Wykład 10.

10.1 Funkcje rekurencyjne

Będziemy rozważamy funkcje o argumentach i wartościach ze zbioru liczb naturalnych. Funkcje te nie muszą być określone dla wszystkich dopuszczalnych argunemtów.

Klasa funkcji rekurencyjnych jest najmniejszą klasą takich funkcji zawierającą funkcje $I_{n,k}$ takie, że $I_{n,k}(x_1, \ldots, x_n) = x_k$, dodawanie, mnożenie oraz funkcję $ch_<$ przyjmującą wartości 0 i 1, spełniającą równoważność $ch_<(x, y) = 0 \Leftrightarrow x < y$, oraz zamkniętą ze względu na złożenie i operację minimum. Złożenie i operacja minimum to dwa sposoby definiowania kolejnych funkcji rekurencyjnych. Oba sposoby definiowania można zdefiniować podając schematy algorytmów pozwalające na obliczanie funkcji definiowanych za pomocą tych sposobów. Mając dane algorytmy obliczające funkcje f oraz $g_1, \ldots, g_n$ złożenie tych funkcji h obliczamy stosując następujący algorytm:

$y_1 := g_1(x_1, \ldots, x_m); \ldots; y_n := g_n(x_1, \ldots, x_m);$

$h := f(y_1, \ldots, y_n).$

To złożenie h definiujemy zwykle wzorem $h(x_1, \ldots, x_m) = f(g_1(x_1, \ldots, x_m), \ldots, g_n(x_1, \ldots, x_m))$. Przypuśćmy, że mamy algorytm obliczający funkcję f. Rozważmy następujący algorytm

$y := 0;$

while $f(x_1, \ldots, x_m, y) \neq 0$ do $y := y + 1;$

$h := y.$

Funkcję h obliczaną za pomocą tego algorytmu oznaczamy symbolem $h(x_1, \ldots, x_m) = \min_y(f(x_1, \ldots, x_m, y) = 0)$ i określamy jako funkcję zdefiniowaną za pomocą operacji minimum zastosowanej do funkcji f. Zauważmy, że jeżeli nawet zastosujemy operację minimum do funkcji określonej dla wszystkich możliwych argumentów, to otrzymujemy funkcję, które dla pewnych argumentów nie są określone.

Klasę funkcji rekurencyjnych uważa się za dobrą formalizację pojęcia obliczalności. Oznacza to, że jeżeli mamy funkcję, której wartości potrafimy obliczać za pomocą jakiegoś algorytmu, określoną i przyjmującą wartości w zbiorze liczb naturalnych, to funkcja ta powinna być rekurencyjna. Przekonanie o prawdziwości podanego stwierdzenia pozwala łatwo uzasadniać rekurencyjność interesujących nas funkcji (uzasadnienia takie nie są jednak formalnie poprawne). Nie budzi wątpliwości, że dla każdej funkcji rekurencyjnej istnieje algorytm obliczający jej wartości i zapętlający się dla takich danych, dla których funkcja ta nie jest określona.

Zbiór $X \subseteq N^k$ jest rekurencyjny, jeżeli dla pewnej rekurencyjnej funkcji f, określonej dla wszystkich argumentów naturalnych, równoważność $(x_1, \ldots, x_k) \in X \Leftrightarrow f(x_1, \ldots, x_k) = 0$ jest spełniona dla wszystkich $x_1, \ldots, x_k \in N$.

Zbiór $X \subseteq N^k$ jest rekurencyjnie przeliczalny, jeżeli istnieje rekurencyjny zbiór Y taki, że dla dowolnych $x_1, \ldots, x_k$ zachodzi równoważność

$$(x_1, \ldots, x_k) \in X \Leftrightarrow \exists y \, (x_1, \ldots, x_k, y) \in Y.$$

