

Kilka polimorficznych systemów typów
dla języków funkcyjnych z podtypami

Tomasz Łukaszewicz

PRACA MAGISTERSKA

PROMOTOR: prof. dr hab. Jerzy Tiuryn

Uniwersytet Warszawski
Wydział Matematyki, Informatyki i Mechaniki

Warszawa, kwiecień 1999

Spis rzeczy

1	Wstęp	1
1.1	Wprowadzenie	1
1.2	Układ pracy	3
2	Podstawowe pojęcia	5
2.1	Lambda-termny	5
2.2	Typy	7
3	Systemy intersekcyjne	9
3.1	Podstawowe pojęcia	9
3.2	System \mathbf{I}_c^s	10
3.2.1	Podstawowe własności systemu \mathbf{I}_c^s	10
3.2.2	β -redukcja	15
3.2.3	Typowalność w systemie \mathbf{I}_c^s	17
3.3	System \mathbf{I}_c	25
3.4	Porównanie \mathbf{I}_c z \mathbf{I}_c dla języka bez podtypów	28
4	System \mathbf{ML}_c^s	33
4.1	Podstawowe pojęcia	33
4.2	Podstawowe własności	35
4.3	Typowalność	42
5	Porównanie systemu \mathbf{I}_c^s z systemem \mathbf{ML}_c^s	53
5.1	Od \mathbf{ML}_c^s do \mathbf{I}_c^s	54
5.2	Od \mathbf{I}_c^s do \mathbf{ML}_c^s	57
5.3	Wnioski	63
	Podsumowanie	65
	Literatura	67

Rozdział 1

Wstęp

W niniejszej pracy analizuję kilka zaprojektowanych przeze mnie, rozstrzygalnych polimorficznych systemów typów dla języków funkcyjnych z podtypami. Po pierwsze, rozważam język będący rachunkiem lambda ze stałymi i odpowiednio zmodyfikowane systemy typów intersekcyjnych rangi drugiej — \mathbf{I}_c^s oraz \mathbf{I}_c ; dowodzę między innymi poprawności i rozstrzygalności tych systemów, a także to, że oba są sobie równoważne. Po drugie, rozważam język ML i odpowiednio zmodyfikowany system typów ML: \mathbf{ML}_c^s . System \mathbf{ML}_c^s powstał w odpowiedzi na pytanie, w jaki sposób należy rozszerzyć tradycyjny system typów ML, aby miał on siłę typowania możliwie zbliżoną do siły intersekcyjnego systemu \mathbf{I}_c^s . Po wyprowadzeniu niezbędnych do wykazania tego związku własności systemu \mathbf{ML}_c^s (między innymi własności typu głównego, rozumianej tak samo, jak w przypadku tradycyjnego systemu ML), przeprowadzam dowód częściowej równoważności systemów \mathbf{I}_c^s i \mathbf{ML}_c^s .

W pracy starałem się zamieścić wyłącznie własne wyniki — jeśli jakiejś jej fragmenty są zapożyczeniami z pomysłów zawartych we wcześniejszych publikacjach, to informuję o tym w tekście.

1.1 Wprowadzenie

Mianem *języków funkcyjnych* określa się języki programowania, w których funkcje są wartościami „pierwszej kategorii” — czyli języki, w których istnieją wyrażenia tworzące funkcje, a wartości funkcyjne mogą być przypisywane zmiennym, przekazywane jako argumenty innym funkcjom, zwracane jako argumenty funkcji i tak dalej. W praktyce taki opis wydaje się być równoważny stwierdzeniu, że język funkcyjny to język posiadający jako swój podjęzyk rachunek lambda (rachunek anonimowych funkcji, dokładnie wyłożony między innymi w pracy [1]). Żaden z tych dwóch opisów nie jest oczywiście formalną definicją języka funkcyjnego, bo i takowa nie istnieje. Jako ciekawostkę można wspomnieć, że regulamin międzynarodowych zawodów w programowaniu funkcyjnym ICFP'98, zorganizowanych w 1998 roku na

Massachusetts Institute Of Technology, dopuszczał użycie dowolnego języka do napisania programu stanowiącego przedmiot konkursu, zaś przykładowy program, dołączony do regulaminu jako „wzorcowy“, był napisany w języku UNIX-owego interpretera poleceń. A to dlatego, że organizatorzy nie byli w stanie podać kryteriów, według których można by ustalić, czy dany język jest „funkcyjny“.

W naszej pracy będziemy rozważać bardzo okrojone języki funkcyjne, nie wykraczające specjalnie poza rachunek lambda. Dokładniej, programami w rozważanych przez nas językach będą po prostu termy lambda-rachunku, rozszerzonego o stałe (na przykład interpretowane jako liczby), a czasem o konstrukcję `let`, zaczerpniętą z popularnego języka funkcyjnego ML. Przez wykonanie każdego takiego programu będziemy rozumieli obliczenie jego wartości — poprzez uproszczenie go zgodnie z tak zwaną β -redukcją oraz obliczenie pewnych predefiniowanych funkcji „pierwotnych“ (na przykład operacji arytmetycznych).

Typy służą do klasyfikacji termów według pewnych ich podstawowych własności i spełniają różne funkcje w językach programowania i w logice. W niniejszej pracy będą one służyły do klasyfikacji termów rozszerzonego rachunku lambda względem tego, jakiego rodzaju wartości te termy wyrażają. Przykładowo: możemy powiedzieć, że wyrażenie `7.0` jest typu rzeczywistego — co zapiszemy umownie przez `7.0 : Real`, wyrażenie `“foo”` oznacza pewien ciąg znaków — co zapiszemy przez `“foo” : String`, oraz, że wyrażenie `+` oznacza funkcję biorącą dwa argumenty będące liczbami rzeczywistymi i zwracającą jako wynik liczbę rzeczywistą — co zapiszemy przez `+ : Real → Real → Real`. Korzyścią z rozważania tak rozumianych typów jest między innymi to, że możemy natychmiast odrzucić pewne programy jako niepoprawne. Na przykład, o ile program `7.0 + 7.0` przy powyższych ustaleniach może mieć jakiś sens, to program `7.0 + “foo”` sensu żadnego nie posiada — i rzeczywiście, prawdopodobnie w praktyce `+` byłby wbudowaną w język funkcją realizującą dodawanie liczb rzeczywistych, która dałaby nieprzewidywalny rezultat przy próbie obliczenia jej dla jednego z argumentów będącego nie liczbą rzeczywistą, lecz ciągiem znaków. Innym miejscem, gdzie odpowiednio zaprojektowany system typów pomaga, są programy mogące pętlić się podczas wykonania — jak się bowiem okazuje, niektóre wyrażenia rachunku lambda można upraszczać zgodnie z β -redukcją w nieskończoność, nie uzyskując żadnego ostatecznego rezultatu.

W dużym uproszczeniu rzecz ujmując, systemy typów określa się mianem *polimorficznych*, jeśli umożliwiają one zadeklarowanie pewnej wartości — na przykład funkcji — jako mającej wiele różnych typów naraz. Przykładowo, moglibyśmy potrzebować funkcji „daj-7.0“, która dla każdego argumentu działa tak samo: po jego pobraniu zwraca liczbę rzeczywistą 7.0. Możliwość zadeklarowania takiej funkcji jako mającej jednocześnie wiele różnych typów pozwala nam zaoszczędzić na definiowaniu funkcji wielokrotnie, dla różnych typów argumentów.

Wróćmy jeszcze na chwilę do omawianej wcześniej funkcji $+$. Zauważmy, że jeśli 7 uznamy za liczbę całkowitą — co zapiszemy przez $7 : \text{Int}$, to wyrażenie $7.0 + 7$ będzie tak samo niepoprawne, jak wyrażenie $7.0 + \text{“foo”}$: typ drugiego argumentu nie będzie się zgadzał. Tymczasem w matematyce zwykło się przyjmować, że liczba całkowita jest równocześnie liczbą rzeczywistą, i dlatego niemożność obliczenia $7.0 + 7$ wydaje się nienaturalna. Tu właśnie pomocna okazuje się być koncepcja *podtypów*: jeśli T_1, T_2 są pewnymi zbiorami, to stwierdzenie, że $T_1 \leq T_2$ (T_1 jest podtypem T_2) oznacza, że każdy element zbioru T_1 jest także elementem zbioru T_2 . Możemy zatem zaprojektować system typów uwzględniający, że zbiór liczb całkowitych jest podtypem zbioru liczb rzeczywistych, a wówczas $7.0 + 7$ będzie poprawnym wyrażeniem. Inna sprawa, że takie założenie prawdopodobnie będzie pociągało za sobą zmiany w implementacji reszty kompilatora/interpretera, gdyż zazwyczaj wewnętrzne reprezentacje liczby całkowitej 7 i liczby rzeczywistej 7.0 będą się od siebie różniły, a zatem przed wykonaniem funkcji $+$ wymagana będzie konwersja z typu całkowitego na rzeczywisty.

W naszej pracy będziemy rozważać tylko pewne dwa zbiory \perp i \top typów atomowych, takie, że $\perp \leq \top$ (mogą one odpowiadać na przykład odpowiednio zbiorowi liczb całkowitych i rzeczywistych); wszystkie inne relacje podtypowe będą indukowane przez ten jeden porządek. Moglibyśmy rozważać szerszy zbiór typów atomowych z odpowiednio zdefiniowaną relacją podtypu pomiędzy nimi; w szczególności prawdopodobnie większość — jeśli nie całość — pracy dałaby się uogólnić na dowolny zbiór typów atomowych z relacją podtypowania pomiędzy nimi będącą kratą lub sumą rozłącznych krat. Tym niemniej, dla uproszczenia i ustalenia uwagi zdecydowaliśmy się na rozwiązanie ograniczone do \perp i \top .

Powyższe wprowadzenie jest bardzo pobieżne; wprawdzie wszystkie pojęcia wykorzystywane w pracy zostaną uściślone, to jednak całość wciąż nie będzie przystępna dla osób, które nie zetknęły się dotychczas z programowaniem funkcyjnym i z koncepcjami z nim związanymi. Takim osobom warto polecić literaturę wprowadzającą w tę dziedzinę, na przykład kilka początkowych rozdziałów książki [5].

1.2 Układ pracy

W rozdziale 2 wprowadzamy podstawowe pojęcia, których znajomość jest niezbędna do zrozumienia niniejszej pracy.

W rozdziale 3 szczegółowo omawiamy systemy intersekcyjne \mathbf{I}_c^s i \mathbf{I}_c . Najpierw wprowadzamy system \mathbf{I}_c^s i dowodzimy jego zamkniętości ze względu na β -redukcję oraz to, że termy w nim typowalne są silnie normalizowalne. Następnie sprowadzamy problem typowalności w tym systemie do rozstrzygalnego problemu istnienia rozwiązania pewnego układu równań i nierówności podtypowych pomiędzy typami prostymi. Później wprowadzamy system \mathbf{I}_c ,

będący rozszerzeniem systemu \mathbf{I}_c^s o silniejszą regułę podtypowania. Okazuje się, że tak powstały system jest równoważny systemowi \mathbf{I}_c . Na zakończenie rozdziału wspominamy o różnicach pomiędzy naszymi systemami a systemem typów intersekcyjnych dla języka bez podtypów, omawianym przez Jima w pracy [6].

W rozdziale 4 omawiamy system \mathbf{ML}_c^s . Ograniczymy się do zademonstrowania tylko tych własności, które będą nam potrzebne w rozdziale 5 do udowodnienia częściowej równoważności systemów \mathbf{I}_c^s i \mathbf{ML}_c^s . Dlatego właśnie większą część rozdziału stanowią dowody substytutowości i własności typu głównego.

W rozdziale 5 pokazujemy, że systemy \mathbf{ML}_c^s i \mathbf{I}_c^s mają podobną siłę typowania: są one równoważne dla pewnych klas termów (z dokładnością do zamiany konstrukcji języka ML na konstrukcje rachunku lambda i odwrotnie). Wprawdzie system \mathbf{I}_c^s okazuje się być nieco silniejszym od systemu \mathbf{ML}_c^s , to jednak można mówić o pokrewieństwie obu systemów.

Rozdział 2

Podstawowe pojęcia

W niniejszym rozdziale zdefiniujemy podstawowe pojęcia wykorzystywane w całej pracy. Dodatkowo, podstawowe pojęcia ściśle związane z systemami intersekcyjnymi zostaną zdefiniowane w części „Podstawowe pojęcia“ rozdziału 3, zaś podstawowe pojęcia ściśle związane z systemem \mathbf{ML}_c^s zostaną zdefiniowane w części „Podstawowe pojęcia“ rozdziału 4. Różne bardziej zaawansowane definicje będą natomiast wprowadzane — stosownie do potrzeb — w toku wywodu.

Będziemy używać liter x, y, \dots do oznaczania zmiennych rachunku lambda, pochodzących z przeliczalnego zbioru \mathbf{Vars} , litery c do oznaczania stałych rachunku lambda oraz liter M, N, \dots do oznaczania termów rachunku lambda. Ponadto, będziemy posługiwali się literami s, t, \dots do oznaczania zmiennych typowych, pochodzących z przeliczalnego zbioru $\mathbf{TypeVars}$, symbolami \perp i \top do oznaczania stałych typowych, symbolami σ, π, τ, \dots do oznaczania typów oraz symbolami C, D, \dots do oznaczania zbiorów warunków. Zwyczajowo, litera A będzie oznaczać środowiska, zaś litery S, T, \dots — podstawienia.

2.1 Lambda-termny

Termy rachunku lambda, będące naszymi programami podczas rozważania systemów typów intersekcyjnych \mathbf{I}_c^s i \mathbf{I}_c , są zdefiniowane następującą gramatyką:

$$M_\lambda ::= x \mid c^\perp \mid c^\top \mid (M_\lambda M_\lambda) \mid \lambda x.M_\lambda.$$

W przypadku systemu typów \mathbf{ML}_c^s naszymi programami będą natomiast *termy ML*, zawierające prócz konstrukcji z M_λ dodatkową konstrukcję *let*:

$$M_{ml} ::= x \mid c^\perp \mid c^\top \mid (M_{ml} M_{ml}) \mid \lambda x.M_{ml} \mid \mathbf{let} \ x = M_{ml} \ \mathbf{in} \ M_{ml}.$$

Dla termu M (z M_λ lub M_{ml}) definiujemy zbiór $\mathbf{FV}(M) \subseteq \mathbf{Vars}$ *zmiennych wolnych termu* M w sposób następujący:

- (i) $\text{FV}(c) = \emptyset$ (dla $c \in \{c_{\perp}, c^{\top}\}$),
- (ii) $\text{FV}(x) = \{x\}$,
- (iii) $\text{FV}(\lambda x N) = \text{FV}(N) - \{x\}$,
- (iv) $\text{FV}(M_1 M_2) = \text{FV}(M_1) \cup \text{FV}(M_2)$,
- (v) $\text{FV}(\mathbf{let } x = M_1 \mathbf{ in } M_2) = \text{FV}(M_1) \cup (\text{FV}(M_2) - \{x\})$
(dotyczy termów z M_{ml}).

Jeśli $\text{FV}(M) = \emptyset$, to term M nazywamy *zamkniętym*.

Przez *podstawienie* rozumiemy funkcję ze zbioru zmiennych w zbiór termów, która jest identycznością dla wszystkich zmiennych, z wyjątkiem skończonej ich liczby. *Dziedzinę* i *ranę* podstawienia definiujemy w sposób następujący:

$$\begin{aligned} \mathbf{dom}(S) &= \{x \mid Sx \neq x\}, \\ \mathbf{rng}(S) &= \bigcup_{x \in \mathbf{dom}(S)} \text{FV}(Sx). \end{aligned}$$

Jeśli $\mathbf{dom}(S) = \{x_1, \dots, x_n\}$ i $Sx_i = N_i$ dla wszystkich $i = 1, \dots, n$, to S możemy zapisać w formie $\{x_1 := N_1, \dots, x_n := N_n\}$. Dwa podstawienia S_1 i S_2 są rozłączne, jeśli zbiory $\mathbf{dom}(S_1)$ i $\mathbf{dom}(S_2)$ są rozłączne. Jeśli podstawienia S_1 i S_2 są rozłączne, to podstawienie $S_1 \cup S_2$ definiujemy w sposób następujący:

$$(S_1 \cup S_2)(x) = \begin{cases} S_1(x) & \text{jeśli } x \in \mathbf{dom}(S_1), \\ S_2(x) & \text{jeśli } x \in \mathbf{dom}(S_2), \\ x & \text{w przeciwnym przypadku.} \end{cases}$$

Podstawienia działające na zmiennych **Vars** będziemy rozszerzali na całe termy M_{λ} i M_{ml} . Wystarczą nam wówczas podstawienia S takie, że zbiór $\mathbf{dom}(S)$ będzie co najwyżej jednoelementowy. Nieformalnie powiemy, że w termie $\lambda x N$ zmienna x jest *zmienną związaną*. Rozszerzając działanie podstawienia na całe termy, w razie potrzeby będziemy przemianowywać zmienne związane. Formalnie:

- (i) $\{x := N\}c = c$ (dla $c \in \{c_{\perp}, c^{\top}\}$),
- (ii) $\{x := N\}PQ = \{x := N\}P \{x := N\}Q$,
- (iii) $\{x := N\}(\lambda x P) = \lambda x P$,
- (iv) jeśli $x \neq y$, to $\{x := N\}(\lambda y P) = \lambda y(\{x := N\}P)$, o ile $y \notin \text{FV}(N)$ lub $x \notin \text{FV}(P)$,

- (v) jeśli $x \neq y$, to $\{x := N\}(\lambda y P) = \{x := N\}(\lambda z(\{y := z\}P))$, o ile $y \in \text{FV}(N)$ i $x \in \text{FV}(P)$, przy czym z wybieramy jako $v_i \in \mathbf{Vars}$ o najmniejszym i takim, że $v_i \notin \text{FV}(P) \cup \text{FV}(N)$,
- (vi) $\{x := N\}(\mathbf{let } y = M_1 \mathbf{ in } M_2) = \mathbf{let } y' = M'_1 \mathbf{ in } M'_2$, gdzie $(\lambda y' M'_2)M'_1 = \{x := N\}((\lambda y M_2)M_1)$ (dotyczy termów z M_{ml}).

Będziemy utożsamiać termy różniące się jedynie nazwami zmiennych związanych. Formalnie, wprowadzamy pojęcie α -konwersji, oznaczanej przez $=_\alpha$. Będzie to najmniejsza relacja równoważności pomiędzy termami spełniająca warunek

$$\lambda x N =_\alpha \lambda y(\{x := y\}N), \quad \text{jeśli } y \notin \text{FV}(N)$$

i rozszerzona na nadtermy (to znaczy taka, że jeśli $M =_\alpha M'$, to $MN =_\alpha M'N$, jeśli $M =_\alpha M'$, to $NM =_\alpha NM'$ i tak dalej). Termy uważamy za klasy abstrakcji α -konwersji. Odstąpimy od tego w podrozdziale 5.2.

Wprowadzamy relację jednokierunkowej redukcji/ewaluacji termów M_λ i M_{ml} , zwaną β -redukcją, zgodnie z którą będą upraszczane nasze programy podczas ich wykonywania. β jest regułą:

$$(\lambda x M)N \rightarrow \{x := N\}M.$$

Przez \rightarrow_β będziemy rozumieli rozszerzenie reguły β na nadtermy. Wówczas przez \rightarrow_β oznaczamy domknięcie przechodnio-zwrotne relacji \rightarrow_β .

β -redexem jest każdy term pasujący do lewej strony reguły β . Mówimy, że term M jest w postaci β -normalnej, lub w postaci β -nf, jeśli żaden podterm M nie jest β -redexem. Term M jest *normalizowalny*, jeśli dla pewnego N : $M \rightarrow_\beta N$ i N jest w postaci normalnej. Wówczas term N jest postacią normalną termu M . Jeśli term ma postać normalną, to jest ona unikalna (patrz na przykład [1]), a zatem jest sens mówić o funkcji β -nf, przyporządkowującej termom normalizowalnym ich postaci β -normalne. Term M jest *silnie normalizowalny*, jeśli nie istnieje nieskończona sekwencja β -redukcji $M \rightarrow_\beta M_1 \rightarrow_\beta M_2 \rightarrow_\beta \dots$.

W termach M_{ml} występuje konstrukcja **let**, operacyjnie odpowiadająca aplikacji do abstrakcji: **let** $x = M$ **in** N tłumaczy się na $(\lambda x N)M$, a to drugie wyrażenie jest ewaluowane zgodnie z β -redukcją.

2.2 Typy

Rozważane typy będą zupełnie różne — zależnie od tego, czy będziemy omawiać systemy intersekcyjne, czy też rozszerzenia ML. Wszędzie jednak będą obecne *typy proste (bazowe)*, zdefiniowane jako najmniejszy zbiór spełniający równanie:

$$\mathbf{T}_0 = \mathbf{TypeVars} \cup \{\perp, \top\} \cup \{(\sigma \rightarrow \tau) \mid \sigma, \tau \in \mathbf{T}_0\}.$$

Dla typu σ z \mathbf{T}_0 definiujemy zbiór $\text{FV}(\sigma) \subseteq \mathbf{TypeVars}$ *wolnych zmiennych typowych typu* σ jako zbiór wszystkich zmiennych typowych występujących w σ . Dla typów z innych zbiorów definicje FV zostaną podane później.

Środowisko to skończony zbiór $\{x_1 : \sigma_1; \dots; x_n : \sigma_n\}$ par (zmienna, typ), gdzie zmienne x_1, \dots, x_n są parami różne. $A(x)$ oznacza typ sparowany z x w środowisku A , $\mathbf{dom}(A)$ oznacza zbiór $\{x \mid \exists \tau. (x : \tau) \in A\}$, zaś A_x oznacza środowisko A po usunięciu z niego pary odpowiadającej zmiennej x . Środowisko A jest typu T , jeśli $A(x) \in T$ dla wszystkich $x \in \mathbf{dom}(A)$. Pojęcie zbioru wolnych zmiennych typowych rozszerzamy na środowiska: $\text{FV}(A) = \bigcup_{(x:\tau) \in A} \text{FV}(\tau)$.

Asercja jest relacją pomiędzy środowiskami, termami i typami, zapisywaną w formie $A \vdash M : \sigma$. Term M jest typowalny, jeśli $A \vdash M : \sigma$ dla pewnych A i σ . Parę $\langle A, \sigma \rangle$ będziemy nazywać *parą*. Pojęcie wolnych zmiennych typowych rozszerzamy na pary: $\text{FV}(\langle A, \sigma \rangle) = \text{FV}(A) \cup \text{FV}(\sigma)$. Dwie pary są *rozłączne*, jeśli zbiory ich zmiennych wolnych są rozłączne.

Pojęcie podstawienia dotyczy też typów. Podstawienia takie będą zawsze funkcjami ze zbioru $\mathbf{TypeVars}$ w zbiór \mathbf{T}_0 ; **nigdzie nie będziemy rozważać podstawień o przeciwdziedzinach będących innymi zbiorami typów, niż zbiór \mathbf{T}_0 .**

Działanie podstawienia na zmiennych typowych jest w naturalny sposób rozszerzane na całe typy z \mathbf{T}_0 :

$$(i) \ S(\perp) = \perp, \ S(\top) = \top,$$

$$(ii) \ S(\sigma \rightarrow \tau) = S(\sigma) \rightarrow S(\tau).$$

W przypadku innych typów odpowiednie definicje zostaną podane później. Gdy już umiemy aplikować podstawienia do typów, to w naturalny sposób rozszerzamy ich działanie na środowiska i pary.

Na zbiorze \mathbf{T}_0 definiujemy porządek \leq_0 jako najmniejszy porządek częściowy zamknięty ze względu na następujące reguły:

$$(i) \ \perp \leq_0 \top,$$

$$(ii) \ \text{jeśli } \tau_1 \leq_0 \sigma_1 \text{ i } \sigma_2 \leq_0 \tau_2, \text{ to } (\sigma_1 \rightarrow \sigma_2) \leq_0 (\tau_1 \rightarrow \tau_2).$$

Rozdział 3

Systemy intersekcyjne

W tym rozdziale omówimy szczegółowo dwa systemy typów intersekcyjnych: \mathbf{I}_c^s i \mathbf{I}_c . Udowodnimy między innymi zamkniętość tych systemów ze względu na β -redukcję, ich równoważność oraz to, że problem typowości w nich jest rozstrzygalny. Wspomnimy także o tym, co różni nasze systemy od podobnych systemów przeznaczonych dla języków funkcyjnych bez podtypów.

3.1 Podstawowe pojęcia

Wiele z poniższych definicji ma swoje odpowiedniki w pracy Jima [6] — tutaj rozszerzamy je na język z podtypami.

W przypadku systemów intersekcyjnych \mathbf{I}_c^s i \mathbf{I}_c każdy rozważany typ będzie należał do \mathbf{T}_0 lub do jednego z następujących zbiorów, zdefiniowanych jako najmniejsze zbiory spełniające odpowiednie równania:

$$\begin{aligned}\mathbf{T}_1 &= \mathbf{T}_0 \cup \{(\sigma \wedge \tau) \mid \sigma, \tau \in \mathbf{T}_1\}, \\ \mathbf{T}_2 &= \mathbf{T}_0 \cup \{(\sigma \rightarrow \tau) \mid \sigma \in \mathbf{T}_1, \tau \in \mathbf{T}_2\}.\end{aligned}$$

\mathbf{T}_1 to *typy intersekcyjne rangi pierwszej*, będące przecięciami pewnej liczby typów prostych. \mathbf{T}_2 natomiast jest zbiorem *typów intersekcyjnych rangi drugiej*, z dodatkowym ograniczeniem, że przecięcia znajdują się co najwyżej z lewej strony strzałki.

Właściwie czytelnikowi należy się uściślenie pojęcia rangi typu. Otóż, rangę typu najłatwiej ustalić mając na uwadze jego reprezentację w postaci drzewa. Typ intersekcyjny jest rangi k , jeśli żadna ścieżka od korzenia do operatora intersekcji „ \wedge ” nie wiedzie na lewo od k lub więcej strzałek. (Taka sama zasada obowiązuje w stosunku do typów z kwantyfikatorami — wówczas zamiast ścieżki do „ \wedge ” rozważamy ścieżkę do „ \forall ”.)

Dla uproszczenia przyjmujemy konwencję syntaktyczną, że „ \wedge ” jest łącznym, przemennym i idempotentnym operatorem, tak więc każdy typ z \mathbf{T}_1 może być uważany za skończony i niepusty zbiór typów prostych, zapisywany w postaci $(\bigwedge_{i \in I} \sigma_i)$, gdzie każde $\sigma_i \in \mathbf{T}_0$.

Na zbiorach \mathbf{T}_1 i \mathbf{T}_2 definiujemy odpowiednio *porządki* \leq_1 i \leq_2 jako najmniejsze porządki częściowe zamknięte ze względu na następujące reguły:

- Dla \leq_1 : jeśli $\forall j \in J. \exists i \in I. \sigma_i \leq_0 \tau_j$, to $\bigwedge_{i \in I} \sigma_i \leq_1 \bigwedge_{j \in J} \tau_j$.
- Dla \leq_2 :
 - (i) jeśli $\sigma \leq_0 \tau$, to $\sigma \leq_2 \tau$,
 - (ii) jeśli $\tau_1 \leq_1 \sigma_1$ i $\sigma_2 \leq_2 \tau_2$, to $(\sigma_1 \rightarrow \sigma_2) \leq_2 (\tau_1 \rightarrow \tau_2)$.

Lemat 3.1 *Dla $i \in \{0, 1, 2\}$ i dowolnego podstawienia S , jeśli $\sigma \leq_i \tau$, to $S\sigma \leq_i S\tau$.*

Dowód: wynika bezpośrednio z definicji \leq_i dla $i \in \{0, 1, 2\}$. ■

Dla środowisk typu \mathbf{T}_1 definiujemy *operator* $+$. Jeśli A_1, A_2 są środowiskami typu \mathbf{T}_1 , to $A_1 + A_2$ – środowisko również typu \mathbf{T}_1 – definiujemy następująco:

$$(A_1 + A_2)(x) = \begin{cases} A_1(x) & \text{jeśli } x \notin \mathbf{dom}(A_2), \\ A_2(x) & \text{jeśli } x \notin \mathbf{dom}(A_1), \\ A_1(x) \wedge A_2(x) & \text{w przeciwnym przypadku.} \end{cases}$$

Wolne zmienne typowe dla typów z \mathbf{T}_1 i z \mathbf{T}_2 definiujemy analogicznie, jak dla typów z \mathbf{T}_0 : są to zmienne typowe występujące w typie. Tak samo z podstawieniami: ich działanie na typy z \mathbf{T}_1 i \mathbf{T}_2 rozszerzamy w naturalny sposób.

3.2 System \mathbf{I}_c^s

Definicja 3.2 *Asercje w systemie \mathbf{I}_c^s są zdefiniowane indukcyjnie regułami przedstawionymi na rysunku 3.1. Piszemy $\mathbf{I}_c^s \triangleright A \vdash M : \sigma$, jeśli asercja $A \vdash M : \sigma$ została otrzymana przy pomocy reguł systemu \mathbf{I}_c^s . ■*

Górny indeks „s” w nazwie \mathbf{I}_c^s sygnalizuje, że system jest sterowany składnią, w odróżnieniu od wprowadzonego dalej systemu \mathbf{I}_c .

System \mathbf{I}_c^s jest rozszerzeniem systemu \mathbf{I}_2^s , omawianego przez Jima w pracy [6], który to z kolei system jest modyfikacją podobnego systemu omawianego przez van Bakela w pracy [10].

3.2.1 Podstawowe własności systemu \mathbf{I}_c^s

Regułę (VAR) można zdefiniować na kilka sposobów; definicja z rysunku 3.1 wydaje się najbardziej praktyczna. Czasem jednak przydatna jest nieco inaczej sformułowana reguła (VAR) — taka, jak w poniższym lemacie.

(CONST \perp)	$A \vdash c^\perp : \perp$
(CONST \top)	$A \vdash c^\top : \top$
(VAR)	$A \cup \{x : (\bigwedge_{i \in I} \tau_i)\} \vdash x : \sigma$ (jeśli $\exists i \in I. \tau_i \leq_0 \sigma$)
(ABS)	$\frac{A_x \cup \{x : \sigma\} \vdash M : \tau}{A \vdash (\lambda x M) : \sigma \rightarrow \tau}$
(APP)	$\frac{A \vdash M : (\bigwedge_{i \in I} \tau_i) \rightarrow \sigma, \quad (\forall i \in I) A \vdash N : \tau_i}{A \vdash (MN) : \sigma}$
(SUB \perp)	$A \vdash c^\perp : \top$

Rysunek 3.1: Reguły \mathbf{I}_c^s . Środowiska są typu \mathbf{T}_1 , wyprowadzone typy należą do \mathbf{T}_2 .

