Sztuczna Inteligencja

Soma Dutta

Wydział Matematyki i Informatyki, UWM w Olsztynie
soma.dutta@matman.uwm.edu.pl

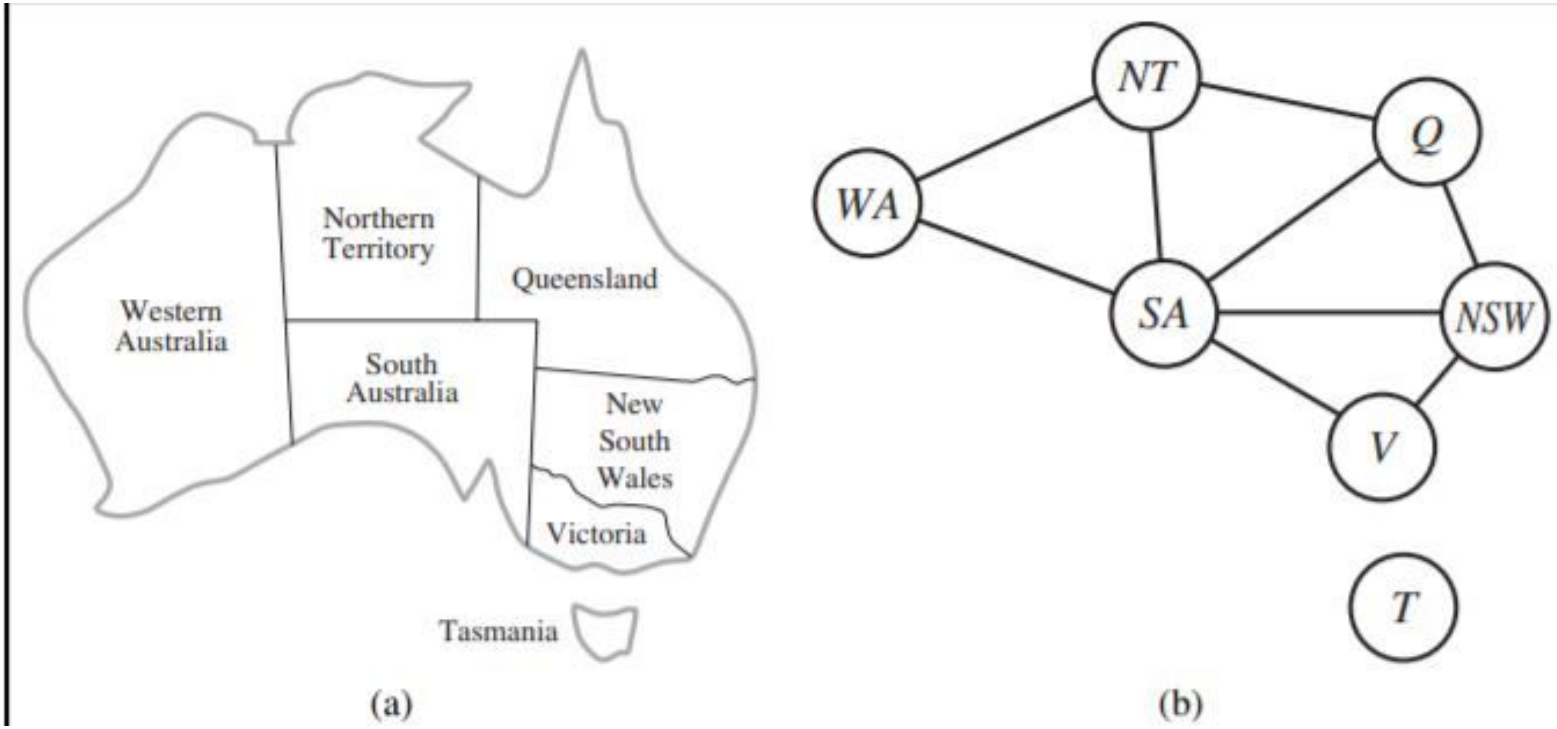
Wykład - 5: Algorytmy poszukujące rozwiązań spełniających więzy

i Zbiory przybliżone

Semestr letni 2022

CSP-Graf: mapa Australii

- Jak możemy znaleźć rozwiązanie, biorąc pod uwagę CSP przy użyciu agenta oprogramowania?
- Pierwszym krokiem jest to, że możemy przedstawić CSP jako graf.



