 \vdash_{\lambda\forall} x : \tau} \quad \text{(app)} \frac{\Gamma \vdash_{\lambda\forall} N : \tau \rightarrow \sigma \quad \Gamma \vdash_{\lambda\forall} M : \tau}{\Gamma \vdash_{\lambda\forall} NM : \sigma} \\
\\
\text{(inst)} \frac{\Gamma \vdash_{\lambda\forall} N : (\forall x : 0) \sigma \quad \Gamma \vdash_{\lambda\forall} M : 0}{\Gamma \vdash_{\lambda\forall} NM : \sigma[x := M]} \\
\\
\text{(arr-abs)} \frac{\Gamma_1, x : \tau, \Gamma_2 \vdash_{\lambda\forall} M : \sigma \quad \Gamma_1, \Gamma_2 \vdash_{\lambda\forall} \text{poprawny}}{\Gamma_1, \Gamma_2 \vdash_{\lambda\forall} \lambda x. M : \tau \rightarrow \sigma} \\
\\
\text{(abs)} \frac{\Gamma_1, x : 0, \Gamma_2 \vdash_{\lambda\forall} M : \sigma \quad \Gamma_1, \Gamma_2 \vdash_{\lambda\forall} \text{poprawny}}{\Gamma_1, \Gamma_2 \vdash_{\lambda\forall} \lambda x. M : (\forall x : 0) \sigma}
\end{array}$$

Rysunek 3.2: Reguły przypisywania typów dla $\lambda\forall$

Dowód:

(\Rightarrow) Indukcja ze względu na budowę wyprowadzenia przez przypadki ze względu na ostatnio użytą regułę. Załóżmy, że nasz osąd jest wyprowadzalny w $\lambda\forall$ za pomocą wyprowadzenia \mathcal{P} .

Dla osądu $\Gamma \vdash M : \tau$, gdzie τ jest typem pierwszego rzędu stosowalne są w $\lambda\forall$ tylko reguły (var), (app), (inst), (arr-abs), (abs).

Jeśli ostatnią regułą jest (var), to jedynym miejscem w wyprowadzeniu, gdzie występuje osąd postaci $\Gamma \vdash M : \tau$, przy czym τ jest typem, jest konkluzja reguły (var). Dla tej konkluzji warunek z definicji 3.2.6 jest spełniony z założenia o $\Gamma \vdash M : \tau$.

Jeśli ostatnią regułą jest (app), to $M = M_1 M_2$ i wyprowadzenie dla naszego osądu składa się z dwu wyprowadzeń w $\lambda\forall$ dla $\Gamma \vdash M_1 : \tau \rightarrow \sigma$ oraz $\Gamma \vdash M_2 : \tau$, gdzie τ jest pewnym typem pierwszego rzędu. Z założenia indukcyjnego oba te wyprowadzenia są wyprowadzeniami pierwszego rzędu, a zatem wszystkie znajdujące się tam osądy mają własność wskazaną w definicji 3.2.6. Własność tę z założenia ma także osąd końcowy, zatem całe wyprowadzenie tego osądu ma tę własność i wyprowadzenie jest wyprowadzeniem pierwszego rzędu.

Dla reguł (inst), (arr-abs) i (abs) dowód wygląda podobnie, jak w poprzednim przypadku, więc go tutaj pomijamy, zostawiając jego szczegóły bardziej dociekliwym czytelnikom.

(\Leftarrow) Indukcja ze względu na budowę wyprowadzenia pierwszego rzędu przez przypadki ze względu na ostatnią użytą regułę.

Jeśli ostatnią użytą regułą jest (var), to można tę regułę zastąpić regułą (var) z systemu $\lambda\forall$, przy czym wyprowadzenie dla $\Gamma_1, x : \tau, \Gamma_2 \vdash$ poprawny da się wykonać w $\lambda\forall$, gdyż ze względu na fakt, że wyprowadzenie jest pierwszego rzędu, τ jest typem pierwszego rzędu, a dla takiego typu można wyprowadzić poprawność wspomnianego kontekstu w $\lambda\forall$.

Jeśli ostatnią użytą regułą jest (app), to są możliwe dwa przypadki: albo $\tau = 0$, albo $\tau \neq 0$ (oznaczenia jak na rysunku 3.1). W tej pierwszej sytuacji przekształcimy regułę (app) na regułę (inst). Możliwe to jest, ponieważ wiemy, że $\tau = 0$.

W drugiej sytuacji wiadomo, że x nie występuje w σ , gdyż wyprowadzenie jest pierwszego rzędu, co dla osądu $\Gamma \vdash N : (\forall x : \tau)\sigma$ oznacza w związku z tym, że $\tau \neq 0$, że σ nie zawiera x (w typach pierwszego rzędu dozwolona jest kwantyfikacja tylko po zmiennych typu 0). W związku z tym można opuścić podstawienie $[x := M]$ i zastąpić $(\forall x : \tau)\sigma$ przez $\tau \rightarrow \sigma$. Operacje te pozwalają zastąpić regułę (app) przez regułę (app) w $\lambda\forall$.

Jeśli ostatnią użytą regułą jest (abs), to znowu możliwe są dwa przypadki: gdy $\tau = 0$ i gdy $\tau \neq 0$. W tym pierwszym przypadku regułę (abs) w systemie λP można natychmiast przetłumaczyć na regułę (abs) w $\lambda\forall$.

W drugim przypadku, ponieważ σ z założenia o wyprowadzeniu pierwszego rzędu jest typem pierwszego rzędu, to nie może zawierać wolnego wystąpienia zmiennej typu różnego od 0. Oznacza to, że typ $(\forall x : \tau)\sigma$ ma tak

naprawdę postać $\tau \rightarrow \sigma$. Uwaga ta pozwala natychmiast zastąpić w tej sytuacji regułę (abs) w λP przez regułę (arr-abs) w λV .

Jeśli ostatnią regułą jest reguła (kd-konv) lub (typ-conv), to opuszczamy tę regułę, gdyż typy pierwszego rzędu są w postaci normalnej, w związku z czym użyta tutaj β -równość jest po prostu tożsamością. \square

Wykazaliśmy zatem, że dwa obrane przez nas pojęcia wyprowadzenia pierwszego rzędu są równoważne. Warto jeszcze wykonać wysiłek wskazujący, że podane pojęcia mają związek z logiką pierwszego rzędu. Wprowadzimy najpierw pewną pomocniczą definicję

Definicja 3.2.9 (kontekst pochodny)

Powiemy, że dla danej sygnatury kontekst $\Gamma_{\Gamma'}^{\phi}$ w λP jest *kontekstem pochodnym* od kontekstu Γ' i typu ϕ w logice pierwszego rzędu, gdy

$$\Gamma_{\Gamma'}^{\phi} = \Gamma_{\Sigma}, x_1 : 0, \dots, x_n : 0, y_1 : \phi_1, \dots, y_m : \phi_m,$$

gdzie Γ_{Σ} to kontekst sygnaturowy dla sygnatury, nad którą pracujemy, symbole x_1, \dots, x_n to zmienne wolne w Γ i ϕ oraz $\Gamma' = \{\phi_1, \dots, \phi_m\}$. Czasami, gdy nie będzie istotne, o jaki typ chodzi, będziemy opuszczali w oznaczeniu $\Gamma_{\Gamma'}^{\phi}$ górny indeks ϕ , przy czym zmienne wolne będą brane już tylko z kontekstu Γ .

Powiemy, że kontekst Γ jest *prostym rozszerzeniem kontekstu pochodnego* $\Gamma_{\Gamma'}^{\phi}$, gdy ma postać

$$\Gamma_1, z_1 : 0, \dots, z_k : 0, \Gamma_2,$$

gdzie $\Gamma_1, \Gamma_2 = \Gamma_{\Gamma'}^{\phi}$.

Oczywiście kontekst pochodny jest swoim własnym prostym rozszerzeniem.

Twierdzenie 3.2.10 (równoważność logiki pierwszego rzędu i λV)

Osąd $\Gamma \vdash_{\forall} \phi$, gdzie ϕ jest zdaniem nad sygnaturą Σ , jest wyprowadzalny w systemie naturalnej dedukcji dla logiki pierwszego rzędu wtedy i tylko wtedy, gdy istnieje term M oraz proste rozszerzenie Γ' kontekstu pochodnego $\Gamma_{\Gamma'}^{\phi}$, takie że osąd $\Gamma' \vdash M : \phi$ jest wyprowadzalny w λV .

Dowód:

(\Rightarrow) Dowód przez indukcję ze względu na wyprowadzenie $\Gamma \vdash_{\forall} \phi$ przez przypadki ze względu na ostatnio użytą regułę.

Jeśli ostatnią użytą regułą jest (ax), to tworzymy dowód w λV złożony z reguły (var), której założenie stanowi wyprowadzenie dla $\Gamma_1, x : \phi, \Gamma_2 \vdash$ poprawny, gdzie $\Gamma_1, x : \phi, \Gamma_2$ to kontekst pochodny od Γ i ϕ . Takie wyprowadzenie jest możliwe, gdyż w λV wyprowadzalna jest poprawność każdego kontekstu pochodnego.

Jeśli ostatnią użytą regułą jest (\rightarrow I), to $\phi = \phi_1 \rightarrow \phi_2$ i z założenia indukcyjnego możemy utworzyć wyprowadzenie w λV dla $\Gamma_{\Gamma', \phi_1}^{\phi_2} \vdash M_0 : \phi_2$. Możemy też uzyskać wyprowadzenie dla $\Gamma_{\Gamma}^{\phi_1 \rightarrow \phi_2} \vdash$ poprawny, gdyż w systemie

$\lambda\forall$ wyprowadzalna jest poprawność każdego kontekstu pochodnego. Konteksty $\Gamma_{\Gamma, \phi_1}^{\phi_2}$ i $\Gamma_{\Gamma}^{\phi_1 \rightarrow \phi_2}$ różnią się tylko tym, że ten pierwszy ma na końcu $y_m : \phi_2$. Możemy w tym momencie skorzystać z reguły (arr-abs) z $\lambda\forall$ i otrzymać wyprowadzenie dla $\Gamma_{\Gamma} \vdash \lambda x.M_0 : \phi_1 \rightarrow \phi_2$, co daje nam pożądaný rezultat.

Jeśli ostatnią użytą regułą jest (\rightarrow E), to z założenia indukcyjnego możemy utworzyć wyprowadzenia w $\lambda\forall$ dla $\Gamma_1 \vdash M_1 : \phi_1 \rightarrow \phi_2$ oraz dla $\Gamma_2 \vdash M_2 : \phi_1$ dla pewnych termów M_1 i M_2 , gdzie Γ_1 to proste rozszerzenie $\Gamma_{\Gamma}^{\phi_1 \rightarrow \phi_2}$, a Γ_2 , to proste rozszerzenie $\Gamma_{\Gamma}^{\phi_1}$. Te dwa wyprowadzenia można poprawić, tak by dawały w wyniku wspólny zestaw zmiennych pierwszego rzędu w kontekście, a następnie w $\lambda\forall$ połączyć w wyprowadzenie dla $\Gamma_3 \vdash M_1 M_2 : \phi_2$ za pomocą reguły (app), gdzie Γ_3 to proste rozszerzenie $\Gamma_{\Gamma}^{\phi_2}$ o zmienne pierwszego rzędu.

Jeżeli ostatnią użytą regułą jest (\forall I), to z założenia indukcyjnego można wyprowadzić osąd $\Gamma' \vdash M : \phi$, gdzie Γ' jest prostym rozszerzeniem kontekstu pochodnego kontekstu Γ . Jeśli $x \notin FV(\Gamma)$, to można zakładać, że x nie występuje w typach pierwszego rzędu w Γ' . Skoro tak, to można wyprowadzić $\Gamma_1, \Gamma_2 \vdash$ poprawny, gdzie $\Gamma' = \Gamma_1, x : 0, \Gamma_2$, a co za tym idzie można zastosować regułę (abs) i uzyskać osąd $\Gamma_1, \Gamma_2 \vdash \lambda x.M : (\forall x : 0).\phi$. Kontekst Γ_1, Γ_2 jest oczywiście kontekstem rozszerzającym kontekst pochodny $\Gamma_{\Gamma}^{(\forall x : 0).\phi}$.

Jeśli ostatnią użytą regułą jest (\forall E), to z założenia indukcyjnego możemy wyprowadzić osąd $\Gamma' \vdash N : (\forall x : 0)\phi$ dla kontekstu Γ' będącego rozszerzeniem kontekstu pochodnego od Γ i $(\forall x : 0)\phi$ oraz pewnego termu N . Ponieważ M (ozn. z rysunku 1.7) jest termem nad sygnaturą Σ , to można wyprowadzić osąd $\Gamma' \vdash M : 0$. W tym momencie można zastosować regułę (inst) systemu $\lambda\forall$ i uzyskać osąd $\Gamma' \vdash NM : \phi[x := M]$.

(\Leftarrow) Dowód znowu przeprowadzimy przez indukcję ze względu na wyprowadzenie w $\lambda\forall$ przez przypadki wynikające z ostatniej użytej reguły.

Jeżeli ostatnią regułą było (var), to wyprowadzony został osąd $\Gamma, x : \phi \vdash x : \phi$. W tej sytuacji tworzymy dowód w logice pierwszego rzędu składający się z reguły (ax) i osądu $\Gamma', \phi \vdash \phi$, gdzie Γ' jest kontekstem, którym występują dokładnie te formuły, które występują w Γ .

Jeżeli ostatnią regułą było (app), to wyprowadzonym osądem jest $\Gamma \vdash NM : \psi$. Z założenia indukcyjnego możemy wyprowadzić w logice pierwszego rzędu $\Gamma' \vdash \phi \rightarrow \psi$ oraz $\Gamma' \vdash \phi$, gdzie Γ' jest kontekstem, w którym występują dokładnie te formuły, które występują w Γ . Do tych dwóch osądów możemy zastosować regułę (\rightarrow E) i uzyskać oczekiwane $\Gamma \vdash \psi$.

Jeżeli ostatnią regułą było (inst), to wyprowadzonym osądem jest $\Gamma \vdash NM : \psi[x := M]$. Z założenia indukcyjnego możemy wyprowadzić w logice pierwszego rzędu $\Gamma' \vdash \forall x\psi$, gdzie Γ' jest kontekstem, w którym występują dokładnie te formuły, które występują w Γ . Do tak uzyskanego osądu możemy zastosować regułę (\forall E) z termem M . Uzyskujemy w ten sposób oczekiwany osąd $\Gamma' \vdash \psi[x := M]$.

Jeżeli ostatnią regułą było (arr-abs), to wyprowadzonym w $\lambda\forall$ osądem jest $\Gamma \vdash \lambda x.M : \phi \rightarrow \psi$. Z założenia indukcyjnego możemy wyprowadzić w logice pierwszego rzędu $\Gamma', \phi \vdash \psi$, przy czym Γ' zawiera dokładnie te formuły, które występują w Γ . To pozwala nam zastosować regułę (\rightarrow I) i uzyskać $\Gamma' \vdash \phi \rightarrow \psi$.

Jeżeli ostatnią regułą było (abs), to wyprowadzonym w $\lambda\forall$ osądem jest $\Gamma \vdash \lambda x.M : (\forall x:0)\phi$. Z założenia indukcyjnego możemy wyprowadzić w logice pierwszego rzędu $\Gamma' \vdash \psi$, przy czym Γ' zawiera dokładnie te formuły, które występują w Γ . Przy czym x nie występuje w Γ , gdyż Γ jest poprawnie zbudowanym kontekstem po usunięciu $x:0$. To pozwala nam zastosować regułę (\forall I) i uzyskać $\Gamma' \vdash \forall x\psi$. \square

Udowodnimy teraz kilka interesujących faktów, które opisują wyprowadzenia pierwszego rzędu.

Fakt 3.2.11 (wyprowadzenia sterowane składnią)

Jeśli $\Gamma \vdash M : \tau$, gdzie τ jest typem pierwszego rzędu, ma wyprowadzenie pierwszego rzędu \mathcal{P} , to istnieje wyprowadzenie pierwszego rzędu dla tego osądu, które

1. *jeśli $M = x$, to ostatnią użytą regułą jest (var);*
2. *jeśli $M = M_1M_2$ i M_2 jest homogenicznym termem pierwszego rzędu oraz M_1M_2 nie jest podtermem homogenicznego termu pierwszego rzędu, to ostatnią użytą regułą jest (app) i M_1 ma typ postaci $(\forall x:0)\sigma$;*
3. *jeśli $M = M_1M_2$ i M_2 nie jest homogenicznym termem pierwszego rzędu, to ostatnią użytą regułą jest (app) i M_1 ma typ postaci $\sigma_1 \rightarrow \sigma_2$;*
4. *jeśli $M = \lambda x.M_1$, to ostatnią użytą regułą jest (abs).*

Dowód:

Dowód przez indukcję ze względu na liczbę reguł wyprowadzenia w \mathcal{P} , które wyprowadzają typ pierwszego rzędu.

Jeśli rozmiar 1, to mamy możliwą tylko regułę (var). Ta reguła się stosuje tylko, gdy M jest zmienną, stąd nasza teza jest spełniona.

Niech rozmiar dowodu będzie większy od 1. Dla dowodu (1) stwierdzamy, że osąd $\Gamma \vdash x : \tau$ może zostać uzyskany tylko w wyniku działania reguły (var) lub (typ-conv). Ponieważ typy w wyprowadzeniach pierwszego rzędu są zawsze w postaci β -normalnej, regułę (typ-conv) można usunąć i zastosować założenie indukcyjne do pozostałej lewej przesłanki tej reguły.

Dla dowodu (2) stwierdzamy, że osąd $\Gamma \vdash M_1M_2 : \tau$, gdzie M_2 jest termem homogenicznym, a M_1M_2 nie jest podtermem termu homogenicznego, może być uzyskany za pomocą reguł: (app) oraz (typ-conv). Tę ostatnią regułę eliminujemy jak w poprzednim przypadku. Trzeba jeszcze pokazać, że dla

M_1 mamy typ postaci $(\forall x:0)\sigma$. Ponieważ wyprowadzenie jest pierwszego rzędu, a M_1M_2 nie jest termem homogenicznym, to $\sigma[x := M_2]$ jest typem pierwszego rzędu, w związku z czym typ w lewym założeniu reguły musi mieć postać $(\forall x:0)\sigma$ (typ $0 \rightarrow \sigma$ nie jest dozwolonym typem w wyprowadzeniach pierwszego rzędu).

Dla dowodu (3) znowu stwierdzamy, że $\Gamma \vdash M_1M_2 : \tau$ może być uzyskany wyłącznie za pomocą reguł: (app), (typ-conv). Dla reguły (typ-conv) postępujemy jak w poprzednich przypadkach. Dla reguły (app) trzeba jeszcze wykazać, że M_1 ma typ postaci $\sigma_1 \rightarrow \sigma_2$. Ponieważ M_2 nie jest termem homogenicznym pierwszego rzędu, to nie może mieć typu 0 (w wyprowadzeniach pierwszego rzędu taki typ mogą mieć tylko homogeniczne termy pierwszego rzędu). Zatem, żeby typ lewej przesłanki (app) był typem pierwszego rzędu, musi mieć postać $\sigma_1 \rightarrow \sigma_2$.

Dla dowodu (4) stwierdzamy, że $\Gamma \vdash \lambda x.M_1 : \tau$ może zostać uzyskany za pomocą: (abs) lub (typ-conv). Dla reguły (typ-conv) postępujemy jak w poprzednich przypadkach. W związku z tym ostatnią regułą może być tylko (abs). \square

Powyższy fakt pozwala myśleć o wyprowadzeniach pierwszego rzędu jak o wyprowadzeniach składających się z powyżej wymienionych reguł. Co więcej w wyprowadzeniu typu dla termu M ścieżka ρ do jego podtermu N jednoznacznie identyfikuje miejsce w wyprowadzeniu dla M , w którym występuje N , a co za tym idzie typ, jaki został w wyprowadzeniu przypisany temu wystąpieniu N .

Przedstawimy teraz dwa problemy decyzyjne, którymi będziemy się tutaj zajmować.

Problem 3.2.12 (wyprowadzanie typów z kontekstami pierwszego rzędu)

Dane: *Term M rachunku lambda w wersji Curry’ego oraz kontekst pierwszego rzędu Γ .*

Pytanie: *Czy istnieje wyprowadzenie w λP , które kończy się osądem $\Gamma \vdash M : \tau$ dla jakiegoś typu pierwszego rzędu τ ?*

Problem 3.2.13 (kontekstowe wyprowadzanie typów w logice pierwszego rzędu)

Dane: *Term M rachunku lambda w wersji Curry’ego oraz kontekst pierwszego rzędu Γ .*

Pytanie: *Czy istnieje wyprowadzenie pierwszego rzędu, które kończy się osądem $\Gamma \vdash M : \tau$ dla jakiegoś typu pierwszego rzędu τ ?*

Pierwszy z tych dwóch problemów pojawia się naturalnie, gdy zastanawiamy się, czy wystarczy ograniczyć się do samych danych wejściowych pierwszego rzędu, by uzyskać rozstrzygalny problem wyprowadzania typów (wiadomo, że wyprowadzanie typów w λP jest nierozstrzygalne — zob. [Dow93]). Okazuje się jednak, jak zobaczymy w sekcji 3.2.2, że problem ten jest nierozstrzygalny. Następnym naturalnym ograniczeniem systemu jest przyjęcie, że

interesują nas tylko te osady, które można dostać stosując wyprowadzenia pierwszego rzędu. Z tym ograniczeniem właśnie związany jest nasz drugi problem, którym zajmiemy się w sekcji 3.2.3.

3.2.2 Nierozstrzygalność wyprowadzania typów z kontekstami pierwszego rzędu

Przedstawimy tutaj dowód nierozstrzygalności wyprowadzania typów z kontekstami pierwszego rzędu. Konstrukcja tego dowodu opiera się zasadniczo na dowodzie G. Doweka z pracy [Dow93].

