1 Maszyna Minsky'ego

Zacznijmy od sprecyzowania pojęcia maszyny Minsky'ego. Będziemy ją definiować jako specyficzną maszynę Turinga.

1.1 Maszyna Turinga

Będziemy rozważać maszyny Turinga:

1) z taśmą wejściową i dwoma taśmami roboczymi, taśmy te będą jednostronnie nieograniczone, będą więc miały pierwszą komórkę, w której będzie stale zapisana specjalna litera \$ oznaczającą początek taśmy. Głowice maszyny Turinga nie potrafią pisać tej litery, nie mogą jej też zmienić (zastąpić inną), potrafią ją odczytać i jeżeli ją widzą, nie wykonują ruchu w lewo.

2) służące do akceptowania danego słowa. W słowie tym nie może występować symbol początku taśmy. Litery danego słowa są zapisywane przed uruchomieniem maszyny w kolejnych komórkach taśmy wejściowej, począwszy od drugiej. Głowica taśmy wejściowej, widząc niezapisaną komórkę, nie wykonuje ruchu w prawo.

3) które w każdym momencie znajdują się w pewnym stanie. Rozpoczynają pracę w stanie początkowym. Akceptują dane, jeżeli podczas pracy w pewnym momencie znajdą się w stanie akceptującym.

4) działają zgodnie z ustaloną relacją przejścia, ktora może być rozumiana jako program (a więc wykonują program) będący skończonym zbiorem rozkazów. Bywa, że rozkazy są zapisywane w postaci

$$q_i\ w\ r_1\ r_2 \rightarrow s_1\ s_2\ R_0\ R_1\ R_2\ q_j$$

(symbol $\rightarrow$ można więc uznać za oznaczenie relacji przejścia). Taki rozkaz może być wykonany, gdy maszyna znajduje się w stanie q_i i obserwuje na taśmie wejściowej literę w, a na taśmach roboczych – odpowiednio litery r_1 i r_2. Jeżeli jest wykonywany, to jego wykonanie polega na zapisaniu w obserwowanych komórkach taśm roboczych liter s_1 i s_2 odpowiednio (zastąpieniu obserwowanych liter wskazanymi), przesunięciu głowic w lewo lub w prawo, bądź pozostawieniu ich na swoim miejscu zależnie od znajdujących się w rozkazie wartości $R_0\ R_1\ R_2$ i przejściu do stanu q_j. Oczywiście, rozkazy muszą spełniać wyżej podane ograniczenia. Oprócz liter w rozkazach maszyny Turinga występuje także specjalny symbol B oznaczający zależnie od kontekstu albo pustą komórkę, albo polecenie usunięcia z obserwowanej komórki zapisanej tam litery.

5) działa zgodnie z następującą zasadą: spośród swoich rozkazów maszyna wybiera jeden, możliwy do wykonania (niedetrministycznie) i wykonuje go, a gdy nie jest to możliwe, to zatrzymuje się.

Maszyny Turinga (i pozostałe) będziemy utożsamiać z programami opisującymi ich działanie. Jest to bardzo wygodne, ale aby było to możliwe, mając dany program powinniśmy móc ustalić, który ze stanów jest początkowym i jakie stany są akceptujące.

Konfiguracją maszyny Turinga nazywamy trzy słowa utworzone z liter zapisywanych na taśmach, symboli B i liter będących stanami maszyny (odpowiadających stanom). Słowa te odpowiadają poszczególnym taśmom. W każdym występuje dokładnie jeden symbol stanu, ten sam we wszystkich słowach. Maszyna znajduje

się w danej konfiguracji, jeżeli znajduje się w stanie podanym w tej konfiguracji i na poszczególnych taśmach, przed komórką obserwowaną znajdują się litery wymienione w odpowiednim słowie przed stanem, a począwszy od obserwowanej komórki – litery wymienione po symbolu stanu. Symbol B jest rozumiany odpowiednio.

