Algorytmy sztucznej inteligencji

dr Radosław Matusik

radoslaw.matusik@wmii.uni.lodz.pl

Przeszukiwanie z ograniczeniami

Zagadnienie przeszukiwania z ograniczeniami stanowi grupę problemów przeszukiwania w przestrzeni stanów, które składa się ze:

1. skończonego zbioru zmiennych;
2. skończonego zbioru odpowiadających tym zmiennym wartości, zwanych dziedziną zmiennej;
3. skończonego zbioru ograniczeń.

Rozwiązaniem problemu z ograniczeniami jest każda kombinacja wartości zmiennych, która spełnia wszystkie nałożone ograniczenia.

Sudoku

Zagadnienie przeszukiwania z ograniczeniami zrealizujemy na przykładzie popularnej gry, jaką jest sudoku. Gra toczy się na planszy o wymiarze 9 na 9 kwadratów, podzielonego na 9 mniejszych (każdy z nich o wymiarze 3 na 3). Zadanie polega na wypełnieniu każdego z mniejszych kwadratów cyframi od 1 do 9 w taki sposób, aby w każdym wierszu i każdej kolumnie dużego kwadratu oraz w każdym wierszu i każdej kolumnie małego kwadratu każda cyfra wystąpiła jednokrotnie. Na wstępie gry duży kwadrat wypełnia się minimum 17 cyframi. Tyle bowiem potrzeba do znalezienia unikalnego rozwiązania. Z reguły wypełnia się $24 - 30$ pól.

	9		6			7		1
2					3		8	4
7		3						
	3			6	1			
6								8
			9	4			7	
						5		2
1	5		3					9
9		6			2		1	

4	9	8	6	2	5	7	3	1
2	6	5	7	1	3	9	8	4
7	1	3	8	9	4	2	5	6
8	3	7	2	6	1	4	9	5
6	4	9	5	3	7	1	2	8
5	2	1	9	4	8	6	7	3
3	7	4	1	8	9	5	6	2
1	5	2	3	7	6	8	4	9
9	8	6	4	5	2	3	1	7

Algorytm

Do realizacji zagadnienia wykorzystamy mało efektywny, ale prosty w implementacji algorytm przeszukiwania przyrostowego z powracaniem. Innymi algorytmami są m.in.: sprawdzanie wprzód, wybór zmiennej najbardziej ograniczonej, wybór zmiennej najbardziej ograniczającej, czy wybór zmiennej najmniej ograniczającej.

Przebieg algorytmu jest następujący:

1. Wybieramy pierwszą wolną komórkę na planszy. Jeśli ją znajdujemy, to przechodzimy do kroku 2. Jeśli nie, to przechodzimy do kroku 6.
2. Wybieramy liczbę do wstawienia. Jeśli istnieje, to przechodzimy do kroku 3. Jeśli nie istnieje, to przechodzimy do kroku 5.
3. Jeśli jest liczba, która nie powoduje konfliktu, to przechodzimy do kroku 4. Jeśli nie ma, to ją usuwamy i przechodzimy do kroku 2.
4. Wstawiamy liczbę i przechodzimy do kroku 1.
5. Usuwamy uprzednio wstawioną liczbę i przechodzimy do kroku 2.
6. Zwracamy "sukces" i kończymy działanie algorytmu.

Przykład

Krok algorytmu: 1

	9		6			7		1
2					3		8	4
7		3						
	3			6	1			
6								8
			9	4			7	
						5		2
1	5		3					9
9		6			2		1	

Krok algorytmu: 2
Wybieramy pierwszą cyfrę z dostępnych do wstawienia: 1, 2, 3, 4, 5, 6, 7, 8, 9.

Krok algorytmu: 3

1	9		6			7		1
2					3		8	4
7		3						
	3			6	1			
6								8
			9	4			7	
						5		2
1	5		3					9
9		6			2		1	

Konflikt!

	9		6			7		1
2					3		8	4
7		3						
	3			6	1			
6								8
			9	4			7	
						5		2
1	5		3					9
9		6			2		1	

Krok algorytmu: 2
Wybieramy pierwszą cyfrę z dostępnych do wstawienia: 2, 3, 4, 5, 6, 7, 8, 9.

Krok algorytmu: 3

2	9		6			7		1
2					3		8	4
7		3						
	3			6	1			
6								8
			9	4			7	
						5		2
1	5		3					9
9		6			2		1	

Konflikt!

	9		6			7		1
2					3		8	4
7		3						
	3			6	1			
6								8
			9	4			7	
						5		2
1	5		3					9
9		6			2		1	

Krok algorytmu: 2
Wybieramy pierwszą cyfrę z dostępnych do wstawienia: 3, 4, 5, 6, 7, 8, 9.

Krok algorytmu: 3

3	9		6			7		1
2					3		8	4
7		3						
	3			6	1			
6								8
			9	4			7	
						5		2
1	5		3					9
9		6			2		1	

Konflikt!

	9		6			7		1
2					3		8	4
7		3						
	3			6	1			
6								8
			9	4			7	
						5		2
1	5		3					9
9		6			2		1	

Krok algorytmu: 2
Wybieramy pierwszą cyfrę z dostępnych do wstawienia: 4, 5, 6, 7, 8, 9.

Krok algorytmu: 3

4	9		6			7		1
2					3		8	4
7		3						
	3			6	1			
6								8
			9	4			7	
						5		2
1	5		3					9
9		6			2		1	

Brak konfliktu.