Lemat 3.3 *Reguła (VAR) równoważna jest reguła (VAR'):*

$$(VAR') \quad A \cup \{x : \tau\} \vdash x : \sigma \quad (\text{jeśli } \sigma \in \mathbf{T}_0 \text{ i } \tau \leq_1 \sigma).$$

Dowód: Korzystamy z reguły (VAR) i definicji \leq_1 . ■

Poniższy lemat wyraża ideę, że dorzucenie do środowiska niepotrzebnych par (zmienna, typ) nie psuje jakości wyprowadzalnego typu.

Lemat 3.4 *Jeśli $\mathbf{I}_c^s \triangleright A \vdash M : \tau$ i $x \notin \mathbf{dom}(A)$, to $\mathbf{I}_c^s \triangleright A \cup \{x : \sigma\} \vdash M : \tau$.*

Dowód: Trywialny dowód indukcyjny ze względu na budowę termu M . ■

Następny zaś lemat mówi, że wyrzucenie ze środowiska niepotrzebnych par (zmienna, typ) także nie psuje jakości wyprowadzalnego typu.

Lemat 3.5 *Jeśli $\mathbf{I}_c^s \triangleright A \cup \{x : \sigma\} \vdash M : \tau$ i x nie jest zmienną wolną w M , to $\mathbf{I}_c^s \triangleright A \vdash M : \tau$.*

Dowód: Trywialny dowód indukcyjny ze względu na budowę termu M . ■

Nasuwa się też pytanie, czy przypadkiem wzmocnienie środowiska nie popsuje nam jakości wyprowadzalnego typu. I tym razem wynik jest zgodny z intuicją.

Lemat 3.6 *Jeśli $\mathbf{I}_c^s \triangleright A \cup \{x : \sigma_1\} \vdash M : \tau$ i $\sigma_2 \leq_1 \sigma_1$, to $\mathbf{I}_c^s \triangleright A \cup \{x : \sigma_2\} \vdash M : \tau$.*

Dowód: Dowód indukcyjny ze względu na budowę termu M :

- (i) Jeśli $M = c^\perp$, to $\tau = \perp$ lub $\tau = \top$, niezależnie od środowiska A . Analogicznie dla $M = c^\top$.
- (ii) Jeśli $M = v$ (zmienna), to rozważamy sytuację, w której $\mathbf{I}_c^s \triangleright A \cup \{x : \sigma_1\} \vdash v : \tau$.
 - a) Jeśli $v \neq x$, to zmiana założeń o x w środowisku dalej pozwala wyprowadzić $v : \tau$, bo założenia w środowisku o v nie są zmieniane, a używaną regułą jest (VAR).
 - b) Jeśli $v = x$, to mamy sytuację $\mathbf{I}_c^s \triangleright A \cup \{x : \sigma_1\} \vdash x : \tau$. Asercję tę otrzymano przy pomocy reguły (VAR), a ta z kolei równoważna jest regule (VAR'), a zatem zachodzi $\sigma_1 \leq_1 \tau$. Stąd i z faktu $\sigma_2 \leq_1 \sigma_1$ mamy przez przechodność \leq_1 również $\sigma_2 \leq_1 \tau$. Stąd przez (VAR') otrzymujemy $\mathbf{I}_c^s \triangleright A \cup \{x : \sigma_2\} \vdash x : \tau$.
- (iii) Jeśli $M = \lambda v N$, to rozważamy sytuację, w której $\mathbf{I}_c^s \triangleright A \cup \{x : \sigma_1\} \vdash \lambda v N : \tau_1 \rightarrow \tau_2$.
 - a) Jeśli $v = x$, to rozważamy sytuację, w której $\mathbf{I}_c^s \triangleright A \cup \{x : \sigma_1\} \vdash \lambda x N : \tau_1 \rightarrow \tau_2$. Ta asercja została otrzymana z przesłanki $\mathbf{I}_c^s \triangleright A \cup \{x : \tau_1\} \vdash N : \tau_2$ przez (ABS). Z takiej przesłanki w sposób trywialny otrzymamy również przez (ABS) $\mathbf{I}_c^s \triangleright A \cup \{x : \sigma_2\} \vdash \lambda x N : \tau_1 \rightarrow \tau_2$.
 - b) Jeśli $v \neq x$, to rozważamy sytuację, w której $\mathbf{I}_c^s \triangleright A \cup \{x : \sigma_1\} \vdash \lambda v N : \tau_1 \rightarrow \tau_2$. Ta asercja została otrzymana z przesłanki $\mathbf{I}_c^s \triangleright A \cup \{x : \sigma_1\} \cup \{v : \tau_1\} \vdash N : \tau_2$ przez (ABS). Przesłanka ta przez indukcję implikuje $\mathbf{I}_c^s \triangleright A \cup \{x : \sigma_2\} \cup \{v : \tau_1\} \vdash N : \tau_2$, a stąd przez (ABS) otrzymujemy $\mathbf{I}_c^s \triangleright A \cup \{x : \sigma_2\} \vdash \lambda v N : \tau_1 \rightarrow \tau_2$.
- (iv) $M = M_1 M_2$. Asercja $\mathbf{I}_c^s \triangleright A \cup \{x : \sigma_1\} \vdash M_1 M_2 : \tau$ została otrzymana z przesłanek $\mathbf{I}_c^s \triangleright A \cup \{x : \sigma_1\} \vdash M_1 : (\bigwedge_{i \in I} \mu_i) \rightarrow \tau$ i $(\forall i \in I) \mathbf{I}_c^s \triangleright A \cup \{x : \sigma_1\} : M_2 : \mu_i$. Założenia te przez indukcję implikują $\mathbf{I}_c^s \triangleright A \cup \{x : \sigma_2\} \vdash M_1 : (\bigwedge_{i \in I} \mu_i) \rightarrow \tau$ oraz $(\forall i \in I) \mathbf{I}_c^s \triangleright A \cup \{x : \sigma_2\} : M_2 : \mu_i$, a stąd przez (APP) otrzymujemy $\mathbf{I}_c^s \triangleright A \cup \{x : \sigma_2\} \vdash M_1 M_2 : \tau$. ■

I jeszcze jeden lemat, łączący dwa poprzednie. Nie wykorzystuje on całej mocy lematu 3.6 i tak naprawdę mógłby być udowodniony indukcyjnie bezpośrednio; bez korzystania z jakichkolwiek lematów. W dalszej części pracy będziemy jednak korzystać z pełnej mocy lematu 3.6 i dlatego i tak musieliśmy go udowodnić.

Lemat 3.7 (Osłabianie) *Jeśli $\mathbf{I}_c^s \triangleright A \vdash M : \tau$, to $\mathbf{I}_c^s \triangleright A + A' \vdash M : \tau$, dla dowolnego środowiska A' .*

Dowód: Wynika bezpośrednio z definicji operatora $+$ oraz z lematów 3.4 i 3.6. ■

Substytutywność to kolejna istotna cecha naszego systemu.

Lemat 3.8 (Substytutywność) *Jeśli $\mathbf{I}_c^s \triangleright A \vdash M : \sigma$, to $\mathbf{I}_c^s \triangleright SA \vdash M : S\sigma$, dla dowolnego podstawienia S .*

Dowód: Dowód indukcyjny ze względu na budowę termu M :

- (i) Jeśli $M = c^\top$, to do uzyskania asercji użyto reguły (CONST \top) i $\sigma = \top$. Z reguły tej mamy również $\mathbf{I}_c^s \triangleright SA \vdash c^\top : \top$, dla dowolnego podstawienia S . Ponadto, $S(\top) = \top$, więc mamy $\mathbf{I}_c^s \triangleright SA \vdash c^\top : S(\top)$, co należało wykazać. Analogiczny dowód przeprowadzamy dla $M = c^\perp$; tylko, że wówczas trzeba rozważyć dwa przypadki — dla $\sigma = \perp$ i dla $\sigma = \top$.
- (ii) Jeśli $M = x$ (zmienna), to do uzyskania asercji użyto reguły (VAR), czyli $A(x) = (\bigwedge_{i \in I} \sigma_i)$ i $\sigma_{i_0} \leq_0 \sigma$ dla pewnego $i_0 \in I$. Wówczas $SA(x) = (\bigwedge_{i \in I} S\sigma_i)$ i z lematu 3.1, skoro $\sigma_{i_0} \leq_0 \sigma$, to $S\sigma_{i_0} \leq_0 S\sigma$. Czyli, na mocy (VAR), $\mathbf{I}_c^s \triangleright SA \vdash x : S\sigma$.
- (iii) Jeśli $M = \lambda x N$, to σ musi być postaci $\tau_1 \rightarrow \tau_2$ i skorzystano z przesłanki $\mathbf{I}_c^s \triangleright A_x \cup \{x : \tau_1\} \vdash N : \tau_2$. Wówczas przez indukcję mamy $\mathbf{I}_c^s \triangleright S(A_x \cup \{x : \tau_1\}) \vdash N : S\tau_2$, a stąd przez (ABS) otrzymujemy $\mathbf{I}_c^s \triangleright SA_x \vdash \lambda x N : S\tau_1 \rightarrow S\tau_2 = S(\tau_1 \rightarrow \tau_2)$. Stąd przez osłabianie mamy również $\mathbf{I}_c^s \triangleright SA \vdash \lambda x N : S(\tau_1 \rightarrow \tau_2)$.
- (iv) Jeśli $M = M_1 M_2$, to dla pewnego $(\bigwedge_{i \in I} \tau_i) \in \mathbf{T}_1$ mamy: $\mathbf{I}_c^s \triangleright A \vdash M_1 : (\bigwedge_{i \in I} \tau_i) \rightarrow \sigma$ i $\mathbf{I}_c^s \triangleright A \vdash M_2 : \tau_i$ dla wszystkich $i \in I$. Stosując do tych dwóch przesłanek założenie indukcyjne i korzystając z (APP) otrzymamy żądany rezultat. ■

Reguła podtypowania („subsumption“), spotykana zwykle w systemach typów dla języków z podtypami, ma postać:

$$\frac{A \vdash M : \tau}{A \vdash M : \sigma} \quad (\text{gdzie } \tau \leq \sigma).$$

Ma ona w zamierzeniu umożliwiać otypowanie programów korzystających z istnienia zależności podtypowych, czyli — innymi słowy — jej obecność wyraża fakt, że język ma podtypy. W systemie \mathbf{I}_c^s reguły takiej postaci nie ma; jest tylko specjalny jej przypadek dla stałej c^\perp oraz odpowiednio

zdefiniowana reguła (VAR). Nasuwa się pytanie, czy może własność równoważna regule podtypowania po prostu wynika z pozostałych reguł systemu \mathbf{I}_c^s . Odpowiedź jest negatywna, bo na przykład wyprowadzimy asercję $\mathbf{I}_c^s \triangleright \{x : \sigma \rightarrow \tau\} \vdash x : \sigma \rightarrow \tau$ i zachodzi $\sigma \rightarrow \tau \leq_2 (\sigma \wedge \sigma') \rightarrow \tau$, ale nie wyprowadzimy $\mathbf{I}_c^s \triangleright \{x : \sigma \rightarrow \tau\} \vdash x : (\sigma \wedge \sigma') \rightarrow \tau$, bo reguła (VAR) nie pozwala na to. Jak się jednak okaże w podrozdziale 3.3, brak reguły podtypowania nam nie przeszkadza: dzięki odpowiednio zdefiniowanym regułom (VAR) i (SUB \perp), w systemie \mathbf{I}_c^s jesteśmy w stanie otypować dokładnie te same termy, co w systemie \mathbf{I}_c , powstałym z \mathbf{I}_c^s przez dodanie reguły podtypowania.

Jak się okazuje, dla \mathbf{I}_c^s daje się udowodnić nieco słabsza od podtypowania własność:

Lemat 3.9 (Częściowa własność podtypowania) *Jeśli $\mathbf{I}_c^s \triangleright A \vdash M : \sigma$ i $\sigma \leq_2 \tau$, gdzie $\tau \in \mathbf{T}_0$, to $\mathbf{I}_c^s \triangleright A \vdash M : \tau$.*

Dowód: Lemat ten wynika bezpośrednio z własności silniejszej:

Jeśli $\mathbf{I}_c^s \triangleright A \vdash M : \tau_1 \rightarrow \dots \rightarrow \tau_n \rightarrow \sigma$ ($n \geq 0$, $\forall i. (\tau_i \in \mathbf{T}_1)$, $\sigma \in \mathbf{T}_0$), oraz $\tau_1 \rightarrow \dots \rightarrow \tau_n \rightarrow \sigma \leq_2 \tau'_1 \rightarrow \dots \rightarrow \tau'_n \rightarrow \sigma'$, przy czym $\forall i. (\tau'_i \in \mathbf{T}_0 \vee \tau'_i = \tau_i)$, $\sigma' \in \mathbf{T}_0$, to $\mathbf{I}_c^s \triangleright A \vdash M : \tau'_1 \rightarrow \dots \rightarrow \tau'_n \rightarrow \sigma'$.

Tę silniejszą własność można udowodnić indukcją ze względu na budowę termu M :

- (i) $M = c^\perp$, $M = c^\top$ — przypadki trywialne.
- (ii) $M = x$. Asercję $\mathbf{I}_c^s \triangleright A \vdash x : \tau_1 \rightarrow \dots \rightarrow \tau_n \rightarrow \sigma$ otrzymano przy pomocy reguły (VAR), a zatem $A(x) = (\bigwedge_{i \in I} \mu_i)$ i dla pewnego $i_0 \in I$: $\mu_{i_0} \leq_0 \tau_1 \rightarrow \dots \rightarrow \tau_n \rightarrow \sigma$ (gdzie $\tau_1, \dots, \tau_n, \sigma \in \mathbf{T}_0$). Jeśli teraz $\tau'_1, \dots, \tau'_n, \sigma'$ będą takie, jak w treści lematu, to $\tau'_1, \dots, \tau'_n, \sigma' \in \mathbf{T}_0$, zatem z definicji \leq_2 zachodzi również $\tau_1 \rightarrow \dots \rightarrow \tau_n \rightarrow \sigma \leq_0 \tau'_1 \rightarrow \dots \rightarrow \tau'_n \rightarrow \sigma'$, a stąd i z przechodności relacji \leq_0 otrzymujemy, że $\mu_{i_0} \leq_0 \tau'_1 \rightarrow \dots \rightarrow \tau'_n \rightarrow \sigma'$. Stąd i z (VAR) otrzymujemy $\mathbf{I}_c^s \triangleright A \vdash x : \tau'_1 \rightarrow \dots \rightarrow \tau'_n \rightarrow \sigma'$.
- (iii) Jeśli $M = \lambda x N$, to $n \geq 1$. Wniosek $\mathbf{I}_c^s \triangleright A \vdash \lambda x N : \tau_1 \rightarrow \dots \rightarrow \tau_n \rightarrow \sigma$ został otrzymany przez (ABS) z przesłanki

$$\mathbf{I}_c^s \triangleright A \cup \{x : \tau_1\} \vdash N : \tau_2 \rightarrow \dots \rightarrow \tau_n \rightarrow \sigma. \quad (3.1)$$

Niech $\tau'_1, \dots, \tau'_n, \sigma'$ będą takie, jak w treści lematu. To wówczas z definicji \leq_2 zachodzi również $\tau_2 \rightarrow \dots \rightarrow \tau_n \rightarrow \sigma \leq_2 \tau'_2 \rightarrow \dots \rightarrow \tau'_n \rightarrow \sigma'$ oraz $\tau'_1 \leq_1 \tau_1$. A zatem (3.1) z założenia indukcyjnego oraz z lematu 3.6 implikuje $\mathbf{I}_c^s \triangleright A \cup \{x : \tau'_1\} \vdash N : \tau'_2 \rightarrow \dots \rightarrow \tau'_n \rightarrow \sigma'$. Stąd zaś przez (ABS) otrzymujemy $\mathbf{I}_c^s \triangleright A \vdash \lambda x N : \tau'_1 \rightarrow \dots \rightarrow \tau'_n \rightarrow \sigma'$.

(iv) $M = M_1M_2$. Wniosek $\mathbf{I}_c^s \triangleright A \vdash M_1M_2 : \tau_1 \rightarrow \dots \rightarrow \tau_n \rightarrow \sigma$ został otrzymany przez (APP) z przesłanek:

$$\mathbf{I}_c^s \triangleright A \vdash M_1 : \left(\bigwedge_{i \in I} \mu_i \right) \rightarrow \tau_1 \rightarrow \dots \rightarrow \tau_n \rightarrow \sigma \quad (3.2)$$

oraz

$$(\forall i \in I) \quad \mathbf{I}_c^s \triangleright A \vdash M_2 : \mu_i. \quad (3.3)$$

Niech $\tau'_1, \dots, \tau'_n, \sigma'$ będą takie, jak w treści lematu. To wówczas (3.2) z definicji \leq_2 oraz z założenia indukcyjnego daje $\mathbf{I}_c^s \triangleright A \vdash M_1 : (\bigwedge_{i \in I} \mu_i) \rightarrow \tau'_1 \rightarrow \dots \rightarrow \tau'_n \rightarrow \sigma'$. A stąd i z (3.3) przez (APP) otrzymujemy $\mathbf{I}_c^s \triangleright A \vdash M_1M_2 : \tau'_1 \rightarrow \dots \rightarrow \tau'_n \rightarrow \sigma'$. ■

Z powyższego lematu wynika bezpośrednio następująca prosta własność:

Lemat 3.10 *Jeśli $\mathbf{I}_c^s \triangleright A \vdash M : (\bigwedge_{i \in I} \tau_i) \rightarrow \sigma$, $(\forall i \in I) A \vdash N : \mu_i$ oraz $(\forall i \in I) \mu_i \leq_2 \tau_i$, to $A \vdash MN : \sigma$.*

Dowód: Jeśli $A \vdash N : \mu_i$, to z lematu o częściowej własności podtypowania otrzymujemy $A \vdash N : \tau_i$, bo $\tau_i \in \mathbf{T}_0$. Końcowy wniosek otrzymujemy przez (ABS). ■

3.2.2 β -redukcja

Najpierw pokażemy, że system \mathbf{I}_c^s jest zamknięty ze względu na β -redukcję (tak zwana własność „subject reduction“ dla systemu \mathbf{I}_c^s).

Twierdzenie 3.11 (Zamkniętość \mathbf{I}_c^s ze względu na β -redukcję)

- (i) *Jeśli $\mathbf{I}_c^s \triangleright A \vdash (\lambda xM)N : \sigma$, to $\mathbf{I}_c^s \triangleright A \vdash \{x := N\}M : \sigma$.*
- (ii) *Jeśli $\mathbf{I}_c^s \triangleright A \vdash M : \sigma$ oraz $M \rightarrow_\beta N$, to $\mathbf{I}_c^s \triangleright A \vdash N : \sigma$.*
- (iii) *Jeśli $\mathbf{I}_c^s \triangleright A \vdash M : \sigma$ oraz $M \rightarrow \rightarrow_\beta N$, to $\mathbf{I}_c^s \triangleright A \vdash N : \sigma$.*

Dowód:

- (i) Asercję $\mathbf{I}_c^s \triangleright A \vdash (\lambda xM)N : \sigma$ otrzymano przy pomocy reguł (APP) i (ABS) z przesłanek

$$\mathbf{I}_c^s \triangleright A_x \cup \{x : (\bigwedge_{i \in I} \tau_i)\} \vdash M : \sigma \quad (3.4)$$

oraz

$$(\forall i \in I) \quad \mathbf{I}_c^s \triangleright A \vdash N : \tau_i. \quad (3.5)$$

Chcemy pokazać, że

$$\mathbf{I}_c^s \triangleright A \vdash \{x := N\}M : \sigma. \quad (3.6)$$

Rozważmy drzewo wyprowadzenia dla (3.4). Gdy w ciele M pojawia się zmienna x , to w celu wyprowadzenia dłań typu najpierw stosujemy regułę (VAR), otrzymując jakiś typ $\tau \in \mathbf{T}_0$ taki, że dla pewnego $i_0 \in I$: $\tau_{i_0} \leq_0 \tau$. Rozważmy teraz term $\{x := N\}M$ z (3.6). Tam, gdzie przed podstawieniem w M występował x , w $\{x := N\}M$ występuje N . No, ale dla tego termu wyprowadzimy na mocy (3.5): $A \vdash N : \tau_{i_0}$, a więc z twierdzenia o słabej własności podtypowania wyprowadzimy także $\mathbf{I}_c^s \triangleright A \vdash N : \tau$. (Podczas wyprowadzania typu dla $\{x := N\}M$ będziemy wprawdzie mieli czasem do czynienia ze środowiskiem innym, niż A , ale na mocy definicji podstawienia tak przemianowaliśmy zmienne związane w M , aby nie kolidowały, a zatem środowisko takie będzie co najwyżej większe od A , a to na mocy lematu 3.4 nie przeszkadza.) Podsumowując — dla $\{x := N\}M$ w A uda nam się wyprowadzić taki sam typ, jak dla M w $A_x \cup \{x : (\bigwedge_{i \in I} \tau_i)\}$. Pozwoliliśmy sobie na bardziej nieformalny dowód na korzyść czytelności jego idei (równie dobrze moglibyśmy przeprowadzić żmudny dowód indukcyjny).

- (ii) Dowód indukcyjny ze względu na budowę termu M z wykorzystaniem części (i) niniejszego lematu.
- (iii) Dowód indukcyjny ze względu na liczbę redukcji z wykorzystaniem części (ii) niniejszego lematu. ■

Własność (iii) z powyższego lematu jest bardzo istotna. Obliczanie naszych programów polega głównie na niedeterministycznym upraszczaniu ich zgodnie z relacją \rightarrow_β . Zamkniętość systemu ze względu na β -redukcję oznacza zatem w szczególności, że jeśli M jest termem posiadającym typ w naszym systemie, to żadna sekwencja β -redukcji zaczynająca się od M nie skończy się błędem wynikającym z aplikacji argumentu o niewłaściwym typie. Innymi słowy, poprawność typowa termów jest zachowywana podczas ich ewaluacji. Gdyby system \mathbf{I}_c^s nie posiadał takiej własności, sens jego rozważania byłby co najmniej dyskusyjny.

Będziemy jeszcze chcieli wykazać, że wszystkie termy typowalne w naszym systemie są silnie normalizowalne (co zapewni, że programy nie będą się pętlić podczas ich obliczania). W tym celu najpierw pokażemy, że jeśli jakiś term M jest typowalny w naszym systemie, to term powstały z termu M wskutek zastąpienia wszystkich wystąpień stałych c^\perp i c^\top przez pewną świeżą zmienną y też jest typowalny.

Lemat 3.12 *Niech $\mathbf{I}_c^s \triangleright A \vdash M : \sigma$. Niech t będzie pewną zmienną typową nie występującą w $\langle A, \sigma \rangle$. Niech y będzie pewną zmienną rachunku lambda nie występującą w M . Niech T będzie funkcją ze stałych typowych w zmienne typowe, przyporządkowującą obu stałym typowym \perp i \top zmienną typową t , którą to funkcję rozszerzamy następnie w naturalny sposób na całe typy \mathbf{T}_0 ,*

\mathbf{T}_1 i \mathbf{T}_2 . Niech L będzie funkcją ze stałych w zmienne rachunku lambda, przyporządkowującą obu stałym c^\perp i c^\top zmienną y , którą to funkcję rozszerzamy następnie w naturalny sposób na całe termy M_λ . To wówczas:

- (i) dla dowolnych $\sigma_1, \sigma_2 \in \mathbf{T}_0$, jeśli $\sigma_1 \leq_0 \sigma_2$, to $T(\sigma_1) \leq_0 T(\sigma_2)$, a nawet $T(\sigma_1) = T(\sigma_2)$, oraz
- (ii) zachodzi $\mathbf{I}_c^s \triangleright TA \cup \{y : t\} \vdash L(M) : T\sigma$, i ponadto jeśli w celu wyprowadzenia tej asercji skorzystano z faktu, że $\tau_1 \leq_0 \tau_2$ dla pewnych $\tau_1, \tau_2 \in \mathbf{T}_0$, to zachodziło $\tau_1 = \tau_2$.

Dowód: (i) – wynika bezpośrednio z definicji \leq_0 , (ii) – dowód indukcyjny ze względu na budowę termu M przebiega identycznie, jak dowód lematu o substytucyjności dla \mathbf{I}_c^s , tyle, że w przypadku $M = x$ zamiast z lematu 3.1 korzystamy z części (i) niniejszego lematu. ■

Teraz możemy już udowodnić własność, o której wspominaliśmy.

Twierdzenie 3.13 *Jeśli $\mathbf{I}_c^s \triangleright A \vdash M : \sigma$, to M jest silnie normalizowalny.*

Dowód: Dowód nie wprost. Niech M nie będzie silnie normalizowalny, czyli istnieje pewna nieskończona sekwencja β -redukcji startująca z M . Wówczas term M' , powstały przez zastąpienie stałych w M pewną świeżą zmienną y , też będzie typowalny w naszym systemie na mocy lematu 3.12 — dokładniej, będzie istniała taka para $\langle A', \sigma' \rangle$ nie zawierająca stałych, że $\mathbf{I}_c^s \triangleright A' \vdash M' : \sigma'$, oraz jeśli w celu wyprowadzenia tej asercji skorzystano z faktu, że $\tau_1 \leq_0 \tau_2$ dla pewnych $\tau_1, \tau_2 \in \mathbf{T}_0$, to zachodziło $\tau_1 = \tau_2$. Stąd wniosek, że term M' będzie także typowalny w systemie typów intersekcyjnych \vdash_\wedge o nieograniczonym rzędzie, opisanym między innymi w [3]. Ponadto, M' też nie będzie silnie normalizowalny — bo w M' będzie istniała ta sama nieskończona sekwencja β -redukcji, co w M (stałe w przypadku β -redukcji nie odgrywają roli, bo nie wiążą). No, ale z drugiej strony — udowodniono, że każdy term M' typowalny w \vdash_\wedge jest silnie normalizowalny (autorzy [3] odsyłają między innymi do [8]). A zatem otrzymaliśmy sprzeczność. ■

3.2.3 Typowość w systemie \mathbf{I}_c^s

W tym podrozdziale sprowadzimy problem typowości termów w systemie \mathbf{I}_c^s do problemu istnienia rozwiązania pewnego układu równań i nierówności podtypowych. Oto formalizacja tego drugiego problemu:

Definicja 3.14

- (i) Układ równań i $\leq_{2,0}$ -nierówności to zbiór, którego każdy element to albo równanie pomiędzy typami prostymi, albo nierówność \leq_2 pomiędzy typem z \mathbf{T}_2 i typem z \mathbf{T}_0 .

- (ii) Podstawienie S jest rozwiązaniem układu równań i $\leq_{2,0}$ -nierówności C , jeśli $S\sigma \leq_2 S\tau$ dla wszystkich nierówności $(\sigma \leq_2 \tau) \in C$ oraz $S\sigma = S\tau$ dla wszystkich równań $(\sigma = \tau) \in C$. O takim podstawieniu mówimy również, że spełnia ono ten układ. ■

W dalszej części tego rozdziału będziemy pisać „układ“, mając na myśli układ równań i $\leq_{2,0}$ -nierówności.

Dla danego termu, odpowiadający mu układ będziemy budować indukcyjnie. Dokładniej, każdemu termowi będziemy przyporządkowywać zbiór trójek (środowisko \mathbf{T}_1 , typ \mathbf{T}_2 , układ równań i $\leq_{2,0}$ -nierówności), nazywany zbiorem PC. Nazwa ta, wymyślona na potrzeby niniejszej pracy, jest angielskim skrótem od „par i warunków“ i oddaje intuicyjny sens takiego zbioru: dla danego termu M , jeśli $\langle A, \sigma, C \rangle \in PC(M)$, to oznacza to, że $A \vdash M : \sigma$, o ile na zmienne z A i z σ będziemy podstawiali wartości spełniające zbiór C (będzie o tym mowa w twierdzeniu 3.18). Co więcej, podstawiając na zmienne z A i z σ wszystkie możliwe rozwiązania zbioru C , otrzymamy wprawdzie nie wszystkie możliwe typowania termu M , to jednak wszystkie w pewnym sensie „najlepsze“ typowania termu M (będzie o tym mowa w twierdzeniu 3.20 oraz w podrozdziale 3.4).

Definicja 3.15 Dla dowolnego termu M definiujemy zbiór $PC(M)$ trójek (środowisko \mathbf{T}_1 , typ \mathbf{T}_2 , układ równań i $\leq_{2,0}$ -nierówności) indukcyjnie:

- (i) Jeśli $M = x$, to dla każdej zmiennej typowej t , $\langle \{x : t\}, t, \{\} \rangle \in PC(M)$.
- (ii) a) Jeśli $M = c^\perp$, to $\langle \{\}, \perp, \{\} \rangle \in PC(M)$.
 b) Jeśli $M = c^\top$, to $\langle \{\}, \top, \{\} \rangle \in PC(M)$.
- (iii) Jeśli $M = \lambda x N$ i $\langle A, \sigma, C \rangle \in PC(N)$, to:
- a) jeśli $x \notin \mathbf{dom}(A)$, zaś t jest zmienną typową nie występującą w $\langle A, \sigma \rangle$, to $\langle A, t \rightarrow \sigma, C \rangle \in PC(M)$.
 b) jeśli $x \in \mathbf{dom}(A)$, to $\langle A_x, A(x) \rightarrow \sigma, C \rangle \in PC(M)$.
- (iv) Jeśli $M = M_1 M_2$, to:
- a) jeśli $\langle A_1, t, C_1 \rangle \in PC(M_1)$ i $\langle A_2, \sigma_2, C_2 \rangle \in PC(M_2)$ to rozłączne (w sensie występowania tych samych zmiennych typowych) trójki, to niech $D = \{t = t_1 \rightarrow t_2; \sigma_2 \leq_2 t_1\}$, gdzie t_1, t_2 — nowe zmienne. Wówczas $\langle A_1 + A_2, t_2, C_1 \cup C_2 \cup D \rangle \in PC(M)$.
- b) jeśli $\langle A_1, (\bigwedge_{i \in I} \sigma_i) \rightarrow \sigma_1, C_1 \rangle \in PC(M_1)$ i dla wszystkich $i \in I$ $\langle A_i, \tau_i, C_i \rangle \in PC(M_2)$, gdzie wszystkie trójki są rozłączne, to niech $D = \{\tau_i \leq_2 \sigma_i \mid i \in I\}$. Wówczas $\langle A_1 + \sum_{i \in I} A_i, \sigma_1, C_1 \cup \bigcup_{i \in I} C_i \cup D \rangle \in PC(M)$. ■

Nie przypadkowo powyższa definicja przypomina definicję par $\text{PP}_{\mathbf{I}_S^s}$ z pracy Jima [6] oraz definicję par PP_R z pracy van Bakela [10]. W przypadku braku podtypów, układy, które kumulujemy w trójkach PC, redukują się do układów, które można rozwiązywać na bieżąco — tak właśnie czynią van Bakel i Jim. Nasze trójki PC są zatem rozszerzeniem par ze wspomnianych prac — stąd liczne podobieństwa zarówno w ich definicji, jak i w dowodach o ich poprawności i pełności. W podrozdziale 3.4 wyjaśni się, dlaczego nie możemy rozwiązywać układów na bieżąco, tylko musimy je kumulować.