Konfiguracją $k_0(s)$ początkową ze słowem s nazywamy konfigurację, w której maszyna znajduje się w wyróżnionym stanie początkowym q_0, na taśmie wejściowej jest zapisane słowo s, taśmy robocze są puste (a właściwie zawierają wyłącznie symbole początku taśmy, które pomijamy w konfiguracjach), a głowice obserwują komórki znajdujące się bezpośrednio za komórkami z symbolem początku taśmy. Taką konfigurację można opisać jako trójkę $(q_0 s, q_0 B, q_0 B)$ lub nawet $(q_0 s, q_0, q_0)$.

Relację przejścia maszyny Turinga rozszerzamy do relacji $\rightarrow$ w zbiorze konfiguracji. Zapis $k_1 \rightarrow k_2$ oznacza, że maszyna znajdująca w konfiguracji k_1 po wykonaniu jednego z rozkazów z jej programu przechodzi do konfiguracji k_2. Symbolem $\rightarrow^*$ oznaczamy zwrotne i przechodnie domknięcie relacji $\rightarrow$ w zbiorze konfiguracji.

Mówimy, że maszyna akceptuje słowo s (nad ustalonym alfabetem Σ) jeżeli $k_0(s) \rightarrow^* k$ dla pewnej konfiguracji k ze stanem akceptującym. Jeżeli M jest maszyną Turinga, to $L(M)$ oznacza język akceptowany przez maszynę M, czyli

$$L(M) = \{s \in \Sigma^* : \ M \ \text{ akceptuje } \ s\}.$$

Opisany wyżej model maszyny Turinga jest równoważny każdemu innemu sensownemu modelowi maszyny Turinga. W związku z tym możemy się nim równie dobrze posługiwać, jak każdym innym.

1.2 Maszyny Minsky'ego

Wydaje się, że maszyna Minsky'ego jest to maszyna Turinga, taka jak wyżej, która na taśmach roboczych nie zapisuje żadnych symboli. Jej rozkazy mają więc postać

$$q_i \ w \ b_1 \ b_2 \rightarrow b_1' \ b_2' \ R_0 \ R_1 \ R_2 \ q_j,$$

gdzie b_1, b_2, b_1', b_2' są symbolami B lub symbolami początku taśmy \$.

Pojęcia opisujące maszyny Minsky'ego definiujemy tak, jak w przypadku maszyn Turinga.

Maszyny Minsky'ego pozwalają symulować maszyny Turinga. Wiadomo w szczególności, że mamy

Twierdzenie 1.1 *Istnieje redukcja f, która programowi P maszyny Turinga rozpoznającej język L, przyporządkowuje program $f(P)$ maszyny Minsky'ego również rozpoznającej język L.* □

Także problem stopu dla maszyn Turinga daje się zredukować do problemu stopu dla maszyn Minsky'ego.

W dalszym ciągu będziemy posługiwać się jeszcze bardziej uproszczonym modelem maszyn Minsky'ego. Będą to maszyny bez taśmy wejściowej. Będą też wykonywać uproszczone rozkazy następujących postaci:

$$q_i \ t \ \$ \rightarrow \langle \text{w prawo} \rangle \ q_j,$$

$$q_i \ t \ B \rightarrow \langle \text{w prawo} \rangle \ q_j,$$

$$q_i \ t \ B \rightarrow \langle \text{w lewo} \rangle \ q_j.$$

Rozkazy te analizują zapis i przesuwają głowice tylko na jednej taśmie, znajdujące się w nich t jest numerem taśmy, której rozkaz dotyczy, $t = 1, 2$. Interpretacja tych rozkazów jest, jak sądzę, oczywista.

Przez konfigurację uproszczonej maszyny Minsky'go będziemy rozumieć trójkę złożoną ze stanu maszyny i dwóch liczb naturalnych. Tak rozumiana konfiguracja z liczbami m i n odpowiada zwykłej konfiguracji, w której głowice taśm roboczych obserwują odpowiednio m-tą i n-tą z kolei pustą komórkę, a głowica taśmy wejściowej – pierwszą. Konfiguracją początkową uproszczonej maszyny Minsky'ego jest więc trójka $(q_0, 1, 1)$. Inne pojęcia opisujące pracę takich maszyn definiujemy tak, jak w przypadku zwykłych maszyn Minsky'ego lub maszyn Turinga.