Krok algorytmu: 4

4	9		6			7		1
2					3		8	4
7		3						
	3			6	1			
6								8
			9	4			7	
						5		2
1	5		3					9
9		6			2		1	

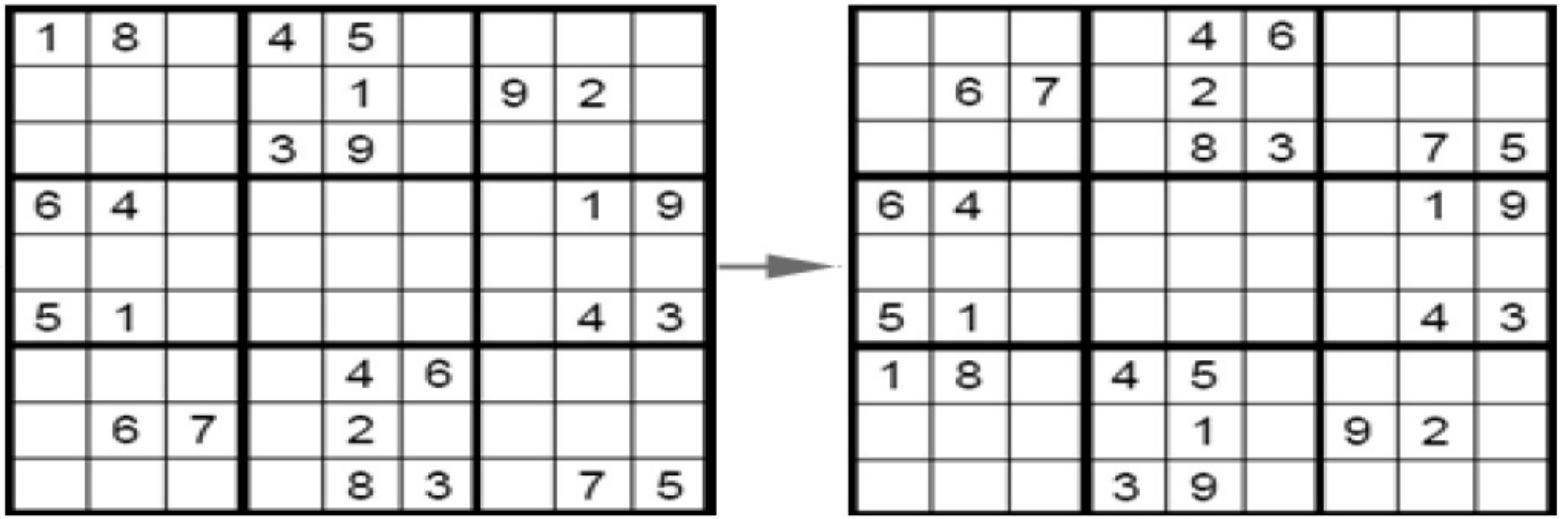
Krok algorytmu: 1

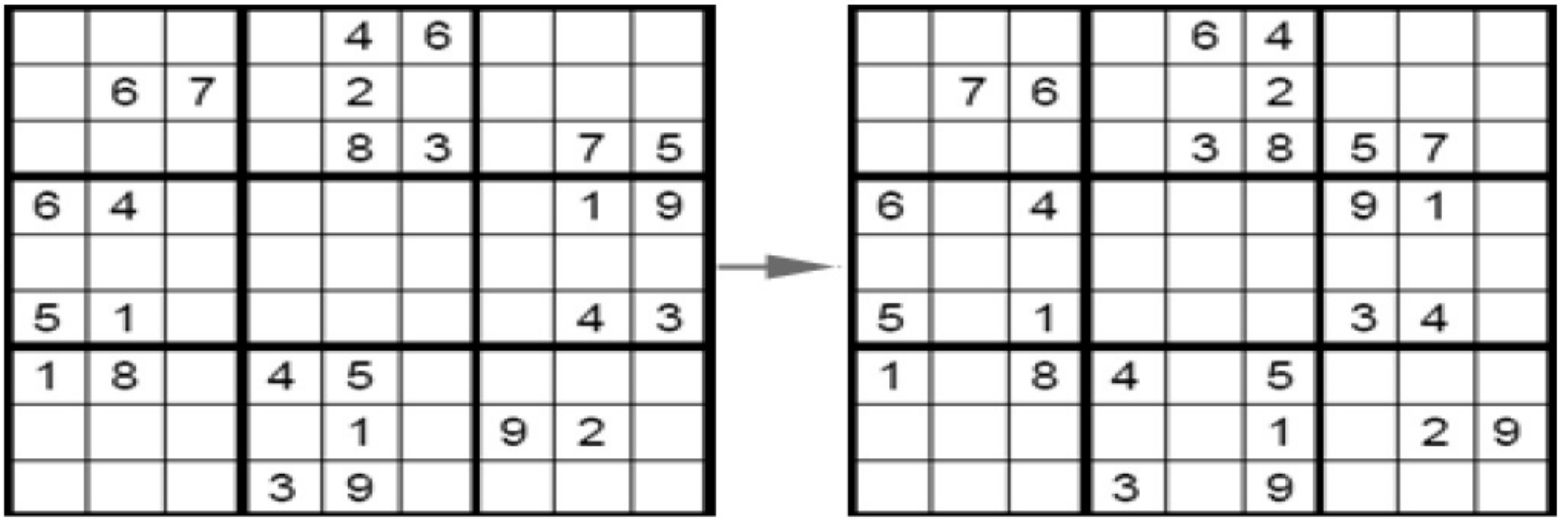
4	9		6			7		1
2					3		8	4
7		3						
	3			6	1			
6								8
			9	4			7	
						5		2
1	5		3					9
9		6			2		1	