Dla danego termu M , zbiór $\text{PC}(M)$ zawiera wprawdzie wiele elementów, lecz tak naprawdę wszystkie te elementy różnią się jedynie nazwami zmiennych typowych. Mówi o tym poniższy lemat.

Lemat 3.16 *Niech $\langle A_1, \sigma_1, C_1 \rangle \in \text{PC}(M)$. To $\langle A_2, \sigma_2, C_2 \rangle \in \text{PC}(M)$ wtedy i tylko wtedy, gdy istnieje bijekcja R ze zmiennych typowych w zmienne typowe taka, że $R\langle A_1, \sigma_1, C_1 \rangle = \langle A_2, \sigma_2, C_2 \rangle$.*

Dowód: Dowód indukcyjny na podstawie definicji PC. ■

Poniższy pomocniczy lemat wyraża fakt, że w konstruowanych trójkach PC środowiska nie zawierają par (zmienna, typ) odpowiadających niepotrzebnym zmiennym. Lemat ten, podobnie jak i poprzedni, okaże się pomocny w dowodach bardziej skomplikowanych własności.

Lemat 3.17 *Jeśli $\langle A, \sigma, C \rangle \in \text{PC}(M)$, to $x \in \text{dom}(A)$ wtedy i tylko wtedy, gdy x jest zmienną wolną w termie M .*

Dowód: Dowód indukcyjny na podstawie definicji PC. ■

Pora na twierdzenie wyrażające pierwszą ze wspomnianych wcześniej nieformalnie własności trójek: dla danego termu M , jeśli $\langle A, \sigma, C \rangle \in \text{PC}(M)$, to oznacza to, że $A \vdash M : \sigma$, o ile na zmienne z A i z σ będziemy podstawiali wartości spełniające zbiór C .

Twierdzenie 3.18 *Jeśli $\langle A, \sigma, C \rangle \in \text{PC}(M)$, zaś S jest rozwiązaniem układu C , to $\mathbf{I}_C^s \triangleright SA \vdash M : S\sigma$.*

Dowód: Dowód indukcyjny względem budowy termu M :

- (i) Jeśli $M = x$, to $\langle A, \sigma, C \rangle = \langle \{x : t\}, t, \{\} \rangle$, a mamy $\mathbf{I}_C^s \triangleright \{x : t\} \vdash x : t$ z (VAR), oraz z substytutuwności $\mathbf{I}_C^s \triangleright S\{x : t\} \vdash x : St$ dla dowolnego podstawienia S .
- (ii) Jeśli $M = c^\perp$, to $\langle A, \sigma, C \rangle = \langle \{\}, \perp, \{\} \rangle$, a mamy $\mathbf{I}_C^s \triangleright \{\} \vdash c^\perp : \perp$ z (CONST \perp), oraz z substytutuwności $\mathbf{I}_C^s \triangleright S\{\} \vdash c^\perp : S\perp$ dla dowolnego podstawienia S .

Dla $M = c^\top$ dowód analogiczny.

(iii) Jeśli $M = \lambda xN$, to z lematu 3.17 mamy następujące dwa przypadki:

- a) x nie jest wolne w N . To $\sigma = t \rightarrow \sigma'$, gdzie $\langle A, \sigma', C \rangle \in \text{PC}(N)$, t — nowa zmienna. Niech S — rozwiązanie układu C ; wówczas przez indukcję mamy $\mathbf{I}_c^s \triangleright SA \vdash N : S\sigma'$, a stąd przez osłabianie $\mathbf{I}_c^s \triangleright SA \cup \{x : St\} \vdash N : S\sigma'$. Stąd przez (ABS) otrzymujemy $\mathbf{I}_c^s \triangleright SA \vdash \lambda xN : St \rightarrow S\sigma' = S(t \rightarrow \sigma')$.
- b) x jest wolne w N . To $\langle A, \sigma, C \rangle = \langle A'_x, A'(x) \rightarrow \sigma', C \rangle$, gdzie $\langle A', \sigma', C \rangle \in \text{PC}(N)$. Niech S będzie rozwiązaniem układu C ; wówczas przez indukcję mamy $\mathbf{I}_c^s \triangleright SA' \vdash N : S\sigma'$. Stąd przez (ABS) otrzymujemy $\mathbf{I}_c^s \triangleright SA'_x \vdash \lambda xN : SA'(x) \rightarrow S\sigma'$, a to jest to samo, co $\mathbf{I}_c^s \triangleright SA \vdash \lambda xN : S\sigma$.

(iv) Jeśli $M = M_1M_2$, to zachodzi jeden z przypadków:

- a) $\langle A, \sigma, C \rangle = \langle A_1 + A_2, t_2, C_1 \cup C_2 \cup D \rangle$, gdzie $\langle A_1, t, C_1 \rangle \in \text{PC}(M_1)$, $\langle A_2, \sigma_2, C_2 \rangle \in \text{PC}(M_2)$, $D = \{t = t_1 \rightarrow t_2; \sigma_2 \leq_2 t_1\}$, t_1, t_2 — nowe zmienne. Niech S będzie rozwiązaniem C . To (z definicji rozwiązania) S będzie również rozwiązaniem C_1 , rozwiązaniem C_2 i rozwiązaniem D . Zatem, z założenia indukcyjnego dla M_1 i przez osłabianie, mamy:

$$\mathbf{I}_c^s \triangleright S(A_1 + A_2) \vdash M_1 : St = S(t_1 \rightarrow t_2) = St_1 \rightarrow St_2. \quad (3.7)$$

Ponadto, z założenia indukcyjnego dla M_2 i przez osłabianie, mamy:

$$\mathbf{I}_c^s \triangleright S(A_1 + A_2) \vdash M_2 : S\sigma_2. \quad (3.8)$$

Ponieważ S jest rozwiązaniem D , to $S\sigma_2 \leq_2 St_1$. Zatem, na mocy lematu o częściowej własności podtypowania, z (3.8) otrzymujemy $\mathbf{I}_c^s \triangleright S(A_1 + A_2) \vdash M_2 : St_1$. To, w połączeniu z (3.7), przez (ABS) daje $\mathbf{I}_c^s \triangleright S(A_1 + A_2) \vdash M_1M_2 : St_2$.

- b) $\langle A, \sigma, C \rangle = \langle A_1 + \sum_{i \in I} A_i, \sigma_1, C_1 \cup \bigcup_{i \in I} C_i \cup D \rangle$, gdzie $\langle A_1, (\bigwedge_{i \in I} \sigma_i) \rightarrow \sigma_1, C_1 \rangle \in \text{PC}(M_1)$ i dla wszystkich $i \in I$: $\langle A_i, \tau_i, C_i \rangle \in \text{PC}(M_2)$, $D = \{\tau_i \leq_2 \sigma_i \mid i \in I\}$. Niech S będzie rozwiązaniem C . To (z definicji rozwiązania) S będzie również rozwiązaniem C_1 , C_i (dla każdego $i \in I$) i D . Zatem, z założenia indukcyjnego dla M_1 i przez osłabianie, mamy:

$$\mathbf{I}_c^s \triangleright S(A_1 + \sum_{i \in I} A_i) \vdash M_1 : S((\bigwedge_{i \in I} \sigma_i) \rightarrow \sigma_1) = (\bigwedge_{i \in I} S\sigma_i) \rightarrow S\sigma_1. \quad (3.9)$$

Ponadto, z założenia indukcyjnego dla M_2 i przez osłabianie, mamy:

$$(\forall i \in I) \quad \mathbf{I}_c^s \triangleright S(A_1 + \sum_{i \in I} A_i) \vdash M_2 : S\tau_i. \quad (3.10)$$

Ponieważ S jest rozwiązaniem D , to dla wszystkich $i \in I$: $S\tau_i \leq_2 S\sigma_i$. Dzięki temu faktowi, (3.9) i (3.10) na mocy lematu 3.10 dają, że $S(A_1 + \sum_{i \in I} A_i) \vdash M_1 M_2 : S\sigma_1$. ■

Inna własność, o której wspomnieliśmy nieformalnie wcześniej, jest taka, że jeśli $\langle A, \sigma, C \rangle \in \text{PC}(M)$, to podstawiając na zmienne z A i z σ wszystkie możliwe rozwiązania zbioru C , otrzymamy wprawdzie nie wszystkie możliwe typowania termu M , to jednak wszystkie w pewnym sensie „najlepsze“ typowania termu M . Poniżej wprowadzamy zatem definicje służące formalizacji pojęcia jakości typowań.

Definicja 3.19

(i) Niech A, A' - środowiska typu \mathbf{T}_1 . $A \leq_1 A'$, jeśli dla każdego $x \in \text{dom}(A')$: $x \in \text{dom}(A)$ i $A(x) \leq_1 A'(x)$.

(ii) Porządek na parach $\langle \text{środowisko } \mathbf{T}_1, \text{typ } \mathbf{T}_2 \rangle$ definiujemy następująco:

$\langle A, \sigma \rangle \leq \langle A', \sigma' \rangle$ wtedy i tylko wtedy, gdy $A' \leq_1 A$ i $\sigma \leq_2 \sigma'$. ■

Zwróćmy uwagę, że porządek \leq_1 dla środowisk jest naturalnym rozszerzeniem porządku \leq_1 dla typów. Jeśli $A \leq_1 B$, to w środowisku A założenia o wszystkich zmiennych są silniejsze, niż w B — w sensie porządku \leq_1 na typach przyporządkowanych zmiennym w tych środowiskach. Porządek \leq dla par wyraża natomiast ideę porównywania typowań: pewne typowanie jest silniejsze, a więc „lepsze“, od innego, jeśli ze słabszych założeń w środowisku pozwala wyprowadzić silniejszy typ. (Tak naprawdę w powyższych zdaniach zamiast „silniejszy“ i „lepszy“ powinniśmy byli napisać „nie słabszy“ i „nie gorszy“, jednak przyjęte słownictwo sprzyja przejrzystości wyводу.)

Zauważmy, że relacje \leq i \leq_1 są przechodnie, oraz, że $A + A' \leq A$ dla dowolnych środowisk A, A' typu \mathbf{T}_1 . Zauważmy również, że jeśli $\langle A, \sigma \rangle \leq \langle A', \sigma' \rangle$, to dla dowolnego podstawienia S : $S\langle A, \sigma \rangle \leq S\langle A', \sigma' \rangle$. Wynika to z lematu 3.1.

Teraz możemy już przystąpić do udowodnienia własności „pełności“ zbiorów PC.

Twierdzenie 3.20 *Jeśli $\mathbf{I}_C^S \triangleright A \vdash M : \sigma$ i $\langle A', \sigma', C \rangle \in \text{PC}(M)$, to istnieje rozwiązanie S układu C takie, że $S\langle A', \sigma' \rangle \leq \langle A, \sigma \rangle$.*

Dowód: Dowód indukcyjny ze względu na budowę termu M . Rozpatrujemy następujące przypadki:

(i) $M = x$. To $A \vdash M : \sigma$ przez (VAR), czyli $A(x) = \bigwedge_{i \in I} \sigma_i$ i $\sigma_{i_0} \leq_0 \sigma$ dla pewnego $i_0 \in I$. Dla dowolnego t , $\langle \{x : t\}, t, \{\} \rangle \in \text{PC}(M)$. To $\{t := \sigma\}$ jest rozwiązaniem pustego w tym przypadku układu i ponieważ $A_x \cup \{x : \bigwedge_{i \in I} \sigma_i\} \leq \{x : \sigma\}$, gdzie dla pewnego $i \in I$, $\sigma_i \leq_0 \sigma$, to mamy $\{t := \sigma\} \langle \{x : t\}, t \rangle = \langle \{x : \sigma\}, \sigma \rangle \leq \langle A, \sigma \rangle$.

- (ii) $M = c^\top$. To $\mathbf{I}_c^s \triangleright A \vdash M : \sigma$ przy pomocy reguły (CONST \top), czyli $\sigma = \top$. No, i $\langle \{\}, \top, \{\} \rangle \in \text{PC}(M)$. Identyczność jest rozwiązaniem pustego w tym przypadku układu, a że $\langle \{\}, \top \rangle \leq \langle A, \top \rangle$, to mamy to, co chcieliśmy udowodnić.

Jeśli $M = c^\perp$, to mamy dwa przypadki: albo $\sigma = \perp$ przez (CONST \perp), albo $\sigma = \top$ przez (SUB \perp). W obu przypadkach dowód analogiczny, jak dla $M = c^\top$.

- (iii) $M = \lambda x N$. Wtedy σ musi być postaci $\sigma_1 \rightarrow \sigma_2$ i $\mathbf{I}_c^s \triangleright A_x \cup \{x : \sigma_1\} \vdash N : \sigma_2$. Przez indukcję, jeśli $\langle A', \sigma'_2, C \rangle \in \text{PC}(N)$ to istnieje rozwiązanie S układu C takie, że

$$S\langle A', \sigma'_2 \rangle \leq \langle A_x \cup \{x : \sigma_1\}, \sigma_2 \rangle. \quad (3.11)$$

Rozważamy dwa przypadki:

- a) Jeśli $x \notin \mathbf{dom}(A')$, to dla dowolnej nowej zmiennej t , $\langle A', t \rightarrow \sigma'_2, C \rangle \in \text{PC}(\lambda x N)$. σ_1 jest postaci $\bigwedge_{i \in I} \sigma_i$, można więc znaleźć $\sigma'_1 \in \mathbf{T}_0$ takie, że $\sigma_1 \leq_2 \sigma'_1$ (po prostu wybierzmy dowolne σ_i). Wówczas niech $S' = S \cup \{t := \sigma'_1\}$ (można przyjąć, że S nie działało na t , bo zmienna t nie występuje w C). S' jest również rozwiązaniem C (bo S - rozwiązanie C , i w C nie występuje t). Ponieważ z (3.11) na podstawie definicji \leq mamy $A_x \cup \{x : \sigma_1\} \leq_1 SA'$ i $x \notin \mathbf{dom}(A')$, to z definicji \leq_1 mamy także $A_x \leq_1 SA'$. Stąd, z (3.11) i z definicji \leq : $S'\langle A', t \rightarrow \sigma'_2 \rangle = \langle SA', \sigma'_1 \rightarrow S\sigma'_2 \rangle \leq \langle A_x, \sigma_1 \rightarrow \sigma_2 \rangle$.

Ponieważ $A \leq_1 A_x$, to powyższe daje ostatecznie, że $S'\langle A', t \rightarrow \sigma'_2 \rangle \leq \langle A, \sigma_1 \rightarrow \sigma_2 \rangle$.

- b) Jeśli $x \in \mathbf{dom}(A')$. To $\langle A'_x, A'(x) \rightarrow \sigma'_2, C \rangle \in \text{PC}(\lambda x N)$. Ponieważ z (3.11) na podstawie definicji \leq mamy $A_x \cup \{x : \sigma_1\} \leq_1 SA'$, to z definicji \leq_1 mamy także $A_x \leq_1 SA'_x$. Stąd, z (3.11) i z definicji \leq : $S\langle A'_x, A'(x) \rightarrow \sigma'_2 \rangle \leq \langle A_x, \sigma_1 \rightarrow \sigma_2 \rangle$.

Ponieważ $A \leq A_x$, to powyższe daje ostatecznie, że $S\langle A'_x, A'(x) \rightarrow \sigma'_2 \rangle \leq \langle A, \sigma_1 \rightarrow \sigma_2 \rangle$.

- (iv) $M = M_1 M_2$. Wówczas $\mathbf{I}_c^s \triangleright A \vdash M_1 : (\bigwedge_{i \in I} \sigma_i) \rightarrow \sigma$ i dla wszystkich $i \in I$: $A \vdash M_2 : \sigma_i$. Na mocy lematu 3.16 wystarczy rozważyć następujące przypadki struktury trójek w $\text{PC}(M_1)$:

- a) $\langle A_1, t, C_1 \rangle \in \text{PC}(M_1)$. Przez indukcję, istnieje podstawienie S_1 będące rozwiązaniem C_1 , takie, że:

$$S_1\langle A_1, t \rangle \leq \langle A, (\bigwedge_{i \in I} \sigma_i) \rightarrow \sigma \rangle.$$

Aby powyższa nierówność zachodziła, z definicji \leq musi zachodzić $S_1 t = \mu \rightarrow \sigma'$ i $\bigwedge_{i \in I} \sigma_i \leq_1 \mu$ i $\sigma' \leq_2 \sigma$, gdzie $\mu, \sigma' \in \mathbf{T}_0$. Z definicji

\leq_1 i \leq_2 , skoro $\bigwedge_{i \in I} \sigma_i \leq_1 \mu$, to dla pewnego $i \in I$ zachodzi $\sigma_i \leq_0 \mu$, a więc i $\sigma_i \leq_2 \mu$.

Niech $\langle A_2, \tau, C_2 \rangle \in \text{PC}(M_2)$ będzie trójką rozłączną względem $\langle A_1, t, C_1 \rangle$. Przez indukcję, istnieje podstawienie S_2 będące rozwiązaniem C_2 takie, że:

$$S_2 \langle A_2, \tau \rangle \leq \langle A, \sigma_i \rangle.$$

Niech, zgodnie z definicją PC dla aplikacji, $D = \{t = t_1 \rightarrow t_2; \tau \leq_2 t_1\}$, gdzie t_1, t_2 — nowe zmienne. Można przyjąć, że $\text{dom}(S_1) \cap \text{dom}(S_2) = \emptyset$, bo zbiory zmiennych z C_1 i z C_2 są rozłączne, oraz, że S_1, S_2 nie działają na t_1, t_2 — bo te zmienne, jako świeże, nie występują ani w C_1 , ani w C_2 . Wówczas $S = S_1 \cup S_2 \cup \{t_1 := \mu; t_2 := \sigma'\}$ jest rozwiązaniem $C_1 \cup C_2 \cup D$, bo: skoro S_1 jest rozwiązaniem C_1 , to S też; skoro S_2 jest rozwiązaniem C_2 , to S też; ponadto S jest rozwiązaniem D , bo $St = S_1 t = \mu \rightarrow \sigma' = St_1 \rightarrow St_2 = S(t_1 \rightarrow t_2)$, oraz $S\tau = S_2 \tau \leq_2 \sigma_i \leq_2 \mu = St_1$.

No, i zachodzi $S \langle A_1 + A_2, t_2 \rangle = \langle S_1 A_1 + S_2 A_2, \sigma' \rangle \leq \langle A, \sigma \rangle$.

- b) $\langle A_1, (\bigwedge_{j \in J} \sigma'_j) \rightarrow \sigma', C_1 \rangle \in \text{PC}(M_1)$. Przez indukcję, istnieje podstawienie S_1 , będące rozwiązaniem C_1 , takie, że:

$$S_1 \langle A_1, (\bigwedge_{j \in J} \sigma'_j) \rightarrow \sigma' \rangle \leq \langle A, (\bigwedge_{i \in I} \sigma_i) \rightarrow \sigma \rangle.$$

Aby powyższa nierówność zachodziła, z definicji \leq musi zachodzić $(\bigwedge_{j \in J} S_1 \sigma'_j) \rightarrow S_1 \sigma' \leq_2 (\bigwedge_{i \in I} \sigma_i) \rightarrow \sigma$, a stąd z definicji \leq_2 : $S_1 \sigma' \leq_2 \sigma$ i $\bigwedge_{i \in I} \sigma_i \leq_1 \bigwedge_{j \in J} S_1 \sigma'_j$. To ostatnie, z definicji \leq_1 , wyraża fakt, że dla każdego $j \in J$ istnieje $i_j \in I$ takie, że $\sigma_{i_j} \leq_0 S_1 \sigma'_j$, a więc i $\sigma_{i_j} \leq_2 S_1 \sigma'_j$.

Niech dla wszystkich $j \in J$: $\langle A_j, \rho_j, C_j \rangle \in \text{PC}(M_2)$ będą trójkami rozłącznymi względem siebie i względem $\langle A_1, t, C_1 \rangle$. Przez indukcję, istnieją podstawienia S_j będące rozwiązaniami C_j takie, że:

$$S_j \langle A_j, \rho_j \rangle \leq \langle A, \rho_{i_j} \rangle.$$

Niech, zgodnie z definicją PC dla aplikacji, $D = \{\rho_j \leq_2 \sigma'_j \mid j \in J\}$. Można przyjąć, że dziedziny wszystkich rozważanych podstawień (S_1, S_j dla $j \in J$) są parami rozłączne, bo zbiory zmiennych pojawiających się w układach, których te podstawienia są rozwiązaniami, są rozłączne. Wówczas $S = S_1 \cup \bigcup_{j \in J} S_j$ jest rozwiązaniem $C_1 \cup \bigcup_{j \in J} C_j \cup D$, bo: skoro S_1 jest rozwiązaniem C_1 , to S też; skoro S_j jest rozwiązaniem C_j , to S też; ponadto S jest rozwiązaniem D , bo: $S\rho_j = S_j \rho_j \leq_2 \sigma_{i_j} \leq_2 S_1 \sigma'_j$.

No, i zachodzi $S \langle A_1 + \sum_{j \in J} A_j, \sigma' \rangle = \langle S_1 A_1 + \sum_{j \in J} S_j A_j, S_1 \sigma' \rangle \leq \langle A, \sigma \rangle$. ■

$$\begin{array}{ll}
(\sigma_1 \rightarrow \sigma_2) \leq_{2,1} t & \Rightarrow \{t_1 \leq_{2,1} \sigma_1, \sigma_2 \leq_{2,1} t_2, t = t_1 \rightarrow t_2\} \\
(\sigma_1 \rightarrow \sigma_2) \leq_{2,1} (\tau_1 \rightarrow \tau_2) & \Rightarrow \{\tau_1 \leq_{2,1} \sigma_1, \sigma_2 \leq_{2,1} \tau_2\} \\
\sigma \leq_{2,1} (\tau_1 \wedge \tau_2) & \Rightarrow \{\sigma \leq_{2,1} \tau_1, \sigma \leq_{2,1} \tau_2\}
\end{array}$$

Rysunek 3.2: Reguły transformacji układów równań i $\leq_{2,1}$ -nierówności podtypowych. Zmienne występujące z prawej strony reguły, a nie występujące z lewej, są świeże.

Reszta lematów i twierdzeń nie wymaga komentarza i zmierza do pokazania, że problem typowości termu w \mathbf{I}_c^s jest rozstrzygalny.

Lemat 3.21 *Istnieje algorytm znajdujący dla danego termu M element zbioru $PC(M)$.*

Dowód: Należy postępować zgodnie z definicją $PC(M)$. Lemat 3.16 wyraża fakt, że dla każdego termu M , wszystkie elementy $PC(M)$ mają taką samą postać, z dokładnością do nazw zmiennych. ■

Twierdzenie 3.22 *Istnieje algorytm stwierdzający, czy dany układ równań i $\leq_{2,0}$ -nierówności ma rozwiązanie, i jeśli odpowiedź jest twierdząca, to zwracający jakieś rozwiązanie.*

Dowód: Układ równań i $\leq_{2,0}$ -nierówności podtypowych jest szczególnym przypadkiem układu równań i $\leq_{2,1}$ -nierówności podtypowych, gdzie układ równań i $\leq_{2,1}$ -nierówności podtypowych jest zdefiniowany analogicznie, jak układ równań i $\leq_{2,0}$ -nierówności podtypowych, z tym, że znajdujące się w nim nierówności są pomiędzy typami z \mathbf{T}_2 i z \mathbf{T}_1 , a relacja $\leq_{2,1}$ pomiędzy typami z \mathbf{T}_2 i z \mathbf{T}_1 jest zdefiniowana w sposób następujący:

$$\text{„Jeśli } \sigma \leq_2 \tau_i \text{ dla wszystkich } i \in I, \text{ to } \sigma \leq_{2,1} (\bigwedge_{i \in I} \tau_i)\text{.”}$$

Na układy równań i $\leq_{2,1}$ -nierówności podtypowych przenosimy wszystkie pojęcia zdefiniowane dla układów równań i $\leq_{2,0}$ -nierówności podtypowych.

Układ równań i $\leq_{2,1}$ -nierówności podtypowych, który ma rozwiązania, daje się równoważnie sprowadzić do układu, w którym żaden typ nie zawiera operatora intersekcji \wedge , czyli — innymi słowy — każdy typ jest z \mathbf{T}_0 . W tym celu należy zastosować reguły podane na rysunku 3.2. Reguły te są trochę podobne do reguł zaproponowanych przez Jima w pracy [6] do rozwiązywania analogicznego problemu dla systemu typów intersekcyjnych bez stałych. Zauważmy, że reguły te są poprawne: po pierwsze, przekształcają układ równań i $\leq_{2,1}$ -nierówności w układ równań i $\leq_{2,1}$ -nierówności i umożliwiają

stopniową eliminację wszystkich operatorów intersekcji \wedge . Po drugie, rozwiązanie układu po transformacji jest również rozwiązaniem układu przed transformacją. Odwrotna zależność nie zachodzi, bo wprowadzamy nowe zmienne, ale zauważmy, że jeśli pierwotny układ nie miał rozwiązań, to nie będzie ich miał także układ po transformacji.

Po wyeliminowaniu operatorów intersekcji przy pomocy podanych reguł otrzymamy układ zawierający wyłącznie typy z \mathbf{T}_0 , a dla takich układów istnieją algorytmy sprawdzające spełnialność i ewentualnie znajdujące jakieś rozwiązanie (jest to tak zwany problem SSI, omawiany między innymi w pracy [9]). ■

W dalszym ciągu rozdziału zapomnimy o układach równań i $\leq_{2,1}$ -nierówności podtypowych i pisząc „układ“ znowu będziemy mieli na myśli układ równań i $\leq_{2,0}$ -nierówności podtypowych.

Twierdzenie 3.23 *Problem typowości w systemie \mathbf{I}_c^s jest rozstrzygalny. Jeśli term M jest typowalny w \mathbf{I}_c^s , to można łatwo znaleźć takie A, σ , że $\mathbf{I}_c^s \triangleright A \vdash M : \sigma$.*

Dowód: Twierdzenie 3.18 mówi, że jeśli $\langle A, \sigma, C \rangle \in \text{PC}(M)$ i S jest rozwiązaniem układu C , to $SA \vdash M : S\sigma$, czyli M jest typowalny. Twierdzenie 3.20 mówi, że jeśli M jest typowalny i $\langle A, \sigma, C \rangle \in \text{PC}(M)$, to istnieje rozwiązanie S układu C . A zatem problem typowości termu M w \mathbf{I}_c^s jest równoważny problemowi istnienia pewnego podstawienia S będącego rozwiązaniem układu C z trójki $\langle A, \sigma, C \rangle \in \text{PC}(M)$. Wystarczy więc znaleźć dowolną taką trójkę, co jest możliwe na mocy lematu 3.21, a następnie sprawdzić, czy C ma rozwiązanie, co z kolei jest możliwe na mocy twierdzenia 3.22.

Twierdzenie 3.22 mówi również, że można znaleźć jakieś rozwiązanie S układu C (o ile takowe istnieje), a wtedy na mocy twierdzenia 3.18 mamy $SA \vdash M : S\sigma$, czyli uzyskujemy przykładowe typowanie termu M . ■

3.3 System \mathbf{I}_c

System \mathbf{I}_c powstaje z systemu \mathbf{I}_c^s przez wprowadzenie reguły (SUB), zastąpienie nierówności \leq_0 równością w regule (VAR) oraz usunięcie reguły (SUB \perp).

Definicja 3.24 *Asercje w systemie \mathbf{I}_c są zdefiniowane indukcyjnie regułami przedstawionymi na rysunku 3.3. Piszemy $\mathbf{I}_c \triangleright A \vdash M : \sigma$, jeśli asercja $A \vdash M : \sigma$ została otrzymana przy pomocy reguł systemu \mathbf{I}_c . ■*

System \mathbf{I}_c jest rozszerzeniem systemu \mathbf{I}_2 , omawianego przez Jima w pracy [6]. W rzeczy samej, definicje obu systemów wyglądają identycznie — tyle, że u nas porządek \leq_2 jest rozszerzony w stosunku do porządku \leq_2 u Jima, który nie rozważa podtypów. Takie podobieństwo obu systemów wydaje

(CONST \perp)	$A \vdash c^\perp : \perp$
(CONST \top)	$A \vdash c^\top : \top$
(VAR)	$A \cup \{x : (\bigwedge_{i \in I} \tau_i)\} \vdash x : \tau_{i_0}$ (gdzie $i_0 \in I$)
(ABS)	$\frac{A_x \cup \{x : \sigma\} \vdash M : \tau}{A \vdash (\lambda x M) : \sigma \rightarrow \tau}$
(APP)	$\frac{A \vdash M : (\bigwedge_{i \in I} \tau_i) \rightarrow \sigma, \quad (\forall i \in I) A \vdash N : \tau_i}{A \vdash (MN) : \sigma}$
(SUB)	$\frac{A \vdash M : \tau}{A \vdash M : \sigma}$ (gdzie $\tau \leq_2 \sigma$)

Rysunek 3.3: Reguły \mathbf{I}_c . Środowiska są typu \mathbf{T}_1 , wyprowadzone typy należą do \mathbf{T}_2 .

się dobitnie świadczyć o tym, że systemy intersekcyjne szczególnie łatwo rozszerzają się na języki z podtypami.

Jak łatwo przewidzieć, system \mathbf{I}_c jest co najmniej tak silny, jak system \mathbf{I}_c^s . Na pytanie, czy wprowadzenie do \mathbf{I}_c^s ogólnej reguły podtypowania rzeczywiście coś zmienia, odpowiedzieliśmy już w części poświęconej podstawowym własnościom systemu \mathbf{I}_c^s . Dla porządku sformalizujmy tu jednak oba te wnioski.

Twierdzenie 3.25 *Jeśli $\mathbf{I}_c^s \triangleright A \vdash M : \sigma$, to $\mathbf{I}_c \triangleright A \vdash M : \sigma$. Odwrotna zależność nie zachodzi.*

Dowód: Najpierw zauważmy, że stosując po kolei reguły (VAR) i (SUB) systemu \mathbf{I}_c otrzymujemy taki sam rezultat, jak byśmy zastosowali regułę (VAR) systemu \mathbf{I}_c^s . Ponadto, z (CONST \perp) przez (SUB) otrzymujemy $\mathbf{I}_c \triangleright A \vdash c^\perp : \top$, co odpowiada regule (SUB \perp) z \mathbf{I}_c^s . Oznacza to, że reguły \mathbf{I}_c zawierają w sobie reguły \mathbf{I}_c^s . Zatem, jeśli $\mathbf{I}_c^s \triangleright A \vdash M : \sigma$, to także $\mathbf{I}_c \triangleright A \vdash M : \sigma$.