Zredukujemy problem unifikacji ze zmiennymi trzeciego rzędu w pozycjach czołowych (w przypadku dla instancji z jednym typem bazowym i czterema symbolami: f_0, f_1, c, d jak w sekcji 2.3; dla przejrzystości prezentacji rezygnujemy z dowodu w ogólnym przypadku, przy czym powinno być jasne na podstawie niniejszego materiału, jak taki dowód przeprowadzić). Ta redukcja wraz z redukcją z sekcji 3.2.3 oznacza silny związek między problemami unifikacji ze zmiennymi w pozycjach czołowych a problemami wyprowadzania typów w systemie λP .

Zacniemy naszą konstrukcję od określenia kontekstu, którego będziemy używali w instancjach problemu wyprowadzania typów. Będziemy zakładać, że E jest instancją problemu unifikacji ze zmiennymi trzeciego rzędu w pozycjach czołowych.

Definicja 3.2.14 (kontekst dla wyprowadzania typów)

Określmy kontekst $\Gamma_{w,0}^E$ jako ciąg:

$$0 : *, f_0 : 0 \rightarrow 0, f_1 : 0 \rightarrow 0, c : 0, d : 0, P : 0 \rightarrow *, x_\phi : (\forall x : 0)((Px) \rightarrow (Pc)).$$

Kontekst Γ_w^E określimy jako $\Gamma_{w,0}^E, x_{\tau_1} : \tau_1, \dots, x_{\tau_k} : \tau_k$, gdzie $\{\tau_1, \dots, \tau_k\}$ jest zbiorem typów argumentów zmiennych wolnych w układzie równań E .

Zauważmy, że kontekst Γ_w jest kontekstem pierwszego rzędu nad sygnaturą $\Sigma = \{f_0, f_1, c, d, P\}$, gdzie f_1, f_2 oznaczają symbole funkcyjne o arności 1, symbole c, d oznaczają symbole stałych (symbole funkcyjne o arności 0), zaś symbol P oznacza symbol predykatu o jednym argumentem.

Obecnie określimy term, który będzie stanowił drugi składnik instancji problemu wyprowadzania typów.

Definicja 3.2.15 (term dla wyprowadzania typów)

Dla każdej zmiennej wolnej F w równaniach E wprowadzamy zmienną przedmiotową x_F . Wprowadzamy zmienne $x_{\text{eq},i}$, gdzie i przebiega numery kolejne równań w zadanym układzie równań. Dodajemy jeszcze jedną zmienną przedmiotową: x_{list} . Definiujemy dla każdej zmiennej F term

$$M_F = x_F x_{\tau_1} \dots x_{\tau_n},$$

przy czym typ F równy jest $\tau_1 \rightarrow \dots \rightarrow \tau_n \rightarrow 0$; dla każdego równania postaci $FM_1 \dots M_n \doteq GN_1 \dots N_m$ o numerze i określamy dwa termy:

$$\begin{aligned} M_{\text{eq},i}^1 &= x_{\text{eq},i}(x_F M_1 \dots M_n), \\ M_{\text{eq},i}^2 &= x_{\text{eq},i}(x_G N_1 \dots N_m). \end{aligned}$$

(Termy M_i, N_j są legalne w Γ_w^E , gdyż składają się wyłącznie z symboli z sygnatury Σ .)

Wreszcie dla każdego równania postaci $FM_1 \dots M_n \doteq N$, mającego numer i , wprowadzamy term

$$M_{\text{eq},i}^1 = x_\phi N(x_F M_1 \dots M_n).$$

Zdefiniujemy jeszcze dla jednolitości przedstawienia $M_{\text{eq},i}^2 = c$. Wreszcie określimy term M_0^E :

$$M_0^E = \lambda x_{\text{list}} x_{F_1} \dots x_{F_n} x_{\text{eq},1} \dots x_{\text{eq},m} \cdot (x_{\text{list}} M_{F_1} \dots M_{F_n} \quad M_{\text{eq},1}^1 \dots M_{\text{eq},m}^1 \quad M_{\text{eq},1}^2 \dots M_{\text{eq},m}^2),$$

gdzie n jest liczbą zmiennych wolnych występujących w E , a m jest liczbą równań w E . Cały term określimy jako $M^E = KIM_0^E$, gdzie K jest termem rzutowania na pierwszą współrzędną $(\lambda xy.x)$, I jest termem identycznościowym $(\lambda x.x)$.

Przedstawimy teraz lemat opisujący główną własność przedstawionej tutaj translacji:

Lemat 3.2.16 (równoważność unifikacji i wyprowadzania typów z kontekstami pierwszego rzędu)

Niech E będzie instancją problemu (nad wspomnianą wcześniej sygnaturą) unifikacji ze zmiennymi trzeciego rzędu w pozycjach czołowych. Układ E jest unifikowalny wtedy i tylko wtedy, gdy dla otoczenia Γ_w^E i termu M^E istnieje typ pierwszego rzędu ψ , taki że osąd $\Gamma_w^E \vdash M^E : \psi$ ma wyprowadzenie w λP .

Dowód:

(\Rightarrow) Załóżmy, że mamy rozwiązanie S układu równań E . Szukany typ ψ będzie równy $(Pc) \rightarrow (Pc)$. Typowanie termu M_0^E uzyskamy, wskazując typy dla zmiennych po λ -abstrakcji:

$$\begin{aligned} x_F : \forall x_1 : \tau_1 \dots x_k : \tau_{k_F} \cdot P(\text{NF}(S(F)x_1 \dots x_{k_F})) \\ x_{\text{eq},i} : P(\text{NF}(S(F)M_1 \dots M_{k_F})) \rightarrow P(c) \end{aligned}$$

dla wszystkich zmiennych wolnych F w E oraz dla wszystkich równań postaci $FM_1 \dots M_{k_F} \doteq GN_1 \dots N_{k_G}$, gdzie k_F, k_G oznaczają arność odpowiednio zmiennej F i G , a $\tau_1, \dots, \tau_{k_F}$ to typy argumentów F .

Dla równania i postaci $FM_1 \dots M_{k_F} \doteq GN_1 \dots N_{k_G}$ na podstawie powyższej definicji $x_F M_1 \dots M_{k_F}$ ma typ $P(\text{NF}(S(F)M_1 \dots M_{k_F}))$ i analogicznie term $x_G N_1 \dots N_{k_G}$ ma typ $P(\text{NF}(S(G)N_1 \dots N_{k_G}))$. Ponieważ S jest unifikatorem typy te są równe, stąd można założyć, że dobrze typują się termy $M_{\text{eq},i}^1, M_{\text{eq},i}^2$ do typu $P(c)$. Rzeczywiście zgodnie z definicją typu zmiennej $x_{\text{eq},i}$ typ pierwszego jej argumentu jest równy $P(\text{NF}(S(F)M_1 \dots M_{k_F}))$, a ten z kolei (ponieważ S unifikuje) jest równy typowi termu $x_G N_1 \dots N_{k_G}$. Oznaczmy typ termu $M_{\text{eq},i}^j$ jako $\tau_{\text{eq},i}^j$.

Dla równania i postaci $FM_1 \dots M_{k_F} \doteq N$ na podstawie powyższej definicji typ dla $x_F M_1 \dots M_{k_F}$ jest równy $P(\text{NF}(S(F)M_1 \dots M_{k_F}))$. Ponieważ typ x_ϕ w otoczeniu Γ^E jest równy $(\forall x:0)((Px) \rightarrow (Pc))$, to term $x_\phi N$ ma typ $(PN) \rightarrow (Px)$. Ponieważ S jest unifikatorem, to aplikacja $x_\phi N$ do $x_F M_1 \dots M_{k_F}$ ma typ Pc . Znowu tutaj oznaczamy typ $M_{\text{eq},i}^1$ przez $\tau_{\text{eq},i}^1$. Typ $M_{\text{eq},i}^2$, to oczywiście 0 i oznaczamy go przez $\tau_{\text{eq},i}^2$.

Pozostaje określenie typu dla termów M_F , gdzie F jest zmienną w równaniu E . Tutaj zgodnie z definicją kontekstu Γ^E mamy, że typem $x_F x_{\tau_1} \dots x_{\tau_n}$ jest $P(\text{NF}(S(F)x_{\tau_1} \dots x_{\tau_n}))$, przy czym τ_i jest typem i -tego argumentu F . Określmy typ M_F jako τ_F .

Zmienna x_{list} w tym momencie ma przypisany typ:

$$\begin{aligned} \tau_{F_1} &\rightarrow \dots \rightarrow \tau_{F_n} \rightarrow \\ \tau_{\text{eq},1}^1 &\rightarrow \dots \rightarrow \tau_{\text{eq},m}^1 \rightarrow \\ \tau_{\text{eq},1}^2 &\rightarrow \dots \rightarrow \tau_{\text{eq},m}^2 \rightarrow Pc. \end{aligned}$$

Oznaczmy ten typ przez τ_* . Lista λ -abstrakcji na początku termu M_0^E pozwala wyprowadzić dla tego termu typ $\tau_* \rightarrow \tau_*$. Oznacza to, że term M_0^E się typuje, a ponieważ dla I można wyprowadzić typ $(Pc) \rightarrow (Pc)$, a dla K można wyprowadzić typ $((Pc) \rightarrow (Pc)) \rightarrow (\tau_* \rightarrow \tau_*) \rightarrow (Pc) \rightarrow (Pc)$, to M^E typuje się do $(Pc) \rightarrow (Pc)$.

(\Leftarrow) Załóżmy teraz, że $\Gamma^E \vdash M^E : \psi$ ma wyprowadzenie w λP dla pewnego ψ . Oznacza to, że term M_0^E ma pewien typ ψ' w otoczeniu Γ^E .

W związku z tym, że jako podtermy M_0^E typują się termy M_F , wiemy, iż typ τ_F dla zmiennej F ma postać $(\forall x:\tau_1 \dots x:\tau_n)\text{NF}(\beta_F x_{\tau_1} \dots x_{\tau_n})$, gdzie $\beta_F = \lambda x_1, \dots, x_n. \beta'_F$ dla pewnego β'_F .

Wiemy też, że typują się termy $M_{\text{eq},i}^1$ i $M_{\text{eq},i}^2$ dla i -tego równania w E , które ma postać $FM_1 \dots M_{k_F} \doteq GN_1 \dots N_{k_G}$, gdzie k_F, k_G to arności F i G . W związku z tym typują się termy $x_F M_1 \dots M_{k_F}$ i $x_G N_1 \dots N_{k_G}$. W związku z naszymi rozważaniami na temat M_F i M_G termy te mają typy $\text{NF}(\beta_F M_1 \dots M_{k_F})$ oraz $\text{NF}(\beta_G N_1 \dots N_{k_G})$. Ponieważ termy te są pierwszymi argumentami $x_{\text{eq},i}$, to mamy równość

$$\text{NF}(\beta_F M_1 \dots M_{k_F}) \simeq \text{NF}(\beta_G N_1 \dots N_{k_G}). \quad (3.2)$$

Z kolei, ponieważ typuje się term $M_{\text{eq},i}^1$ dla i wskazującego na równanie w E postaci $FM_1 \dots M_{k_F} \doteq N$ i ponieważ pierwszym argumentem x_ϕ jest

tutaj N , to mamy typ dla $x_\phi N$ równy $(PN) \rightarrow (Pc)$. Z kolei typ termu $x_F M_1 \dots M_{k_F}$ to, jak już stwierdziliśmy, $\text{NF}(\beta_F M_1 \dots M_{k_F})$. Ponieważ zaś term $x_\phi N$ jest aplikowany do $x_F M_1 \dots M_{k_F}$, to mamy równość

$$P(\text{NF}(\beta_F M_1 \dots M_{k_F})) \simeq P(N),$$

co po opuszczeniu P daje

$$\text{NF}(\beta_F M_1 \dots M_{k_F}) \simeq N. \quad (3.3)$$

Równości (3.2) i (3.3) natychmiast implikują, że podstawienie S określone jako $S(F) = \beta_F$ dla wszystkich zmiennych F występujących w E jest unifikatorem E . \square

Uwaga 3.2.17 Warto zwrócić uwagę, że jedynym miejscem, gdzie powyższa konstrukcja wykracza poza logikę pierwszego rzędu, jest miejsce typowania zmiennej x_F . Jej typ ma postać $(\forall x_1 : \tau_1 \dots x_n : \tau_n) \sigma$, gdzie τ_i są typami postaci $0 \rightarrow \dots \rightarrow 0$. Oznacza to, że powyższy dowód działałby już dla rozszerzenia logiki pierwszego rzędu, w którym dysponujemy możliwością kwantyfikowania po symbolach o typach drugiego rzędu.

Możemy teraz pokazać twierdzenie:

Twierdzenie 3.2.18 (nierozstrzygalność wyprowadzania typów z kontekstami pierwszego rzędu)

Problem wyprowadzania typów z kontekstami pierwszego rzędu jest nierozstrzygalny.

Dowód:

Gdyby problem ten był rozstrzygalny, to istniałby algorytm \mathcal{A} określający, czy dla danego kontekstu i termu da się wyprowadzić zawierający je osąd. Po przetłumaczeniu instancji E problemu unifikacji ze zmiennymi trzeciego rzędu w pozycjach czołowych zgodnie z definicjami 3.2.15 oraz 3.2.14 uzyskalibyśmy kontekst i term, które byłyby danymi wejściowymi dla \mathcal{A} . Zakończenie działania \mathcal{A} sukcesem lub porażką implikowałoby na mocy lematu 3.2.16, że początkowa instancja E odpowiednio ma lub nie ma rozwiązania. Zdefiniowalibyśmy w ten sposób algorytm unifikacji ze zmiennymi trzeciego rzędu w pozycjach czołowych, to zaś jest sprzeczne z twierdzeniem 2.3.6. \square

3.2.3 Rozstrzygalność kontekstowego wyprowadzania typów w logice pierwszego rzędu

W niniejszej sekcji przedstawiamy dowód rozstrzygalności problemu kontekstowego wyprowadzania typów w logice pierwszego rzędu. Rozważamy tutaj przypadek ograniczony do sygnatur, w których znajduje się co najmniej jeden symbol stałej. Ten dodatkowy element pozwala na pewne ułatwienia

w stosowanych tutaj konstrukcjach. Stosujemy go w konstrukcji pozbywania się więzów (fakt 3.2.67) oraz w procesie unifikacji ze zmiennymi w pozycjach czołowych (twierdzenie 2.2.1), gdzie unifikator dla zmiennej spoza sumy warstw uzyskujemy wstawiając ustaloną stałą z sygnatury.

Zauważmy przy okazji, że takie założenie jest bardzo wygodne, gdyż dla kontekstu $\Gamma = (0 : *, P : 0 \rightarrow *)$ (w którym nie ma stałej) i termu $\lambda x.x$ nie można znaleźć wyprowadzenia w $\lambda\forall$. Natomiast dołożenie stałej $c : 0$ powoduje, że można wyprowadzić dla naszego termu typ $P(c) \rightarrow P(c)$.

Ogólna idea algorytmu wygląda tak:

- najpierw generujemy równania między pewnymi specjalnymi obiektami zwanymi e-tyпами,
- następnie upraszczamy te równania,
- ponieważ w równaniach występują kwantyfikatory, pozbywamy się ich,
- wreszcie tłumaczymy uzyskane równania na równania unifikacji drugiego rzędu ze zmiennymi w pozycjach czołowych,
- rozwiązujemy uzyskany układ równań, jeśli da się rozwiązać, to mamy wyprowadzenie, jeśli nie, to wyprowadzenia brak.

E-typy, e-termy i równania na nich

Wprowadzimy teraz wspomniane już e-typy.

Definicja 3.2.19 (e-typy)

Zbiór *e-tyków nad sygnaturą* Σ ze zmiennymi pierwszego rzędu X i zmiennymi typowymi \mathcal{X} , oznaczany $\mathcal{T}_\Sigma^e(X, \mathcal{X})$, zdefiniowany jest jako zbiór napisów wyprowadzalnych przy pomocy symbolu nieterminalnego \mathcal{T} w następującej gramatyce

$$\begin{aligned} T & ::= X \mid C \mid F_1(T) \mid F_2(T_1, T_2) \mid \cdots \mid F_n(T_1, \dots, T_n) \mid T_1 \langle X := T_2 \rangle \\ \mathcal{T} & ::= \mathcal{X} \mid P_1(T) \mid P_2(T_1, T_2) \mid \cdots \mid P_n(T_1, \dots, T_n) \mid \\ & \quad T_1 \rightarrow T_2 \mid (\forall X : 0)T \mid \mathcal{T} \langle X := T \rangle \end{aligned}$$

przy czym X oznacza zmienne pierwszego rzędu, C oznacza symbole stałych z Σ , F_i oznacza symbole funkcji o arności i w Σ , a P_i oznacza symbole relacji o arności i w Σ .

Zbiorem *e-termów nad sygnaturą* Σ i zmiennymi pierwszego rzędu X , oznaczanym $T_\Sigma^e(X)$ nazywamy zbiór napisów wyprowadzalnych w powyższej gramatyce przy pomocy symbolu nieterminalnego T .

E-typy będziemy zwykle oznaczali symbolami typu $\tau, \tau', \tau_1, \dots, \sigma, \dots$, zaś e-termy — symbolami typu t, t_1, \dots, s, \dots

Aby uprościć zapis będziemy pisać $\tau\langle\vec{x} := \vec{t}\rangle$, gdy będziemy chcieli skrócić $\tau\langle x_1 := t_1 \rangle \cdots \langle x_n := t_n \rangle$.

Operację $\langle \cdot := \cdot \rangle$ określamy nazwą *jawnej substytucji* lub w skrócie *substytucji*.

Określimy tutaj zbiór zmiennych wolnych:

Definicja 3.2.20 (zmiennne wolne)

Zbiór *wolnych zmiennych pierwszego rzędu* w e-typie (e-termie) τ , oznaczany $\text{Vars}(\tau)$, jest zdefiniowany indukcyjnie jako

- dla $\tau = c$ mamy $\text{Vars}(\tau) = \emptyset$;
- dla $\tau = x$ mamy $\text{Vars}(\tau) = \{x\}$;
- dla $\tau = f(t_1, \dots, t_n)$ mamy $\text{Vars}(\tau) = \bigcup_{i=1}^n \text{Vars}(t_i)$;
- dla $\tau = t'\langle x := t \rangle$ mamy $\text{Vars}(\tau) = \text{Vars}(t') \setminus \{x\} \cup \text{Vars}(t)$;
- dla $\tau = \alpha$ mamy $\text{Vars}(\tau) = \emptyset$;
- dla $\tau = P(t_1, \dots, t_n)$ mamy $\text{Vars}(\tau) = \bigcup_{i=1, \dots, n} \text{Vars}(t_i)$;
- dla $\tau = \tau_1 \rightarrow \tau_2$ mamy $\text{Vars}(\tau) = \text{Vars}(\tau_1) \cup \text{Vars}(\tau_2)$;
- dla $\tau = (\forall x : 0)\tau'$ mamy $\text{Vars}(\tau) = \text{Vars}(\tau') \setminus \{x\}$;
- dla $\tau = \tau'\langle x := t \rangle$ mamy $\text{Vars}(\tau) = \text{Vars}(\tau') \setminus \{x\} \cup \text{Vars}(t)$.

Oczywiście zmienne, które nie są wolne (są związane operatorem $\langle \cdot := \cdot \rangle$) można przemianowywać. Relację równoważności utożsamiającą termy różniące się tylko przemianowanymi zmiennymi związanymi oznaczamy jak zwykle przez \equiv_α .

Dalej wprowadzamy notację $\text{TV}(\tau)$ dla zbioru zmiennych typowych w typie τ . Notacje $\text{Vars}(\cdots)$ oraz $\text{TV}(\cdots)$ rozszerzamy, tak by można je było stosować do zbiorów e-typów.

Pojęcie podstawienia analogiczne do podstawienia z definicji 1.1.7 nie jest tutaj oczywiste, stąd przedstawiamy jego pełną definicję indukcyjną. W dalszej części tekstu wprowadzamy też dla omawianych tutaj typów jeszcze jedno, odmienne od przedstawionego tutaj, pojęcie podstawienia (zob. definicja 3.2.38). Dla odróżnienia tych dwóch rodzajów podstawień, pojęcie definiowane tutaj nazywamy podstawieniem pierwszego rzędu.

