Inżynieria wiedzy

metody reprezentacji wiedzy

II. Metody reprezentacji wiedzy

1. Wiedza oraz sposoby jej reprezentacji
2. Rachunek zadań
3. Stwierdzenia
4. Regułowa reprezentacja wiedzy
5. Rachunek predykatów
6. Sieci semantyczne
7. Reprezentacja wiedzy za pomocą ram
8. Modele obliczeniowe

1. Wiedza oraz sposoby jej reprezentacji

***Wiedza*:**

- **jej reprezentacja w dziedzinie sztucznej inteligencji jest problemem, który nie został jeszcze w pełni rozwiązany,**
- **najczęściej oznacza zbiór wiadomości z określonej dziedziny,**
- **oznacza wszelkie zobiektywizowane i utrwalone formy kultury umysłowej i świadomości społecznej powstałe w wyniku kumulowania doświadczeń i uczenia się,**
- **jest symbolicznym opisem otaczającego nas świata rzeczywistego charakteryzującym aksjomatyczne i empiryczne relacje zawierającymi procedury,**

***Wiedza* składa się z następujących elementów:**

- ***Opisy* – stanowią zdania w jakimś języku, którego elementarnymi składnikami są pierwotne cechy i pojęcia. Służą one do identyfikacji i rozróżniania obiektów i klas.**
- ***Relacje* – odzwierciedlają zależności i asocjacje (skojarzenia) pomiędzy faktami w bazie wiedzy.**
- ***Procedury* – elementy „manipulujące" relacjami.**

Jedną z najczęściej spotykanych form wiedzy jest *asocjacja empiryczna*. Pod tym sformułowaniem kryje się wiedza, jaką posiadają różni specjaliści (np. lekarze, geolodzy). Opiera się ona na wielu skojarzeniach dotyczących przyczyn obserwowanych danych i faktów. Ekspert zaopatrzony w wiedzę używa metod heurystycznych do powiązania problemów probabilistycznych, często błędnych danych w celu postawienia diagnozy.

Bardzo powszechną metodą reprezentacji wiedzy jest reprezentacja modeli opartych na logice. Logika klasyczna jest podstawowym narzędziem umożliwiającym pewne zautomatyzowanie procesu wnioskowania i pozyskiwania wiedzy. W konkretnych jednakże zastosowaniach, w których od systemu ekspertowego jest wymagana elastyczność i odporność na niepewne dane wejściowe, systemy te w wielu przypadkach okazały się mniej przydatne w porównaniu z systemami opartymi na innych metodach.

Z punktu widzenia logiki *wiedza* stanowi zbiór reguł i faktów. Dodając język, otrzymuje się teorię, czyli pewne skodyfikowane, uporządkowane wyobrażenie o pewnej klasie obiektów przedstawione za pomocą wyrażeń, formuł, reguł i gramatyki arbitralnie przyjętego języka. Należy podkreślić, że język jest pewną abstrakcyjną strukturą składającą się ze słownika i gramatyki, umożliwiającą mu operowanie na symbolach. Stanowi on *syntaktykę* danego systemu.

Semantyka z kolei nadaje znaczenie tworzonych zgodnie z gramatyką wyrażeń. Dodając do języka strukturę dedukcyjną – otrzymuje się pewien mechanizm wnioskujący umożliwiający produktywne funkcjonowanie teorii. Dzięki temu można przewidywać przyszłe stany systemu na podstawie obserwacji, obserwowanych faktów oraz tworzyć pewne uogólnienia na podstawie zaobserwowanej prawidłowości.

W strukturze dedukcyjnej znajduje się ponadto zbiór *aksjomatów logicznych* – ogólnych prawd o świecie, nie dowodzonych, lecz przyjmowanych za pewniki. Najistotniejszym elementem struktury jest zbiór *aksjomatów specyficznych*. Stanowią one właściwą bazę wiedzy. Mieszczą się w niej wszystkie udowodnione i zapisane fakty o opisywanym obiekcie lub systemie. Na podstawie informacji zgromadzonych w bazie wiedzy i w aksjomatach logicznych mechanizmy dedukcyjne udowadniają nowe twierdzenia lub je obalają.

Taka struktura wiedzy w postaci teorii ma zasadniczą zaletę, gdyż wszystkie fakty dowiedzione za pomocą danej spójnej teorii są prawdziwe. Wynika to z podstawowego twierdzenia logiki, które mówi, że pewne twierdzenie poprawne syntaktyczne jest prawdziwe semantycznie i na odwrót. Twierdzenie to stanowi podstawę tworzenia systemów zawierających wiedzę, opartych na metodach stosowanych w sztucznej inteligencji.

Zdefiniowanie pewnej syntaktyki, jest sprawą oczywistą przy tworzeniu oprogramowania maszyny cyfrowej. Stanowi ono element umożliwiający komunikację operator-maszyna pod warunkiem, że operator używa języka z jego gramatyką. Maszyna cyfrowa może bowiem operować tylko na wyrażeniach powstałych w wyniku działań syntaktycznych pozbawionych dla niej wszelkiego znaczenia, a więc na symbolach, literałach, ciągach znaków.

Za pomocą pewnych reguł może ona wyprowadzać z jednych ciągów liter i cyfr, inne ciągi znaków. Na tej podstawie mając wyprowadzone przez maszynę poprawne syntaktycznie wyrażenia można im przypisać semantykę, czyli nadać znaczenie wynikowi przetwarzań symbolicznych.

Dwa podstawowe typy symbolicznej reprezentacji wiedzy:

- *reprezentacja proceduralna* – polegająca na określeniu zbioru procedur, działanie których reprezentuje wiedzę w dziedzinie. Jej zaletą jest duża efektywność reprezentowania procesów, np. zapisanie w postaci równania jakiegoś prawa fizyki.
- *reprezentacja deklaratywna* – polegająca na określeniu zbioru specyficznych dla rozpatrywanej dziedziny faktów, stwierdzeń, reguł. Posiada ona łatwiejszy opis i formalizację, niż reprezentacja proceduralna.

Metody reprezentacji wiedzy zajmują się modelowaniem świata rzeczywistego z wykorzystaniem komputerów. Do najczęściej stosowanych metod reprezentowania wiedzy (technik organizowania baz wiedzy) zalicza się:

- metody bazujące na bezpośrednim zastosowaniu logiki: rachunek zdań, rachunek predykatów,
- metody wykorzystujące zapis stwierdzeń,
- metody wykorzystujące systemy regułowe (wektory wiedzy),
- metody z wykorzystaniem sieci semantycznych,
- metody oparte na ramach,
- metody używające modeli obliczeniowych.

Inną metodą reprezentacji wiedzy są *reprezentacje niesymboliczne*. Metody te odwołują się do obserwacji i doświadczeń zebranych na podstawie otaczającego nas świata żywych istot. Tzw. *sztuczne sieci neuronowe* symulują właściwości reprezentacji wiedzy i jej przetwarzania, podobnie jak komórki nerwowe zwierząt i ludzi. Przetwarzanie wiedzy odbywa się w sposób dynamiczny.

Natomiast inną technikę reprezentacji wiedzy stanowią tzw. *algorytmy ewolucyjne (genetyczne)*, które umożliwiają przekazywanie następnym generacjom wiedzy o całym gatunku. Wiedza jest zapisana w genach. W kolejnych generacjach następuje poprawa cecha całej populacji.

2. Rachunek zdań

Podstawowe koncepcje mechanizmów wnioskowania większości znanych systemów ekspertowych wywodzą się z *logiki dwuwartościowej*. Opis cech otaczającego nas świata formułujemy w postaci zdań. O zdaniu prawdziwym mówimy, że ma wartość logiczną 1, o zdaniu fałszywym, że ma wartość logiczną 0. Zdania oznaczamy symbolami, np. A, B, C, D…

Mogą one być łączone za pomocą funktorów zdaniotwórczych (spójników logicznych):

- **negacja ¬**
- **koniunkcja ∧**
- **alternatywa ∨**
- **implikacja =>**
- **równoważność <=>**

Łącząc zdania funktorami można tworzyć wyrażenia logiczne, tzw. *formuły*. Formuły, które zawsze dają zdanie prawdziwe niezależnie od wartości logicznych zmiennych zdaniowych, nazywa się *prawami rachunku zdań* (*tautologiami*).

Dwie metody sprawdzania, czy dana metoda jest tautologią:

1. Metoda zerojedynkowa

W metodzie tej wartość logiczna formuły złożonej jest wyznaczona przez wartości logiczne jej składników. Aby rozstrzygnąć, czy formuła jest tautologią, należy rozważyć wszystkie możliwe kombinacje wartości logicznych zmiennych w niej występujących. Jeżeli w każdym przypadku wartość formuły wynosi 1, to ta formuła jest tautologią. Można to zaobserwować na poniższym przykładzie, czy formuła o postaci

$$A = (\neg P \vee Q) <=> (\neg Q => \neg P)$$

jest tautologią. Są możliwe cztery kombinacje wartości logicznych zmiennych P i Q.

1) $P = 1$, $Q = 1$

$$(\neg 1 \vee 1) <=> (\neg 1 => \neg 1)$$
$$(0 \vee 1) <=> (0 => 0)$$
$$1 <=> 1$$
$$1$$

2) $P = 1, Q = 0$

$$(\neg 1 \vee 0) <=> (\neg 0 => \neg 1)$$
$$(0 \vee 0) <=> (1 => 0)$$
$$0 <=> 0$$
$$1$$

3) $P = 0, Q = 1$

$$(\neg 0 \vee 1) <=> (\neg 1 => \neg 0)$$
$$(1 \vee 1) <=> (0 => 1)$$
$$1 <=> 1$$
$$1$$

4) $P = 0, Q = 0$

$$(\neg 0 \vee 0) <=> (\neg 0 => \neg 0)$$
$$(1 \vee 0) <=> (1 => 1)$$
$$1 <=> 1$$
$$1$$

Ponieważ dla wszystkich kombinacji wartości logicznych zmiennych *P* i *Q* wartość formuły *A* wynosi 1, więc formuła *A* jest tautologią.

2. Metoda dedukcji (wnioskowania)

Wnioskowanie w tej metodzie odbywa się za pomocą reguł wnioskowania, które umożliwiają, na podstawie prawdziwości pewnych zadań zwanych *przesłankami*, wnioskowanie o prawdziwości innego zdania zwanego *wnioskiem*. Głównym zadaniem logiki rachunku zdań jest upraszczanie wyrażeń logicznych dla ich lepszego zrozumienia. Wykorzystuje się tu równoważność wyrażeń.

