AITECH®
Artificial Intelligence Laboratory

ZINTEGROWANY PAKIET SZTUCZNEJ INTELIGENCJI AITECHSPHINX® 4.5

KRZYSZTOF MICHALIK

# PC-SHELL

## SZKIELETOWY SYSTEM EKSPERTOWY

### CZĘŚĆ 1

### PODRĘCZNIK UŻYTKOWNIKA

KATOWICE 2006

AITECH®

Artificial Intelligence Laboratory

ul. Kossutha 9, 40-844 KATOWICE
tel./fax: tel.: (0-32) 782-28-90
tel. kom. 0 502-99-27-28
e-mail: aitech@aitech.com.pl
WWW: http://www.aitech.com.pl

# SPIS TREŚCI

# 1

# WPROWADZENIE DO SYSTEMÓW EKSPERTOWYCH

## WSTĘP

SYSTEMY EKSPERTOWE (ang. *expert systems*) przestały być wyłącznie domeną naukowców i laboratoriów naukowych zajmujących się badaniami w dziedzinie sztucznej inteligencji (ang. *artificial intelligence*). Od kilku lat, głównie w Europie Zachodniej, USA i Japonii wyraźnie wzrasta liczba zastosowań tych systemów w praktyce. Możliwości zastosowań tej nowoczesnej technologii informatycznej są doprawdy ogromne, począwszy od medycyny, poprzez geologię, technikę aż do zastosowań, w dziedzinie wspomagania podejmowania decyzji gospodarczych i finansowych.

Systemy ekspertowe można określić jako programy, których podstawowym zadaniem jest symulowanie ludzkiej ekspertyzy w określonej, na ogół wąskiej, dziedzinie. Trudno podać ścisłe wyznaczniki tej technologii, jest ona bowiem w trakcie intensywnego rozwoju. Nie mniej jednak, można podać kilka cech odróżniających te systemy od konwencjonalnych programów:

- jawna interpretacja wiedzy i oddzielenie jej od procedur sterowania;
- zdolność wyjaśniania znalezionych przez system rozwiązań problemów;
- przetwarzanie wiedzy wykorzystujące głównie przetwarzanie symboli, w mniejszym zaś stopniu przetwarzanie numeryczne;
- do rozwiązywania problemów wykorzystane są głównie różne metody rozumowania (wnioskowania), w mniejszym zaś stopniu algorytmy.

Te cechy, jak się wydaje, tworzą współczesny paradygmat systemów ekspertowych.

Systemy ekspertowe mogą być tworzone z wykorzystaniem różnych narzędzi programowych, ogólnie jednak można je podzielić na dwie grupy:

1. mniej lub bardziej typowe języki programowania;
2. szkieletowe systemy ekspertowe (ang. *expert system shells* lub *skeletal systems*).

Opracowanie systemu ekspertowego, przy zastosowaniu pierwszego podejścia, jest zadaniem bardzo pracochłonnym i wymaga zatrudnienia programistów o wysokich kwalifikacjach. Jest to zatem podejście kosztowne, choć elastyczne. Alternatywą jest zastosowanie systemu szkieletowego, zawierającego gotowy podsystem przetwarzania wiedzy. W takim przypadku zadanie twórcy systemu polega głównie na pozyskaniu i sformalizowaniu wiedzy eksperckiej, co samo w sobie bywa zadaniem niełatwym. System PC-Shell jest narzędziem umożliwiającym to drugie podejście.

System PC-Shell zawiera w sobie część doświadczeń zyskanych przez autora podczas budowy szkieletowego systemu ekspertowego PC-Expert (lata 1985-87)[1] oraz prototypowego systemu do diagnostyki procesu produkcji układów scalonych Diagnosta MC 14007 (1988)[2]. Ten ostatni powstał we współpracy z dr Tomaszem Guttem z ITE w Warszawie.

System PC-Shell jest predysponowany głównie do rozwiązywania problemów o charakterze diagnostycznym i klasyfikacyjnym (taksonomicznym) oraz związanych z interpretacją danych. Może służyć również jako narzędzie prototypowania, wspomagając różnego rodzaju prace dyplomowe (magisterskie, doktorskie, itp.). Może także służyć jako system edukacyjny, ilustrujący wybrane zagadnienia w ramach tematów poświęconych problematyce sztucznej inteligencji i systemów ekspertowych.

[1] Michalik K. „PC-Expert: Szkieletowy system ekspercki w Prologu". II Ogólnopolskie Konwersatorium „Sztuczna inteligencja", ZG PTC, Warszawa 1987.

[2] Gutt T., Michalik K.: „Eksperymentalny system ekspercki do diagnostyki procesów technologicznych MOS przy użyciu struktury próbnej". Prace Instytutu Technologii Elektronowej CEMI, z. 5, PWN, Warszawa 1989.

## SKALA PROBLEMU

Postawienie problemu zbudowania maszyn zdolnych do naśladowania ludzkiej – naturalnej inteligencji, można traktować jako jedno z najpoważniejszych wyzwań dla nauki i techniki. Powaga tego zagadnienia wynika między innymi ze skali i złożoności problemu. Oto bowiem mózg ludzki ważący średnio 1,5 kg zawiera około 100 mld neuronów, tj. w przybliżeniu tyle, ile gwiazd liczy nasza galaktyka, skala zatem dosłownie i w przenośni jest astronomiczna. Jeśli założyć dalej, że każdy neuron może być połączony z 10 tysiącami innych, to całkowita liczba połączeń wynosi około $10^{15}$. Niektóre oszacowania dla pamięci dają w rezultacie pojemność około $4 \cdot 10^{15}$, przy założeniu pojemności 4 bity/synapsę. Podobnie szybkość przetwarzania jest szacowana na $4 \cdot 10^{16}$ ($10^{11} \cdot 10^{4} \cdot 10$ sygnałów/sekundę·4 bity/synapsę). Porównanie ze współczesnymi komputerami Connection Machine, wg Hugo de Garis, uwzględniające zarówno liczbę procesorów (65536), cenę (4 mln $) jak i ograniczenie wartości zakupu (20 mln $ = 5 maszyn tego typu), daje w rezultacie następujące wartości: zdolność pamiętania $\approx 2{,}2 \cdot 10^{10}$ bitów (200000 razy mniej od mózgu ludzkiego), szybkość przetwarzania $2 \cdot 10^{13}$ (2000 razy mniej).

Niezależnie od naszkicowanej skali problemu, sztuczna inteligencja, rozumiana jako dział informatyki dotyczący projektowania i budowy inteligentnych systemów komputerowych, odnotowała już wiele spektakularnych sukcesów, w tym również w sferze praktyki. W szczególności dotyczy to, już nie tylko potencjalnych, zastosowań w sferze wspomagania decyzji ekonomicznych i finansowych.

## TROCHĘ HISTORII

Historia badań w dziedzinie sztucznej inteligencji jest niemal tak długa jak historia informatyki. Jackson [2] wyróżnia trzy okresy w rozwoju sztucznej inteligencji: klasyczny, romantyczny oraz współczesny.

Okres *klasyczny* rozpoczyna się według Jacksona od publikacji Shannona [3] w 1950 roku i kończy pracą Feigenbauma i Feldmana [1] w 1963 roku. Podstawowym pojęciem, które wtedy wprowadzono jest przeszukiwanie przestrzeni stanów (ang. *state space search*). Wiele rodzajów problemów można sformułować za pomocą trzech podstawowych elementów:

- punkt początkowy (stan),
- test wykrywający stan końcowy,
- zbiór operacji, które mogą być stosowane do zmiany bieżącego stanu problemu.

Najprostszą formą przeszukiwania przestrzeni stanów jest algorytm określany jako generuj i testuj. Innym, ważnym zagadnieniem rozważanym w tym okresie było automatyczne dowodzenie twierdzeń. Jego istotą było reprezentowanie wiedzy odnoszącej się do problemu w postaci zbioru aksjomatów zwanych teorią. Rozwiązanie problemu traktowano jako proces mający na celu pokazanie, że rozwiązanie problemu jest twierdzeniem, tzn. wynika ze zbioru aksjomatów.

Okres *„romantyczny”* rozciąga się od połowy lat sześćdziesiątych do około połowy lat siedemdziesiątych. Badania i prace aplikacyjne koncentrowały się na problemie rozumienia języka naturalnego oraz ogólnych zasadach reprezentacji wiedzy. Jednym z bardziej znaczących systemów w tych latach był SHRDLU Winograda, zdolny do rozumienia podzbioru języka angielskiego, przez reprezentowanie i rozumowanie o pojęciach, w zakresie ograniczonej dziedziny (świat klocków – ang. *toy worlds*). Wiele ze współcześnie wykorzystywanych metod reprezentacji wiedzy powstało właśnie w tym okresie, m.in. wtedy pojawił się artykuł Quilliana dający początek sieciom semantycznym.

Okres *współczesny* rozciąga się od drugiej połowy lat siedemdziesiątych do chwili obecnej. Cechą charakterystyczną tego okresu jest uświadomienie sobie ograniczeń dotychczasowych rozwiązań. Nastąpiło przede wszystkim rozczarowanie ogólnymi metodami rozwiązywania problemów. Zaczęto uświadamiać sobie niejednorodność natury ludzkiej inteligencji, w przeciwieństwie do wcześniejszych poglądów reprezentowanych przez psychologów. Jednocześnie zdano sobie sprawę z faktu, że zdolność do rozwiązywania problemów jest silnie związana z wiedzą dziedzinową, a mniej z wyrafinowanymi mechanizmami wnioskowania. Tym samym rozpoczęto budowę systemów, w których wiedza z danej

dziedziny była reprezentowana explicite. Opracowano techniki umożliwiające kodowanie ludzkiej wiedzy w postaci modułów, które mogły być uaktywniane za pomocą wzorców. Zrozumienie roli wiedzy w rozwiązywaniu problemów doprowadziło m.in. do burzliwego rozwoju badań nad systemami ekspertowymi. Systemy ekspertowe dokonały istotnego przełomu nie tylko w teorii sztucznej inteligencji, ale również w zbliżeniu jej do praktyki. Są one bowiem do tej pory, choć nie jedynym, to z pewnością najbardziej utylitarnym produktem badań w dziedzinie sztucznej inteligencji. Potencjalne możliwości zastosowań tych systemów obejmują niemal wszystkie dziedziny, od medycyny, poprzez technikę, sferę obronności, a na wspomaganiu decyzji finansowych lub bankowych skończywszy. Nie tyle istotna jest więc dziedzina zastosowania, co raczej klasa problemów. Uważa się, że systemy te są szczególnie predysponowane do rozwiązywania problemów o charakterze klasyfikacyjnym, diagnostycznym, związanych z interpretacją (analizą) danych, a także niektórych problemów planowania i konstrukcji. W praktyce mogą występować w formie systemów doradczych, wspomagających decyzje, konsultacyjnych oraz monitorujących, o różnym stopniu autonomii działania. Z drugiej strony, badania oraz rozwój zastosowań praktycznych ujawniły pewne ograniczenia, np. związane z akwizycją wiedzy, o czym dalej. Jednocześnie należy podkreślić, że jest to dziedzina będąca w dalszym ciągu przedmiotem intensywnych badań naukowych.

## PROJEKT KOMPUTERÓW PIĄTEJ GENERACJI

Ogromny wpływ na rozwój badań i zastosowań sztucznej inteligencji wywarł tzw. Projekt Komputerów Piątej Generacji (KPG), i to nie tyle z powodu osiągniętych rezultatów utylitarnych, co przez samo zainicjowanie zakrojonych na szeroką skalę badań w tej dziedzinie. W niektórych ówczesnych publikacjach przyrównywano nawet wpływ japońskiego projektu komputerów piątej generacji do roli, jaką w rozwoju techniki odegrało wystrzelenie pierwszego sztucznego satelity.

W 1979 roku powołano z inicjatywy organizacji JIPDEC (ang. *Japan Information Processing Development Association*) komitet przeglądowy KPG, którego przewodniczącym został T. Moto-oka. W ciągu dwóch lat od daty utworzenia komitetu w jego pracach wzięło udział ponad 100 naukowców. W rezultacie podjętych w nim badań, w październiku 1981 roku, na międzynarodowej konferencji w Tokio, po raz pierwszy poinformowano świat o tym projekcie. Prace w ramach projektu KPG rozpoczęto 1 kwietnia 1982 roku wraz z powołaniem instytutu ICOT (ang. *Institute for New Generation Computer Technology*). Projektowi, zaplanowanemu na dziesięć lat, przyznano budżet w wysokości około 855 mln dolarów. Głównymi członkami ICOT były takie firmy jak: Fujitsu, Toshiba, Mitsubishi, NEC, Oki, Sharp i Hitachi. Jest to o tyle istotne, że pierwotnym zamierzeniem było przekazanie rezultatów badań do produkcji na skalę przemysłową. Jak wiadomo ten cel nie został osiągnięty.

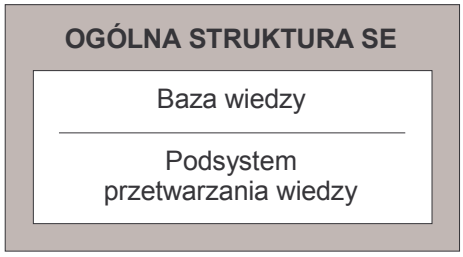
Głównym celem projektu KPG było stworzenie superkomputera do przetwarzania wiedzy, o architekturze równoległej. Podstawą do opracowania języka maszynowego tego komputera stało się programowanie logiczne (ang. *logic programming*).

Wpływ tego projektu na rozwój badań w dziedzinie sztucznej inteligencji ujawnił się m.in. w tym, że w wielu krajach uświadomiono sobie jaki przełom mógłby dokonać się w informatyce za sprawą powstania KPG. Dlatego odpowiedzią było powstanie kilku analogicznych programów badawczych. W USA uruchomiono dwa programy badawcze. Pierwszy z nich był koordynowany w ramach DARPA (ang. *Defense Advanced Research Project Agency*), drugi natomiast był realizowany przez konsorcjum MCC założone przez Williama Norrisa. Jego członkami były między innymi następujące firmy: Motorola, National Cash Register oraz National Semiconductor. Odpowiedzią Wielkiej Brytanii na japoński projekt było powołanie komitetu pod kierownictwem Johna Alveya, który przygotował propozycję pięcioletniego programu badawczego, z budżetem około 350 mln funtów.

[Szerzej zob. – Michalik K.: „Projekt komputerów piątej generacji”. Informatyka, nr 2, 1988, s. 28-30]

## CECHY SYSTEMÓW EKSPERTOWYCH

Systemy ekspertowe (SE) można określić jako programy służące do rozwiązywania problemów w sposób przypominający postępowanie eksperta lub specjalisty z określonej, na ogół wąskiej dziedziny. Do tego celu wykorzystywana jest wiedza zgromadzona w tak zwanej bazie wiedzy oraz różne procedury przetwarzania wiedzy.




RYS. 1-1. OGÓLNA STRUKTURA SE

Jak już wspomniano we wstępie, SE mają kilka cech charakterystycznych, odróżniających je od systemów konwencjonalnych:

1. jawna reprezentacja wiedzy,
2. zastosowanie do rozwiązywania problemów określonych procedur rozumowania (wnioskowania),
3. zdolność do objaśniania znalezionych przez system rozwiązań,
4. przetwarzanie wiedzy dotyczy głównie przetwarzania symboli, w mniejszym zaś stopniu obliczeń numerycznych.

W odróżnieniu od klasycznych programów komputerowych wiedza zawarta w bazie wiedzy opisuje dziedzinę problemową bez podania szczegółowego sposobu rozwiązania danego problemu (algorytmu). Dzięki temu jest dużo bardziej czytelna, nawet dla osób nie będących specjalistami w dziedzinie SE, ujmuje bowiem głównie merytoryczną stronę zagadnienia. Baza wiedzy przyjmuje na ogół postać pliku na dysku komputera.

Przechowywana w nim wiedza jest zapisana za pomocą określonego języka reprezentacji wiedzy, na który najczęściej składa się opis faktów (wiedza o charakterze faktograficznym), reguł stosowanych w procesie wnioskowania oraz w przypadku niektórych systemów – metareguł, opisujących strategię rozwiązywania danego problemu. Ponadto niektóre systemy wykorzystują tzw. ramy, które z jednej strony mogą być wykorzystywane w procesie wnioskowania, z drugiej zaś odwzorowują wiedzę o strukturze problemu. Pojęcie ram koresponduje do pewnego stopnia z podejściem obiektowym w programowaniu.

Wiedza znajdująca się w bazie wiedzy może pochodzić z różnych źródeł, najczęściej jednak pochodzi od ekspertów lub innych specjalistów z danej dziedziny. Pozyskaniem wiedzy eksperckiej oraz jej formalizacją, tj. zapisaniem za pomocą określonego języka reprezentacji wiedzy, zajmuje się informatyk – tzw. inżynier wiedzy. Inżynier wiedzy to nowa profesja powstająca na gruncie sztucznej inteligencji. Proces pozyskiwania wiedzy jest na ogół bardzo pracochłonny i realizowany w toku współpracy inżyniera wiedzy i eksperta. W przypadku tradycyjnego podejścia do pozyskiwania wiedzy, stosuje się różne techniki takie jak dialog z ekspertem lub obserwacja eksperta podczas rozwiązywania przez niego konkretnych problemów. Panuje pogląd, że proces pozyskiwania wiedzy stanowi „wąskie gardło" budowy systemów ekspertowych. Dlatego badania naukowe zmierzają do automatyzacji tego procesu za pomocą tzw. algorytmów indukcyjnych (np. ID3 lub AQ11), które są w stanie wygenerować reguły dla bazy wiedzy na podstawie zbiorów uczących. Zbiory te, ujęte w postaci tzw. plików uczących, zawierają najczęściej dane z przeszłości opisujące sytuację problemową oraz wynikającą z niej konkluzję człowieka. Na podstawie tych danych system uczy się rozwiązywania podobnych problemów samodzielnie.

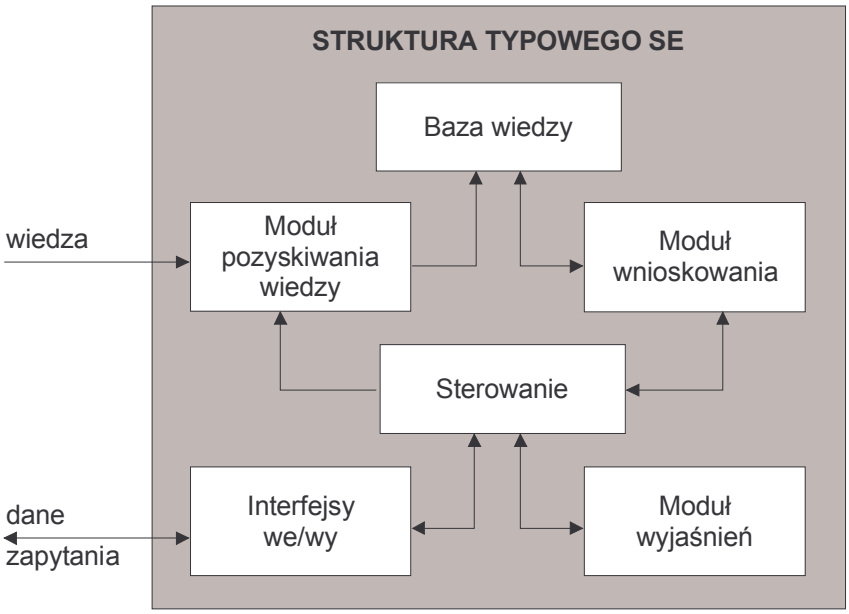
W tzw. tablicowych systemach ekspertowych wiedza może być rozproszona w kilku plikach, tzw. źródłach wiedzy. Każde ze źródeł wiedzy może przechowywać wtedy wiedzę służącą do rozwiązania innego podproblemu.