Definicja 3.2.21 (podstawienie pierwszego rzędu)

Podstawienie pierwszego rzędu to funkcja częściowa o skończonej dziedzinie zawartej w zbiorze zmiennych pierwszego rzędu i dająca wyniki w zbiorze e-termów. Takie podstawienie zapisujemy zwykle jako $[x_1 := t_1, \dots, x_n := t_n]$. Na e-typach funkcja ta działa w sposób następujący

- $x_i[x_1 := t_1, \dots, x_i := t_i, \dots, x_n := t_n] = t_i$;

- $y[x_1 := t_1, \dots, x_n := t_n] = y$, jeśli dla każdego $i = 1, \dots, n$ mamy $x_i \neq y$;

- $f(s_1, \dots, s_n)[x_1 := t_1, \dots, x_n := t_n] = f(s'_1, \dots, s'_n)$, gdzie

$$s'_i = s_i[x_1 := t_1, \dots, x_n := t_n];$$

- $t\langle x := u \rangle[x_1 := t_1, \dots, x_n := t_n] = t'\langle x' := u' \rangle$, gdzie

$$u' = u[x_1 := t_1, \dots, x_n := t_n],$$

zmienna x' nie występuje w żadnym z termów $x_1, \dots, x_n, t_1, \dots, t_n$ oraz $t' = t[x := x'] [x_1 := t_1, \dots, x_n := t_n]$;

- $\alpha[x_1 := t_1, \dots, x_n := t_n] = \alpha$;

- $P(s_1, \dots, s_n)[x_1 := t_1, \dots, x_n := t_n] = P(s'_1, \dots, s'_n)$, gdzie

$$s'_i = s_i[x_1 := t_1, \dots, x_n := t_n];$$

- $\tau_1 \rightarrow \tau_2[x_1 := t_1, \dots, x_n := t_n] = \tau'_1 \rightarrow \tau'_2$, gdzie $\tau'_i = \tau_i[x_1 := t_1, \dots, x_n := t_n]$;

- $((\forall x : 0)\tau)[x_1 := t_1, \dots, x_n := t_n] = ((\forall x' : 0)\tau')$, gdzie x' nie występuje w $x_1, \dots, x_n, t_1, \dots, t_n$ oraz $\tau' = \tau[x := x'] [x_1 := t_1, \dots, x_n := t_n]$;

- $\tau\langle x := u \rangle[x_1 := t_1, \dots, x_n := t_n] = \tau'\langle x' := u' \rangle$, gdzie

$$u' = u[x_1 := t_1, \dots, x_n := t_n],$$

zmienna x' nie występuje w żadnym z termów $x_1, \dots, x_n, t_1, \dots, t_n$ oraz $\tau' = \tau[x := x'] [x_1 := t_1, \dots, x_n := t_n]$.

Definicja 3.2.22 (wskazywanie podtermu)

Określamy pojęcie *wskazania podtermu* i *ścieżki* podobnie jak w definicji 1.3.3 traktując symbole z sygnatury dla e-typów jak symbole pierwszego rzędu, symbol \rightarrow jako dodatkowy symbol dwuargumentowy, każde $(\forall x : 0)$ jak dodatkowy symbol jednoargumentowy oraz operację $\cdot\langle \cdot := \cdot \rangle$ jak dodatkowy symbol trójargumentowy.

Określimy teraz pewną redukcję, która pozwoli nam określić swego rodzaju semantykę dla e-termów i e-typów.

Definicja 3.2.23 (redukcja dla e-termów oraz e-typów)

Redukcję dla e-termów i e-typów, oznaczaną $t \rightsquigarrow t'$ lub $\tau \rightsquigarrow \tau'$, określamy rekurencyjnie jak następuje:

1. $f(s_1, \dots, s_n) \rightsquigarrow f(s'_1, \dots, s'_n)$, gdy f jest symbolem operacji pierwszego rzędu w Σ oraz dla któregoś $i \in \{1, \dots, n\}$ mamy $s_i \rightsquigarrow s'_i$, zaś dla $j \neq i$ mamy $s_j = s'_j$;
2. $t\langle x := s \rangle \rightsquigarrow t'\langle x := s \rangle$, gdy $t \rightsquigarrow t'$;
3. $t\langle x := s \rangle \rightsquigarrow t[x := s]$, gdy t jest nieredukowalne;
4. $P(s_1, \dots, s_n) \rightsquigarrow P(s'_1, \dots, s'_n)$, gdy P jest symbolem predykatu w sygnaturze Σ oraz dla któregoś $i \in \{1, \dots, n\}$ mamy $s_i \rightsquigarrow s'_i$, zaś dla $j \neq i$ mamy $s_j = s'_j$;
5. $\sigma_1 \rightarrow \sigma_2 \rightsquigarrow \sigma'_1 \rightarrow \sigma_2$, gdy $\sigma_1 \rightsquigarrow \sigma'_1$;
6. $\sigma_1 \rightarrow \sigma_2 \rightsquigarrow \sigma_1 \rightarrow \sigma'_2$, gdy $\sigma_2 \rightsquigarrow \sigma'_2$;
7. $(\forall y:0)\sigma_1 \rightsquigarrow (\forall y:0)\sigma'_1$, gdy $\sigma_1 \rightsquigarrow \sigma'_1$;
8. $\sigma\langle x := s \rangle \rightsquigarrow \sigma'\langle x := s \rangle$, gdy $s \in T_\Sigma^e(X)$, $x \in X$ oraz $\sigma \rightsquigarrow \sigma'$;
9. $(\sigma_1 \rightarrow \sigma_2)\langle x := s \rangle \rightsquigarrow \sigma_1\langle x := s \rangle \rightarrow \sigma_2\langle x := s \rangle$, gdy $s \in T_\Sigma^e(X)$, $x \in X$;
10. $((\forall y:0)\sigma)\langle x := s \rangle \rightsquigarrow ((\forall y:0)\sigma\langle x := s \rangle)$, gdy $s \in T_\Sigma^e(X)$, $x \neq y$ (gdy $x = y$ wykonujemy najpierw α -konwersję, a następnie redukujemy zgodnie z niniejszą regułą);
11. $P(t_1, \dots, t_m)\langle x := s \rangle \rightsquigarrow P(t'_1, \dots, t'_m)$, gdy $s \in T_\Sigma^e(X)$, $x \in X$, P jest predykatem z Σ o arności m , zaś $t'_i = t_i\langle x := s \rangle$ dla $i = 1, \dots, m$.

Jak zwykle rozszerzamy \rightsquigarrow do jej zwrotno-przechodniego domknięcia \rightsquigarrow^* .

Zwracamy tutaj uwagę na fakt, że zgodnie z przedstawioną tutaj definicją e-typ postaci $\alpha\langle x := t \rangle$, gdzie α jest zmienną typową, jest nieredukowalny.

Zajmiemy się teraz pokazywaniem, że \rightsquigarrow^* ma własność Churcha-Rossera oraz własność silnej normalizacji. Zaczniemy od własności Churcha-Rossera. Pokażemy najpierw trochę inną własność — własność słabej konfluencji. Własność ta łącznie z własnością silnej normalizacji daje na mocy lematu Newmana własność Churcha-Rossera.

Definicja 3.2.24 (słaba konfluencja)

Mówimy, że system redukcyjny \rightarrow ma własność słabej konfluencji, gdy dla dowolnych trzech elementów M_1, M_2, M_3 w dziedzinie relacji, takich że $M_1 \rightarrow M_2$ oraz $M_1 \rightarrow M_3$, istnieje element M_4 , taki że $M_2 \rightarrow^* M_4$ oraz $M_3 \rightarrow^* M_4$, gdzie \rightarrow^* jest domknięciem zwrotno-przechodnim relacji \rightarrow .

Fakt 3.2.25 (słaba konfluencja)

Relacja redukcji \rightsquigarrow ma własność słabej konfluencji.

Dowód:

Przypuśćmy, że $\tau_1 \rightsquigarrow \tau_2$ oraz $\tau_1 \rightsquigarrow \tau_3$. Udowodnimy przez indukcję ze względu na budowę e-typu (e-termu) τ_1 , że istnieje e-typ (e-term) τ_4 , taki że $\tau_2 \rightsquigarrow^* \tau_4$ oraz $\tau_3 \rightsquigarrow^* \tau_4$. (W całym dowodzie pomijamy sytuację, gdy $\tau_2 = \tau_3$. Dla niej dowód jest natychmiastowy.)

Najpierw przedstawimy przypadki, gdy τ_1, τ_2, τ_3 są e-termami:

- Gdy τ_1 jest zmienną, to nie jest możliwy żaden krok redukcji, więc nasza teza jest spełniona.
- Gdy $\tau_1 = f(\sigma_1, \dots, \sigma_n)$, to mamy dwie możliwości:

- Jeśli $\tau_2 = f(\sigma_1, \dots, \sigma_i^2, \dots, \sigma_n)$ oraz $\tau_3 = f(\sigma_1, \dots, \sigma_i^3, \dots, \sigma_n)$, gdzie $\sigma_i \rightsquigarrow \sigma_i^2$ and $\sigma_i \rightsquigarrow \sigma_i^3$, to możemy zastosować założenie indukcyjne do σ_i, σ_i^2 i σ_i^3 , uzyskując σ_i^4 , takie że $\sigma_i^2 \rightsquigarrow^* \sigma_i^4$ and $\sigma_i^3 \rightsquigarrow^* \sigma_i^4$. Definicja \rightsquigarrow implikuje $f(\sigma_1, \dots, \sigma_i^2, \dots, \sigma_n) \rightsquigarrow^* f(\sigma_1, \dots, \sigma_i^4, \dots, \sigma_n)$ oraz

$$\tau_3 = f(\sigma_1, \dots, \sigma_i^3, \dots, \sigma_n) \rightsquigarrow^* f(\sigma_1, \dots, \sigma_i^4, \dots, \sigma_n).$$

To oznacza, że $\tau_4 = f(\sigma_1, \dots, \sigma_i^4, \dots, \sigma_n)$ spełnia warunki naszej tezy.

- Jeśli $\tau_2 = f(\sigma_1, \dots, \sigma_i^2, \dots, \sigma_n)$ i $\tau_3 = f(\sigma_1, \dots, \sigma_j^3, \dots, \sigma_n)$, gdzie $\sigma_i \rightsquigarrow \sigma_i^2$, $\sigma_j \rightsquigarrow \sigma_j^3$ oraz $i \neq j$, to możemy określić

$$\tau_4 = f(\sigma_1, \dots, \sigma_i^2, \dots, \sigma_j^3, \dots, \sigma_n)$$

(bez utraty ogólności możemy założyć, że $i < j$). Uzyskujemy teraz $\tau_2 \rightsquigarrow^* \tau_4$ wykonując wszystkie redukcje między σ_j a σ_j^2 oraz podobnie uzyskujemy $\tau_3 \rightsquigarrow^* \tau_4$ wykonując wszystkie redukcje między σ_i a σ_i^3 .

- Gdy $\tau_1 = \tau \langle x := s \rangle$ i τ ma podterm postaci $u \langle y := v \rangle$, to $\tau_2 = \tau'_2 \langle x := s \rangle$ i $\tau_3 = \tau'_3 \langle x := s \rangle$, przy czym $\tau \rightsquigarrow \tau'_2$ oraz $\tau \rightsquigarrow \tau'_3$. Z założenia indukcyjnego istnieje pewien term τ'_4 , taki że $\tau'_2 \rightsquigarrow^* \tau'_4$ oraz $\tau'_3 \rightsquigarrow^* \tau'_4$. Definicja \rightsquigarrow implikuje, że $\tau'_i \langle x := s \rangle \rightsquigarrow^* \tau'_4 \langle x := s \rangle$ dla $i = 2, 3$. To z kolei daje $\tau_2 \rightsquigarrow^* \tau_4$ oraz $\tau_3 \rightsquigarrow^* \tau_4$ dla $\tau_4 = \tau'_4 \langle x := s \rangle$.
- Gdy $\tau_1 = \tau \langle x := s \rangle$ i τ nie ma podtermu postaci $u \langle y := v \rangle$, to możliwy jest tylko jeden krok redukcji. Stąd mamy $\tau_2 = \tau_3$ i możemy określić $\tau_4 = \tau_2$, które oczywiście spełnia naszą tezę.

Obecnie przedstawimy przypadki, gdy τ_1, τ_2, τ_3 są właściwymi e-typami:

- Gdy $\tau_1 = P(s_1, \dots, s_n)$, to mamy dwie możliwości — albo obie redukcje (do τ_2 i do τ_3) dotyczą tego samego termu s_i albo termów

różnych s_{i_1} oraz s_{i_2} . W pierwszym przypadku korzystamy z założenia indukcyjnego dla e-termów i definicji redukcji dla przypadku typu predykatowego. W drugim przypadku możemy uzyskać τ_4 jako $P(s_1, \dots, s'_{i_1}, \dots, s'_{i_2}, \dots, s_n)$, gdzie $s_{i_j} \rightsquigarrow s'_{i_j}$ dla $j = 1, 2$ i zdefiniować redukcję do τ_4 jako redukcję w odpowiednim, dotąd nieredukowanym terminie spośród s_{i_1}, s_{i_2} .

- Gdy $\tau_1 = \sigma_1 \rightarrow \sigma_2$, dowód jest podobny do dowodu w poprzednim przypadku.
- Gdy $\tau_1 = (\forall x:0)\sigma_1$, redukcje do τ_2 i τ_3 muszą być wykonywane wewnątrz σ_1 , zatem naszą tezę uzyskujemy natychmiast, stosując założenie indukcyjne.
- Gdy $\tau_1 = \sigma_1 \langle x := s \rangle$, mamy dwa przypadki w zależności od tego, czy redukcje $\tau_1 \rightsquigarrow \tau_2$ oraz $\tau_1 \rightsquigarrow \tau_2$ wykonywane są z udziałem najbardziej zewnętrznego podstawienia w τ_1 .

W przypadku, gdy takie podstawienie nie bierze udział w redukcji, mamy $\tau_2 = \sigma_2 \langle x := s \rangle$ oraz $\tau_3 = \sigma_3 \langle x := s \rangle$, gdzie $\sigma_1 \rightsquigarrow \sigma_2$ oraz $\sigma_1 \rightsquigarrow \sigma_3$. W związku z tym nasza teza jest spełniona na mocy założenia indukcyjnego oraz definicji redukcji.

W przypadku, gdy najbardziej zewnętrzne podstawienie bierze udział w jednej z redukcji, to możemy założyć bez utraty ogólności, że bierze ono udział w redukcji $\tau_1 \rightsquigarrow \tau_2$. Mamy tutaj trzy przypadki w zależności od postaci τ_1 .

- Gdy $\tau_1 = (\sigma_1 \rightarrow \sigma_2) \langle x := s \rangle$, to $\tau_2 = \sigma_1 \langle x := s \rangle \rightarrow \sigma_2 \langle x := s \rangle$, gdzie $s \in T_\Sigma^e(X)$, $x \in X$. Co więcej, $\tau_3 = (\sigma'_1 \rightarrow \sigma_2) \langle x := s \rangle$, gdzie $\sigma_1 \rightsquigarrow \sigma'_1$ lub $\tau_3 = (\sigma_1 \rightarrow \sigma'_2) \langle x := s \rangle$, gdzie $\sigma_2 \rightsquigarrow \sigma'_2$. Możemy oczywiście wykonać redukcję $\tau_2 \rightsquigarrow \tau_4$ zgodnie z regułami (5, 8) lub regułami (6, 8) oraz $\tau_3 \rightsquigarrow \tau_4$ zgodnie z regułą (9) definicji 3.2.23, przy czym

$$\begin{aligned} \tau_4 &= (\sigma'_1 \langle x := s \rangle \rightarrow \sigma_2 \langle x := s \rangle) \\ &\text{lub} \\ \tau_4 &= (\sigma_1 \langle x := s \rangle \rightarrow \sigma'_2 \langle x := s \rangle). \end{aligned}$$

- Gdy $\tau_1 = P(t_1, \dots, t_m) \langle x := s \rangle$, to bez utraty ogólności możemy założyć, że $\tau_2 = P(t'_1, \dots, t'_m)$, gdzie $s \in T_\Sigma^e(X)$, $x \in X$, $P \in \Sigma$ i ma arność m , zaś $t'_i = t_i \langle x := s \rangle$ dla $i = 1, \dots, m$. W tej sytuacji $\tau_3 = P(t''_1, \dots, t''_m) \langle x := s \rangle$, gdzie dla pewnego $i \in \{1, \dots, m\}$ mamy $t_i \rightsquigarrow t''_i$ oraz $t_j = t''_j$ dla $j \neq i$. Określimy teraz $\tau_4 = P(t'_1, \dots, t''_i \langle x := s \rangle, \dots, t'_m)$. Mamy $\tau_2 \rightsquigarrow \tau_4$ w wyniku działania reguł (4, 8) definicji 3.2.23 oraz $\tau_3 \rightsquigarrow \tau_4$ w wyniku działania reguły (11) definicji 3.2.23.

- Gdy $\tau_1 = ((\forall y:0)\sigma_1)\langle x := s \rangle$, to bez utraty ogólności możemy założyć, że $\tau_2 = ((\forall y:0)\sigma_1)\langle x := s \rangle$ oraz $\tau_3 = ((\forall y:0)\sigma'_1)\langle x := s \rangle$, gdzie $\sigma_1 \rightsquigarrow \sigma'_1$. Określamy $\tau_4 = ((\forall y:0)\sigma'_1)\langle x := s \rangle$. Redukcja $\tau_2 \rightsquigarrow \tau_4$ zachodzi na mocy reguł (7, 8) definicji 3.2.23, zaś $\tau_3 \rightsquigarrow \tau_4$ na mocy reguły (10) definicji 3.2.23.

□

Będziemy teraz pokazywać własność silnej normalizacji dla redukcji \rightsquigarrow . Własność ta oznacza, że każdy ciąg redukcji wykonywanych za pomocą tej relacji jest skończony. Wprowadzimy kilka definicji pomocniczych oraz udowodnimy trochę technicznych własności.

Przy pokazywaniu silnej normalizacji będziemy korzystać z dwóch porządków częściowych. Przedstawimy tutaj ich definicje dla ustalenia odpowiedniej notacji.

Definicja 3.2.26 (porządki dla dowodu silnej normalizacji)

Częściowy porządek \sqsubseteq na zbiorze skończonych multizbiorów liczb naturalnych ($\text{Multi}(\mathbb{N})$) jest określony tak:

$$A_1 \sqsubseteq A_2 \Leftrightarrow A_1 = A_2 \text{ lub istnieje } n \in \mathbb{N}, \text{ takie że dla każdego } m > n \text{ mamy } A_1(m) = A_2(m) \text{ oraz } A_1(n) < A_2(n).$$

(Uwaga: powyżej stosujemy notację wykorzystującą fakt, że multizbiory liczb naturalnych można traktować jako funkcje $\mathbb{N} \rightarrow \mathbb{N}$.) Drugi porządek określony jest na zbiorze $\mathbb{N} \times \text{Multi}(\mathbb{N})$:

$$\langle k, A \rangle \sqsubseteq_\mu \langle k', A' \rangle \Leftrightarrow k < k' \text{ lub } k = k' \text{ oraz } A \sqsubseteq A'.$$

Oba powyższe porządki są dobrze ufundowane.

Definicja 3.2.27 (zagnieżdżenie e-typu)

Określimy funkcję nest obliczającą stopień *zagnieżdżenia* e-typu

- $\text{nest}(x) = 0$;
- $\text{nest}(f(t_1, \dots, t_n)) = \max_{i=1, \dots, n} \text{nest}(t_i)$;
- $\text{nest}(t\langle x := s \rangle) = \max(\text{nest}(t), \text{nest}(s) + 1)$;
- $\text{nest}(P(t_1, \dots, t_n)) = \max_{i=1, \dots, n} \text{nest}(t_i)$;
- $\text{nest}(\alpha) = 0$;
- $\text{nest}(\sigma_1 \rightarrow \sigma_2) = \max(\text{nest}(\sigma_1), \text{nest}(\sigma_2))$;
- $\text{nest}((\forall x:0)\sigma_2) = \text{nest}(\sigma_2)$;
- $\text{nest}(\sigma\langle x := s \rangle) = \max(\text{nest}(\sigma), \text{nest}(s) + 1)$.

Powyższa funkcja ma interesującą własność

Fakt 3.2.28 (nest i redukcja)

Dla dowolnych e-typów σ_1, σ_2 , jeśli $\sigma_1 \rightsquigarrow \sigma_2$, to $\text{nest}(\sigma_1) \geq \text{nest}(\sigma_2)$.

Dowód:

Indukcja ze względu na budowę typu σ_1 . Nietrywialny przypadek ma miejsce w sytuacji, gdy $\sigma_1 = t\langle x := s \rangle$, gdzie t jest e-termem, i to gdy wykonywany jest najbardziej zewnętrzny redeks. W tej sytuacji t jest nieredukowalne i $\sigma_2 = t[x := s]$, a to oznacza, że s zostanie podstawione na wszystkie wystąpienia zmiennej x , jednak żadne z tych wystąpień nie znajduje się w podtermie postaci $\langle x := s' \rangle$, co oznacza, że wszystkie ewentualne substytucje, które występowały w s będą przy liczeniu maksimum będącego wartością $\text{nest}(t[x := s])$ brane pod uwagę jako liczba $\text{nest}(s)$, a nie jak do tej pory jako $\text{nest}(s) + 1$. \square

Definicja 3.2.29 (miara dla silnej normalizacji)

Określimy teraz $\mu : \mathcal{T}_{\Sigma}^e(X, \mathcal{X}) \rightarrow \mathbb{N} \times \text{Multi}(\mathbb{N})$. Przez $\mu_i(\sigma)$ będziemy oznaczali i -tą współrzędną $\mu(\sigma)$ dla $i = 1, 2$. Niech T_σ będzie zbiorem tych ścieżek w σ , które prowadzą do typu atomowego (tzn. podtypu jednej z postaci: $P(\dots)$ lub α).

Samą funkcję μ określimy rekurencyjnie po budowie σ .

- $\mu(x) = \langle 0, \{\text{nest}(x)\} \rangle$;
- $\mu(f(t_1, \dots, t_n)) = \langle 0, \{\text{nest}(f(t_1, \dots, t_n))\} \cup \bigcup_{i=1}^n \mu_2(t_i) \rangle$;
- $\mu(t\langle x := s \rangle) = \langle 0, \{\text{nest}(t\langle x := s \rangle)\} \cup \mu_2(t) \cup \mu_2(s) \rangle$, gdzie t jest e-termem;
- $\mu(\alpha) = \langle 0, \{\text{nest}(\alpha)\} \rangle$;
- $\mu(P(t_1, \dots, t_n)) = \langle 0, \{\text{nest}(P(t_1, \dots, t_n))\} \cup \bigcup_{i=1}^n \mu_2(t_i) \rangle$;
- $\mu(\sigma_1 \rightarrow \sigma_2) = \langle \mu_1(\sigma_1) + \mu_1(\sigma_2), \{\text{nest}(\sigma_1 \rightarrow \sigma_2)\} \cup \mu_2(\sigma_1) \cup \mu_2(\sigma_2) \rangle$;
- $\mu((\forall x : 0)\sigma) = \langle \mu_1(\sigma), \{\text{nest}((\forall x : 0)\sigma)\} \cup \mu_2(\sigma) \rangle$;
- $\mu(\tau\langle x := s \rangle) = \langle \mu_1(\tau) + k, \{\text{nest}(\tau\langle x := s \rangle)\} \cup \mu_2(\tau) \cup \mu_2(s) \rangle$, gdzie τ jest e-typem, a k jest sumą długości ścieżek w $T_{\tau\langle x := s \rangle}$.

