Ks. Józef Kloch
Instytut Teologiczny, Tarnów

OGRANICZENIA SZTUCZNEJ INTELIGENCJI W UJĘCIU HUBERTA L. DREYFUSA (CZ. 1)[1]

WSTĘP

Pięćdziesiąt lat po upadku mechanicyzmu w fizyce, zaczęto opracowywać modele mechanistyczne w antropologii. Stosując osiągnięcia informatyki, próbowano odnaleźć ukryte mechanizmy rządzące myśleniem człowieka. Pojęciami informacji i systemów otwartych zastępowano istotne dla mechaniki Newtona pojęcia energii oraz układu zamkniętego. „W kategoriach kodowania, szumów czy przechowywania informacji zaczęto tłumaczyć różnorodne procesy od zachowania obwodów elektrycznych do zachowania się komórek w ludzkim organizmie. W nowych interpretacjach coraz częściej zaczęły występować elementy starej filozofii mechanicyzmu".[2]

Osiągnięcia w dziedzinie informatyki zostały potraktowane jako rozwiązanie wielu niejasności — nowa dziedzina sztucznej inteligencji, przejęła funkcję pełnioną w przeszłości kolejno przez kamień filozoficzny, mechanikę klasyczną, ewolucję i walkę klas.[3]

Sztuczna inteligencja (*artificial intelligence*, skrót: AI) jest próbą symulowania ludzkiego, inteligentnego zachowania przy zastosowaniu technik programowania, które w małym stopniu wymagają upodobnienia do ludzkich procesów myślowych lub nie wymagają takiego upodobnienia.[4] Nowa dziedzina badań

---

[1] Artykuł stanowi pierwszą część pracy licencjackiej, napisanej pod kierunkiem bpa prof. dra hab. J. Życińskiego na Wydziale Filozoficznym PAT w Krakowie.

[2] J. Życiński, *Filozofia sztucznej inteligencji*, w: M. Heller, J. Życiński, *Wszechświat — maszyna czy myśl? Filozofia mechanicyzmu. Powstanie — rozwój — upadek*, Kraków: PTT 1988, s. 277.

[3] Por. J. Życiński, *W kręgu nauki i wiary*, Kalwaria: Calvarianum 1989, s. 199.

[4] Definicja według: H.L. Dreyfus, *What Computers Can't Do. The Limits of Artificial Intelligence*, New York: Harper Colophon Books ²1979, s. 85. W dalszej części pracy w odniesieniu do tej pozycji stosuję skrót: *WCCD*. Por. też określenie M.A. Boden, *Artificial Intelligence and Natural Man*, New York: Basic Books 1977, s. 3.

została zapoczątkowana przez Alana Turinga w październiku 1950 r., samo zaś określenie *artificial intelligence* pochodzi od Johna McCarthy'ego i zostało wprowadzone w 1956 r. Z czasem w AI wyodrębnione zostały cztery dziedziny:[5] rozgrywanie gier, rozwiązywanie problemów, tłumaczenia językowe oraz rozpoznawanie obrazów. Hubert Dreyfus wprowadził w swojej klasyfikacji[6] dodatkowy podział każdej dziedziny na cztery podpola: asocjacjonistyczne, sformalizowane prosto, sformalizowane w sposób złożony i nieformalizowalne.

W latach 50. oraz 60. AI odnotowała na swoim koncie wiele sukcesów; w latach siedemdziesiątych w sztucznej inteligencji zaczęto nawet doszukiwać się uniwersalnych wszechtłumaczeń, czego przykładem jest 800-stronicowa praca Douglasa Hofstadtera.[7] Podawano różne przedziały czasu, od trzech do dwunastu lat, potrzebne do zbudowania komputerów o inteligencji dorównującej a następnie przewyższającej człowieka; W.W. Bledsoe zapowiadał istnienie jedynej nauki w przyszłości — sztucznej inteligencji.

Niektórzy jednak uczeni już wówczas zaczęli podnosić głosy wzywające do większego umiarkowania w optymistycznych prognozach i do zauważania różnic zachodzących między człowiekiem a komputerem. Jednym z nich był Hubert L. Dreyfus, profesor filozofii w Berkeley. Jest on zdania, że mechanistyczne poglądy wielu informatyków mają swe źródło w niedopuszczalnym rozciąganiu na człowieka praw z pewnych dziedzin fizyki. „Analizy naszej świadomości nie przyniesie nam fizyka, można je natomiast znaleźć w fenomenologii po uwzględnieniu prac Husserla czy Merleau-Ponty'ego".[8]

Zastosowane przez Dreyfusa metody analiz w odniesieniu do AI oraz wyciągnięte przez niego wnioski stały się przedmiotem krytyki, głównie ze strony uczonych sprzyjających mechanicyzmowi. Odwoływanie się przez Dreyfusa do znaczenia obrzeży świadomości, rozróżniania elementów istotnych od nieistotnych, tolerancji wieloznaczności i wyraźnego grupowania w ludzkim przetwarzaniu informacji dało jednak wartościowe analizy. W miarę upływu czasu uczeni coraz bardziej przekonywali się do tych analiz i nawiązują do nich po dzień dzisiejszy.

We wszystkich etapach prac nad AI, niezależnie od klasyfikacji, można znaleźć deklaracje podkreślające zalety filozofii mechanicyzmu jako systemu maksymalnie prostego, w którym eliminuje się teologiczno-metafizyczne wtręty. Szybkie i spektakularne postępy w badaniach nad AI w latach 50. i 60. spowodowały próbę obrony nowej wersji mechanicyzmu — mechanicyzmu statystycznego. Jego obrońcami są głównie zwolennicy tzw. mocnego podejścia do AI zwani obliczeniowcami; również i dziś kwestionują oni zachodzenie różnic między przetwarzaniem informacji w komputerach i ludzkim mózgu.[9] Elementy mecha-

---

[5] *WCCD*, s. 85.

[6] Tamże, s. 291-305; zwięzłe przedstawienie tej klasyfikacji znajduje się w tabeli 1 na s. 292.

[7] D. Hofstadter, *Gödel, Escher, Bach. An Eternal Golden Braid*, New York: Basic Books 1979.

[8] J. Życiński, *Filozofia sztucznej inteligencji*, s. 281.

[9] J. Życiński, *Granice racjonalności. Eseje z filozofii nauki*, Warszawa: PWN 1993, s. 230-231.

nicyzmu tkwią więc nadal w dziedzinie sztucznej inteligencji, zaś Dreyfus i jemu podobni podejmowali i podejmują polemiki z jego zwolennikami.[10] Ci ostatni zdają się nie dostrzegać, że „komputery o takich samych właściwościach fizycznych mogą działać w sposób zróżnicowany w zależności od wprowadzonych do nich programów, co potwierdzałoby tezę, że zasady logiki są ontycznie pierwotne w stosunku do ich materialnych konkretyzacji. Trudno dostrzec możliwość pogodzenia podobnych faktów z podstawowymi tezami ontologii mechanicyzmu".[11]

Niniejszy artykuł dotyczy ograniczeń sztucznej inteligencji, na jakie wskazał Hubert L. Dreyfus. Ponieważ stosowany przez niego podział obejmuje cztery zasadnicze dziedziny sztucznej inteligencji, dlatego cztery rozdziały tej pracy (publikowanej w dwóch częściach) podają charakterystykę i ograniczenia, jakim podlegają poszczególne pola i podpola AI. I tak pierwsza dziedzina AI dotyczy formalizowania gier kombinatorycznych oraz badań nad uczeniem się komputerów i stosowania heurystyk w programowniu gry w szachy, a także roli jaką odgrywa obrzeże świadomości w przetwarzaniu informacji przez człowieka. Kolejne pole sztucznej inteligencji to rozwiązywanie problemów oraz związane z nim zagadnienie specyficznie ludzkiej zdolności do stukturalizowania zagadnień istotnych i nieistotnych. Druga część artykułu będzie mówić o tłumaczeniach językowych — sukcesie słownika mechanicznego i nieprzezwyciężalnych trudnościach związanych z tłumaczeniem języka naturalnego. Dreyfus odnośnie do tego pola AI podkreśla umiejętność tworzenia pojęć przez człowieka, złożoność syntaktyki oraz różnice zachodzące między uczeniem się człowieka i komputera. Ostatnia dziedzina sztucznej inteligencji to rozpoznawanie obrazów; analizy Dreyfusa pokreślają w tym wypadku ludzką zdolność do wyraźnego grupowania, której komputery są pozbawione. Ta część pracy poświęcona też będzie krytycznym ocenom analiz Dreyfusa, ewolucji jego poglądów oraz trwałym elementom w dorobku filozofa z Berkeley. Zakończenie będzie zasygnalizowaniem niektórych przemian, jakie zaszły w dziedzinie AI oraz filozofii nauki i sztucznej inteligencji na przestrzeni 10 lat — od 2. wydania *What Computers Can't Do* do ukazania się *The Emperor's New Mind* Rogera Penrose'a.

Należy mocno podkreślić, że niniejsza praca dotyczy wczesnego okresu badań nad AI — do połowy lat 70. Ponad dwadzieścia lat, jakie upłynęły od pierwszego wydania *What Computers Can't Do* (1972), przyniosły nowe analizy oraz inny podział poszczególnych pól AI.[12] Niemniej znaczna część analiz Dreyfusa stanowi trwały dorobek, a okres do połowy lat siedemdziesiątych to pierwszy, ważny etap kształtowania się filozofii sztucznej inteligencji.

---

[10] Por. J.D. Bolter, *Człowiek Turinga*, Warszawa: PIW 1990, s. 355-366, gdzie autor krótko omawia treść poszczególnych źródeł, zaznaczając, który z autorów jest zwolennikiem a który przeciwnikiem mechanicyzmu.

[11] J. Życiński, *W kręgu nauki i wiary*, s. 206.

[12] Por. np. R. Penrose, *The Emperor's New Mind*, Oxford: University Press 1989, s. 14-15.

I. GAME PLAYING — PIERWSZE POLE SZTUCZNEJ INTELIGENCJI

Programowanie gier było w latach 50. i 60. jedną z dziedzin badań nad AI. Gry łatwo poddawały się formalizowaniu; poszukiwanie rozwiązań miało różny stopień trudności i stanowiło dobry przedmiot dla analiz. Nie bez znaczenia było też uzyskanie spektakularnych efektów i zainteresowanie opinii publicznej badaniami nad AI; chodziło bowiem o uzyskanie odpowiednich funduszy na kontynuowanie prac.

Gry losowe i zręcznościowe nie zyskały sobie uznania wśród uczonych. Niewielu naukowców i stosunkowo krótko (1961-1970) zajmowało się grami karcianymi. Odnotować należy najwyżej fakt rozgłosu i efektownych wyników jakie w latach 1961-1963 zdobył matematyk z MIT prof. Edward Thorp. Jego program obliczał prawdopodobieństwo wygranej i określał optymalną strategię prowadzenia gry w kasynach.[13]

1. Prace nad formalizowaniem gier kombinatorycznych

Dreyfus w swojej podstawowej pracy *What Computers Can't Do* nic nie pisze na temat gier losowych i zręcznościowych; nie umieszcza ich również w swojej klasyfikacji. Pisze natomiast o grach pamięciowych (*memory games*),[14] które umieszcza w podpolu asocjonistycznym. Podaje też charakterystykę tego typu czynności — są one wrodzone lub wyuczone przez powtarzanie oraz oderwane od znaczenia i sytuacji. Jako przykład podaje grę „Geografia"; przypomina ona nieco znane w Polsce „Państwa, miasta". W grze „Geografia" jeden z uczestników podaje nazwę państwa, a drugi na zasadzie skojarzeń wymienia państwo na literę, która kończyła nazwę wypowiedzianą przez pierwszego uczestnika.

a. Kolejne wersje tic-tac-toe

Najwięcej uwagi naukowcy poświęcili grom kombinatorycznym. Na początku lat 50. sztandarowym osiągnięciem były programy typu „Kółko i krzyżyk". Dreyfus określa tę grę (wraz z inną: nim) jako łatwo nadającą się do sformalizowania według określonych zasad, których znaczenie jest całkowicie jasne. Można przewidzieć kolejne posunięcia i opisać ją przy pomocy algorytmu.[15]

Poszczególne kratki można ponumerować; dzięki symetryczności pola walki zamiast 9 ruchów rozpoczynających grę można analizować tylko trzy (kratkę środkową, narożną i boczną). W dalszych fazach gry można wprowadzić kolejne uproszczenia, co znacznie ogranicza liczbę kombinacji. Prace nad grą „Kółko

---

[13] O innych tego typu programach patrz M. Hołyński, *Sztuczna inteligencja*, Warszawa: Wiedza Powszechna 1979, s. 23-24.