W odróżnieniu od konwencjonalnych systemów, systemy ekspertowe nie zawierają jawnego opisu sposobu rozwiązania danego problemu (algorytmu). To system ekspertowy, a ściślej jego część zwana modułem wnioskującym (ang. *inference engine*) rozwiązuje problem, wykorzystując wiedzę deklaratywną w bazie wiedzy. Moduł wnioskowania realizowany jest najczęściej w oparciu o zasady logiki formalnej. W praktyce wykorzystywane są na ogół dwie metody wnioskowania: do tyłu (ang. *backward chaining*) oraz do przodu (ang. *forward chaining*). Czasami stosowane jest tzw. wnioskowanie mieszane, będące połączeniem w różnych proporcjach obu wcześniej wymienionych metod. Wnioskowanie do tyłu rozpoczyna się od postawienia hipotezy (celu), która ma być potwierdzona w trakcie wnioskowania. W tym przypadku proces wnioskowania przebiega od hipotezy, poprzez reguły do faktów, potwierdzających warunki reguł. Zadaniem systemu jest jej potwierdzenie, zaprzeczenie lub wskazanie niemożliwości rozwiązania danego problemu. W przypadku wnioskowania do przodu, proces wnioskowania przebiega w kierunku odwrotnym i rozpoczyna się od zbioru faktów podanych na wejściu. W rezultacie zostają wygenerowane wszystkie logiczne konsekwencje tych faktów, stanowiące ostateczne konkluzje systemu.

Unikatową cechą systemów ekspertowych jest ich zdolność do automatycznego generowania różnego rodzaju objaśnień w trakcie konsultacji z użytkownikiem, w szczególności zaś wyjaśnień dotyczących sposobu rozwiązania problemu. Wyjaśnienia tworzone są przez tzw. moduł wyjaśniający (ang. *explanation facility*). Obecnie spotyka się najczęściej trzy rodzaje wyjaśnień:

- „**jak?**” (ang. *how*) – odpowiadające na pytanie „w jaki sposób system wyprowadził dany zbiór konkluzji”; wyjaśnienia mają w tym przypadku charakter retrospektywny i pokazują logiczny wywód systemu;
- „**dlaczego?**” (ang. *why*) – odpowiadające na pytanie „dlaczego system zadał użytkownikowi dane pytanie”; wyjaśnienia tego typu uzasadniają celowość pytania, poprzez pokazanie bieżącego kontekstu wnioskowania oraz tego jak odpowiedź na dane pytanie przyczyni się do rozwiązania problemu;
- „**co to jest?**” (ang. *what is*) – objaśniające pojęcia zawarte w bazie wiedzy.

Wyjaśnienia są tak ważnym i specyficznym elementem technologii systemów ekspertowych, że można zaryzykować stwierdzenie, że system który jest ich pozbawiony, nie jest systemem ekspertowym. Typową architekturę SE przedstawiono na rys. 1-2




RYS. 1-2. ARCHITEKTURA SYSTEMU EKSPERTOWEGO

**Dlaczego warto stosować systemy ekspertowe?** – podsumowanie walorów systemów ekspertowych:

1. Jawna reprezentacja wiedzy w postaci zrozumiałej dla użytkownika końcowego.
2. Możliwość przyrostowej budowy bazy wiedzy.
3. Względna łatwość modyfikowania bazy wiedzy.
4. SE są narzędziem kodyfikacji wiedzy eksperckiej.
5. Systemy ekspertowe mają zdolność rozwiązywania problemów specjalistycznych, o charakterze jakościowym, w których dużą rolę odgrywa doświadczenie.
6. Wiedza ekspercka jest dobrem rzadkim i kosztownym.
7. Systemy ekspertowe mają zdolność wyjaśniania własnych konkluzji.
8. Systemy ekspertowe zwiększają dostępność ekspertyzy.
9. Systemy tablicowe mogą mieć kompetencje większe od pojedynczego eksperta.
10. Możliwość prowadzenia jednolitej polityki przez centrale firm mających wiele oddziałów.
11. Poziom ekspertyzy jest stabilny – jej jakość nie zależy od warunków zewnętrznych i czasu pracy systemu.

# REPREZENTACJA WIEDZY I WNIOSKOWANIE

## WPROWADZENIE

Problematyka reprezentacji wiedzy jest jednym z najważniejszych nurtów badań w dziedzinie sztucznej inteligencji. Systemy ekspertowe wykorzystują bowiem do rozwiązywania problemów jawnie wyrażoną wiedzę z określonej dziedziny. Wiedza ta musi być wcześniej opisana (sformalizowana) za pomocą tzw. języka reprezentacji wiedzy i wprowadzona do systemu. Kluczowym pojęciem jest tu wiedza, czyli w uproszczeniu zbiór wiadomości z określonej dziedziny. W kontekście systemów ekspertowych wiedzę można określić jako informacje o świecie, umożliwiające ekspertom rozwiązywanie problemów i podejmowanie decyzji. Przez reprezentację wiedzy będziemy tu rozumieć sposób odwzorowania tej wiedzy w określony formalizm, który jest „zrozumiały" dla systemu ekspertowego. Oznacza to zdolność SE nie tylko do statycznego przechowywania fragmentu wiedzy o świecie, ale również jej efektywne przetwarzanie w celu znalezienia rozwiązania postawionego przed nim problemu. Efektywność jest bardzo ważnym kryterium z punktu widzenia praktycznej realizacji systemów ekspertowych. O ile w początkowym okresie rozwoju dużą rangę przypisywano znalezieniu uniwersalnych metod wnioskowania, to po pierwszych spektakularnych sukcesach systemów ekspertowych zasadniczą wagę dla sukcesu tej technologii zaczęto przywiązywać do wiedzy jako takiej. Dziś oba punkty widzenia wydają się skrajne.

Wynika to ze skali problemu, który ma rozwiązywać system ekspertowy. W przypadku niewielkich systemów (baz wiedzy) zarówno sposób wnioskowania jak i reprezentowania wiedzy może nie mieć zauważalnego wpływu na sposób działania systemu. Jednakże w ostatnim czasie systemy ekspertowe stosowane są do rozwiązywania coraz bardziej złożonych problemów. W przypadku tego rodzaju zastosowań staje się mniej ważny sam fakt posiadania wiedzy eksperckiej. Zaczynają coraz większą rolę odgrywać efektywne mechanizmy wnioskowania oraz reprezentacji wiedzy. I nie chodzi tu wyłącznie o czas reakcji systemu, lecz problemy związane z czytelnością baz wiedzy, a w konsekwencji również trudnościami z ich utrzymywaniem i rozwojem. Dlatego wydaje się, że we współczesnych systemach ekspertowych ich architektura, w tym zastosowane mechanizmy wnioskowania i reprezentacji wiedzy, decydują na równi z jakością wiedzy eksperta o powodzeniu aplikacji.

W dalszej części tego rozdziału omówiono podstawowe, dziś już uważane za niemal klasyczne, sposoby reprezentacji wiedzy, a mianowicie: reguły, sieci semantyczne oraz ramy. Należy podkreślić, że opisane tam ogólne metody reprezentowania wiedzy przybierają w konkretnych systemach szczegółową postać w formie tzw. języka reprezentacji wiedzy. Ponieważ dziedzina systemów ekspertowych w dalszym ciągu szybko się rozwija, zatem trudno mówić jeszcze o jakiejś formie standaryzacji tych języków, choć zauważalne są pewne ogólne podobieństwa i tendencje.

Można wyróżnić dwa aspekty języka reprezentacji wiedzy: syntaktyczny i inferencyjny. Ten pierwszy dotyczy sposobu w jaki wiedza jest zapisywana (przechowywana) w pamięci komputera, natomiast drugi aspekt dotyczy sposobu uzyskania wiedzy, która tkwi w tym zapisie w sposób niejawny. Ogólnie języki reprezentowania wiedzy można podzielić na deklaratywne i proceduralne (imperatywne). Języki reprezentacji wiedzy systemów ekspertowych należą na ogół do tej pierwszej kategorii. Języki imperatywne reprezentują wiedzę, jawnie określając sposób rozwiązania problemu. W odróżnieniu od takiego podejścia, języki deklaratywne opisują pewien fragment dziedziny i co należy zrobić. Sposób rozwiązania jest w dużej mierze powierzany systemowi, w którym taki język jest zaimplementowany. Dobrym przykładem podejścia deklaratywnego jest język Prolog.

## REGUŁY

Większość zrealizowanych dotychczas systemów ekspertowych wykorzystuje do reprezentowania wiedzy reguły. Systemy, wykorzystujące jako podstawę reprezentacji wiedzy reguły nazywane są systemami regułowymi. Powszechność tego sposobu reprezentowania wiedzy wynika z wielu zalet tego formalizmu. Głównymi zaletami reguł są, jak się wydaje, prostota i ogólność. Ta pierwsza właściwość czyni je zrozumiałymi nawet dla osób nie będących specjalistami w dziedzinie systemów ekspertowych. Prostota tego formalizmu może, po pierwszym kontakcie z nim, budzić wątpliwości co do jego walorów utylitarnych i tzw. siły wyrazu. Jest to jednak wrażenie bardzo mylne, o czym świadczy względnie duża liczba praktycznie działających systemów regułowych. Druga właściwość powoduje, że dziedzina nie jest ograniczeniem w stosowaniu tej metody reprezentacji, o tym z kolei świadczą przykłady udanych aplikacji w tak różnych dyscyplinach jak: medycyna, technika, ekonomia czy geologia.

Najogólniej regułę można zapisać w następujący sposób:

```
JEŚLI  przesłanka
TO     konkluzja
```

gdzie przesłanka jest zbiorem prostych zdań logicznych, połączonych funktorami i (*and*) oraz lub (*or*). Przesłanka jest zbiorem warunków, których spełnienie pozwala uznać prawdziwość konkluzji. Warunek najczęściej jest poznaną już trójką OAW. Można zatem przedstawić, teraz bardziej szczegółowo, podany wcześniej ogólny schemat:

```
JEŚLI  warunek 1  i / lub
       warunek 2  i / lub
       ...
       warunek_n
TO     konkluzja
```

Można w dalszym ciągu mieć wątpliwości jak ten prosty schemat ma się do wiedzy o złożonych nieraz aspektach świata rzeczywistego. Niech jako pierwsze przybliżenie posłuży odniesienie tego schematu do rzeczywistości niestety dobrze nam znanej, w formie przykładu o niekoniecznie zachowanej poprawności merytorycznej.

Przykład:

```
pacjent_jest_chory_na_grypę   jeśli
    pacjent_ma_wysoką_temperaturę i
    pacjent_ma_bóle_mięśni  i
    pacjent_ma_kaszel
```

W tym miejscu pojawia się okazja, aby pokazać jeszcze jedną istotną cechę systemów regułowych, a mianowicie możliwość tzw. przyrostowej rozbudowy bazy wiedzy. Właściwość ta wynika m.in. bezpośrednio z faktu, że reguły traktowane są jako silnie wyodrębnione fragmenty wiedzy, o ściśle ograniczonym kontakcie z innymi regułami. Dzięki temu, inżynier wiedzy może po pewnym czasie bardziej precyzyjnie określić, uzupełnić pojęcie *wysoka_temperatura* przez dodanie nowej reguły, stanowiącej definicję tego pojęcia:

```
pacjent_ma_wysoką_temperaturę  jeśli
    temperatura_ciała > 37,5
```

Walorem regułowych systemów ekspertowych jest to, że ten zbliżony w treści do języka naturalnego zapis może być bezpośrednio wykorzystany w procesie wnioskowania. Wnioskowanie jest podstawowym sposobem rozwiązywania problemów w systemach ekspertowych. Tradycyjne algorytmy mogą być jedynie uzupełnieniem opisu problemu. W regułowych systemach ekspertowych spotyka się trzy podstawowe metody wnioskowania (ang. *inference*):

- do przodu (ang. *forward chaining*),
- wstecz (ang. *backward chaining*),
- mieszane (ang. *mixed reasoning*).

Wnioskowanie do przodu przebiega od faktów, poprzez reguły aż do końcowych konkluzji, co można ująć następującym schematem:

**fakty → reguły → konkluzje**

Podstawę logiczną może stanowić reguła modus ponens, która mówi że jeśli spełnione (prawdziwe) są przesłanki, to prawdziwa jest również konkluzja. Tę regułę wnioskowania, zapisuje się w postaci schematu:

$$\frac{(A \rightarrow B), A}{B}$$

co można interpretować następująco: „Jeśli z przesłanki *A* wynika *B* i *A* jest prawdziwe, to uznajemy, że *B* jest również prawdziwe". W notacji używanej w systemie ekspertowym $A \rightarrow B$ można zapisać jako regułę: „*B jeśli A*". W praktyce systemy ekspertowe pozwalają wiązać ze sobą kolejne reguły, poprzez generowanie kolejnych faktów potwierdzających następnie przesłanki następnych reguł. Proces ten najczęściej jest kończony, gdy zbiór wygenerowanych w procesie wnioskowania faktów jest pusty lub wygenerowany fakt (konkluzja) jest rozwiązaniem wcześniej postawionego problemu. W ten sposób proces wnioskowania może tworzyć „łańcuch" powiązanych ze sobą reguł (stąd *chaining* – co można określić jako tworzenie łańcucha). Słabością tej metody wnioskowania jest to, że w niektórych przypadkach może prowadzić do tzw. kombinatorycznej (!) eksplozji wygenerowanych faktów, z których tylko część będzie ważna z punktu widzenia postawionego przed systemem problemu.

Wnioskowanie wstecz przebiega od postawionego problemu, nazywanego często celem (ang. *goal*) lub hipotezą, poprzez reguły aż do faktów, co można ująć następującym schematem:

**cel → reguły → fakty**

Tę metodę wnioskowania, zwłaszcza w systemach silniej opartych na logice matematycznej (np. systemy prologowe), można wiązać z regułą *modus tollens*, która stwierdza, że:

$$\frac{(A \rightarrow B), \sim B}{\sim A}$$

Wnioskowanie w tym przypadku polega na wybraniu reguł, których konkluzje dotyczą postawionego celu (głównego). Następnie system stara się potwierdzić prawdziwość tych konkluzji. W trakcie tego procesu potwierdzane są warunki wybranych reguł, które w momencie, gdy są uaktywnione (potwierdzane) stają się bieżącym celem. Proces ten kończy się, gdy choć jedna z reguł dotyczących celu głównego ma wszystkie warunki spełnione (mówi się czasami, że wyprowadziła konkluzję). Możliwa jest oczywiście sytuacja, gdy kilka z wybranych reguł wyprowadziło ostateczne konkluzje (a więc takie, które odpowiadają na postawiony cel główny), wtedy system ekspertowy znajduje nie jedno, a kilka rozwiązań problemu (niektóre systemy mają taką właściwość). Druga sytuacja, gdy wnioskowanie jest kończone ma miejsce wtedy, gdy żadna z reguł uaktywnionych (wybranych) podczas procesu wnioskowania nie wyprowadziła konkluzji, czyli ich przesłanki nie zostały spełnione. Jak wynika z tego omówienia, cały proces rozwiązywania problemu w tej metodzie

wnioskowania jest podporządkowany celowi głównemu, co w istotny sposób ogranicza przestrzeń poszukiwania rozwiązań. Dzięki temu w wielu zastosowaniach ta metoda okazuje się bardzo efektywna.

Wnioskowanie mieszane polega na połączeniu w różnych proporcjach obu wcześniej omówionych metod. Oznacza to, że część konkluzji, podczas rozwiązywania problemu, jest wyprowadzona w procesie wnioskowania wstecz, a pozostałe są rezultatem wnioskowania do przodu.

## PROGRAMOWANIE W JĘZYKU LOGIKI

Programowanie w języku logiki, dalej określane skrótowo programowaniem logicznym (ang. *logic programming*) narodziło się na początku lat siedemdziesiątych jako rezultat wcześniejszych prac w zakresie automatycznego dowodzenia twierdzeń i sztucznej inteligencji. Z chwilą opracowania w 1972 roku przez Colmarauera i Roussela języka Prolog, programowanie logiczne nabrało waloru praktycznej użyteczności. Do wzrostu popularności tego języka przyczyniło się zwiększenie efektywności jego implementacji, a także jego wybór jako podstawy opracowania języka maszynowego komputerów piątej generacji.

Pojęcie programowania logicznego nie jest dobrze zdefiniowane. Najczęściej jednak ogranicza się jego zakres do sytuacji, gdy informacja jest reprezentowana za pomocą tzw. klauzul Horna, a dedukcja jest przeprowadzana z wykorzystaniem wnioskowania wstecz. Można jednak rozumieć programowanie logiczne szerzej, włączając w to pojęcie negację przez porażkę lub operator obcięcia (ang. *cut operator*).

Programowanie logiczne opiera się na dwóch zasadniczych ideach: traktowania logiki jako języka programowania oraz reprezentowania algorytmów w postaci dwóch rozdzielonych składników: logiki oraz sterowania. Wykonywanie programów logicznych stało się możliwe po nadaniu im interpretacji proceduralnej.

Programowanie logiczne wykorzystuje głównie klauzulową postać logiki. Klauzula jest wyrażeniem o postaci:

$$\forall x_1,..,\forall x_m\left(L_1 \vee .. \vee L_n\right)$$

gdzie $L_i$ są literałami, a $x_1,..,x_m$ są zmiennymi występującymi w $L_1,..,L_n$. Literał jest wyrażeniem atomowym lub negacją tego wyrażenia. Wyrażenia atomowe mają następującą postać: $P(t_1,..,t_m)$, gdzie $P$ jest symbolem predykatu, a $t_1,..,t_m$ są termami oraz $m \geq 1$. Termy są stałymi, zmiennymi lub wyrażeniami o postaci: $f(t_1,..,t_m)$, gdzie $f$ jest symbolem funkcji, a $t_1,..,t_m$ są termami oraz $m \geq 1$.

Klauzula o postaci:

$$\forall x_1,..,\forall x_m\left(A_1 \vee .. \vee A_m \vee \sim B_1 \vee .. \vee \sim B_n\right)$$

gdzie $A_1,..,A_m$, $B_1,..,B_n$ są wyrażeniami atomowymi, będzie zapisywana w następujący sposób:

$$A_1,..,A_m \leftarrow B_1,..,B_n$$

Jeśli $m=0$, to taką klauzulę nazywa się celem.
Jeśli $n=0$, to klauzula jest stwierdzeniem bezwarunkowym (asercją, faktem).
Jeśli $m=0$ i $n=0$, to taką klauzulę nazywa się klauzulą pustą.

W programowaniu logicznym wykorzystuje się głównie podzbiór klauzul o postaci:

$$A \leftarrow B_1,..,B_n$$

Wyrażenia o tej postaci są nazywane klauzulami Horna i w sensie formalnym są w istocie regułami zdolnymi do reprezentowania wiedzy. Klauzule Horna mogą być również interpretowane jako deklaracje procedur, gdzie $A$ jest nagłówkiem procedury, a $B_1,..,B_n$ stanowią jej treść (ciało). W przypadku programowania logicznego nie używa się pojęcia bazy wiedzy, lecz szerszego – programu logicznego. Program logiczny składa się z celu (o takim samym znaczeniu jak wspomniany w części poświęconej omówieniu wnioskowania wstecz) oraz skończonej liczby procedur.

Podobnie jak w zwykłych regułach w procesie rozwiązywania problemu przez program logiczny, występuje mechanizm uzgadniania. Formalną definicję uzgadniania rozpoczniemy od zdefiniowania podstawienia $\Theta$. $\Theta$ jest zbiorem o postaci: $\{x_1/t_1,..,x_n/t_n\}$, gdzie $x_i$ są zmiennymi, a $t_i$ termami. Każdy element $x_i/t_i$ nazywany jest

wiązaniem dla $x_i$. Jeśli $W$ jest wyrażeniem (wyrażenie atomowe, term, klauzula), to rezultatem podstawienia $\Theta=\{x_1/t_1,..,x_n/t_n\}$ do W jest nowe wyrażenie $\Theta W$. $\Theta W$ jest nazywane również wcieleniem (ang. *instance*) lub konkretyzacją wyrażenia $W$. Mówi się, że $\delta$ uzgadnia dwa wyrażenia $W_1$ i $W_1$ wtedy, gdy powoduje ono, że te wyrażenia stają się identyczne.

