Testowanie własności drzew generowanych przez termy rachunku λY

Celem niniejszego wykładu jest przedstawienie algorytmu rozstrzygającego następujący problem: dany jest zamknięty λY-term K typu o oraz jakaś własność drzew φ; stwierdzić, czy drzewo generowane przez K spełnia własność φ. Własność φ jest zwykle zadana za pomocą formuły odpowiedniej logiki, bądź za pomocą automatu wczytującego drzewa nieskończone; my skoncentrujemy się na tej drugiej opcji.

Automaty z trywialnym warunkiem akceptacji

Powyższy problem jest rozstrzygalny, gdy własność φ opisuje należenie do dowolnego języka regularnego. My jednak skupimy się na nieco bardziej ograniczonej klasie własności, opisywanej przez tzw. automaty z trywialnym warunkiem akceptacji, które dodatkowo ignorują występowanie symbolu ω.

Niedeterministycznym automatem z trywialnym warunkiem akceptacji nazwiemy krotkę $A = (\Sigma, Q, F, \delta)$, gdzie Σ to alfabet wejściowy (faktycznie automat będzie czytał drzewa nad alfabetem $\Sigma \cup \{\omega\}$), Q to zbiór stanów, $F \subseteq Q$ to zbiór stanów akceptujących (dozwolonych w korzeniu drzewa) oraz $\delta\colon Q \times (\Sigma \cup \{\omega\}) \to \bigcup_{r=0}^{\infty} \mathcal{P}(Q^r)$ to funkcja przejścia spełniająca warunek $\delta(q,a) \in \mathcal{P}(Q^{rank(a)})$. Dla etykiety a o randzie k wartość funkcji przejścia $\delta(q,a)$ to zbiór k-krotek stanów – są to stany, które mogą znaleźć się w dzieciach rozważanego wierzchołka.

Biegiem automatu A na drzewie t nazwiemy funkcję przypisującą wierzchołkom drzewa t stany automatu A uwzględniającą w każdym wierzchołku funkcję przejścia: jeśli wierzchołkowi o etykiecie a przypisujemy stan q, a jego dzieciom stany $q_1, \ldots, q_k$, to $(q_1, \ldots, q_k) \in \delta(q,a)$. Bieg jest akceptujący jeśli stan przypisany korzeniowi drzewa pochodzi ze zbioru F. Drzewo jest akceptowane przez automat, jeśli istnieje bieg akceptujący.

W szczególnym przypadku etykiety a o randze 0 (umieszczanej w liściach) mamy dwie możliwości: $\delta(q,a) = \{()\}$ (zbiór zawierający 0-krotkę) lub $\delta(q,a) = \emptyset$ (zbiór pusty). Pierwsza z nich oznacza, że stan q może być umieszczony w liściu o etykiecie a, druga – że nie może. Dla zwiększenia czytelności będziemy w tym kontekście oznaczać $\{()\}$ przez true, a $\emptyset$ przez false. W ten sposób określamy jakie stany mogą znaleźć się w liściach. Jednocześnie nie mamy żadnego wymagania odnośnie układu stanów na ścieżkach nieskończonych – to stanowi słabość naszych automatów.

Przypomnijmy, że jeśli λY-term jest rozbieżny (nie redukuje się do termu postaci $a\,N_1\,\ldots\,N_m$), to w drzewie generowanym przez ten term umieszczamy symbol ω. Chcemy się ograniczyć do automatów nie będących w stanie wykryć czy drzewo jest zakończone symbolem ω, czy ciągnie się dalej. Zakładamy więc, że wszystkie nasze automaty spełniają następującą własność: dla każdego stanu q zachodzi $\delta(q,\omega) = \mathsf{true}$.

Przykład. Rozważmy język drzew (nad ustalonym alfabetem Σ), które nie zawierają litery b. Aby rozpoznać ten język, wystarczy automat z jednym stanem. Stan ten jest oczywiście akceptujący. Definiujemy $\delta(q,b) = \emptyset$ oraz $\delta(q,a) = \{(q,\ldots,q)\}$ dla każdego $a \neq b$ (gdzie długość krotki $(q,\ldots,q)$ to ranga litery a).

Przykład. Rozważmy dopełnienie powyższego języka – zbiór drzew, które zawierają literę b. Można dowieść (ćwiczenie), że języka tego nie da się rozpoznać za pomocą automatu z trywialnym warunkiem akceptacji. Jest to niemożliwe nawet w sytuacji, gdy ograniczamy się do drzew nie zawierających symbolu ω. Zauważmy jednak, że jeśli chcemy stwierdzić, czy drzewo generowane przez dany λY-term zawiera symbol b, nie musimy tworzyć automatu dla języka „istnieje b": równie dobrze można użyć automat dla języka „nie istnieje b", zbudowany w poprzednim przykładzie.

Przykład. Niech L będzie językiem regularnym słów skończonych. Wówczas możemy stworzyć automat z trywialnym warunkiem akceptacji sprawdzający, czy każda skończona ścieżka drzewa (nie kończąca się symbolem ω) należy do L (ćwiczenie).

