Mechanika, komputer, człowiek (V)

Prof. dr Dominik ROGULA

Programowanie heurystyczne

Jeżeli zadania niealgorytmizowalne miałyby nie nadawać się do maszynowego wykonywania, to należałoby sądzić, że wszelkie próby pisania programów komputerowych dla takich zadań są z góry skazane na niepowodzenie i nie warto w ogóle ich podejmować. Tymczasem dla wielu zadań niealgorytmizowalnych zostały napisane skutecznie działające programy ich rozwiązywania. Można tutaj wymienić programy Gelertnera do dowodzenia twierdzeń geometrycznych, Hao Wanga do dowodzenia twierdzeń logicznych, program GPS (General Program Solver) Nevella, Shawa i Simona, program SAINT (Symbolic Automatic INTegrator) Slagle'a i wiele innych.

Jak to jest możliwe? Czyżby jednak maszyna mogła wykonywać zadania nieałgorytmizowalne? Otóż odpowiedź na ostatnie pytanie jest w pewnym sensie twierdząca. Odpowiedź ta brzmi wprawdzie „nie", jeżeli żądać od maszyny pewności osiągnięcia wyniku, tzn. działania według reguł a priori zapewniających osiągalność wyniku dla każdego zadania szczegółowego. Jeżeli jednak dopuścić możliwość niepowodzenia w pewnych przypadkach, to odpowiedź brzmi „tak". Programowanie tego rodzaju nie może już polegać na wykorzystaniu algorytmu, który miałby być niezawodną procedurą na wszystkie możliwe okazje, a który przecież nie istnieje. Polega ono na stosowaniu tzw. heurystyk, czyli „dobrych rad", mówiących jakie próby w różnych sytuacjach warto podejmować. Jest to podejście zwane programowaniem heurystycznym w szerokim sensie tego terminu. Programy oparte na tej zasadzie mogą być lepsze lub gorsze w zależności od zastosowanych heurystyk i sposobu posługiwania się nimi. Im lepsze heurystyki, tym lepsze programy, czyli tym rzadziej nie umiejące poradzić sobie z postawionym zadaniem. Niealgorytmizowalność oznacza tylko, że nie można napisać „absolutnie dobrego" programu.

Aby lepiej uzmysłowić sobie praktyczną rolę programowania heurystycznego, posłużymy się analogią z popularnym sposobem rozwiązywania zadań matematyki stosowanej za pomocą szeregów nieskończonych. Do niedawna wydawało się, że metoda taka jest dobra, jeżeli otrzymuje się szereg zbieżny, i niedobra, jeżeli rozbieżny. Z czasem jednak wyjaśniło się, że zbieżność szeregu, aczkolwiek istotna w ogólnych zagadnieniach teoretycznych, jest mniej ważna w konkretnych obliczeniach, gdzie trzeba poprzestać na przybliżeniu określonym skończoną liczbą wyrazów. Zbieżność szeregu bowiem ani nie gwarantuje, że ustalona skończona liczba jego wyrazów da dobre przybliżenie, ani też nie jest do tego celu niezbędna: często udaje się uzyskać asymptotyczne rozwinięcie, wyrażone szeregiem rozbieżnym, lecz takim, że kilka jego pierwszych wyrazów przybliża rozwiązanie z wystarczającą dokładnością.

W programowaniu heurystycznym nie chodzi wprawdzie o liczbowe przybliżanie rozwiązań problemów obliczeniowych, lecz o stosowanie procedur, które są niepewne w tym sensie, że w niektórych przypadkach mogą nie znaleźć rozwiązania, lecz jeśli rozwiązanie zostanie znalezione, to na pewno dobre. „Przybliżony" jest jak gdyby sposób poszukiwania rozwiązań. Zastosowanie takiego podejścia może być podyktowane faktem, że „ścisły" sposób poszukiwania rozwiązania (algorytm) nie istnieje, a więc że zadanie jest nieałgorytmizowalne. Nie jest to jednak jedyny możliwy powód: może się zdarzyć, że algorytm zadania, choć istnieje, jest nieznany. Albo znany, lecz trudny i nadmiernie uciążliwy. W takich sytuacjach programy heurystyczne mogą stanowić praktyczne rozwiązanie.

Przykładem może tutaj być gra w szachy. Jako gra skończona jest ona oczywiście algorytmizowalna. Jednakże prymitywny algorytm oparty na systematycznym przeglądaniu wszystkich możliwości jest praktycznie nierealizowalny, a lepsze algorytmy nie są znane. Mimo to można pisać programy heurystyczne, a jakość ich gry zależy od jakości heurystyk (Botwinnik).