Przykład klauzuli programu logicznego:

```
lubi( jaś, X ) ← interesuje_się( X, sztuczna_inteligencja )
interesuje_się( małgosia, sztuczna_inteligencja ) ←
```

Pierwsze wyrażenie można odczytać następująco: „Jaś lubi każdego, kto interesuje się sztuczną inteligencją”, a drugie, że „Małgosia interesuje się sztuczną inteligencją”. Chcąc utworzyć kompletny program logiczny powinniśmy jeszcze dodać cel, np.

```
← lubi( jaś, X )
```

co można interpretować jako pytanie „Kogo lubi Jaś?”. Rezultatem pracy programu logicznego będzie w tym przypadku wygenerowanie rozwiązania o postaci:

```
X = małgosia
```

co oznacza, że Jaś lubi Małgosię. Natomiast cel o postaci:

```
← lubi( X, Y )
```

który można interpretować jako pytanie „Kto kogo lubi?”, wygeneruje następujące rozwiązanie:

```
X = jaś
Y = małgosia
```

Gdyby do podanego wcześniej programu dodać jeszcze jeden fakt:

```
interesuje_się( baba_jaga, sztuczna_inteligencja ) ←
```

to rezultatem postawienia ostatniego celu będą dwa rozwiązania:

```
1)   X = jaś
     Y = małgosia
2)   X = jaś
     Y = baba_jaga
```

Dzieje się tak dlatego, że programy logiczne, dzięki tzw. mechanizmowi nawrotów (ang. *backtracking*) mają zdolność znajdowania wszystkich, dających się logicznie wyprowadzić, rozwiązań.

Podany dalej przykład ilustruje możliwości reprezentowania przez reguły programu logicznego bardziej złożonych relacji, definiujących w tym przypadku rodzaj pokrewieństwa rodzinnego. Jednocześnie zastąpimy symbol wynikania (implikacji) „←” słowem *if* (jeśli), natomiast w faktach ten symbol będzie opuszczony. W celu symbol „←” będzie zastąpiony przez znak zapytania „?”. Ta nowa notacja jest zgodna z zasadami przyjętymi w języku Prolog, w niektórych jego dialektach (np. edynburskim) zamiast symbolu „←” używa się dwuznaku **:–** (dwukropek, myślnik).

Jednocześnie w notacji prologowej najczęściej zmienne rozpoczynają się od dużych liter, natomiast stałe tekstowe (symbole) rozpoczynają się od małej litery.

reguły :

```
kobieta( X )  if matka( X, Y )
mężczyzna( X ) if ojciec( X, Y )
brat( X, Y ) if
    mężczyzna( X ), mężczyzna( Y ),
    rodzic( Z, X ), rodzic( Z, Y ),
    X <> Y
dziadek( X, Y ) if ojciec( X, Z ), rodzic( Z, Y )
rodzic( X, Y ) if ojciec( X, Y )
rodzic( X, Y ) if matka( X, Y )
```

fakty:

```
ojciec( druzus, germanik )
ojciec( druzus, klaudiusz )
ojciec( germanik, neron )
ojciec( klaudiusz, brytanik )
matka( liwia, druzus )
```

Podany dalej zbiór pytań wygeneruje następujące odpowiedzi:

1. „Kto był matką Druzusa?"
   cel: matka( *X*, druzus )?
   odp.: *X* = liwia.

2. „Kto był dziadkiem Klaudiusza?"
   cel: dziadek( *X*, klaudiusz )?
   odp.: *X* = druzus.

## PODSUMOWANIE

Podsumowując cechy reprezentacji wiedzy przy wykorzystaniu reguł można stwierdzić, że:

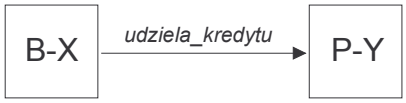
- Reguły umożliwiają reprezentowanie wiedzy w sposób jawny, zwłaszcza w porównaniu z tradycyjnymi technikami opartymi na algorytmach. Są szczególnie predysponowane do reprezentowania wiedzy o charakterze heurystycznym (zależnej od doświadczenia i intuicji eksperta), służącej do symulacji procesów decyzyjnych.
- Eksperci z różnych dziedzin nie mają na ogół problemów z wyrażeniem części swojej wiedzy za pomocą tego formalizmu, co ułatwia współdziałanie inżyniera wiedzy i eksperta.
- Reguły są na ogół czytelne dla użytkowników końcowych aplikacji, co ma duże znaczenie dla powodzenia praktycznych zastosowań technologii SE. Jednocześnie ta cecha ułatwia wprowadzanie mechanizmów wyjaśnień, które w niektórych obszarach zastosowań mają ogromne znaczenie praktyczne.
- Często akcentuje się deklaratywny charakter tego sposobu reprezentacji, prowadzący m.in. do łatwości modyfikacji bazy wiedzy. Jak się wydaje, można uznać ten pogląd za prawdziwy, choć z pewnymi zastrzeżeniami. Po pierwsze, o ile reguły nie są w jakiś sposób zróżnicowane (jawnie podzielone na klasy, konteksty, źródła wiedzy), to stwierdzenie to jest prawdziwe do pewnej wielkości bazy wiedzy. Po drugie, deklaratywność nie jest wyłącznie cechą związaną z danym formalizmem. W dużym stopniu deklaratywność, więc i czytelność bazy wiedzy, zależy od sposobu jej kodowania. Wyraźnie widać to w programowaniu logicznym, gdzie program składa się wyłącznie z reguł i faktów, a deklaratywność jest silnie akcentowaną cechą tego formalizmu. Ostateczny efekt jest jednak zależny od tego, czy inżynier wiedzy lub programista stara się wykorzystać to narzędzie w sposób deklaratywny, czy też bardziej proceduralny, traktując je wyłącznie jako język programowania, a nie środek reprezentacji wiedzy. Dla przykładu w Prologu bardzo przejrzyście definiująca problem procedura rekurencyjna może być zrealizowana za pomocą mniej czytelnego mechanizmu powtórzeń. Dodatkowo na deklaratywność wpływają aspekty mnemotechniczne kodowania. Typowe rozwiązania mnemotechniczne nie powinny być raczej stosowane w systemach regułowych, symbole (identyfikatory) powinny być sformułowane w sposób bardziej pełny (samoobjaśniający).
- Reguły ułatwiają przyrostową budowę bazy wiedzy i powiększanie jej (doskonalenie) w miarę zdobywanych doświadczeń oraz wyników weryfikacji praktycznej. Właściwość ta jest pochodną kilku cech, a zwłaszcza faktu, że reguły są wyraźnie wyodrębnionymi fragmentami wiedzy, względnie niezależnymi od siebie. Dla przykładu, jeśli w regule pojawiają się zmienne, to mają one charakter lokalny dla danej reguły.
- Nie cała wiedza ekspercka lub ogólniej wiedza służąca do rozwiązania problemu wyraża się za pomocą reguł heurystycznych.

- Formalizm reguł, w swej czystej postaci nie pokazuje wyraźnie ogólnej struktury problemu, choć wiele zależy tu znowu od sposobu kodowania.
- Większość interpreterów reguł nie wykorzystuje mechanizmu nawrotów, utrudniając tym samym automatyczny powrót do wcześniejszych stanów obliczeń.
- Wielkim walorem systemów regułowych jest to, że pokazały już swoją skuteczność w rozwiązywaniu złożonych problemów praktycznych, w bardzo wielu różnych dziedzinach wiedzy.

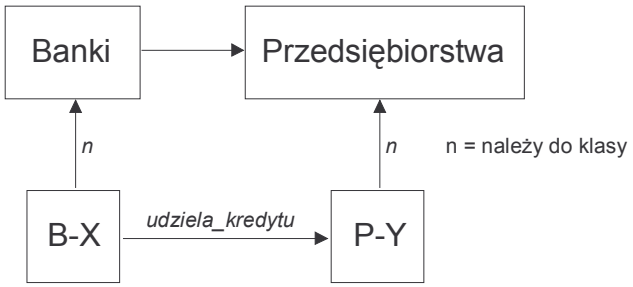
## SIECI SEMANTYCZNE

Sieci semantyczne są strukturami służącymi do reprezentacji wiedzy za pomocą węzłów i gałęzi. Węzły reprezentują na ogół pojęcia, a gałęzie relacje pomiędzy nimi. Węzły mogą być również podzbiorami elementów sieci. Idea ta jest przypisywana Quillianowi. Początkowo były one zaprojektowane jako język pośredni do celów tłumaczenia maszynowego. Na gruncie psychologii prowadzono badania nad sieciami semantycznymi jako ewentualnym modelem pamięci długoterminowej człowieka. Jednakże późniejszy rozwój sieci doprowadził do tego, że stały się bardziej elastyczne i posiadały większą siłę wyrazu, mogąc w ten sposób konkurować z systemami ram.

Stwierdzenie (fakt) „Bank *X* udziela kredytu przedsiębiorstwu *Y*" może być reprezentowane w następujący sposób:




Pojęcia są zazwyczaj elementami grup (klas) pojęć, co można przedstawić za pomocą relacji




Charakteryzując sieci semantyczne jako narzędzie do reprezentowania wiedzy należy zauważyć, że są one użyteczne do organizowania wiedzy w postaci hierarchii lub sieci obiektów. W tym charakterze pojawiają się często jako element innych sposobów reprezentacji (np. ram). Niektórzy specjaliści (np. Nilsson, Puppe) wręcz nie traktują sieci semantycznych jako niezależnego sposobu reprezentacji, lecz jedynie jako graficzną formę przedstawienia reprezentacji obiektowej. Wychodzą bowiem z założenia, że węzeł wraz z powiązaniami z innymi węzłami jest wewnętrznie reprezentowany jako obiekt wraz z atrybutami i wartościami. W przypadku klasycznych sieci brak jest wskazówek, w jaki sposób przypisywać znaczenia do węzłów (co one mają reprezentować), czy chodzi np. o pojęcie, klasę wszystkich pojęć lub typowego przedstawiciela danej klasy. Poszukiwanie informacji nie jest samo w sobie oparte na wiedzy, brak zatem w czystych systemach tego typu wiedzy mówiącej o tym, jak dotrzeć do określonej informacji. Aby usunąć niektóre niedogodności reprezentowania wiedzy za pomocą sieci (np. tzw. *partitioned networks*) zaproponowano wiele rozwiązań rozszerzających możliwości sieci semantycznych. Jednakże w ten sposób zaczęły one coraz bardziej przypominać inne metody reprezentacji wiedzy, takie jak logika, ramy, przez co tracono istotną cechę sieci semantycznych, a mianowicie ich prostotę.

## RAMY

Teoria ram pojawiła się na gruncie sztucznej inteligencji jako rezultat raportu technicznego Marvina Minsky'ego. Początkowo teoria ta miała związek z rozumowaniem naturalnym (ang. *commonsense reasoning*).

Wynikała z jednej strony, z chęci wyjaśnienia efektywności rozumowania naturalnego w odniesieniu do problemów świata rzeczywistego, z drugiej zaś – zbudowania bazy danych zawierającej dużą ilość wiedzy deklaratywnej, niezbędnej w tego typu rozumowaniu. Wiedza miała być kodowana w sposób dostatecznie strukturalny, by osiągnąć pewien stopień zgodności w bazie danych, i elastyczny, by system mógł odzyskiwać własną informację w sytuacjach nieprzewidzianych. Minsky zaproponował strukturę, w której wiedza była zgrupowana w jednostki nazywane obecnie ramami (ang. *frames*). Na marginesie, źródłowe określenie „*frame"* jest oparte na skojarzeniu z pojedynczą klatką (kadrem) filmu, stąd wydaje się, że tłumaczenie jako „rama" nie oddaje istoty intencji autora pojęcia. Sam Minsky określił ramę w następujący sposób:

„Rama jest strukturą danych służącą do reprezentowania stereotypowych sytuacji, takich jak bycie w jakimś pokoju lub wyjście na przyjęcie urodzinowe dziecka. Z każdą ramą związanych jest kilka rodzajów informacji. Niektóre z nich mówią, jak użyć daną ramę. Inne informują o tym, jakiego zdarzenia należy oczekiwać. Jeszcze inne informują co robić, jeśli te oczekiwania nie są spełnione."

Wprowadzenie tego pojęcia spowodowało lawinę badań w dziedzinie sztucznej inteligencji oraz doprowadziło do powstania pierwszych języków praktycznie implementujących idee Minsky'ego, takich jak KRL, FRL czy KL-ONE. Należy tu jednak zauważyć, że pojęcie ramy koresponduje do pewnego stopnia z niektórymi wcześniej proponowanymi pojęciami, np. schematu (ang. *schema*) wprowadzonym przez Bartletta.

Ramy, tak jak je początkowo nakreślono, były modułem wiedzy, który stawał się aktywny w odpowiedniej sytuacji i dostarczał zarówno jej interpretacji, jak również umożliwiał nowe przewidywania. Minsky dał bardzo ogólne wskazówki, co do natury struktury danych, która miała to zapewniać. Jak się wydaje, najlepszym sposobem przekazania intencji związanych z teorią ram jest posłużenie się jednym z przykładów Minsky'ego, podanym tu za Kuiperem. Opisuje on związki pomiędzy oczekiwaniem, percepcją a doświadczeniem zmysłowym. Przyjęto w nim założenie, że znajdujemy się w domu i naszym zadaniem jest otwarcie drzwi do nieznanego pokoju. Przed wykonaniem tej czynności mamy na ogół pewne wyobrażenie tego, co będzie można dostrzec po otwarciu drzwi. Jeśli po otwarciu drzwi zobaczylibyśmy brzeg morza lub inny krajobraz, początkowo pojawiłyby się trudności z jego rozpoznaniem i zaskoczenie. Następnie prawdopodobnie bylibyśmy zdezorientowani, ponieważ nie moglibyśmy prawidłowo zinterpretować tego, co widzimy. Dzieje się tak dlatego, że została uaktywniona rama „pokój", jako funkcja otwarcia drzwi i rama ta odgrywa ważną rolę w interpretacji napływającej informacji. Oczekiwania z tym związane dotyczą między innymi kształtu pokoju. Jeśli zobaczylibyśmy w pokoju łóżko uaktywniona zostałaby rama „sypialnia". Innymi słowy, nastąpiłby dostęp do najbardziej uszczegółowionej, dostępnej ramy. Taki proces uszczegółowiania nazywa się w kontekście teorii ram – rozpoznawaniem sterowanym przez ramy (ang. *frame-driven recognition*). W istocie, psycholodzy pokazali doświadczalnie, że rysunki obiektów są łatwiejsze do zidentyfikowania (w sensie czasu reakcji i stopy błędów) w ich naturalnym kontekście.

Od momentu pojawienia się, pojęcie ramy ewoluowało, stając się określeniem pewnej struktury danych bardzo wysokiego poziomu do opisu obiektów, która zazwyczaj składa się z nazwy ramy i pewnej liczby tzw. klatek (ang. *slots*), z których każda ma swoją nazwę i opisuje konkretną właściwość obiektu. W ramie nie mogą wystąpić dwie klatki o tej samej nazwie, podobnie jak w klatce nie może być dwóch faset o takiej samej nazwie. Każda klatka może zawierać zbiór tzw. faset. W najprostszym przypadku rama może zawierać nazwę obiektu i zbiór par <*atrybut*, *wartość*>, co koresponduje z zapisem <*obiekt*, *atrybut*, *wartość*>. Rama, o takiej uproszczonej budowie, przypomina budową rekordy znane z konwencjonalnych języków programowania. Ramy, w ogólniejszym przypadku, mogą jednak reprezentować bardziej złożoną wiedzę niż zwykły rekord. Ramy umożliwiają bowiem deklaratywną i proceduralną reprezentację wiedzy. Wartość jest bowiem jedną z wykorzystywanych w ramach faset, w tym wypadku chodzi o fasetę, którą czasami określa się jako VALUE. Dodatkowo mogą pojawiać się fasety ustalające dopuszczalne wartości danego atrybutu, wartości domyślne oraz procedury, które uruchamiane są automatycznie, gdy spełnione zostają pewne warunki. Przykładem mogą być następujące procedury:

- IF-ADDED – (jeśli dodane), procedury tego typu uruchamiane są wtedy, gdy nowa wartość dodawana jest do klatki. Mogą być użyte, np. do kontroli poprawności danych.

- IF-NEEDED – (jeśli potrzebne), procedury te są uruchamiane wtedy, gdy system chce pobrać informację z klatki, która nie zawiera jawnie reprezentowanej wartości. W szczególności, procedury te mogą zawierać algorytm wyliczający tę wartość, gdy podanie jej jawnie jest niewygodne lub wręcz niemożliwe (np. inżynier wiedzy nie zna jej „z góry" podczas tworzenia bazy wiedzy). Dla przykładu, jeśli rama zawiera informacje typu: *stawka_godzinowa* oraz *liczba_przepracowanych_godzin*, to atrybut *płaca* nie musi zawierać wartości explicite. Przy zmiennej płacy byłoby to w najlepszym razie rozwiązanie niewygodne. Wystarczy, że w tej sytuacji będzie dołączona procedura zawierająca algorytm obliczania płacy na podstawie dostępnych w ramie wartości innych atrybutów, w tym przypadku będących podstawą do wyliczenia płacy.
- IF-REMOVED – (jeśli usunięte), procedury te uruchamiają się, gdy określona wartość jest usuwana z klatki.

Ze względu na specyfikę tego rodzaju procedur, polegającą na automatycznym ich wykonaniu w razie potrzeby, są one w literaturze nazywane demonami.

Ramy mogą być łączone ze sobą, tworząc w ten sposób strukturę hierarchiczną (graf), przy czym wierzchołkami takiego grafu są ramy, a jego gałęzie określają relację podrzędności. W większości języków opartych na ramach, ramy dzielą się na dwie kategorie: klasy (ang. *class*) i egzemplarze (ang. *instances*). Klasa stanowi opis jednego lub więcej podobnych obiektów. Klasa może być podklasą innej klasy. Egzemplarze są ramami nie będącymi klasami dla innych ram i tworzone są na podstawie wzorca będącego klasą. Jedną z najciekawszych cech reprezentacji wiedzy w formie ram jest możliwość dziedziczenia właściwości ramy nadrzędnej przez ramę podrzędną, innymi słowy informacje zapisane w klatkach jednej ramy mogą być dziedziczone przez drugą. Powiązania pomiędzy poszczególnymi ramami definiują tzw. hierarchię dziedziczenia. Niektóre systemy dopuszczają tzw. wielodziedziczenie (ang. *multiple inheritance*), co oznacza, że jedna rama może dziedziczyć właściwości kilku ram nadrzędnych. Ramy i obiekty były po raz pierwszy zaimplementowane w takich językach jak FRL, KRL i Smalltalk.

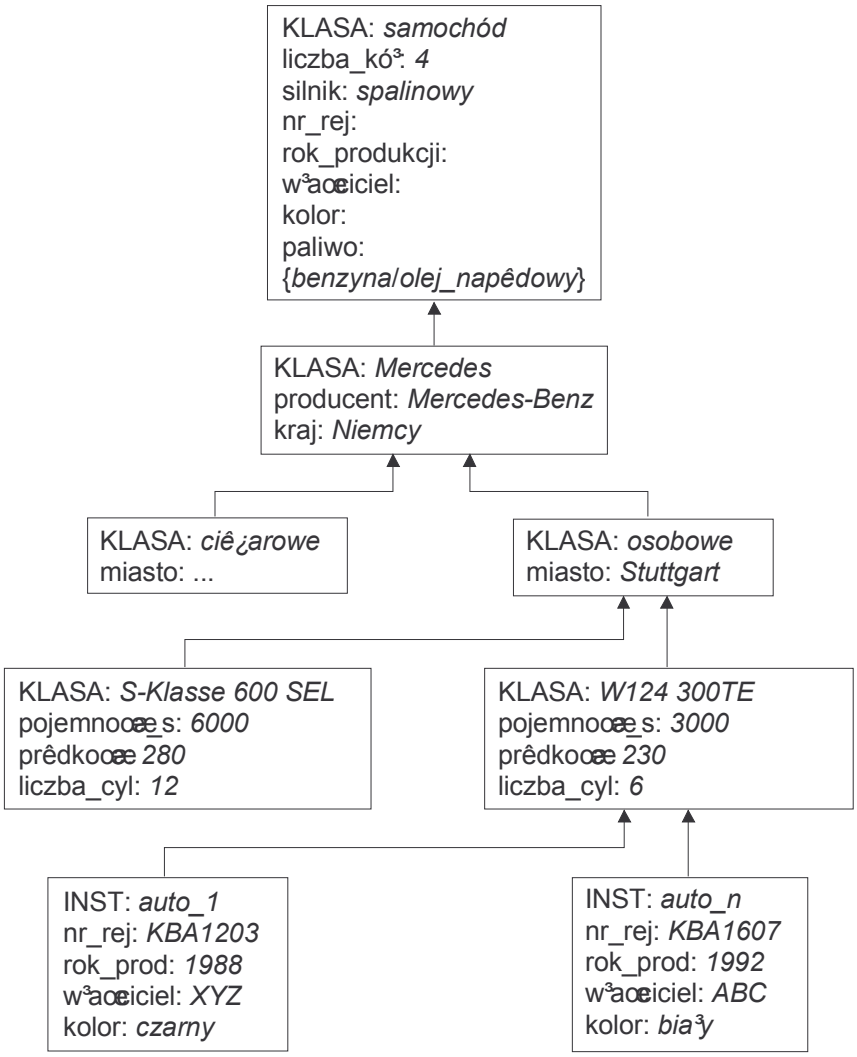
## PODSUMOWANIE

Podsumowując, można stwierdzić, że ramy charakteryzują się następującymi właściwościami:

- Umożliwiają jawną reprezentację struktury wiedzy, co jest mniej wyraźne w typowych systemach regułowych.
- Ramy dostarczają mechanizmów do budowy hierarchii (struktur taksonomicznych) i dziedziczenia atrybutów. Daje im to właściwość określaną jako oszczędność lub efektywność poznawczą (ang. *cognitive economy*). Cecha ta może mieć duże znaczenie w systemach z dużymi bazami wiedzy o rozbudowanej hierarchii obiektów.
- Zależność interpretacji od kontekstu, a także lokalny przepływ sterowania, są zgodne z postulatami modularyzacji w programowaniu. Cecha ta daje szansę „uporządkowania" wiedzy, przyczyniając się do wzrostu niezawodności jej kodowania. Właściwość ta może być osiągnięta w systemach tablicowych przez wyodrębnienie tzw. źródeł wiedzy.
- ramach mówi się, że umożliwiają reprezentowanie stereotypów, tj. typowych przykładów określonych obiektów. Zależnie od konkretnej implementacji jest to tylko częściowo prawdziwe, bowiem ramy często wprowadzają rozróżnienia pomiędzy cechami zasadniczymi, tzn. niezbędnymi do tego, by dany obiekt mógł być uważany za egzemplarz określonego pojęcia, a właściwościami, które często współwystępują z tymi podstawowymi (por. Jackson);
- Istotną cechą ram jest możliwość reprezentowania wyjątków (przez zastosowanie przesłaniania). Jednakże nadużycie tej cechy może prowadzić do sytuacji, że „cokolwiek" można zdefiniować za pomocą „czegokolwiek".

