Testowanie własności drzew generowanych przez schematy rekurencyjne – podejście praktyczne

Dwa wykłady temu podaliśmy algorytm rozwiązujący następujący problem: czy drzewo generowane przez dany schemat rekurencyjny (lub λY-term) jest akceptowane przez dany automat na drzewach. Algorytm ten był n-krotnie wykładniczy. Co gorsza, jak udowodnimy wkrótce, problem ten jest n-EXPTIME-zupełny. Okazuje się jednak, że istnieją algorytmy, które w praktyce działają w rozsądnym czasie dla typowych wejść.

Głównym problemem poprzedniego algorytmu jest fakt, że zawsze oblicza on duże obiekty. Konkretnie, dla każdego podtermu M liczymy wartość $[\![M]\!]^v$. Jeśli M jest typu $\alpha{\to}\beta$, to $[\![M]\!]^v$ jest funkcją (monotoniczną) z $D[\alpha]$ w $D[\beta]$, reprezentowaną jako $|D[\alpha]|$ wartości (po jednej dla każdego elementu dziedziny); liczba $|D[\alpha]|$ jest bardzo duża, gdy α jest stosunkowo skomplikowanym typem. Zatem pierwsze wyzwanie, z którym musimy sobie poradzić, to bardziej zwięzłe reprezentowanie takich funkcji. Używa się do tego tzw. typów iloczynowych. Zanim przejdziemy do ich definicji, wprowadzimy jeszcze jeden model automatów na drzewach nieskończonych, wygodniejszy w tym przypadku.

Automaty z ko-trywialnym warunkiem akceptacji

Wcześniej rozważaliśmy automaty z trywialnym warunkiem akceptacji. Ich biegi na ścieżkach nieskończonych nie musiały spełniać żadnego dodatkowego warunku, poza zgodnością z funkcją przejścia. W przypadku automatu z ko-trywialnym warunkiem akceptacji będziemy wymagać, aby bieg „kończył się" na skończonym prefiksie drzewa. W tym celu wprowadzamy specjalny stan $\top$ oznaczający „akceptuję wszystko".

Formalnie, niedeterministyczny automat z ko-trywialnym warunkiem akceptacji to krotka $A = (\Sigma, Q, F, \delta)$, gdzie Σ to alfabet wejściowy (faktycznie automat będzie czytał drzewa nad alfabetem $\Sigma \cup \{\omega\}$), Q to zbiór stanów ($\top \notin Q$), $F \subseteq Q$ to zbiór stanów akceptujących (dozwolonych w korzeniu drzewa) oraz $\delta\colon Q \times (\Sigma \cup \{\omega\}) \to \bigcup_{r=0}^{\infty} \mathcal{P}((Q \cup \{\top\})^r)$ to funkcja przejścia spełniająca warunek $\delta(q,a) \in \mathcal{P}((Q \cup \{\top\})^{rank(a)})$. Zakładamy dodatkowo, że $\delta(q,\omega) = \emptyset$.

Biegiem automatu A na drzewie t nazwiemy funkcję ρ przypisującą wierzchołkom drzewa t stany z $Q \cup \{\top\}$ uwzględniającą w każdym wierzchołku funkcję przejścia: jeśli wierzchołkowi o etykiecie a przypisujemy stan $q \neq \top$, a jego dzieciom stany $q_1, \ldots, q_k$, to $(q_1, \ldots, q_k) \in \delta(q,a)$; jeśli zaś wierzchołkowi przypisujemy stan $\top$, to jego dzieciom także. Bieg jest akceptujący jeśli stan przypisany korzeniowi drzewa pochodzi ze zbioru F oraz tylko skończenie wielu wierzchołkom przypisaliśmy stan z Q (pozostałe mają stan $\top$). Drzewo jest akceptowane przez automat, jeśli istnieje bieg akceptujący.

Można zobaczyć (ćwiczenie), że język jest rozpoznawany przez automat z ko-trywialnym warunkiem akceptacji wtedy i tylko wtedy, gdy jego dopełnienie jest rozpoznawane przez automat z trywialnym warunkiem akceptacji.