Odwrotna zależność nie zachodzi, bo mamy na przykład $\mathbf{I}_c \triangleright \{x : \sigma \rightarrow \tau\} \vdash x : (\sigma \wedge \sigma') \rightarrow \tau$, ale nie wyprowadzimy $\mathbf{I}_c^s \triangleright \{x : \sigma \rightarrow \tau\} \vdash x : (\sigma \wedge \sigma') \rightarrow \tau$. ■

Aby móc sprowadzić dowody zaawansowanych twierdzeń dla systemu \mathbf{I}_c do dowodów odpowiadających im twierdzeń dla systemu \mathbf{I}_c^s , musimy najpierw pokazać, że pewne podstawowe własności systemu \mathbf{I}_c^s , wykorzystywane w tych dowodach, dotyczą także systemu \mathbf{I}_c .

Lemat 3.26 (Osłabianie) *Jeśli $\mathbf{I}_c \triangleright A \vdash M : \tau$, to $\mathbf{I}_c \triangleright A + A' \vdash M : \tau$, dla dowolnego środowiska A' .*

Dowód: Dowód indukcyjny względem długości wyprowadzenia $\mathbf{I}_c \triangleright A \vdash M : \tau$. ■

Zauważmy, że dowód własności osłabiania dla systemu \mathbf{I}_c^s mogliśmy również przeprowadzić indukcyjnie, ale wówczas mieliśmy lemat 3.6, z którego (oraz z banalnego lematu 3.4) osłabianie wynikało bezpośrednio.

Lemat 3.27 (Substytutywność) *Jeśli $\mathbf{I}_c \triangleright A \vdash M : \sigma$, to $\mathbf{I}_c \triangleright SA \vdash M : S\sigma$, dla dowolnego podstawienia S .*

Dowód: Dowód indukcyjny względem długości wyprowadzenia $\mathbf{I}_c \triangleright A \vdash M : \sigma$. Jeśli ostatnim krokiem było zastosowanie reguły (VAR), to $M = x$ i patrz dowód lematu o substytutywności dla \mathbf{I}_c^s , punkt (ii), z dokładnością do zastąpienia nierówności \leq_0 równością (ze względu na różnice pomiędzy regułami (VAR) w obu systemach). Tak samo dla reguł (CONST \perp), (CONST \top), (VAR), (ABS) i (APP) — patrz punkty (i), (iii) i (iv) tamże; tym razem dowód przebiega identycznie. Jeśli natomiast ostatnim krokiem było zastosowanie (SUB), to istnieje krótsze wyprowadzenie $\mathbf{I}_c \triangleright A \vdash M : \sigma'$ i $\sigma' \leq_2 \sigma$. Z założenia indukcyjnego mamy $\mathbf{I}_c \triangleright SA \vdash M : S\sigma'$, a z lematu 3.1, skoro $\sigma' \leq_2 \sigma$, to $S\sigma' \leq_2 S\sigma$. Stąd przez (SUB) otrzymujemy $\mathbf{I}_c \triangleright SA \vdash M : S\sigma$. ■

Mając te wszystkie własności, możemy teraz sprawnie sprowadzić problem typowości termu w \mathbf{I}_c do problemu typowości termu w \mathbf{I}_c^s .

Twierdzenie 3.28 *Term M jest typowalny w \mathbf{I}_c wtedy i tylko wtedy, gdy jest on typowalny w \mathbf{I}_c^s .*

Dowód: Sprowadzimy problem typowości termu w \mathbf{I}_c do problemu istnienia rozwiązania pewnego układu równań i $\leq_{2,0}$ -nierówności podtypowych, przy czym układ ten będzie taki sam, jak układ równoważny problemowi typowości tego samego termu w \mathbf{I}_c^s .

Niech dla danego termu M , zbiór $\text{PC}(M)$ będzie określony jak w definicji 3.15. Dla \mathbf{I}_c również zachodzi twierdzenie 3.18 (wynika to z twierdzenia 3.25). Aby udowodnić, że dla \mathbf{I}_c zachodzi również twierdzenie 3.20, posłużymy się indukcją ze względu na długość wyprowadzenia $\mathbf{I}_c \triangleright A \vdash M : \sigma$. Jeśli ostatnim krokiem było wykorzystanie reguły (VAR), to $M = x$ i patrz dowód twierdzenia 3.20, punkt (i), z dokładnością do zastąpienia nierówności \leq_0 równością (ze względu na różnice pomiędzy regułami (VAR) w obu systemach). Analogicznie, dla reguł (CONST \perp), (CONST \top), (ABS) i (APP) — patrz punkty (ii), (iii) i (iv) tamże. Jeśli zaś $\mathbf{I}_c \triangleright A \vdash M : \sigma$ uzyskano przez (SUB), to istnieje krótsze wyprowadzenie $\mathbf{I}_c \triangleright A \vdash M : \sigma'$ i $\sigma' \leq_2 \sigma$.

Z założenia indukcyjnego, istnieje $\langle A', \sigma'', C \rangle \in \text{PC}(M)$ i rozwiązanie S układu c , takie, że $S\langle A', \sigma'' \rangle \leq \langle A, \sigma' \rangle$. A że $\langle A, \sigma' \rangle \leq \langle A, \sigma \rangle$, to z przechodniości \leq otrzymujemy $S\langle A', \sigma'' \rangle \leq \langle A, \sigma \rangle$. ■

Zauważmy, że z powyższego twierdzenia oraz z twierdzenia 3.13 wynika, że każdy term typowalny w \mathbf{I}_c jest silnie normalizowalny. Wypadałoby jeszcze udowodnić, że system \mathbf{I}_c jest zamknięty ze względu na β -redukcje.

Twierdzenie 3.29 (Zamkniętość \mathbf{I}_c ze względu na β -redukcję) *Jeśli $\mathbf{I}_c \triangleright A \vdash M : \sigma$ oraz $M \rightarrow \rightarrow_{\beta} N$, to $\mathbf{I}_c \triangleright A \vdash N : \sigma$.*

Dowód: Dowód taki sam, jak dowód twierdzenia 3.11, a nawet prostszy: w punkcie (i) nierówność \leq_0 zastępujemy równością (ze względu na różnice pomiędzy regułami (VAR) w systemach \mathbf{I}_c^s i \mathbf{I}_c). ■

Moglibyśmy od razu rozważać system \mathbf{I}_c — zaoszczędziłoby nam to dowodzenia dosyć silnego jak na system \mathbf{I}_c^s lematu 3.9, wyrażającego własność bezpośrednio wynikającą z reguły (SUB) systemu \mathbf{I}_c , a więc trywialną w tym systemie. Przy okazji ominęłaby nas konieczność dowodzenia lematu 3.6, użytego w dowodzie lematu 3.9 (lemat 3.6 jest wprawdzie wykorzystywany również w dowodzie lematu o osłabianiu w \mathbf{I}_c^s , ale ten dowód równie dobrze można przeprowadzić indukcyjnie, jak dowód lematu o osłabianiu dla systemu \mathbf{I}_c).

Tym niemniej interesujące wydawało się pokazanie, że system bez (SUB) jest równie silny, jak system \mathbf{I}_c . Ponadto, w rozdziale 5 będziemy pokazywać równoważność systemów \mathbf{I}_c^s i \mathbf{ML}_c^s ; analogiczna równoważność pomiędzy systemami \mathbf{I}_c i \mathbf{ML}_c^s byłaby trudniejsza do udowodnienia — na przykład dla \mathbf{I}_c nie zachodzi lemat 5.8.

3.4 Porównanie \mathbf{I}_c z \mathbf{I}_c dla języka bez podtypów

W niniejszym podrozdziale często będziemy odwoływać się do twierdzeń 3.18 i 3.20, dotyczących systemu \mathbf{I}_c^s , chociaż mowa będzie o systemie \mathbf{I}_c , a nie \mathbf{I}_c^s . Mamy prawo tak robić, bo jak już napisaliśmy w dowodzie twierdzenia 3.28, wspomniane dwa twierdzenia zachodzą także dla systemu \mathbf{I}_c . Tak naprawdę, równie dobrze wszystkie rozważania z niniejszego podrozdziału moglibyśmy przeprowadzić w odniesieniu do systemu \mathbf{I}_c^s , a nie \mathbf{I}_c , zdecydowaliśmy się jednak — dla ustalenia uwagi — na ten drugi system.

Jim [6] dla swojego systemu \mathbf{I}_2 , będącego odpowiednikiem systemu \mathbf{I}_c dla języka bez podtypów, wprowadza definicję par głównych, która zaadaptowana do systemu \mathbf{I}_c brzmiałaby następująco:

Definicja 3.30 *Para $\langle A, \sigma \rangle$ jest parą główną dla termu M , jeśli $\mathbf{I}_c \triangleright A \vdash M : \sigma$ i dla wszystkich innych par $\langle A', \sigma' \rangle$ takich, że $\mathbf{I}_c \triangleright A' \vdash M : \sigma'$, istnieje podstawienie S takie, że $S\langle A, \sigma \rangle \leq \langle A', \sigma' \rangle$. ■*

Intuicyjnie, para główna jest typowaniem wyprowadzającym z najsłabszych założeń w środowisku najsilniejszy typ, a więc typowaniem w pewnym sensie najlepszym.

Jim dowodzi, że w jego systemie każdy term ma parę główną. Okazuje się, że system \mathbf{I}_c nie ma takiej własności. Nie jest to faktem trywialnym, bo wiele termów ma w \mathbf{I}_c parę główną. Przykładowo, rozważmy term $M = \lambda x.x$. Ponieważ $\langle \emptyset, t \rightarrow t, \emptyset \rangle \in \text{PC}(M)$, to na mocy twierdzenia 3.20: jeśli $A \vdash M : \sigma$, to istnieje takie podstawienie S , że $S\langle \emptyset, t \rightarrow t \rangle \leq \langle A, \sigma \rangle$. Ponadto, ponieważ podstawienie idyntycznościowe jest rozwiązaniem pustego układu z $\text{PC}(M)$, to z twierdzenia 3.18: $\mathbf{I}_c \triangleright \emptyset \vdash M : t \rightarrow t$. Podsumowując, $\langle \emptyset, t \rightarrow t \rangle$ jest parą główną dla termu $\lambda x.x$. Oczywiście, takie rozumowanie daje się rozszerzyć na wszystkie termy M , dla których układ w $\text{PC}(M)$ jest pusty, lub — ogólnie — tożsamościowy (to znaczy taki, że jego rozwiązaniem jest podstawienie idyntycznościowe). A zatem wniosek jest taki, że wszystkie takie termy mają parę główną w \mathbf{I}_c .

Pozostaje zastanowić się, jak ma się rzecz z termami M , dla których układ z $\text{PC}(M)$ nie jest tożsamościowy. Można przypuszczać, że dla niektórych z takich termów istnieje para główna, a dla niektórych nie istnieje. Analiza, od czego to zależy, wykracza poza ramy niniejszej pracy i wydaje się nie przynosić żadnych konstruktywnych wniosków. Ograniczymy się zatem do zaprezentowania termu nie posiadającego pary głównej w \mathbf{I}_c — uzasadni to, dlaczego — w odróżnieniu od Jima — nie byliśmy w stanie zaprezentować algorytmu obliczającego dla dowolnego termu jego parę główną.

W tym celu najpierw wprowadzamy pomocniczy lemat.

Lemat 3.31 *Niech $\langle A', \sigma', C \rangle \in \text{PC}(M)$. Jeśli $\langle A, \sigma \rangle$ jest parą główną dla M , to dla pewnego rozwiązania S układu C , para $S\langle A', \sigma' \rangle$ jest również parą główną dla M .*

Dowód: Skoro $\langle A, \sigma \rangle$ jest parą główną dla M , to $\mathbf{I}_c \triangleright A \vdash M : \sigma$. Wówczas, na mocy twierdzenia 3.20, istnieje rozwiązanie S układu C takie, że

$$S\langle A', \sigma' \rangle \leq \langle A, \sigma \rangle. \quad (3.12)$$

Pokażemy, że $S\langle A', \sigma' \rangle$ jest parą główną dla M . Po pierwsze — zauważmy, że na mocy twierdzenia 3.18, zachodzi $\mathbf{I}_c \triangleright SA' \vdash M : S\sigma'$. Po drugie — niech $\langle A_1, \sigma_1 \rangle$ będzie dowolną parą taką, że $\mathbf{I}_c \triangleright A_1 \vdash M : \sigma_1$. Ponieważ $\langle A, \sigma \rangle$ jest parą główną dla M , to dla pewnego podstawienia R :

$$R\langle A, \sigma \rangle \leq \langle A_1, \sigma_1 \rangle. \quad (3.13)$$

No, ale z (3.12) wynika również, że $RS\langle A', \sigma' \rangle \leq R\langle A, \sigma \rangle$. To wraz z (3.13) z przechodniości \leq daje, że $RS\langle A', \sigma' \rangle \leq \langle A_1, \sigma_1 \rangle$. ■

Wniosek z powyższego lematu jest taki, że dla danego termu M i trójki $\langle A, \sigma, C \rangle \in \text{PC}(M)$, pary głównej dla M wystarczy szukać w zbiorze

$\{S\langle A, \sigma \mid S - \text{rozwiązanie układu } C \rangle\}$. Jeśli zatem wśród takich par nie ma pary głównej, to term M nie ma jej w ogóle.

Udowodnimy, że term $M = z(\lambda x.x(xy))$ nie ma pary głównej w \mathbf{I}_c . Jednym z elementów w $\text{PC}(M)$ jest $\langle A, \sigma, C \rangle = \langle \{y : t_1, z : t_8\}, t_{10}, \{t_8 = t_9 \rightarrow t_{10}, (t_2 \wedge t_5) \rightarrow t_7 \leq_2 t_9, t_2 = t_3 \rightarrow t_4, t_1 \leq_2 t_3, t_5 = t_6 \rightarrow t_7, t_4 \leq_2 t_6\} \rangle$. Niech u będzie dowolną zmienną typową. Zauważmy, że rozwiązaniem układu C jest m. in. podstawienie $\{t_1 := u, t_2 := u \rightarrow u, t_3 := u, t_4 := u, t_5 := u \rightarrow u, t_6 := u, t_7 := u, t_8 := ((u \rightarrow u) \rightarrow u) \rightarrow u, t_9 := (u \rightarrow u) \rightarrow u, t_{10} := u\}$. Na mocy twierdzenia 3.18 otrzymujemy zatem, że poprawnym typowaniem termu M jest typowanie

$$\mathbf{I}_c \triangleright \{y : u, z : ((u \rightarrow u) \rightarrow u) \rightarrow u\} \vdash M : u. \quad (3.14)$$

Stąd wniosek, że jeśli S ma być takim rozwiązaniem układu C , że $S\langle A, \sigma \rangle$ będzie parą główną dla M , to w $S(t_1)$, $S(t_8)$ i w $S(t_{10})$ nie mogą występować stałe (c^\perp i c^\top) — bo w przeciwnym przypadku nie będzie takiego podstawienia R , żeby $RS\langle A, \sigma \rangle$ było porównywalne z parą z typowania (3.14). Wynika to z faktu, że stała jest porównywalna wyłącznie ze stałą, a zmienna — z sobą samą.

Skoro w $S(t_8)$ nie mogą występować stałe, a $(t_8 = t_9 \rightarrow t_{10}) \in C$, to w $S(t_9)$ i w $S(t_{10})$ także nie mogą występować stałe. Skoro w $S(t_9)$ nie mogą występować stałe, a $((t_2 \wedge t_5) \rightarrow t_7 \leq_2 t_9) \in C$, to w $S(t_2)$, $S(t_5)$ i w $S(t_7)$ też nie mogą występować stałe (wynika z faktu, że zmienne są porównywalne wyłącznie ze zmiennymi, i z definicji \leq_2). Tak rozumując, dochodzimy w końcu do wniosku, że w żadnym z $S(t_i)$, dla $i = 1, \dots, 10$, nie mogą występować stałe, a zatem układ równań i nierówności C redukuje się do układu równań, bo nierówności pomiędzy zmiennymi wymuszają po prostu równość tych zmiennych (nierówność $(t_2 \wedge t_5) \rightarrow t_7 \leq_2 t_9$ redukuje się do $t_2 \rightarrow t_7 = t_9$ oraz $t_2 = t_5$). Okazuje się, że najbardziej ogólnym unifikatorem tak powstałego układu jest podstawienie $S = \{t_2 := t_1 \rightarrow t_1, t_3 := t_1, t_4 := t_1, t_5 := t_1 \rightarrow t_1, t_6 := t_1, t_7 := t_1, t_8 := ((t_1 \rightarrow t_1) \rightarrow t_1) \rightarrow t_{10}, t_9 := (t_1 \rightarrow t_1) \rightarrow t_1\}$. A zatem jedynym możliwym kandydatem na parę główną dla M jest para $S\langle A, \sigma \rangle = \langle \{y : t_1, z : ((t_1 \rightarrow t_1) \rightarrow t_1) \rightarrow t_{10}\}, t_{10} \rangle$.

No, ale rozwiązaniem układu C jest również podstawienie $\{t_1 := \top, t_2 := \top \rightarrow \perp, t_3 := \top, t_4 := \perp, t_5 := \perp \rightarrow \perp, t_6 := \perp, t_7 := \perp, t_8 := ((\top \rightarrow \perp) \rightarrow \perp) \rightarrow t_{10}, t_9 := (\top \rightarrow \perp) \rightarrow \perp\}$, a zatem na mocy twierdzenia 3.18 poprawnym typowaniem dla M jest

$$\mathbf{I}_c \triangleright \{y : \top, z : ((\top \rightarrow \perp) \rightarrow \perp) \rightarrow t_{10}\} \vdash M : t_{10}. \quad (3.15)$$

Zastanówmy się, czy istnieje takie podstawienie R , że $R\langle \{y : t_1, z : ((t_1 \rightarrow t_1) \rightarrow t_1) \rightarrow t_{10}\}, t_{10} \rangle \leq \langle \{y : \top, z : ((\top \rightarrow \perp) \rightarrow \perp) \rightarrow t_{10}\}, t_{10} \rangle$. Z definicji \leq musiałyby wówczas zachodzić $\top \leq_1 R(t_1)$ oraz $((\top \rightarrow \perp) \rightarrow \perp) \rightarrow t_{10} \leq_1 ((R(t_1) \rightarrow R(t_1)) \rightarrow R(t_1)) \rightarrow R(t_{10})$. Skoro $\top \leq_1 R(t_1)$, to $R(t_1) = \top$. Wówczas druga nierówność ma postać $((\top \rightarrow \perp) \rightarrow \perp) \rightarrow t_{10} \leq_1 ((\top \rightarrow$

$\top) \rightarrow \top) \rightarrow \top$, czyli m. in. $(\top \rightarrow \top) \rightarrow \top \leq_0 (\top \rightarrow \perp) \rightarrow \perp$, a to nie zachodzi.

Stąd wniosek, że term M nie ma pary głównej.

Rozdział 4

System \mathbf{ML}_c^s

W tym rozdziale wprowadzimy system \mathbf{ML}_c^s , będący rozszerzeniem tradycyjnego systemu typów ML na język z podtypami. Szczególnie dużo uwagi poświęcimy własnościom substytutuwności oraz typu głównego, będziemy z nich bowiem korzystać, pokazując w rozdziale 5, że system \mathbf{ML}_c^s ma podobną siłę typowania do systemu \mathbf{I}_c^s .

4.1 Podstawowe pojęcia

Zbiór warunków, zwany dalej po prostu *warunkami*, to zbiór, którego każdy element to albo równanie, albo nierówność pomiędzy typami prostymi. A zatem jest to po prostu szczególny przypadek układu równań i $\leq_{2,0}$ -nierówności, zdefiniowanego w podrozdziale 3.2.3. Analogicznie, jak tam, definiuje się pojęcie rozwiązania: podstawienie S jest rozwiązaniem zbioru warunków C , jeśli $S\sigma \leq_0 S\tau$ dla wszystkich nierówności $(\sigma \leq_0 \tau) \in C$ oraz $S\sigma = S\tau$ dla wszystkich równań $(\sigma = \tau) \in C$. O takim podstawieniu mówimy również, że spełnia ono ten zbiór warunków. Warunki są niesprzeczne, jeśli mają rozwiązanie, zaś sprzeczne, jeśli go nie posiadają. Zbiór warunków jest tożsamościowy, jeśli jego rozwiązaniem jest podstawienie identycznościowe, lub inaczej: gdy po prostu wszystkie warunki do niego należące zachodzą.

Z faktu, że tak zdefiniowany zbiór warunków jest szczególnym przypadkiem układu równań i $\leq_{2,0}$ -nierówności, wynika przy okazji, że dla zbioru warunków również zachodzi twierdzenie 3.22, czyli, że problem istnienia/znalezienia rozwiązania dla danego układu jest rozstrzygalny.

Działanie podstawienia rozszerzamy w naturalny sposób na zbiór warunków: wynikiem jest zbiór warunków powstałych z warunków wejściowych poprzez zaaplikowanie podstawienia do wszystkich typów występujących w warunkach. Zauważmy, że aplikując podstawienie do warunków tożsamościowych otrzymujemy warunki tożsamościowe — wynika to z lematu 3.1.

W przypadku systemu \mathbf{ML}_c^s każdy rozważany typ będzie należeć do \mathbf{T}_0 lub do jednego z następujących zbiorów, zdefiniowanych jako najmniejsze

zbiory spełniające odpowiednie równania:

$$\begin{aligned}\mathbf{T}_C &= \{\langle C, \tau \rangle \mid C \text{ – warunki, } \tau \in \mathbf{T}_0\}, \\ \mathbf{T}_{\forall C} &= \mathbf{T}_C \cup \{\forall t\sigma \mid t \in \mathbf{TypeVars}, \sigma \in \mathbf{T}_{\forall C}\}.\end{aligned}$$

\mathbf{T}_C to *typy proste z warunkami*, $\mathbf{T}_{\forall C}$ natomiast to *kwantyfikowane typy proste z warunkami*. Zauważmy, że oba te zbiory powstają przez rozszerzenie o warunki „tradycyjnych“ zbiorów typów systemu ML.

Jeśli $\langle C, \tau \rangle \in \mathbf{T}_C$, to zmienne wolne typu $\langle C, \tau \rangle$ (czyli $FV(\langle C, \tau \rangle)$) to zmienne występujące w typie τ oraz zmienne występujące w równaniach i nierównościach w zbiorze C . Typ kwantyfikowany $\forall t_1 \dots \forall t_n \tau$ (który możemy zapisać w uproszczeniu jako $\forall t_1 \dots t_n \tau$ lub $\forall \vec{t} \tau$) ma zmienne *związane* (*kwantyfikowane, generyczne*) t_1, \dots, t_n , pozostałe zaś zmienne w takim typie to zmienne wolne. Innymi słowy, $FV(\forall t_1 \dots t_n \tau) = FV(\tau) - \{t_1, \dots, t_n\}$. Możemy utożsamiać typy różniące się jedynie nazwami zmiennych związanych — analogicznie, jak utożsamiamy termy rachunku lambda różniące się jedynie nazwami zmiennych związanych.

W niniejszej pracy przyjmujemy konwencję, że jeśli nie zaznaczono inaczej, to pisząc $\forall t_1 \dots t_n \tau$ rozumiemy, że $\tau \in \mathbf{T}_C$.

Jeśli S jest podstawieniem, $\sigma = \forall t_1 \dots t_n \tau \in \mathbf{T}_{\forall C}$, to $S\sigma$ jest typem $\forall t_1 \dots t_n S'\tau$, gdzie S' to podstawienie dające taką samą wartość, jak S , dla wszystkich argumentów z wyjątkiem t_1, \dots, t_n , na których to zmiennych jest ono identycznościowe. Innymi słowy, w przypadku typu kwantyfikowanego podstawienie działa na zmienne wolne, nie działa zaś na zmienne generyczne. Istotne jest to, żeby przed zaaplikowaniem podstawienia S na typie kwantyfikowanym tak przemianować zmienne związane, by nie kolidowały one ze zmiennymi z $\mathbf{rng}(S)$. Przykładowo, $\{t := t_1\}(\forall t_1(\emptyset, t \rightarrow t_1))$ jest równe $\forall t_2(\emptyset, t_1 \rightarrow t_2)$, a nie $\forall t_1(\emptyset, t_1 \rightarrow t_1)$. (Jak łatwo zauważyć, wystarczy przemianować tylko te zmienne związane, które występują wśród wartości, jakie zwraca S dla zmiennych wolnych z typu, na który działa.) Tak naprawdę, działanie podstawień na typach z kwantyfikatorami jest analogiczne do działania podstawień na lambda-termach.

Niech $\sigma = \forall t_1 \dots t_n \tau \in \mathbf{T}_{\forall C}$ i $\tau, \tau' \in \mathbf{T}_C$. Mówimy, że τ' jest *instancją* σ (albo, że σ *instancjonuje się* do τ'), co zapisujemy jako $\sigma \succ \tau'$, wtedy i tylko wtedy, gdy dla pewnych $\rho_1, \dots, \rho_n \in \mathbf{T}_0$, mamy $\tau' = \{t_1 := \rho_1, \dots, t_n := \rho_n\}\tau$. Obliczenie jakiejś instancji pewnego typu z $\mathbf{T}_{\forall C}$ określamy jako *instancjonowanie* (albo *instancjację*) tego typu. Relację \succ rozszerzamy w sposób następujący: $\sigma \succ (\forall u_1 \dots u_m \tau')$, wtedy i tylko wtedy, gdy u_1, \dots, u_m nie są wolne w σ i $\sigma \succ \tau'$. Relację \succ w naturalny sposób rozszerzamy na środowiska typu $\mathbf{T}_{\forall C}$: $A \succ A'$, jeśli dla każdego $x \in \mathbf{dom}(A')$: $x \in \mathbf{dom}(A)$ oraz $A(x) \succ A'(x)$.

Lemat 4.1 *Niech $\sigma_1 \succ \sigma_2$, gdzie $\sigma_1, \sigma_2 \in \mathbf{T}_{\forall C}$. Jeśli $\sigma_2 \succ \tau$, gdzie $\tau \in \mathbf{T}_C$, to także $\sigma_1 \succ \tau$.*

Dowód: Skoro $\sigma_1 \succ \sigma_2$, to niech $\sigma_1 = \forall t_1 \dots t_n \pi$, $\sigma_2 = \forall u_1 \dots u_m \zeta$, i wówczas z definicji \succ zachodzi: $\sigma_1 \succ \zeta$ i u_1, \dots, u_m nie są wolne w σ_1 .

Niech S będzie najmniejszym podstawieniem odpowiadającym instancjacji $\sigma_2 \succ \tau$, to znaczy takie, że $S\zeta = \tau$ i S działa tylko na u_1, \dots, u_m . A że u_1, \dots, u_m nie są wolne w σ_1 , to stąd wynika, że S nie działa na zmienne wolne z σ_1 .

Niech R będzie najmniejszym podstawieniem odpowiadającym instancjacji $\sigma_1 \succ \zeta$, to znaczy takim, że $R\pi = \zeta$ i R działa tylko na t_1, \dots, t_n , nie działa zaś na zmienne wolne z σ_1 .

Rozważmy podstawienie (SR) . Zarówno S , jak i R nie działają na zmienne wolne z σ_1 . A zatem (SR) również nie działa na zmienne wolne z σ_1 . Stąd wynika, że wartość $(SR)\pi$ odpowiada pewnej instancji typu σ_1 . Co więcej, $(SR)\pi = S(R\pi) = S\zeta = \tau$, a więc σ_1 instancjonuje się do τ . ■

Będziemy jeszcze potrzebowali pojęcia *generalizacji typu względem środowiska*: dla $\tau \in \mathbf{T}_C$ i środowiska A typu $\mathbf{T}_{\forall C}$: $\text{Gen}(A, \tau) = \forall t_1 \dots t_n \tau$, gdzie t_1, \dots, t_n są wszystkimi zmiennymi wolnymi w typie τ , nie wolnymi w środowisku A .

4.2 Podstawowe własności

Definicja 4.2 *Asercje w systemie \mathbf{ML}_c^s są zdefiniowane indukcyjnie regułami przedstawionymi na rysunku 4.1. Piszemy $\mathbf{ML}_c^s \triangleright A \vdash M : \sigma$, jeśli asercja $A \vdash M : \sigma$ została otrzymana przy pomocy reguł systemu \mathbf{ML}_c^s . ■*

Górny indeks „s” w nazwie \mathbf{ML}_c^s sygnalizuje, że system jest sterowany składnią, powstał on bowiem jako modyfikacja sterowanej składnią wersji systemu ML, przedstawionej między innymi w pracy [6].

Na początek lemat wyrażający kilka najprostszych własności systemu \mathbf{ML}_c^s .

Lemat 4.3

- (i) Jeśli $\mathbf{ML}_c^s \triangleright A \vdash M : \sigma$ i $x \notin \mathbf{dom}(A)$, to $\mathbf{ML}_c^s \triangleright A \cup \{x : \tau\} \vdash M : \sigma$.
- (ii) Jeśli $\mathbf{ML}_c^s \triangleright A \cup \{x : \tau\} \vdash M : \sigma$ i x nie jest zmienną wolną w M , to $\mathbf{ML}_c^s \triangleright A \vdash M : \sigma$.
- (iii) Jeśli $\mathbf{ML}_c^s \triangleright A \cup \{x : \tau_1\} \vdash M : \sigma$ i x nie jest zmienną wolną w M , to $\mathbf{ML}_c^s \triangleright A \cup \{x : \tau_2\} \vdash M : \sigma$.
- (iv) Jeśli $\mathbf{ML}_c^s \triangleright A \cup \{x : \forall \vec{t} \langle C_1, \tau_1 \rangle\} \vdash M : \langle C_2, \tau_2 \rangle$, to $\mathbf{ML}_c^s \triangleright A \cup \{x : \forall \vec{t} \langle \emptyset, \tau_1 \rangle\} \vdash M : \langle C_2', \tau_2 \rangle$, gdzie $C_2' \subseteq C_2$.