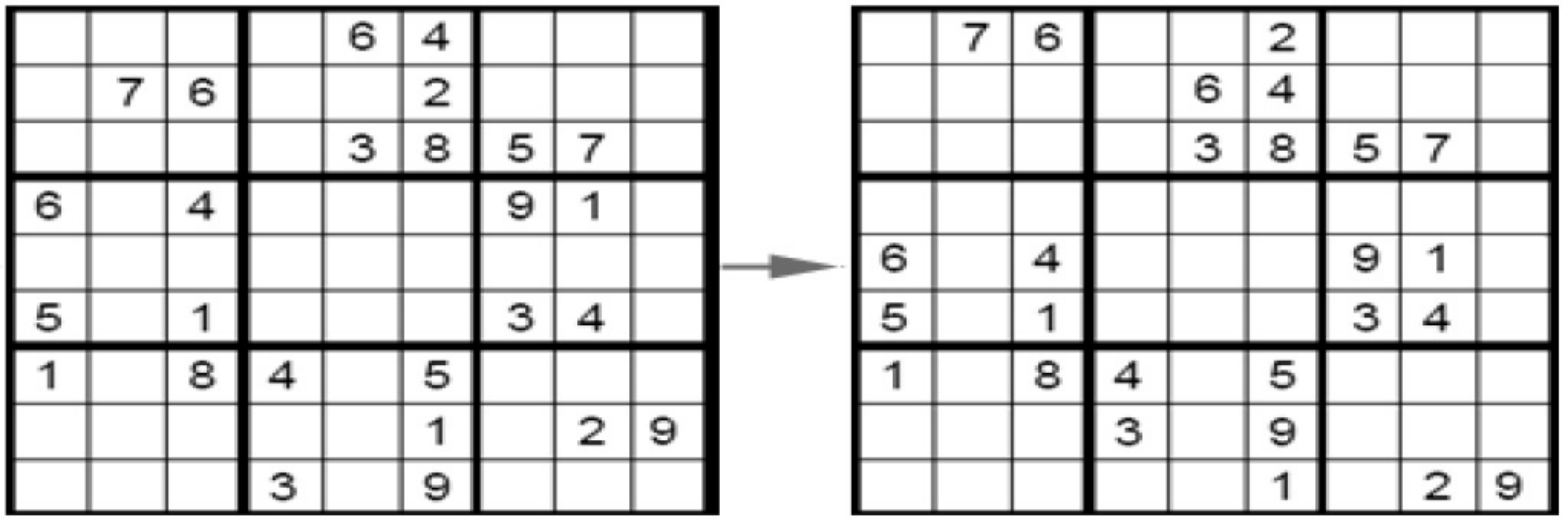
Wykonując kolejne kroki algorytmu po kilkunastu(?), kilku tysiącach(?), kilkudziesięciu tysiącach(?) kroków rozwiążemy problem przeszukiwania z ograniczeniami.

Generowanie planszy

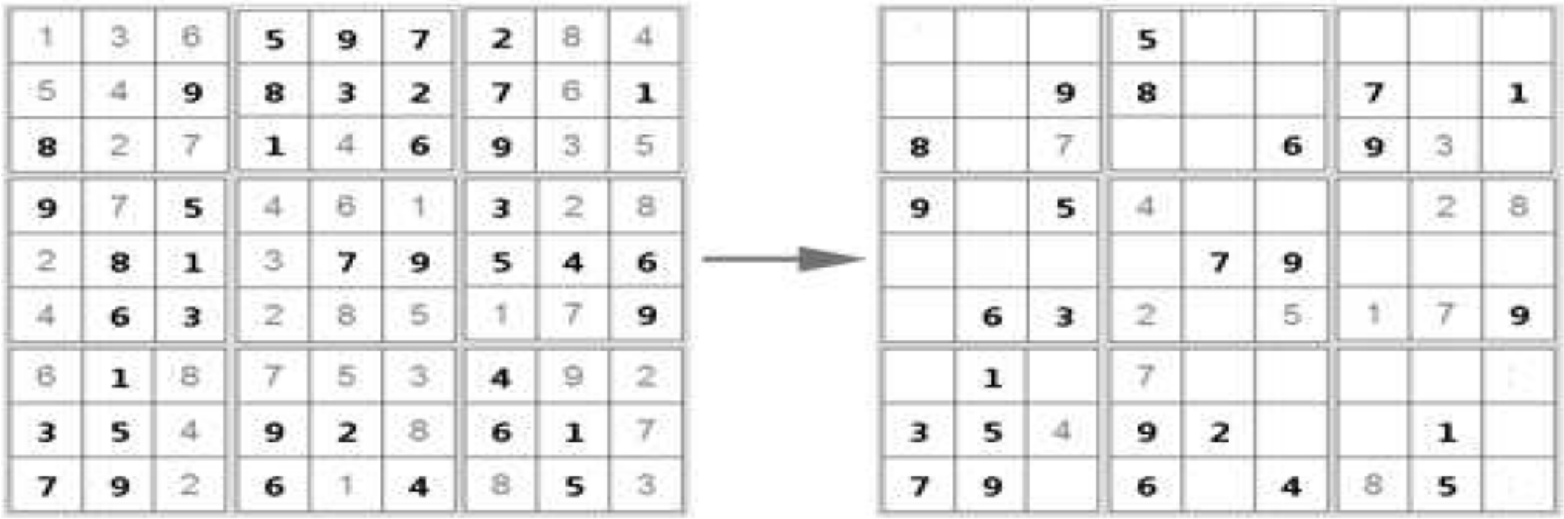
- Losowanie planszy;
- Permutowanie kolumn / wierszy