Przykład:

***Wybrany w przykładzie model samochodu jest jedynie pretekstem do pokazania koncepcji ram, stąd nie wszystkie dane muszą być poprawne.***



## PIONIERSKIE SYSTEMY EKSPERTOWE

### DENDRAL

Jeden z pierwszych systemów ekspertowych, opracowany w połowie lat sześćdziesiątych w Stanford University, przez zespół naukowców w składzie: Bruce Buchanan, Edward Feigenbaum oraz Joshua Lederberg (laureat Nagrody Nobla w dziedzinie chemii). Opracowanie tego systemu zajęło około 15 osobo-lat. Podstawowym zadaniem tego systemu było ustalanie struktury molekularnej nieznanych związków chemicznych. System opracowano z wykorzystaniem języka Interlisp. W systemie Dendral wykorzystano specjalny algorytm opracowany przez J. Lederberga, w celu systematycznego generowania wszystkich możliwych struktur cząsteczkowych. Wiedza w systemie Dendral jest reprezentowana zarówno w sposób proceduralny (generowanie struktur), jak i w formie reguł (moduł sterowany danymi) oraz ewaluacji. System osiągnął sprawność porównywalną, a w niektórych przypadkach przewyższającą, ekspertów-ludzi. System Dendral i pochodne od niego systemy stały się typowymi narzędziami w warsztacie zawodowym chemików.

### MYCIN

Prace nad systemem MYCIN rozpoczęły się w 1972 roku w ramach Projektu Programowania Heurystycznego realizowanego w Stanford University. Uważany dziś za niemal wzorcowy, był pierwszym dużym systemem ekspertowym o wysokim poziomie kompetencji w zakresie generowanych konkluzji. Pomaga lekarzom w wyborze terapii przeciwbakteryjnej dla pacjentów z chorobami infekcyjnymi krwi. System diagnozował przyczyny infekcji, poprzez identyfikację drobnoustroju odpowiedzialnego za jej powstanie. Ponadto Mycin proponował terapię poprzez wybór leku (typu i dawkowania).

Podstawowym sposobem reprezentacji wiedzy były reguły i fakty. W procesie rozwiązywania problemu wykorzystywał wnioskowanie wstecz. System zrealizowano z wykorzystaniem języka Lisp na maszyny DEC-20 (*mainframe*). Nakład pracy obliczono na 52 osobo-lata, z czego znaczna część zawarta jest w oprogramowaniu. W 1974 dokonano oceny konkluzji systemu przez grupę ekspertów-ludzi. Zaakceptowali oni 72% zaleceń systemu MYCIN w odniesieniu do 15 faktycznych przypadków infekcji. Testy porównawcze przeprowadzone w 1979 roku w odniesieniu do ulepszonej wersji tego systemu wykazały, że poprawność rozwiązań generowanych przez ten system dorównuje lekarzom ekspertom (z ośrodka Stanford) i przewyższa rozwiązania proponowane przez lekarzy niespecjalistów. Był to jednocześnie poważny dowód, że system ekspertowy może z powodzeniem rozwiązywać specjalistyczne problemy do tej pory „zarezerwo-wane" wyłącznie dla ekspertów-ludzi.

### INTERNIST/CADUCEUS

System został opracowany przez naukowców Harry Pople'a Jr. (informatyka) i Jacka D. Myersa (lekarza specjalistę w dziedzinie interny) z University of Pittsburgh. Prace rozpoczęto w 1974 roku. Podstawowym zadaniem systemu było diagnozowanie chorób bez dawania zaleceń co do sposobu leczenia. Liczba diagnozowanych jednostek chorobowych wynosiła 500. System osiągnął sprawność eksperta w zakresie około 85% medycyny wewnętrznej (interny). Od momentu powstania system przeanalizował wiele trudnych, klinicznych problemów. Jego baza wiedzy była jedną z największych wśród SE. Do opracowania systemu wykorzystano język INTERLISP.

## PROSPECTOR

Prace rozpoczęto w 1974 roku. Autorami systemu był zespół naukowców ze SRI International, w składzie: Peter Hart, Richard Duda, R. Reboh, K. Konolige, P. Barret i M. Einandi. Podstawowym zadaniem systemu było doradztwo w dziedzinie geologii, w szczególności pomoc przy poszukiwaniu złóż minerałów. W 1980 roku, jako test, system otrzymał dane geologiczne i geochemiczne, dostarczone przez grupę, która zakończyła w 1978 roku badania terenu Mt. Tolman (Waszyngton). W rezultacie analizy i sugestii systemu o możliwej obecności złóż molibdenu, przeprowadzono odpowiednie wiercenia. Wiercenia potwierdziły obecność bogatych złóż molibdenu o wartości około 100 mln USD, co stanowi niezwykle spektakularny sukces systemu ekspertowego. Czołowi naukowcy pracujący w tym projekcie, którzy opracowali system PROSPECTOR i KAS (narzędzie wywodzące się z systemu PROSPECTOR), odeszli z SRI i założyli specjalistyczną firmę o nazwie Syntelligence, znaną w dziedzinie systemów ekspertowych.

## CASNET

Prace nad systemem rozpoczęto na początku lat siedemdziesiątych w Rutgers University. System diagnozował stany chorobowe związane z jaskrą i proponował terapię leczniczą. Ponadto system wspiera swoje konkluzje odnośnikami do literatury fachowej. Wiedza w systemie CASNET była reprezentowana za pomocą sieci przyczynowej, będącej formą sieci semantycznej. System opracowano w języku Fortran. Sprawność systemu oceniono jako zbliżoną do poziomu ekspertów.

## R1/XCON

Prace nad systemem XCON, początkowo nazywanym R1, rozpoczęły się w 1979 roku. Jego twórcami byli naukowcy z Carnegie-Mellon University oraz grupa osób z firmy DEC. Głównym twórcą systemu był John McDermott z Wydziału Informatyki Uniwersytetu Carnegie-Mellon. Podstawowym zadaniem tego systemu było konfigurowanie komputerów VAX. Opracowany w 1979 roku prototyp liczył około 250 reguł i później rozwinął się do ponad 3000, co potwierdza przyrostowy sposób tworzenia baz wiedzy. Do opracowania systemu XCON wykorzystywano początkowo język OPS5.

Wielki sukces tego systemu polega na tym, że jest wykorzystywany w codziennej praktyce firmy DEC. Do 1986 roku przetworzył około 80.000 zamówień na komputery! Czas konfigurowania przez XCON wynosił poniżej minuty, przez ludzi około 20 minut. System osiągnął poprawność konfiguracji porównywalną ze specjalistami-ludźmi.

## WSPOMAGANIE DECYZJI EKONOMICZNYCH

Zastosowania ekonomiczne stanowią poważne wyzwanie dla systemów ekspertowych. Liczba praktycznych zastosowań SE w tej dziedzinie zaczęła wyraźnie wzrastać stosunkowo późno, bowiem dopiero w drugiej połowie lat osiemdziesiątych i w latach dziewięćdziesiątych. Dla ilustracji możliwych zastosowań w dziedzinie wspomagania decyzji ekonomicznych przedstawiono niżej kilka wybranych systemów z tej dziedziny:

### INVEST

Data opracowania: 1988.

Autorzy: Rezultat współpracy naukowców z Uniwersytetu w Karlsruhe oraz banków w Muenster.

Zadanie: Doradztwo w dziedzinie finansów, zwłaszcza w zakresie inwestycji. Podczas dialogu z urzędnikami bankowymi system uzyskuje informacje o życzeniach klientów i dostarcza dobrze uzasadnionych propozycji inwestycji. System udziela między innymi porad dotyczących papierów wartościowych i długoterminowego wzrostu kapitału. System INVEST opracowano za pomocą szkieletowego systemu ekspertowego DONALD, w którym podstawę reprezentacji wiedzy stanowią ramy.

### LENDING ADVISOR

Data opracowania: 1987.

Autorzy: Grupa naukowców z firmy Syntelligence przy Stanford Research Institute.

Zadanie: System pomaga w podejmowaniu decyzji kredytowych przez analizę podań o pożyczki. Jednym z jego zadań jest oszacowanie ryzyka związanego z udzieleniem pożyczki.

Użytkownicy: System jest adresowany do oddziałów banków udzielających pożyczek firmom o obrotach od 5 do 100 mln USD.

Pierwszą prototypową wersję systemu opracowano na maszynę Xerox D, wykorzystującą system Syntel, który jest wzmocnioną wersją systemu KAS.

### UNDERWRITING ADVISOR

Data opracowania: 1987.

Autorzy: firma Syntelligence przy współpracy z American International Group, Saint Paul Companies oraz Fireman's Fund Insurance.

Zadanie: Ocena ryzyka na podstawie podań o ubezpieczenie, w celu określenia wysokości płatności z tego tytułu. Ponadto każde z tych podań może być zapamiętane dla okresowych przeglądów. W ten sposób, mimo rozproszenia możliwe jest prowadzenie jednolitej polityki ubezpieczeniowej.

Użytkownik: System adresowany jest do firm ubezpieczeniowych, które mają wielu agentów rozproszonych na obszarze USA.

## AUDITOR

Data opracowania: 1985.

Autorzy: Naukowcy z Uniwersytetu w Illinois.

Zadanie: System wspomaga przeprowadzenie rewizji ksiąg w przedsiębiorstwach. W USA przedsiębiorstwa są zobowiązane do sporządzania raportów z działalności finansowej, zgodnie ze standardami GAAP (ang. *Generally Accepted Accounting Principles*). Firma, która nie przestrzega tych standardów, może stracić wiarygodność w oczach bankowców i akcjonariuszy. Przedsiębiorstwo musi między innymi sporządzić tzw. kartę bilansową, pokazującą aktywa i pasywa firmy. Do aktywów zalicza się m.in. długi klientów. Część należności z tytułu długów może być trudna do odzyskania z różnych powodów, np. bankructwa firmy, bezrobocia, załamania koniunktury, śmierci itd. Tego typu długów nie można szczegółowo oszacować a priori, natomiast łączna ich kwota musi być oszacowana w postaci tzw. funduszy asygnowanych z tytułu nieściągalnych długów (*Allowance for Bad Debts*). Zadaniem biegłych księgowych jest sprawdzenie zgodności bilansu. W związku z tym są oni żywotnie zainteresowani poprawnym oszacowaniem długów nieściągalnych. Pomoc w dokonaniu tego oszacowania jest podstawowym zadaniem systemu Auditor. Auditor jest systemem regułowym, zrealizowanym za pomocą narzędzia AL/X (Advice Language/X).

## ISAF (INTELIGENTNY SYSTEM ANALIZ FINANSOWYCH)

Data opracowania: 1993-1995.

Autorzy: firma AITECH, Artificial Intelligence Laboratory S.C. z Katowic.

Zadanie: System ISAF jest przeznaczony do diagnostyki kondycji finansowej przedsiębiorstw. Istnieją dwie wersje tego systemu, z których jedna służy do zewnętrznej oceny podmiotów gospodarczych, np. podczas badania zdolności kredytowej, druga natomiast do planowania i bieżącej kontroli planów finansowych.

System umożliwia monitoring finansowy firmy, rozumiany jako ciągłą obserwację zjawisk ekonomiczno-finansowych. Podstawowym zadaniem monitoringu jest przechowywanie i agregacja danych, umożliwiających długookresową analizę trendów i kierunków rozwoju przedsiębiorstwa. Pozwala to na budowę oryginalnych strategii działania lub korektę dotychczasowych kierunków, z uwzględnieniem kształtowania się sytuacji finansowej w branży.

Informacje niezbędne do budowy baz danych systemu ISAF pobierane są z systemów ewidencyjnych (finanse-koszty, gospodarka materiałowa, obrót towarowy itp.). System ISAF wyposażony jest w interfejsy do plików typu *dbf*, *oracle*, *txt* i innych. W przypadku trudności z uzyskaniem potrzebnych informacji z systemów ewidencyjnych, możliwe jest wprowadzanie danych z klawiatury. W tym celu zaprojektowano specjalne formularze, których wypełnienie możliwe jest w oparciu o okresowe zestawienia księgowe.

Prezentacja wyników pracy systemu odbywa się w formie:

- zestawień tabelarycznych,
- graficznej prezentacji informacji,
- interpretacji wyników analiz finansowych, z wykorzystaniem systemu ekspertowego PC-Shell, stanowiącego podsystem przetwarzania wiedzy systemu ISAF.

Użytkownicy: banki, towarzystwa leasingowe, firmy doradcze oraz wyższe uczelnie.

# 2

# INSTALACJA PAKIETU AITECHSPHINX

## INSTALACJA PAKIETU

Pakiet AitechSPHINX jest sprzedawany na płycie kompaktowej. Przed rozpoczęciem użytkowania pakietu należy dokonać jego instalacji. Wymagany jest prawidłowo zainstalowany system Windows 95 lub nowszy.

Pakiet jest przystosowany do instalacji indywidualnej na stanowiskach pracujących zarówno samodzielnie jak i w sieci. Aplikacje pakietu AitechSPHINX instalowane są indywidualnie, nie mniej możliwe jest umieszczenie np. baz wiedzy we wspólnym, sieciowym katalogu dla większej ilości użytkowników. Aby tego dokonać należy skontaktować się z administratorem sieci lokalnej.

Po włożeniu płyty CD-ROM do napędu program instalacyjny powinien uruchomić się automatycznie, z wyjątkiem sytuacji gdy opcja autostartu płyty jest wyłączona w ustawieniach systemu Windows. W tym przypadku należy uruchomić program **start.exe** znajdujący się w głównym katalogu płyty CD-ROM. Program startowy umożliwia uruchomienie instalacji pakietu AitechSPHINX, uruchomienie instalacji programu Adobe Acrobat Reader wymaganego do obsługi doumentacji elektronicznej. Trzecia opcja umożliwia zainstalowanie mechanizmu ODBC ( część starszych wersji Windows np. 95 wymaga ręcznego zainstalowania tegoż mechanizmu ).

Po prawidłowym uruchomieniu programu instalacyjnego pakietu AitechSPHINX, pojawia się okno dialogowe z przyciskiem *Opcje*. Przycisk *Opcje* umożliwia usawienie instalacji żądanych systemów w systemie. Domyślnie instalowane są wszystkie systemy. Wybranie przycisku *Instaluj* rozpoczyna proces instalacji. System instalowany jest w katalogu podanym przez użytkownika w oknie. Domyślnie ustawiony jest katalog C:\AitechSPHINX 4.5, lecz można go dowolnie zmienić. W czasie instalacji sprawdzana jest obecność w systemie tego katalogu. Jeżeli nie istnieje, to wtedy użytkownik pytany jest o zgodę na jego założenie. Jeżeli katalog istnieje, to wtedy proszony jest o potwierdzenie użycia go. Po prawidłowym podaniu katalogu rozpoczyna się proces instalacji katalogów oraz plików.

Po prawidłowym zainstalowaniu plików następuje proces utworzenia grupy *AitechSPHINX 4.5* w menu *Start*, zawierającej ikony programów pakietu AitechSPHINX. W obecnej program instalacyjny generuje zapytanie o to czy utworzyć skróty dla wszystkich użytkowników danego komputera czy tylko dla tylko dla użytkownika aktualnie instalującego pakiet.

Po prawidłowym zainstalowaniu systemu program instalacyjny po potwierdzeniu zostaje zamknięty. Programy pakietu AitechSPHINX są od tej chwili dostępne w menu *Start* w grupie *AitechSPHINX 4.5*.

**Uwaga!**

Wskazane jest, by podczas instalacji w systemie Windows nie była uruchomiona żadna inna aplikacja. W przypadku zaistnienia konfliktów używania bibliotek dll przez inne aplikacje należy je zamknąć i ewentualnie uruchomić ponownie system Windows, po czym rozpocząć instalację od nowa.

## DEINSTALACJA PAKIETU

W przypadku prawidłowego zainstalowania pakietu AitechSPHINX 4.5, możliwe jest późniejsze usunięcie oprogramowania oraz baz wiedzy z systemu. Dokonać tego można poprzez *Panel sterowania* uruchamiając aplikację *Dodaj/usuń programy* a następnie wybranie na zakładce *Instaluj/Odinstaluj* wpisu *Pakiet AitechSPHINX 4.5*. Program deinstalacyjny usuwa pliki pakietu oraz dodatkowo usuwa grupę *Pakiet AitechSPHINX* z menu startowego systemu Windows.

# 3

# PRZEWODNIK PO SYSTEMIE PC-SHELL

## ARCHITEKTURA SYSTEMU PC-SHELL

System PC-Shell stanowi główny element pakietu narzędziowego firmy AITECH. Po instalacji pakietu jest on obecny jako jeden z elementów utworzonej grupy, identyfikowany jest przez ikonę:

## OGÓLNA CHARAKTERYSTYKA SYSTEMU PC-SHELL

System PC-Shell jest systemem silnie hybrydowym o następujących cechach:

1. W zakresie struktury systemu:
   - elementy architektury tablicowej;
   - pełna integracja w ramach systemu PC-Shell systemu ekspertowego oraz symulatora sieci neuronowej.
2. W zakresie reprezentacji wiedzy:
   - deklaratywna reprezentacja wiedzy w formie reguł i faktów;
   - algorytmiczna (proceduralna) reprezentacja wiedzy w formie programu zawartego w bloku sterowania (*control*);
   - pełne rozdzielenie wiedzy eksperckiej i procedur sterowania;
   - wiedza o charakterze rozproszonym zawarta w sieci neuronowej;
   - wiedza ekspercka może być zawarta w kilku źródłach wiedzy.
3. W zakresie wyjaśnień:
   - wyjaśnienia typu „jak?" (ang. *how*),
   - wyjaśnienia typu „dlaczego?" (ang. *why*),
   - wyjaśnienia typu „co to jest?" (ang. *what is)*, dla pojęć zawartych w bazie wiedzy,
   - *metafory* dotyczące reguł użytych w bazie wiedzy.