Programowanie heurystyczne jest atrakcyjną metodą również z tego względu, że odzwierciedla istotną cechę myślenia ludzkiego i, być może, inteligencji w ogóle. Zatrzymamy się jeszcze przez chwilę na kwestii, czy programowanie heurystyczne jest istotnie nową metodą w stosunku do zwykłego „programowania algorytmicznego". Można bowiem przedstawić takie mniej więcej rozumowanie.

Obmyślając bądź objaśniając swój program, autor operuje pojęciem heurystyki, mówi, że program „próbuje", „stawia hipotezę" itp. Jest to oczywiście w porządku, ale są to tylko wyobrażenia autora programu, mające dla tegoż autora znaczenie heurystyczne. Ostatecznie napisany program jest jednak algorytmem, czyli przepisem skłaniającym maszynę do działania w ściśle określony sposób. Cała tajemnica leży w tym, że nie jest on algorytmem realizacji zadania, które autor miał na myśli, tylko czymś w rodzaju „gorszej wersji" takiego algorytmu. Jednakże jest to zapewne algorytm jakiegoś innego zadania i gdyby ktoś zadał sobie trud i zbadał ten algorytm dokładnie, to mógłby znaleźć szczegółowy wariant zadania, o którym dałoby się z góry udowodnić, że nie zostanie zrealizowany — należy do „wyjątków". Mamy więc do czynienia nie tyle z niedoskonałym sposobem rozwiązywania doskonałego zadania, ile z doskonałym sposobem rozwiązywania niedoskonałego zadania.

Pod pewnymi względami należy się z powyższym rozumowaniem zgodzić, z następującymi jednakże uwagami. Po pierwsze, w przedstawionej wyżej sytuacji mamy do czynienia z dwoma poziomami opisu programu: spojrzenie na program jako na algorytm odnosi się do jego mikrostruktury, podczas gdy opisanie go jako programu heurystycznego oznacza odwołanie się do pewnej jego makrostruktury.

Ta makrostruktura może pod wieloma względami okazać się dużo ważniejsza. Z podobną sytuacją często mamy do czynienia w fizyce: stan gazu w zbiorniku można opisać szczegółowo, nie zapominając o żadnej jego cząsteczce, bądź też bardziej globalnie, przy pomocy pojęć kinetycznej teorii materii. W pierwszym przypadku mamy do czynienia z opisem nadmiernie szczegółowym i przez to praktycznie nieprzydatnym. Prawie wszystkie użyteczne wyniki opierają się na opisie drugim. Analogia ta jest pouczająca jeszcze pod jednym względem: w mikroskopowym opisie molekularnym gaz jest układem deterministycznym i odwracalnym, a w opisie kinetycznym — stochastycznym i nieodwracalnym.

Oznacza to, że tak ważne cechy układu jak determinizm zależeć mogą od poziomu opisu. Odpowiednikiem fizycznego pojęcia procesu deterministycznego jest, według powyższej analogii, proces algorytmiczny; sam algorytm odpowiada przy tym deterministycznym prawom ewolucji układu.

Po drugie, fakt, że w opisie postępowania heurystycznego dopuszczalna jest kategoria wyboru oznacza, że realizujący to postępowanie proces nie musi już być procesem zalgorytmizowanym czy deterministycznym. Proces wyboru może być deterministyczny, ale może też, bez szkody dla postępowania heurystycznego, być stochastyczny.

Odpowiednio skonstruowane procedury zawierające operacje wyboru z nieskończonego zbioru możliwości wykazują następującą cechę: nie istnieje żaden szczegółowy wariant zadania ogólnego, który nie mógłby być wykonany w wyniku realizacji takiej procedury, nawet jeżeli zadanie to jest niealgorytmizowalne. Jest to więc istotnie nowa cecha w stosunku do mikro-algorytmicznych programów heurystycznych.

Czy w konkretnych przypadkach lepszy jest wybór odpowiednio umotywowany, czy przypadkowy, to jeszcze inna sprawa, dotycząca już samej techniki konstruowania procedur wyboru. Ogólnie biorąc ważne jest, że wybór może zawierać elementy stochastyczne, wskutek czego klasa maszyn nadających się do realizacji procedur wyboru jest rzeczywiście szersza niż klasa maszyn nadających się do realizacji algorytmów.