[14] *WCCD*, s. 292-293.

[15] *WCCD*, s. 292.

i krzyżyk" rozpoczęto od jej najprostszej wersji i zrealizowano ją na urządzeniu MENACE. Z rezultatów 220 rozgrywek wynikało, że urządzenie znacznie częściej wygrywało niż jego konstruktorzy. 1962 r. przyniósł rozwiązania dla bardziej skomplikowanych wersji gry. William Daly[16] opracował program dla przestrzennej wersji w sześcianie o 64 polach (4x4x4); komputer prowadził grę na bardzo dobrym poziomie. Jeszcze bardziej skomplikowaną adaptacją był program J. Weizenbauma, R.C. Sheferdsona i D. Konivera dla nieograniczonego pokratkowanego pola, na którym wygraną stanowiło pięć jednakowych symboli w jednej linii. W tym wypadku program grał ze zmiennym szczęściem. Dalsze analizowanie „Kółka i krzyżyka" przestało być przedmiotem badań naukowców. Od początku lat 50. trwały próby nad formalizowaniem innych gier; uwaga uczonych skupiona była przede wszystkim na warcabach oraz szachach.

b. Badania A. L. Samuela nad uczeniem się programów

Prace nad programami grającymi w warcaby dały pierwsze rezultaty w roku 1952 w ośrodku doświadczalnym IBM. Wtedy to grupa badawcza pod kierunkiem dr. A.L. Samuela uruchomiła pierwszy program tego typu na komputerze IBM 701. Program ten wielokrotnie był poprawiany i w 1954 r. gotowa była kolejna wersja dla komputera IBM 704. Rok później działał pierwszy program uczący się — gromadził on w pamięci doświadczenia z danej partii i poprzednich rozgrywek. W 1962 r. miał miejsce jeden z najbardziej spektakularnych sukcesów w badaniach nad AI. Ulepszona wersja programu Samuela uruchomiona na komputerze IBM 7090 wygrała z Robertem Nealeyem, byłym warcabowym mistrzem stanu Connecticut.

Samuel opisał w artykule[17] swe kilkuletnie badania nad programem grającym w warcaby. We wstępie określa cele, jakie zamierza osiągnąć: a) szczegółowe rozwiązanie problemów związanych z uczeniem się maszyn, b) zaprogramowanie komputera tak, by uczył się na podstawie własnego doświadczenia. Wówczas istniały dwie możliwości rozwiązania problemu uczenia się maszyn: metodą sieci neuronowych oraz metodą polegającą na wytwarzaniu równoważnika wysoko zorganizowanego obwodu przeznaczonego do uczenia się określonych czynności. Samuel wybrał drugi sposób — realizowalne w ówczesnych czasach układanie nowego programu dla każdego nowego zastosowania.

Rezultatem pierwszego etapu analiz był podstawowy komputerowy program warcabowy; badał on wszystkie możliwości wykonania kolejnego ruchu na określoną głębokość. Przykładowo głębokość 2 — odpowiada jednemu ruchowi

---

[16] W. Daly, *Computer Strategies for the Game of Qubic*, Cambridge 1962.

[17] A.L. Samuel, *Some Studies in Machine Learning using the Game of Checkers*, Journal od Research and Development, 3 (1959) s. 211-229. Polski przekład: *Uczenie się maszyn na przykładzie gry w warcaby*, w: *Maszyny matematyczne i myślenie*, pod red. E. A. Feigenbauma i J. Feldmana, PWN, Warszwa 1972, s. 83-116 (w dalszej części w odniesieniu do ostatniej pozycji stosuję skrót *MMiM*).

programu i jednej odpowiedzi człowieka. Dla tej wersji programu głębokość nie miała stałej wartości (od 2 do 20) i była zależna od sytuacji na warcabnicy. Każdej z pozycji przypisywana była ocena; liniowy wielomian stanowił sposób jej uzyskania. Wybór najlepszego posunięcia był dokonywany przy pomocy metody minimaksowej.[18] Program mógł grać sam ze sobą, rozgrywał partie na podstawie podręczników (dane zapisane były na taśmie magnetycznej lub na dziurkowanych kartach), potrafił też prowadzić kilka partii równocześnie. Możliwości te okazały się cenne w chwili, gdy kolejna wersja programu porafiła uczyć się — najpierw na pamięć a następnie przez uogólnianie.

Samuel ze swoim zespołem opracował najpierw i wprowadził do programu algorytmy zapamiętywania wszystkich pozycji wraz z ocenami. Uczenie się na pamięć pozwoliło również na skrócenie czasu potrzebnego na badanie umiejscowienia i na zwiększenie głębokości analizy. Oceniając daną pozycję do głębokości np. 3, program przeglądał najpierw spis poprzednich rozwiązań. Odnaleziona, oceniona już pozycja, zastosowana przy wyborze posunięcia w danej sytuacji, daje właściwie głębokość analizy równą 6. Również to rozwiązanie zostanie zapisane w pamięci i będzie mogło być wykorzystane w przyszłości. Samuel wyposażył program w jeszcze jeden element — pojęcie kierunku, aby program zdążał ku wygranej. Mając przewagę na warcabnicy, program dążył do jak najszybszego zakończenia gry, a przegrywając, odwlekał zwycięstwo partnera, dokonując wyboru posunięcia w odpowiednim kierunku na podstawie danych z pamięci komputera o współrzędnych pozycji, głębokości oraz oceny.

Kierujący badaniami w ośrodku IBM zdawał sobie sprawę z szybko rosnącej objętości gromadzonych danych. Wpisał więc do programu algorytmy katalogowania przechowywanych informacji, wymazywania zbędnych danych oraz usuwania mało wartościowych. Katalogowanie pozwoliło zmniejszyć liczbę pozycji o połowę. Polegało ono m.in. na zmienieniu koloru bierek i ich położenia na warcabicy, tak by we wszystkich posunięciach kolejny ruch należał do czarnych. Kolejna procedura ograniczająca objętość przechowywanych danych polegała na kontrolowaniu częstotliwości wykorzystywania określonego miejsca; dodatkowy wskaźnik zwany umownie „wiekiem" świadczył o tym, czy i jak często określona pozycja była wykorzystywana w wykonaniu kolejnego ruchu. Po przekroczeniu dopuszczalnej wartości wieku, informacje były wymazywane z pamięci. Kolejna procedura opierała się na głębokości analizy. Każda lista pozycji miała ograniczoną objętość; po osiągnięciu określonej granicy, dane z najmniejszą głębokością analizy były usuwane z listy. Ta wersja programu osiągnęła sprawność lepszą od przeciętnego poziomu początkujących warcabistów.

Dalszym krokiem była metoda uczenia się przez uogólnianie gromadzonych doświadczeń. Ocenianie pozycji odbywało się za pomocą wielomianu. Samuel ułożył program, który wybierał pewien podzbiór możliwych wyrazów tego

[18] Była to procedura analizowania możliwych posunięć i oceniania, które z nich zapewnia minimum strat przy maksimum korzyści.

wielomianu oraz określał znaki i wielkości współczynników, przez które były mnożone te wyrazy. Po pierwszych nieudanych próbach program został przekształcony — zachowywał się jak dwaj gracze: pierwszy (Alfa) po każdym posunięciu uogólniał swoje doświadczenia i zmieniał współczynniki wielomianu oceniającego oraz zamieniał nieistotne wyrazy na nowe; drugi z graczy (Beta) używał tego samego oceniającego wielomianu przez całą partię. Na tej wersji nie wykonano jednak tak wielkiej liczby partii, aby można było powiedzieć, która procedura uczenia się jest całkowicie stabilna albo prowadząca z całą pewnością do wyboru najlepszego z możliwych zbioru parametrów i ich współczynników. Samuel nosił się z zamiarem połączenia obu procedur uczenia się — na pamięć oraz uogólniającej. Plany te nie zostały jednak przez niego zrealizowane.

c. Trzon badań — heurystyki

W pracach nad formalizowaniem gier kombinatorycznych prym wiodły szachy. Na podstawie programowania tej gry uczeni mieli nadzieję wniknąć w istotę intelektualnej działalności człowieka.[19] Zasady rozgrywania partii i dopuszczalne możliwości decydują o znacznym stopniu skomplikowania analiz — nie tak łatwo było formalizować szachy. Prace te powiązane były z innym nurtem badań nad AI — symulacją procesów rozwiązywania problemów (*problem solving*). Niektórzy uczeni pozostawiali nawet na kilka lat prace nad programem szachowym i poświęcili się zagadnieniu rozwiązywania problemów, by następnie zastosować te doświadczenia w programie szachowym. Koncepcje Shanonna, Turinga, grupy matematyków z Los Alamos oraz Bernsteina poprzedzają zasadnicze analizy w dziedzinie programów szachowych.

Szachy są grą o skończonej liczbie pozycji i możliwych posunięć oraz kończą się przegraną, remisem lub wygraną. Czy więc gracz może prześledzić całe drzewo gry? Zanalizowanie wszystkich możliwości, za pomocą procedury minimaksowej, wymagałoby badania 10^{120} wariantów. Stulecie zawiera mniej niż 10^{16} sekund więc analiza małej tylko części możliwości zajęłaby komputerowi całe lata.[20] Shannon zaproponował więc inne rozwiązanie — należy przeanalizować: a) wszystkie możliwe posunięcia i wszystkie warianty ale do ustalonej głębokości n, b) statycznie ocenić pozycje i zastosować metodę minimaksową dla poszczególnych posunięć, c) wybrać posunięcie z najwyższą oceną.

Metoda Shannona wymagała jednak ogromnej liczby obliczeń już dla $n=2$ czy $n=3$ i sam pomysłodawca tej metody nigdy nie uruchomił na komputerze żadnego programu grającego w szachy.

Turing w programie *Turochamp* (1951) wprowadził do metody następne ograniczenie — zmniejszył liczbę analizowanych wariantów przez zastosowanie pojęcia tzw. „martwej pozycji" czyli pozycji na tyle stabilnej, by mogła podlegać

---

[19] Por. A. Newell, J.C. Shaw, H.A. Simon, *Programy grające w szachy i problem złożoności*, w: *MMiM*, s. 50.

[20] Oszacowania według Shannona za: *MMiM*, s. 53-54.

statycznej ocenie. Dominującym czynnikiem w ocenie statycznej uczynił on przewagę materialną. Dodatkowo stosował ocenę pozycji, otrzymanych po wykonaniu posunięć, uwzględniającą możliwość wykonania następnego ruchu przez bierkę bądź jej obronę. Dla każdej możliwości przyporządkował odpowiedni współczynnik. Efekty były jednak mizerne: program miał skłonności do znaczących przeoczeń i gra nie miała określonego celu, oceny rzadko wpływały na wybór posunięcia. Program Turinga pozostał na poziomie ręcznej symulacji.

Rok 1956[21] był uwieńczeniem prac matematyków: J. Kistera, P. Steina, S. Ulama, W. Waldena i M. Wellsa;[22] zespół ten określa się zwykle grupą z Los Alamos. Ich program szachowy przeznaczony dla komputera cyfrowego MANIAC I był niemal idealną realizacją koncepcji Shannona. Uproszczone zostały jednak zasady gry: szachownica miała jedynie 36 zamiast 64 pól, w grze nie brał udziału goniec, zakazane były przesunięcia piona o dwa pola, bicia „w przelocie" i roszada. Powodem wprowadzenia ułatwień było dążenie do skrócenia czasu potrzebnego do wykonania analiz. Po tych modyfikacjach program badał wszystkie możliwe posunięcia i ich warianty na głębokość $n=2$, wybierał najlepsze posunięcie w oparciu o najwyższą ocenę na podstawie zmodyfikowanej procedury minimaksowej.[23] Zredukowanie szachownicy do 36 pól dawało 20 zamiast 30 możliwych posunięć z każdej pozycji. Badając warianty dla $n=2$, otrzymuje się więc 160.000 możliwości (20^4) zamiast 800.000 (30^4). Objętość programu wynosiła jedynie 600 słów, co znacznie przyspieszyło dokonywanie analiz. Program grupy z Los Alamos wykonywał kolejne posunięcie po analizach trwających średnio 12 minut. Odbiło się to jednak na jakości programu — oceny przed wykonaniem ruchu mają wiele braków, program robił poważne przeoczenia i był zdolny pokonać jedynie słabego gracza. MANIAC I rozegrał w sumie jedynie 3 partie.