Modele – ogólnie

Problem wspomniany na wstępie (sprawdzenie, czy drzewo generowane przez dany λY-term jest akceptowane przez dany automat) rozwiązano na wiele sposobów. Omówimy tutaj rozwiązanie polegające na skonstruowaniu odpowiedniego modelu dla rachunku λY. Zanim przystąpimy do opisu tego rozwiązania, podamy parę ogólnych definicji związanych z modelami.

Strukturą aplikatywną nazwiemy kolekcję zbiorów $D[\alpha]$ dla każdego typu α, wyposażoną w operację składania, która dla dowolnych $\sigma \in D[\alpha{\to}\beta]$ i $\tau \in D[\alpha]$ daje $(\sigma\,\tau) \in D[\beta]$. Powiemy, że struktura aplikatywna jest *ekstensjonalna* jeśli w żadnym $D[\alpha{\to}\beta]$ nie ma dwóch różnych elementów opisujących tę samą funkcję z $D[\alpha]$ w $D[\beta]$; formalnie, dla dowolnych typów α, β i dowolnych $\sigma_1, \sigma_2 \in D[\alpha{\to}\beta]$, jeśli $\sigma_1\,\tau = \sigma_2\,\tau$ dla każdego $\tau \in D[\alpha]$, to $\sigma_1 = \sigma_2$. W dalszych rozważaniach mówiąc o strukturze aplikatywnej mamy na myśli wyłącznie strukturę ekstensjonalną (modele można budować również na bazie struktur, które nie są ekstensjonalne; jednak, aby je uwzględnić, poniższe definicje i lematy wymagałyby wprowadzenia dodatkowych szczegółów).

Celem modelu jest interpretacja każdego λY-termu K w pewnej konkretnej (ekstensjonalnej) strukturze aplikatywnej. Ponieważ mamy w termach zmienne wolne, interpretacja termu zależeć będzie dodatkowo od wartościowania przypisującego zmiennym wolnym pewne elementy struktury aplikatywnej. Konkretnie, *D-wartościowaniem* nazwiemy funkcję częściową v, idącą w strukturę aplikatywną, przy czym $v(x^\alpha) \in D[\alpha]$. Interpretację termu M, przy D-wartościowaniu v określonym przynajmniej na zmiennych wolnych termu M, oznaczymy $[\![M]\!]^v$. Oczywiście typy muszą być zachowane, czyli jeśli M jest typu α, to $[\![M]\!]^v \in D[\alpha]$. Taką interpretację nazwiemy *modelem* jeśli spełnia następujące warunki (dla dowolnych termów odpowiednich typów, itp.):

- $[\![x]\!]^v = v(x)$,
- $[\![K\,M]\!]^v = [\![K]\!]^v\,[\![M]\!]^v$,
- $[\![\lambda x^\alpha.K]\!]^v\,\sigma = [\![K]\!]^{v[x\mapsto\sigma]}$ (gdzie $\sigma \in D[\alpha]$),
- $[\![a\,K_1\,\ldots\,K_r]\!]^v = f_a([\![K_1]\!]^v, \ldots, [\![K_r]\!]^v)$ dla pewnej funkcji $f_a\colon (D[o])^{rank(a)} \to D[o]$,
- $[\![Yx^\alpha.K]\!]^v = F^\alpha_Y([\![\lambda x.K]\!]^v)$ dla pewnej funkcji $F^\alpha_Y\colon D[\alpha{\to}\alpha] \to D[\alpha]$ takiej, że $\sigma\,F^\alpha_Y(\sigma) = F^\alpha_Y(\sigma)$.

Ponieważ rozważamy wyłącznie struktury ekstensjonalne, warunek trzeci na powyższej liście określa jednoznacznie czym jest $[\![\lambda x^\alpha.K]\!]^v$ na postawie $[\![K]\!]^{v[x\mapsto\sigma]}$ dla różnych $\sigma \in D[\alpha]$. Swoboda w określeniu interpretacji pozostaje więc jedynie w wyborze funkcji f_a i F^α_Y. Warunek na funkcję F^α_Y mówi po prostu, że $[\![Yx^\alpha.K]\!]^v \in D[\alpha]$ ma być pewnym punktem stałym funkcji z $D[\alpha]$ do $D[\alpha]$ opisanej przez $[\![\lambda x.K]\!]^v$.

Zauważmy, że jeśli wartościowania v i w są równe na wszystkich zmiennych wolnych termu M, to $[\![M]\!]^v = [\![M]\!]^w$. Łatwo można tego dowieść przez indukcję po budowie termu M (dla termu postaci $\lambda x.K$ korzystamy z ekstensjonalności). W szczególności oznacza to, że dla termów bez zmiennych wolnych wartościowanie nie ma znaczenia; dla takich termów będziemy czasem pisać $[\![M]\!]$ zamiast $[\![M]\!]^v$, pomijając wartościowanie v.

Lemat. *Dla dowolnych termów zachodzi* $[\![M[N/x]]\!]^v = [\![M]\!]^{v[x\mapsto[\![N]\!]^v]}$.

Dowód. Indukcja po budowie termu M. W większość przypadków korzystamy po prostu z założenia indukcyjnego; dla $M = \lambda x.K$ dodatkowo z ekstensjonalności. W przypadku bazowym $M = x$ mamy

$$[\![x[N/x]]\!]^v = [\![N]\!]^v = v[x \mapsto [\![N]\!]^v](x) = [\![x]\!]^{v[x\mapsto[\![N]\!]^v]}. \qquad \square$$

Lemat. *Jeśli termy M i N są $\beta\delta$-równoważne, to* $[\![M]\!]^v = [\![N]\!]^v$.