Bazy wiedzy oparte na logice, mimo swej modularności, deklaratywności i nieproceduralności, są trudne do przetwarzania. Bardzo szybko następuje w nich eksplozja kombinatoryczna, czyli lawinowy i niekontrolowany rozrost bazy wiedzy o fakty będące powieleniem niepożądanych struktur. Systemy ekspertowe oparte na klasycznej logice wykazują tendencje do wytwarzania tego typu nadmiarowych tautologii, np.:

jeżeli $A \vee B$ jest prawdą w bazie wiedzy,

to prawdą jest $A \vee B \vee B$

również prawdą jest $A \vee B \vee B \vee B \vee B$

3. Stwierdzenia

Stwierdzenia:

- **są jednym z głównych elementów baz wiedzy,**
- **dotyczą takich zagadnień jak zdarzenia, zjawiska, objawy, czynności,**
- **najczęściej są zapisywane w postaci uporządkowanej trójki**

(<OBIEKT> , <ATRYBUT> , <WARTOŚĆ>)

Dla uproszczenia zapisów stwierdzeń stosuje się słowniki nazw obiektów i atrybutów oraz ich wartości. Umożliwia to identyfikowanie zapisów za pomocą etykiet – bez wielokrotnego powtarzania nazw. Dzięki temu uzyskuje się oszczędniejszy zapis, który zajmuje mniej miejsca w komputerze. Na ogół stosuje się słowniki otwarte. Do wyrażania relacji między obiektami często używa się sieci semantycznych lub ram. W niektórych systemach stwierdzenia są zapisywane w postaci uporządkowanej czwórki

(<OBIEKT> , <ATRYBUT> , <WARTOŚĆ> , <CF>)

przy czym CF jest *stopniem niepewności* (Certainy Factor), prowadząc do tzw. stwierdzeń przybliżonych, prowadząc do tzw. stwierdzeń przybliżonych.

Każdemu stwierdzeniu jest wówczas przypisany pewien współczynnik określający stopień pewności stwierdzenia. Na ogół jest to liczba z pewnego przedziału [-1, 1] lub [0, 1] bądź [0, 10]. Jeżeli współczynnik ten jest definiowany dla pierwszego wymienionych przedziałów, to CF = 1 oznacza, że stwierdzenie jest w pełni prawdziwe. Z kolei dla CF = -1 stwierdzenie jest fałszywe. Natomiast CF = 0 oznacz stwierdzenie, o którym nie wiadomo, czy jest prawdziwe, czy fałszywe.

Stopnie pewności wyznacza się subiektywnie. Nie wprowadzono ich precyzyjnych definicji; nie sformalizowano również metod ich określania. W niektórych systemach stopnie pewności przypisuje się nie stwierdzeniom, lecz wartościom atrybutów, co może prowadzić do wielu paradoksów.

Rozwinięto także statystyczne metody wnioskowania wykorzystujące tzw. metody Bayesa, teorię podejmowania decyzji i metody weryfikacji hipotez statystycznych. Baza wiedzy takich systemów jest zbiorem stwierdzeń i relacji o pewnym wycinku świata rzeczywistego, której on dotyczy.

4. Regułowa reprezentacja wiedzy

Wśród różnych metod reprezentacji wiedzy ważną rolę odgrywają metody oparte na regułach. Zauważyć można, że zbiór stwierdzeń nie jest wystarczający do opisania jakiejś dziedziny wiedzy. Są jeszcze potrzebne reguły, których ogólna postać może być wyrażana jako:

JEŚLI przesłanka, TO konkluzja

oraz

JEŚLI przesłanka, TO działanie (produkcja)

lub inaczej za pomocą zdania

IF przesłanka THEN konkluzja

oraz

IF przesłanka THEN działanie (produkcja)

Przesłanka może zawierać pewną liczbę stwierdzeń połączonych funktorami logicznymi. Należy zwrócić uwagę, że niekontrolowany rozrost bazy wiedzy o nadmiarowe tautologie nie występuje w systemach opartych na regułach. Baza wiedzy zawiera w tym przypadku zbiór reguł oraz zbiór faktów. Możliwość reprezentowania świata ogranicza się do powyższych struktur zdaniowych.

Zdecydowana większość powstałych do tej pory systemów ekspertowych jest oparta na regułach. Podejście to umożliwia uzyskanie dużej modularności bazy wiedzy. Reprezentacja Regułowa często jest wykorzystywana w systemach dedukcyjnych, gdzie zbiór faktów początkowych jest przekształcany w pewien zbiór faktów końcowych.

W zależności od przeznaczenia system może spełniać różne funkcje, np. klasyfikowanie, diagnozowanie, dowodzenie, ustalanie przyczyn, najlepszy dobór, planowanie, prognozowanie, monitorowanie itp. Stosuje się czasem bardziej formalny zapis reguł, gdzie jest opuszczany symbol IF, z kolei zamiast napisu THEN używa się symbolu implikacji. Przesłanka jest wówczas wyrażana jako połączenie za pomocą funktorów logicznych pewnej liczby stwierdzeń. Przykładem może być reguła, w której przesłanka zawiera dwa warunki połączone funktorem koniunkcji

IF A przyjmuje wartość u
AND F przyjmuje wartość w
THEN G przyjmuje wartość y

Korzystając z symbolu implikacji regułę tę można zapisać w postaci

$$(A, u) \wedge (F, w) \Rightarrow (G, y)$$

która jest bardziej zwarta i przejrzysta.

Praktycznie działające systemy ekspertowe oparte na regułach mogą zawierać reguły charakteryzowane stopniami pewności. Podobnie jak przy reprezentowaniu stwierdzeń są to liczby zazwyczaj z przedziału [-1, 1] albo [0, 1]. Dzięki temu można za pomocą tych liczb dać znać systemowi, jaki jest stosunek użytkownika lub twórcy systemu do pewności konkluzji występującej w danej regule.

Zbiór można rozpatrywać jako szczególny sposób zapisu pewnej sieci stwierdzeń, ponieważ z prawdziwości jednego stwierdzenia mogą wynikać inne. W niektórych systemach jest dopuszczalna rozwinięta (tzw. pełna) postać reguł, która zawiera dodatkowe stwierdzenie uznawane za prawdzie w razie niespełnienia przesłanki. Ogólna postać reguły rozwiniętej jest następująca:

IF przesłanka THEN konkluzja1 ELSE konkluzja2

przy czym konkluzja2 jest tym dodatkowym stwierdzeniem. Taka postać reguł może jednak czasami prowadzić do uznania nieoczekiwanych konkluzji. Ze względu na złożoną nieraz kompletność bazy wiedzy, brak przesłanki jest utożsamiany z uznaniem tej przesłanki za przesłankę fałszywą. W związku z tym wskazane jest stosowanie reguł w postaci podstawowej (IF …THEN …). Postępowanie takie upraszcza działanie maszyny wnioskującej.

Jeżeli warunki w złożonej przesłance są połączone funktorami koniunkcji, to proces analizowania takiej przesłanki jest przerywany (z wynikiem negatywnym) po napotkaniu pierwszego niespełnionego warunku. W niektórych systemach w regułach może występować funktor alternatywny, np.:

IF padał deszcz OR jechała polewaczka THEN ulica jest mokra

Regułę taką można zmienić w zestaw dwóch równoważnych reguł nie zawierających funktora alternatywy:

IF padał deszcz THEN ulica jest mokra
IF jechała polewaczka THEN ulica jest mokra

Stosowanie spójnika OR jest dopuszczalne, lecz nie zalecane ze względu na większe skomplikowanie modułu wnioskującego.

Rodzaje reguł

Podział reguł ze względu na sposób uzyskiwania ostatecznych konkluzji w procesie wnioskowania:

- *reguły złożone* – to takie reguły, które umożliwiają bezpośrednie wyznaczanie wniosków przez system,
- *reguły proste* – to takie reguły, które mają postać wniosków pośrednich.

Przykład reguły złożonej:

IF są spełnione wszystkie warunki niezbędne do przyjęcia wniosku, że u pacjenta stwierdzono określoną jednostkę chorobową
THEN zastosować określoną terapię

Zaletą *reguł złożonych* jest fakt, że nie wymagają maszyn wnioskujących skomplikowanym sposobie działania; ponieważ każda z reguł w konkluzji zawiera jakiś wniosek końcowy. Wystarczy więc uaktywnić jedną regułę, aby osiągnąć wynik. Wadą takiego podejścia jest trudność formułowania odpowiedniego zbioru reguł oraz złożony sposób jego weryfikacji i uzupełniania.

Zaletą *reguł prostych* jest łatwość weryfikacji zbioru reguł i ograniczenie ich redundancji; wadą jest potrzeba realizacji złożonych działań przez maszynę wnioskującą, ponieważ trzeba uaktywnić wiele reguł. Kolejne udowodnione wnioski pośrednie tworzą łańcuch wnioskowania.

W systemach ekspertowych badanie elementów przesłanek jest najczęściej związane z licznymi efektami ubocznymi, np. analizowanie innych reguł, zadawanie pytań użytkownikowi itp. Dla podkreślenia znaczenia kolejności elementów przesłanki wprowadza się pojęcie kontekstu. Przykładem może być następująca reguła

$$A_1 \wedge A_2 \wedge A_3 \wedge \ldots \wedge A_{k-1} \wedge A_k \Rightarrow B$$

Jeśli zapiszemy tę regułę w postaci

$$C_k \wedge A_k \Rightarrow B$$

to $C_k = A_1 \wedge A_2 \wedge \ldots \wedge A_{k-1}$ jest kontekstem dla warunku A_k.

W podobny sposób $C_{k-1} = A_1 \wedge A_2 \wedge \ldots \wedge A_{k-2}$ jest kontekstem dla warunku A_{k-1}

$C_3 = A_1 \wedge A_2$ jest kontekstem dla A_3

$C_2 = A_1$ jest kontekstem dla A_2

Kontekst umożliwia jawne rozpatrywanie warunków koniecznych po to, aby badanie określonego elementu przesłanki było celowe. W celu ograniczenia liczby operacji wykonywanych przez system podczas sprawdzania przesłanek ważna jest odpowiednia kolejność reguł.