Jako następny zabrał głos pracownik korporacji IBM Alex Bernstein — programista i szachista w jednej osobie.[24] Napisał on program na szachownicę złożoną z 64 pól i realizował koncepcję Shannona w sposób o wiele bardziej złożony. Program główny zawierał szereg podprogramów; można określić je jako generatory sensownych ruchów proponujące posunięcia do analizy. Obrona własnych figur, ochrona króla czy atak na figury przeciwnika to cechy poszczególnych generatorów. Program Bernsteina kontrolował grę na dwa posunięcia w przód a następnie badał tylko 7 możliwości dzięki pracy generatorów sensownych posunięć. Daje to tylko 2500 końcowych pozycji zamiast 800.000

---

[21] W literaturze problemu w latach 1951-1956 ma miejsce luka, stąd brak danych na temat badań w tym okresie.

[22] Por. ich artykuł: *Experiments in chess*, Journal of the Association for Computing Machinery, April (1957) s. 147-177.

[23] Modyfikacja ta dotyczyła dodatkowego elementu analizy — możliwości poruszania się bierek a nie tylko przewagi materialnej.

[24] Por. A. Bernstein [et al.], *A Chess-playing Program for the IBM 704 Computer*, w: *Proceedings of the Western Joint Computer Conference*, 1958, s. 157-159; Por. też A. Bernstein, M. de V. Roberts, *Computer vs. Chess-player*, Scientific American, June (1958) s. 96-105.

możliwości (320-krotna redukcja). Wyznacznikiem, dzięki któremu dokonywany jest wybór posunięcia, jest stosunek dwóch sum ocen rozmieszczenia figur — białych oraz czarnych; do decyzji wykorzystywana jest procedura minimaksowa. Program zawiera 7000 słów języka maszynowego, na wykonanie kolejnego posunięcia potrzebował ok. 12 minut. Trudno ocenić poziom programu, nigdy bowiem nie grał on z graczami o różnym stopniu umiejętności. W czasie gry, po kilku bardzo dobrych posunięciach, zdarzały się bardzo wyraźne przeoczenia. Wiadomo tylko o jednej partii, jaką rozegrał program. Była to gra z dobrym graczem i program Bernsteina ją przegrał.

Po zniechęceniu się Alexa Bernsteina do dalszych analiz, od 1958 r. ton pracom nad programami szachowymi nadawała trójka naukowców:[25] A. Newell, J.C. Shaw oraz H.A. Simon. Ich program NSS, z szeregiem podprogramów, przyciągnął uwagę psychologów, specjalistów teorii systemów oraz szachistów. Również Dreyfus poświęca wiele miejsca analizom tego programu. Prace nad programem rozpoczęły się w 1955 r. potem jednak miała miejsce przerwa trwająca do 1958 r., spowodowana przygotowywaniem programów udowadniających twierdzenia logiki symbolicznej. Jak twierdzili Newell, Shaw i Simon „w zasadzie dowodzenie twierdzeń oraz gra w szachy stanowią jeden i ten sam problem: poszukiwanie heurystyk, które pozwoliłyby wybrać najbardziej obiecujące drogi w morzu możliwości, których liczba rośnie wykładniczo. Zarówno dowodzenie twierdzeń jak i gra w szachy zawierają te same dylematy: szybkość czy selektywność wyboru, sztywne działanie programu czy zależne od sytuacji".[26]

Złożoność działania jest zasadniczą cechą inteligentnych czynności. Trzeba więc odkryć heurystyki, przy pomocy których człowiek rozwiązuje problemy związane z grą w szachy i przekazać je programowi. Przeprowadzane przez nich analizy partii szachowych miały za zadanie gromadzenie informacji i tworzenie pewnych celów w grze typu: „atak na pozycję króla", „opanowanie centrum", „równowaga materialna" itp. Uczeni ci wyrażali nadzieję, że te analizy doprowadzą ich do wielkiego stopnia złożoności programu komputerowego grającego w szachy. Złożoność ta będzie bliska złożoności myślenia dobrego szachisty.

Twórcy programu szachowego NSS dostrzegali też inny problem związany ze złożonością — jest nią komunikatywność, porozumiewanie się z maszyną. Nie mieli oni zamiaru budować specjalistycznego elektronicznego urządzenia grającego w szachy. Chcieli zrealizować swe pomysły dla dowolnego komputera cyfrowego. Program musiał być więc napisany w zrozumiałym dla urządzenia języku maszynowym. Newell, Shaw i Simon przewidywali dobrą przyszłość dla takiego

---

[25] Por. A. Newell, J.C. Shaw, H. Simon, *Chess Playing Programs and the Problem of Complexity*, Journal od Research and Development, 2 (1958) s. 320-335. Polski przekład: *Programy grające w szachy i problem złożoności*, w: *MMiM*, s. 49-79. W dodatku (*MMiM*, s. 79-82) znajduje się opis przebiegu partii rozegranej przez H.A. Simona i Program Szachowy NSS.

[26] *MMiM*, s. 62.

rozwiązania. W ich analizach można zauważyć nadzieję na jeszcze bogatsze języki i lepsze komunikowanie się z coraz bardziej inteligentnym komputerem.[27]

Newell, Shaw i Simon wykorzystali doświadczenie Shannona (podstawowy schemat analizy), Turinga (martwa pozycja), oraz Bernsteina (generatory posunięć związane z określonym celem). Zajęli się jednak przede wszystkim opisaniem i zrozumieniem procesów myślenia i podejmowania decyzji zachodzących w ludzkim mózgu.[28] W ocenie, decydującej o kolejnym ruchu, zrezygnowali z addytywnej liczbowej funkcji.

Program NSS działa zgodnie z ustalonym zbiorem celów; są nimi: opanowanie centrum, równowaga materialna, atak na pozycję króla itp. Każdy cel związany jest z kilkoma podprogramami: a) z podprogramem podającym opis celu (zależnie od sytuacji na szachownicy), b) z generatorem posunięć, znajdującym korzystne posunięcia (z punktu widzenia danego celu), c) z podprogramem oceny statycznej dla każdej pozycji względem określonego celu, d) z generatorem szukającym posunięć prowadzących do martwej pozycji. Generator posunięć w porównaniu z rozwiązaniem bernsteinowskim ma istotną nowość — jest on odpowiedzialny jedynie za tworzenie całej gamy propozycji kolejnego ruchu nie zaś na generowanie kontynuacji. Tworzenie propozycji odbywa się w porządku zależnym od priorytetów przyporządkowanych im celów. Kolejny podprogram przyporządkowywał oceny każdemu posunięciu w odniesieniu do danego celu. Ocenę tę można uważać za wektor; zbiór jego składowych może zmieniać się w czasie partii, bowiem składowe wybierane są z podstawowej listy celów tworzonej na podstawie początkowej pozycji. W skład analiz (generatora d) wchodzą: badanie kontynuacji do pewnej głębokości, tworzenie ocen statycznych i ich integracja oraz efektywna ocena danego posunięcia; analizy te przebiegają podobnie jak w innych programach szachowych.

Każdy z wymienionych wyżej podprogramów działał niezależnie od pozostałych; można więc było usuwać je z programu głównego lub dodawać nowe. W tych podprogramach zawarte było jeszcze inne dążenie — chęć nadania programowi elastyczności w trakcie partii; zakres obliczeń miał zależeć od sytuacji na szachownicy.

Po podprogramach związanych z określonym celem rozpoczynają swą pracę podprogramy wspólne dla całego programu: integracja ocen statycznych w ocenę efektywną, dla mającego nastąpić posunięcia, oraz procedura końcowego wyboru. Metodą integracji ocen jest procedura minimaksowa; założenie działania tej procedury jest jednak inne niż w programach poprzedzających NSS, w których

---

[27] Newell, Shaw, Simon, jw. s. 51-52.

[28] Por. A. Newell, J.C. Shaw, H.A. Simon, *Elements of a Theory of Human Problem Solving*, Psychological Review, March (1958) s. 151-166 oraz A. Newell, J.C. Shaw, H.A. Simon, *The Process of Creative Thinking*, Rand Corporation Paper P-1320, [Santa Monica, California] 1958.

wybierane było posunięcie z najwyższą oceną. Newell, Shaw i Simon przyjęli pewien poziom akceptowalności — wybierane było jako pierwsze to posunięcie, którego ocena przekroczyła ów poziom. Posunięcia do zaakceptowania przedstawiane były w kolejności zgodnej z hierarchią celów (równowaga materialna, opanowanie centrum, etc.). Jeśli ani jedna ocena nie osiągnie poziomu akceptowalności — wykonany zostanie ruch z najwyższą oceną, NSS zapamiętywał bowiem najlepsze z wygenerowanych posunięć. W ten sposób dokonany zostaje końcowy wybór odnośnie do kolejnego posunięcia w danej partii.

Warto zwrócić uwagę na fakt, że w chwili publikowania artykułu program NSS był dopiero pisany[29] i uruchamiany częściami. Autorzy programu chcieli uzupełnić go o kilka innych celów: „zabezpieczenie Króla", „poważne groźby", oraz „gambity" Newell, Shaw i Simon opracowali specjalne języki w czasie prac nad programami dowodzącymi twierdzenia; zostały one nazwane IPL (Information Processing Languages) i miały znacznie większe możliwości od języka maszynowego. W programie szachowym NSS został wykorzystany jeden z nich (IPL-IV) wraz z językiem wewnętrznym komputera JOHANNIAC. Czas potrzebny na obliczenie kolejnego ruchu w partii wynosił od 1 do 10 godzin; przy rozpoczęciu gry obliczanie było znacznie krótsze (kilka minut). Autorzy NSS chcieli za wszelką cenę odwzorować w programie reguły stosowane przez szachistów, dlatego zastosowali tak bardzo specyficzne heurystyki. Korzystanie w pracach nad AI z obserwacji poczynionych w czasie gry szachistów polecał również w 1963 r. de Groot w wykładzie na XVII Międzynarodowym Kongresie Psychologicznym. W tym samym roku światowej sławy mistrz szachowy Michał Botwinnik przełamał swe wewnętrzne opory i włączył się w Związku Radzieckim do prac nad programem dla komputera BESM a następnie dla M-220.[30] W 1967 r. I.J. Good, matematyk i szachista z Oxfordu, zaproponował nawet objęcie badań jednolitym planem pięcioletnim. W tym samym roku miał miejsce pojedynek programów szachowych USA — ZSRR a R.D. Grenblatt ukończył swój program, który został wysoko oceniony.

Koniec lat 60. zaowocował więc wieloma ciekawymi programami i opracowaniami. Podobnie było z początkiem lat 70. Zespół radziecki G.M. Adelsona-Wielskiego poszukiwał metod eliminujących posunięcia nie mające znaczenia dla przebiegu gry i przekształcających algorytmy wyboru. Efektem tych prac był program „Kaissa", który wygrał w komputerowym turnieju szachowym (Sztokholm, 1974). W tym samym roku ukończono prace nad programem Mac Hack a w dwa lata później gotowy był Chess 4.6 opracowany w Northwestern

---

[29] Program miał wówczas objętość 6000 słów a autorzy spodziewali się, że rozrośnie się do 16000 słów.

[30] Interesujące wyniki prac opublikował M. Botwinnik, *Algoritm igry w szachmaty*, Moskwa: Nauka 1968. Ukazał się również angielski przekład tej książki: *Computers, Chess and Long Range Planning*, Berlin: Springer Verlag, 1970.

University. Z okazji kolejnego kongresu IFIP (International Federation for Information Processing) miał miejsce II komputerowy turniej szachowy w Kanadzie.