(CONST \perp)	$A \vdash c^\perp : \langle \{\perp \leq_0 \sigma\}, \sigma \rangle$
(CONST \top)	$A \vdash c^\top : \langle \{\top \leq_0 \sigma\}, \sigma \rangle$
(VAR)	$A \cup \{x : \forall t_1 \dots t_n \langle C, \tau \rangle\} \vdash x : \langle C_2, \tau_2 \rangle$
	jeśli $\forall t_1 \dots t_n \langle C, \tau \rangle \succ \langle C_1, \tau_1 \rangle$ i $C_2 = C_1 \cup \{\tau_1 \leq_0 \tau_2\}$
(ABS)	$\frac{A_x \cup \{x : \langle \emptyset, \tau \rangle\} \vdash M : \langle C, \sigma \rangle}{A \vdash (\lambda x M) : \langle C, \tau \rightarrow \sigma \rangle}$
(APP)	$\frac{A \vdash M : \langle C_1, \tau_1 \rangle, \quad A \vdash N : \langle C_2, \tau_2 \rangle}{A \vdash (MN) : \langle C_1 \cup C_2 \cup \{\tau_1 = \tau_2 \rightarrow \pi\}, \pi \rangle}$
(LET)	$\frac{A \vdash M : \langle C_1, \tau_1 \rangle, \quad A_x \cup \{x : \text{Gen}(A, \langle C_1, \tau_1 \rangle)\} \vdash N : \langle C_2, \tau_2 \rangle}{A \vdash (\mathbf{let } x = M \mathbf{ in } N) : \langle C \cup C_2, \tau_2 \rangle}$
	gdzie $C = \begin{cases} \emptyset & \text{jeśli } x \text{ wolny w } N, \\ C_1 & \text{w przeciwnym przypadku.} \end{cases}$

Rysunek 4.1: Reguły \mathbf{ML}_c^s . Środowiska są typu $\mathbf{T}_{\forall C}$, wyprowadzone typy należą do \mathbf{T}_C . W żadnym z wyprowadzonych typów zbiór warunków nie może być sprzeczny.

Dowód: (i), (ii), (iv) — dowód indukcyjny ze względu na budowę termu M , (iii) — wynika z (i) i (ii). ■

Następny lemat odpowiada analogicznemu lematowi dla ML z pracy [4], przypomina też trochę lemat 3.6. Wyraża fakt, że wzmocnienie środowiska nie psuje jakości wyprowadzonego typu.

Lemat 4.4 *Jeśli $\mathbf{ML}_c^s \triangleright A \cup \{x : \sigma\} \vdash M : \tau$ i $\sigma' \succ \sigma$, to $\mathbf{ML}_c^s \triangleright A \cup \{x : \sigma'\} \vdash M : \tau$.*

Dowód: Dowód indukcyjny ze względu na budowę termu M . Wszystkie przypadki są trywialne, z wyjątkiem przypadku $M = x$, kiedy to należy skorzystać z lematu 4.1. ■

Poniższy lemat będzie wykorzystany w dowodzie twierdzenia o substytucyjności dla \mathbf{ML}_c^s . Jego intuicyjne znaczenie jest takie, że typy w środowiskach nie mogą zawierać warunków sprzecznych — chyba, że typy te odpowiadają zmiennym, które nie występują w typowanym termie.

Lemat 4.5 *Jeśli $\mathbf{ML}_c^s \triangleright A \cup \{x : \forall \vec{t} \langle C, \tau \rangle\} \vdash M : \langle D, \sigma \rangle$ i x jest zmienną wolną w M , to C jest niesprzecznym zbiorem warunków.*

Dowód: Dowód indukcyjny ze względu na budowę termu M . Rozpatrujemy następujące przypadki, w których x jest wolny w M :

(i) $M = x$. Mamy zatem $\forall \vec{t} \langle C, \tau \rangle \succ \langle C_1, \tau_1 \rangle$ i $D = C_1 \cup \{\tau_1 \leq_0 \sigma\}$ i D jest niespreczny. Istnieje zatem rozwiązanie R zbioru D , czyli podstawienie R takie, że RD jest zbiorem warunków tożsamościowych, więc w szczególności RC_1 jest zbiorem warunków tożsamościowych.

Niech S będzie najmniejszym podstawieniem odpowiadającym instancjacji $\forall \vec{t} \langle C, \tau \rangle \succ \langle C_1, \tau_1 \rangle$, to znaczy działającym tylko na t_i z \vec{t} i takim, że $S \langle C, \tau \rangle = \langle C_1, \tau_1 \rangle$. W szczególności mamy więc $SC = C_1$. Zatem zachodzi również $RSC = RC_1$ i skoro RC_1 jest tożsamościowy, to RSC - również. A więc C jest niespreczny, bo ma rozwiązanie (RS) .

(ii) $M = \lambda v N$, $v \neq x$, x zmienna wolna w N . Asercję $\mathbf{ML}_c^s \triangleright A \cup \{x : \forall \vec{t} \langle C, \tau \rangle\} \vdash \lambda x N : \langle D, \sigma \rangle$ uzyskano przez (ABS), a zatem $\sigma = \sigma_1 \rightarrow \sigma_2$ i skorzystano z przesłanki $\mathbf{ML}_c^s \triangleright A_v \cup \{x : \forall \vec{t} \langle C, \tau \rangle\} \cup \{v : \langle \emptyset, \sigma_1 \rangle\} \vdash N : \langle D, \sigma_2 \rangle$. x jest zmienną wolną w N , a zatem, z założenia indukcyjnego, C jest niespreczny.

(iii) $M = M_1 M_2$. Asercję $\mathbf{ML}_c^s \triangleright A \cup \{x : \forall \vec{t} \langle C, \tau \rangle\} \vdash M_1 M_2 : \langle D, \sigma \rangle$ uzyskano przez (APP), a zatem $D = C_1 \cup C_2 \cup \{\tau_1 = \tau_2 \rightarrow \sigma\}$ i skorzystano z przesłanek $\mathbf{ML}_c^s \triangleright A \cup \{x : \forall \vec{t} \langle C, \tau \rangle\} \vdash M_1 : \langle C_1, \tau_1 \rangle$ oraz $\mathbf{ML}_c^s \triangleright A \cup \{x : \forall \vec{t} \langle C, \tau \rangle\} \vdash M_2 : \langle C_2, \tau_2 \rangle$. Jeśli x jest zmienną

wolną w M_1 , to korzystamy z założenia indukcyjnego dla pierwszej z tych przesłanek, w przeciwnym przypadku — korzystamy z założenia indukcyjnego dla drugiej z nich.

(iv) $M = \mathbf{let } v = M_1 \mathbf{ in } M_2$. Rozpatrujemy dwa przypadki:

- a) x jest zmienną wolną w M_1 . Asercję z treści lematu uzyskano przez (LET) m. in. z przesłanki $\mathbf{ML}_c^s \triangleright A \cup \{x : \forall \vec{t} \langle C, \tau \rangle\} \vdash M_1 : \langle C_1, \tau_1 \rangle$, więc skoro x jest wolny w M_1 , to z założenia indukcyjnego C - niesprzeczny.
- b) x jest zmienną wolną w M_2 . A zatem musi być $v \neq x$ i asercję z treści lematu uzyskano przez (LET) m. in. z przesłanki $\mathbf{ML}_c^s \triangleright A_v \cup \{x : \forall \vec{t} \langle C, \tau \rangle\} \cup \{v : \pi\} \vdash M_2 : \langle C_2, \tau_2 \rangle$, więc skoro x wolny w M_2 , to z założenia indukcyjnego C - niesprzeczny. ■

I wreszcie — ostatnie, najważniejsze twierdzenie w tej części rozdziału o systemie \mathbf{ML}_c^s .

Twierdzenie 4.6 (Substytutyność) *Niech $\mathbf{ML}_c^s \triangleright A \vdash M : \langle C, \tau \rangle$. To wówczas:*

- (i) *jeśli dla pewnego podstawienia S warunki SC są niesprzeczne, to $\mathbf{ML}_c^s \triangleright SA \vdash M : S \langle C, \tau \rangle$, oraz*
- (ii) *jeśli $\text{Gen}(A, \langle C, \tau \rangle) \succ \langle C', \tau' \rangle$ i warunki C' są niesprzeczne, to $\mathbf{ML}_c^s \triangleright A \vdash M : \langle C', \tau' \rangle$.*

Dowód: Dowód indukcyjny względem budowy termu M przeprowadzamy jednocześnie dla obu części lematu, przy czym (ii) bezpośrednio wynika z (i). Niech bowiem S będzie najmniejszym podstawieniem, którego działanie na $\langle C, \tau \rangle$ odpowiada instancjacji $\text{Gen}(A, \langle C, \tau \rangle) \succ \langle C', \tau' \rangle$, to znaczy takim, że $S \langle C, \tau \rangle = \langle C', \tau' \rangle$ oraz S działa tylko na zmiennych związanych w $\text{Gen}(A, \langle C, \tau \rangle)$, a więc nie wolnych w A . Z punktu (i), skoro $\mathbf{ML}_c^s \triangleright A \vdash M : \langle C, \tau \rangle$, to $\mathbf{ML}_c^s \triangleright SA \vdash M : S \langle C, \tau \rangle = \langle C', \tau' \rangle$. A że S działa tylko na zmiennych nie wolnych w A , to $SA = A$, a zatem otrzymujemy, że $\mathbf{ML}_c^s \triangleright A \vdash M : \langle C', \tau' \rangle$.

Zauważmy, że z punktu (ii) wynika, że możemy przemianować zmienne z $\langle C, \tau \rangle$ nie występujące wolno w A na jakieś dowolne inne zmienne. Z własności tej będziemy korzystać w dowodzie części (i) lematu. Oto ten dowód — rozpatrujemy następujące przypadki:

- a) $M = c^\perp$. Mamy zatem $\mathbf{ML}_c^s \triangleright A \vdash c^\perp : \langle \{\perp \leq_0 \tau\}, \tau \rangle$ przez (CONST \perp). Przez tę samą regułę mamy również $\mathbf{ML}_c^s \triangleright SA \vdash c^\perp : \langle \{\perp \leq_0 S(\tau)\}, S(\tau) \rangle = S \langle \{\perp \leq_0 \tau\}, \tau \rangle$ — o ile tylko zbiór $S \langle \{\perp \leq_0 \tau\}, \tau \rangle$ niesprzeczny, a tak jest z założenia.

b) $M = x$. Asercję $\mathbf{ML}_c^s \triangleright A \vdash x : \langle C, \tau \rangle$ otrzymano przez (VAR), a zatem $A(x) = \forall \vec{t} \langle C_0, \tau_0 \rangle, \forall \vec{t} \langle C_0, \tau_0 \rangle \succ \langle C_1, \tau_1 \rangle$ i $C = C_1 \cup \{\tau_1 \leq_0 \tau\}$. Niech zmienne z \vec{t} będą przemianowane tak, by nie kolidowały ze zmiennymi występującymi w $\mathbf{rng}(S)$, bo za chwilę będziemy rozważać typ $S\forall \vec{t} \langle C_0, \tau_0 \rangle$.

Niech S' będzie podstawieniem zdefiniowanym następująco:

$$S'(u) = \begin{cases} S(u) & \text{dla } u \text{ nie występujących w } \vec{t}, \\ u & \text{w przeciwnym przypadku.} \end{cases}$$

Wówczas $SA(x) = S\forall \vec{t} \langle C_0, \tau_0 \rangle = \forall \vec{t} S' \langle C_0, \tau_0 \rangle$.

Niech R będzie najmniejszym podstawieniem odpowiadającym instancjacji $\forall \vec{t} \langle C_0, \tau_0 \rangle \succ \langle C_1, \tau_1 \rangle$, to znaczy takim, że $R \langle C_0, \tau_0 \rangle = \langle C_1, \tau_1 \rangle$ oraz R działa tylko na zmiennych z \vec{t} . Niech R' będzie podstawieniem zdefiniowanym następująco:

$$R'(u) = \begin{cases} u & \text{dla } u \text{ nie występujących w } \vec{t}, \\ SR(u) & \text{w przeciwnym przypadku.} \end{cases}$$

Zauważmy, że dla u z wektora \vec{t} mamy: $R'S'(u) = R'(u) = SR(u)$, natomiast dla pozostałych u mamy: $R'S'(u) = S'(u) = S(u) = SR(u)$ (w tym drugim przypadku przekształcenia wynikają z faktów, że zmienne z \vec{t} nie kolidują z $\mathbf{rng}(S)$ oraz, że w $\mathbf{dom}(R)$ są tylko zmienne z \vec{t}). A zatem mamy:

$$R'S' = SR.$$

Instancjonując w $SA(x)$ zmienne t_i na $R'(t_i)$ otrzymujemy $R'S' \langle C_0, \tau_0 \rangle = SR \langle C_0, \tau_0 \rangle = S(R \langle C_0, \tau_0 \rangle) = S \langle C_1, \tau_1 \rangle$. A zatem $SA(x) \succ \langle SC_1, S\tau_1 \rangle$. Ponadto, z faktu, że SC niesprzeczny, wynika, że $SC_1 \cup \{S\tau_1 \leq_0 S\tau\}$ też niesprzeczny. A zatem przez (VAR) otrzymujemy, że $\mathbf{ML}_c^s \triangleright SA \vdash x : \langle SC_1 \cup \{S\tau_1 \leq_0 S\tau\}, S\tau \rangle = S \langle C, \tau \rangle$.

c) $M = \lambda x N$. Asercję $\mathbf{ML}_c^s \triangleright A \vdash \lambda x N : \langle C, \tau \rangle$ otrzymano przez (ABS), a zatem τ jest postaci $\tau_1 \rightarrow \tau_2$ i skorzystano z przesłanki $\mathbf{ML}_c^s \triangleright A_x \cup \{x : \langle \emptyset, \tau_1 \rangle\} \vdash N : \langle C, \tau_2 \rangle$. Z założenia indukcyjnego zachodzi zatem $\mathbf{ML}_c^s \triangleright SA_x \cup \{x : \langle \emptyset, S\tau_1 \rangle\} \vdash N : \langle SC, S\tau_2 \rangle$, a stąd przez (ABS) mamy $\mathbf{ML}_c^s \triangleright SA \vdash \lambda x N : \langle SC, S\tau_1 \rightarrow S\tau_2 \rangle = S \langle C, \tau_1 \rightarrow \tau_2 \rangle = S \langle C, \tau \rangle$.

d) $M = M_1 M_2$. Asercję $\mathbf{ML}_c^s \triangleright A \vdash M_1 M_2 : \langle C, \tau \rangle$ otrzymano przez (APP), a zatem C jest postaci $C_1 \cup C_2 \cup \{\tau_1 = \tau_2 \rightarrow \tau\}$ i skorzystano z przesłanek $\mathbf{ML}_c^s \triangleright A \vdash M_1 : \langle C_1, \tau_1 \rangle$ oraz $\mathbf{ML}_c^s \triangleright A \vdash M_2 : \langle C_2, \tau_2 \rangle$. Skoro SC jest niesprzeczny, to SC_1 oraz SC_2 również są niesprzeczne. A zatem możemy zastosować założenie indukcyjne, otrzymując $\mathbf{ML}_c^s \triangleright SA \vdash M_1 : \langle SC_1, S\tau_1 \rangle$ i $\mathbf{ML}_c^s \triangleright SA \vdash M_2 : \langle SC_2, S\tau_2 \rangle$.

Stąd przez (APP) otrzymujemy $\mathbf{ML}_c^s \triangleright SA \vdash M_1 M_2 : \langle SC_1 \cup SC_2 \cup \{S\tau_1 = S\tau_2 \rightarrow S\tau\}, S\tau \rangle$.

- e) $M = \mathbf{let } x = M_1 \mathbf{ in } M_2$ i x nie występuje wolno w M_2 . Wówczas $\langle C, \tau \rangle$ jest postaci $\langle C_1 \cup C_2, \tau_2 \rangle$ i asercję $\mathbf{ML}_c^s \triangleright A \vdash \mathbf{let } x = M_1 \mathbf{ in } M_2 : \langle C_1 \cup C_2, \tau_2 \rangle$ uzyskano z przesłanek: $\mathbf{ML}_c^s \triangleright A \vdash M_1 : \langle C_1, \tau_1 \rangle$ oraz $\mathbf{ML}_c^s \triangleright A_x \cup \{x : \text{Gen}(A, \langle C_1, \tau_1 \rangle)\} \vdash M_2 : \langle C_2, \tau_2 \rangle$. Skoro $SC = SC_1 \cup SC_2$ jest niesprzeczny, to SC_1 i SC_2 są również niesprzeczne. Zatem możemy do tych przesłanek zaaplikować założenie indukcyjne, otrzymując odpowiednio

$$\mathbf{ML}_c^s \triangleright SA \vdash M_1 : \langle SC_1, S\tau_1 \rangle \quad (4.1)$$

i $\mathbf{ML}_c^s \triangleright SA_x \cup \{x : S\text{Gen}(A, \langle C_1, \tau_1 \rangle)\} \vdash M_2 : \langle SC_2, S\tau_2 \rangle$. Z tego ostatniego faktu, ponieważ x nie jest zmienną wolną w M_2 , na mocy lematu 4.3 (iii) wynika, że $\mathbf{ML}_c^s \triangleright SA_x \cup \{x : \text{Gen}(SA, \langle SC_1, S\tau_1 \rangle)\} \vdash M_2 : \langle SC_2, S\tau_2 \rangle$. To oraz (4.1) przez (LET) daje $\mathbf{ML}_c^s \triangleright SA \vdash \mathbf{let } x = M_1 \mathbf{ in } M_2 : \langle SC_1 \cup SC_2, S\tau_2 \rangle = S\langle C, \tau \rangle$.

- f) $M = \mathbf{let } x = M_1 \mathbf{ in } M_2$ i x występuje wolno w M_2 . Wówczas asercję $\mathbf{ML}_c^s \triangleright A \vdash \mathbf{let } x = M_1 \mathbf{ in } M_2 : \langle C, \tau \rangle$ uzyskano z przesłanek:

$$\mathbf{ML}_c^s \triangleright A \vdash M_1 : \langle C_1, \tau_1 \rangle \quad (4.2)$$

oraz

$$\mathbf{ML}_c^s \triangleright A_x \cup \{x : \text{Gen}(A, \langle C_1, \tau_1 \rangle)\} \vdash M_2 : \langle C, \tau \rangle. \quad (4.3)$$

Możemy przyjąć, że

$$S(v) = v \quad \text{dla } v \in \text{FV}(\langle C_1, \tau_1 \rangle) - \text{FV}(A). \quad (4.4)$$

Gdyby bowiem tak nie było, to moglibyśmy odpowiednio przemianować te zmienne z $\langle C_1, \tau_1 \rangle$ na mocy części (ii) niniejszego lematu, na jakieś inne zmienne $\langle C'_1, \tau'_1 \rangle$ spełniające ten warunek i nigdzie nie kolidujące. W punkcie (4.2) mielibyśmy wówczas w wyprowadzonym typie $\langle C'_1, \tau'_1 \rangle$ zamiast $\langle C_1, \tau_1 \rangle$, a w punkcie (4.3) w środowisku $(x : \text{Gen}(A, \langle C'_1, \tau'_1 \rangle))$ zamiast $(x : \text{Gen}(A, \langle C_1, \tau_1 \rangle))$, natomiast wyprowadzony typ dla M_2 , a więc i dla całego $(\mathbf{let } \dots)$, byłyby taki sam, bo środowiska byłyby takie same — z dokładnością do nazw zmiennych związanych. Z tego samego powodu możemy także założyć, że

$$\forall v \in \text{FV}(\langle C_1, \tau_1 \rangle). (v \notin \text{FV}(A) \Rightarrow v \notin \text{FV}(SA)). \quad (4.5)$$

— znowu wystarczy odpowiednio przemianować zmienne z $\langle C_1, \tau_1 \rangle$ nie wolne w A .

Najpierw udowodnimy, że przy powyższych założeniach zachodzi: $\text{FV}(\langle C_1, \tau_1 \rangle) - \text{FV}(A) \subseteq \text{FV}(\langle SC_1, S\tau_1 \rangle) - \text{FV}(SA)$. Niech bowiem

$t \in \text{FV}(\langle C_1, \tau_1 \rangle) - \text{FV}(A)$. To z (4.4) zachodzi $t \in \text{FV}(\langle SC_1, S\tau_1 \rangle) - \text{FV}(A)$. To oznacza, że $t \in \text{FV}(\langle SC_1, S\tau_1 \rangle)$ oraz $t \notin \text{FV}(A)$. No, ale jeśli takie $t \notin \text{FV}(A)$, to z (4.5) mamy również $t \notin \text{FV}(SA)$. A zatem $t \in \text{FV}(\langle SC_1, S\tau_1 \rangle) - \text{FV}(SA)$. Ponieważ t dowolne, to stąd wynika, że

$$\text{FV}(\langle C_1, \tau_1 \rangle) - \text{FV}(A) \subseteq \text{FV}(\langle SC_1, S\tau_1 \rangle) - \text{FV}(SA). \quad (4.6)$$

Do $\text{Gen}(A, \langle C_1, \tau_1 \rangle)$ będziemy chcieli zaaplikować podstawienie S . $\text{Gen}(A, \langle C_1, \tau_1 \rangle) = \forall t_1 \dots t_n \langle C_1, \tau_1 \rangle$, gdzie $\{t_1, \dots, t_n\} = \text{FV}(\langle C_1, \tau_1 \rangle) - \text{FV}(A)$. Zmienne t_1, \dots, t_n trzeba by przemianować przed zaaplikowaniem podstawienia S , gdyby S mogło podstawić na zmienne z $\langle C_1, \tau_1 \rangle$ (różne od t_1, \dots, t_n , bo na zmienne związane podstawienie nie działa) wartości, w których występowałyby zmienne t_1, \dots, t_n (takie działanie zapewnia poprawność podstawienia; patrz definicja podstawienia). No, ale S nie podstawia na zmienne z $\langle C_1, \tau_1 \rangle$ różne od t_1, \dots, t_n (a więc wolne w A) wartości, w których występują zmienne t_1, \dots, t_n . Gdyby bowiem tak było, to nie zachodziłby warunek (4.5) — bo w środowisku SA pojawiłyby się pewne zmienne z $\{t_1, \dots, t_n\}$ jako wolne. Wniosek: zmiennych związanych z $\text{Gen}(A, \langle C_1, \tau_1 \rangle)$ nie trzeba przemianowywać przed zaaplikowaniem do tego typu podstawienia S . Ułatwi nam to dalszy wywód.

Mamy: $S\text{Gen}(A, \langle C_1, \tau_1 \rangle) = S\forall t_1 \dots t_n \langle C_1, \tau_1 \rangle$, gdzie $\{t_1, \dots, t_n\} = \text{FV}(\langle C_1, \tau_1 \rangle) - \text{FV}(A)$. Zmiennych t_1, \dots, t_n nie trzeba przemianowywać na mocy wywodu z poprzedniego akapitu, oraz z (4.4), S nie działa na nich, a zatem $S\text{Gen}(A, \langle C_1, \tau_1 \rangle) = \forall t_1 \dots t_n \langle SC_1, S\tau_1 \rangle$. Natomiast $\text{Gen}(SA, \langle SC_1, S\tau_1 \rangle) = \forall u_1 \dots u_m \langle SC_1, S\tau_1 \rangle$, gdzie $\{u_1, \dots, u_m\} = \text{FV}(\langle SC_1, S\tau_1 \rangle) - \text{FV}(SA)$. Z faktu (4.6) wynika, że $\{t_1, \dots, t_n\} \subseteq \{u_1, \dots, u_m\}$, a zatem

$$\text{Gen}(SA, \langle SC_1, S\tau_1 \rangle) \succ S\text{Gen}(A, \langle C_1, \tau_1 \rangle). \quad (4.7)$$

Z założenia indukcyjnego, (4.3) daje

$$\mathbf{ML}_c^s \triangleright SA_x \cup \{x : S\text{Gen}(A, \langle C_1, \tau_1 \rangle)\} \vdash M_2 : S\langle C, \tau \rangle.$$

No, ale ponieważ zachodzi (4.7), to z lematu 4.4 zachodzi również

$$\mathbf{ML}_c^s \triangleright SA_x \cup \{x : \text{Gen}(SA, \langle SC_1, S\tau_1 \rangle)\} \vdash M_2 : S\langle C, \tau \rangle. \quad (4.8)$$

Ponieważ x występuje wolno w M_2 , to z lematu 4.5 zastosowanego do (4.8) wynika, że SC_1 - niesprzeczny. Możemy zatem zastosować założenie indukcyjne do (4.2), otrzymując $\mathbf{ML}_c^s \triangleright SA \vdash M_1 : \langle SC_1, S\tau_1 \rangle$. To wraz z (4.8) przez (LET) ostatecznie daje, że $\mathbf{ML}_c^s \triangleright SA \vdash \text{let } x = M_1 \text{ in } M_2 : S\langle C, \tau \rangle$. ■

W powyższym dowodzie w punktach e) i f) wyszło na jaw, dlaczego reguła (LET) ma dwa przypadki — w zależności od tego, czy x jest wolne w M_2 , czy też nie. Musimy umieć wywnioskować, że warunki z typu dla M_1 po zaaplikowaniu podstawienia nie są sprzeczne. Dla x nie występującego wolno w M_2 musimy więc w warunkach wynikowych po prostu mieć warunki z typu dla M_1 . Jednak dla x występującego wolno w M_2 nie możemy w warunkach wynikowych mieć warunków z typu dla M_1 , bo warunki z typu dla M_1 przemianowujemy. W tym drugim przypadku fakt niesprzeczności warunków dla M_1 zapewnia lemat 4.5.

Pottier, roważając system w pewnych aspektach podobny do naszego, w pracy [7] proponuje dla **let** regułę, która w naszym formalizmie wyglądałaby następująco:

$$(LET') \quad \frac{A \vdash M : \langle C_1, \tau_1 \rangle, \quad A_x \cup \{x : \text{Gen}(A, \langle C_1, \tau_1 \rangle)\} \vdash N : \langle C_2, \tau_2 \rangle}{A \vdash (\mathbf{let} \ x = M \ \mathbf{in} \ N) : \langle \varphi(C_1) \cup C_2, \tau_2 \rangle}$$

— gdzie φ jest dowolnym podstawieniem takim, że $\mathbf{dom}(\varphi) = \text{FV}(\langle C_1, \tau_1 \rangle) - \text{FV}(A)$. Taka reguła też umożliwiałaby przeprowadzenie dowodu substytutowności, bo z jednej strony w warunkach wynikowych mamy pewną postać warunków z typu dla M_1 , a z drugiej strony nie zachodzi problem w przypadku, gdy trzeba przemianować zmienne z typu dla M_1 . Tym niemniej pozostaniemy przy regule (LET), bo daje ona mniejszą dowolność w kształtowaniu warunków wynikowych, a co za tym idzie, niektóre dowody z dalszego ciągu pracy będą łatwiejsze do przeprowadzenia.

Wspomniana praca Pottiera poświęcona jest metodom upraszczania układów nierówności (co u nas w ogóle nie ma miejsca), a system typów w niej zaprezentowany nie za wiele ma wspólnego z naszym — w szczególności nie ma własności typu głównego, niezbędnej do udowodnienia równoważności z systemem \mathbf{I}_c^s (będzie o tym mowa w dalszym ciągu pracy), jak również nie wspomina się o własności substytutowności itp. Podsumowując — wbrew pozorom, praca Pottiera nie za wiele ma wspólnego z treścią niniejszego rozdziału.

4.3 Typowalność

Definicja 4.7 (Typ główny) *Typ $\sigma \in \mathbf{T}_C$ jest typem głównym dla termu M w środowisku A wtedy i tylko wtedy, gdy $\mathbf{ML}_c^s \triangleright A \vdash M : \sigma$ oraz dla wszystkich typów σ' takich, że $\mathbf{ML}_c^s \triangleright A \vdash M : \sigma'$, zachodzi $\text{Gen}(A, \sigma) \succ \sigma'$.*

■

Będziemy starali się wykazać, że system \mathbf{ML}_c^s posiada własność typu głównego — czyli, że dla dowolnego termu i środowiska, jeśli dany term jest typowalny w tym środowisku, to ma on w nim typ główny.

W tym celu wprowadzamy algorytm T i dowodzimy, że oblicza on typ główny. Można dostrzec pewne podobieństwo w idei działania tego algorytmu do idei działania algorytmu W, znajdującego typ główny w systemie ML i opisanego między innymi przez Damasa i Milnera w pracy [4].

Definicja 4.8 (Algorytm T) *W poniższym opisie algorytmu, b oznacza wszędzie świeżą zmienną typową, czyli różną od wszystkich zmiennych typowych, z jakimi miał do czynienia algorytm (w rozważanych środowiskach i typach) do chwili generacji tej zmiennej. Dodatkowo, algorytm T kończy działanie z błędem, jeśli w obliczonym typie, albo w typie będącym wynikiem pomocniczego obliczenia, zbiór warunków będzie sprzeczny. Mając to na uwadze, definiujemy $T(A, M) = \sigma$, gdzie:*

(i) *Jeśli $M = c^\perp$, to $\sigma := \langle \{\perp \leq_0 b\}, b \rangle$.
Jeśli $M = c^\top$, to $\sigma := \langle \{\top \leq_0 b\}, b \rangle$.*

(ii) *Jeśli $M = x$, to jeśli $x \notin \mathbf{dom}(A)$, to błąd, w przeciwnym razie niech $A(x) = \forall t_1 \dots t_n \langle C, \tau \rangle$.
Typ ten instancjonujemy do $\langle C_1, \tau_1 \rangle := \{t_1 := b_1, \dots, t_n := b_n\} \langle C, \tau \rangle$,
gdzie b_1, \dots, b_n są świeżymi zmiennymi typowymi.
Zwracamy $\sigma := \langle C_1 \cup \{\tau_1 \leq_0 b\}, b \rangle$.*

(iii) *Jeśli $M = \lambda x N$, to niech:
 $\langle C, \tau \rangle := T(A_x \cup \{x : \langle \emptyset, b \rangle\}, N)$.
Zwracamy $\sigma := \langle C, b \rightarrow \tau \rangle$.*

(iv) *Jeśli $M = M_1 M_2$, to niech:
 $\langle C_1, \tau_1 \rangle := T(A, M_1)$,
 $\langle C_2, \tau_2 \rangle := T(A, M_2)$.
Zwracamy $\sigma := \langle C_1 \cup C_2 \cup \{\tau_1 = \tau_2 \rightarrow b\}, b \rangle$.*

(v) *Jeśli $M = \mathbf{let } x = M_1 \mathbf{ in } M_2$, to niech:
 $\langle C_1, \tau_1 \rangle := T(A, M_1)$,
 $\langle C_2, \tau_2 \rangle := T(A_x \cup \{x : \mathbf{Gen}(A, \langle C_1, \tau_1 \rangle)\}, M_2)$.
Zwracamy $\sigma := \langle C \cup C_2, \tau_2 \rangle$, gdzie:*

$$C = \begin{cases} \emptyset & \text{jeśli } x \text{ jest zmienną wolną w } M_2, \\ C_1 & \text{w przeciwnym przypadku.} \end{cases}$$

■

Pierwsza część dowodu, że algorytm T oblicza typ główny, nie jest zbyt skomplikowana:

Twierdzenie 4.9 (Poprawność algorytmu T) *Jeśli $T(A, M) = \sigma$, to $\mathbf{ML}_c^s \triangleright A \vdash M : \sigma$.*

Dowód: Trywialny dowód indukcyjny ze względu na budowę termu M . ■

Bardziej złożony jest natomiast dowód, że typ obliczony przez algorytm T jest rzeczywiście główny. W tym celu najpierw będziemy potrzebowali kilku dodatkowych definicji i lematów. Na początek podkreślimy, że zmienne generowane podczas działania algorytmu T , będąc świeżymi, zapewniają brak jakichkolwiek kolizji.