W systemach ekspertowych są stosowane reguły, które odnoszą się do konkretnych obiektów, jak też reguły obejmujące pewien zbiór obiektów – tzw. *reguły ogólne*. Mogą one być spełnione przez różne obiekty zbioru. Dzięki temu reguły ogólne umożliwiają znaczne zwiększenie stopnia ogólności utworzonej bazy wiedzy. Uzyskuje się to rozpatrując elementy stwierdzeń nie jako wartości stałe, lecz jako zmienne. Na przykład w przedstawionej regule

---

IF (@syn,jest_synem,@ojciec) AND (@ojciec,jest_synem,@dziadek)
THEN (@syn,jest_wnukiem,@dziadek)

---

napisy poprzedzone symbolem @ są traktowane jako zmienne.

Podczas wnioskowania zmienne są zastępowane odpowiednimi stałymi określonymi w wyniki dopasowania reguły do istniejącego zbioru stwierdzeń. Jest to proces tzw. unifikacji.

Dla powyższej reguły można na przykład zmienne zastąpić następującymi stałymi:

@syn => Jan, @ojciec => Adam, @dziadek => Jacek

---

co prowadzi do reguły

IF (Jan,jest_synem,Adam) AND (Adam,jest_synem,Jacek)
THEN (Jan,jest_wnukiem,Jacek)

---

w ten sposób na podstawie jednej reguły zawierającej zmienne można generować wiele reguł dopasowanych do danej bazy faktów.

Wektory wiedzy

Wektory wiedzy są pewnego rodzaju uogólnieniem reguł, w wyniku którego otrzymuje się zapis w postaci wektorowej. Aby zilustrować ten sposób reprezentacji wiedzy, przyjmuje się, że w wektorach występują trzy symbole: *, T, N.

Znaczenie poszczególnych symboli jest następujące:

* - dany warunek/wniosek w regule nie występuje

T - dany warunek/wniosek jest prawdziwy (tak)

N – dany warunek/wniosek jest fałszywy (nie)

W podejściu tym najpierw zapisuje się daną bazę w tradycyjny sposób, przy czym poszczególne reguły powinny zawierać jednakową liczbę warunków i wniosków. Następnie dokumentuje się kodowania poszczególnych członów reguł z wykorzystaniem trzech wprowadzonych symboli. W rezultacie zamiast pisać pełną reprezentację poszczególnych reguł, otrzymuje się bardzo zwarty opis w postaci wektorów, zawierających trzy wymienione symbole.

Aby zobrazować podstawową koncepcję tej reprezentacji rozważyć można przykład bazy wiedzy złożonej z trzech reguł.

R1 IF jest ładna pogoda THEN pójdę na spacer

R2 IF jestem zmęczony THEN nie pójdę na spacer

R3 IF pada deszcz THEN wezmę parasol

W regułach tych występuje pięć stwierdzeń, w tym również jedno w postaci zanegowanej.

Dlatego reguły te mogą być zakodowane za pomocą pięcioelementowych wektorów, przy czym każdemu stwierdzeniu będzie przypisana określona pozycja w wektorze. Poszczególnym stwierdzeniom zostały przyporządkowane następujące pozycje:

Pozycja w wektorze	Znaczenie symboli w warunkach i wnioskach
1	jest ładna pogoda
2	jestem zm czony
3	pada deszcz
4	pójd na spacer
5	wezm parasol

Wobec tego zostaną utworzone wektory, w których warunkom będą przypisane pozycje 1, 2, 3, natomiast wnioskom – pozycje 4, 5. Stosując wprowadzone wcześniej symbole reguły R1, R2, R3 mogą być zakodowane w następujący sposób:

Wartość symboli			Znaczenie symboli w warunkach i wnioskach
R1	R2	R3	
T	*	*	jest ładna pogoda
*	T	*	jestem zm czony
*	*	T	pada deszcz
T	N	*	pójd na spacer
*	*	T	wezm parasol

Tak więc zamiast pisać całą treść reguł można uzyskać uproszczoną reprezentację w postaci wektorów, co ilustruje poniższy zapis:

Dla reguły R1	Dla reguły R2	Dla reguły R3
T	*	*
*	T	*
*	*	T
T	N	*
*	*	T

Mając postać wektorową, łatwo można przejść na zapis zawierający pełną treść reguł. Wektory wiedzy są wygodne do weryfikacji poprawności bazy wiedzy. W wektorze wiedzy może być zestaw wszystkich pytań (warunków) jak w regule złożonej.

5. Rachunek predykatów

Podstawy rachunku predykatów

***Rachunek predykatów*:**

- **stanowi podstawę programowania w logice,**
- **jest rozszerzeniem rachunku zdań przez wprowadzenie kwantyfikatorów: „dla każdego” ∀ oraz „istnieje takie że” ∃**

Wyrażenie *W*(*x*), w którym występuje zmienna *x* i które staje się zdaniem prawdziwym lub fałszywym, gdy w miejsce *x* podstawimy wartość zmiennej *x*, nazywa się *funkcją zdaniową* lub *predykatem*.

***Predykat* składa się z nazwy i dowolnej liczby argumentów, które są nazywane *termami*. Termami mogą być stałe (symbole) alfanumeryczne, jak też numeryczne i zmienne oraz wyrażenia. W wyniku podstawienia stałych za zmienne otrzymuje się zdania prawdziwe lub fałszywe. Przyporządkowanie termom symboli, a nazwie relacji między obiektami definiuje semantykę języka predykatów. Zaletą predykatów są proste i zrozumiałe interpretacje wyrażania zdań.**

Język rachunku predykatów pierwszego rzędu

Podstawowe elementy języka predykatów pierwszego rzędu:

I. Alfabet teorii

Alfabet rachunku predykatów pierwszego rzędu tworzą:

A. Stałe – Const, które dzieli się na:

1) stałe oznaczające obiekty (symbole podstawowych elementów danego rachunku) – IConst

2) nazwy funkcji i-miejscowe (symbole funkcji) – FConst

3) nazwy predykatów i-miejscowe (symbole predykatów) – PConst

B. Zmienne – Var

C. Symbole operacji logicznych: $\neg$, $\wedge$, $\vee$, => oraz kwantyfikatory $\exists$ - istnieje, $\forall$ - dla każdego

II. Termy - Tm

Termy są argumentami predykatów. Są to stałe zmienne lub funkcje.

Używając definicji rekurencyjnej można opisać termy jako obiekty o następujących własnościach:

1) każda swobodna zmienna jest termem

2) każda stała oznaczająca obiekt jest termem

3) jeśli s,t ∈ Tm, przy czym (st) oznacza rezultat złożenia dwóch termów

4) jeśli f ∈ FConst, a t1,…,ti są termami, to f(t1,…,ti) też jest termem

III. Formuły atomowe - AForm

Formuły atomowe są podstawowymi elementarnymi formułami zbudowanymi na podstawie prostych predykatów bez użycia symboli operacji logicznych. Można je zdefiniować następująco:

jeśli p ∈ PConst i t1,…,ti ∈ Tm to p(t1,…,ti) ∈ AForm

IV. Formuły – Form

Formuły są elementami zbudowanymi z użyciem podstawowych symboli operacji logicznych. Można je zdefiniować następująco:

jeśli $A, B \in$ Form to $A, B, \neg A, \neg B, A \wedge B, A \vee B, A \Rightarrow B \in$ From

jeśli $A \in$ From i $x \in$ Var to $\exists_x A$ i $\forall_x A \in$ From

Ważną rolę odgrywają w tej metodzie tzw. *klauzule*, które są alternatywami literałów, przy czym literałem nazywa się predykat albo zanegowany predykat. Wśród formuł wyróżnia się pewien ich podzbiór, a mianowicie poprawnie zbudowane formuły rachunku predykatów, do których zalicza się literały oraz poprawnie zbudowane formuły rachunków predykatów połączone spójnikami logicznymi oraz objęte kwantyfikatorami.

Istnieje twierdzenie, które mówi, że każdy zbiór poprawnie zbudowanych formuł można przekształcić w zbiór klauzul. W procesie przekształcania występuje 9 etapów:

- usunięcie symbolu implikacji,
- ograniczenie do literału zakresu negacji,
- rozdział zmiennych,
- usunięcie kwantyfikatorów szczegółowych,
- przekształcenie w formę prefiksową,
- przekształcenie w koniunkcyjną postać normalną,
- usunięcie kwantyfikatorów ogólnych,
- usunięcie symbolu koniunkcji,
- przemianowanie zmiennych.

W celu zilustrowania wymienionego procesu przekształcania, została rozpatrzona następująca poprawnie zbudowana formuła rachunku predykatów:

$$\forall_x\{P(x) => \{\forall_y[P(y) => P(f(x,y))] \wedge \neg \forall_y[Q(x,y) => P(y)]\}\}$$

W pierwszym korku trzeba pozbyć się symbolu implikacji, korzystając ze znanego w logice przekształcenia ($U => W <=> U \vee W$) w wyniku czego otrzymuje się

$$\forall_x\{\neg P(x) \vee \{\forall_y[\neg P(y) \vee P(f(x,y))] \wedge \neg \forall_y[\neg Q(x,y) \vee P(y)]\}\}$$

W drugim kroku ograniczyć trzeba do literału zakres negacji

$$\forall_x\{\neg P(x) \vee \{\forall_y[\neg P(y) \vee P(f(x,y))] \wedge \Leftarrow_y[Q(x,y) \wedge \neg P(y)]\}\}$$

po czym należy dokonać rozdziału zmiennych, tzn. każdemu kwantyfikatorowi przypisuje się inną zmienną

$$\forall_x\{\neg P(x) \vee \{\forall_y[\neg P(y) \vee P(f(x,y))] \wedge \Leftarrow_z[Q(x,z) \wedge \neg P(z)]\}\}$$

W następnym kroku usuwamy kwantyfikator szczegółowy $\Leftarrow$

$$\forall_x\{\neg P(x) \vee \{\forall_y[\neg P(y) \vee P(f(x,y))] \wedge [Q(x,g(x)) \wedge \neg P(g(x))]\}\}$$

Piąty krok polega na przeniesieniu kwantyfikatorów ∀ na lewą stronę przekształcalnego wyrażenia, czyli uzyskanie formy prefiksowej

$$\forall_x \forall_y \{\neg P(x) \vee \{[\neg P(y) \vee P(f(x,y))] \wedge [Q(x,g(x)) \wedge \neg P(g(x))]\}\}$$