2. Uwagi Dreyfusa na temat barier pierwszego pola AI

Osiągnięcia z początków lat 50. w dziedzinie programowania gry „Kółko i krzyżyk" nastawiały uczonych bardzo optymistyczne. Udało się całkowicie sformalizować tę grę i programy z wielkim powodzeniem ją symulowały. Dreyfus zgadza się z ogólną opinią, że gry tego typu mogą działać na komputerach przy zastosowaniu współczesnych technik programowania. Algorytm można tak układać, by komputer wygrywał lub osiągał remis.[31] Kolejne wersje programów były coraz bardziej skomplikowane; ich ukoronowaniem był algorytm Wiliama Daly dla przestrzennej wersji „Tic-tac-toe" oraz opracowanie zespołu J. Weizenbauma sporządzone dla nieograniczonego pokratkowanego pola.

Optymistyczne nastawienie podtrzymywane było dzięki pracom grupy kierowanej przez A.L. Samuela i słynnej przegranej warcabowego mistrza Roberta W. Nealeya w 1962 r. Samuel wyrażał odmienne poglądy na temat programowania gier niż twórcy programu NSS. Nie wykorzystywał on w swych pracach metod rozwiązywania problemów stosowanych przez ludzi, nie naśladował też i nie adaptował procesów zachodzących w ludzkim umyśle. Zainteresowanie uczonych wzbudziły jego próby ulepszania poziomu gry przez stosowanie podprogramów wyposażonych w algorytmy uczenia się programu. Dreyfus zauważa, że w odniesieniu do warcab istnieją sposoby określenia prawdopodobnej oceny posunięcia. Trzeba wziąć pod uwagę pewne priorytety jak np. kontrolowanie centrum warcabnicy. Zasady gry umożliwiają „zbadanie wszystkich rozsądnych posunięć na głębokość do dwudziestu naprzód, co okazuje się wystarczające do prowadzenia doskonałej gry".[32]

Dreyfus wskazuje tu na pierwsze ograniczenia — nie jest możliwe przeanalizowanie wszystkich posunięć (10^{40}) do końca partii; badanie ruchów należy ograniczyć. Przy głębokości $n=20$ możliwe jest jeszcze roztrząsanie wszystkich posunięć i wybór najlepszego ruchu; poziom programu będzie mistrzowski. Dreyfus nie wymienia warcab wśród przykładów należących do pola II (Simple-Formal);[33] jednakże zapewne tę grę ma na myśli, pisząc w tabeli Simple-Formal o grach niemal możliwych do obliczenia.[34]

Najwięcej miejsca poświęca Dreyfus dociekaniom na temat programowania szachów.[35] Dla szachów zanalizowanie wszystkich możliwości posunięć oznacza-

---

[31] *WCCD*, s. 100-101.
[32] *WCCD*, s. 101.
[33] *WCCD*, s. 292.
[34] *WCCD*, s. 292.
[35] *WCCD*, s. 101-107.

łoby badanie 10^{120} wariantów, co jest niemożliwe w rozsądnym czasie. Uczeni różnie próbowali rozwiązać problem wykładniczo rosnących możliwości posunięć. Pierwszym pomysłem był tzw. „the random element": maszyna miała od czasu do czasu szukać kombinacji prowadzącej do poświęcenia królówki, by uzyskać przewagę. Jednak sam Newell, pomysłodawca tego rozwiązania, ocenił je jako niewystarczające. Program powinien badać sytuacje, w których takie poświęcenie byłoby znaczące, a nie tylko co jakiś czas odnajdywać możliwość takiego poświęcenia. Kolejnym pomysłem było więc ograniczanie liczby ruchów przeznaczonych do analizy, a przy tym określanie posunięć najbardziej obiecujących. Zwłaszcza Newell, Shaw i Simon mieli nadzieje na odnalezienie heurystyk tego typu. Ale Dreyfus zauważa, że nie znaleziono takich heurystyk, które umożliwiałyby prowadzenie partii na mistrzowskim poziomie i poddaje w wątpliwość dowody na to, że gra w szachy jest rządzona stosowaniem heurystyk. Warto przytoczyć tu cytowany przez Dreyfusa fragment sprawozdania Simona: „Nagle zauważam, że jedna z jego figur — wieża — nie jest broniona, i że muszą być sposoby, by na tym skorzystać. Załóżmy, że wystawię pionka pod bicie gońca, jeśli goniec się wycofa, to mogę zaszachować królówkę i zabić wieżę. Jeśli itd., itd."[36]

Istotnym jest tu sformułowanie: „Nagle zauważam..."; w jaki sposób grający doszedł do wniosku, że wieża partnera nie jest broniona? Czy postępował tak jak komputerowy program i analizował możliwości kolejnego posunięcia? Dreyfus stara się wykazać różnice między człowiekiem a programem komputerowym w sposobie wybierania posunięcia. W tym samym czasie (15 min.) program komputerowy zanalizował 26000 możliwości a gracz od 100 do 200. Jednakże posunięcie człowieka może okazać się bardziej błyskotliwe od komputerowego, choć ten pierwszy dokonał znacznie mniej analiz. Co o tym decyduje? Dreyfus odwołuje się tutaj do koncepcji obrzeży świadomości (*the fringes of consciousness*, określenie Williama Jamesa). Umysł ludzki ocenia globalnie sytuację na szachownicy. „Sygnały z całej szachownicy pozostając na obrzeżach świadomości przyciągają uwagę do pewnych sektorów i powodują, że wyglądają one obiecująco, niebezpiecznie lub po prostu, że warto im się przyjrzeć".[37]

Sytuacja ta przypomina atraktory nieliniowego modelu ewolucji nauki.[38] Po ruchu partnera zmiejącym w znaczący sposób (bifurkacja) sytuację na szachownicy powstaje nowy układ. W trakcie namierzania się gracza, niektóre obszary szachownicy (atraktory) przyciągają jego uwagę.

Ciekawa jest metamorfoza poglądów Newella i Simona odnośnie do prac nad programami szachowymi. Ich artykuł *Program grający w szachy i problem*

---

[36] Cytat za *WCCD*, s. 102.

[37] *WCCD*, s. 104. Dreyfus powołuje się tutaj na koncepcję M. Polanyi o tendencji obrzeży świadomości do koncentrowania informacji dotyczących ludzkich doświadczeń zewnętrznych; tendencja ta rozciąga się w sposób nieokreślony dookoła głównego obiektu uwagi podmiotu.

[38] Por. M. Heller, *Filozofia nauki. Wprowadzenie*, Kraków PAT 1992, s. 65-72.

złożoności napisany w 1958 r. wyrażał nadzieje związane z programami heurystycznymi: „Widoczna jest (...) stała tendencja do konstruowania coraz to bardziej skomplikowanych programów i stosowania coraz bardziej specyficznych heurystyk, a także do kierowania się w grze regułami podobnymi do tych, które stosują szachiści".[39]

Wówczas program NSS był dopiero uruchamiany fragmentami, niektóre pomysły istniały jedynie w formie koncepcji. Newell i Simon sześć lat później wykazują o wiele mniej optymizmu; byli wtedy bogatsi o doświadczenia porażek programu NSS w grze z człowiekiem. Zauważają,[40] że myślenie człowieka w czasie pojedynku szachowego jest o wiele bardziej globalne niż ich algorytmy (np. „opanowanie centrum", „równowaga materialna", „zabezpieczenie króla" itp.) wbudowane do NSS w formie podprogramów. Dają oni do zrozumienia, że nie widzą sposobu na przeanalizowanie ocen pozycji na szachownicy za pomocą metod heurystycznych. To wskazywałoby na zakończenie poszukiwań co do symulacji ludzkich zachowań na drodze skomplikowanych heurystyk. Ale ich opinia nie jest jednak jednoznaczna; wypowiedzi Newella i Simona wydają się sugerować, że badania dotyczące tego zagadnienia nie są jeszcze na dostatecznie wysokim poziomie. Dreyfus krytykuje ich optymistyczną wiarę w możliwość opracowania mechanicznego mistrza szachowego. Wskazuje na wyniki prac de Groota i podkreśla, że powinny one skłonić Newella i Simona do pesymizmu, a tymczasem ich postawa jest wręcz odwrotna (sic!). De Groot w trakcie badań procesem percepcji stwierdził, że spostrzegawczość i struktura gry są ważnymi czynnikami w rozgrywaniu partii a przysługują one wyłącznie człowiekowi. Znakomity szachista Hearst w swych analizach podkreśla zdolności człowieka do postrzegania sytuacji na szachownicy w dużych strukturach współpracujących ze sobą a nie bezładnie porozrzucanych figur; podkreśla w ten sposób, dlaczego gra w szachy nie poddaje się zaprogramowaniu.[41]

Nawiązując do sprawozdania Simona, Dreyfus[42] wylicza elementy gry w szachy przysługujące wyłącznie człowiekowi: rozpoznawanie miejsc słabych i mocnych na szachownicy oraz specyficznych pozycji, podatność na atak w określonych polach szachownicy, znajomość poprzednich ruchów w rozgrywce w połączeniu z ogólnymi wzorami posunięć w szachach oraz zdolność do tzw. namierzania się na dany obszar. Elementy te stanowią jednocześnie bariery nie do przekroczenia dla prac nad programami komputerowymi grającymi w szachy. Autor *What Computers Can't Do* podkreśla, że nie ma szachowego programu choćby próbującego wykorzystywać doświadczenia z wcześniejszych posunięć w danej partii. Zaznacza z naciskiem, że każdy ruch jest traktowany jak

---

[39] Newell, Shaw, Simon, *Elements*, s. 77.

[40] A. Newell, H.A. Simon, *An Exemple Chess Play in the Light of Chess Playing Programs*, Carnegie Institute of Technology, August (1964) s. 10nn. Artykuł ten analizuje Dreyfus, por. *WCCD*, s. 104-105.

[41] Opinie de Groota i Hearsta za *WCCD*, s. 104.

[42] Por. *WCCD*, s. 102-106.

wyizolowany problem szachowy odnaleziony w książce. Nawet gdyby istniał program gromadzący informacje o pozycji każdej z figur, szybko ugrzązłby we własnych danych. Istnieje potrzeba określania przez program szachowy namierzania się na określone części szachownicy, na podstawie zbieranych doświadczeń oraz aktualnego stanu gry; namierzanie takie nie może mieć miejsca w wypadku analizowania krok po kroku, a tak właśnie działają programy komputerowe. Informacje, pozostając na obrzeżach świadomości, przetwarzane są globalnie. Dreyfus wyraźnie sprzeciwia się dwuetapowej koncepcji przetwarzania informacji: początkowo szybkiej i nieświadomej przy pomocy błyskotliwych heurystyk, a następnie wolnego i świadomego badania możliwości kolejnych posunięć.

Podsumowując swoje rozważania na podstawie protokołów z rozgrywek szachowych Dreyfus sugeruje dwa rodzaje zachowań: „(1) namierzanie się *(zeroing in)* przy pomocy globalnej organizacji pola percepcji na obszar znajdujący się poprzednio na obrzeżach świadomości, które to [namierzanie się] czyni interesującymi inne obszary (...) oraz (2) rozpatrywanie (*couting out*) widocznych alternatyw".[43]

3. Podsumowanie

Badania nad sztuczną inteligencją mają szereg dziedzin. Jedną z najwcześniej formalizowanych były gry. Wśród nich największym powodzeniem cieszyły się gry kombinatoryczne — początkowo „Kółko i krzyżyk" (Tic-tac-toe) a następnie warcaby oraz szachy.

W pierwszej fazie analiz miały miejsce spore sukcesy — komputerowy program „Tic-tac-toe" w skomplikowanych wersjach osiągał mistrzowski poziom gry; wielkim osiągnięciem były badania A.L. Samuela poświęcone zagadnieniu uczenia się maszyn na przykładzie gry w warcaby. Sensacją była wygrana jego programu nad eks-mistrzem warcabowym stanu Connecticut. Ale sukcesy te były osiągane dzięki temu, że prace były prowadzone nad takimi grami, których formalizowanie było stosunkowo proste. Algorytmy lub heurystycznie przeprowadzane kalkulacje doprowadzały do całkowitego rozwiązania problemu lub przynajmniej umożliwiały prowadzenie gry na bardzo dobrym poziomie.

Porażki nastąpiły w chwili, gdy rozpoczęły się prace nad bardziej złożonymi grami; widać to szczególnie wyraźnie w wypadku szachów. Człowiek dzięki właściwościom swojego myślenia unika wykładniczego wzrostu możliwości. Jego globalne koncepcje prowadzenia gry, odnajdowanie słabych i mocnych obszarów na szachownicy, spostrzegawczość, tworzenie całościowych powiązań między figurami leżą poza zasięgiem komputerowych programów. Poszukiwania heurystyk odwzorowujących myślenie człowieka podczas gry nie dały zadowalających rezultatów. Po początkowych spektakularnych sukcesach nastąpiła porażka.

