POLITECHNIKA KOSZALIŃSKA

WYDZIAŁ ELEKTRONIKI I ELEKTRONIKI

INFORMATYKA

PROGRAMOWANIE KOMPUTERÓW

I SIECI INFORMATYCZNE

Łukasz Zieliński

62093

MODELOWANIE STRUKTUR BLOKOWYCH SYSTEMÓW ROZMYTYCH ZA POMOCĄ JĘZYKA FUZZY CONTROL LANUGAGE

MODELLING BLOCK STRUCTURES OF FUZZY SYSTEMS WITH FUZZY CONTROL LANGUAGE

Praca inżynierska wykonana pod kierunkiem

dr inż. Marek Popławski

Oświadczenie

Łukasz Zieliński
(Imię i Nazwisko studenta)

Numer 62093
(Nr albumu)

Oświadczam, że moja praca pt.: *Modelowanie struktur blokowych systemów rozmytych za pomocą języka Fuzzy Control Language.*
została przygotowana przeze mnie samodzielnie*,

a. nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 r. o pra- wie autorskim i prawach pokrewnych (Dz. U. Nr 24., poz. 83 z późn. zm.) oraz dóbr osobistych chronionych prawem,
b. nie zawiera danych i informacji, które uzyskałem w sposób niedozwolony,
c. nie była podstawą nadania dyplomu uczelni wyższej lub tytułu zawodowego ani mnie ani innej osobie.

Ponadto oświadczam, że treść pracy przedstawionej przez mnie do obrony, zawarta na przekazywanym nośniku elektronicznym, jest identyczna z jej wersją drukowaną.

……. (data)

…… (podpis studenta)

*Uwzględniając merytoryczny wkład promotora (w ramach prowadzonego seminarium dyplomowego)

Spis treści

Wstęp

Pomysł na logikę rozmytą zrodził się w 1960 r na Uniwersytecie Kalifornijskim w Berkley. Została ona zaproponowana przez Dr Lotfi Zadecha, który pracował nad problemem rozumienia naturalnego języka przez komputer. Naturalny język, podobnie jak praktycznie wszystko w otaczającym nas świecie ciężko jest przetłumaczyć na komputerowe 0/1. Samo pytanie czy wszystko da się wytłumaczyć binarnie jest czysto filozoficzne, choć zazwyczaj dane które mamy do przetworzenia dostajemy już w tej postaci z innego, elektronicznego źródła.

Logika rozmyta do pojęć prawdy i fałszu dodaje również całą gamę wartości pośrednich. Takie podejście znane było już wcześniej w logice trójwarościowiej Łukasiewicza (L3) [1], podobne próby podejmował również Clarence Lewis. Trzecim stanem w logice Łukasiewicza był stan nieznany (0.5). Lotfi ten pomysł rozwinął na całą przestrzeń wartości <0,1>.

W logice rozmytej wartości 0 i 1 stanowią ekstremalne przypadki prawdy (albo stanu ważności, faktu), może ona jednak przybierać dowolne wartości ze zbioru <0,1> co daje możliwość stwierdzenia, że coś jest jednocześnie np. w 0.7 wysokie i w 0.3 ciepłe. Jest to bardzo bliskie naszemu naturalnemu procesowi myślowemu, gdzie nic nigdy nie jest do końca czarne ani białe. Logika rozmyta nie wyklucza swoim istnieniem logiki binarnej, jest jedynie jej uogólnieniem. Prawa, które nią rządzą dają się zaaplikować do logiki Boole'a dlatego, że w zadbano o zachowanie praw Augusta De Morgana [2].