Fakt 3.2.30 (silna normalizacja)

Dla zdefiniowanej w definicji 3.2.29 funkcji $\mu : \mathcal{T}_{\Sigma}^e(X, \mathcal{X}) \rightarrow \mathbb{N} \times \text{Multi}(\mathbb{N})$ jeśli $\sigma_1 \rightsquigarrow \sigma_2$, to $\mu(\sigma_1) \sqsupseteq_\mu \mu(\sigma_2)$.

Dowód:

Dowód przez indukcję ze względu na budowę σ .

- Gdy $\sigma = x$ dla pewnej zmiennej pierwszego rzędu x , to nie może zostać wykonany żaden krok redukcji, zatem nasza teza jest prawdziwa.
- Gdy $\sigma = f(s_1, \dots, s_n)$, gdzie f jest symbolem funkcyjnym z Σ , to $\sigma \rightsquigarrow f(s'_1, \dots, s'_n)$, gdzie dla pewnego $i \in \{1, \dots, n\}$ mamy $s_i \rightsquigarrow s'_i$ oraz dla $j \neq i$ mamy $s_j = s'_j$. Ponieważ σ jest e-termem, to $\mu_1(\sigma) = 0$. W związku z tym wystarczy pokazać, że druga współrzędna μ ściśle maleje przy tej redukcji. Mamy $\mu_2(\sigma) = \{\text{nest}(\sigma)\} \cup \bigcup_{i=1}^n \mu_2(s_i)$ oraz $\mu_2(f(s'_1, \dots, s'_n)) = \{\text{nest}(f(s'_1, \dots, s'_n))\} \cup \bigcup_{i=1}^n \mu_2(s'_i)$. Założenie indukcyjne daje nam $\mu_2(s_i) \sqsupseteq \mu_2(s'_i)$. Dla $j \neq i$ mamy $s_j = s'_j$, zatem wystarczy stwierdzić, że $\text{nest}(\sigma) \geq \text{nest}(f(s'_1, \dots, s'_n))$, by uzyskać $\mu_2(\sigma) \sqsupseteq \mu_2(f(s'_1, \dots, s'_n))$. To zaś zachodzi na mocy faktu 3.2.28.
- Gdy $\sigma = \sigma_1 \langle x := s \rangle$, przy czym σ_1 jest e-termem i ma podterm postaci $u \langle y := s' \rangle$, to σ redukuje się do $\sigma'_1 \langle x := s \rangle$, gdzie mamy dodatkowo $\sigma_1 \rightsquigarrow \sigma'_1$. Ponieważ σ jest e-termem, to $\mu_1(\sigma) = 0$. W związku z tym wystarczy pokazać, że druga współrzędna μ ściśle maleje przy tej redukcji. W naszej sytuacji mamy $\mu_2(\sigma) = \mu_2(\sigma_1) \cup \mu_2(s) \cup \{\max(\text{nest}(\sigma_1), 1 + \text{nest}(s))\}$ oraz $\mu_2(\sigma'_1 \langle x := s \rangle) = \mu_2(\sigma'_1) \cup \mu_2(s) \cup \{\max(\text{nest}(\sigma'_1), 1 + \text{nest}(s))\}$. Z założenia indukcyjnego mamy $\mu_2(\sigma_1) \sqsupseteq \mu_2(\sigma'_1)$. Co więcej, $\max(\text{nest}(\sigma_1), 1 + \text{nest}(s)) \geq \max(\text{nest}(\sigma'_1), 1 + \text{nest}(s))$, bo $\text{nest}(\sigma_1) \geq \text{nest}(\sigma'_1)$ na mocy faktu 3.2.28. To razem daje $\mu_2(\sigma) \sqsupseteq \mu_2(\sigma'_1 \langle x := s \rangle)$.
- Gdy $\sigma = \sigma_1 \langle x := s \rangle$, przy czym σ_1 jest e-termem i nie ma podtermu postaci $u \langle y := s' \rangle$, to σ redukuje się do $\sigma[x := s]$. Znowu wystarczy tutaj rozważanie μ_2 . Ponieważ σ_1 nie ma podtermu postaci $u \langle y := s' \rangle$, zachodzi sytuacja, że $\text{nest}(\sigma)$ jest ostro większe od dowolnego zagnieżdżenia podtermu σ . Co więcej, zgodnie z definicją w tym przypadku, $\mu_2(\sigma_1[x := s])(n)$ ma wartość niezerową tylko dla n równego $\text{nest}(s')$ dla jakiegoś s' będącego podtermem s (a więc będącego podtermem σ). To implikuje, że wartość zagnieżdżenia σ zmniejsza się, pomimo że być może $\mu_2(\sigma)(n) < \mu_2(\sigma_1[x := s])(n)$ dla pewnego n (jest to możliwe, ponieważ podterm s może zostać zwielokrotniony w wyniku podstawienia). Stąd $\mu_2(\sigma) \sqsupseteq \mu_2(\sigma_1[x := s])$.
- Gdy $\sigma = P(s_1, \dots, s_n)$, to σ redukuje się do $\sigma' = P(s'_1, \dots, s'_n)$, gdzie $P \in \Sigma$ jest symbolem predykatu, dla pewnego $i \in \{1, \dots, n\}$ mamy $s_i \rightsquigarrow s'_i$, zaś dla $j \neq i$, mamy $s_j = s'_j$. Znowu, pierwsza współrzędna $\mu(\sigma)$ i $\mu(\sigma')$ jest równa zero. Druga współrzędna μ ostro się zmniejsza, gdyż $\mu_2(\sigma) = \bigcup_{l=1, \dots, n} \mu_2(s_l) \cup \text{nest}(\sigma)$ oraz $\mu_2(\sigma') = \bigcup_{l=1, \dots, n} \mu_2(s'_l) \cup \text{nest}(\sigma')$, zaś $\mu_2(s_l) \sqsupseteq \mu_2(s'_l)$ dla pewnego l oraz $\mu_2(s_j) = \mu_2(s'_j)$ dla $j \neq l$ i na mocy faktu 3.2.28 zachodzi $\text{nest}(\sigma) \geq \text{nest}(\sigma')$.
- Gdy $\sigma = \sigma_1 \rightarrow \sigma_2$, mamy dwa podprzypadki w zależności od typu σ' , będącego wynikiem kroku redukcji:

- Jeśli $\sigma' = \sigma'_1 \rightarrow \sigma_2$, gdzie $\sigma_1 \rightsquigarrow \sigma'_1$, to mamy $\mu(\sigma) = \langle k_1 + k_2, A_1 \cup A_2 \cup \{\text{nest}(\sigma)\} \rangle$ oraz $\mu(\sigma') = \langle k'_1 + k_2, A'_1 \cup A_2 \cup \{\text{nest}(\sigma')\} \rangle$, przy czym $\mu(\sigma_1) = \langle k_1, A_1 \rangle, \mu(\sigma_2) = \langle k_2, A_2 \rangle$ i $\mu(\sigma'_1) = \langle k'_1, A'_1 \rangle$. Z założenia indukcyjnego $\mu(\sigma_1) \sqsupseteq_\mu \mu(\sigma'_1)$. Ponieważ $\text{nest}(\sigma) \geq \text{nest}(\sigma')$, otrzymujemy z założenia indukcyjnego $\mu(\sigma) \sqsupseteq_\mu \mu(\sigma')$.
- Jeśli $\sigma' = \sigma_1 \rightarrow \sigma'_2$, gdzie $\sigma_2 \rightsquigarrow \sigma'_2$, to dowód jest podobny do dowodu z poprzedniego przypadku z dokładnością do zamiany roli σ_1 i σ_2 .
- Gdy $\sigma = ((\forall y:0)\sigma_1)$, to $\sigma \rightsquigarrow \sigma' = ((\forall y:0)\sigma'_1)$, przy czym $\sigma_1 \rightsquigarrow \sigma'_1$. Mamy $\mu(\sigma) = \langle \mu_1(\sigma_1), \{\text{nest}((\forall x:0)\sigma_1)\} \cup \mu_2(\sigma_1) \rangle$ oraz $\mu(\sigma') = \langle \mu_1(\sigma'_1), \{\text{nest}((\forall x:0)\sigma'_1)\} \cup \mu_2(\sigma'_1) \rangle$. Z założenia indukcyjnego mamy $\mu(\sigma_1) \sqsupseteq_\mu \mu(\sigma'_1)$ oraz z faktu 3.2.28 zachodzi $\text{nest}(\sigma) \geq \text{nest}(\sigma')$ to razem daje $\mu(\sigma) \sqsupseteq_\mu \mu(\sigma')$.
- Gdy $\sigma = (\sigma_1 \rightarrow \sigma_2)\langle x := s \rangle$ oraz $\sigma \rightsquigarrow \sigma' = (\sigma_1\langle x := s \rangle \rightarrow \sigma_2\langle x := s \rangle)$, gdzie $s \in T_\Sigma^e(X)$, $x \in X$. W tym przypadku ostro zmniejsza się już pierwsza współrzędna μ . Rzeczywiście, T_σ i $T_{\sigma'}$ zawierają dokładnie te same ścieżki, z dokładnością do ścieżek, które pochodzą z podtypu σ typu σ . Jednak każda taka ścieżka (o długości > 0) w T_σ ma sobie odpowiadającą ścieżkę w $T_{\sigma'}$, która pochodzi albo z $\sigma_1\langle \dots \rangle$, albo z $\sigma_2\langle \dots \rangle$ (w zależności od tego, czy kończy się na 1 lub 2). Stąd suma długości ścieżek w T_σ jest ściśle większa od sumy długości ścieżek w $T_{\sigma'}$, zatem μ_1 ostro się zwiększa podczas takiej redukcji.
- Gdy $\sigma = ((\forall y:0)\sigma_1)\langle x := s \rangle$ oraz $\sigma \rightsquigarrow \sigma' = ((\forall y:0)\sigma_1\langle x := s \rangle)$, to dowód jest podobny do poprzedniego przypadku.
- Gdy $\sigma = P(t_1, \dots, t_m)\langle x := s \rangle$ oraz $\sigma \rightsquigarrow \sigma' = P(t'_1, \dots, t'_m)$, gdzie $s \in T_\Sigma^e(X)$, $x \in X$, $P \in \Sigma$ i P ma arność m , zaś $t'_i = t_i\langle x := s \rangle$ dla $i = 1, \dots, m$, to pierwsza współrzędna μ zmniejsza się z 1 do 0.
- Gdy $\sigma = \tau\langle x := s \rangle$ oraz $\sigma \rightsquigarrow \sigma' = \tau'\langle x := s \rangle$, gdzie $s \in T_\Sigma^e(X)$, $x \in X$, oraz $\tau \rightsquigarrow \tau'$, to $\mu(\sigma) = \langle \mu_1(\tau) + k, \{\text{nest}(\tau\langle x := s \rangle)\} \cup \mu_2(\tau) \cup \mu_2(s) \rangle$ oraz $\mu(\sigma') = \langle \mu_1(\tau') + k', \{\text{nest}(\tau'\langle x := s \rangle)\} \cup \mu_2(\tau') \cup \mu_2(s) \rangle$, gdzie k to długość wszystkich ścieżek w $T_{\tau\langle x := s \rangle}$, a k' to długość wszystkich ścieżek w $T_{\tau'\langle x := s \rangle}$. Z założenia indukcyjnego wynika $\mu(\tau) \sqsupseteq_\mu \mu(\tau')$, z faktu 3.2.28 mamy $\text{nest}(\tau\langle x := s \rangle) \geq \text{nest}(\tau'\langle x := s \rangle)$. Dla pokazania $\mu(\sigma) \sqsupseteq_\mu \mu(\sigma')$ wystarczy pokazać, że $k \geq k'$. Zauważmy jednak, że redukcja w e-typach polega na przesyłaniu substytucji w głąb e-typu, a taka operacja nie zwiększa długości ścieżek wiodących do liści (tzn. zmiennych typowych lub predykatów).

□

Wniosek 3.2.31 (Church-Rosser dla e-typów)*Redukcja \rightsquigarrow ma własność Churcha-Rossera.*

Dowód:

Silna normalizacja z faktu 3.2.30 i słaba konfluencja z faktu 3.2.25 dają łącznie na mocy lematu Newmana własność Churcha-Rossera. \square

Fakt 3.2.32 (\rightsquigarrow^* jest rozstrzygalna)

Istnieje efektywna procedura sprawdzająca, czy $\tau_1 \rightsquigarrow^ \tau_2$.*

Dowód:

W związku z tym, że system redukcyjny \rightsquigarrow ma na mocy faktu 3.2.30 własność silnej normalizacji, to możemy po prostu sprawdzać, czy typ τ_2 pojawia się na którejś ścieżce redukcyjnej zaczynającej się od τ_1 . \square

Definicja 3.2.33 (postać normalna \rightsquigarrow^*)

Fakt 3.2.25 oraz wniosek 3.2.31 implikują, że dobrze zdefiniowane jest pojęcie postaci normalnej e-typu. Dla e-typu τ definiujemy jego postać normalną $\text{NF}^{\rightsquigarrow^*}(\tau)$ jako typ τ' , taki że $\tau \rightsquigarrow^* \tau'$ oraz dla każdego τ'' mamy $\tau' \not\rightsquigarrow \tau''$.

Zachodzi następujący ciekawy fakt, który wskazuje na pewne istotne intuicje dotyczące zdefiniowanej tutaj redukcji.

Fakt 3.2.34 (substytucje w postaci normalnej)

Jeśli τ jest e-typem w postaci normalnej, dla którego $\text{TV}(\tau) = \emptyset$, to τ nie ma podtypu (podtermu) postaci $\sigma\langle x := t \rangle$ ($s\langle x := t \rangle$), a zatem jest pseudotypem rachunku λP .

Dowód:

Założmy, że w τ istnieje podtyp postaci $\sigma\langle x := s \rangle$. Możemy bez utraty ogólności założyć, że σ nie ma już podtypu tej samej postaci. Typ σ może mieć kilka postaci:

- może być e-termem — w tym momencie można zaaplikować regułę redukcji (3) z definicji 3.2.23;
- może być e-typem postaci $P(s_1, \dots, s_n)$ — w tym momencie można zaaplikować regułę redukcji (11) z definicji 3.2.23;
- może być e-typem postaci $\sigma_1 \rightarrow \sigma_2$ — w tym momencie można zaaplikować regułę redukcji (9) z definicji 3.2.23;
- może być e-typem postaci $(\forall x:0)\sigma_1$ — w tym momencie można zaaplikować regułę redukcji (10) z definicji 3.2.23.

W każdym przypadku τ nie jest więc w postaci normalnej. \square

Fakt 3.2.35 (wyprowadzenia dla e-typów)

Niech τ będzie e-typem nad sygnaturą Σ , przy czym $\text{TV}(\tau) = \emptyset$. Osąd $\Gamma_\Sigma \cup \Gamma' \vdash \text{NF}^{\rightsquigarrow^}(\tau) : *$ ma wyprowadzenie pierwszego rzędu w λP dla pewnego Γ' , takiego że $\Gamma'(x) = 0$ dla każdego $x \in \text{Vars}(\tau)$.*

Dowód:

Indukcja ze względu na strukturę e-typu $NF^{\rightsquigarrow}(\tau)$. Dowód zostawiamy Czytelnikowi. \square

Definicja 3.2.36 (równość e-termów i e-typów)

Równość e-termów i e-typów jest zdefiniowana jako najmniejsza kongruencja zawierająca $\rightsquigarrow^* \cup \equiv_\alpha$. Oznaczamy tę relację przez \simeq .

Definicja 3.2.37 (e-równanie)

Dowolną parę e-typów nazwiemy *e-równaniem*. Gdy będziemy chcieli podkreślić fakt, że para e-typów $\langle \tau_1, \tau_2 \rangle$ stanowi równanie, będziemy stosować notację $\tau_1 \doteq \tau_2$. Zbiory e-równań oznaczamy symbolami $\mathcal{E}, \mathcal{F}, \dots$. Napis $\mathcal{E}_\Sigma^e(X, \mathcal{X})$ oznacza zbiór wszystkich podzbiorów $\mathcal{T}_\Sigma^e(X, \mathcal{X})^2$. Zbiory równań nazywamy też czasami *układami równań*.

Skoro mamy równania, to musimy też zdefiniować podstawienia, pośród których będziemy znajdować rozwiązania dla zbiorów e-równań.

Definicja 3.2.38 (podstawienia)

Każda funkcja częściowa ze zbioru zmiennych typowych do pewnego zbioru typów $\mathcal{T}_\Sigma^e(X, \mathcal{Y})$ nazywana jest *podstawieniem*.

Wynik zastosowania podstawienia $S : \mathcal{X} \rightarrow \mathcal{T}_\Sigma^e(X, \mathcal{Y})$ do e-typu τ , oznaczany przez $S(\tau)$, jest indukcyjnie zdefiniowany jako (mamy tutaj lekkie nadużycie notacji, gdyż stosujemy funkcję nie do elementów jej dziedziny):

- $S(0) = 0$;
- $S(P(t_1, \dots, t_m)) = P(t_1, \dots, t_m)$;
- $S(\alpha) = \alpha$, o ile $\alpha \notin \text{Dom}(S)$;
- $S(\alpha) = S(\alpha)$, o ile $\alpha \in \text{Dom}(S)$;
- $S((\forall x : 0)\sigma_2) = (\forall x : 0)S(\sigma_2)$.
- $S(\sigma_1 \rightarrow \sigma_2) = S(\sigma_1) \rightarrow S(\sigma_2)$.
- $S(\sigma\langle x := s \rangle) = S(\sigma)\langle x := s \rangle$.

Zauważmy, że w powyższej definicji nie mamy żadnej formy przemianowywania zmiennych pierwszego podczas podstawiania typu na zmienną typową znajdującą się w zasięgu kwantyfikatora. Takie rozwiązanie w tym miejscu nie jest przypadkowe. Przyjmujemy, że niektóre symbole mogą zostać związane podczas takiego podstawienia.

Definicja 3.2.39 (rozwiązanie układu równań)

Mówimy, że podstawienie $S : \mathcal{X} \rightarrow \mathcal{T}_\Sigma^e(X, \emptyset)$ jest rozwiązaniem układu e-równań \mathcal{E} , gdy dla każdego e-równania $[\tau_1 \doteq \tau_2] \in \mathcal{E}$ mamy $S(\tau_1) \simeq S(\tau_2)$.

Dla tak zdefiniowanych równań niestety nie mamy własności najbardziej ogólnego rozwiązania.

Przykład 1 Rozważmy sygnaturę $\Sigma = \{P:0 \Rightarrow *, c:0\}$. Zbiór równań $\mathcal{E} = \{\alpha\langle x := c \rangle \langle y := c \rangle \doteq P(c)\}$ ma jedyne dwa rozwiązania $S_1(\alpha) = P(x)$ oraz $S_2(\alpha) = P(y)$, jednak nie istnieje żadne S_3 , takie że $S_3 \circ S_2 = S_1$ ani też $S_3 \circ S_1 = S_2$. Stąd ani S_1 , ani S_2 nie może być najbardziej ogólnym rozwiązaniem.

Generowanie równań na e-typach

Zajmiemy się teraz problemem, jak dla danego kontekstu Γ oraz termu Curry'ego M wygenerować układ e-równań, który będzie rozwiązywalny wtedy i tylko wtedy, gdy będzie istniało wyprowadzenie pierwszego rzędu dla $\Gamma \vdash M : \tau$ dla pewnego typu pierwszego rzędu τ . Aby sformułować poprawne założenie indukcyjne musimy troszeczkę rozszerzyć dotychczas używane pojęcie kontekstu.

Definicja 3.2.40 (wzbogacony kontekst pierwszego rzędu)

Kontekst Γ jest *wzbogaconym kontekstem pierwszego rzędu*, gdy ma postać $\Gamma_0 \cup \{x_1 : \tau_1, \dots, x_n : \tau_n\}$, gdzie Γ_0 jest jakimś kontekstem pierwszego rzędu nad sygnaturą Σ , symbole x_1, \dots, x_n są świeżymi zmiennymi przedmiotowymi, zaś $\tau_1, \dots, \tau_n \in \mathcal{T}_\Sigma^e(X, \mathcal{X})$ dla pewnego X i \mathcal{X} . Przez Γ^{var} oznaczymy zbiór zmiennych $\{x_1 : \tau_1, \dots, x_n : \tau_n\}$. Do zbioru $\text{TV}(\Gamma)$ należą wszystkie elementy \mathcal{X} , które występują w Γ .

Definiujemy $S(\Gamma)^{\rightsquigarrow}$ dla kontekstu Γ jako ciąg Γ , w którym każde $x : \tau$ zostało zastąpione przez $x : \text{NF}^{\rightsquigarrow}(S(\tau))$.

Definicja 3.2.41 (generowanie równań)

Przedstawimy tutaj niedeterministyczną procedurę **gener**, która na wejściu bierze term Curry'ego M , wzbogacony kontekst pierwszego rzędu Γ oraz ścieżkę ρ (w intencji prowadzącą do podtermu M w jakimś większym termie) i która generuje zbiór e-równań zawarty w $\mathcal{E}_{\Sigma \cup \{c_0, c_1\}}^e(X, \mathcal{X})$, gdzie c_0, c_1 są świeżymi stałymi pierwszego rzędu, a X jest zbiorem zmiennych mających typ 0 w Γ .