Dowód. Wystarczy rozważać przypadek pojedynczej redukcji, odbywającej się całkowicie na zewnątrz termu. Jeśli $M = (\lambda x.K)\,L$ i $N = K[L/x]$ to

$$[\![(\lambda x.K)\,L]\!]^v = [\![\lambda x.K]\!]^v\,[\![L]\!]^v = [\![K]\!]^{v[x\mapsto[\![L]\!]^v]} = [\![K[L/x]]\!]^v.$$

Tutaj (jak również poniżej) dwie pierwsze równości wynikają z definicji modelu, a trzecia z poprzedniego lematu. Podobnie dla $M = Yx.K$ i $N = K[Yx.K/x]$ mamy

$$[\![Yx.K]\!]^v = [\![\lambda x.K]\!]^v\,[\![Yx.K]\!]^v = [\![K]\!]^{v[x\mapsto[\![Yx.K]\!]^v]} = [\![K[Yx.K/x]]\!]^v. \qquad \square$$

Kraty i największe punkty stałe

Zbiór skończony wyposażony w porządek częściowy nazwiemy *kratą*, jeśli istnieje w nim element najmniejszy i największy.[1] Dla dwóch krat skończonych A i B przez $Mon(A,B)$ oznaczmy zbiór funkcji monotonicznych z A w B. Funkcje te porządkujemy po współrzędnych: $f \le g$ jeśli $f(x) \le g(x)$ dla każdego $x \in A$. Widzimy, że $Mon(A,B)$ z tym porządkiem jest kratą (element najmniejszy/największy to funkcja mapująca każdy element A na najmniejszy/największy element B).

Lemat. *Dla dowolnej kraty skończonej A, każda funkcja $f \in Mon(A,A)$ ma największy punkt stały.*

Dowód. Niech $a_0 \in A$ będzie elementem największym. Iterujemy: $a_{i+1} = f(a_i)$. Ponieważ $a_1 \le a_0$ (bo a_0 jest największy), przez indukcję dostajemy z monotoniczności $a_{i+2} = f(a_{i+1}) \le f(a_i) = a_{i+1}$. Nasza krata jest skończona, więc po pewnym czasie (po najwyżej $|A|$ krokach) ten ciąg nierosnący stabilizuje się: dla pewnego n mamy $a_{n+1} = f(a_n) = a_n$. Element a_n jest punktem stałym funkcji f.

W dodatku jest on największym punktem stałym. Rozważmy bowiem dowolny inny punkt stały x, tzn. $f(x) = x$. Ponieważ $x \le a_0$, to przez indukcję dostajemy z monotoniczności $x = f(x) \le f(a_i) = a_{i+1}$. W szczególności $x \le a_n$. $\square$

Największy punkt stały funkcji f oznaczmy przez $GFP(f)$. Jak widać z powyższego dowodu, możemy go obliczyć n-krotnie aplikując funkcję f do elementu największego, dla dowolnego $n \ge |A|$.

[1] Dla zbiorów nieskończonych definicja kraty jest bardziej skomplikowana.

Modele dla automatów

Niech $A = (\Sigma, Q, F, \delta)$ będzie niedeterministycznym automatem z trywialnym warunkiem akceptacji ignorującym omegę (czyli takim, że $\delta(q, \omega) = \mathsf{true}$ zachodzi dla każdego stanu q). Zdefiniujemy model taki, że patrząc na interpretację zamkniętego termu typu o będziemy w stanie stwierdzić, czy drzewo generowane przez ten term jest akceptowane przez A.

Naszą strukturę aplikatywną definiujemy przez indukcję po typie w następujący sposób: $D[o] = \mathcal{P}(Q)$ oraz $D[\alpha{\to}\beta] = Mon(D[\alpha], D[\beta])$, przy czym porządek na $\mathcal{P}(Q)$ to $\subseteq$ (zawieranie się zbiorów). Interpretację termów definiujemy przez indukcję po ich budowie. Bierzemy $[\![Yx.K]\!]^v = GFP([\![\lambda x.K]\!]^v)$ oraz

$$[\![a\, K_1\, \ldots\, K_r]\!]^v = \{q \mid \delta(q, a) \cap ([\![K_1]\!]^v \times \cdots \times [\![K_r]\!]^v) \neq \emptyset\}\,.$$

Innymi słowy, $[\![a\, K_1\, \ldots\, K_r]\!]^v$ zawiera te stany q, że po wczytaniu litery a nasz automat może przejść do krotki stanów $(q_1, \ldots, q_r)$ takiej, że $q_i \in [\![K_i]\!]^v$ dla $i \in \{1, \ldots, r\}$. Dla termów innej postaci postępujemy w jedyny sposób dozwolony przez definicję modelu. Aby stwierdzić, że $[\![\lambda x.K]\!]^v$ zdefiniowany w ten sposób jest faktycznie funkcją monotoniczną (czyli w ogóle należy do naszej struktury aplikatywnej) musimy wiedzieć, że $[\![K]\!]^{v[x\mapsto\sigma]} \leq [\![K]\!]^{v[x\mapsto\tau]}$ dla $\sigma \leq \tau$. Jest to prosta indukcja po budowie termu K.