W 6 i 7 kroku dokonuje się przekształcenia w koniunkcyjną postać normalną. W tym celu korzysta się ze znanego przekształcenia z logiki

$$A \vee (B \vee C) \wedge D \wedge E <=> (A \vee B \vee C) \wedge (A \vee D) \wedge (A \vee E)$$

które po zastosowaniu w rozważanym przykładzie prowadzi do postaci

$$[\neg P(x) \vee \neg P(y) \vee P(f(x,y))] \wedge [\neg P(x) \vee Q(x,g(x))] \wedge [\neg P(x) \vee \neg P(g(x))]$$

Z ostatniego wzoru wyrażenia po usunięciu symbolu koniunkcji otrzymujemy postać klauzulową - istotną dla zastosowań w języku Prolog:

$$\neg P(x) \vee \neg P(y) \vee P(f(x,y))$$
$$\neg P(x) \vee Q(x,g(x))$$
$$\neg P(x) \vee \neg P(g(x))$$

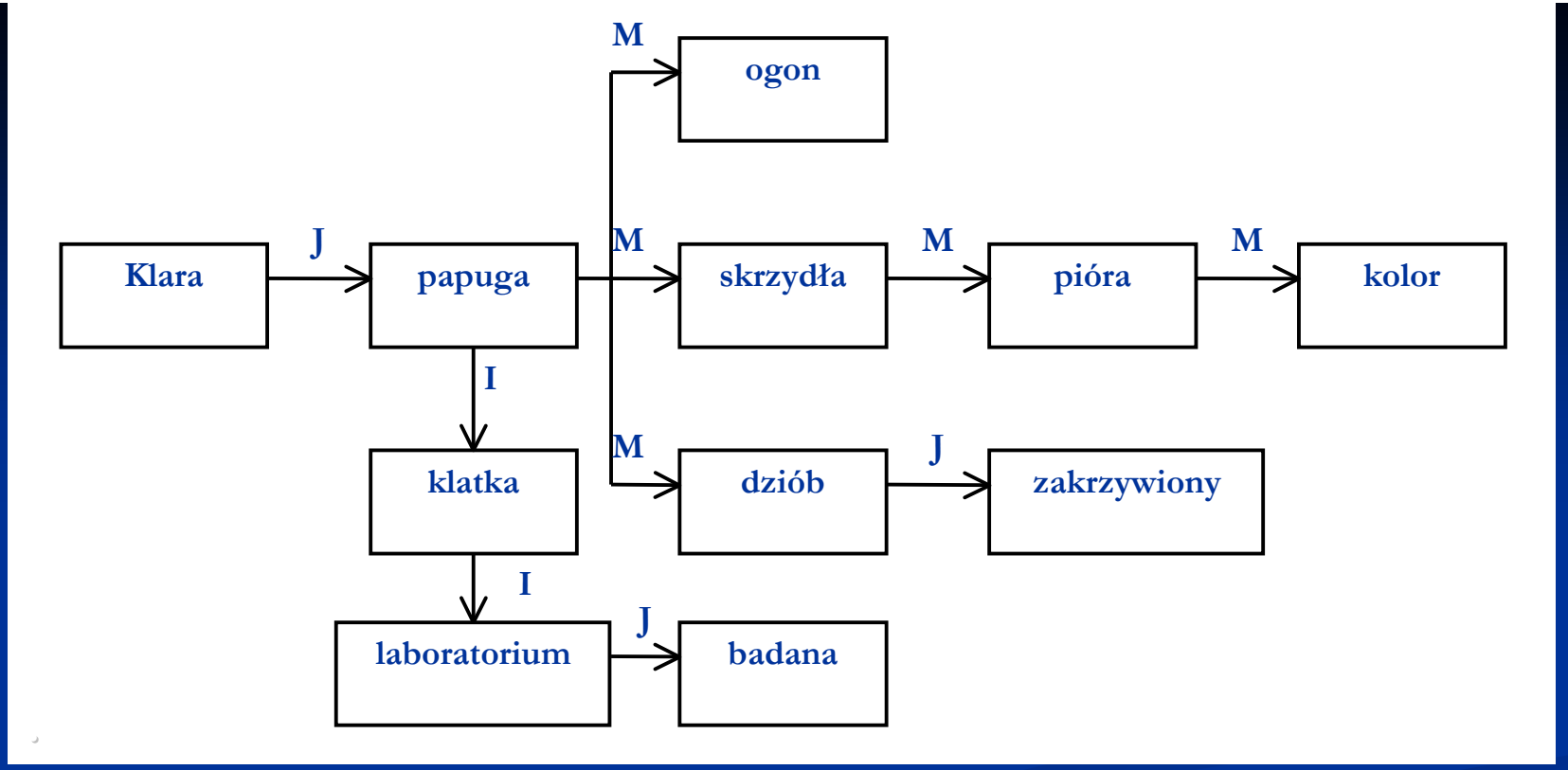
6. Sieci semantyczne

Sieć semantyczna:

- **jest uogólnioną koncepcją sieci stwierdzeń, polegającą na przyjęciu założenia, że węzły odpowiadają kompletnym opisom pojęć lub obiektów i nie są wyłącznie stwierdzeniami,**
- **jest pewnego rodzaju logiką, gdzie relacje między obiektami są przedstawione w postaci rysunku,**
- **została opracowana przez Quilliana.**

Quillian wyszedł z założenia, że pamięć ludzką najlepiej opisuje model asocjacyjny. Oznacza to, że każdy element jest zdefiniowany przez inny element. Powstaje w tym momencie pewna struktura powiązań. Struktura ta może być zamknięta, a grafy mogą być skierowane. Niestety, model ten nie ma ściśle zdefiniowanej syntaktyki, ponieważ Quillian ograniczył się tylko do podania koncepcji. Syntaktyka w tym przypadku zależy od konkretnej implementacji.

W sieciach semantycznych wnioskowanie odpowiada „poruszaniu się" po grafie. Na podstawie inspekcji sieci wprowadza się różne konkluzje. Jest tu więc jak gdyby wyrysowany mechanizm dedukcji.



Przykładowa sieć semantyczna

Powyższa sieć o nazwie Klara tworzy graf skierowany. Między poszczególnymi węzłami grafu zachodzą relacje: J – jest, M – ma, I – jest w. Skierowanie grafu oznacza, że relacje zachodzą w jedną stronę. Na przykład między węzłami Klara i papuga zachodzi relacja „jest". Tak więc na podstawie grafu można wyprowadzić stwierdzenie: Klara jest papugą. Nie można natomiast wyprowadzić stwierdzenia odwrotnego: papuga jest Klarą, co oznaczałoby, że papugi należą do pewnej klasy o nazwie Klara.

Model reprezentacji wiedzy za pomocą sieci semantycznych ma również swoje wady. Problemem jest tutaj na przykład określenie, czy węzły sieci oznaczają jeden obiekt czy klasę obiektów. Dlatego też sieci te wiąże się zazwyczaj ramami lub z regułami. Ramy w takim podejściu odpowiadają obiektom i opisują ich strukturę wewnętrzną, sieć semantyczna natomiast relacjom między ramami.

W sieci semantycznej mogą także występować pewne pułapki dedukcyjne. W przykładzie z poprzedniego slajdu możemy wydedukować: Klara jest badana. Stwierdzenie to nie musi być prawdą. Prawdziwość tego stwierdzenia może zachodzić wtedy, kiedy Klara znajdzie się w laboratorium.

Metoda sieci semantycznych jest często stosowana w systemach analizy i rozumienia języka naturalnego. Wynika to z jasności, z jaką można przedstawiać zawiłe struktury składniowe języka za pomocą tej reprezentacji. Sieci semantyczne są także przydatne do tłumaczenia z jednego języka na inny oraz wspomagania uczenia.

7. Reprezentacja wiedzy za pomocą ram

***Ramy*:**

- **umożliwiają deklaratywną i proceduralną reprezentację wiedzy,**
- **stwarzają one możliwość organizacji bazy wiedzy w taki sposób, że reguły będące reprezentacją wiedzy danej dziedziny są wyraźnie oddzielone od reguł niezbędnych do poprawnego działania systemu ekspertowego,**
- **stwarzają możliwość grupowania informacji dotyczących wybranego fragmentu wiedzy w postaci jednej ramy, co upraszcza późniejszą weryfikację.**

Za twórcę ram uważa się Minsky'ego, który użył tej metody reprezentowania wiedzy do rozpoznawania obrazów. Nazwał on ramą strukturę danych opisującą pewien obiekt, w którym mieszczą się wszystkie typowe i oczekiwane informacje, a także przypuszczenia o tym obiekcie.

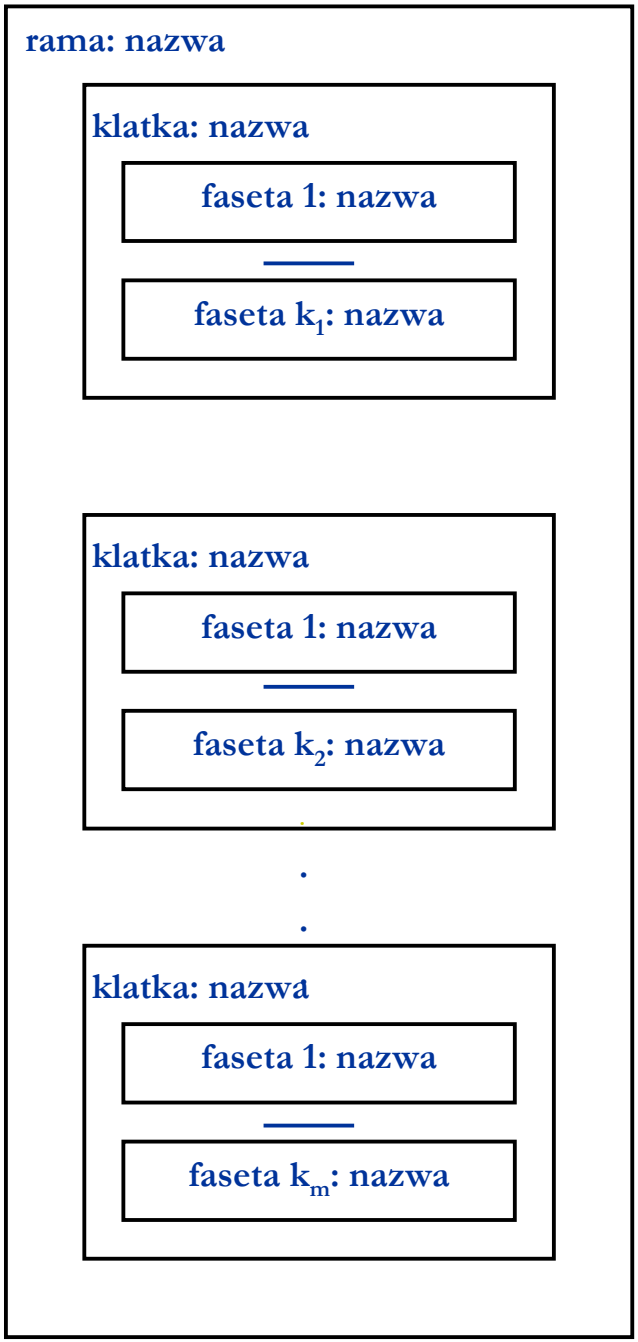
Minsky oparł swój plan na analizie sposobu zachowania człowieka znajdującego się w nowej dla niego sytuacji i w nowym otoczeniu, ale mającego o tej dziedzinie już pewne wcześniejsze wyobrażenia. Człowiek wydobywa wówczas pamięci określoną strukturę, czyli ramę, i konfiguruje tę sytuację z wiedzą zawartą w ramie. Z kolei, gdy człowiek zetknie się z całkowicie nowym obiektem, wówczas jego pierwszą reakcją będzie próba zapamiętania go i wprowadzenia jego nazwy.