## STRUKTURA SYSTEMU PC-SHELL

System PC-Shell składa się z następujących elementów (rys 3-1):

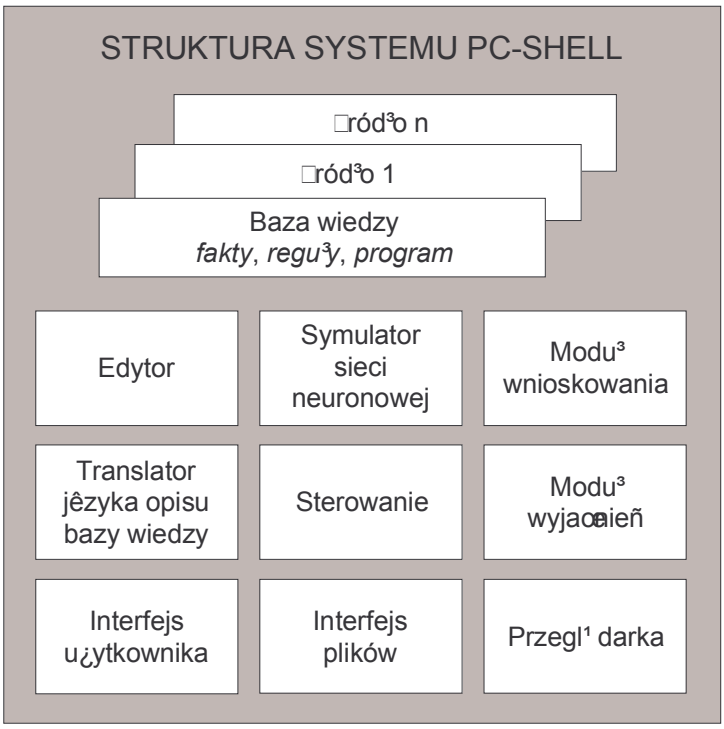
1. modułu sterującego,
2. translatora języka opisu bazy wiedzy,
3. modułu wnioskującego,
4. modułu wyjaśnień,
5. symulatora sieci neuronowych,
6. zewnętrznego edytora bazy wiedzy,
7. interfejsu użytkownika,
8. interfejsu do plików dyskowych.

## MODUŁ STERUJĄCY

Podstawowym zadaniem modułu sterującego jest koordynowanie wszelkich procesów realizowanych przez system PC-Shell. Jednym z jego zadań jest komunikacja z użytkownikiem poprzez interfejs użytkownika. Interfejs ten pracuje w trybie tekstowym i opracowany został w taki sposób, by sugerował użytkownikowi określone warianty działania w danym kontekście, uwalniając go tym samym od konieczności ich pamiętania.

## TRANSLATOR JĘZYKA OPISU BAZY WIEDZY

Systemy ekspertowe rozwiązują problemy wykorzystując wiedzę ekspercką zawartą w ich bazie wiedzy. Wiedza ta może pochodzić z różnych źródeł, na przykład z książek, najczęściej jednak pozyskiwana jest ona od ludzi – specjalistów z określonej dziedziny (ekspertów).




RYS. 3-1. STRUKTURA SYSTEMU PC-SHELL

Zanim wiedza ekspercka zostanie wprowadzona do bazy wiedzy, musi być wcześniej odpowiednio sformalizowana (skodyfikowana). Do tego celu w systemie PC-Shell wykorzystuje się tzw. język opisu bazy wiedzy. Wyrażony w tym języku opis bazy wiedzy może być utworzony i zapisany w pamięci dyskowej komputera za pomocą dowolnego edytora tekstowego pracującego w kodzie ACII. Zadaniem translatora języka opisu bazy wiedzy jest czytanie pliku dyskowego zawierającego opis bazy wiedzy, tłumaczenie i wprowadzenie odpowiedniego kodu do bazy systemu PC-Shell, mieszczącej się w całości w pamięci operacyjnej komputera.

## MODUŁ WNIOSKUJĄCY

Zadaniem modułu wnioskującego jest rozwiązywanie problemów z wykorzystaniem wiedzy zawartej w bazie wiedzy. Do tego celu wykorzystywane są odpowiednie procedury wnioskowania (rozumowania). Moduł wnioskujący systemu PC-Shell począwszy od wersji 4.1 wykorzystuje wnioskowanie wstecz (ang. *backward chaining*) oraz wnioskowanie w przód (ang. *forward chaining*).

System PC-Shell zapewnia dwa tryby konsultacji:
- konwersacyjny;
- programowy, sterowany programem zawartym w bloku *control* bazy wiedzy.

## MODUŁ WYJAŚNIEŃ

Użytkownicy systemów ekspertowych często są zainteresowani sposobem rozwiązania problemu postawionego systemowi. Zdolność systemu ekspertowego do wyjaśnień, w tym zwłaszcza konkluzji i potwierdzanych hipotez, jest jedną z najważniejszych cech tej klasy systemów, odróżniającą tę technologię od konwencjonalnych programów.

Wyjaśnienia udzielane przez system powinny przekonać użytkownika do poprawności zastosowanego rozumowania. Jednakże mogą one być również podstawą do weryfikacji bazy wiedzy, jeśli użytkownik stwierdzi, że wyjaśnienia przedstawione przez system są nieprzekonujące. Wydaje się, że wyjaśnienia można również traktować jako narzędzie uruchomieniowe, ułatwiające lokalizowanie nieprawidłowości w bazie wiedzy, zwłaszcza w początkowych etapach jej tworzenia.

System PC-Shell dostarcza retrospektywnych wyjaśnień typu *jak*, rozumowania, które umożliwiło wyprowadzenie danego zbioru konkluzji lub potwierdzenie postawionej hipotezy. W czasie konsultacji system ekspertowy często pyta o obecność określonych symptomów (faktów). W takich sytuacjach użytkownik może mieć wątpliwości, czy zadane mu pytanie ma związek (lub jakiego rodzaju jest to związek) z rozwiązywanym problemem. W tym celu system PC-Shell dostarcza wyjaśnień typu „dlaczego?" (ang. *"why" explanations*). W ramach tego typu wyjaśnień system pokazuje użytkownikowi, jaka hipoteza jest rozważana oraz w jaki sposób odpowiedź na pytanie systemu dostarczy informacji niezbędnej do potwierdzenia bądź odrzucenia tej hipotezy. PC-Shell dostarcza również wyjaśnień w formie tzw. *metafor*, będących objaśnieniami do reguł prezentowanych podczas wyjaśnień typu *how*. Ponadto dostępne są również wyjaśnienia typu „co to jest?", będące tekstowymi objaśnieniami pojęć zawartych w bazie wiedzy. Należy podkreślić, że system PC-Shell umożliwia kontrolowanie zakresu oraz głębokości wyjaśnień.

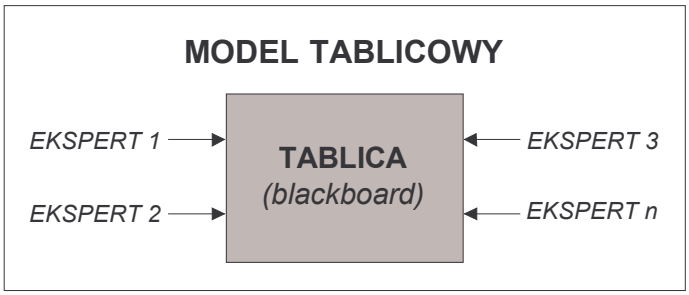
## MODUŁ OPERACJI NA BAZIE WIEDZY

Moduł operacji na bazie wiedzy umożliwia wykonywanie następujących funkcji:

1. przeglądanie bazy wiedzy;
2. usuwanie faktów zawartych w bazie wiedzy;
3. dodawanie faktów do bazy wiedzy.

## ELEMENTY ARCHITEKTURY TABLICOWEJ W SYSTEMIE

Istota systemów tablicowych opiera się na analogii ze sposobem, w jaki może współpracować kilku ekspertów rozwiązujących pewien problem (rys. 3-2). Każdy z nich włącza się do rozwiązania wspólnego problemu w zakresie jego kompetencji, a tworzone przez nich rozwiązania pośrednie zapisywane są na tablicy. Tablicowe systemy ekspertowe (ang. *blackboard systems*) oparte są na takiej właśnie metaforze.




RYS. 3-2. ISTOTA MODELU TABLICOWEGO

W systemach tablicowych wiedza ekspercka nie jest zgromadzona w jednym miejscu (np. pliku), lecz rozproszona w postaci kilku wyodrębnionych fizycznie modułów, zwanych źródłami wiedzy (ang. *knowledge sources*). Poszczególne źródła wiedzy odpowiadają ekspertom z podanej metafory. Ponadto zakłada się istnienie wspólnej, tj. dostępnej dla wszystkich źródeł, tablicy komunikacyjnej (ang. *blackboard*), będącej rodzajem pamięci roboczej. Każde źródło może posiadać własny mechanizm wnioskujący.

W systemie PC-Shell zaimplementowano niektóre elementy przedstawionego modelu tablicowego. W obecnej wersji system umożliwia podzielenie bazy wiedzy na mniejsze fragmenty, wyodrębnione tematycznie (problemowo). Narzędziem umożliwiającym załadowanie i uaktywnienie wybranych źródeł wiedzy są instrukcje *getSource* i *solve*. Nie zakłada się formalnie wyodrębnionych mechanizmów wnioskujących przypisanych do źródeł, ponieważ w przypadku komputerów sekwencyjnych nie ma to większego znaczenia. Natomiast dostępne w systemie PC-Shell instrukcje inicjujące mechanizm wnioskujący pozwalają osiągnąć podobny efekt praktyczny. Rolę tablicy pełni tu część obszaru pamięci operacyjnej, w którym umieszczana jest baza wiedzy.

W systemie PC-Shell wymaga się istnienia modułu głównego bazy wiedzy, począwszy jednak od wersji 2.1 możliwe jest wczytywanie i testowanie samodzielnych źródeł. Wyłącznie moduł główny bazy wiedzy może zawierać opisy plików oraz program. Źródła wiedzy mogą zawierać opisy faset, reguły i/lub fakty. W

przypadku zadeklarowania źródeł wiedzy moduł główny nie może zawierać reguł i faktów, pełniąc jedynie rolę sterującą oraz funkcję kontrolną.

Podstawową korzyścią z podzielenia bazy wiedzy na odrębne źródła jest możliwość rozwiązywania problemów, dla których niezbędne są duże i bardzo duże bazy wiedzy. Zyskuje się przez to na szybkości działania systemu oraz na oszczędności pamięci operacyjnej. Rozwiązanie to ułatwia również proces utrzymywania bazy wiedzy, stanowiąc udogodnienie dla inżyniera wiedzy.

## SYMULATOR SIECI NEURONOWEJ

W dotychczasowym rozwoju badań w zakresie sztucznej inteligencji można wyodrębnić dwa podstawowe nurty:

- symboliczny, związany z reprezentacją wiedzy za pomocą symboli;
- konekcjonizm, związany z badaniami w zakresie sieci neuronowych.

Często te dwa podejścia są przeciwstawiane sobie i zależnie od zainteresowań autorów publikacji jedno z nich jest przedstawiane jako lepsze lub gorsze. Autor traktuje oba podejścia jako komplementarne (przynajmniej na obecnym etapie rozwoju obu technologii), jako że każde z nich ma słabe i mocne strony.

Konsekwencją takiego sposobu myślenia o obu podejściach do rozwiązywania problemów jest hybrydowa architektura systemu PC-Shell, integrująca system ekspertowy i symulator sieci neuronowej. Połączenie to nie jest jedynie mechaniczne, lecz nastąpiła tu integracja zarówno na poziomie architektury systemu jak i języka reprezentacji wiedzy. Symulator sieci neuronowej występuje tu w podobnej roli jak źródło wiedzy w ramach architektury tablicowej. To podobieństwo wyraża się nie tylko w zakresie funkcjonalnym, ale również formalnym, dotyczącym sposobu reprezentacji wiedzy. Korzyścią płynącą z takiego rozwiązania i oryginalną cechą systemu PC-Shell jest możliwość tworzenia hybrydowych aplikacji, gdzie odbywa się pełna kooperacja pomiędzy symulatorami sieci neuronowych a systemem ekspertowym. Dla przykładu, obie technologie mogą być wykorzystane do rozwiązania określonych podproblemów w ramach większego problemu, a rozwiązania cząstkowe, dzięki pełnej integracji, mogą być swobodnie przekazywane z jednej technologii do drugiej.

Aplikacje oparte o sieci neuronowe tworzone są za pomocą systemu Neuronix, którego głównym zadaniem jest uczenie sieci i wygenerowanie pliku z definicją sieci (pliki *.*npr*) oraz pliku zawierającego wagi (pliki *.*wag*). Zadanie inżyniera wiedzy w systemie PC-Shell sprowadza się, po utworzeniu wspomnianych plików, do zadeklarowania pliku definiującego sieć w bloku sterowania. Ogólny schemat aplikacji hybrydowej przedstawiono na rys. 13 (część 2. dokumentacji).

***Formalny sposób deklarowania i uruchamiania symulatora sieci neuronowej został opisany w „Podręczniku inżyniera wiedzy" (część 2. dokumentacji systemu PC-Shell).***

## GŁÓWNE OPCJE SYSTEMU

Główne menu systemu PC-Shell zawiera zbiór opcji, które w dalszej części rozdziału będą opisane bardziej szczegółowo. Ich opis uporządkowany jest według kolejności w jakiej poszczególne opcje występują w menu.

## MENU *PLIK*

Menu zawiera wykaz bezpośrednich operacji na plikach baz wiedzy.

### Otwarcie

Opcja służy do translacji i ładowania wskazanej bazy wiedzy. Po jej wybraniu pojawia się standardowe okienko dialogowe systemu Windows do wyboru plików, w którym użytkownik może wybrać interesującą go bazę wiedzy.

### Zamknięcie

Opcja służy do zakończenia pracy z aktywną bazą wiedzy. Powoduje zwolnienie pamięci.

### Retranslacja

Opcja służy do ponownej translacji i załadowania tej samej bazy wiedzy. Jej zastosowanie związane jest z tym, że do edycji baz wiedzy stosujemy zewnętrzne edytory. Po zmianie za pomocą edytora treści bazy wiedzy i jej zapisaniu, należy dokonać retranslacji w celu załadowania zmienionej bazy wiedzy, bez konieczności wyboru nazwy przy pomocy opcji *Otwarcie*.

### Wyjście

Opcja umożliwia zakończenie pracy i opuszczenie systemu PC-Shell.

W menu Plik znajduje się pięć pozycji, gdzie są wpisywane nazwy do pięciu ostatnio używanych baz wiedzy. Opcje te służą do szybszego wyboru baz, które są często uruchamiane w systemie. Należy tu zaznaczyć, że system PC-Shell może być także uruchamiany z parametrem w postaci nazwy bazy wiedzy, która ma być automatycznie załadowana do systemu:

```
pcshell <nazwa_pliku_bazy_wiedzy> [Enter]
```

## MENU EDYCJA

Opcja ta umożliwia edycję bazy wiedzy. Edycja odbywa się za pomocą wbudowane w system edytora wewnętrznego lub dowolnego, zewnętrznego edytora, obsługującego kod ASCII. W przypadku zewnętrznego edytora jest nim, obecny w każdej wersji systemu Windows, Notatnik (ang. *Notepad*), jednakże istnieje możliwość korzystania z dowolnego edytora, (patrz menu *Opcje*|*Katalogi*). O tym czy użytkownik korzysta z edytora wewnętrznego czy innego zewnętrznego decyduje ustawienie opcji w menu *Opcje*|*Katalogi*.

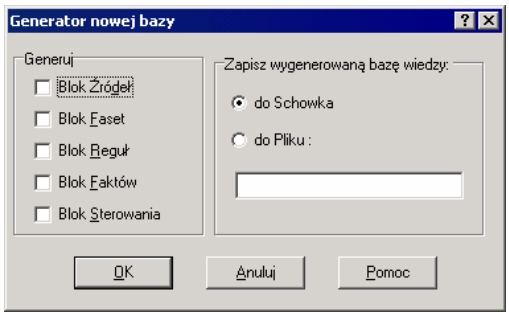
Opcja *Edycja* jest zaimplementowana w taki sposób, że rozpoznaje kontekst pracy systemu. Jeśli w systemie jest załadowana baza wiedzy, to wtedy wybranie tej opcji uruchamia edytor wraz z nazwą bieżącej bazy wiedzy. Natomiast w przypadku, gdy w systemie nie jest załadowana żądana baza, wtedy uruchamiany jest generator baz wiedzy (rys. 3-3).

Okno dialogowe generatora umożliwia użytkownikowi automatyczne wpisanie wybranych bloków bazy wiedzy bezpośrednio do pliku lub poprzez schowek do edytora tekstów. W oknie tym należy zaznaczyć bloki, które mają być wpisane do tworzonej bazy, następnie wybrać sposób generacji i nacisnąć przycisk OK.

Podanie nazwy bez ścieżki dostępu powoduje wygenerowanie pliku w domyślnym katalogu baz wiedzy (patrz menu *Opcje|Katalogi*).

Chcąc mieć dostęp do generatora w przypadku, gdy w systemie jest już załadowana jakaś baza wiedzy, należy wcześniej wybrać z menu opcję *Plik|Zamknięcie*, a następnie *Edycja*.

***Opis menu edytora dostępny jest w rozdziale 11 dokumentacji systemu CAKE.***



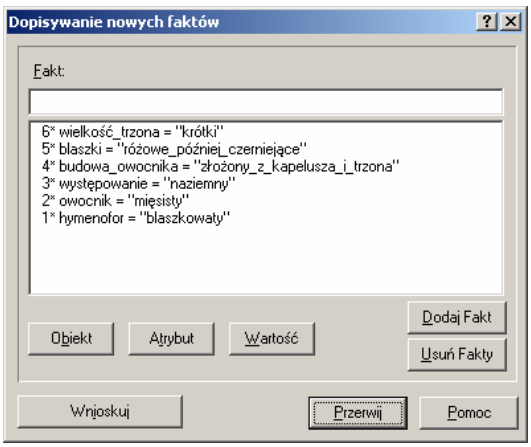

RYS.3-3. GENERATOR NOWEJ BAZY WIEDZY

## MENU WNIOSKOWANIE

Menu służące do uruchomienia mechanizmów wnioskowania w trybie konwersacyjnym lub programu zawartego w bloku *control* załadowanej bazy wiedzy.

### Do przodu

Opcja powoduje uruchomienie modułu wnioskującego systemu i zastosowanie wnioskowania do przodu. Po jej wybraniu pojawia się okno (Rys. 3-4) pokazujące listę faktów wraz z opcjami do ewentualnej jej modyfikacji oraz przyciskiem **Wnioskuj** który uruchamia proces wnioskowania w przód.



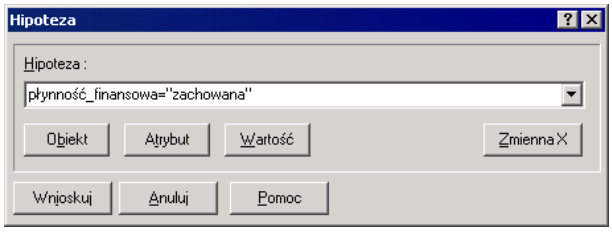

RYS. 3-4 LISTA FAKTÓW BRANYCH POD UWAGĘ W TRAKCIE WNIOSKOWANIA W PRZÓD

Po zakończeniu wnioskowania pojawia się okno rozwiązań zawierające wszystkie rozwiązania wygenerowane przez system (opisane dalej w dokumentacji).

Uwaga ! Po zamknięciu okna 3-4 dopisane fakty pozostają w bazie wiedzy !

### Do tyłu

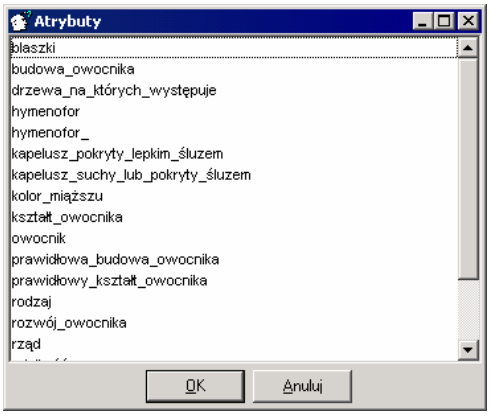
Wybranie tej opcji powoduje uruchomienie wnioskowania „wstecz" w trybie konwersacyjnym. W trybie konwersacyjnym (w odróżnieniu od programowego) użytkownik musi w sposób jawny wprowadzić treść hipotezy-celu (rys. 3-5). Po przekazaniu hipotezy system rozpoczyna proces wnioskowania kończący się potwierdzeniem hipotezy z przekazaniem rozwiązania lub informacją, że hipoteza nie została potwierdzona.