Przykład. Ustalmy alfabet $\Sigma = \{a, b, c\}$, gdzie $rank(a) = 2$, $rank(b) = 1$, $rank(c) = 0$. Rozważmy język drzew, w których pewien wierzchołek i jego ojciec są etykietowani literą b (czyli dwie litery b jedna pod drugą). Język ten możemy rozpoznać za pomocą następującego automatu (oznaczmy go A_1) mającego dwa stany q_0 i q_1, gdzie stan q_0 jest akceptujący (może wystąpić w korzeniu). Funkcję przejścia definiujemy następująco:

$$\delta(q_0,a) = \delta(q_1,a) = \{(\top, q_0), (q_0, \top)\}\,, \qquad \delta(q_0,b) = \{(q_1)\}\,, \qquad \delta(q_1,b) = \{(\top)\}\,, \qquad \delta(q_0,c) = \delta(q_1,c) = \emptyset\,.$$

Typy iloczynowe

Tym razem skorzystamy z formalizmu schematów rekurencyjnych (choć typy iloczynowe można rozważać również w kontekście λY-termów). Ustalmy schemat rekurencyjny $\mathcal{G} = (\Sigma, \mathcal{N}, S, \mathcal{R})$. Ustalmy także automat $A = (\Sigma, Q, F, \delta)$ wczytujący drzewa nieskończone (z trywialnym lub ko-trywialnym warunkiem akceptacji).

Termy będziemy opisywać za pomocą zbioru typów iloczynowych. Typy iloczynowe uszczegóławiają typy proste. Definiujemy je przez indukcję:

$$\mathcal{T}^o = Q\,, \qquad\qquad \mathcal{T}^{\alpha\to\beta} = \mathcal{P}(\mathcal{T}^\alpha) \times \mathcal{T}^\beta\,.$$

Formalnie typ iloczynowy uszczegóławiający $\alpha{\to}\beta$ jest parą $(T,\tau) \in \mathcal{P}(\mathcal{T}^\alpha) \times \mathcal{T}^\beta$, lecz w praktyce zapisujemy go jako $(\bigwedge T){\to}\tau$. Gdy $T = \{\tau_1, \ldots, \tau_k\}$, piszemy także $(\tau_1 \wedge \cdots \wedge \tau_k){\to}\tau$ lub $(\bigwedge_{i=1}^k \tau_i){\to}\tau$. Gdy $k = 0$ piszemy $\top{\to}\tau$ zamiast $(\bigwedge \emptyset){\to}\tau$.

Typ $q \in \mathcal{T}^o$ opisuje term generujący drzewo, które jest akceptowane ze stanu q. Typ $(\tau_1 \wedge \cdots \wedge \tau_k){\to}\tau$ opisuje funkcję, która otrzymawszy argument mający wszystkie typy $\tau_1, \ldots, \tau_k$ zwraca wartość mającą typ τ. Przykładowo $(p \wedge q){\to}q$ opisuje term (mający typ prosty $o{\to}o$), który otrzymawszy drzewo akceptowane z obu stanów p i q tworzy drzewo akceptowane ze stanu q. W szczególności ten sam term może mieć wiele typów iloczynowych (na przykład term typu o generujący drzewo akceptowane zarówno ze stanu p jak i ze stanu q ma zarówno typ p jak i typ q).

Ponieważ termy mają zmienne wolne, musimy otypowywać je w zadanym otoczeniu przypisującym typy zmiennym wolnym. *Otoczeniem* nazwiemy zbiór Γ zawierający pary postaci $x : \tau$, gdzie $\tau \in \mathcal{T}^\alpha$ gdy typ prosty zmiennej x to α. W otoczeniu może być wiele par zawierających tę samą zmienną. *Osądem* nazwiemy wyrażenie postaci $\Gamma \vdash M : \tau$, gdzie Γ jest otoczeniem, M jest termem oraz τ jest typem iloczynowym uszczegóławiającym typ prosty termu M. Pisząc konkretne osądy często piszemy otoczenie Γ jako ciąg par, a nie jako zbiór, na przykład $x : \tau_1, y : \tau_2 \vdash M : \tau$ zamiast $\{x : \tau_1, y : \tau_2\} \vdash M : \tau$.

Osądy wyprowadzamy korzystając z następujących reguł wyprowadzania.

$$\frac{}{\Gamma \cup \{x : \tau\} \vdash x : \tau} \qquad \frac{\Gamma \vdash M : (\bigwedge_{i \in I} \tau_i) \to \tau \qquad \Gamma \vdash N : \tau_i \text{ dla każdego } i \in I}{\Gamma \vdash M\,N : \tau}$$

$$\frac{(q_1, \ldots, q_r) \in \delta(q, a) \qquad \Gamma \vdash N_i : q_i \text{ dla każdego } i \in \{1, \ldots, r\} \text{ takiego, że } q_i \neq \top}{\Gamma \vdash a\,N_1\,\ldots\,N_r : q}$$

$$\frac{(A\,x_1\,\ldots\,x_k \to M) \in \mathcal{R} \qquad \{x_i : \tau_{ij} \mid 1 \leq i \leq k, j \in I_j\} \vdash M : q}{\Gamma \vdash A : (\bigwedge_{j \in I_1} \tau_{1j}) \to \ldots \to (\bigwedge_{j \in I_k} \tau_{kj}) \to q}$$

Reguły powyższe rozumiemy w ten sposób, że aby wyprowadzić osąd znajdujący się pod kreską należy wyprowadzić najpierw osądy znajdujące się nad kreską. Warunki $(q_1, \ldots, q_r) \in \delta(q, a)$ oraz $(A\,x_1\,\ldots\,x_k \to M) \in \mathcal{R}$ (nie będące osądami) to dodatkowe warunki, które muszą być spełnione.