Twierdzenie. *Dla zdefiniowanego powyżej modelu oraz dla zamkniętego termu M typu o, generującego drzewo t, zachodzi równoważność: $[\![M]\!] \cap F \neq \emptyset$ wtedy i tylko wtedy, gdy t jest akceptowane przez A.*

W pozostałej części wykładu skupimy się na dowodzie powyższego twierdzenia. Dowód implikacji z lewej w prawą jest prosty. Będziemy dowodzić silniejszą własność: dla każdego $q \in [\![M]\!]$ skonstruujemy bieg A na t mający q w korzeniu (implikuje to tezę lematu, gdy weźmiemy $q \in [\![M]\!] \cap F$). Jeśli w korzeniu t jest ω, to oczywiście taki bieg istnieje, bo $\delta(q, \omega) = \mathsf{true}$. W przeciwnym przypadku mamy $M \to^*_{\beta\delta} a\, M_1\, \ldots\, M_r$. Udowodniliśmy jednak, że redukcje nie zmieniają wartości w modelu, zatem $q \in [\![a\, M_1\, \ldots\, M_r]\!] = [\![M]\!]$. Z definicji $[\![a\, M_1\, \ldots\, M_r]\!]$ wynika, że $\delta(q, a) \cap ([\![M_1]\!] \times \cdots \times [\![M_r]\!]) \neq \emptyset$, czyli że istnieją stany $(q_1, \ldots, q_r) \in \delta(q, a)$ takie, że $q_i \in [\![M_i]\!]$ dla $i \in \{1, \ldots, r\}$. Postępując w ten sam sposób dla q_i i poddrzewa generowanego przez M_i dostajemy dalszy ciąg biegu. Po nieskończenie wielu takich krokach (w granicy) dostajemy etykietowanie całego drzewa stanami. Jest ono poprawnym biegiem, gdyż dla automatów z trywialnym warunkiem akceptacji musimy jedynie zadbać lokalnie o zachowanie funkcji przejścia; nie ma żadnego warunku akceptacji na ścieżkach nieskończonych.

Dowód w drugą stronę jest nieco trudniejszy, potrzebne jest kilka kroków. Na początku rozważymy termy bez operatora rekurencji.

Lemat. *Niech M będzie zamkniętym termem typu o, generującym drzewo t. Jeśli w M nie pojawia się operator rekurencji Y oraz istnieje bieg A na t mający q w korzeniu, to $q \in [\![M]\!]$.*

Dowód. Kluczowy jest fakt, że t jest skończone, możemy więc przeprowadzić indukcję po jego wysokości (natomiast w ogólnym przypadku, dla nieskończonego drzewa t, nie ma po czym przeprowadzić indukcji). Term M redukuje się do $a\, M_1\, \ldots\, M_r$ (ponieważ M nie zawiera operatora Y, odpada nam możliwość, że M nie redukuje się do termu tej postaci), gdzie a jest etykietą korzenia t, natomiast termy $M_1, \ldots, M_r$ generują poddrzewa $t_1, \ldots, t_r$ drzewa t rozpoczynające się w dzieciach korzenia. Niech $q_1, \ldots, q_r$ będą stanami przypisanymi dzieciom korzenia. Wówczas z założenia indukcyjnego dla t_i (fragment biegu na t jest biegiem na t_i) dostajemy $q_i \in [\![M_i]\!]$ dla $i \in \{1, \ldots, r\}$. Mamy $(q_1, \ldots, q_r) \in \delta(q, a)$, czyli $\delta(q, a) \cap ([\![M_1]\!] \times \cdots \times [\![M_r]\!]) \neq \emptyset$, co implikuje $q \in [\![a\, M_1\, \ldots\, M_r]\!] = [\![M]\!]$. □

Pozostaje uogólnić powyższe rozważanie do termów zawierających operatory rekurencji Y. W tym celu rozwiniemy każdy taki operator dostatecznie wiele razy, a następnie utniemy term. Dla dowolnego termu K i $n \in \mathbb{N}$ zdefiniujemy dwa termy: $\exp_n(K)$ i $\mathrm{cut}_n(K)$. Pierwszy z nich powstaje przez n-krotne rozwinięcie każdego operatora Y, natomiast w drugim dodatkowo „kończymy" rekurencję po n-krotnym rozwinięciu wypisaniem omegi. Dla dowolnego typu $\alpha = \alpha_1{\to}\ldots{\to}\alpha_k{\to}o$ niech $\Omega^\alpha = \lambda x_1^{\alpha_1}.\cdots.\lambda x_k^{\alpha_k}.\omega$. Definiujemy $\exp_0^{\mathrm{loc}}(Yx.K) = Yx.K$ oraz $\mathrm{cut}_0^{\mathrm{loc}}(Yx^\alpha.K) = \Omega^\alpha$. Następnie już tak samo dla obu funkcji: dla $f \in \{\exp, \mathrm{cut}\}$ bierzemy $f^{\mathrm{loc}}_{i+1}(Yx.K) = K[f^{\mathrm{loc}}_i(Yx.K)/x]$ (przykładowo $\exp_2^{\mathrm{loc}}(Yx.K) = K[K[Yx.K/x]/x]$). To było lokalnie dla najbardziej zewnętrznego operatora Y, natomiast $\exp_i$ i cut_i mają wykonywać powyższe przekształcenie dla każdego operatora Y w termie, co definiujemy przez indukcję po jego budowie, dla $f \in \{\exp, \mathrm{cut}\}$ biorąc:

$$f_n(x) = x\,, \qquad f_n(\lambda x.K) = \lambda x.f_n(K)\,, \qquad f_n(a\, K_1\, \ldots\, K_r) = a\, f(K_1)\, \ldots\, f(K_r)\,.$$
$$f_n(K\, L) = f_n(K)\, f_n(L)\,, \qquad f_n(Yx.K) = f_n^{\mathrm{loc}}(Yx.f_n(K))\,,$$

Widzimy, że $M \to^*_\delta \exp_n(M)$; w szczególności term $\exp_n(M)$ generuje drzewo t (to samo co M). Następnie zauważmy, że drzewo generowane przez $\mathrm{cut}_n(M)$ różni się od t tylko tym, że pewne poddrzewa t zostały zastąpione

pojedynczymi wierzchołkami etykietowanymi omegą. Intuicyjnie jest to jasne: $\mathrm{cut}_n(M)$ różni się od $\exp_n(M)$ tylko tylko tym, że pewne podtermy zostały zastąpione omegą. Spróbujmy uzasadnić to bardziej formalnie. Powiemy, że term L jest obcięciem termu K, jeśli powstaje z K poprzez zamianę pewnych podtermów przez termy postaci Ω^α (dla odpowiednich typów α). W szczególności $\mathrm{cut}_n(M)$ jest obcięciem $\exp_n(M)$, nasza obserwacja wynika więc z poniższego lematu.

Lemat. *Niech M będzie zamkniętym termem typu o generującym drzewo t, a N pewnym jego obcięciem. Wówczas drzewo generowane przez N różni się od t tylko tym, że pewne poddrzewa t zostały zastąpione pojedynczymi wierzchołkami etykietowanymi omegą.*

Dowód. Zauważmy najpierw, że jeśli L jest obcięciem K i nie jest postaci $\Omega^\alpha\, L_1 \ldots L_k$ oraz wykonanie czołowej redukcji z L prowadzi do L', to wykonanie czołowej redukcji z K prowadzi do takiego termu K', że L' jest jego obcięciem (w K mamy po prostu ten sam czołowy redeks, tylko nie jest on przycięty).

Rozważamy trzy przypadki. Jeśli N redukuje się do termu postaci $\Omega^\alpha\, L_1 \ldots L_k$, to term ten w k krokach redukuje się do symbolu ω, więc teza zachodzi. Podobnie, jeśli N nie redukuje się do termu postaci $a\, N_1 \ldots N_r$, to w korzeniu drzewa generowanego przez N jest ω i teza zachodzi. W przeciwnym przypadku N redukuje się za pomocą redukcji czołowych do termu postaci $a\, N_1 \ldots N_r$, bez przejścia przez termy postaci $\Omega^\alpha\, L_1 \ldots L_k$. Wówczas z powyższej obserwacji wiemy, że M redukuje się do pewnego M', którego obcięciem jest $a\, N_1 \ldots N_r$. Musi być $M' = a\, M_1 \ldots M_r$, gdzie N_i jest obcięciem M_i dla $i \in \{1, \ldots, r\}$. Zatem korzeń obu drzew ma tę samą etykietę. Kontynuując dalej w ten sam sposób dostajemy tezę lematu na coraz to głębszych poziomach. □

Skoro więc drzewo t' generowane przez $\mathrm{cut}_n(M)$ powstaje z t poprzez zamianę pewnych poddrzew na omegi, to bieg akceptujący A na t pozostaje poprawnym biegiem także na t' (po ograniczeniu go do tych wierzchołków, które faktycznie występują w t'). Niech $q \in F$ będzie stanem przypisanym korzeniowi w tym biegu. Widzimy, że w termie $\mathrm{cut}_n(M)$ nie występują operatory Y, więc z poprzedniego lematu dostajemy $q \in [\![\mathrm{cut}_n(M)]\!]$, czyli $[\![\mathrm{cut}_n(M)]\!] \cap F \neq \emptyset$.

Powyższe rozumowanie było prawdziwe dla dowolnych n. Pozostaje dowieść, że dla wystarczająco dużych n zajdzie $[\![\mathrm{cut}_n(M)]\!] = [\![M]\!]$. Wynika to łatwo (przez indukcję po budowie termu M) z poniższego lematu, jeśli za n weźmiemy liczbę większą niż $|D[\alpha]|$ dla wszystkich α będących typami podtermów M.

Lemat. *Jeśli $n \geq |D[\alpha]|$, to $[\![\mathrm{cut}^{\mathrm{loc}}_n(Y x^\alpha.K)]\!]^v = [\![Y x^\alpha.K]\!]^v$.*

Dowód. Niech $\alpha = \alpha_1 \to \ldots \to \alpha_k \to o$. Zobaczmy, że $[\![\Omega^\alpha]\!]^v = [\![\lambda x_1^{\alpha_1}. \cdots . \lambda x_k^{\alpha_k}.\omega]\!]^v$ to funkcja mapująca swoje argumenty $\sigma_1, \ldots, \sigma_k$ na $[\![\omega]\!]^{v[x_1 \mapsto \sigma_1, \ldots, x_k \mapsto \sigma_k]} = \{q \mid \delta(q, \omega) = \mathsf{true}\} = Q$. Zatem $[\![\Omega^\alpha]\!]^v$ jest elementem największym kraty $D[\alpha]$.