				4	6			
	6	7		2				
				8	3		7	5
6	4						1	9
5	1						4	3
1	8		4	5				
				1		9	2	
			3	9				
				6	4			
	7	6			2			
				3	8	5	7	
6		4				9	1	
5		1				3	4	
1		8	4		5			
					1		2	9
			3		9			


6 4
7 6 2
3 8 5 7
6 4 9 1
5 1 3 4
1 8 4 5
1 2 9
3 9
7 6 2
6 4
3 8 5 7
6 4 9 1
5 1 3 4
1 8 4 5
3 9
1 2 9

- Usuwanie pól



Algorytm

1. Sprawdzamy, czy liczba wypełnionych pól jest większa od oczekiwanej. Jeśli tak, przechodzimy do punktu 2; jeśl nie, przechodzimy do punktu 5.
2. Losujemy pole.
3. Usuwamy wylosowane pole.
4. Sprawdzamy, czy istnieje dokładnie jedno rozwiązanie. Jeśli tak, to przechodzimy do punktu 1; jeśli nie, to przywracamy usnięte pole i przechodzimy do punktu 1.
5. Koniec algorytmu.

Malowanie liczbami

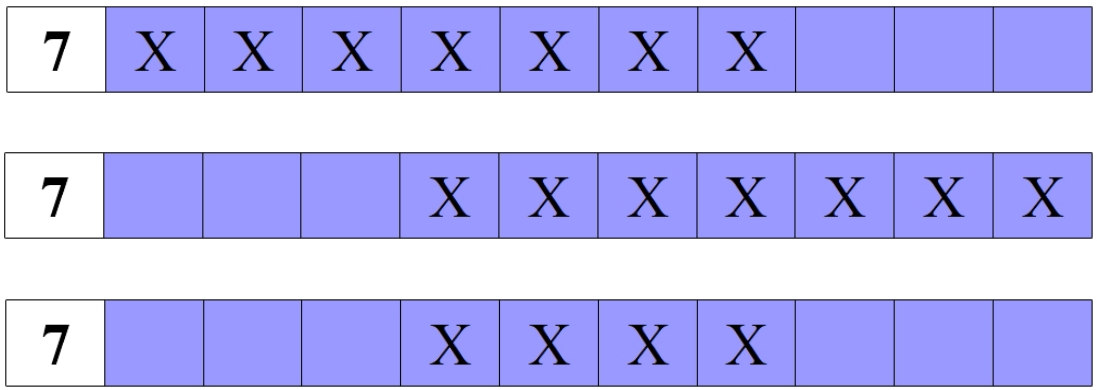
Malowanie liczbami jest łamigłówką, która polega na zamalowywaniu pól diagramu, które utworzą rysunek. Pola, które należy zamalować wskazują liczby obok diagramu. Na przykład zapis 7, 2, 10 oznacza, że w danym wierszu należy zamalować 7 kolejnych pól, następnie 2 i na końcu 10. Taki zapis oznacza, że między grupą zawierającą 7 pól zamalowanych, a grupą zawierającą 2 pola zamalowane, znajduje się przynajmniej jedno pole niezamalowane.

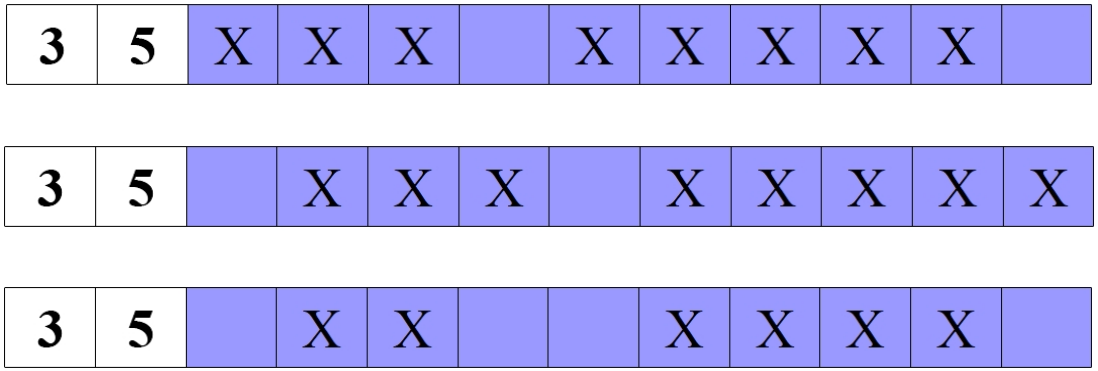
Algorytm

Podczas rozwiązywania problemu obrazków logicznych zaznaczanie pól pustych jest równie ważne, jak zaznaczanie pól zamalowanych. Proste obrazki można rozwiązywać rozpatrując jedynie dany wiersz lub kolumnę, natomiast rozpatrując bardziej skomplikowane należy brać pod uwagę więcej wierszy i kolumn.

Metoda "koniecznie pełne"

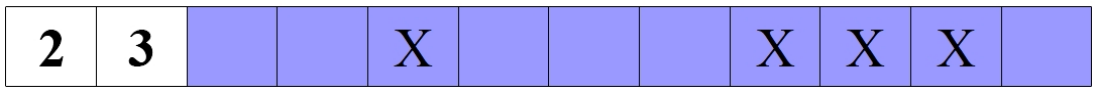
Najprostszą metodą rozwiązywania zadania logicznych obrazków jest "koniecznie pełne". Metody tej należy używać na samym początku do zaznaczania możliwie największej liczby pól. Polega ona na określeniu komórek, które muszą być zamalowane ze względu na ograniczone miejsce.