- Węzły grafu odpowiadają zmiennym CSP
- Krawędź łączy dwie zmienne, jeśli uczestniczą w ograniczeniu

Sprawdzanie lokalnej spójności

- Aby stworzyć algorytm przeszukiwania rozwiązania dla CSP, oprócz grafu reprezentującego przestrzeń stanu problemu musimy dodać metodę sprawdzania spójności lokalnej.
- W algorytmach CSP występują dwie części;
  (i) jedna polega na poszukiwaniu przypisania z kilku możliwych przypisań dla zmiennych lub
  (ii) za pomocą propagacji więzów algorytm ustala, które wartości zmiennej (połączonej z bieżącą zmienną) należy uznać za dopuszczalne (tj. które wartości należy wykluczyć z rozważania)
- Tak więc potrzebne jest sprawdzanie lokalnej spójności. Rozpoznając CSP jako graf, metoda sprawdzania lokalnej spójności wymusza eliminację niespójnych wartości na rozpatrywanej części grafu.

Plany

- Algorytmy i techniki dla problemów CSP
- Wyznaczanie reguł i ograniczeń z tablic danych z zastosowaniem zbiorów przybliżónych

Etapy w poszukiwaniu rozwiązania CSP

- Tworzenie grafu ograniczenia dla danego CSP
- Opracowanie algorytmu przeszukiwania, który przypisuje wartości kolejnym zmiennym, poprzez sprawdzenie jego lokalnej spójności
- Istnieją różne kryteria sprawdzania spójności, np.
  - Spójność węzła (Node consistency)
  - Spójność łuku (Arc consistency)
  - Spójność ścieżki (Path consistency)

Spójność węzła

- Zwykle ma to zastosowanie w przypadku, gdy ograniczenie jest unarne: na przykład w przypadku problemu z kolorowaniem mapy SA $\neq$ green.
- Ponieważ dziedziną dla każdej zmiennej jest $\{red, green, blue\}$, algorytm może zacząć się od niej, ale po zastosowaniu spójności węzłów zmniejszy dziedzinę do $\{red, blue\}$.
- Definicja: Pojedyncza zmienna (odpowiadająca węzłowi na grafie CSP) jest spójna węzłowo, jeśli wszystkie wartości w dziedzinie zmiennej spełniają unarne ograniczenia zmiennej.