Z definicji $[\![Y x^\alpha.K]\!]^v$ to największy punkt stały funkcji $[\![\lambda x^\alpha.K]\!]^v$. Jak zaobserwowaliśmy wcześniej, największy punkt stały funkcji monotonicznej na kracie skończonej $D[\alpha]$ możemy obliczyć aplikując n-krotnie tę funkcję do elementu największego kraty, dla dowolnego $n \geq |D[\alpha]|$. Mamy więc:

$$[\![Y x^\alpha.K]\!]^v = GFP([\![\lambda x^\alpha.K]\!]^v) = \underbrace{[\![\lambda x^\alpha.K]\!]^v\, (\ldots\, ([\![\lambda x^\alpha.K]\!]^v}_{n}\, [\![\Omega^\alpha]\!]^v) \ldots) = [\![\underbrace{(\lambda x^\alpha.K)\, (\ldots\, ((\lambda x^\alpha.K)}_{n}\, \Omega^\alpha) \ldots)]\!]^v .$$

Ten ostatni term redukuje się (po wykonaniu n widocznych w nim redukcji) do $\mathrm{cut}^{\mathrm{loc}}_n(Y x^\alpha.K)$. Mamy zatem $[\![Y x^\alpha.K]\!]^v = [\![\mathrm{cut}^{\mathrm{loc}}_n(Y x^\alpha.K)]\!]^v$. □

Powyższą konstrukcję modelu przeprowadziliśmy dla automatu niedeterministycznego. Łatwo można ją uogólnić na automaty alternujące: wystarczy poprawić definicję $[\![a\, K_1 \ldots K_r]\!]^v$ tak, aby uwzględniała funkcję przejścia automatu alternującego zamiast niedeterministycznego (zbiory $D[\alpha]$ pozostają bez zmian, w szczególności nadal $D[o] = \mathcal{P}(Q)$). Przy tym uogólnieniu złożoność algorytmu (obliczona poniżej) nie zmieni się.

Złożoność

Na koniec udowodnimy, że jeśli maksymalna złożoność wejściowego termu jest ustalona na n, to przedstawiony powyżej algorytm działa w czasie n-krotnie wykładniczym.

Trzeba w tym celu przede wszystkim oszacować rozmiar poszczególnych składowych modelu. Zdefiniujmy rozmiar $|\alpha|$ typu α jako:

$$|o| = 1\,, \qquad\qquad |\alpha \to \beta| = |\alpha| + |\beta|\,,$$

natomiast rozmiar $|M|$ termu M jako:

$$|x| = 1\,, \quad |M\, N| = |M| + |N| + 1\,, \quad |\lambda x.M| = |Y x.M| = 1 + |M|\,, \quad |a\, N_1 \ldots N_r| = 2 + |N_1| + \cdots + |N_r|\,.$$

Niech $\exp_0(k) = k$ oraz $\exp_{n+1} = 2^{\exp_n(k)}$. Przez $\mathrm{esize}(D[\alpha])$ oznaczymy liczbę bitów potrzebnych do reprezentacji elementów zbioru $D[\alpha]$; mamy $|D[\alpha]| \leq 2^{\mathrm{esize}(D[\alpha])}$. Ponadto, niech $\mathrm{height}(D[\alpha])$ będzie największą liczbą k, że w $D[\alpha]$ istnieje ciąg zstępujący długości $k+1$.

Udowodnimy przez indukcję po $|\alpha|$, że $\max(\mathrm{esize}(D[\alpha]), \mathrm{height}(D[\alpha])) \leq \exp_{\mathrm{ord}(\alpha)}(|\alpha| \cdot |Q|)$.

Rozważmy najpierw $\alpha = o$. Wówczas elementy $D[\alpha]$ to podzbiory Q. Można je reprezentować za pomocą $|Q|$ bitów. Ciąg zstępujący w $D[\alpha]$ zawiera coraz to mniejsze podzbiory Q (pod względem zawierania, a więc także rozmiaru). Największy z tych podzbiorów ma $|Q|$ elementów, więc będzie ich co najwyżej $|Q|+1$. Mamy więc faktycznie $\max(\mathrm{esize}(D[\alpha]), \mathrm{height}(D[\alpha])) \leq |Q| = \exp_{\mathrm{ord}(\alpha)}(|\alpha| \cdot |Q|)$.