Z uproszczoną maszyną Minsky'ego wiąże się problem akceptowania przez takie maszyny, czyli pytanie, czy dana taka maszyna podczas pracy w pewnym momencie znajduje się w stanie akceptującym. Problem ten jest formalizowany zwykle jako język

$$\mathcal{L}_m = \{P : \ P \ \text{ jest programem dla uproszczonej maszyny Minsky'ego, która po uruchomieniu przechodzi do stanu akceptującego } \}$$

(zamiast języka programów można oczywiście rozważać język numerów programów).

Niech $\mathcal{L}_t$ oznacza język złożony z (numerów) programów dla maszyn Turinga, które akceptują słowo puste. Wiadomo, że język ten jest rozpoznawalny, ale nierozstrzygalny. Powinno zachodzić

Twierdzenie 1.2 *Język $\mathcal{L}_t$ redukuje się do języka $\mathcal{L}_m$. Wobec tego język $\mathcal{L}_m$ nie jest rozstrzygalny.*

Dowód. Powinien korzystać z tych samych idei co dowód poprzedniego twierdzenia. □

2 Pies

Pies to taki stwór, który porusza się po płaszczyźnie, zostawia po sobie zapach i czasem szczeka. Poza tym, podobnie jak maszyna Turinga, zachowuje się zgodnie z określonym programem. Taki pies wydaje się być raczej robotem przypominającym psa, choć Turing upierał się, że definiując swoje maszyny opisał także zasadę działania mózgu człowieka, a więc może to jednak pies, a nie robot. (Przytoczony pogląd Turinga jest ważnym argumentem przemawiającym za tezą Turinga-Churcha: jeżeli nasz mózg działa jak maszyna Turinga, to trudno sobie wyobrazić, że (używając mózgu) policzymy coś, czego nie da się policzyć na maszynie Turinga).

Płaszczyzna, po której biega pies, składa się z pól (komórek) o całkowitych współrzędnych. Pola tej płaszczyzny mogą być albo bez zapachu, czyli puste lub niezapisane, a więc nieodwiedzone jeszcze przez psa, albo z zapachem, czyli zapisane, a więc już odwiedzone. Pola bez zapachu będziemy uważać za pola z symbolem B (także przez analogię z niezapisanymi polami maszyn Turinga), z zapachem – za pola z symbolem Z.

W każdym momencie pies znajduje się w jednym ze skończenie wielu stanów. W niektórych stanach pies szczeka. Dokładniej: pies szczeka wtedy i tylko wtedy, gdy znajduje się w jednym ze stanów „szczekających".

Zachowanie psa określa program. Taki program jest wykonywany analogicznie, jak program maszyny Turinga. Tym razem składa się jednak z dwóch rodzajów rozkazów:

$$q_i \ B \to R \ q_j \ \ \text{oraz} \ \ q_i \ Z \to R \ q_j,$$

gdzie R opisuje przemieszczanie się psa. Podane rozkazy są wykonywane, gdy pies znajduje się w stanie q_i i przebywa na polu, które nie pachnie (w przypadku pierwszego rozkazu), lub które już posiada zapach (rozkaz drugi). Pies znajdujący się

na jakimś polu w wyniku wykonania rozkazu może znaleźć się na jednym z czterech pól sąsiednich, albo znajdujących się powyżej lub poniżej, albo też znajdujących się z lewej lub z prawej strony. Wykonanie tych rozkazów polega na pozostawieniu zapachu (zapisaniu litery Z) na polu, na którym pies się aktualnie znajduje, przemieszczeniu się na sąsiednie pole wskazane przez R i przejściu do stanu q_j.

Aby uprościć programowanie będziemy się też posługiwać złożonymi rozkazami postaci

$$q_i\ X \to R_1,\ R_2,\ \ldots,\ R_n\ q_j.$$

Rozkaz taki jest wykonywany, gdy pies znajduje w stanie q_i i „obserwuje" X ($X = B$ lub $X = Z$). Jego wykonanie polega na wykonaniu serii przejść opisanych przez $R_1,\ R_2,\ \ldots,\ R_n$, przy czym drugi ruch i dalsze są wykonywane bez względu na zapach pola. Taki rozkaz bez trudu daje się wyrazić za pomocą zbioru $2n - 1$ rozkazów podstawowej postaci.