Struktura ramy

Rama jest strukturą opisującą dany obiekt i składa się z procedur, tzw. *klatek*, nazywanych też slotami (slots). Każda klatka reprezentuje pewną właściwość albo cechę obiektu, opisywanego przez ramę. Ramy są podobne do reprezentacji wiedzy za pomocą stwierdzeń: (<obiekt> <atrybut> <wartość>). W tym przypadku jest to jednak szerszy opis, ponieważ wartość zapisana w klatce jest jednym z wielu możliwych elementów klatki. Klatka dzieli się na mniejsze części – *fasety*.

Każda rama, klatka i faseta musi mieć swoją nazwę. W ramie nie mogą występować dwie klatki o tej samej nazwie, podobnie w klatce nie może być dwóch faset o tej samej nazwie. Natomiast w różnych ramach mogą występować te same rodzaje klatek, to samo dotyczy faset. Mogą one być liczbami, tekstami, piktogramami lub kolejnymi ramami. Rzeczywistą daną, wpisywaną bezpośrednio do klatki, nazywa się *wartością klatki*, która jest zapisana w fasecie typu VALUE.

Ogólnie, każdej klatce jest przyporządkowany określony zbiór faset, w których mogą być zapisane na przykład warunki dotyczące uznania wartości atrybutu jako wartości dopuszczalnej, procedury pozyskiwania wartości itp.



Ogólny schemat ramy

Zbiór faset może być dowolnie uzupełniany fasetami wynikającymi z potrzeb specyficznego zastosowania. Ramy mogą być łączone w strukturę hierarchiczną (graf-drzewo). Wierzchołkami tego grafu są ramy, a jego gałęzie określają relację podrzędności ram. Relacja podrzędności oznacza dziedziczenie właściwości obiektu określonego ramą nadrzędną przez obiekt określony ramą podrzędną.

Wprowadza się klatkę o nazwie „podrzędny względem", zawierającą nazwę ramy nadrzędnej. W ten sposób uzyskuje się odwołania do innych ram. Wprowadzenie mechanizmu dziedziczenia znacznie ogranicza redundancję baz wiedzy.

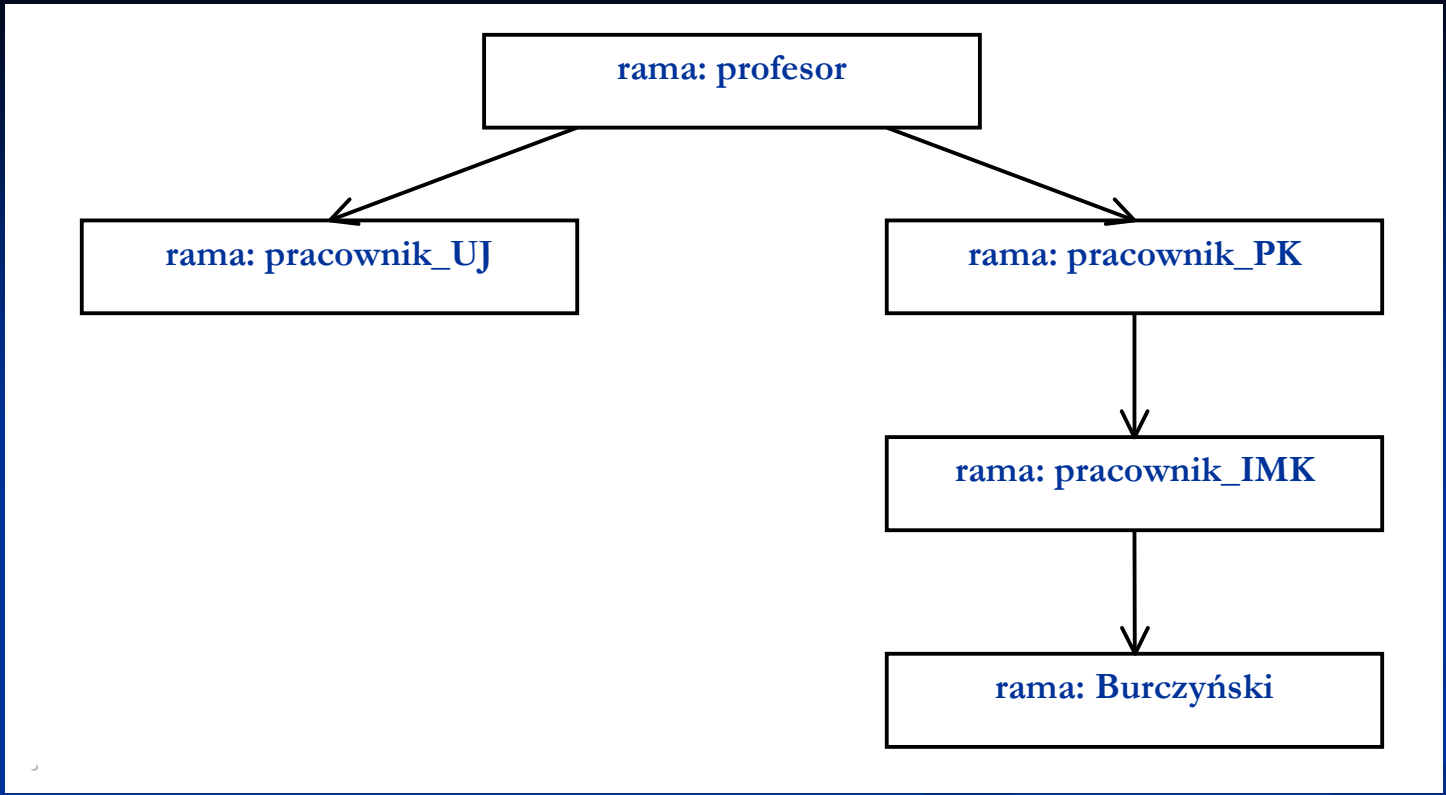
Powiązania pomiędzy ramami definiują tzw. *hierarchię dziedziczenia* (inheritance hierarchy), w której dane zapisane w klatkach jednej ramy mogą być dziedziczone przez drugą. W ten sposób można opisać zawieranie się pojęć, reprezentowanych przez ramy. Jako przykład rozważony został pewien zbiór ram, dla którego hierarchię dziedziczenia określono w następujący sposób (grot strzałki wskazuje ramę podrzędną):