RYS.3-5. OKNO WPROWADZANIA HIPOTEZY

Hipoteza może być wprowadzona dwoma sposobami: „ręcznie" poprzez wpisanie hipotezy lub przy użyciu dostępnych przycisków, które udostępniają wykazy obiektów, atrybutów i wartości (rys. 3-6). Należy podkreślić, że wartość atrybutu może być wprowadzana dopiero po wpisaniu/wybraniu atrybutu. Okno z wykazem wartości pojawia się tylko wtedy, gdy zostały one zadeklarowane w bloku faset. Dodatkowo, okno hipotezy jest wyposażone w przycisk *„Zmienna X"*, który automatycznie dopisuje do hipotezy jako wartość identyfikator zmiennej *X*.

Po prawidłowym podaniu hipotezy i naciśnięciu przycisku *„Wnioskuj"* uruchamiany jest mechanizm wnioskowania. Podczas wnioskowania użytkownik może być zapytany przez system o potrzebne dane, których nie ma w bazie wiedzy. W rezultacie pracy modułu wnioskującego i ewentualnej konsultacji z użytkownikiem pojawia się rozwiązanie. Dokładne omówienie okien dialogowych mogących pojawić się w procesie wnioskowania i sam proces wnioskowania jest opisany bardziej szczegółowo w dalszej części tego rozdziału.




RYS. 3-6. WYKAZ ATRYBUTÓW DOSTĘPNYCH W SYSTEMIE

### Wykonanie programu

Wybranie tej opcji powoduje uruchomienie programu zawartego w bloku *control* załadowanej bazy wiedzy.

### Debugger

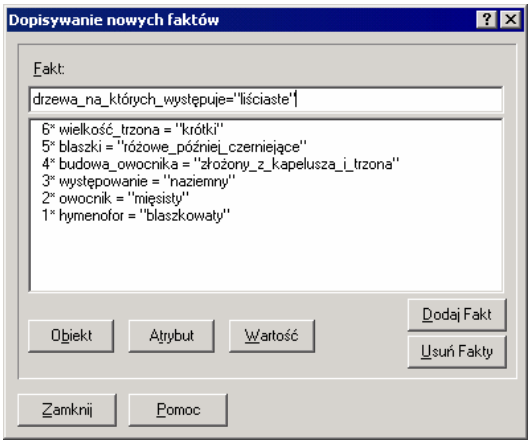
Opcja ta uruchamia moduł tzw. debuggera – systemu wykonywania krokowego bloku sterowania, umożliwiającego kontrolę poprawności bloku sterowania. Dokładnie moduł ten opisany jest w rozdziale 15 w *Podręczniku inżyniera wiedzy*.

## MENU NARZĘDZIA

Menu zawiera zbiór opcji do przeglądania bazy wiedzy i modyfikacji faktów.

### Dopisywanie faktów

Opcja udostępnia okno dialogowe, w którym użytkownik może interakcyjnie wprowadzać nowe fakty do bazy wiedzy oraz – po wcześniejszym wskazaniu – usuwać fakty już zapamiętane w bazie wiedzy.




RYS. 3-7. DOPISYWANIE NOWYCH FAKTÓW

Podczas wprowadzania nowych faktów, okno to udostępnia podobny mechanizm jak w oknie hipotez, z możliwością automatycznego wprowadzania atrybutu, obiektu i wartości poprzez wybór z odpowiedniego wykazu. Należy zauważyć, że w odróżnieniu od hipotezy, fakt nie może zawierać zmiennej, a wartości muszą być podane w postaci liczby lub łańcucha znaków.

Usuwanie odbywa się poprzez zaznaczenie w omawianym oknie faktów do usunięcia i naciśnięciu przycisku *Usuń fakty*.

### Usuwanie faktów⇨Wszystkie fakty

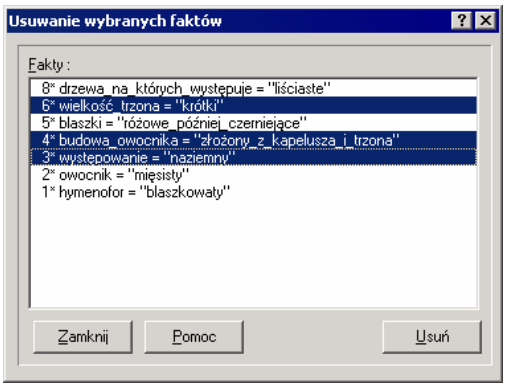
Opcja umożliwia usunięcie wszystkich faktów z bieżącej bazy wiedzy. Użytkownik jest pytany o potwierdzenie polecenia usunięcia wszystkich faktów.

### Usuwanie faktów⇨Tylko nowe fakty

Opcja umożliwia usunięcie nowych faktów tzn. tych faktów, które nie pochodzą z bloku facts bazy wiedzy.

### Usuwanie faktów⇨Fakt o wskazanej pozycji

Opcja wyświetla okno bardzo podobne do okna dodawania faktów, z tym ograniczeniem, że w oknie tym użytkownik może wyłącznie usunąć wybrane fakty z wykazu prezentowanego w oknie (bez możliwości dopisywania nowych faktów).




RYS. 3-8. USUWANIE WYBRANYCH FAKTÓW

### Przeglądanie bazy wiedzy⇨Cała baza wiedzy

Opcja służy do wyświetlenia w oknie wszystkich faktów i reguł zawartych w bieżącej bazie wiedzy.

W pierwszej kolejności są wyświetlane fakty i to w porządku od „najnowszego" do „najstarszego". Fakty poprzedzane są unikatowym numerem, jednoznacznie identyfikującym dany fakt. Numer ten nie występuje w tekście źródłowym bazy wiedzy i ma znaczenie tylko w podczas pracy systemu. Następnie są wyświetlane reguły zgodnie z kolejnością w bazie wiedzy.

### Przeglądanie bazy wiedzy⇨Tylko fakty

Opcja wyświetla wszystkie fakty znajdujące się w bazie wiedzy.

### Przeglądanie bazy wiedzy⇨Tylko nowe fakty

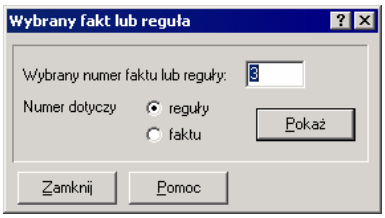
Opcja wyświetla wykaz wszystkich nowych faktów, tzn. wszystkich poza zawartymi w bloku facts bazy wiedzy.

### Przeglądanie bazy wiedzy⇨Tylko reguły

Opcja wyświetla wszystkie reguły znajdujące się w bazie wiedzy.

### Przeglądanie bazy wiedzy⇨Pojedyncze fakty lub reguły

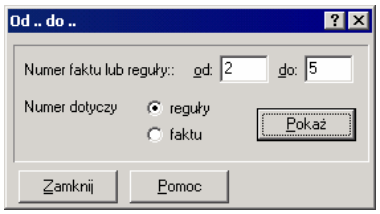
Opcja umożliwia prezentację wskazanej reguły lub faktu, poprzez podanie ich identyfikatora (numeru) w oknie dialogowym tej opcji. Należy zauważyć, że numery faktów i reguł są dwojakiego rodzaju. Pole Numer reguły oznacza numer reguły nadany przez inżyniera wiedzy w tekście źródłowym bazy wiedzy lub numer kolejny reguły przypisany automatycznie przez system w przypadku braku numeracji w bloku rules. Natomiast numer faktu oznacza numer względny, związany z położeniem faktu w bazie wiedzy. Numer względny jest różny od numeru jawnie identyfikującego fakt, po którym podczas przeglądania pojawia się znak „*" (gwiazdka).




RYS. 3-9. PRZEGLĄDANIE WYBRANEGO FAKTU LUB REGUŁY

### Przeglądanie bazy wiedzy⇨Od .. do ..

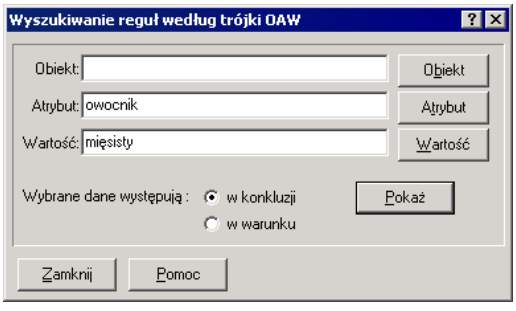
Opcja podobna do poprzedniej z tym, że umożliwia przegląd reguł lub faktów z podanego zakresu numeracji.




RYS. 3-10. PRZEGLĄDANIE FAKTÓW LUB REGUŁ Z PODANEGO ZAKRESU

### Przeglądanie bazy wiedzy⇨Reguły wg trójki OAW

Ta opcja służy do bardziej zaawansowanego wyszukiwania reguł. Udostępnia ona możliwość przeglądania reguł według treści trójki OAW. Po podaniu przez użytkownika atrybutu, wartości i ewentualnie identyfikatora obiektu system wyszukuje i wyświetla wszystkie reguły, które w swej treści zawierają podaną informację.

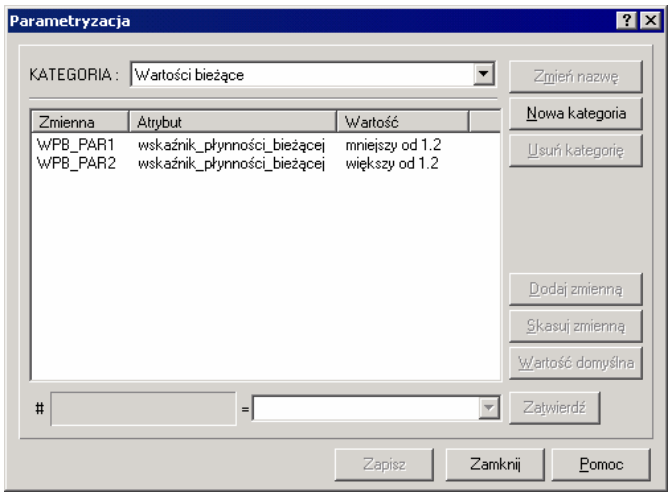



RYS. 3-11. WYSZUKIWANIE REGUŁ WEDŁUG ZADANYCH KRYTERIÓW

Przełącznik *„w konkluzji"* oraz *„w warunku"* umożliwia wybranie czy kryteria dotyczą przeszukiwania konkluzji czy warunków. Zgodnie z przedstawionym na rys. 3-11 przykładem system wyświetli wszystkie reguły, które składają się z konkluzji zawierającej informację „owocnik=mięsisty".

## Parametryzacja

Opcja parametryzacji służy do tworzenia, przeglądania oraz modyfikacji zmiennych parametrycznych. Opcja jest dostępna dla baz wiedzy zawierających zmienne parametryczne. Wybranie tej opcji powoduje otwarcie okna parametryzacji (rys. 3-12).




RYS. 3-12. OKNO PARAMETRYZACJI BAZ WIEDZY

Po wybraniu opcji wyświetla się okno parametryzacji, które zawiera m.in. listę zdefiniowanych kategorii oraz przyporządkowanych im zmiennych parametrycznych. Dwie pierwsze kategorie to tzw. *kategorie systemowe*. Pierwsza z nich („*Wartości domyślne – tylko-do-odczytu”*), zawiera listę wszystkich zmiennych parametrycznych wraz z ich wartościami domyślnymi. Wartości zmiennych tej kategorii nie można zmieniać. Druga kategoria („*Wartości bieżące”*) zawiera listę wszystkich wartości zmiennych parametrycznych wraz z ich bieżącymi wartościami. Wartości te są dostępne z poziomu bloku *control*. Za kategoriami systemowymi znajdują się kategorie tworzone przez użytkownika systemu.

Definicja zmiennych wybranej kategorii jest wyświetlana w arkuszu kalkulacyjnym w postaci dwóch kolumn. W pierwszej kolumnie (*Nazwa zmiennej*) wyświetlona jest nazwa zmiennej wraz z nazwą atrybutu, którego dotyczy dana zmienna, natomiast w drugiej kolumnie znajduje się bieżąca wartość zmiennej. Użytkownik może dokonywać zmian tylko w tej kolumnie. Podczas edycji wartości jest przeprowadzana kontrola poprawności wartości oraz typu, zgodnie z deklaracją odpowiedniej fasety *val*. W przypadku faset *val oneof* lub *val someof* możliwy jest wybór ze zbioru określonych w fasecie wartości. Lista tych wartości jest wyświetlana po naprowadzeniu kursora na odpowiednią komórkę tablicy i dwukrotnym naciśnięciu lewego przycisku myszy. Lista nie jest wyświetlana w trybie edycji, gdy użytkownik wprowadza wartość ręcznie. W takiej sytuacji, jeśli chcemy wybrać wartość z listy, to należy wcześniej zakończyć edycję, np. przez naciśnięcie klawisza *Esc*.

W prawej części okna znajdują się przyciski umożliwiające dodawanie i usuwanie zmiennych z danej kategorii. Usunięcie dotyczy zmiennych z zaznaczonego obszaru (ważne są wiersze) lub z bieżącej zmiennej.

Funkcja (przycisk) „*Dodaj zmienną”* umożliwia wyświetlenie listy zmiennych parametrycznych, nie występujących w danej kategorii. Po wybraniu interesujących nas zmiennych i po naciśnięciu klawisza *OK* wybrane zmienne zostaną dodane do kategorii wraz z wartościami domyślnymi.

Funkcja „*Wartość domyślna”* powoduje przywrócenie zmiennym, wybranym w arkuszu, ich wartości domyślnych, określonych przez inżyniera wiedzy w bloku faset, na etapie tworzenia bazy wiedzy.

Poza wymienionymi wyżej przyciskami są jeszcze trzy ogólne przyciski do zarządzania całymi kategoriami. Po wykonaniu funkcji „*Nowa kategoria”* wyświetlane jest okienko, w którym podajemy nazwę nowej kategorii i zaznaczamy, czy chcemy skopiować zawartość kategorii, którą poprzednio edytowaliśmy do nowo tworzonej wraz z wartościami z tej kategorii. Funkcja „*Zapisz kategorię”* służy do zapamiętania kategorii w pliku typu .ini.

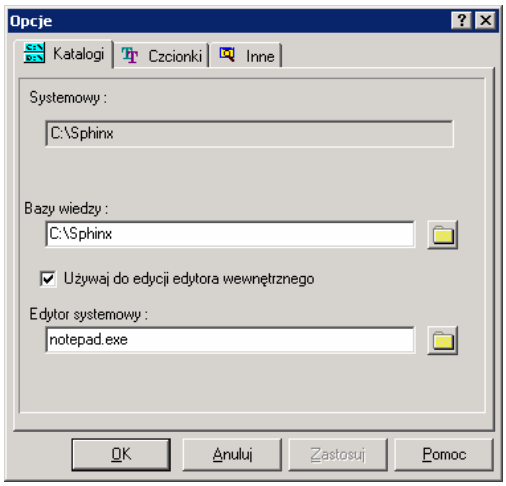
## MENU OPCJE

### Opcje

Ustalenie konfiguracji systemu PC-Shell. Wybranie tej opcji powoduje wyświetlenie okna dialogowego „Opcje" z zakładkami:

### Katalogi

Zakładka na której użytkownik ustawia domyślną ścieżkę do katalogu baz wiedzy, a także ustala jaki edytor użytkownik chce wykorzystywać do edycji baz wiedzy. Należy zaznaczyć, że wszelkie nazwy pliku (np. ze źródłami wiedzy) podawane bez ścieżki dostępu w bazach wiedzy, odnoszą się do katalogu określonego w polu *Bazy wiedzy* tego okna. Dlatego zmiana tej ścieżki podczas pracy z załadowaną bazą wiedzy może zmienić jej sposób pracy. W polu *„Systemowy"* wyświetlony jest katalog roboczy systemu PC-Shell (jego zawartość nie może być zmieniona przez użytkownika).

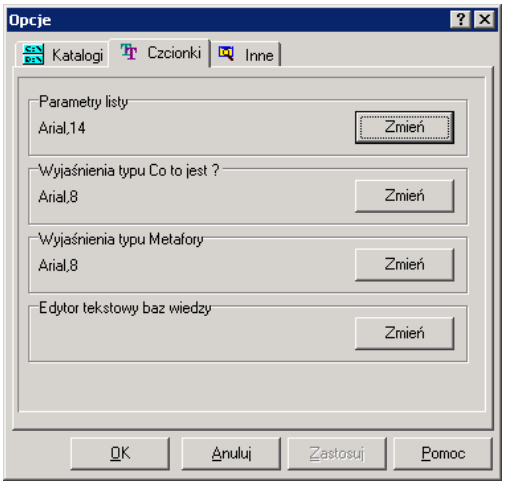



RYS. 3-14. ZAKŁADKA KATALOGI

### Czcionki

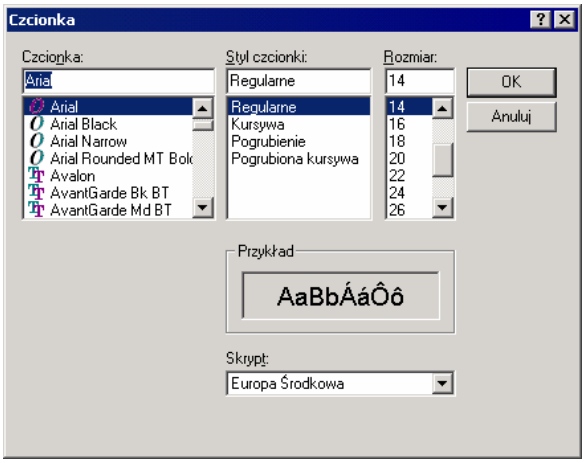
Zakładka umożliwia zdefiniowanie (zmianę) czcionek używanych w oknach systemu:

- oknach przeglądania bazy wiedzy, oknach wyjaśnień tekstowych *Jak ?*,
- wyjaśnieniach typu *Co to jest ?*,
- oknach wyświetlających wyjaśnienia typu Metafora,
- wewnętrznym edytorze baz wiedzy.




RYS. 3-15 ZAKŁADKA CZCIONKI

Po naciśnięciu przycisku *„Zmień”* na ekranie wyświetlone zostanie okno dialogowe zawierające wykaz wszystkich czcionek aktualnie dostępnych w systemie Windows (rys. 3-16)

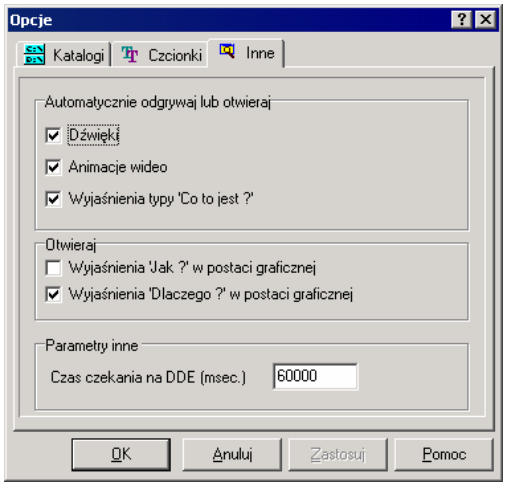



RYS. 3-16. WYBÓR CZCIONKI PODGLĄDU BAZY WIEDZY

Wyjątkiem od tej reguły jest opcja *Zmień* dla *Edytora tekstowego baz wiedzy*. Począwszy od wersji 4.5 pakietu AitechSPHINX system PC-Shell oraz CAKE zostały wyposażone w rozszerzone edytory zawierające m.in. podświetlaną syntaktykę bazy wiedzy. Po wybraniu opcji *Zmień* pojawi się okno definiowania sposobu wyświetlania elementów (CAKE: Rys. 11-10) w oknie edycji bazy wiedzy opisane szczegółowo w rozdziale 11 w podręczniku opisującym system CAKE.

### Inne

Zakładka ta definiuje ustawienia dotyczące pozostałych parametrów systemu PC-Shell.




RYS. 3-17. ZAKŁADKA INNE

Pierwsza grupa opcji dotyczą automatycznego odtwarzania elementów multimedialnych (dźwięk, animacje), a także wyświetlania okna wyjaśnień „co to jest?" w trakcie sesji konsultacji.

Kolejna grupa określa w jaki sposób mają być otwierane wyjaśnienia Jak ? oraz Dlaczego ? – w graficznej czy tekstowej.

Ostatnim elementem konfiguracji jest ustawienie czasu oczekiwania na odpowiedź ze strony serwera DDE. Czas podany jest w milisekundach (zalecana wartość to 60000 ms).