W rozważany przypadku konfiguracją będziemy nazywać układ obejmujący: stan psa, współrzędne pola zajmowanego przez psa oraz współrzędne wszystkich pachnących pól (zwykle powinno być ich skończenie wiele). W konfiguracji początkowej k_0 pies znajduje się w stanie początkowym na polu o współrzędnych $(0, 0)$ i wszystkie pola są bez zapachu.

Mając program, czyli odpowiednią relację przejścia $\to$, rozszerzamy ją do relacji przejścia $\to$ w zbiorze konfiguracji, a następnie bierzemy zwrotne i przechodnie domknięcie $\to^*$ relacji $\to$ (w zbiorze konfiguracji). W razie potrzeby będziemy uzupełniać symbole relacji o symbol programu wykorzystanego w definicji.

W dalszym ciągu będziemy zajmować się problemem szczekania, czyli problemem, czy pies zachowujący się zgodnie z danym programem P w pewnym momencie zaszczeka. Problem ten zwykle formalizuje się jako język $\mathcal{L}_p$ złożony z tych programów P opisujących zachowanie psa, które powodują, że w pewnym momencie pies zaszczeka. Jeszcze inaczej można $\mathcal{L}_p$ zdefiniować przyjmując, że

$$P \in \mathcal{L}_p \Leftrightarrow \exists k\ k_0 \to_P^* k \wedge k \text{ jest konfiguracją ze stanem szczekającym.}$$

Programy powinny być tak rozumiane, aby jakoś z nich wynikało, który stan jest początkowy, i w których stanach pies szczeka.

3 Zadanie

Zadanie, które mamy rozwiązać, wymaga, by dowieść następujące

Twierdzenie 3.1 *Język $\mathcal{L}_m$ można zredukować do języka $\mathcal{L}_p$, a więc w szczególności język $\mathcal{L}_p$, czyli problem szczekania, jest nierozstrzygalny.*

Dowód. Mamy więc zdefiniować pewną redukcję f, która uproszczonej maszynie Minsky'ego M z programem P przyporządkowuje (właściwie) psa zachowującego się zgodnie z programem $f(P)$, taką że

$$P \in \mathcal{L}_m \Leftrightarrow f(P) \in \mathcal{L}_p.$$

Definiowany pies ma symulować działanie maszyny M. Wyjaśnienie, co to znaczy, wymaga dodatkowych pojęć i pewnych ustaleń. W szczególności, jego stanami będą wszystkie stany maszyny M i wiele stanów pomocniczych, które będą stopniowo wymieniane podczas konstruowania programu $f(P)$. Stanem początkowym będzie stan początkowy maszyny M, a stanami „szczekającymi" – stany akceptujące M.

Będziemy mówić, że konfiguracja k koduje liczbę $n > 0$, jeżeli w tej konfiguracji w kolumnie, w której znajduje się pies, kolejne $n-1$ pól bezpośrednio poniżej pola,

na którym stoi pies, to pola z zapachem, a pozostałe, w tym to, na którym jest pies, są bez zapachu oraz pola w kolumnach na prawo od psa są pozbawione zapachu Jak widać, nic nie zakładamy o komórkach na lewo od psa i jest wiele konfiguracji kodujących liczbę n.

Dla danej konfiguracji K maszyny M będziemy definiować odpowiadającą jej konfigurację $\widetilde{K}$. Obie konfiguracje będą miały te same stany. Jeżeli w konfiguracji K głowice taśm roboczych znajdują się odpowiednio nad m-tą i n-tą komórką, to konfiguracja $\widetilde{K}$ będzie kodować liczbę $2^m \cdot 3^n$. Nietrudno zauważyć, że znając konfigurację ze stanem maszyny M, kodującą odpowiednią liczbę, możemy z niej odtworzyć pewną konfigurację maszyny M.

Przypuśćmy, że mamy konfigurację K_1 maszyny M, dla której jest już zdefiniowana konfiguracja $\widetilde{K_1}$, oraz konfigurację K_2 maszyny M taką, że $K_1 \to K_2$. W takiej sytuacji zdefiniujemy fragment programu $f(P)$ oraz konfigurację $\tilde{K}_2$ taką, że $\widetilde{K_1} \to^* \widetilde{K_2}$.