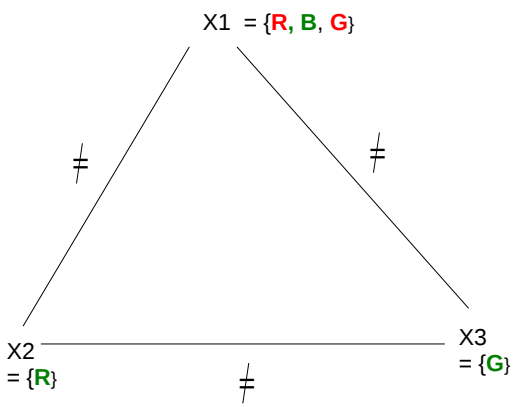
Spójność łuku

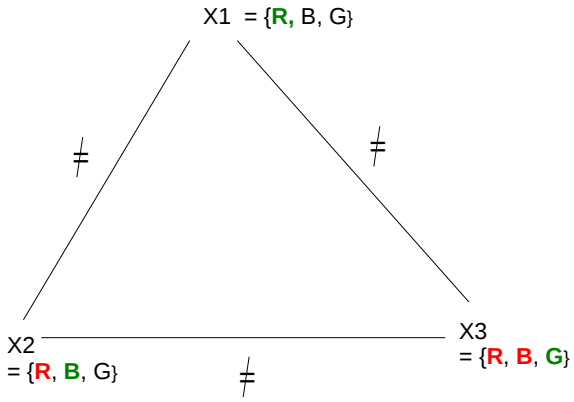
- Definicja: Węzeł (zmienna) X_i jest zgodny (arc consistent) z innym węzłem X_j, jeśli dla każdej wartości w dziedzinie D_i istnieje wartość w dziedzinie D_j, która spełnia binarne ograniczenie (X_i, X_j).
- Na przykład, rozważmy ograniczenie $Y = X^2$ gdzie $X, Y \in \{0, 1, \ldots, 9\}$. Możemy przedstawić ograniczenie jako $\langle (X, Y), \{(0, 0), (1, 1), (2, 4), (3, 9)\}\rangle$. Jeśli zastosujemy spójność łuku w algorytmie przeszukiwania, wówczas metoda zredukuje dziedzinę X do $\{0, 1, 2, 3\}$, a dziedzinę Y do $\{0, 1, 4, 9\}$.
- Jednak spójność łuku nie jest użyteczna w przypadku problemu kolorowania mapy Australii. Wiemy że, $SA \neq WA$ co można przedstawiać jako

  $\{(red, green), (red, blue), (green, red), (green, blue), (blue, red), (blue, green)\}$.

  Ponieważ dla każdej wartości SA istnieje wartość WA, spójność łuku nie zmniejszy dziedziny żadnej ze zmiennych.

Pseudokod dla spójności łuku: AC-3



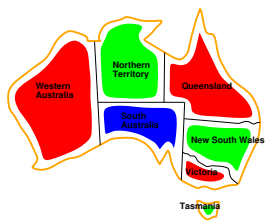



```
function AC-3(csp) returns false if an inconsistency is found and true otherwise
  inputs: csp, a binary CSP with components (X, D, C)
  local variables: queue, a queue of arcs, initially all the arcs in csp
  while queue is not empty do
    (X_i, X_j) ← REMOVE-FIRST(queue)
    if REVISE(csp, X_i, X_j) then
      if size of D_i = 0 then return false
      for each X_k in X_i.NEIGHBORS - {X_j} do
        add (X_k, X_i) to queue
  return true
```

```
function REVISE(csp, X_i, X_j) returns true iff we revise the domain of X_i
  revised ← false
  for each x in D_i do
    if no value y in D_j allows (x, y) to satisfy the constraint between X_i and X_j then
      delete x from D_i
      revised ← true
  return revised
```

Spójność ścieżki

- Rozważmy problem kolorowania mapy Australii dwoma kolorami: czerwonym i niebieskim. Jeśli zastosujemy tutaj spójność łuku, to okaże się, że każda zmienna jest spójna względem łuku, ponieważ dla każdej wartości przyjmowanej przez jedną zmienną inna wartość może być przypisana drugiej zmiennej
- Na pewno problem nie ma rozwiązania, jeśli weźmiemy pod uwagę dwa kolory. To że problem nie ma rozwiązania, można wykazać za pomocą spójności ścieżki.



- Definicja: Zbiór 2 zmiennych $\{X_i, X_j\}$ jest ścieżką spójną względem do innej zmiennej X_m, jeśli dla każdego spójnego przypisania $\{X_i = a, X_j = b\}$ istnieje przypisanie dla X_m, które spełnia ograniczenie $\{X_i, X_m\}$ i $\{X_m, X_j\}$.