## MENU *POMOC*

Menu zawierające opcje obsługujące pomoc systemową oraz opcję wyświetlającą informacje o danej wersji systemu PC-Shell.

### Skorowidz

Wyświetla spis treści pliku pomocy dotyczącego programu PC-Shell.

### Pomoc inżyniera wiedzy

Wybranie tej opcji powoduje wyświetlenie pliku pomocy inżyniera wiedzy.

### O firmie

Opcja wyświetla okno dialogowe z podstawowymi informacjami o firmie AITECH.

### O programie

Wyświetla okno dialogowe z krótką informacją dotyczącą wersji systemu oraz autorów programu PC-Shell.

---

W wielu oknach dialogowych pojawiają się dwa przyciski symboliczne:

 – dający możliwość zapamiętania raportów systemu na dysku,

 – umożliwiający wydrukowanie raportu, opisującego dany kontekst pracy systemu PC-Shell. I tak, możliwe jest zapamiętanie rozwiązań uzyskanych w wyniku konsultacji, tekstów wyjaśnień „dlaczego?", itp.

## ELEMENTY MULTIMEDIALNE

System PC-Shell posiada elementy systemu multimedialnego. Umożliwia on m.in. powiązanie rysunku, dźwięku bądź sekwencji wideo z konkretnym atrybutem lub wartością. W przypadku, gdy wystąpi zapytanie lub pojawi się rozwiązanie dotyczące atrybutu, do którego przypisano rysunek, na ekranie otworzy się automatycznie okno zawierające rysunek (rys. 3-20 oraz rys. 3-22). Po zamknięciu okna ponowne jego otwarcie możliwe jest po naciśnięciu przycisku  .

Jeżeli danemu atrybutowi przypisano sekwencję wideo, w oknie konsultacji lub rozwiązania pojawi się przycisk  .

Po jego naciśnięciu na ekranie wyświetlone zostanie okno odtwarzacza, umożliwiające obejrzenie animacji.

Podobnie, gdy pojawi się zapytanie lub rozwiązanie związane z atrybutem, do którego jest przypisany efekt dźwiękowy – w oknie konsultacji lub rozwiązania pojawi się przycisk  . Jego naciśnięcie umożliwia odtworzenie przypisanego dźwięku.

Na zakładce *„Inne”* w menu *Opcje* umieszczono polecenia związane z automatycznym odtwarzaniem dźwięków bądź sekwencji wideo.

***Dopuszczalny format plików wideo uzależniony jest od rodzaju zainstalowanych w systemie Windows sterowników. Standardowo w systemach windows 9x oraz NT zainstalowane są sterowniki do odtwarzania plików typu avi.***

## WNIOSKOWANIE

Wcześniejsze wersje systemu PC-Shell udostępniały wszystkie podstawowe metody wnioskowania: wstecz, do przodu oraz wnioskowanie mieszane, będące połączeniem wnioskowania wstecz i do przodu. Obecna wersja udostępnia wyłącznie wnioskowanie wstecz, co nie ogranicza możliwości praktycznych zastosowań. Pozostałe metody wnioskowania będą udostępniane i rozszerzane w przyszłych wersjach systemu.

Wnioskowanie wstecz w systemie PC-Shell, przyjmujące często postać weryfikacji hipotez, przebiega od hipotezy (celu), poprzez reguły do faktów. W praktyce odbywa się to w ten sposób, że system stara się uzgodnić hipotezę z faktem lub regułą (fakty mają pierwszeństwo). Jeśli nie udaje się ich uzgodnić, to system szuka następnego faktu lub reguły i powtarza operację uzgadniania (dokładnie tak jak ma to miejsce w języku *Prolog*). Uzgadnianie polega na porównywaniu ze sobą korespondujących elementów hipotezy (lub warunku reguły) oraz faktu (lub konkluzji reguły). Uzgodnienie obiektów i atrybutów ma miejsce wtedy, gdy są one identyczne. Wartości są ze sobą uzgadniane w jednym z następujących przypadków (zob. przykład 1):

1. wartości są identyczne;
2. jedna z wartości jest reprezentowana przez zmienną i zmienna nie ma przypisanej wartości (ściślej – ma wartość *NIL*);
3. zmienna ma przypisaną wartość i jest ona zgodna z wartością drugiego wyrażenia;
4. obydwa wyrażenia zawierają zmienne i nie mają przypisanych wartości lub przypisane wartości są zgodne.

Uzgadnianie wyrażeń zilustrowano przykładem 1. Wartościami zmiennych mogą być liczby rzeczywiste lub łańcuchy znakowe (ciągi znaków zawarte pomiędzy znakami cudzysłowu). Ta sama zmienna może przyjmować różne typy wartości, nie wymaga się deklarowania typów. Istotną cechą zastosowanej w systemie PC-Shell procedury wnioskowania jest to, że w procesie wnioskowania może być utworzony łańcuch wielu wywoływanych kolejno reguł.

Przykład 1. Uzgadnianie wyrażeń

```
Wyrażenia uzgadnialne:
 1.atrybut1(obiekt1) = "wartość1"
   atrybut1(obiekt1) = "wartość1"
 2.atrybut2(obiekt2) = Zmienna2       (Zmienna2=NIL)
   atrybut2(obiekt2) = 24.56
```

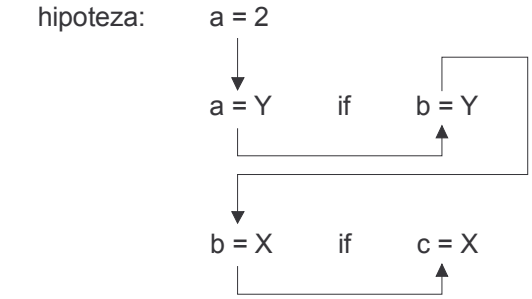
```
Wyrażenia nieuzgadnialne:
 1.atrybut3(obiekt3) = "wartość3"
   atrybut3(obiekt3) = "wartośćA"
```

```
2.atrybut4(obiekt4) = Zmienna4       (Zmienna2="symbol 4")
  atrybut2(obiekt2) = 12.34
```

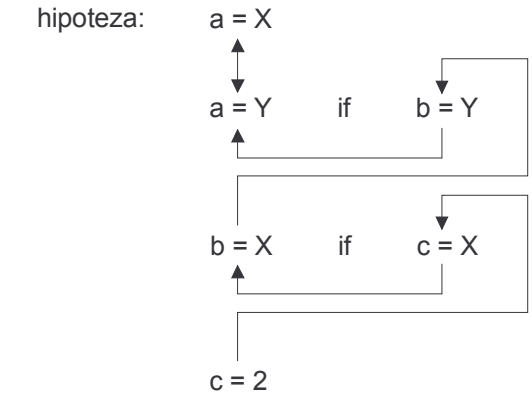
Podczas wywoływania kolejnych reguł, wartości są automatycznie przekazywane w górę lub w dół powiązanych procesem wnioskowania reguł, zależnie od kierunku, z którego te wartości pochodzą (zilustrowano to na przykładzie 2). Należy tu podkreślić, że identyfikatory zmiennych mają charakter lokalny dla danej reguły, oznacza to w praktyce, że zmienna *X* w pewnej regule jest zupełnie inną zmienną niż zmienna *X* w drugiej regule.

Przykład 2. Przekazywanie wartości przez zmienne w procesie wnioskowania

**„od góry”**




**„od do³u”**




System PC-Shell jest wyposażony w mechanizm nawrotów (ang. *backtracking*), działający tak samo jak w języku Prolog. Umożliwia on m.in. odnajdywanie wszystkich dostępnych rozwiązań danego problemu. Ważnym efektem ubocznym nawrotów jest „unieważnienie” wartości zmiennych, które zostały przypisane w punkcie nawrotu i później. Punkt nawrotu stanowi uzgodniony z danym warunkiem reguły lub hipotezą fakt lub reguła, jeśli istnieją inne uzgadnialne fakty lub reguły (zob. przykład 3). O ile reguła 2 w podanym przykładzie zostanie uzgodniona i potwierdzone będą wszystkie warunki tej reguły, to system wygeneruje następujące rozwiązania:

```
X = 1
X = 2
```

Wartość zmiennej *Y*, w warunkach *d=Y* i *e=Y*, będzie się zmieniała, natomiast wartość zmiennej *Z* pozostanie w tym czasie niezmieniona. Wartość zmiennej *Z* pozostałaby niezmieniona w tym czasie nawet, gdyby w bazie wiedzy istniał jeszcze jeden fakt np. *c=2*. Zmieniłaby się ona dopiero po ewentualnym nawrocie związanym z faktem *c=3*.

Przykład 3. Sytuacja uruchamiająca mechanizm nawrotów (ang. *backtracking*)

```
Hipoteza:  a = X
Reguły w bazie wiedzy:
 1.a = X if b = X;
 2.b = Y if c = Z, d = Y, e = Y;
Zbiór faktów:
 1.d = 1;
 2.d = 2;
 3.c = 3;
```

## PRZYPADKI SZCZEGÓŁOWE

W tej części rozdziału będą przeanalizowane konkretne przypadki dotyczące konsultacji w systemie PC-Shell.

Przypadek 1

```
Hipoteza:           atr = X
Zawartość bazy wiedzy:    fakt: atr = 2;
Dialog:             brak
Mechanizm nawrotów: nie uruchomiony
Rozwiązanie:        X = 2
```

Przypadek 2

```
Hipoteza:           atr = X
Zawartość bazy wiedzy:    fakt: atr = 1;
                    fakt: atr = 2;
Dialog:             brak
Mechanizm nawrotów: uruchomiony
Rozwiązanie:        2 rozwiązania: X = 1, X = 2
```

Przypadek 3

```
Hipoteza:           atr1 = X
Zawartość bazy wiedzy:    reguła: atr1 = "alfa"  if  atr2 = "beta"
Dialog:             pytanie: Czy atr2 = "beta" ?
Mechanizm nawrotów: nie uruchomiony
Rozwiązanie:        X = "alfa"
```

Przypadek 4

```
Hipoteza:           atr1 = X
Zawartość bazy wiedzy:    reguła: atr1 = X  if  atr2 = X
                    fakt: atr2 = "alfa"
Dialog:             brak
Mechanizm nawrotów: nie uruchomiony
Rozwiązanie:        X = "alfa"
```

Przypadek 5

```
Hipoteza:           atr1 = X
Zawartość bazy wiedzy:    reguła: atr1 = X  if  atr2 = X
                    fakt: atr2 = "alfa"
                    fakt: atr2 = "beta"
Dialog:             brak
Mechanizm nawrotów: uruchomiony
Rozwiązanie:        2 rozwiązania: X = "alfa", X = "beta"
```

Przypadek 6

```
Hipoteza:           atr = "alfa"
Zawartość bazy wiedzy:    fakt: atr = "alfa"
Dialog:             brak
Mechanizm nawrotów: nie uruchomiony
Rozwiązanie:        hipoteza potwierdzona
```

Przypadek 7

```
Hipoteza:           atr1 = "alfa"
Zawartość bazy wiedzy:    reguła: atr1 = X  if  atr2 = X
                    fakt: atr2 = "alfa"
Dialog:             brak
Mechanizm nawrotów: nie uruchomiony
Rozwiązanie:        hipoteza potwierdzona
```

Przypadek 8

```
Hipoteza:           atr1 = "alfa"
Zawartość bazy wiedzy:    reguła: atr1 = X  if  atr2 = X
Dialog:             pytanie: Czy atr2 = "alfa" ?
Mechanizm nawrotów: nie uruchomiony
Rozwiązanie:        hipoteza potwierdzona
```

Przypadek 9

```
Hipoteza:           atr1 = "alfa"
Zawartość bazy wiedzy:    reguły:      atr1 = X  if  atr2 = X
                          atr2 = X  if  atr3 = X
                          atr3 = X  if  atr4 = X
                          atr4 = X  if  atr5 = X
Dialog:             pytanie: Czy atr5 = "alfa" ?
Mechanizm nawrotów: nie uruchomiony
Rozwiązanie:        hipoteza potwierdzona
```

Przypadek 10

```
Hipoteza:           atr1 = X
Zawartość bazy wiedzy:    reguły:      atr1 = X  if  atr2 = X
                          atr2 = X  if  atr3 = X
                          atr3 = X  if  atr4 = X
                          atr4 = X  if  atr5 = X
                    fakt: atr5 = 123.45
Dialog:             brak
Mechanizm nawrotów: nie uruchomiony
Rozwiązanie:        X = 123.45
```

Przypadek 11

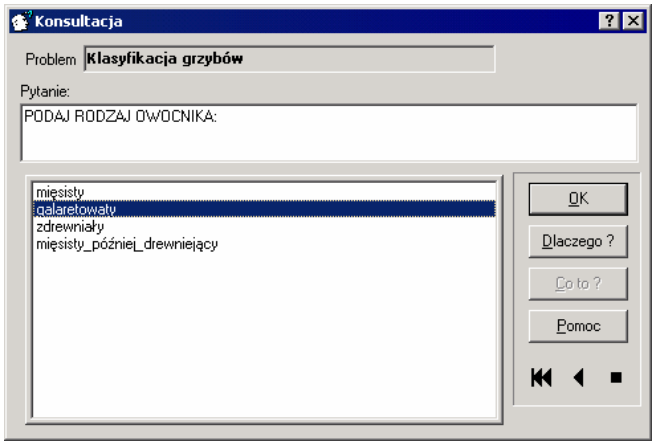
```
Hipoteza:           atr1 = X
Zawartość bazy wiedzy:    reguły:      atr1 = X  if  atr2 = X
                          atr2 = X  if  atr3 = X
                          atr3 = X  if  atr4 = X
                          atr4 = X  if  atr5 = X
Dialog:             pytanie: Dla jakiej wartości X, atr5 = X ?
                    odpowiedź: "alfa"
Mechanizm nawrotów: nie uruchomiony
Rozwiązanie:        X = "alfa"
```

Przypadek 12

```
Hipoteza:           atr1 = X
Zawartość bazy wiedzy:    reguła: atr1 = X  if  atr2 = Y & atr3 = X;
                    fakt: atr2 = 1
                    fakt: atr2 = 2
                    fakt: atr3 = "alfa"
Dialog:             brak
Mechanizm nawrotów: uruchomiony
Rozwiązanie:        2 rozwiązania: X = "alfa", X = "alfa"
```

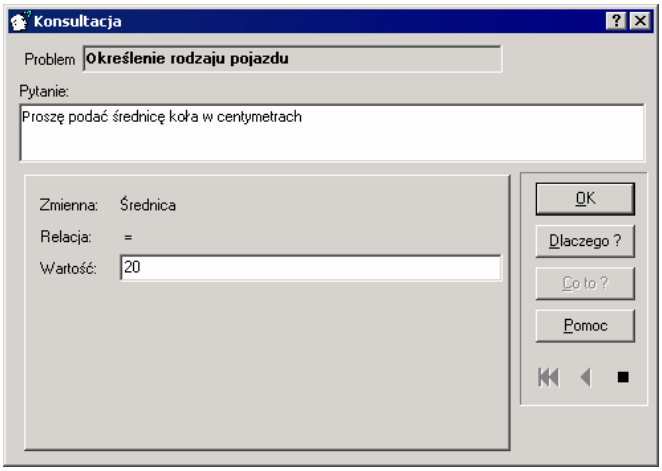
## OKNO KONSULTACJA

Bardzo często rezultatem uzgadniania warunków w procesie wnioskowania jest generowanie przez system zapytań do użytkownika konsultacji. Wyróżnia się dwa rodzaje okien konsultacji. Jedno dające użytkownikowi możliwość odpowiedzi poprzez jej wybór z listy możliwych odpowiedzi (rys. 3-18) lub poprzez możliwość bezpośredniego wprowadzenia wartości, w przypadku wystąpienia w warunku zmiennej bez przypisanej wartości (rys. 3-19). Podczas wpisywania wartości będącej odpowiedzią użytkownika, system sprawdza poprawność wprowadzanych informacji. Wartości typu łańcuch znaków muszą być wprowadzane w postaci dowolnego ciągu znaków ujętych w cudzysłowy, natomiast liczby są rozpoznawane automatycznie zgodnie z syntaktyką liczb rzeczywistych.

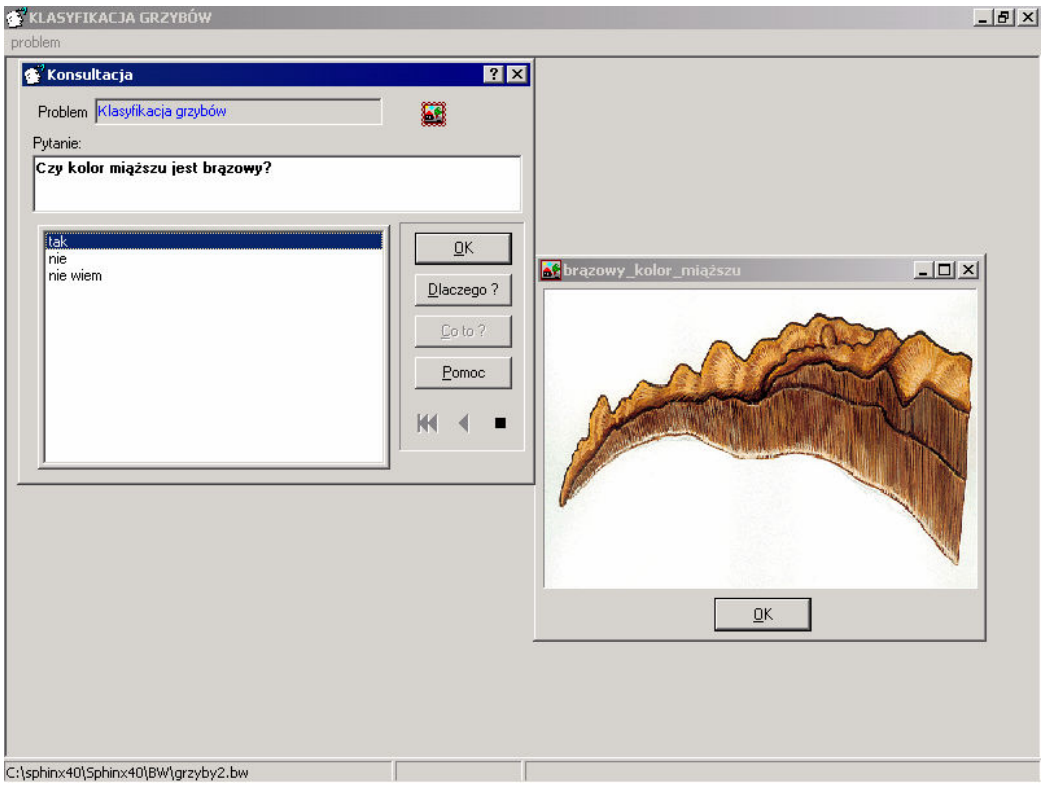



RYS. 3-18. OKNO KONSULTACJI – WYBÓR Z LISTY MOŻLIWYCH ODPOWIEDZI

Okno konsultacji, poza informacjami tekstowymi (zapytaniami systemu), udostępnia zbiór przycisków z funkcjami systemu, użytecznymi podczas dialogu z systemem ekspertowym. Wśród nich są dwie funkcje wyjaśnień typu „dlaczego?" oraz „co to jest?". Opis wyjaśnień znajduje się w następnym rozdziale. Okno konsulatacji zawiera panel z trzema przyciskami nawigacyjnymi służącymi kolejno do wznowienia procesu wnioskowania od początku, cofnięcia wnioskowania o jeden krok wstecz oraz ostatni przycisk przerywający proces wnioskowania.