Istnieje spore podobieństwo między wprowadzonym tutaj systemem typów iloczynowych, a wartościami w modelu zdefiniowanym wcześniej. Wcześniej termowi przypisywaliśmy funkcję monotoniczną, a teraz przypisujemy zbiór typów iloczynowych, które opisują taką funkcję.

Przykład. Rozważmy schemat rekurencyjny $\mathcal{G}_1$ o regułach:

$$S \to F\,c \qquad\qquad F\,x \to a\,x\,(F\,(b\,x))\,.$$

Generuje on drzewo mające a na skrajnie prawej ścieżce; i-te a na tej ścieżce ma w lewym poddrzewie ścieżkę etykietowaną słowem $b^{i-1}c$. Widzimy, że drzewo to jest akceptowane (ze stanu q_0) przez automat A_1 z poprzedniego przykładu. Zobaczmy, jak wygląda wyprowadzenie osądu $\vdash S : q_0$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x : q_0 \vdash x : q_0}}{x : q_0 \vdash a\,x : q_0}}{x : q_0 \vdash a\,x\,(F\,(b\,x)) : q_0}}{x : q_1 \vdash F : q_0 \to q_0} \qquad \dfrac{\dfrac{}{x : q_1 \vdash x : q_1}}{x : q_1 \vdash b\,x : q_0}}{x : q_1 \vdash F\,(b\,x) : q_0}}{\dfrac{\dfrac{\dfrac{x : q_1 \vdash a\,x\,(F\,(b\,x)) : q_0}{\vdash F : q_1 \to q_0} \qquad \dfrac{}{\vdash b\,x : q_1}}{\vdash F\,(b\,x) : q_0}}{\vdash a\,x\,(F\,(b\,x)) : q_0}}{\dfrac{\dfrac{\vdash F : \top \to q_0}{\vdash F\,c : q_0}}{\vdash S : q_0}}$$

Zgodność naszego systemu typów z intuicją jest opisana przez poniższe dwa lematy. Występujący w nich term M to term zbudowany z użyciem aplikacji, konstruktorów drzewa i nieterminali schematu $\mathcal{G}$.

Lemat. *Jeśli rozważamy automat z ko-trywialnym warunkiem akceptacji, to drzewo generowane przez term zamknięty M typu o jest akceptowane ze stanu q wtedy i tylko wtedy, gdy istnieje skończone wyprowadzenie dla osądu $\vdash M : q$.*

Lemat. *Jeśli rozważamy automat z trywialnym warunkiem akceptacji, to drzewo generowane przez term zamknięty M typu o jest akceptowane ze stanu q wtedy i tylko wtedy, gdy istnieje (potencjalnie) nieskończone wyprowadzenie dla osądu $\vdash M : q$.*

Przedstawmy pokrótce ideę dowodu, bez wchodzenia w szczegóły. Po pierwsze, dowodzimy, że typy nie zmieniają się, gdy korzystamy z reguł wyprowadzenia naszego schematu.

Lemat. *Załóżmy, że $M \to_{\mathcal{G}} N$. Wówczas można wyprowadzić osąd $\vdash M : q$ wtedy i tylko wtedy, gdy można wyprowadzić osąd $\vdash N : q$.*

Dowód jest nietrudny: korzystając z drzewa wyprowadzenia dla M konstruujemy drzewo wyprowadzenia dla N i odwrotnie.

W przypadku automatu trywialnego powyższy lemat pozwala nam dzięki wyprowadzeniu dla $\vdash M : q$ znajdować bieg dla dowolnie długich prefiksów drzewa generowanego przez M. W granicy daje to bieg na całym drzewie, mamy więc implikację z lewej w prawą. Przy dowodzie w lewą stronę mamy podobny problem jak w konstrukcji z modelem: nie ma od czego zacząć. Rozważamy więc „skończony wariant" naszego schematu, gdzie po n użyciach danego nieterminala zmienia się on w symbol ω. Taki „schemat" generuje skończone drzewo, na którym jest bieg automatu (bo to prefiks nieskończonego drzewa, na którym jest bieg automatu). Daje to nam wyprowadzenie dla $\vdash M : q$, w którym jednak korzystamy z (niewyprowadzalnych) założeń postaci $\Gamma \vdash A : \top \to \ldots \to \top \to p$ (w miejscach, gdzie nieterminal A rozwijał się jako ω). Założenia te występują jedynie na głębokości co najmniej n (po drodze n razy rozwinęliśmy A normalnie, zanim zamieniliśmy go na ω). Jeśli jednak wzięliśmy n wystarczająco duże, to po drodze od takiego założenia do korzenia pewien osąd powtarza się. Ucinamy więc nasze wyprowadzenie w takim miejscu, zapętlając je – dostajemy nieskończone wyprowadzenie, nie korzystające z niewyprowadzalnych założeń. Szczegóły pomijamy.