Sprawdzanie lokalnej spójności

- W algorytmach CSP występują dwie części;
  (i) pierwsza polega na poszukiwaniu przypisania z kilku możliwych przypisań zmiennym
  (ii) druga - za pomocą propagacji więzów algorytm ustala, które wartości zmiennej (połączonej z bieżącą zmienną) należy uznać za dopuszczalne (tj. które wartości należy wykluczyć z rozważań)
- Sudoku można rozwiązać posługując się tylko propagacją więzów.
- Istnieje wiele problemów CSP, których nie można rozwiązać jedynie poprzez propagację ograniczeń. W takich przypadkach musimy zastosować algorytmy przeszukiwania.

Backtracking search (poszukiwanie z powrotami)

- Jest to przeszukiwanie ‘depth-first’ które wybiera wartośći dla jednej zmiennej w jednym kroku i wycofuje się z tego, gdy zmienna nie ma żadnej dopuszczalnej wartości do przypisania.
- Algorytm wielokrotnie wybiera zmienną, której nie została przypisana wartość, a następnie kolejno wypróbowuje wszystkie wartości z jej dziedziny, próbując znaleźć rozwiązanie.
- Jeśli zostanie wykryta niespójność, zwraca błąd, wymuszając w poprzednim przypisaniu wybranie innej wartości.

Pseudokod: backtracking search

```
function BACKTRACKING-SEARCH(csp) returns a solution, or failure
  return BACKTRACK({ }, csp)

function BACKTRACK(assignment, csp) returns a solution, or failure
  if assignment is complete then return assignment
  var ← SELECT-UNASSIGNED-VARIABLE(csp)
  for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
    if value is consistent with assignment then
      add {var = value} to assignment
      inferences ← INFERENCE(csp, var, value)
      if inferences ≠ failure then
        add inferences to assignment
        result ← BACKTRACK(assignment, csp)
        if result ≠ faliure then
          return result
    remove {var = value} and inferences from assignment
  return failure
```

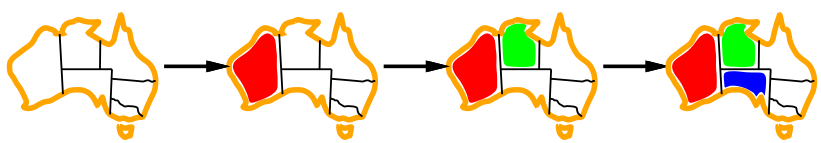
- Algorytm modelowany jest na podstawie ‘depth-first search’.
- Funkcja SELECT-UNASSIGNED-VARIABLE() decyduje, którą zmienną wybrać jako następną, a funkcja ORDER-DOMAIN-VALUES() porządkuje wartości zmiennej, która ma być wybrana do sprawdzenia - mogą być różne definicje tych funkcji
- Funkcja INFERENCE() zasadniczo sprawdza lokalną spójność przypisywania wartości do zmiennych. Ta funkcja może być zdefiniowana wykorzystując spójność łuku lub spójność ścieżki itp.

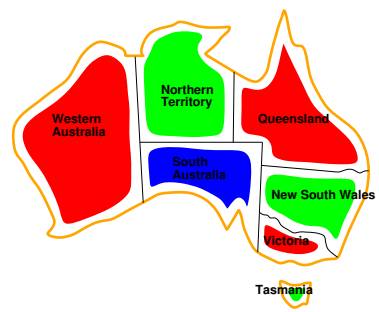
Metody wyboru nieprzypisanej zmiennej

- Minimum Remaining Values (MRV): Wymusza wybranie zmiennej o najmniejszej liczbie dopuszczalnych wartości, nazywanej zmienną z minimalną liczbą pozostałą wartością. Jeśli jakaś zmienna X nie ma żadnej dopuszczalnej wartości, MRV wybierze X i natychmiast wykryje błąd.
- Został również nazwany metodą najbardziej ograniczonej zmiennej (most constrained variable) lub heurystyką pierwszą do porażki (fail-first heuristic), ta druga, ponieważ wybiera zmienną, która najprawdopodobniej spowoduje błąd wkrótce, tym samym przycinając drzewo przeszukiwania.