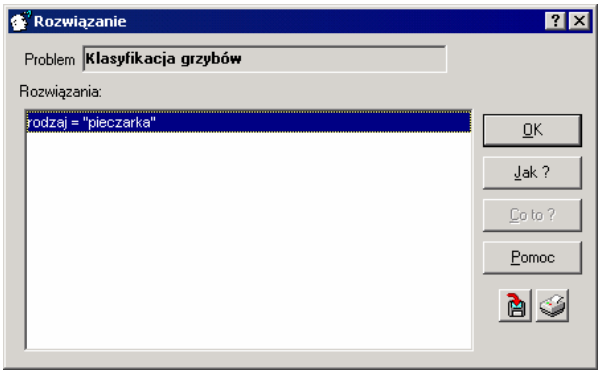

RYS. 3-19. OKNO KONSULTACJI – WPROWADZANIE WARTOŚCI ZMIENNEJ




RYS. 3-20. OKNO KONSULTACJI – ZAPYTANIE WRAZ Z POWIĄZANYM Z NIM RYSUNKIEM

## OKNO ROZWIĄZANIE

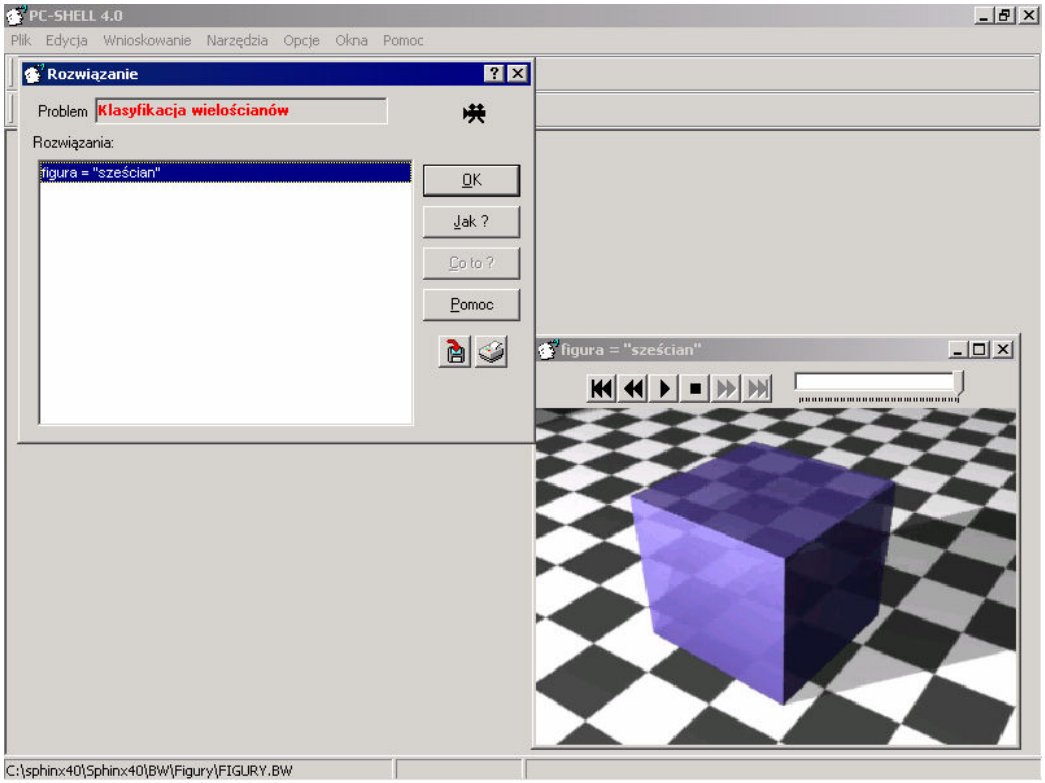
Ostatecznym rezultatem procesu wnioskowania jest zbiór rozwiązań, który jest wyświetlany w postaci przedstawionej na rys. 3-21. W przypadku niepotwierdzenia hipotezy, zbiór rozwiązań jest zbiorem pustym, lista rozwiązań zawiera wtedy tekst *hipoteza nie potwierdzona* (możliwa jest zmiana tego tekstu za pomocą instrukcji **setSysText**)




RYS. 3-21. OKNO ROZWIĄZAŃ

W oknie rozwiązań użytkownik ma do dyspozycji trzy rodzaje wyjaśnień dotyczących otrzymanych rozwiązań: „co to jest?”, „jak?” oraz *metafora*. Poza tym, dostępne są standardowe opcje takie jak *Zapis* oraz *Wydruk*.

Inżynier wiedzy budując aplikację (bazę wiedzy) może, jak wspominaliśmy, dodać elementy multimedialne, jako pewnego rodzaju dodatkowe wyjaśnienia obrazujące rozwiązywany problem. Na rys. 3-22 przedstawiono przykładową bazę wiedzy z dołączoną animacją wideo.




RYS. 3-22. OKNO ROZWIĄZAŃ WRAZ Z POWIĄZANYM ANIMACJĄ

## MECHANIZMY WYJAŚNIEŃ

Podczas budowy i rozwoju systemu PC-Shell szczególna uwaga przywiązywana jest do mechanizmu wyjaśnień. Wynika to z przekonania o zasadniczym znaczeniu wyjaśnień dla specyfiki i utylitarnych walorów technologii systemów ekspertowych. Znajduje to potwierdzenie w literaturze specjalistycznej, jak również w osobistych doświadczeniach autora. Można zaryzykować twierdzenie, że system, który nie dostarcza przynajmniej wyjaśnień konkluzji nie zasługuje na miano systemu ekspertowego.

Jak już wspomniano, system PC-Shell dostarcza czterech rodzajów wyjaśnień: konkluzji („jak?"), zapytań („dlaczego?"), pojęć („co to jest?") oraz *metafor*. W przyszłości przewiduje się dodanie wyjaśnień typu „dlaczego nie?".

## WYJAŚNIENIA TYPU „JAK ?"

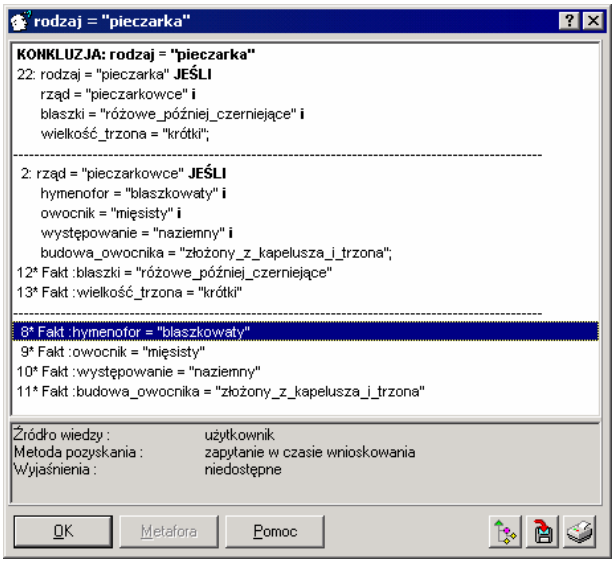
Wyjaśnienia te są dostępne w oknie rozwiązań po zakończeniu procesu wnioskowania (rys. 3-21). Począwszy od wersji 2.3 dostępne są dwa rodzaje wyjaśnień *„Jak ?"*: tekstowe (rys. 3-23) oraz graficzne (rys 3-24)

Wyjaśnienia *„Jak ?"* służą udokumentowaniu i przedstawieniu użytkownikowi, w jaki sposób system wyprowadził dany zbiór konkluzji (rozwiązań). Wyjaśnienia mają w tym wypadku charakter retrospektywny. W odniesieniu do wybranych reguł możliwe są pogłębione wyjaśnienia tekstowe (*metafory*), o ile inżynier wiedzy przygotował je za pomocą systemu CAKE. Wyświetlenie *metafory* nastąpi po dwukrotnym naciśnięciu na regule lewego przycisku myszy lub po wybraniu reguły i naciśnięciu przycisku *Metafora*.

## WYJAŚNIENIA „JAK ?” - REPREZENTACJA TEKSTOWA

Okno wyjaśnień jak w postaci tekstowej wyświetla w postaci listy kolejne poziomy wyjaśnień oddzielone od siebie poziomą linią.

Wyróżnienie faktu na liście powoduje wyświetlenie w dolnej części okna dodatkowych wyjaśnienień na temat sposobu i źródła pozyskania faktu. Jeżeli dany fakt powstał w wyniku wnioskowania ze źródła wiedzy (użycie architektury tablicowej), wtedy dostępne są, po podwójnym kliknięciu lewym przyciskiem myszki, dalsze wyjaśnienia typu „jak?”. Otwiera się wtedy kolejne okno wyjaśnień „jak?” dla wybranego faktu. W prawym dolnym rogu znajduje się przycisk  włączający wyjaśnienia graficzne.




RYS. 3-23 WYJAŚNIENIA „JAK?” - REPREZENTACJA TEKSTOWA

## WYJAŚNIENIA „JAK ?” - REPREZENTACJA GRAFICZNA

Nowym sposobem wyświetlania wyjaśnień jest reprezentacja graficzna w postaci drzewa wnioskowania, w którym rolę węzłów pełnią reguły oraz fakty uzgodnione przez system w procesie wnioskowania. Możliwe jest wyświetlanie wyjaśnień w dwóch trybach:

- trybie ogólnym,
- trybie szczegółowym.

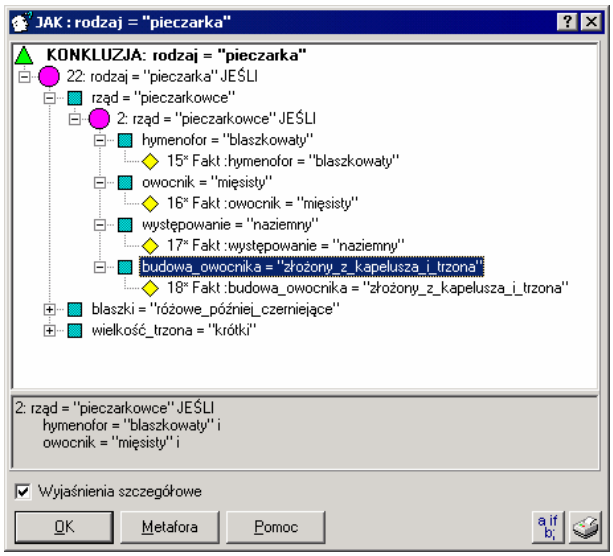
W trybie ogólnym przedstawiane są jedynie reguły oraz fakty. Natomiast w trybie szczegółowym w strukturze drzewa uwzględnione są całe reguły wraz z faktami.

Poniżej drzewa wyjaśnień znajduje się pole tekstowe, w którym wyświetlane są dodatkowe dane dotyczące aktualnie zaznaczonego elementu drzewa. Gdy jest to reguła, wtedy wyświetlana jest jej pełna postać tekstowa, gdy jest to fakt, wyświetlana jest wtedy informacja o sposobie pozyskania faktu.

Przyjęto następujące oznaczenia poszczególnych elementów drzewa wyjaśnień:

- rozwiązanie (konkluzja) wygenerowane przez system,
- reguła zdefiniowana w bazie wiedzy
- warunek wchodzący w skład reguły
- fakt zawarty w bazie wiedzy lub utworzony w wyniku zapytania

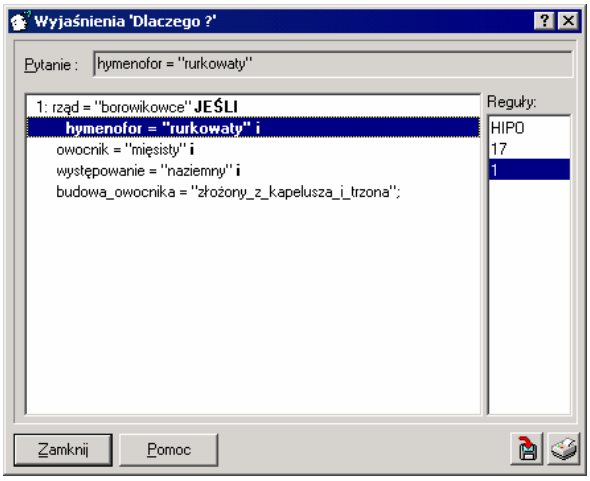
W prawym dolnym rogu jest umieszczony przycisk  dzięki któremu można przełączyć się na reprezentację tekstową wyjaśnień „Jak ?”



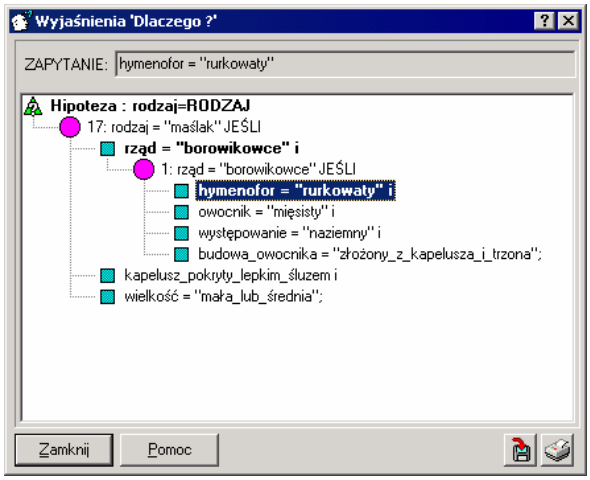

RYS. 3-24. WYJAŚNIENIA „JAK?” - REPREZENTACJA GRAFICZNA

## WYJAŚNIENIA TYPU „DLACZEGO?”

Okno wyjaśnień „dlaczego?” udostępnia wyjaśnienia uzasadniające celowość zapytania skierowanego przez system do użytkownika. Są one generowane automatycznie przez system PC-Shell na podstawie kontekstu bieżącego wnioskowania, w związku z czym nie muszą być wcześniej przygotowywane przez inżyniera wiedzy. System w zależności od ustawień w opcjach wyświetla wyjaśnienia w postaci tekstowej (rys. 3-25) lub w postaci graficznej (rys. 3-26). Sposób wyświetlania ustalamy w menu *Opcje* na zakładce *Inne*.



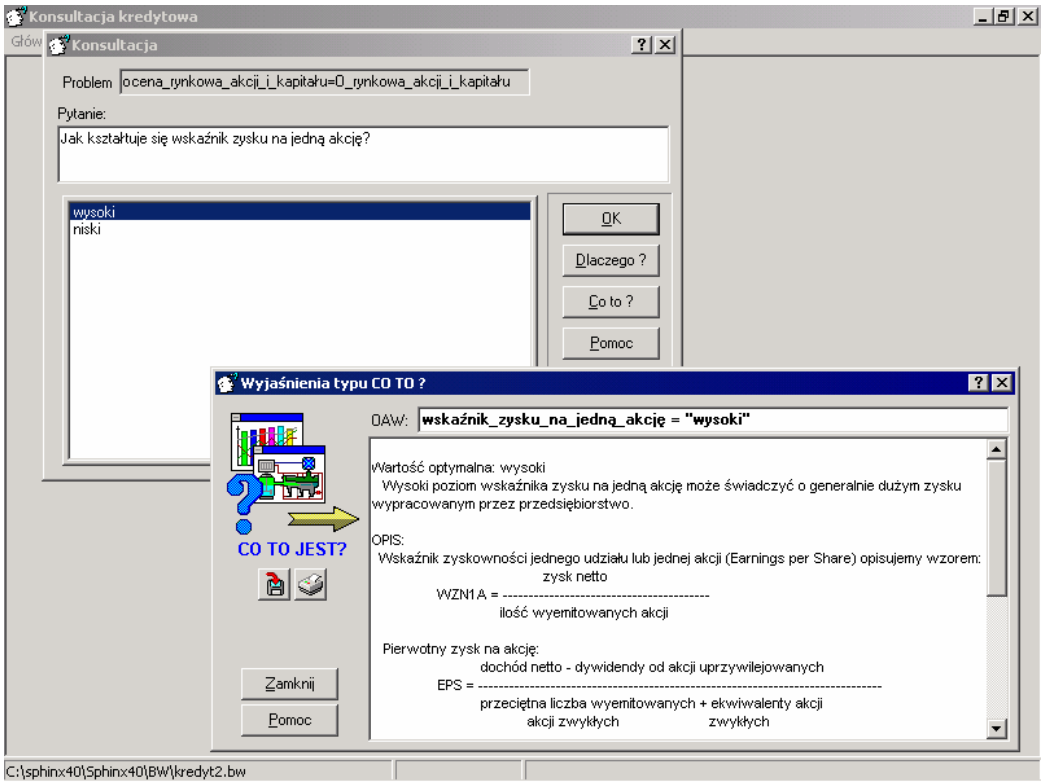

RYS. 3-25. WYJAŚNIENIA „DLACZEGO?” – REPREZENTACJA TEKSTOWA




RYS. 3-26. WYJAŚNIENIA „DLACZEGO ?" – REPREZENTACJA GRAFICZNA

## WYJAŚNIENIA TYPU „CO TO JEST ?"

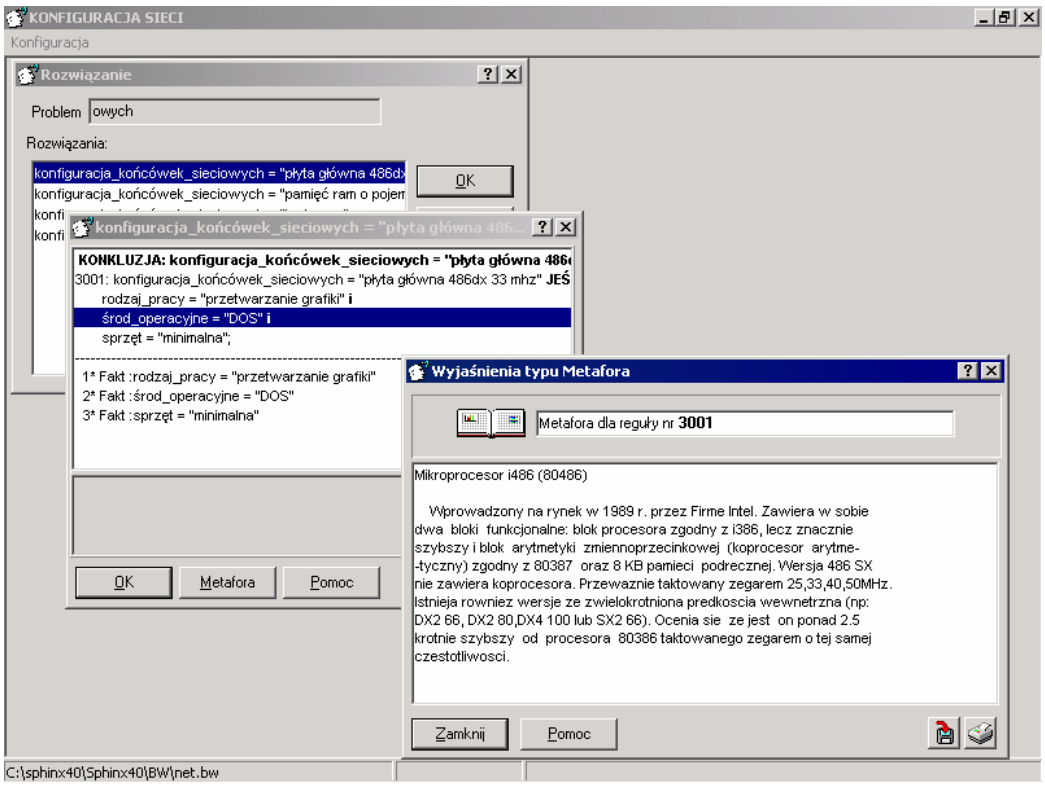
Są to tekstowe wyjaśnienia przygotowywane przez inżyniera wiedzy za pomocą systemu CAKE, objaśniające wybrane pojęcia z bazy wiedzy, użyte w konkluzji lub zapytaniu systemu. Dla przykładu, w systemie diagnostycznym, który ustalił konkretną usterkę, wyjaśnienia te mogą, poza omówieniem samej usterki, opisać sposób jej usunięcia.




RYS. 3-27. WYJAŚNIENIA „CO TO JEST ?"

## WYJAŚNIENIENIA TYPU *METAFORA*

*Metafory* są również wyjaśnieniami tekstowymi przygotowanymi przez inżyniera wiedzy. Dostarczają one użytkownikowi wyjaśnień systemu odnośnie konkretnej reguły. Postać tych wyjaśnień jest całkowicie zależna od inżyniera wiedzy. Powinny one zawierać omówienie strony merytorycznej reguły, będąc jakby nieformalną, tekstową „metaforą" reguły logicznej. Wyjaśnienia te mogą na przykład zawierać odnośniki do literatury fachowej, dotyczącej rozważanego w regule przypadku.




RYS. 3-28. WYJAŚNIENIA TYPU „METAFORA"

## LITERATURA:

[1] Feigenbaum E.A., Feldman J. (Eds.): Computers and thought. McGraw-Hill, New York 1963.

[2] Jackson P.: Introduction to expert systems. Addison-Wesley, Reading, Massachusetts 1986.

[3] Shannon C.E.: Automatic chess player. Scientific American, 182, 1950.