Przejdźmy teraz do przypadku automatu ko-trywialnego. Implikacja z prawej w lewą jest w tym przypadku prostsza: bieg automatu na generowanym drzewie jest tak naprawdę biegiem na pewnym skończonym prefiksie tego drzewa. Wystarczy więc rozwinąć term M skończoną liczbę razy, aby wygenerować ten prefiks i uzyskać typowanie. Jednym ze sposobów, aby dowieść implikację w prawą stronę, jest zdefiniowanie odpowiedniego dobrze ufundowanego porządku na drzewach wyprowadzeń. Będzie to taki porządek, że jeśli w wyprowadzeniu osądu $\vdash M : q$ występuje reguła dla nieterminala, to możemy wykonać redukcję $M \to_{\mathcal{G}} N$ (eliminującą ten nieterminal), a wyprowadzenie osądu $\vdash N : q$ będzie mniejsze w naszym porządku. W efekcie po skończenie wielu krokach dostajemy wyprowadzenie nie używające reguły dla nieterminala; takie wyprowadzenie to po prostu bieg automatu na pewnym prefiksie generowanego drzewa. Szczegóły pomijamy. Zwróćmy jednak uwagę, że zdefiniowanie wspomnianego porządku nie jest całkiem proste. Nie możemy porządkować wyprowadzeń po naiwnie rozumianym rozmiarze, gdyż w wyniku rozwinięcia nieterminala rozmiar wyprowadzenia może wzrosnąć (z powodu wielokrotnego skopiowania wyprowadzeń dla argumentów).

Wyprowadzanie typów

Mamy już zdefiniowany system typów. Pozostaje pytanie: jak sprawdzić, czy można w nim wyprowadzić osąd $\vdash S : q$ (gdzie S jest nieterminalem startowym, a q jednym ze stanów akceptujących). Napotykamy przy tym następujące trudności.

1. W drzewach wyprowadzenia często będzie tak, że takie samo poddrzewo występuje w wielu kopiach. Aby tego uniknąć, przydałaby się jakaś bardziej „grafowa" reprezentacja drzew wyprowadzeń. W tym celu potniemy wyprowadzenia w miejscach użycia reguły dla nieterminala.

2. Widzimy, że jeśli można wyprowadzić $\Gamma \vdash M : \tau$, to można też wyprowadzić $\Gamma' \vdash M : \tau$ dla każdego $\Gamma' \supseteq \Gamma$. W związku z tym, jeśli dla nieterminala A można wyprowadzić typ $(\bigwedge T) \to \tau$, to można też wyprowadzić $(\bigwedge T') \to \tau$ dla każdego $T' \supseteq T$. Powoduje to, że zawsze można wyprowadzić całe mnóstwo typów (do zbioru T możemy dorzucić dowolne typy z odpowiedniego $\mathcal{T}^\alpha$, a jest ich bardzo dużo). Takie dodatkowe typy są jednak mało użyteczne: jeśli do A podaję argument mający wszystkie typy z T', to argument ten ma tym bardziej wszystkie typy z T, mogę więc równie dobrze zamiast typu $(\bigwedge T') \to \tau$ użyć typu $(\bigwedge T) \to \tau$. Mając to na uwadze, chcielibyśmy wyprowadzić jedynie typ $(\bigwedge T) \to \tau$ z minimalnym T. Aby to uzyskać, w otoczeniu Γ będziemy gromadzić jedynie te pary $x : \sigma$, które są gdziekolwiek użyte w wyprowadzeniu.

3. Rozważmy nieterminal A, dla którego mamy regułę $A\, x^{\alpha \to o}\, y^\alpha \to x\, y$. W tym przypadku dla dowolnego zbioru $T \subseteq \mathcal{T}^\alpha$ możemy dla A wyprowadzić typ $((\bigwedge T) \to q) \to (\bigwedge T) \to q$; innymi słowy, o y możemy założyć cokolwiek, jeśli tylko o x założymy, że potrzebuje takiego właśnie argumentu. Daje to bardzo dużo możliwych typów dla powyższego nieterminala A. Tymczasem gdy już będziemy korzystać z A, to za x i y podstawimy jakieś konkretne rzeczy, mające konkretne typy. Aby temu zaradzić, będziemy pozwalali na przypisanie zmiennej jakiegoś zbioru typów tylko wtedy, gdy można skonstruować term mający takie typy.