Logika rozmyta rozpoczyna się od wykreowania reguł spisanych w naturalnym języku. System rozmyty konwertuje te reguły do ich matematycznego odpowiednika na podstawie odpowiedniego modelu rozmytego. Upraszcza to w dużej mierze pracę projektanta takiego systemu, ponieważ reprezentacja rozwiązania problemu jest bardzo czytelna dla człowieka. Również rezultaty prac takiego systemu są celniejsze niż te z systemów opartych na logice dwuwartościowej, ponieważ system pracuje w rzeczywistym świecie, gdzie rozmycie wartości wejściowych i wyjściowych jest czymś naturalnym. Kolejnym plusem logiki rozmytej jest jej prostota i elastyczność. Może ona rozwiązywać problemy na podstawie niekompletnych, niedokładnych danych. Może też modelować nieliniowe funkcje o dużej złożoności. Przydaje się to szczególnie w przypadkach, gdy mamy do czynienia ze środowiskiem, które może się zmieniać, ponieważ przeprojektowanie modelu funkcji rozmytych nie wymaga wiedzy na temat złożoności algorytmicznej problemu,
tylko wiedzy eksperckiej - reguły wyrażamy w czytelnym naturalnym języku.

W praktyce można niewielkim nakładem prac napisać system, który będzie obsługiwał każdy przypadek danych wejściowych. Teoria zbiorów rozmytych cieszy się dość dużą popularnością w dziedzinach związanych ze sterowaniem oraz przetwarzaniem języka. Produkowane są mikroprocesory których rozkazy projektowane są pod przetwarzanie rozmyte. Powstało do tego czasu kilka sposobów reprezentacji wiedzy systemu rozmytego. Jednym z nich jest FCL.

Fuzzy Control Language (FCL) jest językiem który został ustandaryzowany przez Międzynarodową Komisję Elektrotechniczną (IEC) w dokumencie IEC 61131-7. Język ten nie zawiera żadnych funkcjonalności nie związanych z logiką rozmytą, co sprawia, że nie da się w nim napisać najprostszego programu który da jakiekolwiek rezultaty (np "Hello World"). Może on być jednak częścią większego programu. Biblioteka napisana przeze mnie w języku Java ma na celu interpretację pliku skryptowego FCL i jego reprezentację w postaci struktury powiązanych obiektów. Do biblioteki dołączyłem aplikację webową, która jest środowiskiem IDE do języka FCL. Można ją wykorzystać do sterowania inną aplikacją, która np. modeluje problem. Aplikacja pozwala na ukazanie wewnętrznych mechanizmów zaprojektowanego systemu rozmytego

I. Podstawy teorii modelowania systemów rozmytych

1. Reguły sterownika rozmytego

Reguły którymi opisuje się sterownik rozmyty tworzone są następująco :

```
if <przesłanka> then <konkluzja>
```

Przesłankę tworzy się za pomocą serii predykatów połączonych spójnikami logicznymi, Konkluzję tworzą kolejne predykaty. Struktura ta przypomina swą budową bazy wiedzy znane z systemów ekspertowych, jednak wewnątrz każdej reguły "dzieje się" tu dużo więcej.

W zależności od zastosowanego modelu rozmytego możemy spotkać się z różnymi formami predykatów. Aplikacja, która jest przedmiotem tej pracy, bazuje na modelu Mamdaniego [3]. W tym modelu predykaty są postaci :

```
A is B
```

gdzie :
A - nazwa danej zmiennej lingwistycznej
B - jeden z termów opisujących tą zmienną.

Teoria zbiorów rozmytych zakłada, że termy określające zmienną lingwistyczną mają postać zbiorów rozmytych. Zmienna jest więc zbiorem nazwanych zbiorów rozmytych, które mają nazwy wzięte z języka naturalnego. Predykaty w tej postaci może wyglądać tak :

```
temperatura is niska
```

w miejsce niska możemy wstawić też np. wysoka, średnia. Definiujemy w ten sposób kolejne termy ze zbioru termów zmiennej temperatura. Termom tym przypisane są pewne zbiory rozmyte które opisują małą, średnią i wysoką temperaturę.