Lemat 4.10

- (i) *W zwracanych przez algorytm T typach zmienne są albo świeże, albo wzięte z danego jako argument środowiska.*
- (ii) *Podczas wykonywania algorytmu T dla pewnych danych (A, M) czasami ma miejsce rekurencyjne wywołanie algorytmu T poprzez instrukcję postaci $\sigma := T(A_x \cup \{x : \tau\}, N)$, gdzie $FV(\tau) \cap (FV(A(x)) - FV(A_x)) = \emptyset$. To wówczas, o ile takie wywołanie zakończy się sukcesem, zachodzi: $\forall v \in (FV(A(x)) - FV(A_x)).v \notin FV(\sigma)$.*
- (iii) *Podczas wykonywania algorytmu T dla pewnych danych (A, M) czasem ma miejsce dwukrotne rekurencyjne wywołanie algorytmu T poprzez instrukcje postaci: $\sigma_1 := T(A_1, M_1)$; $\sigma_2 := T(A_2, M_2)$. To wówczas, o ile takie wywołania zakończą się sukcesem, $(FV(\sigma_1) - FV(A_1)) \cap (FV(\sigma_2) - FV(A_2)) = \emptyset$.*
- (iv) *Podczas wykonywania algorytmu T dla pewnych danych (A, M) czasem ma miejsce rekurencyjne wywołanie algorytmu T poprzez instrukcję postaci: $\sigma := T(A_1, M_1)$. To wówczas, o ile takie wywołanie zakończy się sukcesem, $FV(A) \cap (FV(\sigma) - FV(A_1)) = \emptyset$.*

Dowód: Punkt (i) można udowodnić przez indukcję ze względu na budowę termu M . Punkt (ii) bezpośrednio wynika z (i): w środowisku A_x nie ma zmiennych należących do $FV(A(x)) - FV(A_x)$, nie ma ich również w typie τ , a zatem nie ma również prawa ich być w $FV(\sigma)$; owszem, mogłyby się tam pojawić jako nowe, ale zmienne z $FV(A(x)) - FV(A_x)$ nie są w chwili wywołania $T(A_x \cup \{x : \tau\}, N)$ świeże — jako, że występują w $FV(A)$.

Sens punktu (iii) jest taki, że w obliczonych typach σ_1 i σ_2 zbiory zmiennych, po których możemy generalizować odpowiednio w A_1 i w A_2 , są rozłączne. Sens punktu (iv) jest taki, że w obliczonym typie σ zmienne, po których można generalizować względem środowiska A_1 , nie kolidują ze zmiennymi z $FV(A)$. Zachodzenie obu tych własności wynika z faktu, że w zwracanych przez algorytm T typach zmienne, po których możemy generalizować, są świeże, a więc nie występują w żadnych ze środowisk i typów obliczanych bądź rozważanych uprzednio. ■

Będziemy również potrzebowali pojęcia semi-instancji, jak również kilku związanych z nią własności:

Definicja 4.11 (Semi-instancja) Niech A, A' będą środowiskami typu $\mathbf{T}_{\forall C}$. A' jest semi-instancją A , jeśli dla pewnego podstawienia S zachodzi $SA \succ A'$. ■

Lemat 4.12 (Jednoznaczność podstawienia w semi-instancji) Jeśli $R_1A \succ A'$ i $R_2A \succ A'$, to $\forall v \in FV(A). R_1(v) = R_2(v)$.

Dowód: Niech $v \in FV(A)$, czyli dla pewnego x , zmienna v występuje w typie $A(x) = \forall t_1 \dots t_n \sigma$. Niech zatem $A'(x) = \forall u_1 \dots u_m \tau$ i zachodzi

$$R_1A(x) \succ A'(x) \quad \text{oraz} \quad R_2A(x) \succ A'(x). \quad (4.9)$$

Niech zmienne t_1, \dots, t_n będą odpowiednio przemianowane — tak, by nie kolidowały z podstawieniami R_1 i R_2 ; to znaczy na przykład tak, by nie należały do $\mathbf{rng}(R_1) \cup \mathbf{rng}(R_2)$. Niech wówczas R'_1, R'_2 będą podstawieniami identycznościowymi na t_1, \dots, t_n , a na pozostałe zmienne działającymi tak, jak R_1 i R_2 — odpowiednio. Ponieważ $v \notin \{t_1, \dots, t_n\}$, to wystarczy pokazać, że $R'_1(v) = R'_2(v)$.

Z (4.9) i z definicji \succ mamy: $R_1A(x) = \forall t_1 \dots t_n R'_1\sigma \succ \tau$ oraz $R_2A(x) = \forall t_1 \dots t_n R'_2\sigma \succ \tau$. Niech S_1 będzie najmniejszym podstawieniem odpowiadającym instancjacji $\forall t_1 \dots t_n R'_1\sigma \succ \tau$, to znaczy takim, że $S_1R'_1\sigma = \tau$ oraz $\mathbf{dom}(S_1) = \{t_1, \dots, t_n\}$. Analogicznie, niech S_2 będzie najmniejszym podstawieniem odpowiadającym instancjacji $\forall t_1 \dots t_n R'_2\sigma \succ \tau$, to znaczy takim, że $S_2R'_2\sigma = \tau$ oraz $\mathbf{dom}(S_2) = \{t_1, \dots, t_n\}$. Mamy zatem $S_1R'_1\sigma = S_2R'_2\sigma$, czyli musi zachodzić $\forall t \in FV(\sigma). S_1R'_1(t) = S_2R'_2(t)$. W szczególności więc zachodzi $S_1R'_1(v) = S_2R'_2(v)$. No, ale $S_1R'_1(v) = R'_1(v)$, bo S_1 działa tylko na $\{t_1, \dots, t_n\}$, a $t_1, \dots, t_n \notin \mathbf{rng}(R'_1)$. Analogicznie, $S_2R'_2(v) = R'_2(v)$. A zatem $R'_1(v) = R'_2(v)$, co mieliśmy wykazać. ■

Lemat 4.13 Niech $RA \succ A'$ oraz $RGen(A, \langle C, \sigma \rangle) \succ \langle D, \tau \rangle$. To wówczas $RGen(A, \langle C, \sigma \rangle) \succ Gen(A', \langle D, \tau \rangle)$.

Dowód: Niech $RGen(A, \langle C, \sigma \rangle) = R\forall t_1 \dots t_n \langle C, \sigma \rangle$, gdzie $\{t_1, \dots, t_n\} = FV(\langle C, \sigma \rangle) - FV(A)$. Niech $Gen(A', \langle D, \tau \rangle) = \forall u_1 \dots u_m \langle D, \tau \rangle$, gdzie $\{u_1, \dots, u_m\} = FV(\langle D, \tau \rangle) - FV(A')$.

Z założenia i z definicji \succ zachodzi $R\forall t_1 \dots t_n \langle C, \sigma \rangle \succ \langle D, \tau \rangle$. Należy więc pokazać, że żadne z u_1, \dots, u_m nie jest wolne w $R\forall t_1 \dots t_n \langle C, \sigma \rangle$, a wówczas z definicji \succ otrzymamy, że $R\forall t_1 \dots t_n \langle C, \sigma \rangle \succ \forall u_1 \dots u_m \langle D, \tau \rangle$.

Dowód nie wprost. Niech $u_x \in \{u_1, \dots, u_m\}$ będzie takie, że u_x występuje wolne w $R\forall t_1 \dots t_n \langle C, \sigma \rangle$. To oznacza, że dla pewnej zmiennej $v \in FV(\langle C, \sigma \rangle) \cap FV(A)$, w typie $R(v)$ występuje u_x (bo R działa wewnątrz typu $\forall t_1 \dots t_n \langle C, \sigma \rangle$ na wszystkie zmienne z wyjątkiem nie należących do

$\text{FV}(A)$ zmiennych t_1, \dots, t_n). No, ale skoro $v \in \text{FV}(A)$, zaś u_x występuje w $R(v)$, to $u_x \in \text{FV}(RA)$, a więc także, ponieważ $RA \succ A'$, mamy $u_x \in \text{FV}(A')$. Sprzeczność, bo $u_x \in \text{FV}(\langle D, \tau \rangle) - \text{FV}(A')$. ■

Teraz możemy już przystąpić do udowodnienia, że typ obliczony przez algorytm T jest rzeczywiście typem głównym. Do przeprowadzenia dowodu indukcyjnego będziemy potrzebowali silniejszego założenia:

Twierdzenie 4.14 (Pełność algorytmu T) *Niech A będzie środowiskiem typu $\mathbf{T}_{\forall C}$, zaś M – termem M_m . Niech A' będzie semi-instancją A , zaś τ – typem takim, że $\mathbf{ML}_C^S \triangleright A' \vdash M : \tau$. To wówczas:*

(i) $T(A, M)$ wykonuje się poprawnie, i

(ii) jeśli $T(A, M) = \sigma$, to dla pewnego podstawienia R :

$$RA \succ A' \quad \text{oraz} \quad R\text{Gen}(A, \sigma) \succ \tau.$$

Dowód: Dowód indukcyjny ze względu na budowę termu M . Rozważamy następujące przypadki:

- $M = c^\perp$ lub $M = c^\top$.

Najpierw rozważmy sytuację $M = c^\perp$. Jeśli $\mathbf{ML}_C^S \triangleright A' \vdash c^\perp : \tau$, to skorzystano z reguły (CONST \perp), a zatem τ jest postaci $\langle \{\perp \leq_0 \pi\}, \pi \rangle$, dla pewnego $\pi \in \mathbf{T}_0$. Zauważmy, że $T(A, c^\perp)$ wykonuje się poprawnie, dając $\sigma = \langle \{\perp \leq_0 b\}, b \rangle$, gdzie b – świeża zmienna typowa.

Skoro A' jest semi-instancją A , to dla pewnego S : $SA \succ A'$. Niech zatem $R = S$, a otrzymamy, że $RA \succ A'$. Ponadto, $R\text{Gen}(A, \sigma) = R\forall b \langle \{\perp \leq_0 b\}, b \rangle = \forall b \langle \{\perp \leq_0 b\}, b \rangle$. Ten typ instancjonuje się do $\langle \{\perp \leq_0 \pi\}, \pi \rangle$ poprzez podstawienie na b typu π .

Dla $M = c^\top$ dowód analogiczny. □

- $M = x$.

Asercję $\mathbf{ML}_C^S \triangleright A' \vdash x : \tau$ otrzymano przez (VAR), a zatem $A'(x) = \forall u_1 \dots u_m \langle C'_0, \tau'_0 \rangle$, $A'(x) \succ \langle C'_1, \tau'_1 \rangle$ i $\tau = \langle C'_1 \cup \{\tau'_1 \leq_0 \pi\}, \pi \rangle$, przy czym zbiór $C'_1 \cup \{\tau'_1 \leq_0 \pi\}$ niesprzeczny.

A' jest semi-instancją A , a zatem istnieje takie podstawienie S , że $SA \succ A'$, czyli w szczególności $x \in \mathbf{dom}(A)$ i $SA(x) \succ A'(x)$. Niech $A(x) = \forall t_1 \dots t_n \langle C_0, \tau_0 \rangle$. Skoro $SA(x) \succ A'(x)$ i $A'(x) \succ \langle C'_1, \tau'_1 \rangle$, to na mocy lematu 4.1 otrzymujemy, że także $SA(x) \succ \langle C'_1, \tau'_1 \rangle$, czyli, że

$$S\forall t_1 \dots t_n \langle C_0, \tau_0 \rangle \succ \langle C'_1, \tau'_1 \rangle. \quad (4.10)$$

Ponieważ zachodzi również $\forall c.c \succ \pi$, gdzie c to nowa zmienna, nie występująca w dotychczas rozważanych typach i w $\mathbf{rng}(S)$, to z (4.10) wynika, że:

$$S\forall t_1 \dots t_n c \langle C_0 \cup \{\tau_0 \leq_0 c\}, c \rangle \succ \langle C'_1 \cup \{\tau'_1 \leq_0 \pi\}, \pi \rangle. \quad (4.11)$$

Ponieważ $x \in \mathbf{dom}(A)$, to $T(A, x)$ wykona się bez błędu spowodowanego brakiem pary dla x w środowisku. Algorytm T zinstancjonuje t_1, \dots, t_n z $A(x)$ na świeże zmienne b_1, \dots, b_n , dając $\langle C_1, \tau_1 \rangle = \{t_1 := b_1, \dots, t_n := b_n\} \langle C_0, \tau_0 \rangle$. Mógłby tu wystąpić błąd, gdyby zbiór C_0 był sprzeczny, ale on sprzeczny nie jest, bo w (4.10) jest on (semi)instancjonowany do zbioru C'_1 , który nie jest sprzeczny. Wynikowym typem, zwróconym przez algorytm T, będzie $\sigma = \langle C_1 \cup \{\tau_1 \leq_0 b\}, b \rangle$, gdzie b to świeża zmienna. Tu również nie będzie żadnego błędu spowodowanego sprzecznymi warunkami.

Rozpatrzmy $\mathbf{Gen}(A, \sigma)$. Generalizując w A typ σ możemy kwantyfikować b_1, \dots, b_n i b (pozostałe zmienne z σ są wolne w A , bo są wolne w $A(x)$). Ponieważ typy kwantyfikowane rozważamy z dokładnością do nazw zmiennych kwantyfikowanych (o ile tylko nie kolidują), to przemianujemy w $\mathbf{Gen}(A, \sigma)$ zmienne b_i z powrotem na t_i . Otrzymamy wówczas, że

$$\mathbf{Gen}(A, \sigma) = \forall t_1 \dots t_n b \langle C_0 \cup \{\tau_0 \leq_0 b\}, b \rangle. \quad (4.12)$$

Niech $R = S$. To $RA \succ A'$ oraz z (4.12) $R\mathbf{Gen}(A, \sigma) = S\forall t_1 \dots t_n b \langle C_0 \cup \{\tau_0 \leq_0 b\}, b \rangle$. Ten ostatni typ możemy utożsamić z typem z lewej strony faktu (4.11), poprzez przemianowanie b w pierwszym z tych typów, a c w drugim z tych typów, na jakąś zmienną nie kolidującą ani tu, ani tu. Wówczas fakt (4.11) da nam, że $R\mathbf{Gen}(A, \sigma) \succ \langle C'_1 \cup \{\tau'_1 \leq_0 \pi\}, \pi \rangle$, czyli, że $R\mathbf{Gen}(A, \sigma) \succ \tau$. \square

- $M = \lambda x N$.

Asercję $\mathbf{ML}_c^s \triangleright A' \vdash \lambda x N : \tau$ otrzymano przez (ABS), a zatem τ jest postaci $\langle C, \tau_1 \rightarrow \tau_2 \rangle$ i użyto przesłanki

$$\mathbf{ML}_c^s \triangleright A'_x \cup \{x : \langle \emptyset, \tau_1 \rangle\} \vdash N : \langle C, \tau_2 \rangle. \quad (4.13)$$

Algorytm T najpierw liczy $\langle C_1, \sigma_1 \rangle = T(A_x \cup \{x : \langle \emptyset, b \rangle\}, N)$, gdzie b to nowa zmienna typowa. Skoro A' jest semi-instancją A , to dla pewnego podstawienia S zachodzi $SA \succ A'$. Ponadto, skoro b , jako świeża, nie występuje w $FV(A)$, to można przyjąć, że S nie działa na b . Mamy wówczas $(S \cup \{b := \tau_1\})(A_x \cup \{x : \langle \emptyset, b \rangle\}) \succ A'_x \cup \{x : \langle \emptyset, \tau_1 \rangle\}$, a zatem $A'_x \cup \{x : \langle \emptyset, \tau_1 \rangle\}$ jest semi-instancją $A_x \cup \{x : \langle \emptyset, b \rangle\}$. Możemy zatem zastosować założenie indukcyjne, z (4.13) otrzymując, że $\langle C_1, \sigma_1 \rangle = T(A_x \cup \{x : \langle \emptyset, b \rangle\}, N)$ obliczy się poprawnie i dla pewnego podstawienia R_1 będziemy mieli:

$$R_1(A_x \cup \{x : \langle \emptyset, b \rangle\}) \succ A'_x \cup \{x : \langle \emptyset, \tau_1 \rangle\} \quad (4.14)$$

oraz

$$R_1 \text{Gen}(A_x \cup \{x : \langle \emptyset, b \rangle\}, \langle C_1, \sigma_1 \rangle) \succ \langle C, \tau_2 \rangle. \quad (4.15)$$

Niech R będzie podstawieniem zdefiniowanym następująco:

$$R(t) = \begin{cases} S(t) & \text{dla } t \in \text{FV}(A(x)) - \text{FV}(A_x), \\ R_1(t) & \text{w przeciwnym przypadku.} \end{cases}$$

Zauważmy, że

$$R(b) = R_1(b) = \tau_1 \quad (4.16)$$

(ostatnie przekształcenie wynika z (4.14)).

Ponieważ $SA_x \succ A'_x$ oraz z (4.14): $R_1 A_x \succ A'_x$, to z lematu o jednoznaczności podstawienia w semi-instancji otrzymujemy, że $\forall v \in \text{FV}(A_x). R(v) = S(v)$. Ponadto, z definicji R : $\forall v \in \text{FV}(A(x)) - \text{FV}(A_x). R(v) = S(v)$. A zatem $\forall v \in \text{FV}(A). R(v) = S(v)$. Czyli, skoro $SA \succ A'$, to także $RA \succ A'$.

Pozostaje dowieść, że $R \text{Gen}(A, \langle C_1, b \rightarrow \sigma_1 \rangle) \succ \langle C, \tau_1 \rightarrow \tau_2 \rangle$. Fakt (4.15) inaczej zapisany oznacza, że $R_1 \forall t_1 \dots t_n \langle C_1, \sigma_1 \rangle \succ \langle C, \tau_2 \rangle$, gdzie $\{t_1, \dots, t_n\} = \text{FV}(\langle C_1, \sigma_1 \rangle) - \text{FV}(A_x) - \{b\}$. Ponieważ R różni się od R_1 tylko na zmiennych z $\text{FV}(A(x)) - \text{FV}(A_x)$, a ich nie ma w typie $\langle C_1, \sigma_1 \rangle$ na mocy lematu 4.10 (ii), to mamy również:

$$R \forall t_1 \dots t_n \langle C_1, \sigma_1 \rangle \succ \langle C, \tau_2 \rangle. \quad (4.17)$$

Rozważamy dwa przypadki:

- (i) $b \notin \text{FV}(\langle C_1, \sigma_1 \rangle)$. Wówczas $\text{FV}(\langle C_1, \sigma_1 \rangle) - \text{FV}(A_x) - \{b\} = \text{FV}(\langle C_1, \sigma_1 \rangle) - \text{FV}(A_x) = \text{FV}(\langle C_1, \sigma_1 \rangle) - \text{FV}(A)$ (to ostatnie przekształcenie wynika z lematu 4.10 (ii)). Mamy zatem z (4.17), że $R \forall t_1 \dots t_n \langle C_1, \sigma_1 \rangle \succ \langle C, \tau_2 \rangle$, gdzie $\{t_1, \dots, t_n\} = \text{FV}(\langle C_1, \sigma_1 \rangle) - \text{FV}(A)$. Stąd, ponieważ $\forall b. b \succ \tau_1$ i $b \notin \{t_1, \dots, t_n\}$ oraz $b \notin \text{FV}(\langle C_1, \sigma_1 \rangle)$, to mamy również: $R \forall t_1 \dots t_n b \langle C_1, b \rightarrow \sigma_1 \rangle \succ \langle C, \tau_1 \rightarrow \tau_2 \rangle$. Ponieważ $b \notin \text{FV}(A)$, to stąd wynika, że $R \text{Gen}(A, \langle C_1, b \rightarrow \sigma_1 \rangle) \succ \langle C, \tau_1 \rightarrow \tau_2 \rangle$.
- (ii) $b \in \text{FV}(\langle C_1, \sigma_1 \rangle)$. Wówczas $(\text{FV}(\langle C_1, \sigma_1 \rangle) - \text{FV}(A_x) - \{b\}) \cup \{b\} = \text{FV}(\langle C_1, \sigma_1 \rangle) - \text{FV}(A_x) = \text{FV}(\langle C_1, \sigma_1 \rangle) - \text{FV}(A)$ (to ostatnie przekształcenie wynika, tak samo jak w punkcie (i), z lematu 4.10 (ii)). Mamy zatem z (4.17) także, że $R \forall t_1 \dots t_n b \langle C_1, \sigma_1 \rangle \succ \langle C, \tau_2 \rangle$, gdzie $\{t_1, \dots, t_n, b\} = \text{FV}(\langle C_1, \sigma_1 \rangle) - \text{FV}(A)$, ponieważ b możemy zinstancjonować tak, jak przedtem działało nań R . Mamy zatem także $R \forall t_1 \dots t_n b \langle C_1, b \rightarrow \sigma_1 \rangle \succ \langle C, \tau_1 \rightarrow \tau_2 \rangle$, bo zauważmy, że instancjonując b do $R(b)$ z (4.16) otrzymujemy τ_1 . A zatem i tym razem udowodniliśmy, że $R \text{Gen}(A, \langle C_1, b \rightarrow \sigma_1 \rangle) \succ \langle C, \tau_1 \rightarrow \tau_2 \rangle$. \square

- $M = M_1 M_2$.

Asercję $\mathbf{ML}_c^s \triangleright A' \vdash M_1 M_2 : \tau$ otrzymano przez (APP) z przesłanek: $\mathbf{ML}_c^s \triangleright A' \vdash M_1 : \langle D_1, \tau_1 \rangle$ oraz $\mathbf{ML}_c^s \triangleright A' \vdash M_2 : \langle D_2, \tau_2 \rangle$, przy czym $\tau = \langle D_1 \cup D_2 \cup \{\tau_1 = \tau_2 \rightarrow \pi\}, \pi \rangle$, gdzie warunki w typie τ są niesprzeczne.

Algorytm T najpierw oblicza $\langle C_1, \sigma_1 \rangle = T(A, M_1)$. Z założenia indukcyjnego obliczy się to prawidłowo i dla pewnego R_1 będzie zachodziło: $R_1 A \succ A'$ oraz $R_1 \text{Gen}(A, \langle C_1, \sigma_1 \rangle) \succ \langle D_1, \tau_1 \rangle$. Niech $R = R_1$. Zachodzi wówczas $RA \succ A'$ oraz

$$R \forall t_1 \dots t_n \langle C_1, \sigma_1 \rangle \succ \langle D_1, \tau_1 \rangle \quad (4.18)$$

— gdzie $\{t_1, \dots, t_n\} = \text{FV}(\langle C_1, \sigma_1 \rangle) - \text{FV}(A)$.

Następnie algorytm T oblicza $\langle C_2, \sigma_2 \rangle = T(A, M_2)$. I tym razem z założenia indukcyjnego obliczy się to prawidłowo i dla pewnego R_2 będzie zachodziło: $R_2 A \succ A'$ oraz $R_2 \text{Gen}(A, \langle C_2, \sigma_2 \rangle) \succ \langle D_2, \tau_2 \rangle$. Ten drugi fakt inaczej wyrażony stanowi, że

$$R_2 \forall u_1 \dots u_m \langle C_2, \sigma_2 \rangle \succ \langle D_2, \tau_2 \rangle \quad (4.19)$$

— gdzie $\{u_1, \dots, u_m\} = \text{FV}(\langle C_2, \sigma_2 \rangle) - \text{FV}(A)$.

Jako wynikowy typ T zwróci $\sigma = \langle C_1 \cup C_2 \cup \{\sigma_1 = \sigma_2 \rightarrow b\}, b \rangle$, gdzie b to świeża zmienna.

Rozważmy wyrażenie $R_2 \forall u_1 \dots u_m \langle C_2, \sigma_2 \rangle$. W wyrażeniu tym R_2 działa tylko na zmienne z $\text{FV}(\langle C_2, \sigma_2 \rangle)$ wolne w A . Ponieważ $R_2 A \succ A'$ i $RA \succ A'$, to, z lematu o jednoznaczności podstawienia w semi-instancji, R_2 i R tak samo działają na zmiennych wolnych w A . A zatem (4.19) implikuje $R \forall u_1 \dots u_m \langle C_2, \sigma_2 \rangle \succ \langle D_2, \tau_2 \rangle$. Ponadto, na mocy lematu 4.10 (iii), $\{t_1, \dots, t_n\} \cap \{u_1, \dots, u_m\} = \emptyset$, a zatem z tych dwóch zależności oraz z (4.18) otrzymujemy, że $R \forall t_1 \dots t_n u_1 \dots u_m \langle C_1, \sigma_1, C_2, \sigma_2 \rangle \succ \langle D_1, \tau_1, D_2, \tau_2 \rangle$ (dla jasności wyводу stworzyliśmy tymczasowo nową klasę typów). A to po przekształceniu oraz skorzystaniu z faktu, że $\forall b. b \succ \pi$ oraz $b \notin \{t_1, \dots, t_n, u_1, \dots, u_m\}$ i $b \notin \text{FV}(\langle C_1, \sigma_1, C_2, \sigma_2 \rangle)$ daje: $R \forall t_1 \dots t_n u_1 \dots u_m b \langle C_1 \cup C_2 \cup \{\sigma_1 = \sigma_2 \rightarrow b\}, b \rangle \succ \langle D_1 \cup D_2 \cup \{\tau_1 = \tau_2 \rightarrow \pi\}, \pi \rangle$.

A że $b \notin \text{FV}(A)$, to z powyższego wynika, że $R \text{Gen}(A, \langle C_1 \cup C_2 \cup \{\sigma_1 = \sigma_2 \rightarrow b\}, b \rangle) \succ \tau$. Przy okazji mamy, że algorytm dokończy działanie bez błędu, bo zbiór $C_1 \cup C_2 \cup \{\sigma_1 = \sigma_2 \rightarrow b\}$ jest niesprzeczny, jako (semi)instancjonujący się do niesprzecznego z założenia zbioru warunków z typu τ . \square

- $M = \text{let } x = M_1 \text{ in } M_2$.

Asercję $\mathbf{ML}_c^s \triangleright A' \vdash \text{let } x = M_1 \text{ in } M_2 : \tau$ uzyskano przez (LET) z przesłanek: $\mathbf{ML}_c^s \triangleright A' \vdash M_1 : \langle D_1, \tau_1 \rangle$ oraz $\mathbf{ML}_c^s \triangleright A'_x \cup \{x : \text{Gen}(A', \langle D_1, \tau_1 \rangle)\} \vdash M_2 : \langle D_2, \tau_2 \rangle$.

Algorytm T najpierw oblicza $\langle C_1, \sigma_1 \rangle = T(A, M_1)$. Z założenia indukcyjnego, obliczenie takie wykona się poprawnie i dla pewnego podstawienia R_1 będzie zachodziło:

$$R_1 A \succ A' \quad (4.20)$$

oraz

$$R_1 \text{Gen}(A, \langle C_1, \sigma_1 \rangle) \succ \langle D_1, \tau_1 \rangle. \quad (4.21)$$

Następnie algorytm T oblicza $\langle C_2, \sigma_2 \rangle = T(A_x \cup \{x : \text{Gen}(A, \langle C_1, \sigma_1 \rangle)\}, M_2)$. Z (4.20) i (4.21) na podstawie lematu 4.13 mamy, że $R_1 \text{Gen}(A, \langle C_1, \sigma_1 \rangle) \succ \text{Gen}(A', \langle D_1, \tau_1 \rangle)$. A zatem $R_1(A_x \cup \{x : \text{Gen}(A, \langle C_1, \sigma_1 \rangle)\}) \succ A'_x \cup \{x : \text{Gen}(A', \langle D_1, \tau_1 \rangle)\}$, czyli $A'_x \cup \{x : \text{Gen}(A', \langle D_1, \tau_1 \rangle)\}$ jest semi-instancją $A_x \cup \{x : \text{Gen}(A, \langle C_1, \sigma_1 \rangle)\}$. A zatem znowu możemy użyć założenia indukcyjnego, otrzymując, że $\langle C_2, \sigma_2 \rangle$ zostanie obliczone bez błędu i dla pewnego podstawienia R_2 będzie zachodziło:

$$R_2(A_x \cup \{x : \text{Gen}(A, \langle C_1, \sigma_1 \rangle)\}) \succ A'_x \cup \{x : \text{Gen}(A', \langle D_1, \tau_1 \rangle)\} \quad (4.22)$$

oraz

$$R_2 \text{Gen}(A_x \cup \{x : \text{Gen}(A, \langle C_1, \sigma_1 \rangle)\}, \langle C_2, \sigma_2 \rangle) \succ \langle D_2, \tau_2 \rangle. \quad (4.23)$$

Z faktu (4.22) wynika w szczególności, że $R_2 \text{Gen}(A, \langle C_1, \sigma_1 \rangle) \succ \text{Gen}(A', \langle D_1, \tau_1 \rangle)$, a więc z definicji \succ także:

$$R_2 \text{Gen}(A, \langle C_1, \sigma_1 \rangle) \succ \langle D_1, \tau_1 \rangle. \quad (4.24)$$

Niech $A_1 = \{x : \text{Gen}(A, \langle C_1, \sigma_1 \rangle)\}$, $A_2 = \{x : \langle D_1, \tau_1 \rangle\}$. Z (4.24) zachodzi $R_2 A_1 \succ A_2$, zaś z (4.21) zachodzi $R_1 A_1 \succ A_2$. Zatem, na mocy lematu o jednoznaczności podstawienia w semi-instancji, mamy: $\forall v \in \text{FV}(A_1). R_1(v) = R_2(v)$. No, ale $\text{FV}(A_1) = \text{FV}(\langle C_1, \sigma_1 \rangle) \cap \text{FV}(A)$. A zatem otrzymujemy, że:

$$\forall v \in \text{FV}(\langle C_1, \sigma_1 \rangle) \cap \text{FV}(A). R_1(v) = R_2(v). \quad (4.25)$$

Niech R będzie podstawieniem zdefiniowanym następująco:

$$R(t) = \begin{cases} R_1(t) & \text{dla } t \in \text{FV}(A(x)) - \text{FV}(A_x), \\ R_2(t) & \text{w przeciwnym przypadku.} \end{cases}$$

Ponieważ z (4.22): $R_2 A_x \succ A'_x$, zaś z (4.20): $R_1 A_x \succ A'_x$, to z lematu o jednoznaczności podstawienia w semi-instancji otrzymujemy, że $\forall v \in \text{FV}(A_x). R(v) = R_1(v)$. Ponadto, z definicji R : $\forall v \in \text{FV}(A(x)) - \text{FV}(A_x). R(v) = R_1(v)$. A zatem, reasumując,

$$\forall v \in \text{FV}(A). R(v) = R_1(v). \quad (4.26)$$

Czyli, skoro $R_1 A \succ A'$, to również $RA \succ A'$.