Przejdźmy teraz do konkretów. Niech $\mathcal{X}$ będzie zbiorem par postaci $A : \tau$, gdzie A jest nieterminalem, a τ jest typem uszczegóławiającym typ prosty nieterminala A (czyli jak otoczenie, ale dla nieterminali zamiast dla zmiennych). Będziemy rozważać osądy postaci $\Gamma \vdash_{\mathcal{X}} M : \tau$ (czyli jak poprzednio, ale z $\mathcal{X}$ w dolnym indeksie). Do wyprowadzania

tych osądów używamy następującego systemu typów (znaczenie napisu $\mathrm{Inh}(\mathcal{X})$ zostanie wyjaśnione później; początkowo można założyć, że zbiór ten zawiera wszystkie możliwe otoczenia):

$$\frac{\{x:\tau\}\in \mathrm{Inh}(\mathcal{X})}{x:\tau \vdash_{\mathcal{X}} x:\tau} \qquad\qquad \frac{}{\vdash_{\mathcal{X}\cup\{A:\tau\}} A:\tau}$$

$$\frac{\Gamma \vdash_{\mathcal{X}} M:(\bigwedge_{i\in I}\tau_i)\to\tau \qquad \Gamma_i \vdash_{\mathcal{X}} N:\tau_i \text{ dla każdego } i\in I \qquad \Gamma\cup\bigcup_{i\in I}\Gamma_i\in \mathrm{Inh}(\mathcal{X})}{\Gamma\cup\bigcup_{i\in I}\Gamma_i \vdash_{\mathcal{X}} M\,N:\tau}$$

$$\frac{(q_1,\ldots,q_r)\in\delta(q,a) \qquad I=\{i\in\{1,\ldots,r\}\mid q_i\neq\top\} \qquad \Gamma_i\vdash_{\mathcal{X}} N_i:q_i \text{ dla każdego } i\in I \qquad \bigcup_{i\in I}\Gamma_i\in\mathrm{Inh}(\mathcal{X})}{\bigcup_{i\in I}\Gamma_i\vdash_{\mathcal{X}} a\,N_1\,\ldots\,N_r:q}$$

Dla $\tau=(\bigwedge_{j\in I_1}\tau_{1j})\to\ldots\to(\bigwedge_{j\in I_k}\tau_{kj})\to q$ oraz $(A\,x_1\,\ldots\,x_k\to M)\in\mathcal{R}$ napiszemy $\mathcal{X}\Rightarrow(A:\tau)$ gdy można wyprowadzić osąd $\{x_i:\tau_{ij}\mid 1\le i\le k, j\in I_j\}\vdash_{\mathcal{X}} M:q$. Innymi słowy, $\mathcal{X}\Rightarrow(A:\tau)$ zachodzi, gdy dla A można wyprowadzić typ τ korzystając z założeń z $\mathcal{X}$ i używając reguły dla nieterminala jedynie w korzeniu. Zdefiniujmy

$$\mathcal{X}_0=\emptyset\,, \qquad\qquad \mathcal{X}_{i+1}=\mathcal{X}_i\cup\{(A:\tau)\mid \mathcal{X}_i\Rightarrow(A:\tau)\}\,.$$

Twierdzimy, że można wyprowadzić $\vdash S:q$ wtedy i tylko wtedy, gdy $(S:q)\in\mathcal{X}_n$ dla n takiego, że $\mathcal{X}_n=\mathcal{X}_{n+1}$ (jednocześnie liczymy na to, że dla nieterminali typów innych niż o będzie teraz można wyprowadzić znacznie mniej typów iloczynowych niż w oryginalnym systemie typów).

Powyższego faktu nie będziemy formalnie udowadniać; zamiast tego zobaczmy jak ma się powyższy system typów do poprzedniego. Po pierwsze zamiast oryginalnej reguły dla nieterminala możemy teraz tylko wyciągnąć typ nieterminala ze zbioru $\mathcal{X}$. Konkretniej przy wyprowadzaniu typów ze zbioru $\mathcal{X}_i$ możemy korzystać z typów nieterminali wyprowadzonych wcześniej w zbiorze $\mathcal{X}_{i-1}$. Zatem, intuicyjnie, zbiór $\mathcal{X}_i$ zawiera takie pary $A:\tau$, że istnieje wyprowadzenie dla $\vdash A:\tau$ używające regułę dla nieterminala co najwyżej i-krotnie na każdej ścieżce korzeń-liść. W ten sposób realizujemy postulat 1. z powyższej listy, czyli pocięcie wielkiego wyprowadzenia na części.