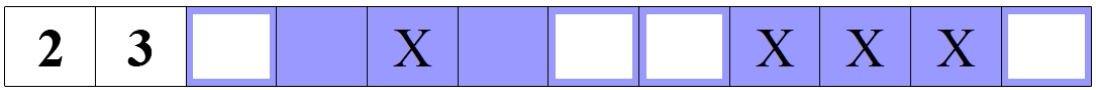

3 5 X X X X X X X X X
3 5 X X X X X X X X X
3 5 X X X X X X X X

Metoda "koniecznie puste"

Metoda "koniecznie puste" jest odwrotnością metody "koniecznie pełne". Polega na określaniu tych pól, które na pewno są puste (niezamalowane), ponieważ nie ma bloków, które mogłyby je pokryć.
Załóżmy, że mamy następujący układ na diagramie:

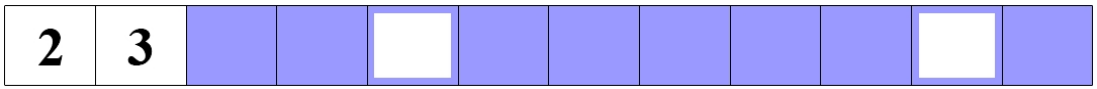


Wówczas na pewno puste będą pola:

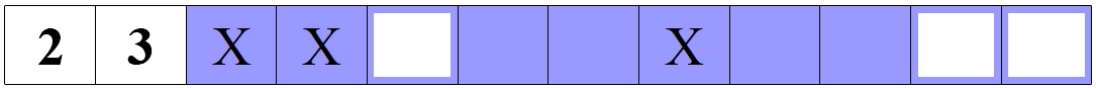


Metoda "wciskanie"

Metoda "wciskanie" polega na dopasowywaniu całych bloków pól w miejsca, w których na pewno muszą się znaleźć, gdyż nie da się ułożyć ich inaczej. W tym przypadku pola niezamalowane wymuszą położenie całego bloku.
Załóżmy, że mamy następujący układ na diagramie:



Wówczas:



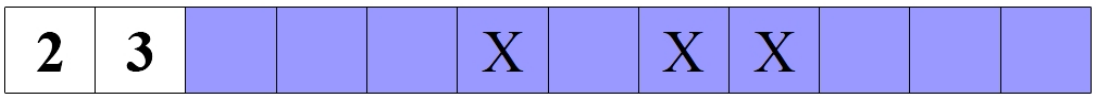
Metoda "klejenie"

Metoda "klejenie" polega na uzupełnianiu pól, gdy odległość pełnego pola od brzegu jest mniejsza, niż długość pierwszego (lub też ostatniego) bloku.
Załóżmy, że mamy następujący układ na diagramie:

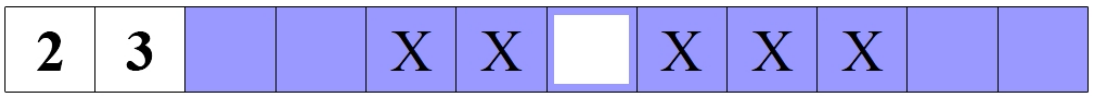
Wówczas:

Metoda "łączenie i dzielenie"

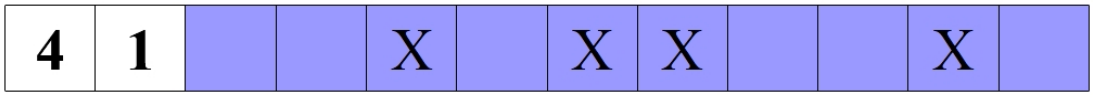
Metoda "łączenie i dzielenie" polega na łączeniu pól, które położone są obok siebie (gdy mamy pewność, że należą do jednego bloku) lub ich rozdzielaniu (gdy na pewno należą do innych bloków).
Załóżmy, że mamy następujący układ na diagramie:



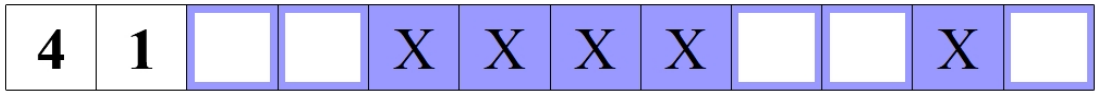
Wówczas:



Załóżmy, że mamy następujący układ na diagramie:



Wówczas:



Metoda "rtęć"

Metoda o nazwie "rtęć" jest odmianą "koniecznie puste". Nazwa pochodzi od zachowania rtęci, która w naturalny sposób odpycha się od krawędzi zbiornika. Jeśli zamalowane pole odsunięte jest od brzegu o tyle pól, jaką długość ma pierwszy blok, to pierwsze pole musi być puste. Dzieje się tak oczywiście dlatego, ponieważ blok nie może być dosunięty maksymalnie, gdyż byłby zbyt długi. Załóżmy, że mamy następujący układ na diagramie:

Wówczas układ na diagramie będzie następujący:

Metoda nie wprost

W przypadku bardziej skomplikowanych obrazków powyższe metody mogą okazać się nieskuteczne. Trzeba wówczas zastosować metodę wnioskowania nie wprost. Podejście to obejmuje wnioskowanie o większej liczbie kolumn i wierszy. Metoda ta zakłada, że dane pole musi być zamalowane, ponieważ gdyby było puste, to w innych polach wystąpiłby konflikt.