Fakt (4.23), inaczej zapisany, brzmi: $R_2 \forall t_1 \dots t_n \langle C_2, \sigma_2 \rangle \succ \langle D_2, \tau_2 \rangle$, gdzie $\{t_1, \dots, t_n\} = \text{FV}(\langle C_2, \sigma_2 \rangle) - \text{FV}(A_x \cup \{x : \text{Gen}(A, \langle C_1, \sigma_1 \rangle)\})$. Rozpatrzmy wyrażenie $R_2 \forall t_1 \dots t_n \langle C_2, \sigma_2 \rangle$. Podstawienie R różni się od R_2 tylko dla zmiennych $v \in \text{FV}(A(x)) - \text{FV}(A_x)$, wówczas bowiem $R(v) = R_1(v)$. No, ale jeśli taka zmienna $v \in \text{FV}(\langle C_2, \sigma_2 \rangle)$, to zachodzi również $v \in \text{FV}(\langle C_1, \sigma_1 \rangle)$. (A to dlatego, że $\langle C_2, \sigma_2 \rangle = \text{T}(A_x \cup \{x : \text{Gen}(A, \langle C_1, \tau_1 \rangle)\}, M_2)$, więc na mocy lematu 4.10 (i), w $\langle C_2, \sigma_2 \rangle$ mają prawo pojawić się tylko te zmienne z $\text{FV}(A(x)) - \text{FV}(A_x)$, które występują w $\text{Gen}(A, \langle C_1, \tau_1 \rangle)$.) A zatem, z (4.25), dla takiej zmiennej v : $R_1(v) = R_2(v)$, czyli znów R działa tak samo, jak R_2 . Reasumując, $R_2 \forall t_1 \dots t_n \langle C_2, \sigma_2 \rangle = R \forall t_1 \dots t_n \langle C_2, \sigma_2 \rangle$ i mamy:

$$R \forall t_1 \dots t_n \langle C_2, \sigma_2 \rangle \succ \langle D_2, \tau_2 \rangle. \quad (4.27)$$

Dodatkowo wiemy, że $t_1, \dots, t_n \notin \text{FV}(A)$, bo z lematu 4.10 (iv), zmienne, po których można generalizować w zwróconym typie, nie kolidują z żadnymi innymi.

Fakt (4.21), inaczej zapisany, brzmi: $R_1 \forall u_1 \dots u_m \langle C_1, \sigma_1 \rangle \succ \langle D_1, \tau_1 \rangle$, gdzie $\{u_1, \dots, u_m\} = \text{FV}(\langle C_1, \sigma_1 \rangle) - \text{FV}(A)$. A zatem jedyne zmienne wolne w $\forall u_1 \dots u_m \langle C_1, \sigma_1 \rangle$ to zmienne należące do $\text{FV}(A)$. Z własności (4.26) mamy więc $R_1 \forall u_1 \dots u_m \langle C_1, \sigma_1 \rangle = R \forall u_1 \dots u_m \langle C_1, \sigma_1 \rangle$ i mamy:

$$R \forall u_1 \dots u_m \langle C_1, \sigma_1 \rangle \succ \langle D_1, \tau_1 \rangle \quad (4.28)$$

Rozpatrujemy dwa przypadki:

- (i) x – wolne w M_2 . Wówczas $\tau = \langle D_2, \tau_2 \rangle$, zaś algorytm T zwraca $\langle C_2, \sigma_2 \rangle$. Z (4.27) zachodzi: $R \forall t_1 \dots t_n \langle C_2, \sigma_2 \rangle \succ \langle D_2, \tau_2 \rangle$ i $t_1, \dots, t_n \notin \text{FV}(A)$ (patrz uzasadnienie pod (4.27)). Stąd bezpośrednio wynika, że $R \text{Gen}(A, \langle C_2, \sigma_2 \rangle) \succ \langle D_2, \tau_2 \rangle$.
- (ii) x – nie wolne w M_2 . Wówczas $\tau = \langle D_1 \cup D_2, \tau_2 \rangle$, zaś algorytm T zwraca $\langle C_1 \cup C_2, \sigma_2 \rangle$. Z lematu 4.10 (iii), $\{t_1, \dots, t_n\} \cap \{u_1, \dots, u_m\} = \emptyset$, a zatem z (4.27) oraz z (4.28) otrzymujemy: $R \forall t_1 \dots t_n u_1 \dots u_m \langle C_1 \cup C_2, \sigma_2 \rangle \succ \langle D_1 \cup D_2, \tau_2 \rangle$ i $t_1, \dots, t_n \notin \text{FV}(A)$ (patrz uzasadnienie pod (4.27)) oraz $u_1, \dots, u_m \notin \text{FV}(A)$. Stąd bezpośrednio wynika, że $R \text{Gen}(A, \langle C_1 \cup C_2, \sigma_2 \rangle) \succ \langle D_1 \cup D_2, \tau_2 \rangle$. ■

Można mieć zastrzeżenia co do długości dowodu powyższego twierdzenia. Wątpliwe jest jednak, czy twierdzenie to daje się udowodnić znacznie prościej i krócej.

Z udowodnionych własności algorytmu T można wyprowadzić własność istnienia typu głównego dla systemu \mathbf{ML}_c^s .

Twierdzenie 4.15 (Własność typu głównego) *Jeśli term M jest typowalny w środowisku A przy pomocy reguł systemu \mathbf{ML}_c^s , to istnieje typ główny dla M w A : jest to typ zwrócony przez algorytm T dla danych (A, M) .*

Dowód: Niech $\mathbf{ML}_C^S \triangleright A \vdash M : \tau$. To wówczas, z twierdzenia 4.14 (o pełności algorytmu T), $T(A, M)$ wykona się poprawnie zwracając pewien typ σ taki, że na mocy twierdzenia 4.9 (o poprawności algorytmu T), $\mathbf{ML}_C^S \triangleright A \vdash M : \sigma$. Z twierdzenia 4.14 wiemy również, że istnieje takie podstawienie R , że $RA \succ A$ oraz $R\text{Gen}(A, \sigma) \succ \tau$. Skoro $RA \succ A$ oraz $IA \succ A$ (gdzie I to podstawienie identycznościowe), to z lematu o jednoznaczności podstawienia w semi-instancji: $\forall v \in \text{FV}(A). R(v) = I(v) = v$ — czyli, że R nie działa na zmiennych z $\text{FV}(A)$. Stąd $\text{Gen}(A, \sigma) = R\text{Gen}(A, \sigma) \succ \tau$, a zatem σ jest typem głównym dla M w A . ■

Zauważmy, że w powyższym twierdzeniu nie zakłada się, że term M jest zamknięty. Wielu autorów (na przykład Benke w [2] — system BC_2) proponuje systemy typów dla języków z podtypami posiadające własność typu głównego ograniczoną do termów zamkniętych.

Dla porządku, na zakończenie podamy jeszcze jedno twierdzenie:

Twierdzenie 4.16 *Problem typowości termu w \mathbf{ML}_C^S jest rozstrzygalny.*

Dowód: Dla danego termu M wystarczy uruchomić $T(A, M)$, gdzie $A = \{x_1 : \forall t \langle \emptyset, t \rangle, \dots, x_n : \forall t \langle \emptyset, t \rangle\}$, zaś x_1, \dots, x_n to wszystkie wolne zmienne z M . Z punktu (ii) lematu 4.3 wynika, że wystarczy rozważać środowiska z parami tylko dla zmiennych wolnych w M , z punktu (iv) lematu 4.3 wynika, że wystarczy rozważać środowiska z typami o pustych warunkach, zaś z lematu 4.4 wynika, że wystarczy rozważać typ $\forall t \langle \emptyset, t \rangle$, jako instancjonujący się do każdego innego typu o pustych warunkach. ■

Warto zauważyć, że w praktycznej implementacji algorytmu T należałoby rozważyć zastąpienie sprawdzania niesprzeczności warunków w każdej fazie działania algorytmu sprawdzeniem niesprzeczności warunków jedynie w typie końcowym. Można by tak zrobić, ponieważ ewentualne sprzeczności propagują się.

Rozdział 5

Porównanie systemu \mathbf{I}_c^s z systemem \mathbf{ML}_c^s

W tym rozdziale będziemy próbowali wykazać, że systemy \mathbf{I}_c^s i \mathbf{ML}_c^s mają podobną siłę typowania. Najpierw udowodnimy, że każdy term typowalny w systemie \mathbf{ML}_c^s po zamianie instrukcji **let** na odpowiadające im konstrukcje aplikacji i abstrakcji staje się termem typowalnym w systemie \mathbf{I}_c^s . Następnie zastanowimy się, które termy typowalne w systemie \mathbf{I}_c^s po zastąpieniu możliwie dużej liczby aplikacji instrukcjami **let** będą typowalne również w systemie \mathbf{ML}_c^s .

Pojęciem przewijającym się w tym rozdziale będzie generalizacja zbioru typów prostych.

Definicja 5.1

- (i) Generalizacją zbioru typów prostych T jest każdy typ $\sigma \in \mathbf{T}_{\forall C}$ taki, że dla każdego $\tau \in T$: $\sigma \succ \langle C, \tau \rangle$ i zbiór warunków C jest tożsamościowy.
- (ii) Jeśli $(\bigwedge_{i \in I} \tau_i) \in \mathbf{T}_1$, to zbiór generalizacji typu $(\bigwedge_{i \in I} \tau_i)$, oznaczany przez $CG(\bigwedge_{i \in I} \tau_i)$, definiujemy jako zbiór wszystkich generalizacji zbioru $\{\tau_i \mid i \in I\}$.
- (iii) Niech A będzie środowiskiem typu $\mathbf{T}_{\forall C}$, zaś A' – środowiskiem typu \mathbf{T}_1 . Wówczas A jest generalizacją A' , co zapisujemy przez $A \preceq A'$, wtedy i tylko wtedy, gdy dla każdego $x \in \mathbf{dom}(A')$: $x \in \mathbf{dom}(A)$ i $A(x) \in CG(A'(x))$. ■

Po pierwsze — zauważmy, że każdy zbiór typów prostych ma swoją generalizację: typ $\forall t \langle \emptyset, t \rangle$, a zatem dla każdego typu $\sigma \in \mathbf{T}_1$, zbiór $CG(\sigma)$ jest niepusty. Stąd wniosek, że dla każdego środowiska A' typu \mathbf{T}_1 istnieje takie środowisko A typu $\mathbf{T}_{\forall C}$, że $A \preceq A'$.

Ponadto — zauważmy, że jeśli $A \preceq A'$ i $A \preceq A''$, to $A \preceq (A' + A'')$, oraz, że jeśli $A_x \preceq A'$, to także $A \preceq A'$. Z własności tych będziemy korzystać dowodach lematów w dalszej części niniejszego rozdziału.

Pojęcie generalizacji zbioru typów prostych nie jest niczym nowym i jest spotykane w literaturze (patrz na przykład [6]). My jednak wprowadzamy nową, bardziej ogólną jego wersję — u nas generalizacją może być typ polimorficzny z warunkami.

5.1 Od \mathbf{ML}_c^s do \mathbf{I}_c^s

Idea dowodu, że system \mathbf{I}_c^s nie ustępuje systemowi \mathbf{ML}_c^s pod względem siły typowania, jest następująca: w systemie \mathbf{ML}_c^s typy w środowiskach są kwantyfikowane, a zatem mogą instancjonować się do nieskończenie wielu różnych typów prostych. Jednak do otypowania termów lambda-rachunku wystarczy skończona liczba typów dla każdej zmiennej, bo każda zmienna występuje w termie skończoną liczbę razy. A zatem typ kwantyfikowany w środowisku można zastąpić typem intersekcyjnym o odpowiednio wielu członach — aby tylko wśród tych członów były wszystkie typy, do których uprzednio instancjonowano typ kwantyfikowany. Szerzej o tych zagadnieniach pisze między innymi van Bakel w pracy [10] (strony 75-77).

Wśród termów ML jest konstrukcja **let**, której nie uwzględnia system \mathbf{I}_c^s . Dlatego najpierw musimy zamienić wszystkie wystąpienia **let** na odpowiadające im konstrukcje aplikacji i abstrakcji:

Definicja 5.2 Definiujemy funkcję **unml** ze zbioru termów M_{ml} do zbioru termów M_λ :

- (i) $\mathbf{unml}(c) = c$ (dla $c \in \{c_\perp, c^\top\}$),
- (ii) $\mathbf{unml}(x) = x$,
- (iii) $\mathbf{unml}(\lambda x N) = \lambda x \mathbf{unml}(N)$,
- (iv) $\mathbf{unml}(M_1 M_2) = \mathbf{unml}(M_1) \mathbf{unml}(M_2)$,
- (v) $\mathbf{unml}(\mathbf{let} \ x = M_1 \ \mathbf{in} \ M_2) = (\lambda x \mathbf{unml}(M_2))(\mathbf{unml}(M_1))$. ■

Teraz możemy już podać kluczowe w tym podrozdziale twierdzenie.

Twierdzenie 5.3 Jeśli $\mathbf{ML}_c^s \triangleright A \vdash M : \langle C, \tau \rangle$ i C jest tożsamościowym zbiorem warunków, to dla pewnego A' takiego, że $A \preceq A'$, zachodzi $\mathbf{I}_c^s \triangleright A' \vdash \mathbf{unml}(M) : \tau$.

Dowód: Dowód indukcyjny ze względu na budowę termu M . Rozpatrujemy następujące przypadki:

- (i) $M = c^\perp$. Mamy $\mathbf{ML}_c^s \triangleright A \vdash c^\perp : \langle \{\perp \leq_0 \tau\}, \tau \rangle$ i zbiór $\{\perp \leq_0 \tau\}$ jest tożsamościowy, a zatem $\tau = \perp$ lub $\tau = \top$. Jeśli $\tau = \perp$, to zauważmy, że $\mathbf{I}_c^s \triangleright \emptyset \vdash c^\perp : \perp$ i $A \preceq \emptyset$, a zatem mamy to, co chcieliśmy wykazać. Jeśli $\tau = \top$, to dowód analogiczny.
Dla $M = c^\top$ dowód analogiczny.

- (ii) $M = x$. Asercję $\mathbf{ML}_C^s \triangleright A \vdash x : \langle C, \tau \rangle$ otrzymano przez (VAR), a zatem $A(x) = \sigma_x \succ \langle C_1, \tau_1 \rangle$ i $\langle C, \tau \rangle = \langle C_1 \cup \{\tau_1 \leq_0 \tau\}, \tau \rangle$. Skoro $C = C_1 \cup \{\tau_1 \leq_0 \tau\}$ jest zbiorem tożsamościowym, to C_1 i $\{\tau_1 \leq_0 \tau\}$ są również zbiorami tożsamościowymi.

Niech $A' = \{x : \tau_1\}$. Wówczas $A \preceq A'$, bo $\sigma_x \succ \langle C_1, \tau_1 \rangle$ i C_1 jest tożsamościowy. Mamy również $\mathbf{I}_C^s \triangleright \{x : \tau_1\} \vdash x : \tau$, bo $\tau_1 \leq_0 \tau$.

- (iii) $M = \lambda x N$. Wówczas τ musi być postaci $\tau_1 \rightarrow \tau_2$ i asercję $\mathbf{ML}_C^s \triangleright A \vdash \lambda x N : \langle C, \tau_1 \rightarrow \tau_2 \rangle$ uzyskano przez regułę (ABS) systemu \mathbf{ML}_C^s z przesłanki $\mathbf{ML}_C^s \triangleright A_x \cup \{x : \langle \emptyset, \tau_1 \rangle\} \vdash N : \langle C, \tau_2 \rangle$, i C tożsamościowy. Z założenia indukcyjnego mamy zatem:

$$\mathbf{I}_C^s \triangleright A' \vdash \mathbf{unml}(N) : \tau_2 \quad \text{i} \quad A_x \cup \{x : \langle \emptyset, \tau_1 \rangle\} \preceq A'. \quad (5.1)$$

Jeśli $x \in \mathbf{dom}(A')$, to z definicji \preceq wynika, że $A'(x) = \tau_1$. Jeśli $x \notin \mathbf{dom}(A)$, to na mocy lematu 3.4 z (5.1) będziemy również mieli $\mathbf{I}_C^s \triangleright A' \cup \{x : \tau_1\} \vdash \mathbf{unml}(N) : \tau_2$. W obu przypadkach mamy więc:

$$\mathbf{I}_C^s \triangleright A'_x \cup \{x : \tau_1\} \vdash \mathbf{unml}(N) : \tau_2.$$

a stąd przez regułę (ABS) systemu \mathbf{I}_C^s i lemat 3.5 otrzymujemy: $\mathbf{I}_C^s \triangleright A'_x \vdash \lambda x \mathbf{unml}(N) : \tau_1 \rightarrow \tau_2$. Ponadto, skoro $A_x \cup \{x : \langle \emptyset, \tau_1 \rangle\} \preceq A'$, to $A_x \preceq A'_x$.

- (iv) $M = M_1 M_2$. Wówczas C , musi być postaci $C_1 \cup C_2 \cup \{\tau_1 = \tau_2 \rightarrow \tau\}$ i skorzystano z przesłanek: $\mathbf{ML}_C^s \triangleright A \vdash M_1 : \langle C_1, \tau_1 \rangle$ oraz $\mathbf{ML}_C^s \triangleright A \vdash M_2 : \langle C_2, \tau_2 \rangle$. Skoro C jest tożsamościowy, to C_1 i C_2 są również tożsamościowe, a zatem z założenia indukcyjnego otrzymujemy, że $\mathbf{I}_C^s \triangleright A_1 \vdash \mathbf{unml}(M_1) : \tau_1$, gdzie $A \preceq A_1$, oraz $\mathbf{I}_C^s \triangleright A_2 \vdash \mathbf{unml}(M_2) : \tau_2$, gdzie $A \preceq A_2$. Mamy wówczas także $A \preceq A_1 + A_2$ oraz z osłabiania: $\mathbf{I}_C^s \triangleright A_1 + A_2 \vdash \mathbf{unml}(M_1) : \tau_1$ i $\mathbf{I}_C^s \triangleright A_1 + A_2 \vdash \mathbf{unml}(M_2) : \tau_2$. Skoro C - tożsamościowy, to $\tau_1 = \tau_2 \rightarrow \tau$, a zatem z tych dwóch asercji na mocy reguły (APP) systemu \mathbf{I}_C^s otrzymujemy, że $\mathbf{I}_C^s \triangleright A_1 + A_2 \vdash \mathbf{unml}(M_1)\mathbf{unml}(M_2) : \tau$.

- (v) $M = \mathbf{let} \ x = M_1 \ \mathbf{in} \ M_2$. Wówczas $\langle C, \tau \rangle$ jest postaci $\langle C_x \cup C_2, \tau_2 \rangle$, gdzie $C_x \cup C_2$ jest tożsamościowy, zaś

$$C_x = \begin{cases} \emptyset & \text{jeśli } x \text{ wolny w } M_2, \\ C_1 & \text{w przeciwnym przypadku} \end{cases}$$

i asercję dla \mathbf{let} otrzymano przez (LET) z przesłanek:

$$\mathbf{ML}_C^s \triangleright A \vdash M_1 : \langle C_1, \tau_1 \rangle \quad (5.2)$$

oraz z $\mathbf{ML}_C^S \triangleright A_x \cup \{x : \text{Gen}(A, \langle C_1, \tau_1 \rangle)\} \vdash M_2 : \langle C_2, \tau_2 \rangle$. Z tej ostatniej asercji przy pomocy założenia indukcyjnego uzyskujemy:

$$\mathbf{I}_c^S \triangleright A' \vdash \mathbf{unml}(M_2) : \tau_2 \quad \text{oraz} \quad A_x \cup \{x : \text{Gen}(A, \langle C_1, \tau_1 \rangle)\} \preceq A'. \quad (5.3)$$

Zauważmy, że z powyższego wynika $A \preceq A'_x$.

Rozpatrujemy dwa przypadki:

- a) $x \in \mathbf{dom}(A')$. Wówczas niech $A'(x) = (\bigwedge_{i \in I} \sigma_i)$. Mamy więc $\mathbf{I}_c^S \triangleright A'_x \cup \{x : (\bigwedge_{i \in I} \sigma_i)\} \vdash \mathbf{unml}(M_2) : \tau_2$, a stąd przez regułę (ABS) systemu \mathbf{I}_c^S i lemat 3.5 otrzymujemy:

$$\mathbf{I}_c^S \triangleright A'_x \vdash \lambda x \mathbf{unml}(M_2) : \left(\bigwedge_{i \in I} \sigma_i \right) \rightarrow \tau_2. \quad (5.4)$$

Ponieważ $A_x \cup \{x : \text{Gen}(A, \langle C_1, \tau_1 \rangle)\} \preceq A'$ i $A'(x) = (\bigwedge_{i \in I} \sigma_i)$, to z definicji \preceq zachodzi $\text{Gen}(A, \langle C_1, \tau_1 \rangle) \in \text{CG}(\bigwedge_{i \in I} \sigma_i)$, czyli z definicji CG mamy:

$$(\forall i \in I) \text{Gen}(A, \langle C_1, \tau_1 \rangle) \succ \langle C_i, \tau_i \rangle \quad (C_i \text{ tożsamościowe}). \quad (5.5)$$

Skoro (5.2) głosi, że $\mathbf{ML}_C^S \triangleright A \vdash M_1 : \langle C_1, \tau_1 \rangle$, to stąd i z (5.5) na mocy twierdzenia 4.6 (ii) otrzymujemy, że dla każdego $i \in I$: $\mathbf{ML}_C^S \triangleright A \vdash M_1 : \langle C_i, \sigma_i \rangle$. Stąd przez założenie indukcyjne otrzymujemy dla każdego $i \in I$: $\mathbf{I}_c^S \triangleright A_i \vdash \mathbf{unml}(M_1) : \sigma_i$, gdzie $A \preceq A_i$. To przez osłabianie daje dla każdego $i \in I$: $\mathbf{I}_c^S \triangleright A'_x + \sum_{i \in I} A_i \vdash \mathbf{unml}(M_1) : \sigma_i$. Z kolei (5.4) przez osłabianie daje, że $\mathbf{I}_c^S \triangleright A'_x + \sum_{i \in I} A_i \vdash \lambda x \mathbf{unml}(M_2) : (\bigwedge_{i \in I} \sigma_i) \rightarrow \tau_2$. Te dwie asercje na mocy reguły (APP) systemu \mathbf{I}_c^S dają, że $\mathbf{I}_c^S \triangleright A'_x + \sum_{i \in I} A_i \vdash (\lambda x \mathbf{unml}(M_2)) \mathbf{unml}(M_1) : \tau_2$. Oraz, skoro $A \preceq A'_x$ i $\forall i \in I. A \preceq A_i$, to $A \preceq A'_x + \sum_{i \in I} A_i$.

- b) $x \notin \mathbf{dom}(A')$. Skoro z (5.3): $\mathbf{I}_c^S \triangleright A' \vdash \mathbf{unml}(M_2) : \tau_2$ i $x \notin \mathbf{dom}(A')$, to oznacza to, że x nie występuje wolno w $\mathbf{unml}(M_2)$ (łatwy lemat dla \mathbf{I}_c^S ; dowód indukcyjny). A zatem $C_x = C_1$, czyli skoro $C = C_x \cup C_2$ jest tożsamościowy, to C_1 jest również tożsamościowy.

Z (5.2) mamy zatem na mocy założenia indukcyjnego:

$$\mathbf{I}_c^S \triangleright A_2 \vdash \mathbf{unml}(M_1) : \tau_1 \quad \text{gdzie} \quad A \preceq A_2. \quad (5.6)$$

Ponadto, z (5.3), ponieważ $x \notin \mathbf{dom}(A')$, to z lematu 3.4 mamy: $\mathbf{I}_c^S \triangleright A'_x \cup \{x : \tau_1\} \vdash \mathbf{unml}(M_2) : \tau_2$, a stąd przez regułę (ABS) systemu \mathbf{I}_c^S : $\mathbf{I}_c^S \triangleright A' \vdash \lambda x \mathbf{unml}(M_2) : \tau_1 \rightarrow \tau_2$. Stąd przez osłabianie mamy $\mathbf{I}_c^S \triangleright A' + A_2 \vdash \lambda x \mathbf{unml}(M_2) : \tau_1 \rightarrow \tau_2$. Z kolei z (5.6) przez osłabianie mamy: $\mathbf{I}_c^S \triangleright A' + A_2 \vdash \mathbf{unml}(M_1) : \tau_1$. Te dwie asercje na mocy reguły (APP) systemu \mathbf{I}_c^S dają: $\mathbf{I}_c^S \triangleright A' + A_2 \vdash (\lambda x \mathbf{unml}(M_2)) \mathbf{unml}(M_1) : \tau_2$. Ponadto, skoro $A \preceq A'_x$ i $A'_x = A'$, to $A \preceq A'$. A że także $A \preceq A_2$, to $A \preceq A' + A_2$. ■

Pora na podsumowanie.

Twierdzenie 5.4 *Jeśli M jest termem typowalnym w \mathbf{ML}_c^s , to $\text{unml}(M)$ jest termem typowalnym w \mathbf{I}_c^s .*

Dowód: Skoro M jest typowalny w \mathbf{ML}_c^s , to niech $\mathbf{ML}_c^s \triangleright A \vdash M : \langle C, \sigma \rangle$. Ponieważ, z definicji systemu \mathbf{ML}_c^s , C nie może być sprzeczny, to ma rozwiązanie S . Z substytucyjności dla \mathbf{ML}_c^s mamy wówczas: $\mathbf{ML}_c^s \triangleright SA \vdash M : \langle SC, S\sigma \rangle$ i SC jest tożsamościowy. A zatem z twierdzenia 5.3 mamy $\mathbf{I}_c^s \triangleright A' \vdash \text{unml}(M) : S\sigma$ dla pewnego A' . ■

5.2 Od \mathbf{I}_c^s do \mathbf{ML}_c^s

Ta część niniejszego rozdziału będzie bardziej skomplikowana. Pytamy się, czy termy typowalne w \mathbf{I}_c^s będą typowalne również w \mathbf{ML}_c^s . W ogólności tak nie będzie, bo polimorfizm w \mathbf{I}_c^s jest osiąganym dzięki polimorficznej abstrakcji (reguła (ABS) dopuszcza wprowadzenie do środowiska typu polimorficznego), natomiast w \mathbf{ML}_c^s abstrakcja jest monomorficzna, a polimorfizm jest osiąganym przy pomocy konstrukcji **let**.

Pierwsze zatem, co powinniśmy zrobić, to próbować zamienić jak najwięcej aplikacji odpowiadającymi im konstrukcjami **let**. Najbardziej naturalna wydaje się konwersja przy pomocy funkcji **ml** zdefiniowanej poniżej (działającej odwrotnie do zdefiniowanej w poprzednim podrozdziale funkcji **unml**):

Definicja 5.5 *Definiujemy funkcję **ml** ze zbioru termów M_λ do zbioru termów M_{ml} :*

$$(i) \quad \mathbf{ml}(c) = c \quad (\text{dla } c \in \{c_\perp, c^\top\}),$$

$$(ii) \quad \mathbf{ml}(x) = x,$$

$$(iii) \quad \mathbf{ml}(\lambda x N) = \lambda x \mathbf{ml}(N),$$

$$(iv) \quad \mathbf{ml}(M_1 M_2) = \begin{cases} \mathbf{let } x = \mathbf{ml}(M_2) \mathbf{ in } \mathbf{ml}(M_1) & \text{jeśli } M_2 = \lambda x N, \\ \mathbf{ml}(M_1) \mathbf{ml}(M_2) & \text{w przeciwnym przypadku.} \end{cases}$$

■

Nasuwa się pytanie, czy rzeczywiście wystarczy zastosować funkcję **ml**, żeby term typowalny w \mathbf{I}_c^s stał się termem typowalnym w \mathbf{ML}_c^s .

Odpowiedź jest negatywna. Po pierwsze — zauważmy, że funkcja **ml** eliminuje tylko abstrakcje tworzące β -redeksy (innymi słowy, abstrakcje z towarzyszącymi im aplikacjami), czyli „początkowe“ abstrakcje z wyrażeniami postaci $(\lambda x N)M$. Jeśli zatem w termie będą nagromadzone abstrakcje bez odpowiadających im aplikacji, i takie, że do ich otypowania w systemie \mathbf{I}_c^s skorzystano z możliwości polimorficznej abstrakcji, to po zadziałaniu funkcją

\mathbf{ml} abstrakcje takie nie znikną i do ich otypowania nadal będziemy potrzebowali polimorficznej abstrakcji, której w systemie \mathbf{ML}_c^s nie ma. Przykładem termu, w którym nie wyeliminujemy abstrakcji na rzecz konstrukcji \mathbf{let} , w dodatku wymagającego do otypowania polimorficznej abstrakcji, jest term $\lambda x.xx$. W \mathbf{I}_c^s ma on typ na przykład $(t_1 \wedge (t_1 \rightarrow t_2)) \rightarrow t_2$. Wyprowadzenie tego typu polegało na wprowadzeniu do środowiska pary $(x : (t_1 \wedge (t_1 \rightarrow t_2)))$, co pozwoliło na otypowanie termu xx . W systemie \mathbf{ML}_c^s w celu wyprowadzenia typu dla $\lambda x.xx$ do środowiska możemy wprowadzić jedynie parę monomorficzną, na przykład $(x : \langle \emptyset, b \rangle)$, a takie środowisko nie pozwoli na otypowanie termu xx .