Postulat 2. zrealizowaliśmy w ten sposób, że w otoczeniu znajdują się tylko te pary $x:\sigma$, które są faktycznie użyte w wyprowadzeniu.

Postulat 3. realizujemy za pomocą zbiorów $\mathrm{Inh}(\mathcal{X})$. W pierwszym przybliżeniu powiedzmy, że otoczenie Γ należy do $\mathrm{Inh}(\mathcal{X})$ wtedy, gdy dla każdej zmiennej x istnieje term mający naraz wszystkie typy σ dla których $(x:\sigma)\in\Gamma$. Mówiąc o istnieniu termu, mamy na myśli term M zbudowany za pomocą aplikacji i konstruktorów drzewa z nieterminali, taki, że $\vdash_{\mathcal{X}} M:\sigma$ (czyli o nieterminalach zakładamy jedynie to, co mówi zbiór $\mathcal{X}$). Chodzi o to, że nie ma sensu wyprowadzanie jakiegokolwiek typu korzystając z założenia, że x-owi możemy przypisać jakiś zbiór typów, w sytuacji, gdy nie umiemy skonstruować żadnego termu mającego taki zbiór typów (bo ostatecznie i tak jedyny sposób na skorzystanie z naszego wyprowadzenia to podstawienie właśnie takiego termu za parametr x).

Mając tak zdefiniowany system typów, po prostu bierzemy zbiór $\mathcal{X}_i$, na wszystkie możliwe sposoby próbujemy wyprowadzać jakieś typy dla prawych stron reguł i dodajemy wyprowadzone typy do zbioru $\mathcal{X}_{i+1}$. Wykonując powyższe liczymy na to, że procedura ta zakończy się stosunkowo szybko (skoro rozważany problem jest n-EXPTIME-zupełny, to nie zawsze tak będzie, ale eksperymenty pokazują, że rzeczywiście przedstawiony tu algorytm w wielu przypadkach działa szybko).

Powiedzmy jeszcze jak sprawdzić czy $\Gamma\in\mathrm{Inh}(\mathcal{X})$. W tym celu postępujemy następująco. Dla każdego typu prostego α będziemy mieli zbiór Z_α zawierający pewne podzbiory $\mathcal{T}^\alpha$. Docelowo elementy Z_α to wszystkie maksymalne zbiory $T\subseteq\mathcal{T}^\alpha$, dla których można skonstruować term mający naraz wszystkie typy z α. Zaczynamy od $Z_\alpha=\{\emptyset\}$. Na początek dla każdego nieterminala A mającego typ prosty α dodajemy do Z_α zbiór $\{\tau\mid(A:\tau)\in\mathcal{X}\}$. Następnie bierzemy jakiś typ prosty $\alpha\to\beta$ oraz jakieś zbiory $T\in Z_{\alpha\to\beta}$ i $U\in Z_\alpha$ i do Z_β dodajemy ich złożenie $\{\tau:((\bigwedge V)\to\tau)\in T\wedge V\subseteq U\}$ (czyli zbiór typów, które możemy przypisać aplikacji termu mającego typy z T do argumentu mającego typy z U). Podobnie postępujemy dla konstruktorów drzewa: wybieramy literę a oraz zbiory $T_1,\ldots,T_r\in Z_o$, gdzie $r=\mathit{rank}(a)$, i do Z_o dodajemy zbiór $\{q\mid\delta(q,a)\cap(T_1\cup\{\top\})\times\cdots\times(T_r\cup\{\top\})\neq\emptyset\}$. To powtarzamy tak długo jak się da, ale jednocześnie za każdym razem w każdym Z_α zostawiamy tylko zbiory maksymalne, czyli jeśli $T\subset U$ dla $T,U\in Z_\alpha$, to usuwamy T z Z_α. Gdy zbiory Z_α się ustabilizują, dla każdej zmiennej x^α sprawdzamy czy w Z_α jest zbiór T taki, że $\{\tau\mid(x:\tau)\in\Gamma\}\subseteq T$.