Zbiory rekurencyjne odpowiadają problemom rozstrzygalnym. Jeżeli potrafimy odpowiadać algorytmicznie na pytanie, czy dane liczby naturalne należą do zbioru X, i algorytm odpowiadający na to pytanie nie zapętla się dla żadnych danych, to zbiór X jest rekurencyjny. Zbiory rekurencyjnie przeliczalne odpowiadają zaś problemom rozpoznawalnym. Jeżeli mamy algorytm udzielający pozytywnej odpowiedzi na pytanie, czy dane liczby naturalne należą do zbioru X wtedy i tylko wtedy, gdy dane rzeczywiście należą do X, to najprawdopodobniej zbiór X jest rekurencyjnie przeliczalny (taki algorytm może zapętlać się dla danych spoza zbioru X). Wątpliwości zawarte w przytoczonych sformułowaniach są spowodowane tym, że nie jesteśmy zgodni co do definicji pojęcia algorytmu. Pojęcie to formalizujemy na wiele sposobów. Dla większości formalizacji sformułowane twierdzenia dają się dowieść. Wyjątkami są formalizacje, które należy uznać za zbyt restrykcyjne.

Niech $f : N^k \to N$. Przyjmijmy, że

$$W_f = \{(n_1, \ldots, n_k, m) \in N : f \text{ jest określona dla argumentu } (n_1, \ldots, n_k) \wedge f(n_1, \ldots, n_k) = m\}.$$

zbiór W_f nazywamy wykresem funkcji f. Dowodzi się następujący lemat:

Lemat 10.1 *Funkcja $f : N^k \to N$ jest rekurencyjna wtedy i tylko wtedy gdy wykres W_f jest rekurencyjnie przeliczalny.*

Dowód. Prawdziwość tego lematu łatwo sprawdza się posługując intuicyjnym pojęciem obliczalności. Jeżeli f jest rekurencyjna, to algorytm sprawdzający, czy $(n, m) \in W_f$, oblicza wartość $f(n)$ i przekazuje wynik porównania obliczonej wartości i liczby m (pamiętajmy, że algorytm obliczający $f(n)$ może zapętlić się). Jeżeli mamy algorytm rozpoznający W_f (algorytm ten może się zapętlać dla niektórych danych), to aby obliczyć $f(n)$ przeglądamy wszystkie możliwe pary liczb t i m tak długo, aż znajdziemy taką parę, że w czasie t stwierdzimy, że $(n, m) \in W_f$. □

Lemat 10.2 *Zbiór R jest rekurencyjnie przeliczalny wtedy i tylko wtedy, gdy jest dziedziną pewnej funkcji rekurencyjnej.*

Dowód. Jeżeli R jest rekurencyjnie przeliczalny, to funkcję określoną w zbiorze R i przyjmującą stale wartość 0 obliczamy stosując następujący algorytm: dla danego argumentu uruchamiamy algorytm sprawdzający, czy należy on do zbioru R. Jeżeli algorytm to stwierdzi, to zwracamy 0, a jeżeli zaprzeczy temu, to uruchamiamy dodatkowo program zapętlający się. Funkcja określoną w zbiorze R i przyjmującą stale wartość 0 jest więc funkcją rekurencyjną.

Aby sprawdzić, czy dane należą do dziedziny funkcji rekurencyjnej f próbujemy dla tych danych obliczyć wartość funkcji f i jeżeli to się powiedzie, to zwracamy informację, że te dane należą do dziedziny f. Podany algorytm świadczy o rozpoznawalności dziedziny funkcji obliczalnej. □

10.2 Predykaty obliczalne

Będziemy rozważać język zawierający dwa wyróżnione symbole: S i 0. Pierwszy z nich jest jednoargumentowym symbolem funkcyjnym, a drugi – stałą. Jeżeli będzie to potrzebne, S

będziemy interpretować operację następnika (operację dodawania 1 do danej liczby naturalnej), a 0 – jako zero. Niech $\underline{n}$ oznacza term $S(\ldots(S(0))\ldots)$, w którym występuje n symboli S. Mówimy, że program P oblicza relację $R \subseteq N^k$ używając symbolu r wtedy i tylko wtedy, gdy dla dowolnych $t_1, \ldots, t_k \in N$ zachodzi równoważność

$(n_1, \ldots, n_k) \in R \Leftrightarrow$ istnieje *SLD*-wyprowadzenie z P klauzuli pustej, rozpoczynające się $\leftarrow r(\underline{n}_1, \ldots, \underline{n}_k)$.