Załóżmy, że mamy do rozpatrzenia następujący układ:

	4	2	1	1	1	2	2	2	2	2
4										
3										

Załóżmy na początek, że pierwsze pola w pierwszym i drugim wierszu są zamalowane. Wówczas:

- w pierwszym rzędzie pola 2, 3 i 4 są zamalowane;
- w drugim rzędzie pola 2 i 3 są zamalowane.

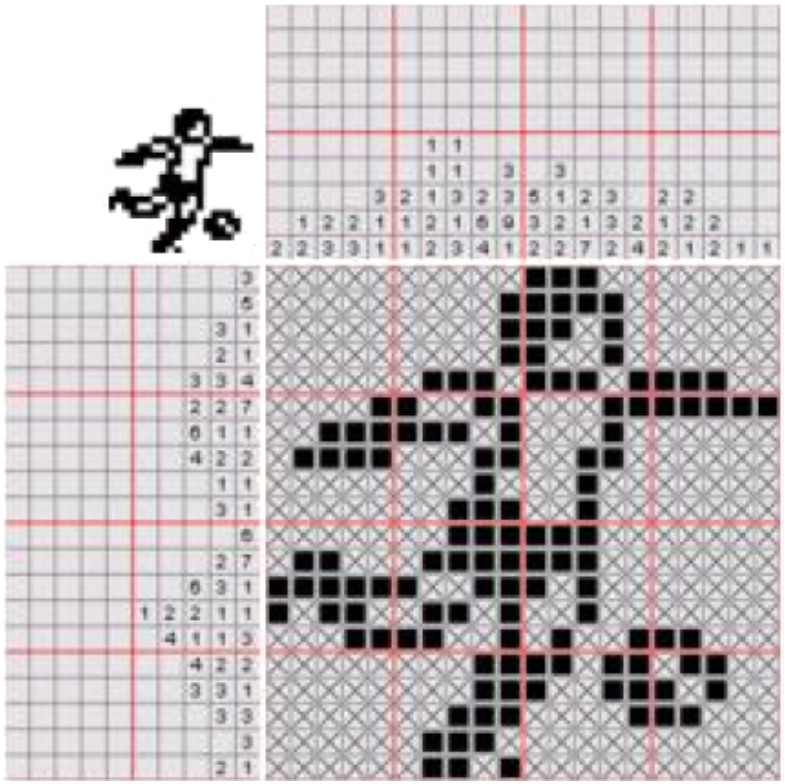
Wówczas:

	4	2	1	1	1	2	2	2	2	2
4	X	X	X	X						
3	X	X	X							

W ten sposób można analizować większą liczbę wierszy i kolumn. Problemem w wykorzystaniu tej metody jest fakt, iż nie wiadomo od którego pola powinniśmy zacząć. Analizę w tym przypadku najlepiej zacząć od pól, które:

- mają wiele zamalowanych pól w swoim otoczeniu;
- są blisko brzegów lub są blisko dużych bloków pustych pól.

Przykład



Automat Moore'a

Automat skończony jest abstrakcyjnym, matematycznym, iteracyjnym modelem zachowania dynamicznego, który składa się z:

- skończonej liczby stanów;
- funkcji definiującej przejścia między stanami;
- akcji.

Zasada działania

Działanie automatu skończonego można przedstawić następująco:

1. Automat wczytuje podane słowo w oparciu o dany alfabet.
2. Każdy wczytany symbol powoduje, że - zgodnie z funkcją przejścia - następuje przejście z bieżącego stanu do stanu innego i wykonanie pewnej akcji.
3. Automat może przyjmować skończoną liczbę różnych stanów.
4. Automat akceptuje dane słowo, gy po jego wczytaniu znajdzie się w jednym ze stanów końcowych.

Model matematyczny

Automat Moore'a definiujemy jako uporządkowaną szóstkę:

$$\{Q, \Sigma, \Delta, \delta, \lambda, q_0\},$$

gdzie

- $Q = \{q_1, q_2, \ldots, q_n\}$ - skończony zbiór n stanów
- Σ - skończony zbiór symboli wejściowych, zwany alfabetem
- Δ - alfabet wyjściowy
- δ - funkcja przejścia odwzorowująca zbiór $Q \times \Sigma$ w Q, czyli taka, która każdemu stanowi $q \in Q$ i każdemu symbolowi wejściowemu $a \in \Sigma$ przyporządkowuje nowy stan
- λ - funkcja zwracająca wyjście związane z każdym ze stanów, czyli odwzorowująca zbiór Q w Δ
- q_0 - stan początkowy

Przykład

Zasadę działania automatu Moore'a omówimy na podstawie symulacji życia czterech osobników. Każdy z osobników ma przypisaną liczbę punktów życia, która zmniejsza się wraz z wykonaniem ruchu przez danego osobnika. Na planszy znajdują się specjalne punkty z pożywieniem. Każdy z tych punktów ma określoną wartość, określającą ilość pożywienia. Jeżeli osobnik znajdzie się na tym polu, to uzupełnia swoje punkty życia, a także zapamiętuje jego położenie oraz ilość pożywienia. Liczba określająca ilość pożywienia w danym punkcie nie jest odnawialna, tzn. jest pomniejszana, aż osiągnie wartość zero.

Jeżeli dany osobnik znajdzie się na polu, które już zajmuje inny osobnik, to wówczas następuje rozmowa, podczas której osobniki przekazują sobie informacje o punktach z pożywieniem. Osobnik zapamiętuje uzyskane informacje lub aktualizuje posiadane przez siebie. Może się bowiem zdarzyć, że dwa osobniki posiadają informacje o tym samym punkcie z pożywieniem. Wówczas aktualną informacją jest ta, która mówi o mniejszej ilości pożywienia w danym punkcie.