Weźmy teraz typ postaci $\alpha{\to}\beta$. Elementy $D[\alpha{\to}\beta]$ to funkcje (monotoniczne) z $D[\alpha]$ w $D[\beta]$. Możemy je reprezentować pamiętając dla każdego z $|D[\alpha]|$ elementów dziedziny jaka jest wartość funkcji. Potrzebujemy zatem $|D[\alpha]| \cdot \mathrm{esize}(D[\beta])$ bitów. Podobnie jest ze zstępującym ciągiem funkcji: w każdym kolejnym kroku musimy zmniejszyć wartość funkcji dla przynajmniej jednego argumentu. Możemy to zrobić $|D[\alpha]| \cdot \mathrm{height}(D[\beta])$ razy. Mamy więc

$$\begin{aligned}\mathrm{esize}(D[\alpha{\to}\beta]) &\leq |D[\alpha]| \cdot \mathrm{esize}(D[\beta]) \leq 2^{\mathrm{esize}(D[\alpha])} \cdot \mathrm{esize}(D[\beta]) \leq 2^{\exp_{\mathrm{ord}(\alpha)}(|\alpha|\cdot|Q|)} \cdot \exp_{\mathrm{ord}(\beta)}(|\beta| \cdot |Q|) = \\ &= \exp_{\mathrm{ord}(\alpha)+1}(|\alpha| \cdot |Q|) \cdot \exp_{\mathrm{ord}(\beta)}(|\beta| \cdot |Q|)\,.\end{aligned}$$

Przypomnijmy, że $\mathrm{ord}(\alpha) + 1 \leq \mathrm{ord}(\alpha{\to}\beta)$ oraz $\mathrm{ord}(\beta) \leq \mathrm{ord}(\alpha{\to}\beta)$. Ponieważ $\exp_n(k) \leq \exp_{n+1}(k)$ oraz $\exp_n(k) \cdot \exp_n(l) \leq \exp_n(k+l)$ dla $n \geq 1$, dostajemy

$$\begin{aligned}\mathrm{esize}(D[\alpha{\to}\beta]) &\leq \exp_{\mathrm{ord}(\alpha{\to}\beta)}(|\alpha| \cdot |Q|) \cdot \exp_{\mathrm{ord}(\alpha{\to}\beta)}(|\beta| \cdot |Q|) \\ &\leq \exp_{\mathrm{ord}(\alpha{\to}\beta)}(|\alpha| \cdot |Q| + |\beta| \cdot |Q|) = \exp_{\mathrm{ord}(\alpha{\to}\beta)}(|\alpha{\to}\beta| \cdot |Q|)\,.\end{aligned}$$

Dokładnie takie same wzory dostajemy przy szacowaniu $\mathrm{height}(D[\alpha{\to}\beta])$.

Ustalmy term wejściowy M_0 o złożoności $n \geq 1$. Niech z będzie maksimum po wszystkich α będących typami podtermów M_0 z

- $\mathrm{esize}(D[\alpha])$ oraz $\mathrm{height}(D[\alpha])$,
- a jeśli $\mathrm{ord}(\alpha) \leq n-1$, to dodatkowo z $|D[\alpha]|$.

Z powyższego wiemy, że z jest (co najwyżej) n-krotnie wykładniczy względem $|Q|$ i $|M_0|$.

Udowodnimy teraz, że jeśli term M jest podtermem M_0, to nasz algorytm liczy $[\![M]\!]^v$ dla ustalonego wartościowania v w czasie $O(z^{|M|})$. Dla $M = x$ jest to oczywiste: odczytujemy tylko z v element wielkości co najwyżej z. Dla $M = K\,L$ liczymy $[\![K]\!]^v$ i $[\![L]\!]^v$ w czasie $z^{|K|} + z^{|L|}$, a następnie odczytujemy wartość funkcji $[\![K]\!]^v$ dla argumentu $[\![L]\!]^v$. Odbywa się to w czasie $z^{|K|} + z^{|L|} + z \leq z^{|K|} \cdot z^{|L|} \cdot z = z^{|M|}$. Dla $M = a\,N_1 \ldots N_r$ obliczamy $[\![N_i]\!]^v$ w czasie $z^{|N_i|}$, dla każdego i, po czym wykonujemy przejścia automatu w czasie proporcjonalnym do $|Q|^{r+1} \leq z^{r+1}$. Łącznie daje to czas $z^{|N_1|} + \cdots + z^{|N_r|} + z^{r+1} \leq z^{|N_1|} \cdot \ldots \cdot z^{|N_r|} + z^{|M|-1} = 2 \cdot z^{|M|-1} \leq z^{|M|}$.

Niech teraz $M = \lambda x.K$. Oznaczmy typ M przez $\alpha{\to}\beta$. Musimy obliczyć $[\![K]\!]^{v'}$ dla $|D[\alpha]|$ różnych wartościowań v'. Zauważmy, że $\mathrm{ord}(\alpha) < \mathrm{ord}(\alpha{\to}\beta) \leq n$, czyli $|D[\alpha]| \leq z$ z definicji z. Czas działania jest więc nie większy niż $z^{|K|} \cdot z = z^{|M|}$.

Na koniec rozważmy przypadek $M = Y x^{\alpha}.K$. Nie chcemy wówczas liczyć całej funkcji $[\![\lambda x.K]\!]^v$ występującej w definicji, gdyż może ona być zbyt duża. Naszym celem jest policzenie jej największego punktu stałego. Powinniśmy więc iterować tę funkcję, poczynając od elementu największego kraty $D[\alpha]$. Aby obliczyć wartość $[\![\lambda x.K]\!]^v$ dla konkretnego argumentu σ musimy po prostu obliczyć $[\![K]\!]^{v[x\mapsto\sigma]}$, co zajmuje nam czas $z^{|K|}$. Iteracji wykonamy co najwyżej $\mathrm{height}(D[\alpha]) \leq z$. Całkowity czas działania jest więc nie większy niż $z^{|K|} \cdot z = z^{|M|}$.