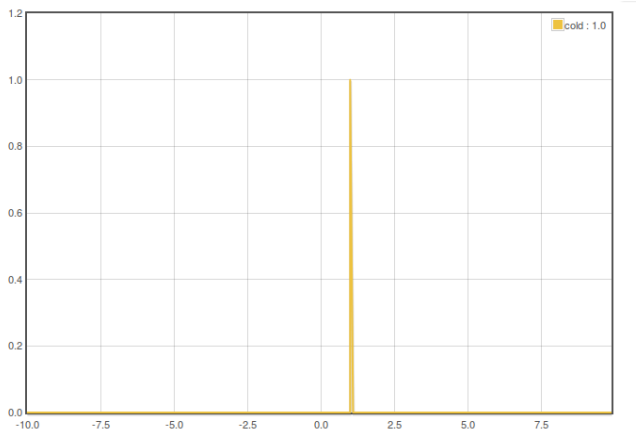
Zmienne lingwistyczne które występują po lewej stronie reguły nazywamy zmiennymi wejściowymi sterownika. Natomiast zmienne znajdujące się w konkluzji to zmienne w-wyjściowe. Do wnioskowania rozmytego jako podstawę użyto uogólnioną na teorię zbiorów rozmytych regułę *modus ponens*. Zachowano element, który określał, że prawdziwość przesłanki implikacji pozwala wnioskować o prawdziwości konkluzji. Dodano pojęcie stopnia prawdziwości albo stopnia spełnienia zarówno przesłanki jak i konkluzji.

2. Bloki i proces wnioskowania rozmytego

2.1. Blok rozmywania

Bazując cały czas na modelu Mamdaniego wyróżniamy kolejne etapy(bloki) wnioskowania rozmytego. Pierwszym z nich jest blok rozmywania. Blok ten przekształca wartość zmiennej wejściowej (zmiennej lingwistycznej) na stopień spełnienia predykatów - przesłanek reguł, które angażują tę zmienną.

Najczęściej spotykaną i najprostszą obliczeniowo metodą, która realizuje te cele jest metoda typu singleton. Polega ona na utworzeniu funkcji - zbioru rozmytego X następującej postaci :



rys I.2.1/1 Singleton

Który zdefiniowany jest funkcją :

$$f(x) = \begin{cases} 1 : x = C \\ 0 : x \neq C \end{cases}$$

Następnie jako wartość spełenia predykatu przesłanki A is B uznaje się zbiór powstały w wyniku przecięcia zbioru X ze ze zbiorem skojarzonym z termem B. Jest to prosta metoda w wyniku której zbiorem wynikowym predykatu przesłanki jest liczba mówiąca o stopniu aktywacji tego predykatu. Istnieje wiele innych, bardziej skomplikowanych metod rozmywania [4].

2.2. Blok i metody inferencji

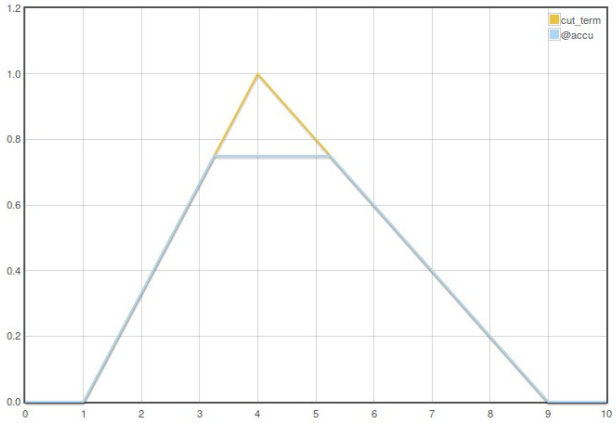
Inferencja jest kolejnym etapem działania sterownika rozmytego. Jest ona jednocześnie najbardziej złożona obliczeniowo. Możemy wyróżnić trzy etapy prac tego bloku :

- agregacja (aggregation)
- wnioskowanie
- kumulacja (accumulation)