Aby uzyskać jeszcze lepsze efekty, zastępuje się powyższą definicję zbioru $\mathrm{Inh}(\mathcal{X})$ inną, jeszcze bardziej restrykcyjną. W podanej wcześniej definicji patrzyliśmy tylko, czy term o jakimś zbiorze typów da się w ogóle skonstruować. Nie uwzględnialiśmy natomiast tego, jakie termy są konstruowane w naszym schemacie rekurencyjnym. Tymczasem chociaż potencjalnie jakiś term można by skonstruować, to być może w naszym schemacie nic takiego się nie dzieje. Bez wnikania w szczegóły powiedzmy, że istnieją sposoby przybliżania (z góry) zbioru termów (czy tak naprawdę zbiorów

typów przypisanych tym termom), które mogą być podstawione za zmienną x w danym schemacie rekurencyjnym. Stosuje się do tego tzw. algorytm 0CFA (gdzie CFA = constrol flow analysis).

Przykład. Rozważmy automat A_1 oraz schemat rekurencyjny $\mathcal{G}_1$ z poprzednich przykładów. Widzimy, że da się wyprowadzić $x : q_0 \vdash_\emptyset a\,x\,(F\,(b\,x)) : q_0$ oraz $x : q_0 \vdash_\emptyset a\,x\,(F\,(b\,x)) : q_1$ (zwróćmy uwagę, że $\{x : q_0\} \in \mathrm{Inh}(\emptyset)$, gdyż istnieje drzewo akceptowane ze stanu q_0). Oznacza to, że $\emptyset \Rightarrow F : q_0{\to}q_0$ oraz $\emptyset \Rightarrow F : q_0{\to}q_1$, w związku z czym $\mathcal{X}_1 = \{F : q_0{\to}q_0, F : q_0{\to}q_1\}$. Dalej mamy:

$$\mathcal{X}_2 = \mathcal{X}_1 \cup \{F : q_1{\to}q_0, F : q_1{\to}q_1\}\,, \qquad \mathcal{X}_3 = \mathcal{X}_2 \cup \{F : \top{\to}q_0, F : \top{\to}q_1\}\,, \qquad \mathcal{X}_4 = \mathcal{X}_3 \cup \{S : q_0, S : q_1\} = \mathcal{X}_5\,.$$

Mamy tutaj $(S : q_0) \in \mathcal{X}_4$, co oznacza, że drzewo generowane przez $\mathcal{G}_1$ jest akceptowane przez A_1.

Automaty z trywialnym warunkiem akceptacji

Powyżej przestawiliśmy algorytm dla automatów z ko-trywialnym warunkiem akceptacji. Zobaczmy teraz jak go zmodyfikować w sytuacji, gdy mamy trywialny warunek akceptacji.

W tym przypadku dozwalamy na nieskończone drzewa wyprowadzenia. Wszystkie typy przypisane nieterminalom w takim wyprowadzeniu możemy zgromadzić w jednym zbiorze $\mathcal{X}$. Pytamy więc, czy istnieje zbiór $\mathcal{X}$ taki, że dla każdej pary $(A : \tau) \in \mathcal{X}$ zachodzi $\mathcal{X} \Rightarrow (A : \tau)$ (czyli da się ją wyprowadzić korzystając z założeń w $\mathcal{X}$), a jednocześnie $(S : q) \in \mathcal{X}$, gdzie S jest nieterminalem startowym, a q stanem akceptującym.

Powstaje pytanie jak znaleźć taki zbiór $\mathcal{X}$. Okazuje się, że dobrze jest zacząć od zbioru $\mathcal{X}_0$ zawierającego dla każdego nieterminala A i stanu q parę $A : \top{\to}\ldots{\to}\top{\to}q$; przypomnijmy, że $\top$ oznacza po prostu $\bigwedge\emptyset$ (liczbę znaczków $\top$ dobieramy oczywiście zgodnie z liczbą argumentów nieterminala A). Następnie, tak jak poprzednio obliczamy

$$\mathcal{X}_{i+1} = \mathcal{X}_i \cup \{(A : \tau) \mid \mathcal{X} \Rightarrow (A : \tau)\}\,.$$

Robimy to, aż uzyskamy $\mathcal{X}_{n+1} = \mathcal{X}_n$.

W ten sposób uzyskujemy na pewno za dużo par w $\mathcal{X}_n$; na przykład już $\mathcal{X}_0$ zawiera pary $S : q$ dla każdego stanu q. Na koniec uzyskany zbiór musimy oczyścić. Powtarzamy w tym celu następującą operację:

$$\mathit{Czysc}(\mathcal{X}) = \{(A : \tau) \in \mathcal{X} \mid \mathcal{X} \Rightarrow (A : \tau)\}\,.$$

Niech $\mathcal{Y}$ będzie zbiorem uzyskanym przez wielokrotne powtarzanie operacji *Czysc* na zbiorze $\mathcal{X}_n$, tak długo, jak długo kolejne wywołanie *Czysc* usuwa coś z aktualnego zbioru. W efekcie mamy $\mathit{Czysc}(\mathcal{Y}) = \mathcal{Y}$, czyli faktycznie dla każdej pary $(A : \tau) \in \mathcal{Y}$ zachodzi $\mathcal{Y} \Rightarrow (A : \tau)$. Jeśli więc $(S : q) \in \mathcal{Y}$ dla pewnego stanu akceptującego q, to osąd $\vdash S : q$ można uzyskać pewnym, być może nieskończonym, wyprowadzeniem.