Lemat 10.3 *Jeżeli program P oblicza relację R, to zbiór R jest rekurencyjnie przeliczalny.*

Dowód. Łatwo można sobie wyobrazić algorytm sprawdzający, czy dane liczby należą do zbioru podanego w sformułowaniu lematu. Algorytm taki może przeglądać wszystkie *SLD*-wyprowadzenia i sprawdzać, czy wśród nich jest wyprowadzenie klauzuli pustej rozpoczynające się klauzulą $\leftarrow r(\underline{n}_1, \ldots, \underline{n}_k)$. Istnienie takiego algorytmu oznacza rekurencyjną przeliczalność rozważanego zbioru. □

Lemat 10.4 *Niech P będzie programem z dwoma klauzulami:*

$$r(x, \underline{0}, x) \leftarrow \quad \textit{oraz} \quad r(x, S(y), S(z)) \leftarrow r(x, y, z).$$

Program P oblicza relację $W_+ \subseteq N^3$ używając symbolu r.

Dowód. Najpierw przez indukcję ze względu na m pokazujemy, że jeżeli $n+m=k$, to istnieje *SLD*-wyprowadzenie z P klauzuli pustej, rozpoczynające się klauzulą $\leftarrow r(\underline{n}, \underline{m}, \underline{k})$. Potrzebne wyprowadzenia konstruujemy bez trudu.

Następnie także przez indukcję ze względu na m pokazujemy, że jeżeli istnieje *SLD*-wyprowadzenie z P klauzuli pustej, rozpoczynające się klauzulą $\leftarrow r(\underline{n}, \underline{m}, \underline{k})$, to $n+m=k$. W dowodzie istotne jest spostrzeżenie, że w pierwszym kroku wyprowadzenia rozpoczynającego się klauzulą $\leftarrow r(\underline{n}, \underline{0}, \underline{k})$, klauzulę tę unifikujemy z pierwszą klazulą z programu P, a w pozostałych wyprowadzeniach – z drugą. Możliwość unifikacji klauzul $\leftarrow r(\underline{n}, \underline{0}, \underline{k})$ i $\leftarrow r(x, \underline{0}, x)$ implikuje, że $n = k$. Jeżeli wyprowadzenie rozpoczyna się klauzulą $\leftarrow r(\underline{n}, \underline{m+1}, \underline{k})$, to $k > 0$ i drugą klauzulą w tym wyprowadzeniu jest $\leftarrow r(\underline{n}, \underline{m}, \underline{k-1})$. Z założenia indukcyjnego otrzymujemy więc, że $n + m = k - 1$. Stąd oczywiście wynika, że $n + (m+1) = k$. □

Twierdzenie 10.5 *Funkcja $f : N^k \to N$ jest rekurencyjna wtedy i tylko wtedy, gdy pewien program oblicza W_f.*

Dowód. Wobec lematów 10.3 i 10.1 wystarczy dowieść istnienia programów obliczających wykresy W_f dla dowolnej funkcji rekurencyjnej f. Fakt ten dowodzimy przez indukcję ze względu na sposób definiowania f. Najpierw sprawdzamy więc, czy istnieją programy obliczające wykresy funkcji $I_{n,k}$, dodawania, mnożenia i $ch_<$. Istnienie takiego programu w przypadku dodawania wynika z lematu 10.4. Istnienie programów dla pozostałych, wymienionych funkcji dowodzimy podobnie.

Pokażemy jeszcze, że jeżeli program P oblicza wykres funkcji f (używając symbolu p), to istnieje także program obliczający wykres funkcji $g(x) = \min_y(f(x,y) = 0)$. Rozważmy program P' złożony z klauzul programu P oraz następujących klauzul:

$s(x, \underline{0}) \leftarrow,$

$s(x, S(y)) \leftarrow s(x, y), p(x, y, S(z)),$

$r(x, y) \leftarrow s(x, y), p(x, y, \underline{0}).$

Zauważmy, że istnieje *SLD*-wyprowadzenie z P' klauzuli pustej, rozpoczynające się $\leftarrow s(\underline{n}, \underline{k})$ wtedy i tylko wtedy, gdy dla wszystkich $m < k$ wartość $f(n, m)$ jest określona i $\neq 0$.