Dreyfus wspomina w *What Computers Can't Do* o jeszcze jednej bardziej skomplikowanej grze niż szachy. Jest nią wywodząca się z Chin gra kom-

[43] *WCCD*, s. 106.

binatoryczna „Go".[44] O ile szachy mają ok. 10^{120} możliwości kombinacji, „Go" przy stosunkowo prostych zasadach ma ich ok. 10^{761}. W grze bierze udział dwóch uczestników, plansza ma 19 przecinających się pionowych i poziomych linii. Grający stawiają na przecięciach kamienne krążki; robią to na przemian — jeden z nich gra białymi a drugi czarnymi. Gracze mają zniszczyć obozy przeciwnika lub otoczyć własnymi pionami jak największy obszar. Formalizowaniem „Go" zajmowali się m.in. E. Thorp i W.E. Walden, teoretyczne podstawy opracował E. Lasker. Pewne osiągnięcia miał na tym polu W. Dobosiewicz, matematyk z Uniwersytetu Warszawskiego, którego program pokonał w rozgrywce algorytm J. L. Rydera. Dreyfus zalicza „Go" do tej samej grupy, co szachy: do gier sformalizowanych w sposób złożony. Próby sformalizowania „Go" trwały aż do 1974 r. i zakończyły się porażką. Autor *What Computers Can't Do* pisze również o zbiorze gier,[45] które w ogóle nie podlegają formalizowaniu; dzieje się tak, gdy są one źle zdefiniowane, np. zagadki zależne od znaczenia i sytuacji, która nie jest zadana *explicite*. Dla tego typu gier Dreyfus nie widzi żadnych przykładów programów komputerowych.

II. ROZWIĄZYWANIE PROBLEMÓW — DRUGIE POLE SZTUCZNEJ INTELIGENCJI

Konkretne zagadnienie warcabowe czy szachowe stawia przed człowiekiem lub komputerem problem rozwiązania określonego zadania. Stąd też programowanie gier było dobrym polem doświadczalnym dla początkowych prac nad sztuczną inteligencją. Inną dziedziną badań nad AI było symulowanie procesu rozwiązywania ogólnych kwestii.

Pojawienie się komputera było nowym impulsem do badań nad analizą rozwiązywania problemów. Wśród zagadnień tego typu dowodzenie skomplikowanych twierdzeń matematycznych stanowi grupę złożonych zadań. Uczeni zajmujący się komputerowym dowodzeniem mieli nadzieję na dokładne zrozumienie przebiegu przetwarzania informacji przy odkryciach matematycznych. Zrozumienie tych procesów uznawali za ważny etap; kolejnym krokiem miało być programowanie jeszcze bardziej ogólnych metod służących do rozwikłania różnych zadań umysłowych.

Podobnie jak w przypadku gier, uczonych przyciągała prostota formalnego systemu; dotyczy to choćby logiki matematycznej czy geometrii euklidesowej. W czasie badań nad rozwiązywaniem problemów ułożone zostały nowe języki komputerowe — IPL do przetwarzania zestawów symboli (Newell, Shaw i Simon) oraz FLPL do przetwarzania danych (Gelernter i Hansen).

---

[44] Por. H o ł y ń s k i, jw. s. 44-49.

[45] *WCCD*, s. 292.

Budując teorię, która miała wyjaśniać, jakimi metodami ludzie próbują rozwiązywać zagadnienia, badacze nawiązywali do asocjacjonizmu, behawioryzmu i gestaltyzmu. Połączeniem idei M. Weltheimera i G. Polya była koncepcja Minsky'ego. Miał on nadzieję, że zastosowanie jej w badaniach nad sztuczną inteligencją dobrze odwzoruje postępowanie człowieka; ówczesne próby (np. Newella i Simona) w tym względzie zakończyły się niepowodzeniem. Złożone przetwarzanie informacji w umyśle człowieka rozwiązującego problem nie zostało przełożone na język komputerowy.

1. Zagadnienie rozwiązywania problemów a programowanie

Podobnie jak w wypadku prac nad formalizowaniem gier, zagadnienie rozwiązywania problemów w latach 50. i na początku lat 60. zostało w znacznym stopniu zdominowane przez badania A. Newella, J.C. Shawa i H.A. Simona. Ich programy Logic Theory Machine (LT, 1957) a zwłaszcza General Problem Solver (GPS, 1959), są szeroko komentowane przez Dreyfusa. Interesujące były również studia zespołu H. Gelerntera nad Geometry Theorem Machine (GTM, 1959); o tym programie również pisze autor *What Computers Can't Do".*

Program Graph Traverser, choć został napisany w 1968 r., zostanie omówiony jako pierwszy, bowiem według klasyfikacji Dreyfusa należy do pierwszego podpola — asocjonistycznego. Następnie przedstawione zostaną prace nad GTM (podpole II: Simple-Formal), LT (podpole II oraz III w zależności od etapu prac) oraz nad GPS (podpole III: Complex-Formal).

a. Problemy-labirynty

Graph Traverser należy do tej grupy programów, którą Dreyfus określa jako asocjonistyczne.[46] Problemy nie odbiegają tu zbytnio od zadań występujących przy grach typu tic-tac-toe; należy określić drzewo decyzji, by rozwiązywać owe problemy-labirynty (*maze problems*), jak je określa Dreyfus. Złożoność zagadnienia nie jest zbyt duża i nie wykracza poza przeciętne ludzkie umiejętności. Właściwym sposobem rozwikłania problemu jest metoda prób i błędów — analiza kolejnych możliwości, których nie jest zbyt dużo, wcześniej czy później doprowadzi do poprawnego rozwiązania.

Program[47] Graph Traverser został ułożony w 1968 roku na uniwersytecie w Edynburgu a jego autorami są J.E. Doran oraz D. Michie. Problem do rozwiązania jest interpretowany jako zbiór stanów i reguł przechodzenia od jednego stanu do drugiego. Zadaniem programu jest przekształcanie kolejnych

---

[46] *WCCD*, s. 292.

[47] J.E. Doran, *New Developments of the Graph Traverser*, w: *Machine Intelligence 2*, ed. E. Dale, D. Michie, Edinburgh: Oliver and Boyd 1968, s. 119-135; por. też D. Michie, R. Ross, *Experiments with the Adaptive Graph Traverser*, w: *Machine Intelligence 5*, ed. B. Malzer, D. Michie, Edinburgh: University Press 1970, s. 301-318.

stanów oraz sprawdzanie, czy wśród nowych znajduje się taki, który może być uznany za rozwiązanie. Schemat działania Graph Traversera można opisać w następujący sposób: po uruchomieniu zachodzi przekształcanie, jeśli nowy stan jest rozwiązaniem — program kończy swoje działanie, jeśli natomiast nie ma rozwiązania — wykonywane jest kolejne przekształcenie.

Program dobrze radził sobie z rozwiązywaniem wielu praktycznych zadań — wyszukiwał najkrótszą drogę przy podróży przez określone miasta (tzw. problem komiwojażera), ustalał miejsca postoju ciężarówek w garażu z uwzględnieniem kolejności ich przyjazdów i wyjazdów bądź układał w kolejności ponumerowane, przesuwalne pola łamigłówek (osiem lub piętnaście pól).

b. Rozwiązywanie zadań geometrycznych

W 1954 r. prof. G. Polya wydał rozprawę poświęconą zagadnieniom heurystyki i odkryć matematycznych. Zrealizowanie jego idei w dziedzinie AI dałoby urządzenie rozwiązujące problemy matematyczne; nie było to jednak wówczas możliwe ze względu na niski stopień zaawansowania prac związanych z programowaniem komputerów w języku maszynowym. Stąd H. Gelernter ograniczył zakres prac[48] i wyznaczył sobie zadanie ułożenia programu dowodzącego jedynie twierdzenia geometrii euklidesowej; w ówczesnych czasach (1959) problem ten był rozwiązywalny.

Gelernter wychodził z następujących założeń: problemem jest wyrażenie lub ciąg napisane w jakimś formalnym systemie logicznym. Pierwszy ciąg stanowić będzie aksjomat lub uprzednio udowodnione twierdzenie; pozostałe ciągi będzie można wywnioskować ze zbioru poprzedzającego go albo sam będzie aksjomatem lub udowodnionym uprzednio twierdzeniem. Ostatnim ciągiem rozwiązania będzie postawiony problem. Program ma za zadanie wybrać takie twierdzenia i aksjomaty z danego zbioru, które będą podstawą dowodu oraz służyć będą do wygenerowania innych ciągów potrzebnych do dowodzenia. Dowodzenie twierdzenia geometrycznego zostanie zakończone w chwili znalezienia sekwencji wyrażeń będących dowodem.

Zagadnienie wydaje się więc pozornie bardzo proste. Jednak analizowanie wszystkich możliwych sekwencji, aż do natrafienia na dowód, generuje ogromną liczbę tychże,[49] podobnie jak w przypadku możliwych posunięć w szachach czy w warcabach. Geometry Theorem Machine (GTM) opierała się na metodzie analitycznej osiągania celu — pracowała wstecz; nie dawało to jednak gwarancji, że każda wygenerowana sekwencja była dowodem czegokolwiek i rzeczywiście

---

[48] Por. H. Gelernter, *Realizacja maszyny dowodzącej twierdzeń geometrycznych*, w: *MMiM*, s. 145-163 oraz H. Gelernter, J.R. Hansen, D.W. Leveland, *Empiryczne badania maszyny dowodzącej twierdzenia geometryczne*, w: *MMiM*, s. 163-174.

[49] Gelernter podaje przykład — na istniejących wówczas maszynach cyfrowych udowodnienie zwykłego dziesięciokrokowego twierdzenia geometrycznego na drodze generowania sekwencji dowodowych zajęłoby czas rzędu tysiecy lat; patrz: H. Gelernter, jw. s. 147.

większość wygenerownych w ten sposób ciągów była fałszywa. Należało więc znaleźć sposób wykazywania fałszywości poszczególnych sekwencji i odrzucania ich; znacznie ograniczyłoby to zakres analizowanych możliwości, a zwiększyłoby prawdopodobieństwo znalezienia poprawnego dowodu. GTM wykorzystywała tu własności heurystyczne diagramu odróżniającego sekwencje prawdziwe od fałszywych. Heurystyki te musiały być starannie dobrane; miały bowiem za zadanie zwiększyć efektywność rozwiązywania problemu przez ograniczanie liczby rozwiązań ale jednocześnie nie mogły dopuścić do usunięcia interesującego rozwiązania.

W programie GMT wyróżnić można 3 części: syntaktyczną — manipulującą formalnym systemem, diagramową — zawierającą dowodzone twierdzenie i szereg programów opisujących diagram oraz część heurystyczną. Ta ostatnia porównuje wygenerowane ciągi z ich interpretacją na diagramie i odrzuca sekwencje nie potwierdzone przez model oraz wykonuje kilka innych zadań: organizuje proces szukania dowodu i rozpoznaje syntaktyczną symetrię pewnych klas ciągów. Część heurystyczna programu w dużym stopniu decyduje o właściwościach GMT i modyfikacja tej części programu wpływa na poziom działania całej Geometry Theorem Machine.

Gelernter wykonał szereg prób[50] dowodzenia twierdzeń o różnym stopniu trudności. Zadania na poziomie szkoły średniej były wykonywane w czasie od kilkudziesięciu sekund do kilku minut. Można było zauważyć wykładnicze powiększanie się grafu rozwiązującego problem w miarę wzrostu stopnia trudności dowodu. Wprowadzone heurystyki dawały różne wyniki: w jednym wypadku skracały czas dowodzenia, w innym nie dawały tego efektu. Rozwiązywanie specjalnego przypadku zadania (z końcowego egzaminu z brooklyńskiej szkoły średniej) przy pomocy podstawowej heurystyki przepełniło pamięć roboczą komputera po pół godzinie; zastosowanie rozszerzonej heurystyki pozwoliło na uporanie się z tym zadaniem w mniej niż pięć minut.