Implikacja w przeciwnym kierunku nie jest całkiem oczywista. Przedstawimy tu tylko szkic jej dowodu, pomijając szczegóły. Postępujemy w następujący sposób. Bierzemy wyprowadzenie dla $\vdash S : q$, które może być nieskończone. Dla odpowiednio dużego n znajdujemy wszystkie wystąpienia reguły dla nieterminala znajdujące się na głębokości co najmniej n w tym wyprowadzeniu i umieszczamy tam po prostu typy postaci $\top{\to}\ldots{\to}\top{\to}p$. Następnie poprawiamy lekko resztę wyprowadzenia tak, aby pozostało ono poprawne. Dostajemy w ten sposób skończone wyprowadzenie, w którym korzystamy bez uzasadnienia z założeń mówiących, że pewne nieterminale mają typy postaci $\top{\to}\ldots{\to}\top{\to}p$. Te założenia są w $\mathcal{X}_0$, w związku z czym jeśli gdziekolwiek w tym wyprowadzeniu uzyskujemy jakiś typ τ dla jakiegoś nieterminala A, to para $A : \tau$ jest w zbiorze $\mathcal{X}_n$. W szczególności jest tam para $S : q$. Trzeba jeszcze wiedzieć, że ta para nie zostanie usunięta z $\mathcal{X}_n$ przez operację *Czysc*. W tym celu musimy nasze wyprowadzenie ponownie zapętlić, usuwając nieuzasadnione założenia. Otóż, rozważmy jakiś węzeł, w którym sztucznie umieściliśmy typ postaci $\top{\to}\ldots{\to}\top{\to}p$. Ponieważ węzeł ten znajduje się bardzo głęboko (na głębokości co najmniej n), to na ścieżce od tego węzła do korzenia w pewnych dwóch węzłach wyprowadzamy tę samą parę $A : \tau$. W tym miejscu możemy nasze wyprowadzenie uciąć i zapętlić. W efekcie dostajemy wyprowadzenie, w którym każda para $A : \tau$ jest uzasadniona, a jednocześnie jest w zbiorze $\mathcal{X}_n$. Zatem pary te (w szczególności para $S : q$) są w zbiorze $\mathcal{Y}$.

Przykład. Zdefiniujmy automat A_2, z trywialnym warunkiem akceptacji. Ma on dwa stany: q_0 i q_1, gdzie q_0 jest akceptujący. Funkcja przejścia jest następująca:

$$\delta(q_0, a) = \{(q_0, q_0)\}\,, \qquad \delta(q_1, a) = \emptyset\,, \qquad \delta(q_0, b) = \delta(q_1, b) = \{(q_1)\}\,, \qquad \delta(q_0, c) = \delta(q_1, c) = \{()\}\,.$$

Automat ten akceptuje drzewa, w których poniżej liter b nie występują już litery a.

Nadal odwołujemy się do gramatyki $\mathcal{G}_1$ rozważanej wcześniej. Mamy:

$$\mathcal{X}_0 = \{S : q_0, S : q_1, F : \top{\to}q_0, F : \top{\to}q_1\}\,, \quad \mathcal{X}_1 = \mathcal{X}_0 \cup \{F : q_0{\to}q_0\}\,, \quad \mathcal{X}_2 = \mathcal{X}_1 \cup \{F : (q_0 \wedge q_1){\to}q_0\} = \mathcal{X}_3\,.$$

Następnie zobaczmy, co dzieje się w wyniku czyszczenia:

$$Czysc(\mathcal{X}_2) = \{S : q_0, S : q_1, F : q_0 \to q_0, F : (q_0 \wedge q_1) \to q_0\},$$
$$Czysc^2(\mathcal{X}_2) = \{S : q_0, F : (q_0 \wedge q_1) \to q_0\} = Czysc^3(\mathcal{X}_2) = \mathcal{Y}.$$

Dostajemy $(S : q_0) \in \mathcal{Y}$, co oznacza, że drzewo generowane przez $\mathcal{G}_1$ jest akceptowane przez A_2.

Uwaga. Całe powyższe rozumowanie można uogólnić z automatów niedeterministycznych do automatów alternujących. W tym celu wystarczy odpowiednio poprawić regułę dla konstruktora drzewa tak, aby była ona zgodna z postacią funkcji przejścia automatu.