Udowodnimy to przez indukcję ze względu na k. Jest to oczywiste dla $k = 0$. Przypuśćmy więc, że $f(n, m)$ jest określone i $\neq 0$ dla wszystkich $m < k+1$. Z założenia indukcyjnego wynika, że istnieją wyprowadzenia klauzuli pustej rozpoczynające się $\leftarrow s(\underline{n}, \underline{k})$ oraz $\leftarrow p(\underline{n}, \underline{k}, S(\underline{m}))$ dla pewnej liczby m (dla $m = f(n, k) - 1$). Klauzule $s(\underline{n}, \underline{k})$ oraz $p(\underline{n}, \underline{k}, S(\underline{m}))$ są konsekwencjami logicznymi programu P' (patrz twierdzenie 2.3). Wobec tego cel $\leftarrow s(\underline{n}, \underline{k}), p(\underline{n}, \underline{k}, S(\underline{m}))$ jest sprzeczny z P' i z twierdzenie 6.3 o zupełności rezolucji wynika, że istnieje wyprowadzenie z P' klauzuli pustej rozpoczynające się tym celem. Z lematu o podstawianiu 5.2 wynika, że istnieje także wyprowadzenie rozpoczynające się celem $\leftarrow s(\underline{n}, \underline{k}), p(\underline{n}, \underline{k}, S(z))$. Zauważmy, że że ten cel jest rezolwentą $\leftarrow s(\underline{n}, \underline{k+1})$ oraz $s(x, S(y)) \leftarrow s(x, y), p(x, y, S(z))$. Wobec tego istnieje także wyprowadzenie klauzuli pustej rozpoczynające się celem $\leftarrow s(\underline{n}, \underline{k+1})$.

Implikacja odwrotna wynika z analizy wyprowadzenia rozpoczynającego się klauzulą $\leftarrow s(\underline{n}, \underline{k+1})$. Drugim wyrazem tego wyprowadzenia jest klauzula $\leftarrow s(\underline{n}, \underline{k}), p(\underline{n}, \underline{k}, S(z))$. Z twierdzenia 2.3 i wniosku 4.5 wynika, że w minimalnym modelu Herbranda dla programu P' są spełnione formuły $s(\underline{n}, \underline{k})$ oraz $p(\underline{n}, \underline{k}, S(\underline{m}))$ dla pewnego m. Wobec lematu 6.2, istnieją wyprowadzenia z P' klauzul pustych rozpoczynających się celami $\leftarrow s(\underline{n}, \underline{k})$ oraz $\leftarrow p(\underline{n}, \underline{k}, S(\underline{m}))$. Z założenia indukcyjnego wartość $f(n, t)$ jest określona i $\neq 0$ dla wszystkich $t < k$. Wyprowadzenie z P' klauzuli pustej rozpoczynające się $\leftarrow p(\underline{n}, \underline{k}, S(\underline{m}))$ jest także wyprowadzeniem z P (nie korzystamy w nim z klauzul z następnikami z symbolami s i r). Wobec tego także $f(n, k)$ jest określone i równe $m + 1$. Tak więc wartość $f(n, t)$ jest określona i $\neq 0$ dla wszystkich $t < k + 1$.

Z udowodnionej własności programu P' wynika, że oblicza on wykres funkcji f używając symbolu r. Dowodzimy metodami analogicznymi do wyżej przedstawionych. W podobny sposób dowodzimy także, że jeżeli pewne funkcje są obliczane przez odpowiednie programy, to także wykres ich złożenia jest obliczane przez odpowiednio określony, i łatwy do zdefiniowania program. □

Twierdzenie 10.6 *Zbiór R jest rekurencyjnie przeliczalny wtedy i tylko wtedy, gdy jest obliczany przez pewien program.*

Dowód. Jeżeli zbiór R jest rekurencyjnie przeliczalny, to jest dziedziną rekurencyjnej funkcji f (patrz lemat 10.2). Przyjmijmy, że wykres funkcji f jest obliczany przez program P za pomocą symbolu p. Nietrudno sprawdzić, że program $P \cup \{r(x) \leftarrow p(x, y)\}$ oblicza R używając symbolu r.

Implikacja odwrotna wynika z lematu 10.3. □