Struktura programu GTM była statyczna; nowe, rozwiązane zadania nie wpływały na udoskonalenie jego działania i na poszerzenie zakresu rozwiązywanych zagadnień. GTM z góry był ograniczony przez Gelerntera do rozwiązywania tylko pewnej klasy problemów; to ograniczenie dało pozytywne rezultaty. Uczony wiązał osobne nadzieje z uczeniem się programu. Nadzieje te pozostały jednak tylko w sferze niezrealizowanych teorii.

Geometry Theorem Machine stanowi przykład programu zaliczanego przez Dreyfusa do podpola Simple-Formal dającego najlepsze rezulataty badań nad sztuczną inteligencją.[51] Algorytm poszukujący rozwiązania wspierany był przez heurystykę ograniczającą liczbę rozwiązań. Program GMT, wraz z warcabowym programem Samuela, wymieniany jest jako jeden z największych sukcesów AI z przełomu lat 50. i 60.

---

[50] Por. tamże s. 154-156.

[51] *WCCD*, 96, s. 292-293.

c. Udowadnianie twierdzeń logicznych

Program Logic Theory Machine (1957) udowadniający twierdzenia logiki matematycznej był częścią obszernego planu badań nad złożonymi systemami przetwarzania informacji. Newell, Shaw i Simon[52] wiązali wielkie nadzieje z programem LT; jego pełna wersja miała służyć do udowadniania twierdzeń matematycznych, odkrywania praw naukowych, prowadzenia rozgrywek szachowych a nawet rozumienia prozy angielskiej. Uczeni wyznaczyli sobie zadanie zrozumienia złożonych procesów zachodzących w ludzkim umyśle podczas rozwiązywania problemów.

Wstępna wersja Logic Theory Machine udowadniała twierdzenia elementarnej logiki formalnej i napisana była dla komputera JOHNNIAC. System aksjomatów, definicji i twierdzeń zaczerpięty został z *Principia mathematica*"; aksjomaty i twierdzenia ponumerowane zostały tak samo, jak zrobili to A. N. Whitehead i B. Russell.

Newell, Shaw, Simon stosowali głównie trzy pojęcia w opisie działania programu; były nimi: rozwiązanie problemu, algorytm i proces heurystyczny. Pierwsze pojęcie rozumieli jako jeden z elementów zbioru wszystkich możliwych rozwiązań; metodą było testowanie poszczególnych możliwości. Ponieważ jednak ten zbiór mógł być bardzo duży, proces generowania poszczególnych ewentualności przebiegał w określonym porządku. Generator gwarantował, że jeśli problem ma rozwiązanie, to zostanie ono wytworzone wcześniej czy później. Newell, Shaw i Simon nazwali proces posiadający taką własność algorytmem rozwiązania problemu. Inny proces mogący rozwiązać dany problem, ale nie dający gwarancji rozwiązania go, określili jako heurystyczny proces rozwiązywania problemu.

Algorytm budował w systematyczny sposób wszystkie możliwe dowody danego twierdzenia i kontrolował je — eliminował wtedy duplikaty oraz sprawdzał, czy końcowe wyrażenie pokrywa się z przedstawionym do udowodnienia. Trzej uczeni zastosowali algorytm do sześćdziesięciu nieparzystych twierdzeń z II rozdziału *Principia mathematica*. Niektóre twierdzenia udowodnione były stosunkowo szybko, np. twierdzenie (2.01) w czwartym kroku generowania rozwiązań jako jedno z 42 dowodów, inne (2.05) w w ósmym kroku jako jedno z 246 dowodów. Część twierdzeń wymagała jednak w dowodzeniu bardzo złożonych metod. „Wygenerowanie w ten sposób dowodów twierdzeń z rozdziału II zajęłoby setki tysięcy lat liczenia".[53]

Wyżej wspomniany algorytm znany jest w literaturze pod nazwą Algorytmu Muzeum Brytyjskiego. Nie zdał on egzaminu w odniesieniu do udowadniania twierdzeń logicznych; w drugim etapie badań nad Maszyną do Teorii Logiki został zastąpiony metodami heurystycznymi: podstawiania, odrywania oraz łań-

---

[52] Por. ich artykuł: *Badania empiryczne przeprowadzone na maszynie do teorii logiki*, w: *MMiM*, s. 118-144.

[53] Tamże s. 126.

cuchowymi. Nie dawały one gwarancji skuteczności, ale zapewniały, że każdy generowany podproblem stanowił część ciągu kończącego się żądanym twierdzeniem; zapewniały również, że każde wyrażenie jest wyprowadzone z poprzednich przy pomocy reguł wnioskowania. Zastosowanie heurystyk skróciło czas dowodzenia twierdzeń od pięciu razy (najprostsze twierdzenia) do kilkuset razy (trudniejsze twierdzenia).

Kończąc swój artykuł Newell, Shaw i Simon zaproponowali szereg modyfikacji LT.[54] W kolejnej wersji program miał wybierać jedynie te podproblemy, które z dużym prawdopodobieństwem prowadziły do dowodu. Uczeni proponowali także inne modyfikacje programu, które doprowadziłyby do selektywnego zredukowania liczby generowanych problemów oraz do analizowania jeszcze większej liczby podproblemów przy takiej samej mocy obliczeniowej komputera. Badania nad rozwiązywaniem problemów doprowadziły Newella, Shawa i Simona do porównań skuteczności procesów algorytmicznych z heurystycznymi; ocena drugiej metody była znacznie pozytywniejsza.

Pierwsza wersja programu dowodzącego twierdzeń logiki oparta była na algorytmie i nadawała się jedynie do udowadniania prostych twierdzeń. Tę wersję zaliczyć można do programów sformalizowanych w sposób prosty czyli do podpola II. Uczeni szybko jednak zrezygnowali z tego rozwiązania i zastosowali metody heurystyczne, właściwe dla III podpola w klasyfikacji Dreyfusa. Metody te lepiej nadawały się do udowadniania tego typu twierdzeń, eliminowały możliwości nie rokujące nadziei na rozwiązanie i dały w rezultacie o wiele lepsze wyniki — w sumie udowodnionych zostało 38 spośród 52 twierdzeń z II rozdziału *Principia mathematica*.

W trzy lata później (1960) H. Wang[55] napisał program zdolny udowodnić wszystkie twierdzenia logiki rachunku zdań podane przez Russella i Whiteheada w *Principia mathematica*. O ile przedmiotem zainteresowań Newella, Shawa i Simona były przede wszystkim metody heurystyczne, o tyle Wang poszukiwał algorytmów wymagających mniejszej mocy obliczeniowej komputerów, a jednocześnie dających interesujące dowody w rozsądnym czasie. W trakcie badań doszedł on jednak do wniosku, że w wypadku złożonych systemów formalnych trzeba stosować heurystyki, aby jego algorytmy pracowały wystarczająco selektywnie i dawały interesujące dowody w czasie możliwym do przyjęcia.

d. Rozwiązywanie ogólnych problemów

Gelernter przewidywał[56] trudności w projektowaniu programów rozwiązujących jakiekolwiek problemy i nie mylił się. Potwierdzeniem jego przypuszczeń były prace nad programem General Problem Solver prowadzone przez Newella

---

[54] Tamże s. 143-144.

[55] Por. jego artykuł: *Toward Mechanical Mathematics*, IBM Journal of Research and Development, 4 (1960) s. 2-22.

[56] Gelernter, *Realizacja*, s. 157.

i Simona. Po badaniach związanych z programem Logic Theory Machine, uczeni ci kontynuowali analizy dotyczące dziedziny AI, określanej jako *problem solving*.[57] Tym razem chcieli oni napisać program, który mógłby opracowywać różnego rodzaju problemy. Przy pomocy programu komputerowego pragnęli odwzorować teorię wyjaśniającą sposób rozwiązywania problemów przez człowieka.

Skomplikowana struktura programu, zastosowanie w nim heurystyk oraz kalkulacje-labirynty są cechami wyszczególnionymi przez Dreyfusa dla programów należących do III podpola klasyfikacji: Complex-Formal; są to również cechy charakteryzujące General Problem Solver. GPS traktuje zadanie jako zespół zagadnień (obiektów); są one przekształcane przez różne operatory. Program wykrywa różnice między zagadnieniami oraz klasyfikuje informacje na temat środowiska zadania w pewne grupy celów. Newell i Simon wyróżniają trzy rodzaje celów: przekształcenie zagadnienia A w prostsze zagadnienie B, zmniejszenie różnicy D pomiędzy A i B oraz zastosowanie operatora Q do zagadnienia A. Program GPS jest więc zasadniczo sposobem rozwiązania danego problemu na drodze przekształcania podcelów i wykorzystania metody stosowanej w podcelach do rozwiązania problemu głównego.

Metody te tworzą system generujący drzewo podcelów dla danego celu. Każda komplikacja powoduje generowanie nowego podcelu dotąd, aż zostanie ona przezwyciężona. GPS dysponuje pewnymi testami, aby nie dopuszczać do nadmiernego rozrastania się drzewa możliwości w bezużytecznych kierunkach. Przykładowo program nie przeprowadza próby dla podcelu trudniejszego od jednego ze swoich nadceli. Inne testy dotyczą zewnętrznych ograniczeń oraz badają, czy nowe obiekty lub cele są identyczne z poprzednio już wygenerowanymi.

Program GPS dobrze sobie radził z rozwiązywaniem problemów typu „kanibale i misjonarze" oraz z o wiele bardziej złożonymi zagadnieniami. W General Problem Solver autorzy programu starali się oddzielić ogólną część programu (analiza krokowa) od szczegółowej części dotyczącej określonego problemu. W ten sposób chcieli uczynić GPS na tyle elastycznym, by mógł służyć do rozwiązywania różnego typu kwestii. I rzeczywiście, jeszcze w tym samym roku (1959) ogólna część GPS została wykorzystana przez Newella, Shawa i Simona do rozwiązywania tożsamości trygonometrycznych. W rok później F.M. Tonge zastosował go w programie zapewniającym równowagę linii produkcyjnych. W 1961 r. Simon zastosował ogólną część GPS do kompilowania programów komputerowych. Na podstawie tych osiągnięć autorzy artykułu „GPS-Program, który symuluje ludzką myśl" wyciągnęli wniosek, że „swobodne za-

---

[57] A. Newell, H.A. Simon, *GPS — program, który symuluje myśl ludzką*, w: *MMiM*, s. 275-290. Por. też A. Newell, J.C. Shaw, H.A. Simon, *Report on a General Problem-solving Program*, w: *Proceedings of the International Conference on Information Processing (ICIP), UNESCO House*, Paris 1959, s. 256-264.

chowanie się opowiednio inteligentnego człowieka można rozumieć jako wynik działania złożonego, ale skończonego i zdeterminowanego zbioru praw"[58].

Wniosek ten okazał się jednak zbyt pochopny i musi dziwić, że znalazł się na końcu artykułu relacjonującego eksperyment, który wcale nie powinien upoważniać do tego typu stwierdzeń. W eksperymencie opisanym przez Newella i Simona człowiek i program mieli udowodnić pewne twierdzenie logiczne, mając do dyspozycji 12 reguł. Porównanie rozwiązań wykazało szereg różnic w podejściu do problemu. Autorzy artykułu wyjaśnili dwie istotne różnice w sposób wymijający i wyrazili nadzieje na zlikwidowanie ich przez dodanie do programu pewnych mechanizmów. Nasuwa się jednak pytanie, czy mimo wspomianych różnic GPS można uznawać za dokładny model ludzkiego przetwarzania informacji podczas rozwiązywania problemu? Dreyfus wątpi w to i, analizując zagadnienie, stara się wykazać przyczyny występujących różnic.[59]

Po pierwszej porażce przyszła następna. Do rowiązania posunięto programowi GPS łamigłówkę „Siedem mostów w Królewcu", od której *nota bene* rozpoczęła się teoria grafów i topologia. Euler udowodnił, w 1736 r. przy pomocy analiz topologicznych, że zadanie jest nie do rozwiązania; GPS natomiast nie potrafił wykazać owej nierozwiązywalności. Badania i dyskusje nad programem General Problem Solver trwały do stycznia 1967 r., kiedy G.W. Ernst oraz A. Newell przyznali, że GPS załamał się pod wpływem swojej własnej struktury: „Poważnym ograniczeniem oczekiwanych wyników od programu GPS jest objętość programu oraz tym bardziej objętość układu danych. Już sam program zajmuje znaczną część pamięci komputera a generowanie nowych stuktur danych szybko wyczerpuje resztę pamięci. Tak więc GPS jest przeznaczony jedynie do rozwiązywania skromnych problemów (...)".[60]

2. Uwagi Dreyfusa na temat barier drugiego pola AI

a. Koncepcje rozwiązywania problemów

Przed przedstawieniem analiz Dreyfusa warto krótko omówić koncepcje rozwiązywania problemów jakie zaproponowali G. Polya i M. Weltheimer. Koncepcje te, nieco zmodyfikowane przez M. Minsky'ego, zostały zastosowane w badaniach nad AI przez Newella i Simona.