Wybór zmiennej w kolejnym kroku przeszukiwania:

- najbardziej ograniczona zmienna, tzn. zmienna z najmniejszą liczbą dopuszczalnych wartości

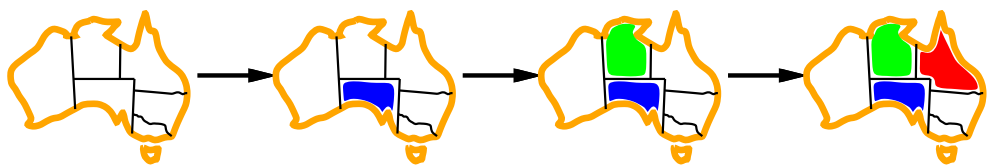




- Degree heuristic: W przypadku, gdy na początku wszystkie zmienne mają taką samą liczbę dopuszczalnych, wartości, stopień heurystyczny staje się bardziej użyteczny.
- Wybiera zmienną, która jest powiązana z największą liczbą więzów z innymi nieprzypisanymi zmiennymi.

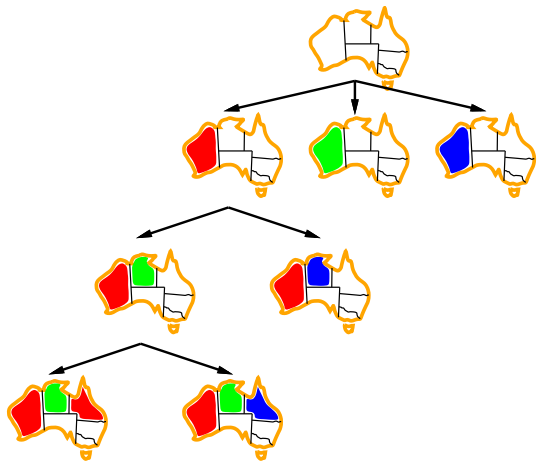
Wybór zmiennej w kolejnym kroku przeszukiwania:

- najbardziej ograniczająca zmienna, tzn. zmienna z największą liczbą więzów z pozostałymi zmiennymi



Tworzenia uporządkowania wartości dziedzin

- Least constraining value: Po wybraniu zmiennej lepiej jest wybrać wartość, która eliminuje najmniejszą liczbę opcji dla sąsiednich zmiennych.
- Ogólnie usiłuje pozostawić maksymalną elastyczność dla kolejnych przypisań zmiennych.
- Na przykład, jeśli wybierzemy 'WA = red', a 'NT = green', a następnie, jeśli naszym następnym wyborem będzie Q, to niebieski będzie złym wyborem, ponieważ eliminuje ostatnią dopuszczalną wartość pozostawioną sąsiadowi Q, SA. Ten sposób preferuje zatem bardziej kolor czerwony od niebieskiego.



Metoda projektowania ‘INFERENCE‘

- Sprawdzanie w przód jest najprostszą formą wnioskowania. Ilekroć zmienna X jest przypisana przy sprawdzaniu w przód, zadaniem metody jest zapewnienie spójności łuku między X a każdą zmienną Y, która jest połączona z X. Oznacza to, że usunie z dziedziny Y każdą wartość, która jest niezgodna z wartością wybraną dla X.