Podsumowując powyższe rozumowanie, dostajemy czas działania $z^{|M_0|}$. Widzimy, że, podobnie jak samo z, jest to liczba n-krotnie wykładnicza względem $|Q|$ i $|M_0|$ (o ile tylko $n \geq 1$).

Dla $n = 0$ powyższe szacowanie daje czas wykładniczy i taki faktycznie jest czas działania naszego algorytmu (jeśli zagnieżdżonych jest k operatorów Y, w każdym dochodzimy do punktu stałego pesymistycznie w $\Theta(|Q|)$ krokach, co daje czas $\Theta(|Q|^k)$). Algorytm można poprawić tak, aby w tym przypadku działał mimo wszystko w czasie wielomianowym, lecz nie będziemy się w to zagłębiać.

Można zauważyć, że n-krotnie wykładnicza złożoność pozostaje nawet wtedy, gdy nasz automat ma jeden stan.

Przypadek o mniejszej złożoności

Okazuje się natomiast, że złożoność można zmniejszyć do $(n-1)$-krotnie wykładniczej w przypadku, gdy automat (z trywialnym warunkiem akceptacji) jest deterministyczny z góry w dół. Warunek ten oznacza, że $|\delta(q, a)| \leq 1$ dla każdego q i a.

Rozważmy term M (zamknięty) typu $\alpha = o^k \to o$, rzędu 1. Taki term opisuje drzewo potencjalnie nieskończone, w którym jako liście mogą występować zmienne $x_1, \ldots, x_k$ (opisujące parametry). Podanie takiemu termowi k argumentów (typu o, czyli drzew) powoduje ich podstawienie za wspomniane zmienne. Zastanówmy jak może wyglądać wartość takiego termu M w modelu. Zgodnie z definicją jest to funkcja monotoniczna z $(\mathcal{P}(Q))^k$ w $\mathcal{P}(Q)$, lecz nie może to być dowolna funkcja. Jeśli bowiem ustalimy stan w korzeniu naszego drzewa to (dzięki determinizmowi automatu) wyznaczy to nam konkretne stany, które muszą pojawić się w wierzchołkach etykietowanych przez $x_1, \ldots, x_k$. Dokładniej, uzyskane funkcje f spełniają dla każdego $q \in Q$ warunek:

- zachodzi $q \notin f(Q_1, \ldots, Q_k)$ dla wszystkich $Q_1, \ldots, Q_k \subseteq Q$ (czyli nie da się uzyskać q w korzeniu drzewa) lub
- istnieją zbiory $P_1, \ldots, P_k$ takie, że $q \in f(Q_1, \ldots, Q_k)$ wtedy i tylko wtedy, gdy $P_1 \subseteq Q_1, \ldots, P_k \subseteq Q_k$ (zbiór P_i to zbiór stanów przypisanych wierzchołkom etykietowanym przez x_i w sytuacji, gdy w korzeniu drzewa jest q).

Niech $D'[o^k \to o] \subseteq D[o^k \to o]$ będzie zbiorem funkcji powyższej postaci. Ponadto pozostawiamy $D'[o] = D[o]$ oraz definiujemy przez indukcję $D'[\beta \to \gamma] = Mon(D'[\beta], D'[\gamma])$ dla wszystkich typów $\beta \to \gamma$ rzędu większego niż 1.

Model definiujemy tymi samymi wzorami co poprzednio. Pomijamy dowód, że wartość $[\![M]\!]^v$ będzie faktycznie elementem $D'[\alpha]$.

Pozostaje zobaczyć, że uzyskujemy szybszy algorytm. Zobaczmy, że aby przechować funkcję $f \in D'[o^k \to o]$, musimy dla każdego stanu q pamiętać, czy w powyższej alternatywie zachodzi przypadek pierwszy, czy drugi, a w drugim przypadku musimy pamiętać zbiory stanów $P_1, \ldots, P_k$. Potrzebujemy na to $\text{esize}(D'[o^k \to o]) = |Q| \cdot (1 + k|Q|)$ bitów. Mamy również $\text{height}(D'[o^k \to o]) = |Q| \cdot (1 + k|Q|)$, gdyż jedyny sposób, w jaki możemy pomniejszyć funkcję f, to zwiększenie któregoś ze zbiorów $P_1, \ldots, P_k$ dla któregoś ze stanów q, bądź przejście dla któregoś stanu q z przypadku drugiego do przypadku pierwszego.

Wielkości te są wielomianowe względem $|Q|$ oraz rozmiaru typu, a nie wykładnicze, jak to było wcześniej. W efekcie złożoność całego algorytmu zmniejsza się do $(n-1)$-krotnie wykładniczej (przynajmniej dla $n \geq 2$; pomijamy przypadek $n = 1$, w którym wymagana jest dodatkowa ostrożność).

Szczególnym przypadkiem automatu deterministycznego z góry w dół jest automat sprawdzający warunek bezpieczeństwa (*safety*): drzewo nie zawiera wierzchołka o etykiecie a. Aby rozpoznać ten warunek wystarczy automat o jednym stanie.