profesor => pracownik_PK

profesor => pracownik_UJ

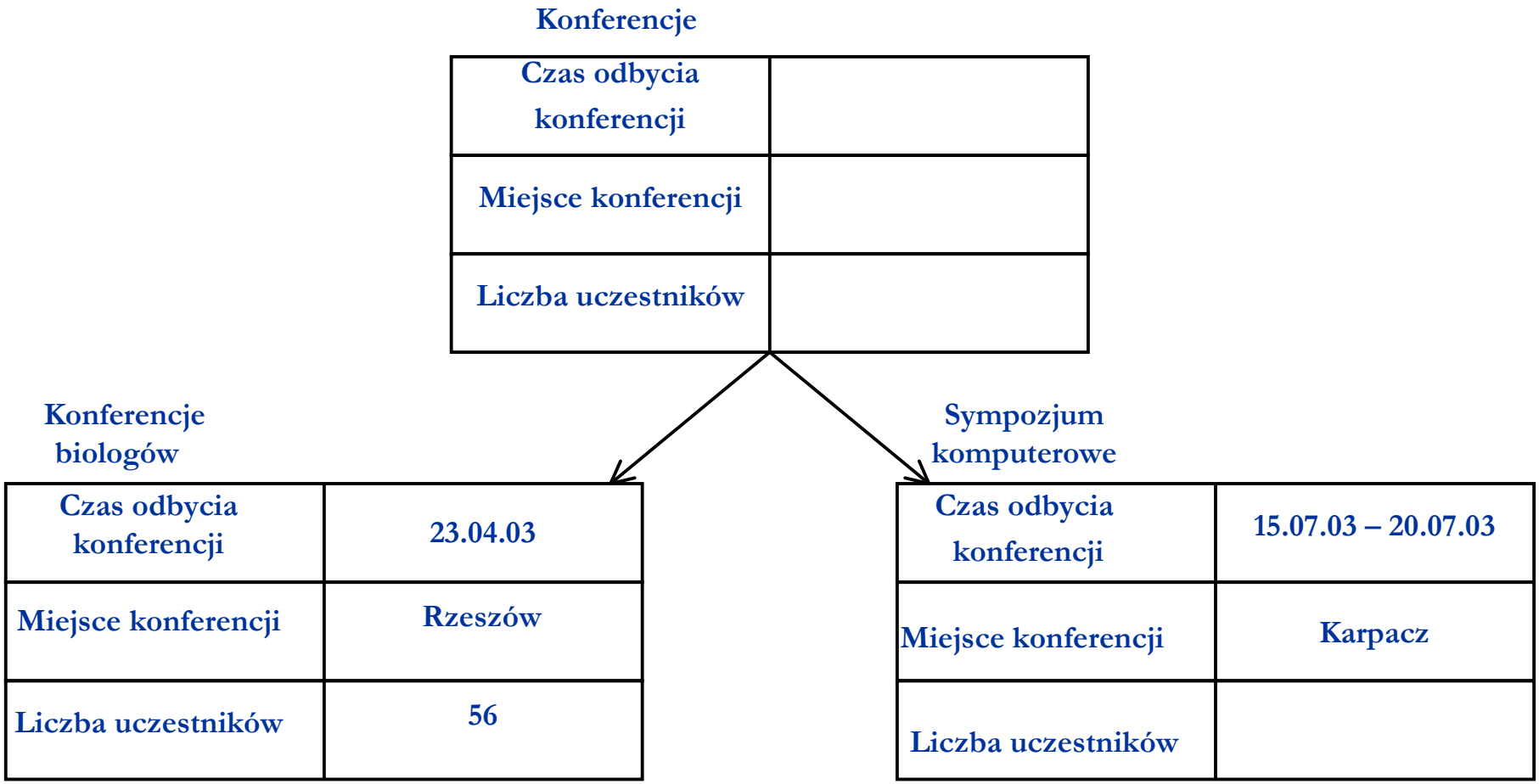
pracownik_PK => pracownik_IMK

pracownik_IMK => Burczyński



Przykładowa struktura hierarchiczna ram

Z powyższego zestawienia można wywnioskować, że rama Burczyński dziedziczy dane zapisane w klatkach ramy pracownik_PK, a ta z kolei dziedziczy dane zapisane w klatkach ramy profesor. Tak więc cech zapisane w ramie profesor mogą być dziedziczone przez ramę Burczyński. W najprostszym przypadku rozważa się jeden rodzaj powiązań, w którym występują tylko dwa rodzaje ram: ramy prototypy i ramy egzemplarze. *Rama prototyp* zawiera ogólny szkielet opisu danego obiektu.



Rama prototyp i ramy egzemplarze

W procesie tzw. konkretyzacji prototypu dochodzi do utworzenia, na podstawie rzeczywistych danych, egzemplarza prototypu. Aby to zilustrować, rozważona została rama prototyp o nazwie konferencje. Zakłada się, że rama ta zawiera trzy klatki: czas_odbycia_konferencji, miejsce_konferencji, liczbę_uczestników. Korzystając z tak zdefiniowanej ramy można określić ramy egzemplarze, które powstaną z danego prototypu w wyniku wypełnienia jego klatek konkretnymi informacjami, co zilustrowano na poprzednim slajdzie.

Na przykład rama o nazwie konferencja_biologów została wypełniona danymi: czas_odbycia_konferencji – 23.04.03, miejsce_konferencji – Rzeszów, liczba_uczestników – 56. Inna rama o nazwie sympozjum_komputerowe zawiera konkrety podane tylko dla dwóch klatek, a mianowicie czas_odbycia_konferencji – 15.07.03, 20.07.03 oraz miejsce_konferencji – Karpacz. Klatka o nazwie liczba uczestników nie została jeszcze wypełniona i można to będzie uczynić później, kiedy będą przesyłane zgłoszenia dla uczestników.

Oprócz powiązań pomiędzy ramami, inną szczególnie uwypuklaną właściwością ram jest możliwość przypisania im procedur. Powiązanie procedur z danymi polega na tym, że klatka oprócz wartości może zawierać odwołanie do procedur. Procedury te są zazwyczaj związane z odpowiednimi fasetami.

Podstawowe właściwości ram

Rozwijając sformułowaną definicję można opisać ramę jako pewnego rodzaju reprezentację wiedzy o następujących właściwościach:

- rama jest strukturą danych, opisującą pewien obiekt albo klasę obiektów i oferującą dostęp do pełnej informacji o tym obiekcie,
- rama jest zbiorem klatek. Całość informacji o obiekcie, zawarta w ramie, jest dzielona na części będące wartościami klatek ramy,
- każda klatka odpowiada pewnej właściwości danego obiektu i jest określonego rodzaju, tzn. ma zdefiniowaną dziedzinę wartości, które mogą być w niej umieszczone,
- w zależności od dziedziny wartości zawartych w klatce istnieją różne rodzaje klatek. Klatka jest jednoznacznie zdefiniowana przez podanie jej rodzaju, jak również do jakiej ramy należy,
- wartości klatki mogą być różne. W szczególności wartością klatki może być odwołanie do innej ramy albo dowolna procedura lub funkcja,
- klatki są dodatkowo dzielone na fasety, zawierające wybrane wartości klatki,
- ramy, klatki i fasety są identyfikowane za pomocą nazw.

Każda klatka może zawierać wiele typów faset, z których każda jest pewną wartością klatki albo określoną funkcją.

Poniżej wyszczególnione zostały z krótkim opisem najważniejsze rodzaje faset spotykane w typowych systemach z reprezentacją wiedzy za pomocą ram:

- ***Faseta typu VALUE*** - w fasecie tej jest zawsze wpisana bieżąca, rzeczywista wartość klatki. Podczas próby odczytania zawartości klatki zawsze najpierw jest przeszukiwana faseta VALUE,

- ***Faseta typu DEFAULT*** - faseta ta zawiera tzw. domyślną wartość klatki. Jest to wartość stereotypowa dla obiektów opisywanych przez daną ramę. Faseta ta jest brana pod uwagę jedynie wówczas, gdy faseta VALUE jest nie wypełniona lub nie istnieje,

- ***Faseta typu REQUIRE*** - zawartość tej fasety ustala ograniczenia dla wartości, które są umieszczane w fasecie VALUE. Opisuje ona dziedzinę wartości możliwych do wpisania do klatki i najczęściej sprowadza się do określania górnej i dolnej wartości granicznej. Zawartość tej fasety jest wykorzystywana przez fasetę IF-ADDED,

- ***Faseta typu COMMENT*** - faseta ta zawiera tekst będący opisem klatki,

- ***Faseta typu CARDINALITY*** - podaje ona liczbę faset typu VALUE występujących w klatce,

- ***Faseta typu RANGE*** - zawiera ona listę lub zakres dopuszczalnych wartości fasety VALUE.

- ***Faseta typu IF-NEEDED*** - zawiera funkcję, która wyznacza nieznaną wartość klatki poszukując jej w ramach nadrzędnych w stosunku do ramy wyjściowej. Faseta ta jest poszukiwana w klatce, jeżeli fasety VALUE i DEFAULT są nie wypeł-nione albo nie istnieją. Faseta IF-NEEDED jest często kojarzona z fasetą DEFAULT, gdyż jej rola jest podobna. W przeciwieństwie jednak do fasety DEFAULT, która zawiera wprawdzie domyślną, ale jednak konkretną wartość, faseta IF-NEEDED jest funkcją, która oblicza i udostępnia tę wartość,

- ***Faseta typu IF-ADDED*** - faseta ta zawiera funkcję, która wpisuje wartości do klatki i jest poszukiwana w klatce wówczas, gdy próbujemy wstawić do klatki nową wartość,

- ***Faseta typu IF-REMOVED*** - faseta ta zawiera funkcję usuwania wartości z klatki i jest poszukiwana w klatce, jeśli tylko próbujemy wymazać jej wartość,

Wymienione nazwy nie są obligatoryjne. Mogą wystąpić jeszcze inne nazwy w zależności od przeznaczenia ramy i spełnianych przez nią funkcji. W stosunku do typowych klatek zawierających konkretne wartości wyróżnia się klatki określające dziedziczenie i wzajemne powiązania pomiędzy ramami.

W praktycznych zastosowaniach można spotkać następujące klatki spełniające tę funkcję:

- *Klatka AKO* (A Kind Of, czyli podrzędny względem) - klatka ta początkowo zawierała jedynie nazwę ramy nadrzędnej w stosunku do danej ramy. Odwołanie do jednostki nadrzędnej odpowiadało bezpośrednio powiązaniom typu IS-A lub KIND-OF związanymi z relacją uogólnienie-wyszczególnienie np. „kanarek IS-A ptak" lub „kanarek jest KIND-OF ptak". Obecnie przyjęło się oznaczać przez AKO klatkę zawierającą całość informacji o dziedziczeniu ram. Zakłada się więc, że każda rama ma klatkę AKO ze standardowym zestawem faset, opisujących w pełni dziedziczenie danej ramy, a więc odwołania zarówno do uogólnień, jak i wyszczególnień,

- *Klatka INSTANCE* (Przykład) - klatka ta zawiera listę ram stanowiących egzemplarze danej ramy, a więc jest odwołaniem do jej egzemplarzy. Egzemplarz ramy stanowi specjalny rodzaj wyszczególnienia, dlatego często nie odróżnia się tej klatki od klatek AKO. Klasyczna klatka AKO dotyczy raczej zawierania się pojęć niż odwołań do egzemplarzy. Dlatego wydaje się czasami uzasadnione wyróżnienie klatki INSTANCE jako wyraźnego odwołania jedynie do konkretnych egzemplarzy ramy,

- *Klatka CLASS* (Klasa) - klatka ta zawiera standardowo informację o tym, czy dana rama jest egzemplarzem, czy też prototypem reprezentującym pewną klasę. Ten rodzaj klatki bywa czasem zaliczany do klatek typu AKO lub INSTANCE. Często jednak powiązania prototyp-egzemplarz są wyraźnie odróżniane od powiązań opisujących zawieranie się klas i wówczas klatka CLASS jawnie opisuje typ ramy,