Zauważmy ponadto, że nawet, jeśli abstrakcjom odpowiadają aplikacje, to z kolejności eliminacji abstrakcji przez funkcję \mathbf{ml} wynika, że aby wszystkie abstrakcje zostały usunięte, to muszą one być równomiernie wymieszane z aplikacjami. Przykładowo, założmy, że term M jest postaci $(\dots (\lambda x_1 \dots x_n N) N_1 \dots) N_n$. Wówczas $\mathbf{ml}(M)$ jest równe

$$(\dots ((\mathbf{let} \ x_1 = \mathbf{ml}(N_1) \ \mathbf{in} \ \lambda x_2 \dots x_n \mathbf{ml}(N)) \ \mathbf{ml}(N_2)) \dots) \ \mathbf{ml}(N_n).$$

Jeśli do otypowania podtermu $\lambda x_2 \dots x_n N$ w systemie \mathbf{I}_c^s użyto polimorficznej abstrakcji, to będzie ona prawdopodobnie również niezbędna do otypowania termu $\lambda x_2 \dots x_n \mathbf{ml}(N)$ w systemie \mathbf{ML}_c^s , a tam polimorficznej abstrakcji nie ma. Gdyby jednak term M był postaci $(\lambda x_1 \dots ((\lambda x_n N) N_n) \dots) N_1$, to wówczas $\mathbf{ml}(M)$ byłoby równe

$$\mathbf{let} \ x_1 = \mathbf{ml}(N_1) \ \mathbf{in} \ (\dots (\mathbf{let} \ x_n = \mathbf{ml}(N_n) \ \mathbf{in} \ \mathbf{ml}(N)) \dots).$$

W tym drugim przypadku rzeczywiście wszystkie abstrakcje x_1, \dots, x_n zostały zastąpione instrukcjami \mathbf{let} . Stwarza to szansę na otypowanie termu $\mathbf{ml}(M)$ w \mathbf{ML}_c^s nawet wówczas, jeśli do otypowania termu M w \mathbf{I}_c^s wykorzystywano możliwości polimorficznej abstrakcji.

Podaliśmy nieformalnie pewne warunki konieczne do tego, by dla termu M , typowalnego w \mathbf{I}_c^s z wykorzystaniem polimorfizmu, term $\mathbf{ml}(M)$ był typowalny w \mathbf{ML}_c^s . Warto by jednak znaleźć warunek dostateczny. Jak się okazuje, można podać dwa kryteria, których jednoczesne spełnienie gwarantuje, że term typowalny w \mathbf{I}_c^s po konwersji funkcją \mathbf{ml} będzie typowalny w \mathbf{ML}_c^s . Po pierwsze, będzie to warunek, by term M był w tak zwanej postaci γ -nf (zdefiniowanej dalej), intuicyjnie zapewniającej, że w całym termie abstrakcje będą możliwie równomiernie wymieszane z aplikacjami. Po drugie, będzie to warunek, by term M miał w systemie \mathbf{I}_c^s wyprowadzalny typ z \mathbf{T}_0 , co intuicyjnie ma zapewnić, że abstrakcji nie będzie więcej, niż odpowiadających im aplikacji, a zatem po „wymieszaniu“ abstrakcji z aplikacjami przy pomocy γ -konwersji każdej abstrakcji będzie odpowiadała aplikacja.

Tyle tytułem wprowadzenia. Teraz pora na część formalną. W dużym stopniu przypomina ona część 2.5 pracy Jima [6], gdzie dowodzi on, że każdy

term typowalny w jego systemie typów intersekcyjnych \mathbf{I}_2^S jest również typowalny w systemie Λ_2^S . Wprowadził nie Jim pierwszy zastosował opisane poniżej chwyt (na przykład koncepcję funkcji **act**), ale wprowadzając je pozwolimy sobie powoływać się właśnie na pracę Jima.

Tak samo, jak Jim, przyjmujemy konwencję, że do końca tego rozdziału nie uznajemy termów za równoważne z dokładnością do nazw zmiennych związanych. Zakładamy również, że żadna zmienna nie jest związana w żadnym terminie więcej, niż raz, jak również, że zbiory zmiennych związanych i wolnych w każdym terminie są rozłączne. Następnie, podobnie jak Jim, wprowadzamy funkcję **act**:

Definicja 5.6 *Niech ϵ oznacza sekwencję pustą. Funkcja **act**, ze zbioru termów w sekwencji zmiennych (tak zwanych aktywnych w tych terminach), jest zdefiniowana indukcyjnie następującymi regułami:*

- (i) $\mathbf{act}(c) = \epsilon$ ($c \in \{c^\perp, c^\top\}$), oraz $\mathbf{act}(x) = \epsilon$,
- (ii) jeśli $\mathbf{act}(M) = x_1, \dots, x_n$, to $\mathbf{act}(\lambda y M) = y, x_1, \dots, x_n$,
- (iii) jeśli $\mathbf{act}(M) = y, x_1, \dots, x_n$ ($n \geq 0$), to $\mathbf{act}(MN) = x_1, \dots, x_n$,
- (iv) jeśli $\mathbf{act}(M) = \epsilon$, to $\mathbf{act}(MN) = \epsilon$. ■

Za Jimem definiujemy również pojęcia związane ze wspomnianą wcześniej γ -konwersją:

Definicja 5.7

- (i) γ jest regułą:

$$(\lambda x(\lambda y M))N \rightarrow \lambda y((\lambda x M)N).$$
- (ii) \rightarrow_γ jest rozszerzeniem reguły γ na nadterminy (tak samo, jak w przypadku α -konwersji, omawianej w rozdziale 2).
- (iii) γ -redksem jest każdy term pasujący do lewej strony reguły γ . Mówimy, że term M jest w postaci γ -normalnej, lub w postaci γ -nf, jeśli żaden podterm M nie jest γ -redksem. ■

Jim dowodzi, że reguła γ jest silnie normalizująca (czyli, że dla każdego termu, każdy ciąg γ -konwersji prowadzi do postaci γ -nf), że ma ona własność rombu, i wreszcie — że dla każdego termu postać γ -nf jest unikalna. Tym uzasadnia wprowadzenie funkcji γ -nf, która każdemu termowi M przypisuje γ -nf termu M .

Następnie Jim wprowadza lemat, który w odniesieniu do naszego systemu także obowiązuje:

Lemat 5.8 *Jeśli $\mathbf{act}(M) = x_1, \dots, x_n$ i $\mathbf{I}_C^S \triangleright A \vdash M : \sigma$, to σ jest postaci $\sigma_1 \rightarrow \dots \rightarrow \sigma_n \rightarrow \tau$, gdzie $\tau \in \mathbf{T}_0$.*

Dowód: Patrz Jim [6], dowód lematu 19. Dowód dla naszego systemu analogiczny, jak tamten. Obecność stałych w systemie nam nie przeszkadza, bo przypadek dla stałych dowodzimy tak samo, jak przypadek dla zmiennych. ■

Na zakończenie jeszcze jedno zapożyczenie z Jima.

Lemat 5.9 *Niech M będzie w γ -nf. Wówczas*

$$\text{act}(M) \neq \epsilon \text{ wtedy i tylko wtedy, gdy } M = \lambda y N \text{ dla pewnych } y, N.$$

Dowód: Patrz Jim [6], dowód lematu 21. ■

Teraz już możemy podać najważniejsze twierdzenie w tym podrozdziale.

Twierdzenie 5.10 *Jeśli $\mathbf{I}_c^s \triangleright A \vdash M : \sigma$, przy czym M jest w postaci γ -nf i $\sigma \in \mathbf{T}_0$, to dla dowolnego $B \preceq A$: $\mathbf{ML}_c^s \triangleright B \vdash \text{ml}(M) : \langle C, \sigma \rangle$, gdzie C jest pewnym tożsamościowym zbiorem warunków.*

Dowód: Dowód indukcyjny ze względu na budowę termu M . Rozważamy następujące przypadki:

(i) $M = c^\perp$. Jeśli $\sigma = \perp$, to zauważmy, że $\mathbf{ML}_c^s \triangleright B \vdash c^\perp : \langle \{\perp \leq_0 \perp\}, \perp \rangle$, dla dowolnego B . Jeśli $\sigma = \top$, to zauważmy, że $\mathbf{ML}_c^s \triangleright B \vdash c^\perp : \langle \{\perp \leq_0 \top\}, \top \rangle$, dla dowolnego B .
Dla $M = c^\top$ dowód analogiczny.

(ii) $M = x$. Asercję $\mathbf{I}_c^s \triangleright A \vdash x : \sigma$ uzyskano przez regułę (VAR) systemu \mathbf{I}_c^s , a zatem $A(x) = (\bigwedge_{i \in I} \tau_i)$ i dla pewnego $i_0 \in I$ zachodzi $\tau_{i_0} \leq_0 \sigma$.

Niech $B \preceq A$; wówczas $B(x) \in \text{CG}(A(x))$, czyli $B(x) \in \text{CG}(\bigwedge_{i \in I} \tau_i)$. A zatem, z definicji generalizacji, mamy w szczególności, że $B(x) \succ \langle C, \tau_{i_0} \rangle$ i C - tożsamościowy. Stąd, przez regułę (VAR) systemu \mathbf{ML}_c^s , otrzymujemy, że $B \vdash x : \langle C \cup \{\tau_{i_0} \leq_0 \sigma\}, \sigma \rangle$. No, i ponieważ C jest tożsamościowy oraz $\tau_{i_0} \leq_0 \sigma$, to $C \cup \{\tau_{i_0} \leq_0 \sigma\}$ jest również tożsamościowy.

(iii) $M = \lambda x N$. Wówczas σ jest postaci $\sigma_1 \rightarrow \sigma_2$ i asercję $\mathbf{I}_c^s \triangleright A \vdash \lambda x N : \sigma_1 \rightarrow \sigma_2$ otrzymano przez regułę (ABS) systemu \mathbf{I}_c^s z przesłanki $\mathbf{I}_c^s \triangleright A_x \cup \{x : \sigma_1\} \vdash N : \sigma_2$.

Zauważmy, że N jest w γ -nf. Niech $B \preceq A$. To wówczas $B_x \cup \{x : \langle \emptyset, \sigma_1 \rangle\} \preceq A_x \cup \{x : \sigma_1\}$. A zatem z założenia indukcyjnego otrzymujemy, że $\mathbf{ML}_c^s \triangleright B_x \cup \{x : \langle \emptyset, \sigma_1 \rangle\} \vdash \text{ml}(N) : \langle C, \sigma_2 \rangle$ i C jest tożsamościowy. Stąd przez regułę (ABS) systemu \mathbf{ML}_c^s otrzymujemy $\mathbf{ML}_c^s \triangleright B \vdash \lambda x \text{ml}(N) : \langle C, \sigma_1 \rightarrow \sigma_2 \rangle$.

- (iv) $M = (\lambda x M_1) M_2$. Asercją $\mathbf{I}_c^s \triangleright A \vdash (\lambda x M_1) M_2 : \sigma$ uzyskano przez reguły (APP) i (ABS) z przesłanek:

$$\mathbf{I}_c^s \triangleright A_x \cup \{x : (\bigwedge_{i \in I} \sigma_i)\} \vdash M_1 : \sigma \quad (5.7)$$

oraz $(\forall i \in I): \mathbf{I}_c^s \triangleright A \vdash M_2 : \sigma_i$. Niech $B \preceq A$. Wówczas ta druga asercja na mocy założenia indukcyjnego daje, że

$$(\forall i \in I) \quad \mathbf{ML}_c^s \triangleright B \vdash \mathbf{ml}(M_2) : \langle C_i, \sigma_i \rangle \quad (C_i \text{ tożsamościowe}). \quad (5.8)$$

Rozpatrujemy dwa przypadki:

- a) x – nie występuje wolno w M_1 . Niech $\pi \in \text{CG}(\bigwedge_{i \in I} \sigma_i)$; wówczas $B_x \cup \{x : \pi\} \preceq A_x \cup \{x : (\bigwedge_{i \in I} \sigma_i)\}$. Z założenia indukcyjnego z (5.7) otrzymujemy zatem $\mathbf{ML}_c^s \triangleright B_x \cup \{x : \pi\} \vdash \mathbf{ml}(M_1) : \langle C, \sigma \rangle$ i C jest tożsamościowy. No, ale ponieważ x nie jest wolny w M_1 , to z lematu 4.3 (iii) mamy także $\mathbf{ML}_c^s \triangleright B_x \cup \{x : \text{Gen}(B, \langle C_{i_0}, \sigma_{i_0} \rangle)\} \vdash \mathbf{ml}(M_1) : \langle C, \sigma \rangle$, dla dowolnie wybranego $i_0 \in I$. To wraz z (5.8) dla $i = i_0$ przez (LET) daje, że $B \vdash \mathbf{let } x = M_2 \text{ in } M_1 : \langle C \cup C_{i_0}, \sigma \rangle$ i $C \cup C_{i_0}$ jest tożsamościowy.
- b) x – występuje wolno w M_1 . Niech σ_x będzie typem głównym dla $\mathbf{ml}(M_2)$ w B . Będziemy wówczas mieli, że

$$\mathbf{ML}_c^s \triangleright B \vdash \mathbf{ml}(M_2) : \sigma_x \quad (5.9)$$

oraz z (5.8) i z definicji typu głównego będzie zachodziło $(\forall i \in I): \text{Gen}(B, \sigma_x) \succ \langle C_i, \sigma_i \rangle$. No, ale ponieważ dla każdego $i \in I$, zbiór warunków C_i jest tożsamościowy, to otrzymaliśmy, że $\text{Gen}(B, \sigma_x) \in \text{CG}(\bigwedge_{i \in I} \sigma_i)$. A zatem $B_x \cup \{x : \text{Gen}(B, \sigma_x)\} \preceq A_x \cup \{x : (\bigwedge_{i \in I} \sigma_i)\}$. Z założenia indukcyjnego (5.7) nam zatem daje: $\mathbf{ML}_c^s \triangleright B_x \cup \{x : \text{Gen}(B, \sigma_x)\} \vdash \mathbf{ml}(M_1) : \langle C, \sigma \rangle$ i warunki C są tożsamościowe. To wraz z (5.9) przez (LET) daje, że $\mathbf{ML}_c^s \triangleright B \vdash \mathbf{let } x = M_2 \text{ in } M_1 : \langle C, \sigma \rangle$.

- (v) $M = M_1 M_2$, gdzie M_1 nie jest abstrakcją. Asercją $\mathbf{I}_c^s \triangleright A \vdash M_1 M_2 : \sigma$ uzyskano przez regułę (APP) systemu \mathbf{I}_c^s z przesłanek

$$\mathbf{I}_c^s \triangleright A \vdash M_1 : \sigma' \rightarrow \sigma \quad (5.10)$$

oraz

$$\mathbf{I}_c^s \triangleright A \vdash M_2 : \sigma'. \quad (5.11)$$

M_1 jest w γ -nf i nie jest abstrakcją, a zatem z lematu 5.9: $\mathbf{act}(M_1) = \epsilon$. Wówczas z lematu 5.8: $\sigma' \rightarrow \sigma \in \mathbf{T}_0$, a zatem $\sigma' \in \mathbf{T}_0$. Możemy więc zastosować założenie indukcyjne do przesłanek (5.10) i (5.11),

otrzymując dla dowolnego $B \preceq A$: $\mathbf{ML}_c^s \triangleright B \vdash \mathbf{ml}(M_1) : \langle C_1, \sigma' \rightarrow \sigma \rangle$ oraz $\mathbf{ML}_c^s \triangleright B \vdash \mathbf{ml}(M_2) : \langle C_2, \sigma' \rangle$ i C_1, C_2 są tożsamościowe. Z tych dwóch asercji przez regułę (APP) systemu \mathbf{ML}_c^s otrzymujemy: $\mathbf{ML}_c^s \triangleright B \vdash \mathbf{ml}(M_1)\mathbf{ml}(M_2) : \langle C_1 \cup C_2 \cup \{\sigma' \rightarrow \sigma = \sigma' \rightarrow \sigma\}, \sigma \rangle$. ■

Pozostaje uogólnić powyższe twierdzenie na termy nie będące w γ -nf. W tym celu wprowadzamy pomocniczy lemat, wyrażający własność domkniętości systemu \mathbf{I}_c^s ze względu na γ -konwersję.

Lemat 5.11

- (i) Jeśli $\mathbf{I}_c^s \triangleright A \vdash (\lambda x(\lambda y M))N : \sigma$, to $\mathbf{I}_c^s \triangleright A \vdash \lambda y((\lambda x M)N) : \sigma$.
- (ii) Jeśli $M \rightarrow_\gamma N$ i $\mathbf{I}_c^s \triangleright A \vdash M : \sigma$, to $\mathbf{I}_c^s \triangleright A \vdash N : \sigma$.
- (iii) Jeśli $\mathbf{I}_c^s \triangleright A \vdash M : \sigma$, to $\mathbf{I}_c^s \triangleright A \vdash \gamma\text{-nf}(M) : \sigma$.

Dowód:

- (i) Asercję $\mathbf{I}_c^s \triangleright A \vdash (\lambda x(\lambda y M))N : \sigma$ otrzymano przy pomocy (APP) z przesłanki

$$(\forall i \in I) \mathbf{I}_c^s \triangleright A \vdash N : \tau_i \quad (5.12)$$

oraz z przesłanki $\mathbf{I}_c^s \triangleright A \vdash \lambda x(\lambda y M) : (\bigwedge_{i \in I} \tau_i) \rightarrow \sigma$. Tę ostatnią asercję otrzymano przez (ABS) z przesłanki $\mathbf{I}_c^s \triangleright A_x \cup \{x : (\bigwedge_{i \in I} \tau_i)\} \vdash \lambda y M : \sigma$. A zatem $\sigma = \sigma_1 \rightarrow \sigma_2$ i asercję tę otrzymano z przesłanki $\mathbf{I}_c^s \triangleright A_{x,y} \cup \{x : (\bigwedge_{i \in I} \tau_i)\} \cup \{y : \sigma_1\} \vdash M : \sigma_2$. Stąd przez (ABS) otrzymujemy, że

$$A_y \cup \{y : \sigma_1\} \vdash \lambda x M : (\bigwedge_{i \in I} \tau_i) \rightarrow \sigma_2. \quad (5.13)$$

Zauważmy, że w termie N nie może występować wolno zmienna y , na mocy założenia o rozłączności zbiorów zmiennych związanych i wolnych, jakie przyjęliśmy na początku tego podrozdziału. A zatem z (5.12) na mocy lematów 3.5 oraz 3.4 otrzymujemy, że $(\forall i \in I) \mathbf{I}_c^s \triangleright A_y \cup \{y : \sigma_1\} \vdash N : \tau_i$. To wraz z (5.13) przez (APP) daje, że $\mathbf{I}_c^s \triangleright A_y \cup \{y : \sigma_1\} \vdash (\lambda x M)N : \sigma_2$, stąd zaś z kolei przez (ABS) otrzymujemy, że $\mathbf{I}_c^s \triangleright A \vdash \lambda y((\lambda x M)N) : \sigma_1 \rightarrow \sigma_2 = \sigma$.

- (ii) Dowód indukcyjny ze względu na budowę termu M z wykorzystaniem części (i) niniejszego lematu.
- (iii) Dowód indukcyjny ze względu na liczbę redukcji z wykorzystaniem części (ii) niniejszego lematu. ■

Wniosek z powyższego lematu i twierdzenia 5.10 jest następujący:

Twierdzenie 5.12 *Jeśli $\mathbf{I}_c^s \triangleright A \vdash M : \sigma$, przy czym $\sigma \in \mathbf{T}_0$, to dla dowolnego $B \preceq A$: $\mathbf{ML}_c^s \triangleright B \vdash \mathbf{ml}(\gamma\text{-nf}(M)) : \langle C, \sigma \rangle$, gdzie C jest pewnym tożsamościowym zbiorem warunków.*

Dowód: Jeśli $\mathbf{I}_c^s \triangleright A \vdash M : \sigma$, to z lematu 5.11: $\mathbf{I}_c^s \triangleright A \vdash \gamma\text{-nf}(M) : \sigma$ i stosujemy twierdzenie 5.10. ■

Na zakończenie warto może nieformalnie wspomnieć, że warunki zawarte w sformułowaniu twierdzenia 5.10 nie są warunkami koniecznymi do tego, żeby term typowalny w systemie \mathbf{I}_c^s po konwersji funkcją \mathbf{ml} był typowalny w systemie \mathbf{ML}_c^s . Jeśli w wyprowadzaniu dla termu typu w \mathbf{I}_c^s nie korzystano z polimorfizmu, to niezależnie od tego, czy term ten jest w postaci $\gamma\text{-nf}$, czy też nie, będzie on typowalny również w \mathbf{ML}_c^s — w dodatku niezależnie od tego, czy najpierw przekonwertujemy go funkcją \mathbf{ml} , czy nie.

5.3 Wnioski

Wniosek z twierzeń 5.4 i 5.12 jest taki, że można mówić o pokrewieństwie systemów \mathbf{I}_c^s i \mathbf{ML}_c^s . Wprawdzie system \mathbf{I}_c^s jest silniejszy od systemu \mathbf{ML}_c^s ze względu na możliwości polimorficznej abstrakcji, to jednak — jak się okazuje — dla dużej klasy termów oba systemy działają tak samo: każdy term typowalny w \mathbf{ML}_c^s po eliminacji wyrażeń **let** jest typowalny w \mathbf{I}_c^s , każdy term mający w \mathbf{I}_c^s typ z \mathbf{T}_0 po zamianie aplikacji na wyrażenia **let** jest typowalny w \mathbf{ML}_c^s . Wprawdzie w tym drugim przypadku potrzebny jest jeszcze warunek, że term jest w postaci $\gamma\text{-nf}$, ale zauważmy, że sprowadzenie termu do postaci $\gamma\text{-nf}$ może być rozumiane — na równi z zastosowaniem funkcji \mathbf{ml} — jako jedna z faz zamiany aplikacji w tym termie na wyrażenia **let**.

Innym pożytecznym wnioskiem z tego rozdziału jest następujące twierdzenie dotyczące systemu \mathbf{ML}_c^s :

Twierdzenie 5.13 *Jeśli $\mathbf{ML}_c^s \triangleright A \vdash M : \langle C, \sigma \rangle$ i zbiór C jest tożsamościowy, to $\mathbf{unml}(M)$ jest silnie normalizowalny i $\mathbf{ML}_c^s \triangleright A \vdash \beta\text{-nf}(\mathbf{unml}(M)) : \langle C', \sigma \rangle$, dla pewnego tożsamościowego zbioru C' .*

Dowód: Jeśli $\mathbf{ML}_c^s \triangleright A \vdash M : \langle C, \sigma \rangle$ i zbiór C jest tożsamościowy, to z twierdzenia 5.3 zachodzi $\mathbf{I}_c^s \triangleright A' \vdash \mathbf{unml}(M) : \sigma$ dla pewnego A' takiego, że $A \preceq A'$. Wówczas, z twierdzenia 3.13, $\mathbf{unml}(M)$ jest silnie normalizowalny i z twierdzenia o zamkniętości systemu \mathbf{I}_c^s ze względu na β -redukcję otrzymujemy, że $\mathbf{I}_c^s \triangleright A' \vdash \beta\text{-nf}(\mathbf{unml}(M)) : \sigma$. Zauważmy, że ponieważ w $\beta\text{-nf}(\mathbf{unml}(M))$ nie ma β -redeksów, to $\gamma\text{-nf}(\beta\text{-nf}(\mathbf{unml}(M))) = \beta\text{-nf}(\mathbf{unml}(M))$ oraz $\mathbf{ml}(\gamma\text{-nf}(\mathbf{unml}(M))) = \gamma\text{-nf}(\mathbf{unml}(M))$. A zatem na mocy twierdzenia 5.12: $\mathbf{ML}_c^s \triangleright B \vdash \beta\text{-nf}(\mathbf{unml}(M)) : \langle C', \sigma \rangle$ i C' jest tożsamościowy — dla każdego B takiego, że $B \preceq A'$, a więc w szczególności dla

$B = A$. ■

Z powyższego twierdzenia wynika, że jeśli term M jest typowalny w systemie \mathbf{ML}_C^s (a więc ma typ z warunkami tożsamościowymi — patrz dowód twierdzenia 5.4), to term otrzymany przez jego uproszczenie będzie również typowalny w \mathbf{ML}_C^s w tym samym środowisku (zawierającym na przykład deklaracje pewnych funkcji pierwotnych).

Nasuwa się pytanie, jaki system typów z kwantyfikatorami opisywany w literaturze dałoby się rozszerzyć tak, aby był on bezwarunkowo równoważny systemowi \mathbf{I}_C^s . Odpowiedź, jakiej tu udzielimy, nie zostanie poparta dowodem. Wszystko jednak wskazuje na to, że systemem równoważnym systemowi \mathbf{I}_C^s byłby system powstały przez modyfikację systemu Λ_2^s — pewnego podzbioru rangi drugiej systemu F, opisywanego między innymi przez Jima w [6]. Należałoby dodać do typów tego systemu warunki oraz zmodyfikować reguły tak, by realizowały kumulację warunków — podobnie, jak to ma miejsce w systemie \mathbf{ML}_C^s . Niestety, po takich modyfikacjach otrzymalibyśmy system na tyle skomplikowany, że sens jego rozważania byłby co najmniej dyskusyjny.

Podsumowanie

Wprowadziliśmy trzy nowe systemy i omówiliśmy szczegółowo ich własności. Czy rzeczywiście są to systemy dla języków z podtypami? Regułę podtypowania zawiera wprawdzie tylko system \mathbf{I}_c , ale pokazaliśmy, że system \mathbf{I}_c^s jest mu równoważny, zaś system \mathbf{ML}_c^s jest wprawdzie nieco słabszy, to jednak różnica w sile \mathbf{I}_c^s i \mathbf{ML}_c^s wiąże się nie z podtypami, tylko z brakiem polimorficznej abstrakcji w systemie \mathbf{ML}_c^s . Po prostu, w systemach \mathbf{I}_c^s i \mathbf{ML}_c^s — jako sterowanych składnią — własność podtypu została wyrażona odpowiednio zdefiniowanymi regułami dla stałych i zmiennych.

Typowalność we wszystkich omawianych systemach sprowadza się do problemu istnienia rozwiązania pewnego układu równań i nierówności podtypowych, podobnie, jak to ma miejsce w przypadku innych systemów typów dla języków z podtypami (patrz na przykład [2]). Ze względu na brak miejsca nie pokusiliśmy się o oszacowanie rozmiarów i kosztów rozwiązywania wygenerowanych układów (w zależności od rozmiaru termu wejściowego). W literaturze (między innymi w [2]) oszacowano koszty typowalności dla kilku systemów typów będących rozszerzeniami ML na język z podtypami, jednak omawiane tam systemy nie są równoważne systemowi \mathbf{ML}_c^s . Nie są nam również znane żadne opracowania dotyczące kosztu typowalności w systemach intersekcyjnych dla języków z podtypami.

Bez wątplenia, najbardziej godny polecenia jest system \mathbf{I}_c . Po pierwsze, jest on najsilniejszy ze wszystkich omawianych w niniejszej pracy systemów (to znaczy, system \mathbf{I}_c^s jest mu równoważny, a \mathbf{ML}_c^s — słabszy) i bardzo silny na tle rozstrzygalnych systemów dla języków z podtypami omawianych w literaturze, ze względu na obecność polimorficznej abstrakcji. Po drugie, jego definicja jest bardzo przejrzysta; trochę czytelniejsza od definicji systemu \mathbf{I}_c^s oraz znacznie czytelniejsza od definicji systemu \mathbf{ML}_c^s , a także od definicji innych systemów dla języków z podtypami, omawianych w literaturze. Wynika to stąd, że zastąpienie typów intersekcyjnych typami z kwantyfikatorami przy jednoczesnym zachowaniu siły typowania wymusza użycie typów z warunkami (lub warunków w innej formie) w definicji systemu, co siłą rzeczy czyni ją mało przejrzystą.

Można podać argument przemawiający za systemem \mathbf{ML}_c^s taki, że w odróżnieniu od systemu \mathbf{I}_c posiada on własność typu głównego. Pytanie tylko, co z tego wynika. W przypadku tradycyjnego systemu typów ML

(i systemów mu pokrewnych) własność typu głównego pozwala w praktyce przede wszystkim na unaocznienie programiście tego, jaki typ ma napisana przez niego funkcja — poprzez podanie mu typu głównego tej funkcji, który zwykle ma czytelną i intuicyjną postać. Tymczasem w systemie \mathbf{ML}_c^s typy składają się między innymi z warunków, co czyni je całkowicie nieczytelnymi dla człowieka — chyba, że są to typy bardzo prostych wyrażeń, na przykład $\langle \{t_1 \leq_0 t_2\}, t_1 \rightarrow t_2 \rangle$ dla termu $\lambda x.x$. Pottier [7] próbuje upraszczać zbiory warunków, ale po pierwsze nie zawsze jest to możliwe, a po drugie nie bardzo wiadomo, w jakim sensie jego system mógłby mieć własność typu głównego.

Podsumowując — można odnieść wrażenie, że szczególnie dobrze i w naturalny sposób na języki z podtypami rozszerzają się systemy intersekcyjne. Gdyby jeszcze okazało się, że koszt typowości w tych systemach jest nie wyższego rzędu, niż koszt typowości w analizowanych w literaturze systemach będących rozszerzeniami systemów z typami zawierającymi kwantyfikatory, to mielibyśmy gotową odpowiedź na pytanie, jakie polimorficzne systemy stosować dla języków funkcyjnych z podtypami.

Literatura

- [1] Henk Barendregt. *The Lambda Calculus. Its Syntax and Semantics*, wydanie drugie, North-Holland, Amsterdam, 1984.
- [2] Marcin Benke. *Complexity of type reconstruction in programming languages with subtyping*. Rozprawa doktorska. Uniwersytet Warszawski, Wydział Matematyki, Informatyki i Mechaniki, Warszawa, 1997.
- [3] Felice Cardone i Mario Coppo. Two Extensions of Curry's Type Inference System. W: P. Odifreddi (ed.), *Logic and Computer Science*, strony 19-66, Academic Press, 1990.
- [4] Luis Damas i Robin Milner. Principal type schemes for functional programs. W: *Proceedings of the 9th ACM Symposium on Principles of Programming Languages*, strony 207-212, Albuquerque, Stany Zjednoczone, styczeń 1982.
- [5] Anthony J. Field, Peter G. Harrison. *Functional Programming*. Addison-Wesley Publishing Company, 1988.
- [6] Trevor Jim. Rank 2 type systems and recursive definitions. Technical Memorandum MIT/LCS/TM-531. Massachusetts Institute of Technology, Massachusetts, Stany Zjednoczone, sierpień 1995; korekta listopad 1995.
- [7] François Pottier. Simplifying subtyping constraints. To appear in the 1996 ACM SIGPLAN International Conference on Functional Programming. INRIA Rocquencourt, Le Chesnay Cedex, Francja, 1996.
- [8] G.Pottinger. A type assignment for the strongly normalizable λ -terms. W: J. P. Seldin i J. R. Hindley (ed.), *To H.B. Curry: Essays in Combinatory Logic, Lambda-Calculus and Formalism*, strony 561-577, Academic Press, Londyn, 1980.
- [9] Jerzy Tiurny. Subtype inequalities. W: *Proceedings of the Seventh Annual IEEE Symposium on Logic in Computer Science*, strony 308-315, IEEE Computer Society Press, Los Alamitos, Kalifornia, 1992.

- [10] Steffen van Bakel. *Intersection Type Disciplines in Lambda Calculus and Applicative Term Rewriting Systems*. Rozprawa doktorska. Matematisch Centrum, Amsterdam, luty 1993.