Prof. Polya rozróżniał następujące fazy z rozwiązywaniu problemów:[61] pierwszy etap to zrozumienie problemu — trzeba jasno określić dane, warunki oraz sprecyzować niewiadomą. Drugim etapem jest określenie planu, dzięki któremu otrzymane zostanie rozwiązanie a dane połączone z niewiadomą.

---

[58] Newell, Simon, jw. s. 290.

[59] Analizom tym poświęcony jest następny paragraf niniejszej pracy.

[60] Cytat za *WCCD*, s. 96.

[61] Por. G. Polya, *How to Solve It: A New Aspect of Mathematical Method*, New York: Doubleday 1945.

Przedstawiciel gestaltyzmu, Weltheimer, był zdania, że najważniejszym zagadnieniem w rozwiązywaniu problemu jest uchwycenie jego istotnej struktury.[62] Po wydobyciu się z powierzchownej struktury, można dostrzec rdzeń zadania, co Weltheimer nazywa głębszą strukturą; pozwala ona na podjęcie odpowiednich kroków zmierzających do rozwiązania.

W metodzie rozwiązywania problemów zaproponowanej przez Minsky'ego, odnajdujemy ślady obydwu wyżej opisanych koncepcji. Wydobycie się z powierzchniowej struktury kwestii i przejście do głębszej (Weltheimer), przyjęło u Minsky'ego postać przekształcenia trudnych problemów na szereg prostszych zagadnień. Nie można jednak robić tego przypadkowo — najpierw należy zrozumieć zadanie, a następnie wytworzyć prostsze modele sytuacji o takiej strukturze, by pozwoliła ona znaleźć sposób (Polya) rozszerzenia ich rozwiązań na pierwotny problem.[63]

b. Strukturalizacja zagadnień istotnych i nieistotnych w złożonych problemach

Czy taką koncepcję można jednak zastosować w AI w odniesieniu do *problem solving*? Istotną rzeczą jest uchwycenie głębszej struktury problemu, podzielenie go na składowe oraz określenie, które z zagadnień są ważniejsze od pozostałych. W jaki sposób zostało to dokonane przede wszystkim w odniesieniu do programu GPS, sformalizowanego w sposób złożony? Newell i Simon wprowadzili do GPS heurystyki planowania, które według nich miały przejąć funkcję określania, co jest istotne a co mniej istotne w problemie. Ów system heurystyk złożony jest z omijania pewnych szczegółów zadania. Uproszczony problem można rozwiązać przy pomocy nieco bardziej znanego planu, a następnie ów plan posłuży jako metoda w rozwiązaniu pierwotnego problemu.

W jednej ze swoich aplikacji program GPS zawierał 12 operatorów dozwolonych w logice O.K. Moora; obaj naukowcy osiem spośród nich zaliczyli do istotnych, pozostałe zostały sklasyfikowane jako nieistotne. Kryterium podziału były zmiany, jakie następowały w wyrażeniu po zastosowaniu operatora. Operator istotny wprowadzał duże zmiany w wyrażeniu, np. zmieniał *P lub P* w *P*; operator nieistotny powodował małe zmiany np. przekształcał *P lub Q* w *Q lub P*. Operatory zastosowane do przekształcenia pierwotnego problemu a w b dają nowy problem przekształcenia *a'* w *b'*. Teraz można dokonać przekształcenia wstecz — do pierwotnego zagadnienia. W ten sposób metody otrzymane w *a'* oraz *b'* dostarczają planu rozwiązania pierwotnego problemu. Wydawałoby się więc, że zostało znalezione rozwiązanie — heurystyki wbudowane w program komputerowy umożliwiają rozwiązywanie jakiegokolwiek ogólnego problemu. Dreyfus nie godzi się jednak z taką opinią i po pierwsze wskazuje na błędną

---

[62] Por. M. Weltheimer, *Productive Thinking*, New York: Harper and Row 1945, s. 202nn.

[63] Por. M. Minsky, *Descriptive Languages and Problem Solving*, w: *Proceedings of the Western Joint Computer Conference, May 1961*, s. 422nn.

metodę zastosowaną przez Newella i Simona we wstępnej klasyfikacji problemów, a po drugie twierdzi, że odróżnianie elementów istotnych od nieistotnych jest specyficznie ludzką zdolnością.

Weltheimer postulował znalezienie głębszej struktury problemu; podział operatorów na istotne i nieistotne w GPS wydaje się spełniać ten postulat. Dreyfus ma jednak podstawowe zastrzeżenie[64] — co do metody zastosowanej przez Newella i Simona. To oni, a nie program GPS, podzielili operatory na istotne i nieistotne; to programiści dokonali tej klasyfikacji. Dreyfus podkreśla: „Mówienie o heurystykach w tym miejscu jest zupełnie mylące, ponieważ nikomu jeszcze nie udało się określić reguł kierujących wstępnym wyborem".[65]

Trudności z określeniem tych reguł związane są z różnicami, jakie zachodzą przy rozwiązywaniu problemów przez człowieka i program komputerowy GPS. Newell i Simon jakby nie dostrzegali tych różnic lub próbali je zatrzeć, a jeden z nich bardzo optymistycznie nawet wyrażał opinię: „Prowadzone badania skłoniły do potwierdzenia początkowych przypuszczeń i pokazały, że heurystyki, bądź praktyczne zasady kształtują integralny trzon ludzkich procesów rozwiązywania problemów. Kiedy zaczęliśmy rozumieć naturę heurystyk, których ludzie używają w myśleniu, tajemnica [myślenia] zaczęła się wyzwalać z tak dotąd niejasno rozumianych procesów jak 'intuicja' i 'osąd'".[66]

Artykuł opublikowany w kilka miesięcy później był dowodem znacznie już ostrożniejszego podejścia — autorzy byli mniej optymistycznie nastawieni pisząc, że „proces 'myślenia' nie może już dłużej uchodzić za zupełnie tajemniczy".[67] Mniej skrajne podejście spowodowane było zapewne trudnościami na jakie napotkali uczeni w próbach z GPS.

Problemy te związane były z różnicami w działaniu człowieka i programu, jakie ujawniły się w eksperymencie[68] przeprowadzonym przez Newella i Simona. Polegał on na rozwiązywaniu tego samego zadania logicznego przez General Problem Solver i amerykańskiego studenta. Uczeni spodziewali się identycznego postępowania komputera i człowieka; uważali bowiem heurystyki programu GPS za dobrze symulujące zachowanie człowieka w czasie rozwiązywania problemów. Zbagatelizowanie różnic i wyciągnięte przez nich wnioski spotykały się z krytyką Dreyfusa.[69]

---

[64] *WCCD*, s. 116.

[65] *WCCD*, s. 117.

[66] H.A. Simon, *Modeling Human Mental Process*, The Rand Corporation, P-2221, February (1961) s. 15.

[67] A. Newell, H.A. Simon, *Computer Simulation of Human Thinking*, Science vol. 134 (1961) s. 19.

[68] Newell, Simon, jw., w: *MMiM*, s. 275-290. Por. zwłaszcza s. 283-290 dotyczące poglądów autorów na temat różnic między sprawozdaniem komputera a protokołem człowieka.

[69] *WCCD*, s. 113n.

W zadaniu dane były dwa wyrażenia logiczne; jedno wyrażenie należało przekształcić w drugie przy pomocy dwunastu dozwolonych reguł. Rozwiązania różniły się w pięciu miejscach, dwa spośród nich są istotne. Pierwsza różnica dotyczyła rozbieżności z stosowaniu reguły nr 8, złożonej z dwóch członów: *A i B implikuje A* oraz *A i B implikuje B* w odniesieniu do wyrażenia:

$$(\sim R \vee \sim P) \cdot (R \vee Q).$$

Simon i Newell w ten sposób komentują tę rozbieżność: „(...) sprzeczność jest zupełnie jasna. Osobnik doświadczalny manipulował jednocześnie obiema postaciami reguły 8, przynajmniej o ile dotyczy to jego komentarza. GPS, z drugiej strony, rozważa każdą postać w oddzielnym cyklu. Może osobnik doświadczalny postępował skrycie według programu i po prostu podał te dwa wyniki razem".[70]

Odnośnie do tej różnicy Dreyfus zauważa, że jednak student miał rację stosując równocześnie oba człony reguły. Uchwycił bowiem najprawdopodobniej koniunkcję jako symetryczną w stosunku do przekształcenia kierowanego przez regułę nr 8. Nawet sami Simon i Newell odnoszą wrażenie, że rozwiązanie studenta jest lepsze i woleliby, aby GPS zastosował jego rozwiązanie; wtedy bowiem GPS lepiej symulowałby tok myślenia człowieka. Nie widzą jednak większego problemu z powodu tej różnicy i tłumaczą ją sekwencyjnym przetwarzaniem tam, gdzie student zastosował przetwarzanie równoległe.[71]

Druga rozbieżność jest jednak większa i autorzy artykułu przyznają, że w programie nie ma nic, co odpowiadałoby następującemu fragmentowi protokołu: „Właściwie powinienem był zastosować regułę 6 tylko do lewej strony równania. Wobec tego zastosuj regułę 6, ale tylko do lewej strony równania".[72]

GPS i student mieli wówczas do dyspozycji takie samo wyrażenie i zarówno program jak i człowiek doszli do takiego samego wniosku — zastosować należy teraz regułę nr 6. I tak właśnie zrobił GPS natomiast student cofnął się i poprawił poprzednie zastosowanie tej reguły.

Jak wyjaśnić tę różnicę? Czy dla Newella i Simona stanowiła ona wskazówkę, by przyjąć niemożność odwzorowania ludzkiej metody rozwiązywania problemów w programie komputerowym? Tej rozbieżności nie dało się już zbyć stwierdzeniami o równoległym i sekwencyjnym przetwarzaniu informacji. Dreyfus zarzuca im, że bez żadnego dowodu zakładają oni istnienie mechanizmu wyjaśniającego ową różnicę. Mechanizm ten miałby być jeszcze dokładniej opracowaną techniką poszukiwań; technika ta pozwalałaby programowi GPS na ciągły wgląd w poprzednie działania.[73] Wyjaśnienia Newella i Simona są nieprzekonujące i stanowią bardziej obejście problemu niż wyjaśnienie.

---

[70] Newell, Simon, jw. s. 286.
[71] Tamże s. 287.
[72] Tamże s. 288.
[73] *WCCD*, s. 114.

Dreyfus inaczej wyjaśnia różnice zachodzące w sprawozdaniu programu GPS i protokole człowieka.[74] Odwołuje się on do specyficznie ludzkiej zdolności rozróżniania spraw nieistotnych od istotnych w strukturze rozwiązywania problemów. Zdaniem Newella i Simona inteligentne zachowanie zawsze jest wynikiem stosowania zasad heurystycznych. Wobec czego decyzja studenta o cofnięciu się, również musiała być wynikiem zastosowania wyspecjalizowanej procedury heurystycznej kontrolującej studenta; według nich należy więc odnaleźć taką heurystykę — wbudowana w program pozwoliłaby programowi komputerowemu na dokładne symulowanie toku myślenia studenta. Pogląd ten Dreyfus uważa za nierealny. Z drugiej jednak strony, gdyby Newell i Simon nie przyjęli takiego założenia musieliby do GPS wprowadzić procedury sprawdzające wszystkie wykonane dotąd kroki i to na każdym etapie programu. Spowodowałoby to naturalnie znaczne obciążenie programu.