Wszystkie te etapy wykonywane są w oparciu o przygotowaną przez eksperta baze reguł. W poprzednim bloku dane zmiennych wejściowych zamienione zostały na stopnie spełnienia odpowiednich predykatów w przesłankach reguł. W tym bloku następuje uruchomienie wszystkich reguł, których przesłanki są spełnione, wyliczenie zbioru rozmytego który jest wynikiem tych reguł oraz kumulacja wyników w każdym bloku reguł.
Na etapie agregacji stopień spełnienia każdej z reguł obliczany jest na podstawie stopnia spełnienia ich przesłanek. W tym celu używane są logiczne operatory rozmyte znane z logiki Boole'a : AND, OR oraz NOT. W zastosowanej przeze mnie implementacji języka Fuzzy Control Language dostępne są następujące funkcje (t-normy) dla tych spójników :

	AND	OR
MIN	$\mu_{A,B}(x,y) = min(\mu_A(x),\mu_B(y))$	$\mu_{A,B}(x,y) = max(\mu_A(x),\mu_B(y))$
PROD	$\mu_{A,B}(x,y) = \mu_A(x)\cdot\mu_B(y)$	$\mu_{A,B}(x,y) = \mu_A(x)+\mu_B(y)-\mu_A(x)\cdot\mu_B(y)$

Te reguły dla których stopień spełnienia zagregowanych przesłanek jest niezerowy zostają aktywowane. W wyniku wnioskowania obliczany jest zbiór rozmyty stanowiący konkluzję danej reguły. Operatorem implikacji jest operator MIN, a polega to obrazowo na "obcięciu" zbioru termu wyjściowego na wysokości stopnia spełnienia zagregowanych przesłanek, na przykład :



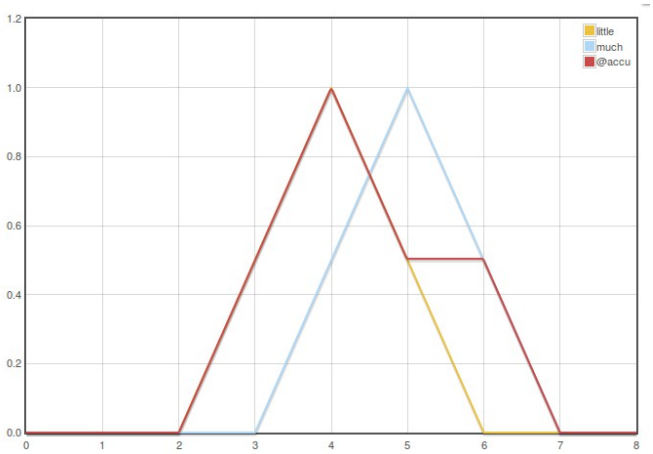
rys I.2.2/1 Przykład obciętego termu

Dodatkowo w konkluzji każdej reguły możemy podać kilka predykatów dotyczących różnych lub nawet tej samej zmiennej, rozdzielając je operatorem AND. Użycie operatora o tej nazwie podyktowane było zachowaniem czytelności reguł, nie ma on żadnego znaczenia pod względem matematycznym.

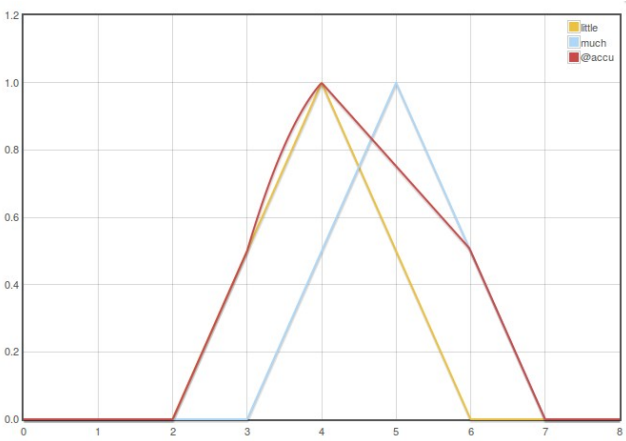
Wynikowe zbiory rozmyte kumulują się w ramach danego bloku funkcji sterownika w zbiór rozmyty, który poddawany jest następnie procesowi wyostrzania. Kumulacja polega na zebraniu wszystkich wynikowych zbiorów rozmytych danej zmiennej w jeden zbiór za pomocą funkcji kumulacji. Funkcje kumulacji dostępne w implementowanym języku FCL :