1. **gener** $(x, \Gamma, \rho) = \{\alpha_{x,\rho} \doteq \Gamma(x)\}$, gdy $x \in \text{Dom}(\Gamma)$ oraz $\Gamma(x) \neq 0$;
2. **gener** $(x, \Gamma, \rho) = \{c_0 \doteq c_1\}$, gdy $x \notin \text{Dom}(\Gamma)$ lub $x \in \text{Dom}(\Gamma)$ i $\Gamma(x) = 0$;
3. **gener** $(MN, \Gamma, \rho) = \{\alpha_{M,1 \cdot \rho} \doteq (\forall x : 0) \alpha_{MN,\rho}^0, \alpha_{MN,\rho}^0 \langle x := N \rangle \doteq \alpha_{MN,\rho}\} \cup \mathcal{E}_M \cup \mathcal{E}_N$, o ile N jest homogenicznym termem pierwszego rzędu; x jest tutaj świeżą zmienną pierwszego rzędu, $\mathcal{E}_M = \mathbf{gener}(M, \Gamma, 1 \cdot \rho)$ a $\mathcal{E}_N = \mathbf{gener}(N, \Gamma, 2 \cdot \rho)$;

4. $\mathbf{gener}(MN, \Gamma, \rho) = \{\alpha_{M,1,\rho} \doteq \alpha_{N,2,\rho} \rightarrow \alpha_{MN,\rho}\} \cup \mathcal{E}_M \cup \mathcal{E}_N$, o ile N nie jest homogenicznym termem pierwszego rzędu; $\mathcal{E}_M = \mathbf{gener}(M, \Gamma, 1 \cdot \rho)$, a $\mathcal{E}_N = \mathbf{gener}(N, \Gamma, 2 \cdot \rho)$;
5. niedeterministycznie wybieramy (5a) lub (5b):
 - (a) $\mathbf{gener}(\lambda x.M, \Gamma, \rho) = \{\alpha_{\lambda x.M,\rho} \doteq \alpha_{x,\rho} \rightarrow \alpha_{M,1,\rho}\} \cup \mathcal{E}_M$, gdzie $\mathcal{E}_M = \mathbf{gener}(M, \Gamma \cup \{x : \alpha_{x,\rho}\}, 1 \cdot \rho)$,
 - (b) $\mathbf{gener}(\lambda x.M, \Gamma, \rho) = \{\alpha_{\lambda x.M,\rho} \doteq (\forall x:0)\alpha_{M,1,\rho}\} \cup \mathcal{E}_M$, gdzie $\mathcal{E}_M = \mathbf{gener}(M, \Gamma \cup \{x : 0\}, 1 \cdot \rho)$.

Obecnie pokażemy dwie kluczowe własności tego algorytmu, które zapewniają, że algorytm **gener** generuje równania, których rozwiązywalność jest równoważna istnieniu wyprowadzenia.

Lemat 3.2.42 (od wyprowadzania typów do równań)

Niech ρ będzie ścieżką, Γ wzbogaconym kontekstem pierwszego rzędu, w którym nie występują zmienne typowe postaci $\alpha_{N,\rho' \cdot \rho}$, gdzie N jest λ -termem, a ρ' ścieżką. Niech Γ^0 będzie zbiorem $\{x \mid x:0 \in \Gamma\}$, zaś M będzie termem Curry'ego. Jeśli istnieje typ τ (postaci różnej od $0 \rightarrow \dots \rightarrow 0$ i podstawienie $S_0 : \mathcal{X} \rightarrow \mathcal{T}_\Sigma^e(\Gamma^0, \emptyset)$, takie że $S_0(\Gamma) \rightsquigarrow^* \vdash M : \tau$ ma wyprowadzenie pierwszego rzędu \mathcal{P} , to istnieje bieg procedury $\mathbf{gener}(M, \Gamma, \rho)$, którego wynik ma rozwiązanie $S : \mathcal{X} \rightarrow \mathcal{T}_\Sigma^e(\Gamma^0, \emptyset)$ o własnościach:

- dla każdego N będącego podtermem M wskazywanym przez ścieżkę ρ' mamy $S(\alpha_{N,\rho' \cdot \rho}) \rightsquigarrow^* \tau_N$, gdzie $\Gamma' \vdash N : \tau_N$ jest osądem występującym w \mathcal{P} i odpowiadającym ρ' ;
- dla $\alpha \in \text{Dom}(S_0)$ mamy $S(\alpha) = S_0(\alpha)$.

Dowód:

Dowód przeprowadzamy przez indukcję ze względu na budowę termu M . Niech $\mathcal{X} = \{\alpha_{N,\rho'} \mid \rho' \succeq \rho\}$ (\succeq oznacza porządek sufiksowy na ścieżkach). Bez utraty ogólności możemy założyć, że $\text{Dom}(S_0) \cap \mathcal{X} = \emptyset$. (Zauważmy, że $\text{Dom}(S_0) = \text{TV}(\Gamma)$.)

Przypadek $M \equiv x$. Przypuśćmy, że $S_0(\Gamma) \vdash x : \tau$ ma wyprowadzenie pierwszego rzędu \mathcal{P} . Wyprowadzenie w tej sytuacji może kończyć się tylko regułą (var). Bieg **gener** określamy w tym przypadku jako składający się z jednego wykonania kroku (1) **gener**, który generuje zbiór $\{\alpha_{x,\rho} \doteq \Gamma(x)\}$. Określamy $S = S_1 \circ S_0$, gdzie $S_1(\alpha_{x,\rho}) = \text{NF}^{\rightsquigarrow^*}(S_0(\Gamma(x)))$. Oczywiście dla $\alpha \in \text{Dom}(S_0)$ mamy $S(\alpha) = S_0(\alpha)$. Aby zakończyć dowód obecnego przypadku, wystarczy pokazać, że

- a. $S(\alpha_{x,\rho}) \simeq S(\Gamma(x))$, oraz
- b. $S(\alpha_{x,\rho}) \rightsquigarrow^* \tau$.

Dla dowodu (a) postępujemy w sposób następujący: $S(\alpha_{x,\rho}) = S_1 \circ S_0(\alpha_{x,\rho}) = S_1(\alpha_{x,\rho})$, gdyż $\text{Dom}(S_0) \cap \mathcal{X} = \emptyset$, i dalej $S_1(\alpha_{x,\rho}) = \text{NF}^{\rightsquigarrow}(S_0(\Gamma(x))) \simeq S_0(\Gamma(x)) = S_1 \circ S_0(\Gamma(x)) = S(\Gamma(x))$, gdyż $\text{Dom}(S_1) \subseteq \mathcal{X}$, kontekst Γ nie zawiera zmiennej typowej z \mathcal{X} nie ma żadnej zmiennej typowej w typach obrazu S_0 .

Dla dowodu (b) postępujemy w sposób następujący: mamy

$$S(\alpha_{x,\rho}) = S_1 \circ S_0(\alpha_{x,\rho}) = S_1(\alpha_{x,\rho}) = \text{NF}^{\rightsquigarrow}(S_0(\Gamma(x))).$$

Ponieważ \mathcal{P} może kończyć się tylko regułą (var), to $S_0(\Gamma)^{\rightsquigarrow*} \vdash x : \tau$ i mamy $\tau = \text{NF}^{\rightsquigarrow}(S_0(\Gamma(x)))$.

Przypadek $M \equiv M_1 M_2$. Przypuśćmy, że $S_0(\Gamma)^{\rightsquigarrow} \vdash M_1 M_2 : \tau$ ma wyprowadzenie pierwszego rzędu \mathcal{P} . Wyprowadzenie to musi kończyć się regułą (app) — decyduje o tym postać M . Stąd $S_0(\Gamma)^{\rightsquigarrow} \vdash M_1 : (\forall x : \sigma)\tau$ i $S_0(\Gamma)^{\rightsquigarrow} \vdash M_2 : \sigma$ mają wyprowadzenia pierwszego rzędu dla pewnych typów σ oraz τ . Z założenia indukcyjnego dostajemy dwa podstawienia S_1 oraz S_2 rozwiązujące wyniki pewnych biegów $\mathbf{gener}(M_1, \Gamma, 1 \cdot \rho)$ oraz $\mathbf{gener}(M_2, \Gamma, 2 \cdot \rho)$ odpowiednio. Zbiór $\text{Dom}(S_1) \cap \text{Dom}(S_2)$ jest zawarty w $\text{TV}(\Gamma^{var})$, ponieważ M_1 i M_2 są typowane we wspólnym kontekście, a odpowiadające im zbiory zmiennych typowych nie mają wspólnych świeżych zmiennych typowych (świeże zmienne mają adnotacje odpowiednio $1 \cdot \rho$ i $2 \cdot \rho$). Dla $y \in \text{Dom}(\Gamma^{var})$ e-typy $S_1(\Gamma(y))$ i $S_2(\Gamma(y))$ są równe. Wynika to stąd, że dla każdej zmiennej $y \in \text{Dom}(\Gamma^{var})$ mamy $\text{TV}(\Gamma(y)) \subseteq \text{Dom}(S_0)$ a dla $\alpha \in \text{Dom}(S_0)$ zachodzi $S_1(\alpha) = S_0(\alpha) = S_2(\alpha)$. To oznacza, że możemy określić $S' = S_1 \cup S_2$, które rozwiązuje wspólnie $\mathcal{E}_{M_1} = \mathbf{gener}(M_1, \Gamma, 1 \cdot \rho)$ oraz $\mathcal{E}_{M_2} = \mathbf{gener}(M_2, \Gamma, 2 \cdot \rho)$. Podstawienie S' oczywiście ma wszystkie wymagane w naszej tezie własności dla $\rho' = 2 \cdot \rho''$ i $\rho' = 1 \cdot \rho''$ oraz $\alpha \in \text{Dom}(S_0)$ — wynika to z założenia indukcyjnego. Aby określić właściwe S , wystarczy S' dookreślić tak, by dodatkowo rozwiązywało

- a. $\{\alpha_{M_1, 1 \cdot \rho} \doteq \alpha_{M_2, 2 \cdot \rho} \rightarrow \alpha_{M_1 M_2, \rho}\}$, gdy M_2 nie jest homogenicznym termem pierwszego rzędu lub
- b. $\{\alpha_{M_1, 1 \cdot \rho} \doteq (\forall x : 0)\alpha_{M_1 M_2, \rho}^0, \alpha_{M_1 M_2, \rho}^0 \langle x := M_2 \rangle \doteq \alpha_{M_1 M_2, \rho}\}$, gdy term M_2 jest homogenicznym termem pierwszego rzędu

W przypadku (a) wystarczy określić $S = S' \cup \{(\alpha_{M_1 M_2, \rho}, 2(\sigma))\}$, gdzie σ stanowi \rightsquigarrow -normalną postać $S'(\alpha_{M_1, 1 \cdot \rho})$. Można tak zrobić, ponieważ

$$\text{typ } 2(\sigma) \text{ jest dobrze określony.} \tag{3.4}$$

Zdanie powyższe wynika stąd, że osąd $S_0(\Gamma)^{\rightsquigarrow} \vdash M_1 : \tau'$, który ma własność, taką że $S'(\alpha_{M_1, 1 \cdot \rho}) \simeq \tau'$, jest lewym założeniem ostatniej reguły (app) w \mathcal{P} . (Zauważmy, że ponieważ \mathcal{P} jest wyprowadzeniem pierwszego rzędu, τ' jest w rzeczywistości typem strzałkowym; typy kwantyfikatorowe zgodnie z faktem 3.2.11 występują tylko w sytuacji, gdy M_2 jest homogenicznym termem pierwszego rzędu.) Tak zdefiniowane S jest oczywiście rozwiązaniem

wcześniej wspomnianych biegów $\mathbf{gener}(M_1, \Gamma, 1 \cdot \rho)$ i $\mathbf{gener}(M_2, \Gamma, 2 \cdot \rho)$. Aby udowodnić rozwiązywalność $\mathbf{gener}(M_1 M_2, \Gamma, \rho)$ konieczne jest dodatkowe sprawdzenie, czy $S(\alpha_{M_1, 1 \cdot \rho}) \simeq S(\alpha_{M_2, 2 \cdot \rho} \rightarrow \alpha_{M_1 M_2, \rho})$. To zaś jest prawdą, gdyż postać normalna $S(\alpha_{M_1, 1 \cdot \rho})$ ma, jak wynika z (3.4), postać $\pi \rightarrow \tau$. Musimy jeszcze sprawdzić, że $S(\alpha_{M_1 M_2, \rho}) \rightsquigarrow^* \tau$. To zaś zachodzi, ponieważ postaci normalne są jednoznaczne.

W przypadku (b) wystarczy przyjąć

$$S = S' \cup \{(\alpha_{M_1 M_2, \rho}^0, 2(\sigma)), (\alpha_{M_1 M_2, \rho}, \sigma')\},$$

gdzie σ jest \rightsquigarrow -normalną postacią $S'(\alpha_{M_1, 1 \cdot \rho})$, zaś σ' jest \rightsquigarrow -normalną postacią $r(\sigma)[x := M_2]$. To dokładnie odpowiada rozwiązywalności dodatkowo wygenerowanych równań. Należy jeszcze sprawdzić, że $S(\alpha_{M_1 M_2, \rho}) \rightsquigarrow^* \tau$. To zaś jest prawdą, ponieważ postać normalna $S(\alpha_{M_1, 1 \cdot \rho})$ jest równa $(\forall x : 0)\tau'$, gdzie $\tau'[x := M_2] = \tau$, co wynika z faktu, że derywacja w λP kończy się regułą (app) oraz z założenia indukcyjnego. To kończy dowód pierwszego warunku dotyczącego ścieżki ρ . Pierwszy warunek dla $\rho' \succ \rho$ wynika z założenia indukcyjnego dla M_1 i M_2 .

Pozostaje udowodnić, że dla $\alpha \in \text{Dom}(S_0)$ mamy $S(\alpha) = S_0(\alpha)$. Warunek ten zachodzi dla S_1 i S_2 , przy czym jedyne nowe w stosunku do S_1 i S_2 zmienne w S to zmienne z adnotacją ρ . Z założenia takie zmienne nie mogą pojawić się w $\text{TV}(\Gamma)$.

Przypadek $M \equiv \lambda x.M'$. Przypuśćmy, że $S_0(\Gamma) \rightsquigarrow \vdash \lambda x.M' : \tau$ ma wyprowadzenie pierwszego rzędu. Oznacza to, że $\tau = (\forall x : \tau_1)\tau_2$. Osąd $S_0(\Gamma) \rightsquigarrow \cup x : \tau_1 \vdash M' : \tau_2$ ma wyprowadzenie pierwszego rzędu \mathcal{P}' , które stanowi część wyprowadzenia \mathcal{P} . Ponieważ τ_1 nie ma wystąpień zmiennych typowych, dostajemy $S_0(\Gamma \cup x : \tau_1) \rightsquigarrow \vdash M' : \tau_2$ i możemy zastosować założenie indukcyjne. W wyniku dostajemy, że układ $\mathbf{gener}(M', \Gamma \cup \{x : \sigma\}, 1 \cdot \rho)$, gdzie $\sigma = 0$ lub $\sigma = \alpha_{x, \rho}$, w zależności od tego, czy $\tau_1 = 0$ lub $\tau_1 \neq 0$ odpowiednio, ma rozwiązanie S' , takie że dla każdego podtermu N termu M' wskazywanego przez ścieżkę ρ' typ N w \mathcal{P}' jest postacią \rightsquigarrow -normalną e-typu $S'(\alpha_{N, \rho' \cdot 1 \cdot \rho})$. Określamy $S = S' \cup \{(\alpha_{\lambda x.M', \rho}, S'(\alpha_{x, \rho}) \rightarrow S'(\alpha_{M', 1 \cdot \rho}))\}$ (lub $S = S' \cup \{(\alpha_{\lambda x.M', \rho}, (\forall x : 0)S'(\alpha_{M', 1 \cdot \rho}))\}$). Z założenia indukcyjnego uzyskujemy, że dla każdego podtermu N wskazywanego przez ścieżkę ρ' typ N w \mathcal{P}' jest postacią \rightsquigarrow -normalną $S'(\alpha_{N, \rho' \cdot 1 \cdot \rho})$, a zatem $S'(\alpha_{M', 1 \cdot \rho}) \rightsquigarrow^* \tau_2$. Stąd reguła (abs) daje nam wyprowadzenie \mathcal{P} , a $S(\alpha_{\lambda x.M', \rho}) \rightsquigarrow^* (\forall x : \tau_1)\tau_2$ (lub $\tau_1 \rightarrow \tau_2$). W \mathbf{gener} wybierana jest opcja (5a), gdy τ_1 jest różny od 0, zaś opcja (5b) w przeciwnym przypadku.

Założenie indukcyjne oraz fakt, że przy wykonywaniu kroku indukcyjnego nie zmienialiśmy nic w części podstawienia dotyczącej podtermów, implikują, że S jest równa S_0 na zmiennych z $\text{Dom}(S_0)$. \square

Lemat 3.2.43 (od równań do wyprowadzania typów)

Niech Γ będzie wzbogaconym kontekstem pierwszego rzędu, a M termem Curry'ego. Jeśli wynik pewnego biegu $\mathbf{gener}(M, \Gamma, \rho)$ ma rozwiązanie S , to

istnieje wyprowadzenie pierwszego rzędu \mathcal{P} osądu $S(\Gamma)^{\rightsquigarrow} \vdash M : \tau_M$, które dla każdego podtermu N występującego w M na ścieżce ρ' spełnia $S(\alpha_{N,\rho'.\rho}) \rightsquigarrow^* \tau_N$, gdzie τ_N jest typem wystąpienia N w \mathcal{P} .

Dowód:

Dowód przez indukcję ze względu na budowę termu M .

Przypadek $M \equiv x$. Sytuacja taka, że $\mathbf{gener}(x, \Gamma, \rho) = \{c_0 \doteq c_1\}$ jest niemożliwa, gdyż $\mathbf{gener}(x, \Gamma, \rho)$ ma rozwiązanie. Stąd $\mathbf{gener}(x, \Gamma, \rho) = \{\alpha_{x,\rho} \doteq \Gamma(x)\}$. Jeśli ten zbiór ma rozwiązanie S , to $S(\alpha_{x,\rho}) \simeq S(\Gamma(x))$. Stosowne wyprowadzenie składa się z reguły (var), przed którą znajduje się wyprowadzenie stwierdzające, że kontekst wzbogacony o e-typ $\text{NF}^{\rightsquigarrow}(S(\alpha_{x,\rho}))$ jest poprawny — wynika to z faktu 3.2.35.

Przypadek $M \equiv M_1 M_2$. Przypuśćmy, że S jest rozwiązaniem pewnego wyniku $\mathbf{gener}(M_1 M_2, \Gamma, \rho)$. Ponieważ

$$\mathbf{gener}(M_1, \Gamma, 1 \cdot \rho) \cup \mathbf{gener}(M_2, \Gamma, 2 \cdot \rho) \subseteq \mathbf{gener}(M_1 M_2, \Gamma, \rho)$$

w każdym ustalonym biegu, uzyskujemy, że S jest rozwiązaniem układu równań $\mathbf{gener}(M_1, \Gamma, 1 \cdot \rho)$ oraz $\mathbf{gener}(M_2, \Gamma, 2 \cdot \rho)$. Z założenia indukcyjnego uzyskujemy wyprowadzenia \mathcal{P}_1 oraz \mathcal{P}_2 osądów $S(\Gamma)^{\rightsquigarrow} \vdash M_1 : \tau_{M_1}$ i $S(\Gamma)^{\rightsquigarrow} \vdash M_2 : \tau_{M_2}$. Co więcej, każdy właściwy podterm N termu $M_1 M_2$ na ścieżce ρ' ma typ τ_N , który jest postacią \rightsquigarrow -normalną $S(\alpha_{N,\rho'.d,\rho})$, gdzie $d = 1, 2$. Zbiór $\mathbf{gener}(M_1 M_2, \Gamma, \rho)$ zawiera następujące dodatkowe równania:

- a. $\{\alpha_{M_1,1,\rho} \doteq \alpha_{M_2,2,\rho} \rightarrow \alpha_{M_1 M_2,\rho}\}$, gdy M_2 nie jest homogenicznym termem pierwszego rzędu, lub
- b. $\{\alpha_{M_1,1,\rho} \doteq (\forall x : 0)\alpha_{M_1 M_2,\rho}^0, \alpha_{M_1 M_2,\rho}^0 \langle x := M_2 \rangle \doteq \alpha_{M_1 M_2,\rho}\}$, gdy term M_2 jest homogenicznym termem pierwszego rzędu.

W przypadku (a) wiemy na podstawie postaci równania, że typ M_1 w \mathcal{P}_1 ma postać $\sigma_1 \rightarrow \sigma_2$, gdzie σ_1 i σ_2 są postaciami \rightsquigarrow -normalnymi $S(\alpha_{M_2,2,\rho})$ oraz $S(\alpha_{M_1 M_2,\rho})$ odpowiednio. Możemy połączyć \mathcal{P}_1 i \mathcal{P}_2 w jedno wyprowadzenie przy pomocy reguły (app), gdyż typy M_1 i M_2 w \mathcal{P}_1 oraz \mathcal{P}_2 są z założenia indukcyjnego równe odpowiednio $\sigma_1 \rightarrow \sigma_2$ i σ_1 . Typ wynikowy dla $M_1 M_2$ jest postacią \rightsquigarrow -normalną $S(\alpha_{M_1 M_2,\rho})$.