Oczywiście najbardziej korzystną jest sytuacja, gdy dany osobnik ma możliwie największą liczbę punktów. Jeśli liczba ta osiąga wartość minimalną, to wówczas dany osobnik powinien udać się do punktu z pożywieniem i uzupełnić punkty życia. Dokonuje tego za pomocą posiadanych przez siebie informacji. Jeśli dany osobnik posiada informacje o kilku punktach pożywienia, to powinien udać się do najkorzystniejszego pod względem odległości oraz możliwych do uzyskania punktów życia. Osobnik umiera, gdy liczba punktów życia będzie wynosiła zero.

Skonstruujmy automat Moore'a dla naszego przykładu:
Zbiór stanów określamy następująco:
$Q = \{\text{patrol}, \text{rozmowa}, \text{poszukiwanie}, \text{jedzenie}, \text{śmierć}\}$, gdzie

- patrol - stan początkowy polegający na losowym poruszaniu się osobnika po planszy
- rozmowa - stan, w którym następuje wymiana danych dotycząca położenia na planszy punktów z pożywieniem oraz ilości pożywienia w danym punkcie
- poszukiwanie - stan, w którym dany osobnik na podstawie zgromadzonych informacji podąża do punktu z pożywieniem
- jedzenie - stan, w którym osobnik uzupełnia punkty życia
- śmierć - stan, w którym osobnik znika z planszy

Alfabet wejściowy będzie zbiorem komunikatów postaci:

- k_1 - wyczerpany zasób punktów życia (0)
- k_2 - mała liczba punktów życia $(1 - 499)$
- k_3 - duża liczba punktów życia $(500 - 999)$
- k_4 - maksymalna liczba punktów życia (1000)
- k_5 - rozmowa
- k_6 - koniec rozmowy
- k_7 - odnaleziono punkt z pożywieniem
- k_8 - wyczerpanie zasobów pożywienia

Funkcję przejścia określimy następująco:
$\delta : (biezacy\_stan, komunikat\_wejsciowy) \longrightarrow (nowy\_stan)$.
W naszym przypadku funkcja przejścia wygląda następująco:

- $(patrol, k_2) \longrightarrow$

Funkcję przejścia określimy następująco:
$\delta : (biezacy\_stan, komunikat\_wejsciowy) \rightarrow (nowy\_stan)$.
W naszym przypadku funkcja przejścia wygląda następująco:

- $(patrol, k_2) \rightarrow$ (poszukiwanie)

Funkcję przejścia określimy następująco:
$\delta : (biezacy\_stan, komunikat\_wejsciowy) \longrightarrow (nowy\_stan)$.
W naszym przypadku funkcja przejścia wygląda następująco:

- $(patrol, k_2) \longrightarrow$ (poszukiwanie)
- $(patrol, k_3) \longrightarrow$

Funkcję przejścia określimy następująco:
$\delta : (biezacy\_stan, komunikat\_wejsciowy) \rightarrow (nowy\_stan)$.
W naszym przypadku funkcja przejścia wygląda następująco:

- $(patrol, k_2) \rightarrow$ (poszukiwanie)
- $(patrol, k_3) \rightarrow$ (patrol)

Funkcję przejścia określimy następująco:
$\delta : (biezacy\_stan, komunikat\_wejsciowy) \longrightarrow (nowy\_stan)$.
W naszym przypadku funkcja przejścia wygląda następująco:

- $(patrol, k_2) \longrightarrow$ (poszukiwanie)
- $(patrol, k_3) \longrightarrow$ (patrol)
- $(rozmowa, k_6 \wedge k_3) \longrightarrow$

Funkcję przejścia określimy następująco:
$\delta : (biezacy\_stan, komunikat\_wejsciowy) \rightarrow (nowy\_stan)$.
W naszym przypadku funkcja przejścia wygląda następująco:

- $(patrol, k_2) \rightarrow$ (poszukiwanie)
- $(patrol, k_3) \rightarrow$ (patrol)
- $(rozmowa, k_6 \wedge k_3) \rightarrow$ (patrol)

Funkcję przejścia określimy następująco:
$\delta : (biezacy\_stan, komunikat\_wejsciowy) \longrightarrow (nowy\_stan)$.
W naszym przypadku funkcja przejścia wygląda następująco:

- $(patrol, k_2) \longrightarrow$ (poszukiwanie)
- $(patrol, k_3) \longrightarrow$ (patrol)
- $(rozmowa, k_6 \wedge k_3) \longrightarrow$ (patrol)
- $(rozmowa, k_6 \wedge k_2) \longrightarrow$

Funkcję przejścia określimy następująco:
$\delta : (biezacy\_stan, komunikat\_wejsciowy) \longrightarrow (nowy\_stan)$.
W naszym przypadku funkcja przejścia wygląda następująco:

- $(patrol, k_2) \longrightarrow$ (poszukiwanie)
- $(patrol, k_3) \longrightarrow$ (patrol)
- $(rozmowa, k_6 \wedge k_3) \longrightarrow$ (patrol)
- $(rozmowa, k_6 \wedge k_2) \longrightarrow$ (poszukiwanie)

Funkcję przejścia określimy następująco:
$\delta : (biezacy\_stan, komunikat\_wejsciowy) \rightarrow (nowy\_stan)$.
W naszym przypadku funkcja przejścia wygląda następująco:

- $(patrol, k_2) \rightarrow$ (poszukiwanie)
- $(patrol, k_3) \rightarrow$ (patrol)
- $(rozmowa, k_6 \wedge k_3) \rightarrow$ (patrol)
- $(rozmowa, k_6 \wedge k_2) \rightarrow$ (poszukiwanie)
- $(poszukiwanie, k_7) \rightarrow$

Funkcję przejścia określimy następująco:
$\delta : (biezacy\_stan, komunikat\_wejsciowy) \rightarrow (nowy\_stan)$.
W naszym przypadku funkcja przejścia wygląda następująco:

- $(patrol, k_2) \rightarrow$ (poszukiwanie)
- $(patrol, k_3) \rightarrow$ (patrol)
- $(rozmowa, k_6 \wedge k_3) \rightarrow$ (patrol)
- $(rozmowa, k_6 \wedge k_2) \rightarrow$ (poszukiwanie)
- $(poszukiwanie, k_7) \rightarrow$ (jedzenie)

Funkcję przejścia określimy następująco:
$\delta : (biezacy\_stan, komunikat\_wejsciowy) \longrightarrow (nowy\_stan)$.
W naszym przypadku funkcja przejścia wygląda następująco:

- $(patrol, k_2) \longrightarrow$ (poszukiwanie)
- $(patrol, k_3) \longrightarrow$ (patrol)
- $(rozmowa, k_6 \wedge k_3) \longrightarrow$ (patrol)
- $(rozmowa, k_6 \wedge k_2) \longrightarrow$ (poszukiwanie)
- $(poszukiwanie, k_7) \longrightarrow$ (jedzenie)
- $(poszukiwanie, k_1) \longrightarrow$

Funkcję przejścia określimy następująco:
$\delta : (biezacy\_stan, komunikat\_wejsciowy) \rightarrow (nowy\_stan)$.
W naszym przypadku funkcja przejścia wygląda następująco:

- $(patrol, k_2) \rightarrow$ (poszukiwanie)
- $(patrol, k_3) \rightarrow$ (patrol)
- $(rozmowa, k_6 \wedge k_3) \rightarrow$ (patrol)
- $(rozmowa, k_6 \wedge k_2) \rightarrow$ (poszukiwanie)
- $(poszukiwanie, k_7) \rightarrow$ (jedzenie)
- $(poszukiwanie, k_1) \rightarrow$ (śmierć)

Funkcję przejścia określimy następująco:
$\delta : (biezacy\_stan, komunikat\_wejsciowy) \rightarrow (nowy\_stan)$.
W naszym przypadku funkcja przejścia wygląda następująco:

- $(patrol, k_2) \rightarrow$ (poszukiwanie)
- $(patrol, k_3) \rightarrow$ (patrol)
- $(rozmowa, k_6 \wedge k_3) \rightarrow$ (patrol)
- $(rozmowa, k_6 \wedge k_2) \rightarrow$ (poszukiwanie)
- $(poszukiwanie, k_7) \rightarrow$ (jedzenie)
- $(poszukiwanie, k_1) \rightarrow$ (śmierć)
- $(jedzenie, k_2 \wedge k_8) \rightarrow$

Funkcję przejścia określimy następująco:
$\delta : (biezacy\_stan, komunikat\_wejsciowy) \rightarrow (nowy\_stan)$.
W naszym przypadku funkcja przejścia wygląda następująco:

- $(patrol, k_2) \rightarrow$ (poszukiwanie)
- $(patrol, k_3) \rightarrow$ (patrol)
- $(rozmowa, k_6 \wedge k_3) \rightarrow$ (patrol)
- $(rozmowa, k_6 \wedge k_2) \rightarrow$ (poszukiwanie)
- $(poszukiwanie, k_7) \rightarrow$ (jedzenie)
- $(poszukiwanie, k_1) \rightarrow$ (śmierć)
- $(jedzenie, k_2 \wedge k_8) \rightarrow$ (poszukiwanie)

Funkcję przejścia określimy następująco:
$\delta : (biezacy\_stan, komunikat\_wejsciowy) \rightarrow (nowy\_stan)$.
W naszym przypadku funkcja przejścia wygląda następująco:

- $(patrol, k_2) \rightarrow$ (poszukiwanie)
- $(patrol, k_3) \rightarrow$ (patrol)
- $(rozmowa, k_6 \wedge k_3) \rightarrow$ (patrol)
- $(rozmowa, k_6 \wedge k_2) \rightarrow$ (poszukiwanie)
- $(poszukiwanie, k_7) \rightarrow$ (jedzenie)
- $(poszukiwanie, k_1) \rightarrow$ (śmierć)
- $(jedzenie, k_2 \wedge k_8) \rightarrow$ (poszukiwanie)
- $(jedzenie, k_3 \wedge k_8) \rightarrow$

Funkcję przejścia określimy następująco:
$\delta : (biezacy\_stan, komunikat\_wejsciowy) \rightarrow (nowy\_stan)$.
W naszym przypadku funkcja przejścia wygląda następująco:

- $(patrol, k_2) \rightarrow$ (poszukiwanie)
- $(patrol, k_3) \rightarrow$ (patrol)
- $(rozmowa, k_6 \wedge k_3) \rightarrow$ (patrol)
- $(rozmowa, k_6 \wedge k_2) \rightarrow$ (poszukiwanie)
- $(poszukiwanie, k_7) \rightarrow$ (jedzenie)
- $(poszukiwanie, k_1) \rightarrow$ (śmierć)
- $(jedzenie, k_2 \wedge k_8) \rightarrow$ (poszukiwanie)
- $(jedzenie, k_3 \wedge k_8) \rightarrow$ (patrol)