	ACCU
MAX	$\mu_{A,B}(x,y) = max(\mu_A(x), \mu_B(y))$
PROD	$\mu_{A,B}(x,y) = \mu_A(x) + \mu_B(y) - \mu_A(x) \cdot \mu_B(y)$

Obie metody nieco się różnią kształtami zbiorów wynikowych np.:



rys I.2.2/1 Akumulacja metodą MAX



rys I.2.2/2 Akumulacja metodą PROD

Różnice te nieznacznie wpływają na rezultat działania całego systemu.

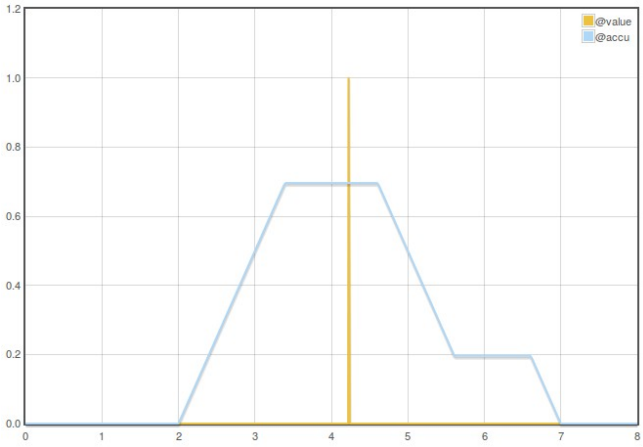
2.3. Blok wyostrzania.

Wyostrzanie ma na celu przekształcenie wynikowego zbioru rozmytego na określoną wartość rzeczywistą stanowiącą wartość wyjścia modelu.

W tworzonym systemie możliwymi operatorami są :

- Środek ciężkości (CoG -center of gravity).
- Środek ciężkości dla singletonów (CoGS - center of gravity for singletons)

	WZÓR
COG	$$v = \frac{\int_{v_{min}}^{v_{max}} v\mu_v(v)\,dv}{\int_{v_{min}}^{v_{max}} \mu_v(v)\,dv}$$
COGS	$$v = \frac{\sum_{t \in S} \mu_t t}{\sum_{t \in S} t}, gdzie\ S - zbiór\ zagregowanych\ singletonów$$



rys I.2.3/1 Wyostrzanie metodą COG

3. Przykład wnioskowania rozmytego

3.1 Modelowanie rozmyte na przykładzie zraszania trawnika

3.1.1 Zbieranie danych

Załóżmy, że mamy trawnik o sporej powierzchni, który codziennie w miarę potrzeb podlewa sztab ogrodników. W celu zmniejszenia kosztów zatrudnienia chcemy utworzyć system, który będzie automatycznie zraszał nasz trawnik i będzie to robił w zależności od aktualnych warunków pogodowych – wilgotności gruntu i temperatury. W tym celu rozmieszczamy czujniki temperatury, wilgotności i nasłonecznienia w strategicznych punktach trawnika, tworzymy prosty system filtrujący błędne dane (powstałe w przypadku awarii jednego z czujników) i stajemy przed problemem implementacji właściwej aplikacji sterującej zraszaniem. Przy projektowaniu takiej aplikacji, w której zastosujemy logikę rozmytą, możemy skorzystać z pomocy jednego z ogrodników, który do tej pory pracował w naszym gospodarstwie. W tym celu prosimy go o spisanie zasad, którymi kierował się podlewając trawnik. Otrzymujemy od niego mniej więcej taki dokument:

Jeżeli temperatura jest bardzo niska to nie podlewamy trawnika
Jeżeli wilgotność jest duża i temperatura jest niska to nie podlewamy trawnika
Jeżeli wilgotność jest duża i temperatura jest średnia to nie podlewamy trawnika
Jeżeli wilgotność jest duża i temperatura jest wysoka to ustawiamy słabe zraszanie
Jeżeli wilgotność jest średnia i temperatura jest niska to nie podlewamy trawnika
Jeżeli wilgotność jest średnia i temperatura jest średnia to ustawiamy słabe zraszanie
Jeżeli wilgotność jest średnia i temperatura jest wysoka to ustawiamy średnie zraszanie
Jeżeli wilgotność jest niska i temperatura jest niska to ustawiamy słabe zraszanie
Jeżeli wilgotność jest niska i temperatura jest średnia to ustawiamy średnie zraszanie
Jeżeli wilgotność jest niska i temperatura jest wysoka to ustawiamy silne zraszanie

Z racji, że pewne określenia zawarte w tym dokumencie nie do końca są jasne zbieramy kolejne informacje:
temperatura bardzo niska jest poniżej 5°C
temperatura niska to około 10°C
temperatura średnia to około 18°C
temperatura wysoka to 25 °C i więcej
dowiadujemy się też, że około to +- 8°C

wilgotność niska to poniżej 10%
wilgotność średnia to około 15%
wilgotność duża to to powyżej 20%
około to +- 4%

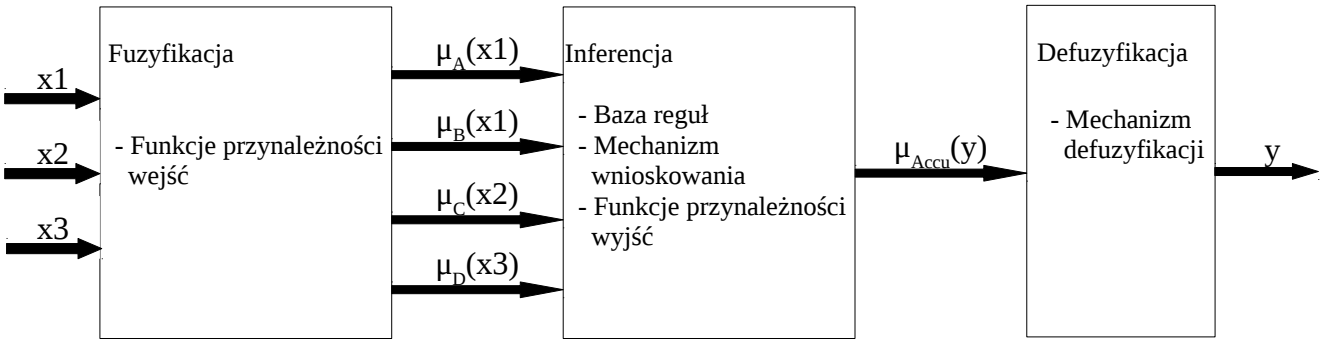
zraszanie słabe to 20%
zraszanie średnie to 40%
zraszanie silne to 60%

Większego zraszania nigdy nie ustawiali, ponieważ podmywało glebę. Z rozmowy wynika również, że sprawdzali warunki co pół godziny i dostrajali cały system, można jednak dostrajać go w trybie ciągłym, cały czas próbkując dane w małych odstępach czasu – wpłynie to pozytywnie na zużycie wody.