W przypadku (b) wiemy na podstawie postaci równania, że typ M_1 jest typem kwantyfikatorowym postaci $(\forall x : 0)\tau$ i, ponieważ M_2 jest homogeniczny, to jego typem jest 0. Stąd możemy połączyć \mathcal{P}_1 oraz \mathcal{P}_2 w jedną derywację przy pomocy reguły (app). Wynikowy typ dla $M_1 M_2$ jest równy $\tau[x := M_2]$, jednak postać równań implikuje, że τ jest postacią \rightsquigarrow -normalną $S(\alpha_{M_1 M_2,\rho}^0)$ oraz, że $S(\alpha_{M_1 M_2,\rho}) \simeq S(\alpha_{M_1 M_2,\rho}^0 \langle x := M_2 \rangle)$. Oznacza to, że $S(\alpha_{M_1 M_2,\rho}) \simeq S(\alpha_{M_1 M_2,\rho}^0)[x := M_2]$, a zatem $S(\alpha_{M_1 M_2,\rho}) \simeq \tau[x := M_2]$. Jednoznaczność postaci \rightsquigarrow -normalnych (fakty 3.2.25 oraz 3.2.30) implikuje, że $\tau[x := M_2]$ jest postacią normalną $S(\alpha_{M_1 M_2,\rho})$, a zatem $S(\alpha_{M_1 M_2,\rho}) \rightsquigarrow^* \tau[x := M_2]$.

Przypadek $M \equiv \lambda x.M'$. Przypuśćmy, że mamy rozwiązanie S pewnego biegu $\mathbf{gener}(\lambda x.M', \Gamma, \rho)$. Podstawienie S jest również rozwiązaniem $\mathbf{gener}(M', \Gamma \cup \{x : \alpha_{x,\rho}\}, 1 \cdot \rho)$ (jeśli niedeterministycznie został wybrany przypadek (5b), to jest rozwiązaniem $\mathbf{gener}(M', \Gamma \cup \{x : 0\}, 1 \cdot \rho)$). Stąd z założenia indukcyjnego mamy wyprowadzenie \mathcal{P}' dla $S(\Gamma \cup x : \alpha_{x,\rho}) \rightsquigarrow \vdash M' : \sigma'$ (lub $S(\Gamma \cup x : 0) \rightsquigarrow \vdash M' : \sigma'$). Dla każdego znajdującego się na ścieżce ρ' podtermu N termu M' mamy z założenia indukcyjnego, że typ τ_N termu N na ρ' w wyprowadzeniu \mathcal{P}' jest postacią \rightsquigarrow -normalną $S(\alpha_{N,\rho'.1,\rho})$. Możemy teraz na końcu \mathcal{P}' dodać regułę (abs) i uzyskać wyprowadzenie \mathcal{P} dla $S(\Gamma) \rightsquigarrow \vdash \lambda x.M' : \sigma \rightarrow \sigma'$ (lub $S(\Gamma) \rightsquigarrow \vdash \lambda x.M' : (\forall x : 0)\sigma'$), gdzie $\sigma = \mathbf{NF}^{\rightsquigarrow}(S(\alpha_{x,\rho}))$ i $\sigma' = \mathbf{NF}^{\rightsquigarrow}(S(\alpha_{M',1,\rho}))$. Ponieważ S rozwiązuje równanie $[\alpha_{x,\rho} \rightarrow \alpha_{M',1,\rho} \doteq \alpha_{\lambda x.M',\rho}]$ (lub $[(\forall x : 0)\alpha_{M',1,\rho} \doteq \alpha_{\lambda x.M',\rho}]$), postać normalna $S(\alpha_{\lambda x.M',\rho})$ musi mieć kształt $\sigma_1 \rightarrow \sigma_2$ (lub $(\forall x : 0)\sigma_2$), gdzie σ_1 jest postacią normalną $S(\alpha_{x,\rho})$, a σ_2 jest postacią normalną $S(\alpha_{M',1,\rho})$. Stąd term $\lambda x.M'$ ma typ $\mathbf{NF}^{\rightsquigarrow}(S(\alpha_{\lambda x.M',\rho}))$ w \mathcal{P} . \square

Twierdzenie 3.2.44 (równoważność wyprowadzania typów i e-równań)

Istnieje niedeterministyczny algorytm, dla którego przy dowolnym kontekście pierwszego rzędu Γ oraz termie Curry'ego M spełniona jest równoważność zdań:

1. Istnieje typ τ , taki że $\Gamma \vdash M : \tau$ ma wyprowadzenie pierwszego rzędu.
2. Istnieje bieg algorytmu, który daje w wyniku rozwiązywalny zbiór e-równań \mathcal{E} .

Dowód:

Algorytm z tezy twierdzenia to **gener**. Procedura ta daje niedeterministycznie w wyniku zbiór równań \mathcal{E} . Implikację (\Rightarrow) uzyskujemy jako natychmiastową konsekwencję lematu 3.2.42 zastosowaną do specjalnego przypadku, gdy $\rho = \varepsilon$ (ścieżka pusta) oraz żadna zmienna z \mathcal{X} nie występuje w Γ . Implikacja (\Leftarrow) wynika z lematu 3.2.43. \square

Upraszczenie równań

W ogólności e-typy w równaniach z poprzedniej sekcji zawierają kwantyfikatory i typy strzałkowe. Obecnie pozbędziemy się strzałek przy pomocy procedury podobnej do procedury stosowanej przy unifikacji pierwszego rzędu.

Wprowadzimy tutaj pewien skrót notacyjny — będziemy pisali $\forall^n \tau$ na oznaczenie e-typu postaci $(\forall x_1 : 0) \cdots (\forall x_n : 0)\tau$, gdzie $n \geq 0$. Aby móc identyfikować różne \forall^n , będziemy czasami dodawali do tego symbolu odpowiedni rozróżniający indeks.

Wprowadzimy teraz pewną pomocniczą definicję, która pozwoli nam zdefiniować algorytm upraszczania

Definicja 3.2.45 (graf zmiennych)

Dla danego zbioru równań \mathcal{E} , określamy graf z wagami $G_{\mathcal{E}} = \langle V, E_s \cup E_e, w \rangle$, gdzie

- $V = \text{TV}(\mathcal{E})$ oraz
- $(\alpha_1, \alpha_2) \in E_s$, o ile istnieje takie równanie $[\tau_1 \doteq \tau_2] \in \mathcal{Q}$, że α_1 występuje na pewnej ścieżce ρ w τ_1 oraz α_2 występuje na ścieżce $\rho' \cdot \rho$ w τ_2 , gdzie $|\rho'| > 0$; określamy tutaj $w(\alpha_1, \alpha_2) = |\rho'| - k$, gdzie k jest liczbą wystąpień e-typu postaci $\tau(x := s)$ na ścieżce ρ' w e-typie $\rho(\tau_2)$.
- $(\alpha_1, \alpha_2) \in E_e$, o ile istnieje równanie $[\tau_1 \doteq \tau_2] \in \mathcal{Q}$ oraz ścieżka ρ , która w τ_1 prowadzi do α_1 , a w τ_2 do α_2 ; określamy tutaj $w(\alpha_1, \alpha_2) = 0$.

Każdy ciąg wierzchołków grafu, w którym każde dwa sąsiednie wierzchołki są połączone krawędzią, nazywamy *drogą*. Drogi będziemy oznaczali literami $\omega, \omega' \dots$

Intuicyjnie, jeśli istnieje rozwiązanie S układu równań, to zdefiniowany powyżej graf zawiera informacje o tym, czy dla danej zmiennej α term $S(\alpha)$ po wykonaniu otaczających α substytucji jest podtermem jakiegoś innego $S(\beta)$ z wykonanymi substytucjami. Oprócz tego zbierane są tutaj informacje, które pary $S(\alpha), S(\beta)$ mają wspólne górne części (gdy termy zapiszemy jako drzewa).

Przedstawimy teraz właściwą procedurę upraszczającą układy równań. Ogólna idea związana z tą procedurą polega na unifikowaniu równań w sposób podobny do sposobu używanego w unifikacji Robinsona, przy czym wykonywana jest dodatkowa praca związana z przenoszeniem ewentualnych jawnych substytucji i kwantyfikatorów w głąb termów.

Definicja 3.2.46 (procedura upraszczania)

Procedura przetwarza krok za krokiem pary (\mathcal{Q}, S) , gdzie \mathcal{Q} jest ciągiem równań, które mają być rozwiązane, zaś S jest pewnym podstawieniem. Początkowe dane dla tej procedury stanowi para $(\mathcal{Q}_0, \emptyset)$, gdzie \mathcal{Q}_0 jest ciągiem równań z zadanego zbioru e-równań \mathcal{E} .

Intencją naszą jest, by po każdym kroku procedury spełniony był następujący niezmiennik:

jeśli S' jest rozwiązaniem \mathcal{Q} , to $S' \circ S$ jest rozwiązaniem \mathcal{Q}_0 .

Wykonujemy tutaj drobne nadużycie notacji, gdyż traktujemy ciąg \mathcal{Q} jak zbiór równań z tego ciągu.

Procedura kończy swoje działanie albo gdy w sposób jawny zostanie ogłoszona jej porażka, albo gdy ciąg \mathcal{Q} składa się wyłącznie z par jednej z trzech następujących postaci:

$$\forall_1^n \alpha \langle \vec{x} := \vec{t} \rangle \doteq \forall_2^n \alpha' \langle \vec{y} := \vec{s} \rangle, \quad (3.5)$$

$$\forall_1^n \alpha \langle \vec{x} := \vec{t} \rangle \doteq \forall_2^n P(s_1, \dots, s_m), \quad (3.6)$$

$$\forall_1^n P(t_1, \dots, t_n) \doteq \forall_2^n P(s_1, \dots, s_n), \quad (3.7)$$

gdzie \vec{x} , \vec{y} , \vec{t} , \vec{s} oznaczają odpowiednie wektory zmiennych lub termów, a $n \geq 0$.

Przy każdym kroku algorytmu wykonujemy operacje odpowiadające poszczególnym przypadkom (ominęliśmy sytuacje symetryczne ze względu na relację \doteq), po których wykonujemy normalizację termów ze względu na relację \rightsquigarrow^* . Oczywiście normalizację wykonujemy także przed pierwszym rozważaniem definiowanych tutaj przypadków.

1. Niech $\mathcal{Q} = [\forall_1^n(\sigma_1 \rightarrow \sigma_2) \doteq \forall_2^n(\tau_1 \rightarrow \tau_2)] \cdot \mathcal{Q}'$. Dane wejściowe przekształcane są na

$$([\forall_1^n \sigma_1 \doteq \forall_1^n \tau_1] \cdot [\forall_2^n \sigma_2 \doteq \forall_2^n \tau_2] \cdot \mathcal{Q}', S).$$

2. Niech $\mathcal{Q} = [\forall_1^n \alpha \langle \vec{x} := \vec{t} \rangle \doteq \forall_2^n(\tau_1 \rightarrow \tau_2)] \cdot \mathcal{Q}'$, gdzie \vec{x} jest zbiorem zmiennych $\{x_1, \dots, x_m\}$, wektor \vec{t} jest zbiorem termów $\{t_1, \dots, t_m\}$, zaś α jest zmienną typową, taką że żaden cykl zawierający α w grafie $G_{\mathcal{Q}}$ nie ma krawędzi z E_s . Dane wejściowe przekształcane są na

$$([\forall_1^n(\alpha_1 \rightarrow \alpha_2) \langle \vec{x} := \vec{t} \rangle \doteq \forall_2^n \tau_1 \rightarrow \tau_2] \cdot \mathcal{Q}'[\alpha := \alpha_1 \rightarrow \alpha_2]),$$

$$[\alpha := \alpha_1 \rightarrow \alpha_2] \circ S),$$

gdzie α_1, α_2 są świeżymi zmiennymi.

3. Let $\mathcal{Q} = [\forall_1^n \alpha \langle \vec{x} := \vec{t} \rangle \doteq \forall_2^n((\forall y : 0)\tau)] \cdot \mathcal{Q}'$, gdzie \vec{x} jest zbiorem zmiennych $\{x_1, \dots, x_m\}$, wektor \vec{t} jest zbiorem termów $\{t_1, \dots, t_m\}$, zaś α jest zmienną typową, dla której żaden cykl w grafie $G_{\mathcal{Q}}$ przechodzący przez α nie zawiera krawędzi z E_s . Dane wejściowe przekształcane są na

$$([\forall_1^n((\forall y' : 0)\alpha_1) \langle \vec{x} := \vec{t} \rangle \doteq \forall_2^n((\forall y : 0)\tau)] \cdot \mathcal{Q}'[\alpha := ((\forall y' : 0)\alpha_1)],$$

$$[\alpha := ((\forall y' : 0)\alpha_1)] \circ S),$$

gdzie α_1 jest świeżą zmienną typową, zaś y' jest świeżą zmienną pierwszego rzędu.

4. Niech $\mathcal{Q} = [\sigma \doteq \tau] \cdot \mathcal{Q}'$ i niech $\sigma \doteq \tau$ będzie jednej z trzech postaci (3.5–3.7). Dane wejściowe przekształcane są na

$$(\mathcal{Q}' \cdot [\sigma \doteq \tau], S).$$

5. We wszystkich pozostałych przypadkach zachodzi porażka.

Przy dowodzie terminacji potrzebna będzie nam definicja pewnych szczególnych ścieżek w grafie G_E :

Definicja 3.2.47 (droga termowa)

Powiemy, że droga ω w grafie G_E jest *termowa*, gdy

- każda krawędź łącząca wierzchołki ω występuje na tej drodze najwyżej raz oraz
- ω nie zawiera żadnego podciągu postaci $\alpha \cdot \alpha' \cdot \alpha$, gdzie (α, α') i (α', α) należą do E_e .

Lemat 3.2.48 (terminacja)

Procedura upraszczania z *definicji 3.2.46* zatrzymuje się dla *każdych danych wejściowych*.

Dowód:

Dla grafu G_Q można określić zbiór

$$A_k = \{\alpha \mid \text{nie ma krawędzi } (\alpha, \beta) \in E_s\}.$$

Dla danej zmiennej α określimy A_α jako zbiór dróg termowych z α do jakiejś zmiennej z A_k . Wreszcie dla danej drogi ω określimy w_ω jako sumę wag jej krawędzi.

Określimy teraz pewną funkcję W jako

$$W(\alpha) = \sum_{\omega \in A_\alpha} w_\omega.$$

Funkcja ta jest dobrze określona, gdyż dróg termowych w danym grafie jest skończenie wiele.

Określimy na ciągach równań funkcję μ jako

$$\mu(Q) = \langle n_t, n_Q, n_f \rangle \in \mathbb{N}^3,$$

przy czym

- $n_t = \sum_{\alpha \in \text{TV}(Q)} W(\alpha)$, gdy G_Q nie ma cyklu zawierającego krawędź ze zbioru E_s ; oraz $n_t = 0$ w przeciwnym przypadku;
- n_Q jest sumą rozmiarów wszystkich typów w Q mierzona liczbą wystąpień strzałek oraz symboli predykatów;
- n_f jest liczbą równań postaci (3.5–3.7) (def. 3.2.46), które znajdują się w ciągu Q przed jakimś równaniem, które nie jest żadnej z postaci (3.5–3.7).

Pokażemy, za pomocą rutynowej analizy przypadków, że wartości funkcji μ po wykonaniu dowolnego z kroków (1–4) maleje w porządku leksykograficznym na \mathbb{N}^3 .

Dla przypadku (1) wielkość n_t się nie zmienia, bo dla niej mają znaczenie tylko ścieżki, które są przedłużeniami ścieżek prowadzących do zmiennych, a przy wykonywanym tutaj przekształceniu takie ścieżki się nie zmieniają. Zmniejsza się za to ostro wielkość n_Q , gdyż zmniejsza się liczba strzałek w Q .

Dla przypadku (2) zmniejsza się wartość n_t , gdyż wszystkie ścieżki, które do tej pory zaczynały się od α zostają rozdzielone na ścieżki zaczynające się od α_1 i α_2 przy czym waga każdej takiej ścieżki jest mniejsza o 1. Dodatkowo wszystkie w_ω dla ścieżek ω w G_Q zaczynających się od jakiejś zmiennej $\alpha_0 \neq \alpha$ i przebiegających przez α nie zmieniają się, gdyż jedność, która do tej pory znajdowała się we fragmencie ścieżki zaczynającym się od α przechodzi do krawędzi przed α_1 lub α_2 , gdyż w stosownym typie α zostało zastąpione przez $\alpha_1 \rightarrow \alpha_2$.

Dla przypadku (3) dowód przebiega podobnie do poprzedniego przypadku.

Dla przypadku (4) ze zmiennymi i ścieżkami się nic nie zmienia, więc wartości n_t i n_Q nie ulegają zmianie. Ponieważ procedura się nie skończyła, to w Q musi się znajdować równanie, które nie jest żadnej z postaci (3.5–3.7). Zatem przeniesienie jednego równania takiej postaci na koniec Q zmniejsza liczbę tych równań postaci (3.5–3.7), które występują przed jakimś równaniem nie mającym takiej postaci. Stąd n_f się zmniejsza.

W związku z tym, że \rightsquigarrow -redukcja powoduje tylko zmianę położenia substytucji w e-typach, a n_t, n_Q oraz n_f nie zależą od ich umiejscowienia w e-typach, to wykonywanie normalizacji po każdym kroku algorytmu nie zmienia wartości funkcji μ . \square

Lemat 3.2.49 (poprawność jednego kroku)

Niech (Q, S) będzie parą, taką że Q jest ciągiem równań, a S jest podstawieniem. Jeśli istnieje podstawienie T , takie że $T \circ S$ rozwiązuje Q i jeden krok procedury upraszczania daje w wyniku (Q', S') , to istnienie podstawienia T' , takie że $T' \circ S'$ rozwiązuje Q' .

Dowód:

Podstawienie S zmienia się tylko w przypadkach (2) i (3) *procedury upraszczania*. Co więcej w przypadku (1) następuje tylko rozbitcie dwóch strzałek, co oczywiście nie zmienia faktu, że S rozwiązuje układ równań; w przypadku (4) zaś nie sam układ nie zmienia się w ogóle, co też nie zmienia rozwiązywalności układu przez S .

W przypadku (2) możemy założyć, że $T(\alpha) = \tau_1 \rightarrow \tau_2$, gdyż pierwsze równanie w Q wymusza taką postać. Określamy $T'(\beta) = T(\beta)$ dla $\beta \neq \alpha, \alpha_1, \alpha_2$ oraz $T'(\alpha_1) = 1(T(\alpha)), T'(\alpha_2) = 2(T(\alpha))$. Tak określone T' spełnia naszą tezę — dowód pozostawiamy Czytelnikowi.

Dowód dla (3) jest podobny do poprzedniego przypadku — szczegóły pozostawiamy Czytelnikowi. \square

Lemat 3.2.50 (rozwiązanie daje krok algorytmu)

Niech (Q, S) będzie parą, gdzie Q jest ciągiem równań, a S podstawieniem. Jeśli istnieje podstawienie T takie, że $T \circ S$ rozwiązuje Q , to procedura upraszczania nie ponosi porażki.

Dowód:

Przypadki, kiedy żaden z punktów (1–4) nie może zostać zastosowany obejmują tylko:

1. $Q = [\forall_1^n((\forall x:0)\sigma) \doteq \forall_2^n \tau] \cdot Q'$, gdzie τ nie zaczyna się od $(\forall x:0)$ i τ nie jest postaci $\alpha\langle \vec{x} := \vec{t} \rangle$;
2. $Q = [\forall_1^n(\sigma_1 \rightarrow \sigma_2) \doteq \forall_2^n \tau] \cdot Q'$, gdzie τ nie ma postaci $\tau_1 \rightarrow \tau_2$ oraz nie jest równe $\alpha\langle \vec{x} := \vec{t} \rangle$;
3. $Q = [\forall_1^{n_1} P(t_1, \dots, t_{m_1}) \doteq \forall_2^{n_2} P'(s_1, \dots, s_{m_2})] \cdot Q'$, gdzie

$$\forall_1^{n_1} P(t_1, \dots, t_{m_1}) \not\equiv \forall_2^{n_2} P'(s_1, \dots, s_{m_2}).$$

4. $Q = [\forall_1^{n_1} \alpha\langle \vec{x} := \vec{t} \rangle \doteq \forall_2^{n_2} \tau] \cdot Q'$, gdzie τ jest albo $\sigma_1 \rightarrow \sigma_2$, albo $(\forall x:0)\sigma$, zaś G_Q zawiera cykl z α przechodzący przez krawędź z E_s .

W przypadkach (1–3), Q w sposób oczywisty nie ma rozwiązania — konflikt symboli.

Dowód w przypadku (4) wykonujemy przez indukcję ze względu na długość cyklu.

Jeśli cykl ma długość zero (krawędź od α do α), to istnieje równanie $[\forall^n \tau_1 \doteq \forall^n \tau_2]$, w którym τ_1 zawiera α na pewnej ścieżce ρ , a τ_2 zawiera zmienną α na pewnej ścieżce $\rho' \cdot \rho$, gdzie $|\rho'| > 0$. Takie równanie jest nierozwiązywalne, gdyż jego rozwiązanie wymagałoby istnienia nieskończonej ścieżki w rozwiązaniu.

Jeśli długość jest większa od zera, to w cyklu mamy równanie postaci $[\forall^{n_1} \tau_1 \doteq \forall^{n_2} \tau_2]$ z pewnym α_1 w τ_1 na końcu ścieżki ρ oraz pewnym α_2 w τ_2 na ścieżce $\rho' \cdot \rho$. Usuwamy α_1 podstawiając na jego miejsce term znajdujący się na ρ w τ_2 (być może dokonując jednocześnie przemianowania odpowiednich zmiennych pierwszego rzędu). W ten sposób eliminujemy jedną zmienną z cyklu i uzyskujemy zbiór równań, którego rozwiązywalność jest równoważna rozwiązywalności pierwotnego układu. Z założenia indukcyjnego uzyskany zbiór równań nie ma rozwiązania, a zatem i pierwotny zestaw równań także go nie ma. \square

Lemat 3.2.51 (pełność pojedynczego kroku)

Niech (Q', S') będzie parą uzyskaną z (Q, S) w jednym kroku procedury upraszczania. Jeśli istnieje podstawienie T' takie, że $T' \circ S'$ rozwiązuje Q' , to istnieje podstawienie T , takie że $T \circ S$ rozwiązuje Q .