3.1 Symulacja poszczególnych rozkazów maszyny Minsky'ego

Przypuśćmy, że mamy już konfiguracje K_1, K_2 oraz $\widetilde{K_1}$ takie, jak wyżej. Konfigurację K_2 otrzymujemy w wyniku wykonania jednego rozkazu w konfiguracji K_1. Przyjmijmy, że tym rozkazem jest

$$q_i\ 1\ \$ \to \langle \text{w prawo} \rangle\ q_j.$$

Odpowiadającą temu rozkazowi zmianę konfiguracji $\widetilde{K_1}$ może uzyskać pies działający zgodnie z następującym programem.

$q\ B \longrightarrow \langle \text{dół} \rangle\ s_{1,1}$	dalej $i = 0, 1$
$s_{1,i}\ Z \longrightarrow \langle \text{prawo} \rangle,\ \langle \text{lewo} \rangle,\ \langle \text{dół} \rangle\ s_{1,i+1}$	dodajemy mod 2
$s_{1,i}\ B \longrightarrow \langle \text{góra} \rangle\ s_{2,i}$	
$s_{2,i}\ Z \longrightarrow \langle \text{góra} \rangle\ s_{2,i}$	
$s_{2,i}\ B \longrightarrow \langle \text{dół} \rangle\ s_{3,i}$	
$s_{3,1}\ Z \longrightarrow \langle \text{lewo} \rangle\ q_{pusty}$	
$s_{3,0}\ Z \longrightarrow \langle \text{lewo} \rangle\ q_{niepusty}$	

Symulacja rozkazu

$$q_i\ t\ B \to \langle \text{w lewo} \rangle\ q_j.$$

$q\ B \longrightarrow \langle \text{prawo} \rangle,\ \langle \text{dół} \rangle\ s_1$	
$s_1\ Z \longrightarrow \langle \text{dół} \rangle\ s_1$	
$s_1\ B \longrightarrow \langle \text{dół} \rangle,\ \langle \text{lewo} \rangle\ s_2$	
$s_2\ Z \longrightarrow \langle \text{prawo} \rangle\ s_3$	
$s_3\ Z \longrightarrow \langle \text{góra} \rangle\ s_3$	
$s_3\ B \longrightarrow \langle \text{dół} \rangle\ s_1$	
$s_2\ B \longrightarrow \langle \text{góra} \rangle\ s_4$	koniec liczenia
$s_4\ Z \longrightarrow \langle \text{góra} \rangle\ s_4$	
$s_4\ B \longrightarrow \langle \text{prawo} \rangle\ s_5$	pocz. kopiowania wyniku
$s_5\ Z \longrightarrow \langle \text{prawo} \rangle,\ \langle \text{lewo} \rangle,\ \langle \text{góra} \rangle\ s_5$	
$s_5\ B \longrightarrow \langle \text{prawo} \rangle\ q_{koniec}$	

Symulacja rozkazu

$$q_i\ t\ B \to \langle \text{w prawo} \rangle\ q_j,$$

$q\ B \longrightarrow \langle \text{dół} \rangle\ s_1$

$s_1\ B \longrightarrow \langle \text{prawo} \rangle\ s_5$ n=1

$s_1\ Z \longrightarrow \langle \text{prawo} \rangle,\ \langle \text{dół} \rangle\ s_2$

$s_2\ Z \longrightarrow \langle \text{dół} \rangle\ s_2$

$s_2\ B \longrightarrow \langle \text{lewo} \rangle\ s_3$

$s_3\ Z \longrightarrow \langle \text{prawo} \rangle\ s_4$

$s_4\ Z \longrightarrow \langle \text{góra} \rangle\ s_4$

$s_4\ B \longrightarrow \langle \text{dół} \rangle\ s_2$

$s_3\ B \longrightarrow \langle \text{prawo} \rangle\ s_5$ koniec liczenia

$s_5\ Z \longrightarrow \langle \text{góra} \rangle\ s_5$

$s_5\ B \longrightarrow \langle \text{góra} \rangle\ q_{koniec}$