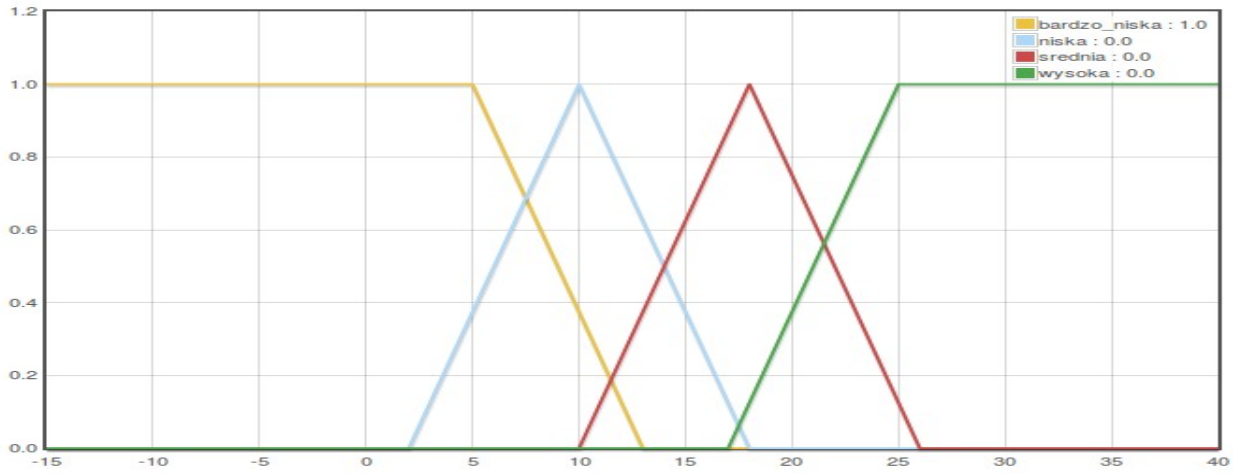
Z tak zebranymi danymi możemy przystąpić do zaprojektowania naszego systemu rozmytego.

3.1.2 Modelowanie rozmyte

Do implementacji naszej aplikacji zraszającej trawnik użyjemy modelu Mamdaniego. W ogólnym zarysie model ten prezentuje się następująco:



Z naszych danych tworzymy funkcje przynależności zmiennych wejściowych:

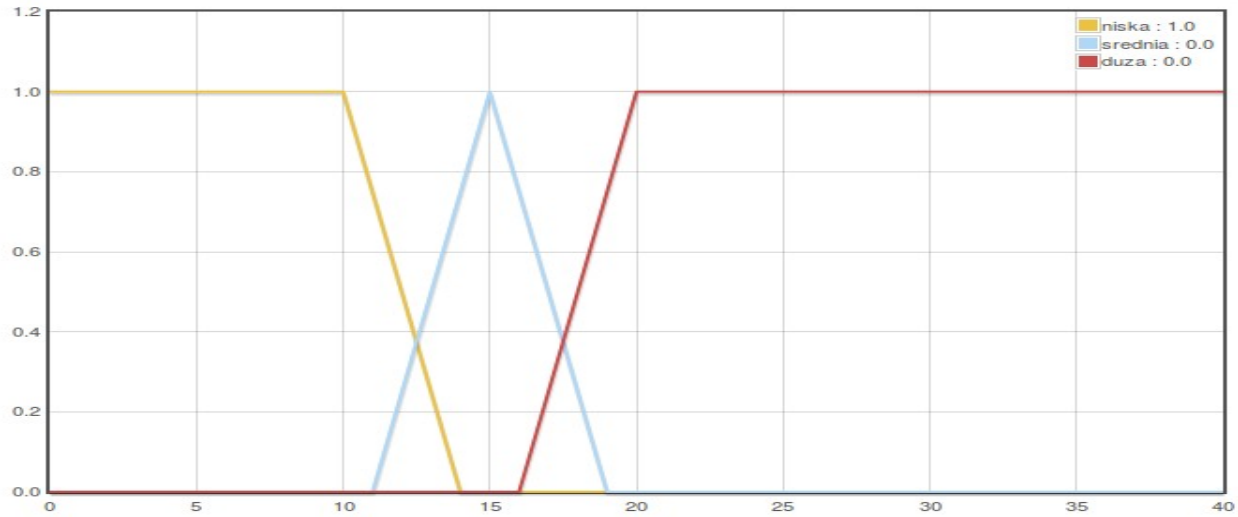


rys I.3.1.2/1 Zraszanie trawnika, funkcje przynależności termów zmiennej temperatura