Dowód:

Podstawienie T' nie musi być zmieniane dla przypadków różnych od (2) i (3). W przypadku (2), przyjmujemy $T(\alpha) = T'(\alpha_1) \rightarrow T'(\alpha_2)$, zaś w przypadku (3): $T(\alpha) = (\forall y' : 0)T'(\alpha_1)$.

Jest kwestią rutynowego sprawdzenia, że tak określone T spełnia naszą tezę. Szczegóły zostawiamy Czytelnikowi. \square

Fakt 3.2.52 (zmienne w podstawieniu i ciągu równań)

Niech (Q, S) będzie parą, w której $\text{TV}(Q) \cap \text{Dom}(S) = \emptyset$. Jeśli (Q', S') zostało osiągnięte w n -tym kroku procedury upraszczania, to również $\text{TV}(Q') \cap \text{Dom}(S') = \emptyset$.

Dowód:

Rutynowa indukcja ze względu na liczbę kroków algorytmu. \square

Twierdzenie 3.2.53 (poprawność procedury upraszczania)

1. Procedura zatrzymuje się dla wszystkich danych wejściowych postaci (Q, \emptyset) .
2. Układ Q ma rozwiązanie wtedy i tylko wtedy, gdy wynik *procedury upraszczania* zastosowanej do (Q, \emptyset) ma postać (Q', S) , gdzie Q' ma rozwiązanie i każde równanie Q' ma którąś z postaci (3.5–3.7) opisanych w definicji 3.2.46.

Dowód:

Lemat 3.2.48 implikuje terminację *procedury upraszczania*, co daje nam pierwszą część naszego twierdzenia i możemy zakładać w dalszym ciągu dowodu, że procedura zatrzymuje się po n krokach.

Niech S_0 będzie rozwiązaniem Q . Podstawienie $S_0 \circ \emptyset$ jest również rozwiązaniem Q . Możemy teraz n razy zastosować lemat 3.2.49 oraz lemat 3.2.50 i w wyniku uzyskać parę (Q', S) , dla której istnieje podstawienie T , takie że $T \circ S$ jest rozwiązaniem Q' .

Analogicznie, jeśli *procedura upraszczania* kończy działanie i daje wynik (Q', S) , taki że Q' ma rozwiązanie T , to ponieważ $\text{TV}(Q') \cap \text{Dom}(S) = \emptyset$ (fakt 3.2.52), mamy, że $T \circ S$ jest rozwiązaniem Q' . Stąd możemy tutaj zastosować n razy lemat 3.2.51 i uzyskać, że Q ma również rozwiązanie. \square

Usuwanie kwantyfikatorów

W tym momencie za pomocą *procedury upraszczania* uzyskaliśmy zbiór równań o specjalnej postaci. Jednak równania te wciąż jeszcze mają w sobie kwantyfikatory, które nie pozwalają na bezpośrednią translację do unifikacji drugiego rzędu. Opiszemy tutaj procedurę usuwania tych kwantyfikatorów. Procedura ta wykorzystuje pewną pośrednią grafową strukturę danych.

Definicja 3.2.54 (graf ustalania)

Graf ustalania dla zbioru równań \mathcal{E} określamy jako dowolny graf o wierzchołkach

$$V_{\mathcal{E}} = X_{\mathcal{E}} \times \text{TV}(\mathcal{E}),$$

gdzie $X_{\mathcal{E}}$ jest zbiorem zmiennych pierwszego rzędu, które znajdują się pod kwantyfikatorami w \mathcal{E} , zaś $\text{TV}(\mathcal{E})$ to zbiór zmiennych typowych w \mathcal{E} .

Powyższy graf jest grafem nieskierowanym (krawędzie są parami nieuporządkowanymi).

Intuicyjnie, procedurę usuwania kwantyfikatorów pierwszego rzędu będziemy uruchamiać z grafem ustalania, w którym krawędź między (x, α) a (x', α') pojawia się, gdy istnieje w układzie równań równanie, w którym występuje zmienna α i α' , a x oraz x' są kwantyfikowane na tej samej pozycji. Następnie będziemy przetwarzać grafy ustalania dochodząc stopniowo do sytuacji, gdzie każda krawędź między (x, α) a (x', α') oznacza, że x i x' powinny występować dokładnie w tych samych miejscach termów podstawianych na α i na α' odpowiednio.

Procedura usuwania kwantyfikatorów daje w wyniku nowy zbiór równań — tym razem bez kwantyfikatorów — oraz pewne dodatkowe *więzy negatywne*. Więzy te, intuicyjnie, mówią, że pewien symbol nie może wystąpić w typie odpowiadającym danej zmiennej typowej. Te więzy wprowadzane są w wyniku pojawiania się pewnych nowych stałych pierwszego rzędu. Stałe, które znajdowały się w oryginalnej sygnaturze, nie występują w wygenerowanych więzach.

Definicja 3.2.55 (e-równania z więzami)

O parze $\langle \mathcal{E}, \phi \rangle$, gdzie \mathcal{E} jest zbiorem e-równań, a $\phi : \text{TV}(\mathcal{E}) \rightarrow \text{Const}$ mówimy, że jest *układem równań z więzami negatywnymi* ϕ . Mówimy, że $S : \text{TV}(\mathcal{E}) \rightarrow \mathcal{T}_{\Sigma}^e(X, \mathcal{X})$ jest rozwiązaniem takiego równania z więzami, gdy dla każdego $\alpha \in \text{TV}(\mathcal{E})$ stałe z $\phi(\alpha)$ nie występują w $S(\alpha)$.

Oczywiście zakładamy, że w przedstawionej powyżej procedurze usuwania kwantyfikatorów *nie* wykonywane są żadne α -konwersje.

Procedura usuwania kwantyfikatorów składa się z dwóch części: tworzenia grafu ustalania dla wejściowego równania oraz właściwego usuwania kwantyfikatorów.

Definicja 3.2.56 (tworzenie grafu ustalania)

Wejście dla tej procedury stanowią trójki (Σ, X, \mathcal{E}) , gdzie Σ jest sygnaturą, X jest zbiorem zmiennych pierwszego rzędu, zaś \mathcal{E} jest zbiorem e-równań zawartym w $\mathcal{E}_{\Sigma}^e(X, \mathcal{X})$. Wynikiem jest graf ustalania $G_{\mathcal{E}}$ oraz funkcja określająca więzy negatywne $\phi_{\mathcal{E}} : \text{TV}(\mathcal{E}) \rightarrow P(X_{\mathcal{E}})$. Obiekty te definiujemy jako $G_{\mathcal{E}}^n = \langle V_{\mathcal{E}}, E_n \rangle$ oraz ϕ_n uzyskane w ostatnim kroku następującej iteracji:

1. Zaczynamy od $G_{\mathcal{E}}^0 = \langle V_{\mathcal{E}}, E_0 \rangle$ i $\phi_0 : \text{TV}(\mathcal{E}) \rightarrow P(X_{\mathcal{E}})$, gdzie

$$E_0 = \{((x_i, \alpha), (x'_i, \alpha')) \mid [\forall x_1 \dots \forall x_i \dots \forall x_n \alpha \langle \dots \rangle \doteq \forall x'_1 \dots \forall x'_i \dots \forall x'_n \alpha' \langle \dots \rangle] \in \mathcal{E}\},$$

$$\phi_0(\alpha) = \emptyset \text{ dla każdego } \alpha.$$

2. Przekształcamy $G_{\mathcal{E}}^n$ w $G_{\mathcal{E}}^{n+1}$ pod warunkiem, że istnieje droga ω w $G_{\mathcal{E}}^n$ od (x_1, α) do (x_2, α) , gdzie $x_1 \neq x_2$. Definiujemy E_{n+1} i ϕ_{n+1} w sposób następujący

(a) weź krawędź $((y, \beta), (y', \beta'))$ między sąsiednimi wierzchołkami ω ,

(b) usuń ją — wynikiem jest zbiór E_{n+1} ,

(c) $\phi_{n+1}(\gamma) = \phi_n(\gamma)$ dla $\gamma \neq \beta$,
 $\phi_{n+1}(\beta) = \phi_n(\beta) \cup \{y\}$, gdy w \mathcal{E} znajduje się równanie

$$\forall z_1 \dots \forall z_i \dots \forall z_m \beta \langle \bar{z}^1 := \bar{t} \rangle \doteq \forall z'_1 \dots \forall z'_i \dots \forall z'_m \beta' \langle \dots \rangle,$$

takie że $z_i = y$ i $z'_i = y'$ oraz $y \notin \bar{z}^1$,

$\phi_{n+1}(\beta) = \phi_n(\beta)$, gdy takie równanie nie istnieje w \mathcal{E} .

Następujący fakt jest konieczny dla ustalenia poprawności procedury usuwania kwantyfikatorów.

Fakt 3.2.57 (różne składowe)

Niech $\forall z_1 \dots \forall z_i \dots \forall z_m \alpha \langle \bar{z}^1 := \bar{t}^1 \rangle \doteq \forall z'_1 \dots \forall z'_i \dots \forall z'_m \beta \langle \bar{z}^2 := \bar{t}^2 \rangle$ będzie równaniem należącym do \mathcal{E} . Jeśli $G_{\mathcal{E}}^n$ nie zawiera krawędzi $((z_i, \alpha), (z'_i, \beta))$, to albo $z_i \in \phi_n(\alpha)$, albo $z'_i \in \phi_n(\beta)$.

Dowód:

Indukcja ze względu na n — zostawiamy Czytelnikowi. Kluczową rolę gra tutaj punkt (2c) definicji 3.2.56. \square

Następujący fakt wyjaśnia, dlaczego w punkcie (2) definicji 3.2.56 powinniśmy przerywać drogi między (x_1, α) a (x_2, α) . Istnienie takiej drogi interpretujemy dalej w definicji 3.2.59 jako konieczność przyjęcia na x_1 i x_2 tej samej stałej. Tymczasem takie ustalenie w przypadkach wyłapywanych w punkcie (2) prowadziłyby do sprzeczności.

Fakt 3.2.58 (kopiowanie)

Niech ω będzie drogą w $G_{\mathcal{E}}^n$ dla pewnego n i niech S będzie rozwiązaniem \mathcal{E} . Jeśli dla każdej krawędzi $((z_i, \alpha), (z'_i, \beta))$, gdzie (z_i, α) jest k . wierzchołkiem drogi ω , a (z'_i, β) jej wierzchołkiem $k+1$., i dla każdego równania postaci

$$\forall z_1 \dots \forall z_i \dots \forall z_n \alpha \langle \bar{z}^1 := \bar{t}^1 \rangle \doteq \forall z'_1 \dots \forall z'_i \dots \forall z'_n \beta \langle \dots \rangle$$

mamy, że $z_i \notin \bar{z}^1$, to jeśli dla pewnego wierzchołka (y, γ) drogi ω istnieje taka ścieżka ρ , że $\rho(S(\gamma)) = y$, to dla dla każdego wierzchołka (y', γ') tej drogi $\rho(S(\gamma')) = y'$.

Dowód:

Indukcja ze względu na długość drogi ω — zostawiamy ją Czytelnikowi. \square

Definicja 3.2.59 (właściwe usuwanie kwantyfikatorów)

Wejście dla tej procedury stanowią piątki $(\Sigma, X, \mathcal{E}, G_{\mathcal{E}}, \phi_{\mathcal{E}})$, gdzie Σ jest sygnaturą, X jest zbiorem zmiennych pierwszego rzędu, \mathcal{E} jest zbiorem e-równań zawartym w $\mathcal{E}_{\Sigma}^e(X, \mathcal{X})$, a $G_{\mathcal{E}}, \phi_{\mathcal{E}}$ są wynikiem tworzenia grafu ustalania. Wynikiem jest trójka $(\Sigma', \mathcal{E}', \phi)$, gdzie Σ' jest sygnaturą, \mathcal{E}' jest zbiorem e-równań zawartym w $\mathcal{E}_{\Sigma'}^e(\emptyset, \mathcal{X})$, zaś $\phi : \text{TV}(\mathcal{E}') \rightarrow P(X_{\mathcal{E}})$.

Korzystając z $G_{\mathcal{E}}$ i $\phi_{\mathcal{E}}$, tworzymy nowy zbiór równań \mathcal{E}' , wykonując następujące dwa kroki:

1. dla każdej spójnej składowej H w $G_{\mathcal{E}}$ wybieramy nową stałą a_H ;
2. każde równanie:

$$\forall z_1 \dots \forall z_n \alpha \langle \bar{z}^1 := \bar{t}^1 \rangle \doteq \forall z'_1 \dots \forall z'_n \beta \langle \bar{z}^2 := \bar{t}^2 \rangle$$

zastępujemy równaniem

$$\alpha \langle \bar{z}^1 := \bar{t}^1 \rangle [z_1 := a_1, \dots, z_n := a_n] \doteq \beta \langle \bar{z}^2 := \bar{t}^2 \rangle [z'_1 := a_1, \dots, z'_n := a_n]$$

przy czym

- $a_i = a_H$, jeśli (z_i, α) oraz (z'_i, β) są w tej samej spójnej składowej H ;
- $a_i = a_H$, jeśli (z_i, α) oraz (z'_i, β) są w różnych spójnych składowych, (z'_i, β) należy do spójnej składowej H i $z_i \in \phi(\alpha)$;
- $a_i = a_H$, jeśli (z_i, α) oraz (z'_i, β) są w różnych spójnych składowych, (z_i, α) należy do spójnej składowej H oraz $z_i \notin \phi(\alpha)$ i $z'_i \in \phi(\beta)$

(fakt 3.2.57 implikuje, że to już wszystkie przypadki) oraz

- $\{z_{i_1}, \dots, z_{i_m}\} = \{z_1, \dots, z_n\} \setminus \bar{z}^1$;
- $\{z'_{j_1}, \dots, z'_{j_m}\} = \{z'_1, \dots, z'_n\} \setminus \bar{z}^2$.

Sygnatura, którą otrzymujemy w wyniku działania procedury zawiera:

1. wszystkie symbole Σ ,
2. wszystkie symbole X ,
3. wszystkie stałe pierwszego rzędu wprowadzane w powyższej procedurze.

Fakt 3.2.60 (poprawność usuwania kwantyfikatorów)

1. Procedura usuwania kwantyfikatorów zatrzymuje się,
2. Układ \mathcal{E} ma rozwiązanie $T : \mathcal{X} \rightarrow \mathcal{T}_{\Sigma}^e(X, \emptyset)$ wtedy i tylko wtedy, gdy pewien wynik \mathcal{E}' procedury usuwania kwantyfikatorów ma rozwiązanie $T' : \mathcal{X} \rightarrow \mathcal{T}_{\Sigma}^e(X \setminus (\Sigma \cup \text{Dom}(\Gamma)), \emptyset)$, które spełnia więzy ϕ .
Co więcej, żadna stała z Σ nie występuje w ϕ .

Dowód:

Własność (1) jest oczywista. Dowód (2) składa się z dwóch części.

(\Rightarrow) Jeśli \mathcal{E} ma rozwiązanie T , to możemy skonstruować rozwiązanie T' układu \mathcal{E}' kładąc:

$$\begin{aligned} T'(\alpha) &= T(\alpha)[z_{i_1} := a_{i_1}, \dots, z_{i_m} := a_{i_m}] \\ T'(\beta) &= T(\beta)[z'_{j_1} := a_{j_1}, \dots, z'_{j_{m'}} := a_{j_{m'}}], \end{aligned}$$

gdzie $\alpha, \beta, z_{i_1}, \dots, z_{i_m}, z'_{j_1}, \dots, z'_{j_{m'}}, a_{i_1}, \dots, a_{i_m}, a_{j_1}, \dots, a_{j_{m'}}$ są określone jak w punkcie 2 definicji 3.2.59. Podstawienie T' jest rozwiązaniem, bo:

- w nowym układzie równań i rozwiązaniu nie zmieniają się wystąpienia symboli funkcyjnych;
- podobnie nie zmieniają się pozycje stałych z sygnatury Σ ;
- nowo wprowadzane stałe wprowadzane są, tak że na i -tą zmienną w ciągu kwantyfikatorów po lewej i prawej stronie trafia ta sama stała.

Tak określone rozwiązanie T' spełnia więzy ϕ , co zapewnia fakt 3.2.57. Szczegóły pozostawiamy Czytelnikowi.

(\Leftarrow) Jeśli \mathcal{E}' ma rozwiązanie T' , to możemy odtworzyć T rozwiązujące \mathcal{E} , zastępując po prostu świeże stałe przez zmienne pierwszego rzędu, które one zastępowały. Przy takim postępowaniu mogą się pojawić pewne zmienne poza zasięgiem kwantyfikatorów. Takie wystąpienia zastępujemy jedną wyróżnioną stałą z wyjściowej sygnatury. Istnienie ω w punkcie (2) definicji 3.2.56 gwarantuje, że tylko dla ustalonej zmiennej typowej α nowo wprowadzonej stałej znajdującej się w $T'(\alpha)$ odpowiada jedna zmienna pierwszego rzędu. \square

W wyniku zastosowanej tutaj procedury dostajemy równania o jeszcze bardziej ograniczonej postaci:

Definicja 3.2.61 (postać równań bez kwantyfikatorów)

Równania, które są wynikiem procedury *usuwania kwantyfikatorów* mają postać:

$$\alpha \langle \vec{x} := \vec{t} \rangle \doteq \alpha' \langle \vec{y} := \vec{s} \rangle, \quad (3.8)$$

$$\alpha \langle \vec{x} := \vec{t} \rangle \doteq P(s_1, \dots, s_m), \quad (3.9)$$

$$P(t_1, \dots, t_n) \doteq P(s_1, \dots, s_n), \quad (3.10)$$

gdzie \vec{x} , \vec{y} , \vec{t} , \vec{s} oznaczają odpowiednie wektory zmiennych lub termów, a $n \geq 0$.

Od e-równań do równań drugiego rzędu

Uzyskaliśmy w końcu zbiór e-równań, który może zostać przekształcony na zbiór równań drugiego rzędu specjalnej postaci.

Definicja 3.2.62 (unifikacja z więzami)

O parze $\langle E, \phi \rangle$, gdzie E jest zbiorem równań unifikacji drugiego rzędu, a $\phi : FV(E) \rightarrow \text{Const}$ mówimy, że jest *układem równań drugiego rzędu z więzami negatywnymi* ϕ .

Mówimy, że $S : FV(\mathcal{E}) \rightarrow \text{Termy}_{\text{Ch}}(\lambda \rightarrow n)$ jest rozwiązaniem takiego równania z więzami, gdy dla każdego $F \in FV(E)$ mamy, że stałe z $\phi(F)$ nie występują w $S(F)$.

Wprowadzimy teraz pewną relację równoważności \equiv pomiędzy typami występującymi w równaniach.

Definicja 3.2.63 (relacja \equiv)

Dla układu E określamy

$$\tau_1 \equiv \tau_2 \quad \text{wtedy i tylko wtedy, gdy} \quad \begin{array}{l} \text{istnieje w } E \text{ równanie } \tau_1 \doteq \tau_2 \\ \text{lub } \tau_1 = \alpha \langle \vec{x} := \vec{t} \rangle \text{ i } \tau_2 = \alpha \langle \vec{y} := \\ \vec{s} \rangle. \end{array}$$

Translacja do unifikacji drugiego rzędu z więzami definiujemy jak następuje:

Definicja 3.2.64 (translacja do drugiego rzędu)

Niech będzie dany układ e-równań z więzami negatywnymi $\langle \mathcal{E}, \phi \rangle$, przy czym postać tych równań jest zgodna z definicją 3.2.61. Zdefiniujemy układ równań drugiego rzędu $\langle E, \phi' \rangle$. Translacja składa się z dwóch kroków

1. Najpierw usuwamy równania $\tau \doteq \tau'$ o tej własności, że klasa abstrakcji relacji \equiv wyznaczona przez τ (i τ') składa się wyłącznie z typów zaczynających się od zmiennej typowej.
2. Dla każdej zmiennej typowej α niech A_α będzie zbiorem wszystkich zmiennych pierwszego rzędu x , które występują w jakimś równaniu z \mathcal{E} w kontekście $\alpha \langle \vec{y} := \vec{t} \rangle \langle x := s \rangle \langle \vec{z} := \vec{u} \rangle$. Niech $A_\alpha = \{x_1, \dots, x_n\}$. Dla każdego $\alpha \langle \vec{y} := \vec{t} \rangle$, zakładając, że $\vec{t} = \{t_1, \dots, t_m\}$, zastępujemy $\alpha \langle \vec{y} := \vec{t} \rangle$ przez $F_\alpha(t'_1, \dots, t'_n)$, gdzie

- $t'_i = t_j[y_{j+1} := t_{j+1}] \cdots [y_m := t_m]$, o ile $x_i = y_j$ oraz
- $t'_i = x_i$, o ile wśród zmiennych \vec{y} nie ma $y_j = x_i$.

Funkcję ϕ' określamy zaś jako $\phi'(F_\alpha) = \phi(\alpha)$.

Natychmiast uzyskujemy następujący fakt:

Fakt 3.2.65 (uproszczone równania i unifikacja drugiego rzędu)

Dla danego zbioru \mathcal{E} równań postaci (3.8–3.10) z definicji 3.2.61 wraz z więzami ϕ istnieje zbiór E równań unifikacji drugiego rzędu oraz więzy ϕ' , takie że \mathcal{E} jest rozwiązywalne z więzami ϕ wtedy i tylko wtedy, gdy E jest rozwiązywalne z więzami ϕ' .