	WA	NT	Q	NSW	V	SA	T
Initial domain	R, G, B	R, G, B	R, G, B	R, G, B	R, G, B	R, G, B	R, G, B
After WA = red	R	G, B	R, G, B	R, G, B	R, G, B	G, B	R, G, B
After Q = green	R	B	G	R, B	R, G, B	B	R, G, B
After V = blue	R	B	G	R	B		R, G, B

- Poprzednio widzieliśmy, że na podstawie opisu środowiska zadań możemy stworzyć agenta bazującego na oprogramowaniu do poszukiwania optymalnego rozwiązania (w algorytmach przeszukiwania), optymalnych strategii (w przypadku środowiska wieloagentowego, takiego jak gry) i wnioskowania prowadzącego do rozwiązania za pomocą reprezentacji ograniczeń (w przypadku CSP)
- Jeśli przed uruchomieniem algorytmu przeszukiwania mamy już opis problemu dostępny w postaci ograniczeń, zadanie staje się łatwiejsze. Ale często musimy nawet najpierw sformułować ograniczenia opisujące problem.
- Kolejnym trudnym zadaniem jest wyrażenie ograniczeń lub reguł z tablic danych, która zawiera opisy zmiennych i ich możliwych wartości.

Teoria Zbiorów przybliżonych

- Zbiory przybliżone (Pawlak, 1981)
- Redukty i reguły generowane z reduktów (Skowron, Rauszer, 1992)
- $A = \{a_1, a_2, \ldots, a_n\}$: zbiór cech (atrybutów lub zmiennych) opisujących przykłady
- U_{trn}: zbiór przykładów opisanych wektorami wartości cech $\langle v_{11}, v_{21}, \ldots, v_{n1} \rangle$ gdzie $v_{11} \in D_1$, $v_{21} \in D_2$, $\ldots v_{n1} \in D_n$, a D_i to domena wartości dla a_i.
- Przykład: $U_{trn} = \{o_1, o_2, o_3, o_4\}$, $A = \{a, b, c, d\}$ - atrybuty warunkowe, (U_{trn}, A): System informacyjny
- dec: atrybut decyzji, (U_{trn}, A, dec): system decyzyjny

System Informacyjny

	a	b	c	d
o_1	0	2	1	0
o_2	1	2	2	1
o_3	2	0	2	1
o_4	0	2	1	1

System decyzyjny

	a	b	c	d	dec
o_1	0	2	1	0	0
o_2	1	2	2	1	0
o_3	2	0	2	1	1
o_4	0	2	1	1	2

Redukty decyzyjny

- Definicja: Zbiór atrybutów $R \subseteq A$ jest reduktem decyzyjnym dla zbioru przykładów U_{trn}, jeśli
  - dla każdej pary przykładów $o_i, o_j \in U_{trn}$ o różnych decyzjach $dec(o_i) \neq dec(o_j)$ o ile istnieje $a \in A$ rozróżniający tę parę przykładów, to istnieje $a_k \in R$ rozróżniający tę pare przykładów: $v_{ik} \neq v_{jk}$ gdzie $v_{ik} = a_k(o_i)$, $v_{jk} = a_k(o_j)$.
  - R jest minimalnym zbiorem mającym powyższą własność, tzn., dla dowolnego $R' \subseteq R$, istnieje para przykładów w U_{trn} o róznych decyzjach i takich samych wartościach na wszystkich atrybutach $a_i \in R'$.

	a	b	c	d	dec
o_1	0	2	1	0	0
o_2	1	2	2	1	0
o_3	2	0	2	1	1
o_4	0	2	1	1	2

- Redukty: $\{a, d\}$, $\{b, c, d\}$

Minimalny redukt

- Definicja: Redukt R jest minimalny, jeśli zawiera najmniejszą możliwą liczbę atrybutów, tzn. dla każdego reduktu R', $|R| \leq |R'|$.
- Fakt: Problem znalezienia minimalnego reduktu jest NP-trudny.
- W powyższym przykładzie minimalny redukt to $\{a, d\}$