Przy tworzeniu zbiorów korzystamy z punktów które przedstawił nam ogrodnik opisując własne podejście do temperatury tworząc z nich kolejne termy:

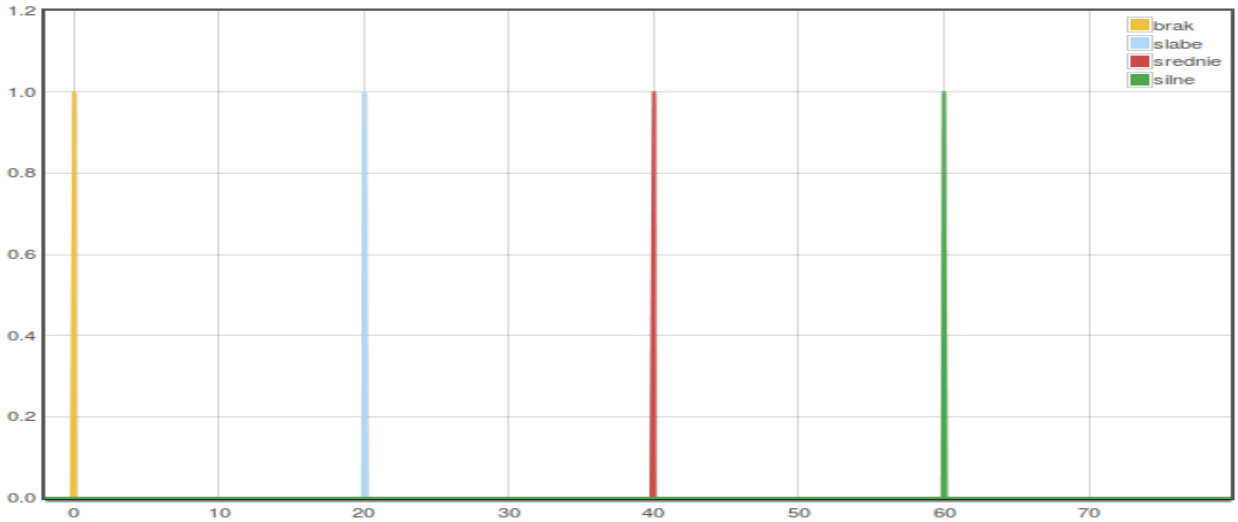
- TERM bardzo_niska := (-15,1) (5,1) (13,0);
- TERM niska := (2, 0) (10, 1) (18, 0);
- TERM srednia := (10, 0) (18, 1) (26,0);
- TERM wysoka := (17,0) (25, 1) (40, 1);

Kolejną zmienną wejściową jest wilgotność gruntu dla uproszczenia nazwana dalej wilgotnością:



rys I.3.1.2/2 Zraszanie trawnika, funkcje przynależności termów zmiennej wilgotność

Pozostaje jeszcze zmienna wyjściowa zraszanie, którą opiszemy specjalnym rodzajem termów – singletonami :



rys I.3.1.2/3 Zraszanie trawnika, funkcje przynależności termów zmiennej wyjściowej zraszanie

Do tego potrzebujemy jeszcze zestawu reguł, które opisał nam ogrodnik. Dla ułatwienia zapiszę je w notacji używanej w FCL:

Bibliografia

1) https://pl.wikipedia.org/wiki/Logika_tr%C3%B3jwarto%C5%9Bciowa
2) https://en.wikipedia.org/wiki/De_Morgan%27s_laws
3) Mamdani, E. H., Application of fuzzy algorithms for the control of a simple dynamic plant. In *Proc IEEE* (1974)
4) Andrzej Piegat; „Modelowanie i sterowanie rozmyte”; Akademicka Oficyna Wydawnicza EXIT, Warszawa 1999