Co więcej, translacja $\langle \mathcal{E}, \phi \rangle$ na $\langle \mathcal{E}', \phi' \rangle$ jest efektywna, a zbiór stałych w ϕ nie zmienia się przy przejściu do ϕ' .

Dowód:

Translacja z definicji 3.2.64 jest bez wątplenia efektywna i nie zmienia zbioru stałych w więzach.

Pierwszy krok translacji, usunięcie pewnych klas abstrakcji relacji \equiv , nie wpływa na rozwiązywalność, gdyż określony w ten sposób podzbiór równań ma zawsze rozwiązanie i w dodatku rozwiązanie to jest niezależne od reszty początkowego układu równań.

Równoważność zachodzi, gdyż drugi krok przedstawionej tutaj translacji sprowadza się do przemianowania zmiennych α na F_α oraz ustalenia odpowiedniości między zmiennymi z lewych stron znaku $:=$ w jawnych substytucjach a argumentami zmiennej drugiego rzędu. Odpowiedniość ta jest wyznaczona przez arbitralne określenie z góry kolejności tych zmiennych. To wystarczy, gdyż na daną zmienną α wszędzie podstawiany jest ten sam e-typ, więc dana zmienna pierwszego rzędu x może być zastąpiona po prostu przez ustalony (dla wszystkich wystąpień α taki sam) argument. \square

Zauważmy, że uzyskujemy tutaj równania ze zmiennymi w pozycjach czołowych (zob. definicja 1.4.6).

Musimy jeszcze poradzić sobie z wprowadzonymi więzami. Transformacja pozwalająca pozbyć się więzów wygląda w sposób następujący:

Definicja 3.2.66 (usuwanie więzów)

Niech E będzie zbiorem równań drugiego rzędu z więzami ϕ . Dla każdej stałej $c \in \Sigma$ wprowadzamy dwie nowe stałe c_1 oraz c_2 . Dla każdej zmiennej drugiego rzędu F o arności n wprowadzamy dwie zmienne F_1 oraz F_2 obie o arności $n+k$, gdzie k jest liczbą stałych w Σ . Określamy dwie operacje $|\cdot|_1$ oraz $|\cdot|_2$ jak następuje:

- $|c|_i = c_i$,
- $|f(t_1, \dots, t_n)|_i = f(|t_1|_i, \dots, |t_n|_i)$,
- $|F(t_1, \dots, t_n)|_i = F_i(|t_1|_i, \dots, |t_n|_i, c_i^1, \dots, c_i^k)$, gdzie $\{c^1, \dots, c^k\}$ jest zbiorem wszystkich stałych w Σ .

Każde równanie $t_1 \doteq t_2$ w E jest zastępowane przez parę równań

$$\begin{aligned} |t_1|_1 &\doteq |t_2|_1 \\ \text{oraz} & \\ |t_1|_2 &\doteq |t_2|_2. \end{aligned} \tag{3.11}$$

Dla każdej zmiennej F w E dokładamy jeszcze równania

$$\begin{aligned} F_1(a_1, \dots, a_k, c_1^1, \dots, c_1^k) &\doteq F_2(a_1, \dots, a_k, c_1^1, \dots, c_1^k) \\ \text{oraz} & \\ F_1(a_1, \dots, a_k, c_2^1, \dots, c_2^k) &\doteq F_2(a_1, \dots, a_k, c_2^1, \dots, c_2^k), \end{aligned} \quad (3.12)$$

gdzie a_1, \dots, a_k są świeżymi stałymi. W końcu dla każdego więzu $\phi(F)$ dodajemy równanie

$$F_1(a_1, \dots, a_1, c_1^1, \dots, c_1^k) \doteq F_2(a_1, \dots, a_1, d^1, \dots, d^k), \quad (3.13)$$

gdzie $d^i = c_1^i$, jeśli $c^i \notin \phi(F)$ oraz $d^i = c_2^i$, jeśli $c^i \in \phi(F)$.

Natychmiast uzyskujemy następujący fakt:

Fakt 3.2.67 (unifikacja drugiego rzędu bez więzów)

Niech a_0 będzie stałą, która nie pojawia się w więzach ϕ . Dla danego zbioru E równań z więzami ϕ istnieje zbiór E' równań unifikacji drugiego rzędu, taki że E jest rozwiązywalne z więzami ϕ wtedy i tylko wtedy, gdy E' jest rozwiązywalne.

Translacja przekształcająca $\langle E, \phi \rangle$ na E' jest efektywna i daje w wyniku tylko równania postaci (1.1–1.3) z definicji 1.4.6.

Dowód:

(\Rightarrow) Przypuśćmy, że E jest rozwiązywalne przez S spełniające więzy ϕ . Konstruujemy podstawienie S' , które jest rozwiązaniem E' , kładąc

$$S'(F_i) = \lambda x_1 \dots x_k x_{k+1} \dots x_{k+m} \cdot |S(F)x_1 \dots x_k|_0$$

dla $i = 1, 2$, gdzie $|\cdot|_0$ jest określone jako operacja, która zastępuje każdą stałą c^j (oznaczenie jak w definicji 3.2.66) przez x_{j+k} , przy czym k jest arnością F , a m liczbą stałych w sygnaturze. Tak określone S' jest rozwiązaniem, gdyż

- w równaniach postaci 3.11 zachodzi dokładne naśladowanie równań z E tyle, że stałe z sygnatury są zastąpione przez odpowiednie argumenty od $k+1$ do $k+m$; każda stała jest potem w równaniu wstawiana na odpowiedni argument, więc w wyniku podstawienia trafia dokładnie w to samo miejsce, w którym była w rozwiązaniu dla E ;
- w równaniach postaci 3.12 zachodzi równość, bo F_1 i F_2 są zdefiniowane dokładnie tak samo;
- w równaniach postaci 3.13 zachodzi równość, bo dla stałych, które mogą wystąpić w F odpowiednie argumenty są dla F_1 i F_2 równe.

(\Leftarrow) Przypuśćmy, że E' jest rozwiązywalne za pomocą pewnego podstawienia S' . Określamy podstawienie S jako $S(F) = |S(F_1)|_{-1}$, gdzie $|\cdot|_{-1}$ działa w ten sposób, że $j + k$ -ty argument $S(F_1)$, gdzie k jest arnością F , jest zastępowany przez c^j (notacja jak w definicji 3.2.66), zaś każde wystąpienie stałej c_m^l jest zastępowane przez stałą a_0 . Tak określone podstawienie jest rozwiązaniem E , gdyż S' unifikuje równania 3.11.

Rozwiązanie S spełnia dodatkowo więzy ϕ . Wynika to z tego, że rozwiązanie S' musiało unifikować równanie postaci 3.13. W związku z tym, że w takim równaniu w argumentach odpowiadających stałym z sygnatury (argumenty $k + 1 - k + m$, gdzie m jest liczbą stałych w sygnaturze) dla stałych z $\phi(F)$ po lewej i po prawej stronie równania mamy różne symbole, a to oznacza, że stosowny argument $S'(F_1) = \lambda x_1 \dots x_{k+m} \cdot M$ nie może występować w M . \square

Kontekstowe wyprowadzanie typów w logice pierwszego rzędu

Możemy teraz wykonać przez nas w tym rozdziale pracę podsumować twierdzeniem:

Twierdzenie 3.2.68 (rozstrzygalność kontekstowego wyprowadzania typów w logice pierwszego rzędu)

Problem kontekstowego wyprowadzania typów w logice pierwszego rzędu jest rozstrzygalny.

Dowód:

Twierdzenia 3.2.44 i 3.2.53 oraz fakty 3.2.60 i 3.2.67 dają redukcję problemu kontekstowego wyprowadzania typów w logice pierwszego rzędu do problemu rozwiązywania równań ze zmiennymi w pozycjach czołowych. Ten ostatni problem zaś na mocy twierdzenia 2.2.1 jest rozstrzygalny. \square

Rozdział 4

Podsumowanie

W niniejszej pracy przedstawione zostały następujące nowe wyniki:

- W dziedzinie unifikacji wyższego rzędu:
 - Zamieszczony został istotnie nowy dowód nierozstrzygalności problemu unifikacji drugiego rzędu. Uzyskany dowód nie odwołuje się do 10. problemu Hilberta, co stanowi istotne koncepcyjne uproszczenie dowodu unifikacji 2. rzędu (rezultaty te znajdują się w podrozdziale 2.1).
 - Dowód ten jednocześnie obejmuje ciekawy podproblem unifikacji drugiego rzędu — problem unifikacji z prostymi instancjami, gdzie w argumentach niewiadomych mogą występować tylko termy bez niewiadomych.
 - Pokazane zostały ciekawe związki między unifikacją a przepisywaniem termów (sekcja 2.1.2).
 - Przedstawiony został ciekawy rozstrzygalny podproblem unifikacji drugiego rzędu — unifikację ze zmiennymi w pozycjach czołowych. W unifikacji tej jeśli w termie równania występuje niewiadoma, to jest ona pierwszym jego symbolem (podrozdział 2.2).
- W dziedzinie wyprowadzania typów
 - Przedstawiony został dowód nierozstrzygalności problemu wyprowadzania typów dla $\lambda 2$ -Church (podrozdział 3.1).
 - Pokazany został dowód rozstrzygalności problemu kontekstowego wyprowadzania typów dla zanurzenia logiki 1. rzędu w λP (podrozdział 3.2).

Powyższe w pełni oryginalne wyniki zostały jeszcze uzupełnione dla pełności przedstawienia o dowody oparte na konstrukcji Doweka z pracy [Dow93]. Mamy tutaj

- nierozstrzygalność unifikacji ze zmiennymi 3. rzędu w pozycjach czołowych (podrozdział 2.3) oraz
- nierozstrzygalność pewnej odmiany wyprowadzania typów dla zanurzenia logiki pierwszego rzędu w λP ; przy tego rodzaju wyprowadzaniu konieczne jest, aby w danych wejściowych podawane były kontekst i term logiki pierwszego rzędu, natomiast nie nakłada się warunku, aby całe wyprowadzenie było pierwszego rzędu — wystarczy, żeby mieściło się ono w λP (sekcja 3.2.2).

Dalsze badania Dalsze badania związane z zaprezentowanymi tutaj systemami powinny obejmować określenie złożoności problemu kontekstowego wyprowadzania typów dla naszego zanurzenia logiki pierwszego rzędu w λP . Również interesującym problemem jest pytanie, czy system ten ma własność typów głównych (czyli czy dla każdego termu istnieje typ, z którego można uzyskać dowolny inny typ przez zastosowanie odpowiedniego podstawienia). Przykład 1 wskazuje, że system ten nie ma własności typów głównych w tradycyjnym znaczeniu. Jednak być może da się uzyskać jakieś zadowalające uogólnienie pojęcia typu głównego tak, żeby system miał własność typów głównych takiego nowego rodzaju.

Podziękowania

Przede wszystkim chciałbym podziękować mojemu promotorowi prof. Pawłowi Urzyczynowi, z którego inspiracji zająłem się przedstawionymi tutaj zagadnieniami. Jego liczne uwagi i nieustrudzone przewodnictwo podczas tworzenia tego materiału są nieocenione. Podziękowania należą się także innym członkom Zakładu Logiki Stosowanej w Instytucie Informatyki ze szczególnym uwzględnieniem prof. Jerzego Tiuryna za jego nieustanne zainteresowanie postępami w mojej pracy oraz mgr. Grzegorza Grudzińskiego za mentalne i fizyczne wsparcie, jakie z jego strony wielokrotnie dostawałem podczas tworzenia niniejszego materiału. Podziękowania należą się także mojej rodzinie za nieprzerwane wspieranie mnie w dążeniu do osiągnięcia kolejnych stopni wykształcenia. W końcu, ale na pewno nie na szarym końcu, podziękowania należą się mojej przyjaciółce Joannie Gręzak, która zawsze miała dla mnie ciepłe słowo, zwłaszcza w chwilach, gdy w moim mniemaniu moje osiągnięcia naukowe nie były zbyt zachwycające.

Dziękuję także Komitetowi Badań Naukowych — praca niniejsza powstała przy wsparciu materialnym ze strony tej właśnie instytucji udzielonym mi w ramach grantów: 8 T11C 034 10 oraz 8 T11C 035 14.

Bibliografia

- [AE82] K.R. Apt i M.H. Emden, *Contributions to the theory of logic programming*, Journal of the Association for Computing Machinery (1982), nr 29, 841–862.
- [Bar84] H. P. Barendregt, *The Lambda Calculus: Its Syntax and Semantics*, North-Holland, 1984.
- [Bar92] H. P. Barendregt, *Lambda Calculi with Types*, Handbook of Logic in Computer Science (S. Abramsky, D. M. Gabbay i T. S. E. Maibaum, eds.), tom 2, Oxford University Press, 1992, str. 117–309.
- [Ben93] L. S. van Benthem Jutting, *Typing in Pure Type Systems*, Information and Computation (1993), nr 105, 30–41.
- [BLRU97] S. van Bakel, L. Liquori, S. Ronchi della Roca i P. Urzyczyn, *Comparing cubes of typed and type assignment systems*, Annals of Pure and Applied Logic (1997), nr 86, 267–303.
- [BN98] Franz Baader i Tobias Nipkow, *Term Rewriting and All That*, Cambridge University Press, 1998.
- [Boe85] H.-J. Boehm, *Partial polymorphic type inference is undecidable*, 26th Annual Symposium on Foundations of Computer Science, IEEE, October 1985, str. 339–345.
- [CF58] H.B. Curry i R. Feys, *Combinatory logic*, North-Holland, 1958.
- [CH85] Th. Coquand i G. Huet, *Constructions: a higher order proof system for mechanizing mathematics*, EUROCAL'85 (Linz), Lecture Notes in Computer Science, nr 203, Springer-Verlag, 1985.
- [Chu40] A. Church, *A formulation of the simple theory of types*, Journal of Symbolic Logic (1940), nr 5, 56–68.
- [Chu33] A. Church, *A set of postulates for the foundation of logic*, Annals of Mathematics **2** (1932/33), nr 33, 34, 346–366, 839–864.

- [Con86] R. Constable et al., *Implementing Mathematics with the Nuprl Proof Development System*, Prentice-Hall, Englewood Cliffs, New Jersey, 1986.
- [Cur30a] H.B. Curry, *Grundlagen der Kombinatorischen Logik. Teil I.*, Journal of Mathematics (1930), nr LII, 509–536.
- [Cur30b] H.B. Curry, *Grundlagen der Kombinatorischen Logik. Teil II.*, Journal of Mathematics (1930), nr LII, 789–834.
- [Cur34] H.B. Curry, *Functionality in combinatory logic*, Proceedings of National Academy of Science USA, 1934, str. 584–590.
- [Cur69] H.B. Curry, *Modified basic functionality in combinatory logic*, Dialectica (1969), nr 23, 83–92.
- [dB80] N.G. de Bruijn, *A survey of the project AUTOMATH*, To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism (J.P. Seldin i J.R. Hindley, eds.), Academic Press Limited, 1980, str. 579–606.
- [Dow93] Gilles Dowek, *The undecidability of typability in the lambda-pi-calculus*, Typed Lambda Calculi and Applications (M. Bezem i J.F. de Groote, eds.), Lecture Notes in Computer Science, nr 664, 1993, str. 139–145.
- [Gir72] J.-Y. Girard, *Interprétation Fonctionnelle et Élimination des Coupures dans l'Arithmétique d'Ordre Supérieur*, rozprawa doktorska, Université Paris VII, 1972.
- [Gol81] W. D. Goldfarb, *The undecidability of the second-order unification problem*, TCS (1981), nr 13, 225–230.
- [Gor87] M.J.C Gordon, *HOL: A proof generating system for higher-order logic*, VLSI Specification, Verification and Synthesis (G. Birtwistle i P.A. Subrahmanyam, eds.), Kluwer, 1987.
- [Gua64] J.R. Guard, *Automated logic for semi-automated mathematics*, Raport techniczny, AFCRL, 1964, Scientific Report No. 1.
- [Her68] J. Herbrand, *Recherches sur la théorie de la démonstration*, Ecrits logiques de Jacques Herbrand, PUF, 1968, (1930).
- [HF92] P. Hudak i J. Fasel, *A gentle introduction to Haskell*, ACM SIGPLAN Notices **5** (1992), nr 27, T1–T53.
- [HHP93] R. Harper, F. Honsell i G. Plotkin, *A framework for defining logics*, Journal of the Association for Computing Machinery **1** (1993), nr 40, 143–184.

- [Hin69] J.R. Hindley, *The principal typescheme of an object in combinatory logic*, Transactions of American Mathematical Society (1969), nr 146, 29–60.
- [Hoo66] P.K. Hooper, *The undecidability of the Turing machine immortality problem*, Journal of Symbolic Logic **2** (1966), nr 31, 219–234.
- [How80] W. Howard, *The formulae-as-types notion of construction*, To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism (J.P. Seldin i J.R. Hindley, eds.), Academic Press Limited, 1980, str. 479–490.
- [HP89] R. Harper i R. Pollack, *Type checking, universe polymorphism, and typical ambiguity in the Calculus of Constructions*, TAPSOFT'89, Proceedings of the International Joint Conference on Theory and Practice of Software Development, tom 2, Lecture Notes in Computer Science, nr 352, 1989, str. 241–258.
- [HU79] J.E. Hopcroft i J.D. Ullman, *Introduction to Automata Theory, Language, and Computation*, Addison-Wesley Publishing Company, 1979.
- [Hue73] G. Huet, *The undecidability of unification in third order logic*, Information and Control **3** (1973), nr 22, 257–267.
- [Hue75] G. Huet, *A unification algorithm for typed λ -calculus*, Theoretical Computer Science (1975), nr 1, 27–57.
- [Hue76] G. Huet, *Résolution d'Équations dans les Langages d'Ordre 1, 2, ..., ω* , rozprawa doktorska, Université de Paris VII, 1976.
- [Hue96] G. Huet et al., *The Coq Proof Assistant Reference Manual*, INRIA-Rocquencourt — CNRS - ENS Lyon, 1996.
- [KR35] S.C. Kleene i J.B. Rosser, *The inconsistency of certain formal logics*, Annals Math. **2** (1935), nr 36, 630–636.
- [Laf96] Yves Lafont, *The undecidability of second order linear logic without exponentials*, The Journal of Symbolic Logic **61** (1996), nr 2, 541–548.
- [Lei91] D. Leivant, *Finitely stratified polymorphism*, Information and Computation **1** (1991), nr 93, 93–113.
- [Ler97] Xavier Leroy, *The Objective Caml system release 1.07*, INRIA, 1997.

- [LP92] Zhaohui Luo i Randy Pollack, *Lego Proof Development System: User's Manual*, Laboratory of Computer Science, The University of Edinburgh, 1992, LFCS Technical Report ECS-LFCS-92-211.
- [Luc72] C.L. Lucchesi, *The undecidability of the unification problem for third-order languages*, Raport techniczny, Departament of Applied Analysis and Computer Science, University of Waterloo, 1972, Technical Report CSRR 2059.
- [Mil78] R. Milner, *A theory of type polymorphism in programming*, J. Comp. Sys. Sci. **3** (1978), nr 17, 348–375.
- [Mil84] R. Milner, *A proposal for Standard ML*, Proceedings of ACM Symposium on Lisp and Functional Programming (Austin, Texas), 1984.
- [Min61] M.L. Minski, *Recursive unsolvability of post's problem of tag and other topics in the theory of turing machines*, Annals of Mathematics **74** (1961), nr 3, 437–455.
- [Pau86] L.C. Paulson, *Natural deduction as higher-order resolution*, Journal of Logic Programming (1986), nr 3, 237–258.
- [Pfe88] F. Pfenning, *Partial polymorphic type inference and higher-order unification*, Proceedings of the 1988 ACM Conference on Lisp and Functional Programming, ACM Press, lipiec 1988, str. 153–163.
- [PJ72] T. Pietrzykowski i D. Jensen, *A complete mechanization of ω -order theory*, Association for Computing Machinery National Conference, tom 1, 1972, str. 82–92.
- [Rey74] John C. Reynolds, *Towards a theory of type structure*, Programming Symposium (B. Robinet, ed.), Lecture Notes in Computer Science, tom 19, Springer-Verlag, 1974.
- [Rob65] J. A. Robinson, *A machine-oriented logic based on the resolution principle*, Journal of Association for Computing Machinery (1965), nr 12, 23–41.
- [SU98] M.H. Sørensen i P. Urzyczyn, *Lectures on Curry-Howard Isomorphism*, Raport techniczny 14, DIKU, 1998.
- [Tur86] D.A. Turner, *Miranda: a non-strict functional language with polymorphic types*, Proceedings of STACS'86, Lecture Notes in Computer Science, nr 210, Springer-Verlag, 1986.
- [Urz94] Paweł Urzyczyn, *The emptiness problem for intersection types*, Proceedings, Ninth Annual IEEE Symposium on Logic in Computer Science, 1994, str. 300–309.

- [Vea98] Margus Veanes, *The relation between second-order unification and simultaneous rigid E-unification*, Proceedings of Thirteenth Annual IEEE Symposium on Logic in Computer Science, 1998, str. 264–275.
- [Vor96] Andrei Voronkov, *Proof search in intuitionistic logic based on constraint satisfaction*, Theorem Proving with Analytic Tableaux and Related Methods (Terrasini, Palermo) (P. Miglioli, U. Moscato, D. Mundici i M. Ornaghi, eds.), LNAI, nr 1071, Springer Verlag, 1996, str. 312–329.
- [Wel99] J. B. Wells, *Typability and type checking in the second-order λ -calculus are equivalent and undecidable*, Annals of Pure and Applied Logic (1999), nr 98, 111–156.