Postaciowa koncepcja zaadoptowana przez Minsky'ego wydaje się więc nie związana z *problem solving*. Niewiele daje przekształcanie głównego problemu na szereg mniej skomplikowanych podproblemów. Dreyfus stanowczo twierdzi, że: „Iluzją jest pogląd, że problem planowania może być rozwiązany w odosobnieniu, że operacje istotne i nieistotne są podane jak bryły i należy je jedynie uporządkować. Łatwo można zostać zahipnotyzowanym przesadnie prostymi i prowizorycznymi przypadkami — takimi jak problem logiczny — aby sądzić, że niektóre operacje same w sobie są istotne lub nieistotne".[75]

Koncepcja Minsky'ego odnośnie do rozwiązywania problemów nie została sformalizowana. Według Dreyfusa nie istnieją zasady heurystyczne pozwalające na dzielenie zagadnień na istotne i nieistotne; zdolność taka przysługuje jedynie człowiekowi.

c. Uczenie się maszyn a rozwiązywanie problemów

Broniąc swoją koncepcję Minsky wykonał typowy zwrot do zagadnienia uczenia się maszyn. Chciał on ograniczyć badanie do danych, które będą miały znaczenie dla bieżącego zadania. Jest co prawda świadomy trudności wbudowania w program takiej funkcji selekcyjnej, sądzi jednak, że funkcja taka musi się rozwijać wraz z danymi gromadzonymi przez doświadczenie.[76] Już w samych założeniach Minsky popełnia ten sam błąd, co Newell i Simon — w jaki bowiem sposób chce nauczyć program rozróżniania rzeczy istotnych od nieistotnych? A poza tym pamięć komputera może przepełnić się danymi, na podstawie których program ma uczyć się funkcji selekcyjnej.

---

[74] *WCCD*, s. 114 oraz s. 117-119.

[75] *WCCD*, s. 119.

[76] Por. Minsky, *Descriptive Languages*, jw. s. 215-218.

Siedem lat później E. Feigenbaum negatywnie podsumował wysiłki uczonych skupione na zagadnieniu programów uczących się.[77] Podkreślił nikłe wyniki osiągnięte pod tym względem w dziedzinie AI. Wymienił warcabowy program Samuela jako jeden z nielicznych programów tego typu zasługujących na uwagę i wskazał na niespełnione nadzieje pokładane w GPS. Zaznaczył, że nikomu z uczonych nie udało się dotąd (1968) pokonać tej bariery.

Nie powinno to jednak dziwić, jeśli przyjąć, że dla komputera niemożność rozróżniania między elementami istotnymi i nieistotnymi jest ograniczeniem nie do pokonania.[78] Jak starał się wykazać Dreyfus, rozgraniczanie to jest specyficznym procesem przetwarzania informacji, przysługującym jedynie człowiekowi. Próby symulowania tej zdolności na komputerze nie powiodły się. Rozpoznawanie rzeczy istotnych i nieistotnych w problemie nie jest więc wynikiem metod heurystycznych zawężających poszukiwanie rozwiązań, jak twierdzili Simon i Newell. Oni sami w programie GPS wprowadzili wstępną strukturalizację zagadnienia, a jeśliby tego nie zrobili, wszelkie poszukiwania rozwiązania problemów byłyby jedynie zamieszaniem.

Dreyfus wymienia jeszcze IV podpole w *problem solving*; należą do niego problemy o strukturze otwartej, zależne od znaczenia i sytuacji nie zadanej *explicite*; problemy takie w ogóle nie poddają się formalizowaniu i nie występuje żaden rodzaj programu odpowiadający tym zagadnieniom. Problemy tego typu są dla komputerów jeszcze jedną barierą nie do pokonania.

3. Podsumowanie

W odniesieniu do kolejnego pola sztucznej inteligencji — rozwiązywania problemów — powtarza się sytuacja, jaka miała miejsce w badaniach nad formalizowaniem gier; na początku prac uczeni odnoszą sukces, a faza zaawansowana przynosi porażki. Prostsze problemy dają się dobrze sformalizować i są rozwiązywalne przy pomocy komputerowych programów. Tak było w przypadku badań Gelerntera. Nie podjął on prac nad ułożeniem programu rozwiązującego jakiekolwiek problemy, lecz zawęził zakres zagadnień do geometrii euklidesowej. Przyniosło to pewien sukces w postaci programu Geometry Theorem Machine. Opierał się on w głównej mierze na algorytmach i korzystał z właściwości heurystycznych diagramu jedynie wtedy, gdy zagrażał wykładniczy wzrost możliwych rozwiązań. GTM, obok warcabowego programu Samuela, należy do większych osiągnięć sztucznej inteligencji przełomu lat 50. i 60.

---

[77] E. Feigenbaum, *Artificial Intelligence. Themes in the Second Decade, IFIP Congress 1968, Supplement, J-15.*

[78] Feigenbaum znał zapewne wyniki analiz Dreyfusa, bowiem zanim ukazały się one w formie książkowej autor *What Computers Can't Do* umieścił je w 1965 r. w artykule *Alchemy and Artificial Intelligence*, P-3244, Rand Corporation, Santa Monica 1965. Por. też Życiński, *Filozofia sztucznej inteligencji*, s. 281.

Wśród uczonych silne były jednak tendencje do opracowania ogólniejszej wersji programu udowadniającego twierdzenia matematyczne, odkrywającego prawa naukowe, rozumiejącego prozę angielską czy grającego w szachy; mieli oni nadzieję na odnalezienie ogólnych mechanizmów ludzkiego myślenia. Tendencje te dały o sobie znać zwłaszcza w badaniach Newella, Simona i Shawa nad Logic Theory Machine a zwłaszcza nad General Problem Solver. W pierwszym etapie prac uczeni ograniczyli jednak swoje plany do rozwiązywania zadań logiki formalnej. Pierwsza wersja programu LT działała jedynie na podstawie algorytmu; rezultaty były jednak mało zadowalające. Lepsze wyniki otrzymali uczeni po wprowadzeniu do programu metod heurystycznych. Efektem było 38 udowodnionych twierdzeń z *Principia mathematica*. Newell i Simon powrócili wtedy do pomysłu jeszcze ogólniejszej wersji programu. Tylko częściowo, w odniesieniu do prostszych problemów, udało im się zrealizować tę ideę w General Problem Solver. Ciekawą koncepcją było wydzielenie z GPS głównej części programu i aplikowanie jej do różnych zadań. Za każdym razem należało jednak przystosować główną część GPS do danego problemu — czy to trygonometrycznego czy też polegającego na kompilowaniu programu komputerowego. Tak więc GPS nie był wcale tak ogólny, jak głosiła jego nazwa i deklarowali twórcy. O ile GPS radził sobie z prostszymi problemami, o tyle w przypadku trudniejszych zagadnień wystąpiły trudności i uczeni ponieśli porażkę, do której przyznali się po latach. Nie udało się przy pomocy programu odwzorować ludzkiego sposobu rozwiązywania problemów, choć Newell i Simon sądzili, że zachowanie człowieka jest wynikiem skończonego i zdeterminowanego zbioru praw. Uczeni próbowali zastosować w *problem solving* koncepcje Polya oraz Weltheimera w wersji zmodyfikowanej przez Minsky'ego. Największe trudności w odwzorowaniu tych koncepcji w programie komputerowym stwarzało klasyfikowanie problemów na istotne i nieistotne. Według twórców GPS to zadanie miały przejąć heurystyki planowania. Ale zamiast heurystyk sami Newell oraz Simon przeprowadzili klasyfikację i Dreyfus zaznacza, że dotychczas nikomu nie udało się określić heurystyk tego typu. Porównanie protokołu człowieka ze sprawozdaniem komputera z rozwiązywania tego samego zadania logicznego ujawniło niemożność rozróżniania przez program GPS elementów istotnych od nieistotnych. Dreyfus podkreśla, że zdolność taka jest cechą specyficznie ludzką. Heurystyki General Problem Solver nie są zdolne dobrze symulować zachowanie człowieka podczas rozwiązywania problemów; próżne też były nadzieje Minsky'ego związane z uczeniem się programów.

Z punktu widzenia filozofii interesujące są deklaracje Newella i Simona umieszczone na początku i na końcu artykułu *GPS — Program, który symuluje myśl ludzką*. Autorzy uznają, jako im najbliższą, koncepcję wypośrodkowaną między behawioryzmem a gestaltyzmem. Przyjmują też, że: „istota ludzka jest ogromnie złożonym zorganizowanym systemem oraz wydaje się, że proste

schematy nowoczesnej behawiorystycznej psychologii jedynie w zarysie to odzwierciedlają".[79]

Po takich konstatacjach trudno uznać za uzasadnione zakończenie artykułu, gdzie autorzy określają programy typu *problem solving* jako dowód na to, że działaniem człowieka rządzi skończony i zdeterminowany zbiór praw.[80] Początek artykułu wskazuje na pewną niechęć Newella i Shawa do mechanistycznego traktowania myśli i wrażeń człowieka jak to ujmował asocjacjonizm. Zakończenie natomiast, mimo wcześniejszych deklaracji, zdradza inklinacje autorów do mechanicyzmu.

W 1967 r. Newell przyznał rację oponentom i jasno stwierdził przydatność programu GPS do rozwiązywania jedynie niezbyt skomplikowanych zagadnień. Wydaje się, że twórcy programu Graph Traverser (1968) wyciągnęli właściwe wnioski i nie rościli już sobie pretensji do napisania jeszcze bardziej ogólnej wersji Ogólnego Rozwiązywacza Problemu — w badaniach nastąpił zwrot ku koncepcji programów sformalizowanych prosto. Nastąpiło to zapewne wskutek uświadomienia sobie, że „tam, gdzie ludzie opierają się na intuicji, programiści popadają w kłopoty"[81] oraz że nie jest możliwe zaprogramowanie rozróżniania rzeczy istotnych od nieistotnych, co podkreślał Dreyfus.

THE LIMITATIONS OF ARTIFICIAL INTELLIGENCE ACCORDING TO HUBERT L. DREYFUS

Summary

Artificial intelligence (AI) is an attempt of simulating of human intelligent behaviour using programming techniques. Since the field first evolved in the mid-1950s, AI researchers believed that insights into the nature of the mind can be gained by studying the operation of computer programs and invented dozens of computer programs that support some sort of intelligent behaviour.

In suggesting that a machine with vast intellectual capability was in the offing and that this new intelligence might quickly be attained, these advocates made themselves hostage to critics, who, increasingly aware of the limitations of certain of the theories being advanced, insisted on their inadequacies. The critics resented the exaggeration implicit in the claims of those, who saw only the promise of quick results and the epistemological assumptions implicit in the name the field had given itself.

Hubert L. Dreyfus in his *What Computers Can't Do* stirred up a controversy among all those interested in the possibility of formal models of man by arguing that, despite a decade of impressive print-outs and dire predictions of superintelligent robots, workers in artificial intelligence were (1967) facing serious difficulties which they tended to cover up with special-purpose solutions and rhetorical claims of generality.

---

[79] Newell, Simon, jw. s. 275.

[80] Tamże s. 290.

[81] *WCCD*, s. 112.

In the seventies experimental AI systems included generally four kinds of programs: games, problem solving, language translating and pattern recognition (classification of H. Dreyfus). This first part of my paper *Limitations of artificial intelligence by Hubert L. Dreyfus* is about games and problem solving.

The basic problem facing workers attempting to use computers in the simulation of human intelligent behaviour should now be clear: all alternatives must be made explicit — in game playing there is the exponential growth of the tree of these alternative paths. Dreyfus set human fringe consciousness against heuristically guided programm search. His analyses reveal two kinds of games: computable or quasicomputable (e.g. tic-tac-toe) and incomputable (e.g. chess). In problem solving the issue is not only how to direct a selective search among the explicit alternatives, but how to structure the problem so as to begin direct a selective search process (human essential/unessential discrimination vs. trialand-error search of computer). Impossible for counting out are open-structured problems, computable — maze problems, combinatorial problems, complex combinatorial problems.

In surveying the two fields of AI (game playing and problem solving) underlying the optimistic interpretation of results in AI Dreyfus observed a current pattern: in each case the assumption was taken to be self-evident — an axiom seldom articulated and never called into question. In fact, the assumption turned out to be only one alternative hypothesis and a questionable one at that. The biological assumption that the brain must function like a digital computer no longer fits the evidence. The others lead to conceptual difficulties. The psychological assumption that the mind must obey a heuristic program cannot be defended on empirical grounds and a priori arguments in its defense fail to introduce a coherent level of discourse between the physical and the phenomenological.