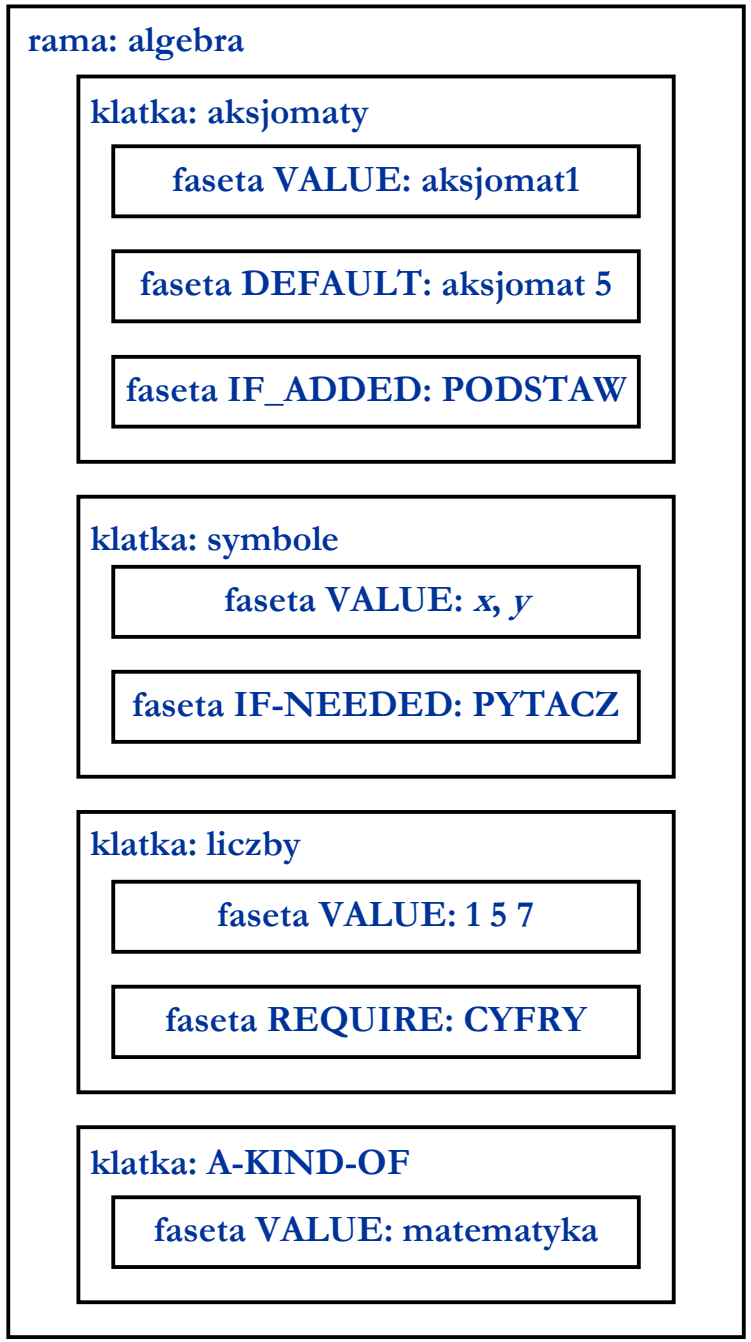
- ***Klatka PART-OF*** **(Część) - klatka ta odpowiada relacji całość-część np. „ściana jest PART-OF pokój". Klatka PART-OF zawiera odwołania do ram stanowiących części danej ramy. W tym odwołaniu nie ma jednak dziedziczenia. Na przykład rama „podwozie" może być PART-OF innej ramy „samochód", ale będzie zawierała klatki o zupełnie odmiennej strukturze.**

Bez względu na wybór reprezentacji ram, można ogólnie wyróżnić dwa podstawowe rodzaje dziedziczenia właściwości:

1. Rama egzemplarz dziedziczy właściwości ramy prototyp, a sama może mieć pewne właściwości dodatkowe, które ją uściślają, odróżniają od prototypu i wyszczególniają w miarę schodzenia do coraz niższych poziomów opisu,

2. Rama prototyp ma wszystkie właściwości, a kolejne ramy, na niższych poziomach opisu dziedziczą tylko ich część.

W celu wyjaśnienia rodzajów wprowadzonych faset i klatek na następnym slajdzie został przedstawiony przykład struktury ramy o nazwie algebra.



Przykładowa struktura ram

W ramie tej występują cztery klatki o nazwach aksjomaty, symbole, liczby, A-KIND-OF. Klatka aksjomaty zawiera trzy fasety o typach VALUE, DEFAULT, IF-ADDED. Są w nich zapisane odpowiednio następujące wartości: aksjomat1, aksjomat5, PODSTAW. Ostatnia z wartości podaje nazwę procedury, która będzie wywoływana, gdy użytkownik będzie chciał wstawić nową wartość do klatki aksjomaty. Jeżeli z klatki symbole będą potrzebne wartości x, y w momencie, kiedy faseta VALUE będzie pusta, to będzie wywołana procedura nazwie PYTACZ. Z kolei klatka o nazwie A-KIND-OF zawiera tylko jedną fasetę typu VALUE. Podaje ona nazwę ramy nadrzędnej, którą w tym przypadku jest „matematyka

Wnioskowanie w systemach ramowych

Ramy w swej podstawowej strukturze nie zawierają jawnie specjalnych mechanizmów sterowania wnioskowaniem. Dzięki jednak powiązaniom i procedurom zawartym w klatkach ramy spełniają zasady opisu proceduralnego. I właśnie te mechanizmy dają możliwość wykorzystania ram przy wnioskowaniu. Tak więc dzięki dziedziczeniu jest możliwe prowadzenie procesu wnioskowania, ponieważ właściwości obiektów nadrzędnych przechodzą na obiekty podrzędne. Mechanizm ten jest wykorzystywany w procesie tworzenia systemów ekspertowych opartych na tego typu reprezentacji wiedzy.

Sterowanie wnioskowaniem w ramach może się odbywać przez zwykłe procedury, lub też z wykorzystaniem mechanizmów wzajemnych powiązań i dziedziczenia. Większość ramowych metod wnioskowania sprowadza się do prostego dziedziczenia. W celu zwiększenia informacji zawartych w ramach, a także potwierdzania i odzyskiwania faktów mieszczących się w nich używa się połączeń przez członkostwo lub poprzez podklasę oraz prototypowe opisy klas członkowskich.

Również klasa ram, czyli te ramy, które reprezentują klasę, może mieć połączenie przez podklasę do jednej lub więcej innych klas ram. Ramowe automatyczne metody wnioskowania mogą również używać konstrukcji określających, czy dana może być wartością klatki. Na przykład przy umieszczaniu wartości w klatce zostanie ona odrzucona, jeśli przekroczy się górny limit liczby możliwych wartości klatki albo, jeśli wartość nie będzie członkiem klasy wartości klatki.

Systemy oparte na ramowej reprezentacji wiedzy potrafią zwiększać zbiór swoich faktów przez automatyczne otrzymywanie wniosków jako fragmentów niektórych stwierdzeń oraz przez operację wyszukiwania. Takie wnioski, bazujące na strukturalnych właściwościach reprezentacji ramowej oraz jej systematyce, mogą odgrywać czasami znaczącą rolę w rozumowaniu systemu z bazą wiedzy, ponieważ w tym przypadku są one „przywiązane" do mechanizmów danej reprezentacji i mają znacznie ograniczony zakres.

Rozszerzona reprezentacja ramowa

Podział klatek ze względu na charakterystyczne operacje wykonywane na nich:

- pobranie nazwy rodzaju klatki,
- pobranie formatu rodzaju klatki,
- odczytanie typu wejściowego dla klatek danego rodzaju,
- odczytanie strategii dziedziczenia klatek danego rodzaju,
- utworzenie nowego rodzaju klatek,
- wykreowanie nowej klatki danego rodzaju itp.

Takie podejście skłania do potraktowania *rodzaju klatki* jako odrębnej struktury danych. Struktura ta powinna opisywać rodzaj klatki i być zapisana również w postaci ramy. Ponadto z praktycznego punktu widzenia interesujące i efektywne okazuje się takie podejście do bazy wiedzy, w którym jest ona rozpatrywana w postaci dwóch odrębnych części. Jedną część stanowią podstawowe obiekty bazy wiedzy, a więc pojęcia reprezentowane przez ramy. Część druga to cała reszta, a więc system funkcji i operacji na ramach i klatkach, rodzaje ram i klatek, mechanizmy rozumowania i reguły dziedziczenia. Elementy tej części powinny być traktowane jako osobne dane programu i również pamiętane w ramach.

Kos (kind of slot)- to rama przedstawiająca wszystkie standardowe i wspólne cechy klatek danego rodzaju. Typowy kos jest uogólnieniem i powinien zawierać minimum takich informacji jak:

- nazwę rozszerzonej reprezentacji ramowej,
- typ wejściowy do klatki opisywanej przez kos,
- format wewnętrzny wartości przechowywanej w klatce opisywanej przez kos.

Każda klatka będzie teraz jednoznacznie zdefiniowana przez ramę, która zawiera określony kos. Niech będzie to przykładowo rama o nazwie F oraz zawarty w niej kos o nazwie K, co zapisuje się jako F.K. Jeśli rozważy się teraz klatkę „prędkość.jednostki", będącą klatką „jednostki" ramy „prędkość", to odpowiednim kosem będzie w tym przypadku rama „jednostki" z takimi klatkami jak typ wejściowy, format wewnętrzny itp.

Taki formalizm z jawnymi kosami ujawnia cechy, umożliwiające wiele ciekawych rozwiązań. Szczególnie interesująca wydaje się możliwość przeniesienia do kosów większości funkcji związanych z operowaniem na wartościach klatek danego rodzaju. Przy takim podejściu baza wiedzy może być rozdzielona na dwie odrębne części: zbiór ram reprezentujących podstawowe pojęcia i obiekty opisywanej rzeczywistości oraz zbiór ram kosów odpowiadających wszystkim rodzajom klatek ramowych.

8. Modele obliczeniowe

Reprezentacja zadań obliczeniowych

Zadania obliczeniowe zawierają zmienne, które będą oznaczane za pomocą identyfikatorów, np.: *AX*, x_1, *x*, POLE, itp. Można przyjąć, że wartościami zmiennych są liczby. Zadania obliczeniowe będą przedstawiane w następującej postaci:

oblicz $y_1,...,y_n$ przy $x_1,...,x_m$ znając *M*

W zapisie tym poszczególne identyfikatory oznaczają:

- zmienne wejściowe: $x_1,...,x_m$
- zmienne wyjściowe: $y_1,...,y_n$
- zmienna *M*, której wartość reprezentuje warunki zadania

Miedzy warunkami zadania, oznaczonymi literą *M*, a zmiennymi wejściowymi i wyjściowymi nie ma bezpośredniego związku. Zakłada się jednak, że warunki te zawierają wszystkie niezbędne opisy zmiennych obu rodzajów. W szczególności przyjmuje się, że warunki zadania wyznaczają zbiór zmiennych, które mogą być użyte jako zmienne wejściowe lub wyjściowe. Dane składające się na wartość zmiennej *M* reprezentują wiedzę potrzebną do rozwiązania zadania.