Generowanie reguł z reduktu

- $Rules(R) = \{\bigwedge_{a_i \in R} a_i(o) = v_i \Rightarrow dec = dec(o) : o \in U_{trn}\}$
- Znajdźmy $Rules(\{b, c, d\})$:

	a	b	c	d	dec
o_1	0	2	1	0	0
o_2	1	2	2	1	0
o_3	2	0	2	1	1
o_4	0	2	1	1	2

- Reguły:
  $b = 2 \wedge c = 1 \wedge d = 0 \Rightarrow dec = 0$
  $b = 2 \wedge c = 2 \wedge d = 1 \Rightarrow dec = 0$
  $b = 0 \wedge c = 2 \wedge d = 1 \Rightarrow dec = 1$
  $b = 2 \wedge c = 1 \wedge d = 1 \Rightarrow dec = 2$

Skracanie reguł z reduktu

- Skracanie reguły polega na odrzuceniu niektórych atrybutów z warunku reguły.
- Reguła $\alpha \wedge s \Rightarrow dec = \beta$ może zostać zastąpiona przez $\alpha \Rightarrow dec = \beta$, jeśli $\alpha \Rightarrow dec = \beta$ pozostaje spójna ze zbiorem treningowym.
- Fakt: Może się zdarzyć, że różne reguły z tą samą decyzją zostaną skrócone do tej samej postaci. To oznacza, zbiór reguł po skróceniu może być mniejszy niż oryginalny.

Przykład

	a	b	c	d	dec
o_1	0	2	1	0	0
o_2	1	2	2	1	0
o_3	2	0	2	1	1
o_4	0	2	1	1	2

- Reguły:
  $b = 2 \wedge c = 1 \wedge d = 0 \Rightarrow dec = 0$
  $b = 2 \wedge c = 2 \wedge d = 1 \Rightarrow dec = 0$
  $b = 0 \wedge c = 2 \wedge d = 1 \Rightarrow dec = 1$
  $b = 2 \wedge c = 1 \wedge d = 1 \Rightarrow dec = 2$
- Po skróceniu:
  $b = 2 \wedge d = 0 \Rightarrow dec = 0$ lub $c = 1 \wedge d = 0 \Rightarrow dec = 0$
  $b = 2 \wedge c = 2 \Rightarrow dec = 0$
  $b = 0 \Rightarrow dec = 1$
  $c = 1 \wedge d = 1 \Rightarrow dec = 2$

Klasyfikacja oparta na wsparciu

- *Rules* = zbiór reguł z jednoznaczną decyzją
- U_{trn} = zbiór przykładów treningowych
- u : obiekt do klasyfikacji
- Klasyfikacja przez maksymalizację wsparcia

$$Rules(u) = \{\alpha \Rightarrow dec = \beta : u \textbf{ spełnia } \alpha\}$$
$$max\ arg_{\beta} |\{y \in U_{trn} : \exists \alpha \Rightarrow dec = \beta \in Rules(x)(y \textbf{ spełnia } \alpha \wedge dec(y) = \beta\}|$$

Przykład

	a	b	c	d	dec
x_1	0	2	1	0	0
x_2	1	2	2	1	0
x_3	2	0	2	1	1
x_4	0	2	1	1	2
u	0	0	2	1	?

- **Reguły**:
  $b = 2 \wedge c = 1 \wedge d = 0 \Rightarrow dec = 0$
  $b = 2 \wedge c = 2 \wedge d = 1 \Rightarrow dec = 0$
  $b = 0 \wedge c = 2 \wedge d = 1 \Rightarrow dec = 1$
  $b = 2 \wedge c = 1 \wedge d = 1 \Rightarrow dec = 2$
- **Po skróceniu**:
  $b = 2 \wedge d = 0 \Rightarrow dec = 0$ **lub** $c = 1 \wedge d = 0 \Rightarrow dec = 0$
  $b = 2 \wedge c = 2 \Rightarrow dec = 0$
  $b = 0 \Rightarrow dec = 1$
  $c = 1 \wedge d = 1 \Rightarrow dec = 2$
- $Rules(u) = \{b = 0 \Rightarrow dec = 1\}$
- Zatem u jest klasyfikowany z $dec = 1$ z maksymalnym wsparciem 1.

Dziękuję za uwagę