Poniżej rozważony został przykład zadania obliczeniowego. Należy obliczyć pole trójkąta, znając wszystkie jego boki. Zadanie można opisać za pomocą zmiennych *S, a, b, c* reprezentujących odpowiednio pole i trzy boki trójkąta:

oblicz *S przy a, b, c* znając trójkąt

Pojęcie trójkąt powinno być opisane przed przystąpieniem do rozwiązywania zadania i musi zawierać te same zmienne *S, a, b, c* na oznaczenie pola i boków trójkąta.

Koncepcja modeli obliczeniowych

Relacje mogą być obdarzone różnymi właściwościami rozpoznawanymi przez system, w którym są użyte. Przyjąć można za punkt wyjścia algebraiczną definicję relacji. Niech $s_1,...,s_k$ będą zbiorami. Iloczyn kartezjański tych zbiorów jest zbiorem złożonym ze wszystkich uporządkowanych kratek $<a_1,...,a_k>$, w których $a_1 \quad s_1$,..., $a_k \quad s_k$. Każdy podzbiór iloczynu kartezjańskiego jest relacją nad $s_1,...,s_k$. Można powiedzieć, że $<l_1,...,l_k>$ spełnia relację *R* wtedy i tylko wtedy, gdy $<l_1,...,l_k>$ należy do *R*.

Najczęściej są stosowane relacje, dla których $k = 2$. Takie relacje nazywa się relacjami binarnymi. Powszechnie są znane niektóre binarne relacje liczbowe, ta-kie jak =, <, >, <=, >=, różne. Wiadomo także, jak opisywać relacje za pomocą równań.

Niech $x_1,...,x_k$ będą zmiennymi rzeczywistymi, a funkcja f – k-argumentową funkcją rzeczywistą. Równanie $f(x_1,...,x_k) = 0$ opisuje pewną k-argumentową relację nad zbiorem liczb rzeczywistych. Stwierdzić można również, że równanie $f(x_1,...,x_k) = 0$ reprezentuje relację wiążącą zmienne $x_1,...,x_k$. Niech $x_1,...,x_k$ będą zmiennymi, których zakresami są odpowiednio zbiory $m_1,\ldots,m_k$. O relacji R zawierającej się w iloczynie kartezjańskim zbiorów $m_1,\ldots,m_k$ powiedzieć można, że wiąże zmienne $x_1,...,x_k$ i oznacza się ją $R(x_1,...,x_k)$ lub $R(x)$, przy czym $x = < x_1,...,x_k>$.

Przyjmuje się dla uproszczenia dodatkowe założenia:

- zbiór wszystkich zmiennych jest liniowo uporządkowany, a więc dla dowolnych dwóch zmiennych x_1, x_2 zachodzi albo $x_1 < x_2$, albo $x_2 < x_1$. Relacją porządkującą może być tutaj np. porządek leksykograficzny w zbiorze nazw zmiennych
- będą rozważane jedynie takie krotki $<x_1,...,x_k>$, w których $x_1 < x_2 < ... < x_k$
- dla dowolnego skończonego zbioru m zmiennych $\{x_1,...,x_m\}$ istnieje zmienna $x = < x_1,...,x_m>$ będąca krotką złożoną ze zmiennych z danego zbioru
- każda relacja wiąże zmienne zgodnie z zadanym uporządkowaniem, a więc jeżeli $R(x_1,...,x_k)$, to $x_1 < ... < x_k$

Dzięki tym założeniom operacje teoriomnogościowe można rozszerzyć na skończone ciągi zmiennych. Rozważono relację $R(x)$ wiążącą zmienne $x_1,...,x_k$. Niech u, v będą ciągami zmiennych takimi, że u zawiera się w x i v zawiera się w x.

Relacja $R(x)$ wyznacza przekształcenie (na ogół wieloznaczne), które danej wartości u zmiennej u przyporządkowuje dokładnie te wartości v zmiennej v, które wraz z u spełniają relację R. Zbiór indukowanych przekształceń prezentuje potencjalne możliwości znalezienia wartości dowolnej zmiennej v z danej wartości dowolnej innej zmiennej u, tzn. reprezentuje potencjalną możliwość obliczenia zmiennych związanych relacją R. Rzeczywiste obliczenie wartości w wyniku zastosowania przekształcenia jest możliwe jedynie dla prostych rodzajów relacji.

Na przykład nie można na ogół wykorzystać w obliczeniach nieskończonej relacji, ale nawet dla zbiorów skończonych czasochłonność obliczeń może być duża. Samą relację można jednak opisać za pomocą indukowanych przez nią przekształceń. W szczególności relacja $R(x)$ jest definiowana jednoznacznie dowolnym przekształceniem. Relacje można przedstawić w bardzo różnych postaciach: tablic, grafów, równań itp. W celu użycia relacji do obliczeń, należy je reprezentować odpowiednimi przekształceniami wykorzystując operatory.

Operator jest zbiorem przypisań

$y_i = f_i(x_1,\ldots,x_m) \quad i = 1,\ldots,n$

przy czym $x_1,\ldots,x_m$ - zmienne wejściowe; $y_i,\ldots,y_n$ - zmienne wyjściowe

Zastosowanie operatora polega na wykonaniu wszystkich jego przypisań. Jeśli nie pojawią się inne założenia, to należy przyjmować, że wartość operatora jest określona dla dowolnych wartości zmiennych wejściowych. *Zbiór operatorów* 1,...,*k* można nazwać *jednoznacznym,* jeśli wynik zastosowania do danego zbioru zmiennych wejściowych jakiegokolwiek ciągu tych operatorów daje zawsze tą samą wartość dla każdej zmiennej. Na przykład operatory:

$$x := y + 1, \qquad y := x - 1$$

tworzą zbiór jednoznaczny, podczas gdy operatory:

$$x := y + 1, \qquad y := x - 2$$

nie tworzą zbioru jednoznacznego. Jednoznaczny zbiór operatorów nazywa się *relacją częściową.* Zbiorem zmiennych związanych taką relacją jest suma zbiorów zmiennych wejściowych i zmiennych wyjściowych jej operatorów.

Proste modele obliczeniowe

Modele obliczeniowe – specjalne sieci reprezentujące wiedzę dotyczącą możliwości obliczania, czyli wyznaczania wartości zmiennych na podstawie zbioru operatorów wiążących te zmienne.

Modele obliczeniowe służą do rozwiązywania prostych problemów z dziedzin elementarnych, np. matematyki, fizyki. Prosty model obliczeniowy składa się ze zbioru zmiennych i zbioru relacji częściowych wiążących te zmienne. Dla takiego modelu obliczeniowego M przyjmuje się, że var(M) oznacza zbiór jego zmiennych, a rel(M) - zbiór jego relacji.

Modele obliczeniowe należy reprezentować jako sieci - w postaci graficznej. Zmienne i relacje będą węzłami sieci. Jeśli zmienna jest związana pewną relacją częściową, to węzeł odpowiadający tej zmiennej należy połączyć łukiem z węzłem odpowiadającym relacji. Jeśli z modelu obliczeniowego usunie się relację zawierającą k operatorów i w zamian doda się k nowych relacji, każdą z jednym operatorem, to otrzyma się nowy model o tej samej mocy obliczeniowej. Tak więc można się ograniczyć do prostych modeli obliczeniowych zawierających jedynie relacje z jednym operatorem.

W celu zilustrowania sposobu tworzenia modelu obliczeniowego rozważono następujący przykład:

Ciało o masie m porusza się po prostej ze stałym przyspieszeniem a w czasie t. Dla $t < 0$ ma prędkość początkową v_0, a po czasie t prędkość końcową v. Przebyta w czasie t droga wynosi s, siła bezwładności wynosi F. Energia kinetyczna ciała dla $t = 0$ wynosi E_1, a po czasie t wynosi E_2. Przedstawiony opis słowny można wyrazić w postaci następujących równań:

$$s = v_0 t + (at^2)/2$$
$$v = v_0 + at$$
$$F = ma$$
$$E_1 = [m(v_0)^2]/2$$
$$E_2 = (mv^2)/2$$

Te równania i zmienne tworzą model obliczeniowy, który zawiera wiedzę o ruchu przyśpieszonym. Oznacza się go identyfikatorem RUCH. Model ten może posłużyć do rozwiązywania wielu zadań dotyczących ruchu przyspieszonego. Jako przykład rozważono zadanie:

Ciało o masie m porusza się prostoliniowo w czasie t ze stałym przyspieszeniem a, osiągając prędkość v. Znaleźć prędkość początkową v_0, siłę F potrzebną do uzyskania przyspieszenia, energię kinetyczną początkową ciała E_1 i energię kinetyczną końcową E_2. Zadanie można przedstawić w postaci:

oblicz v_0, F, E_1, E_2 przy m, t, a, v znając RUCH

Przystępując do rozwiązania tego zadania należy pamiętać o tym, że jedynymi środkami obliczeniowymi są operatory relacji częściowych modelu obliczeniowego RUCH.

Należy również wziąć pod uwagę, że:

- jedynym warunkiem zastosowania dowolnego operatora prostego modelu obliczeniowego jest istnienie określonych wartości wszystkich jego zmiennych wejściowych,
- zastosowanie operatora jest celowe tylko wówczas, gdy umożliwia obliczanie nie znanej dotychczas wartości zmiennej wyjściowej lub innej zmiennej potrzebnej do obliczenia szukanej wartości zmiennej wyjściowej,
- jeśli wartość pewnej zmiennej można określić w prostym modelu obliczeniowym w wyniku różnych obliczeń, to za każdym razem uzyskuje się ten sam wynik, czyli wybór sposobu obliczania wartości nie jest istotny.

Jedna z metod rozwiązywania zadań w prostym modelu obliczeniowym polega na tym, że sprawdza się kolejno wszystkie operatory modelu dopóty, dopóki nie natrafi się na taki operator, który nadaje wartość przynajmniej jednej, dotychczas nieobliczonej zmiennej wyjściowej. Stosując go, otrzymuje się więc nową wartość, zbliżając się do rozwiązania zadania. Proces jest iterowany dopóty, dopóki nie zostaną otrzymane wszystkie wartości zmiennych wyjściowych zadania. Jeżeli okaże się, że nie można za pomocą dostępnych w modelu operatorów obliczyć którejś ze zmiennych wyjściowych, to oznacza to, że postawionego zadania nie można rozwiązać. Podczas konstruowania modeli jest przydatna także operacja łączenia modeli.