Katedra Sztucznej Inteligencji i Matematyki Stosowanej
WI ZUT Szczecin

Sztuczna inteligencja i maszynowe uczenie w systemach interaktywnych

Joanna Kolodziejczyk
jkolodziejczyk@zut.edu.pl

October 27, 2021

Plan wykładu

Knowledge based systems

Systemy ekspertowe

CLIPS

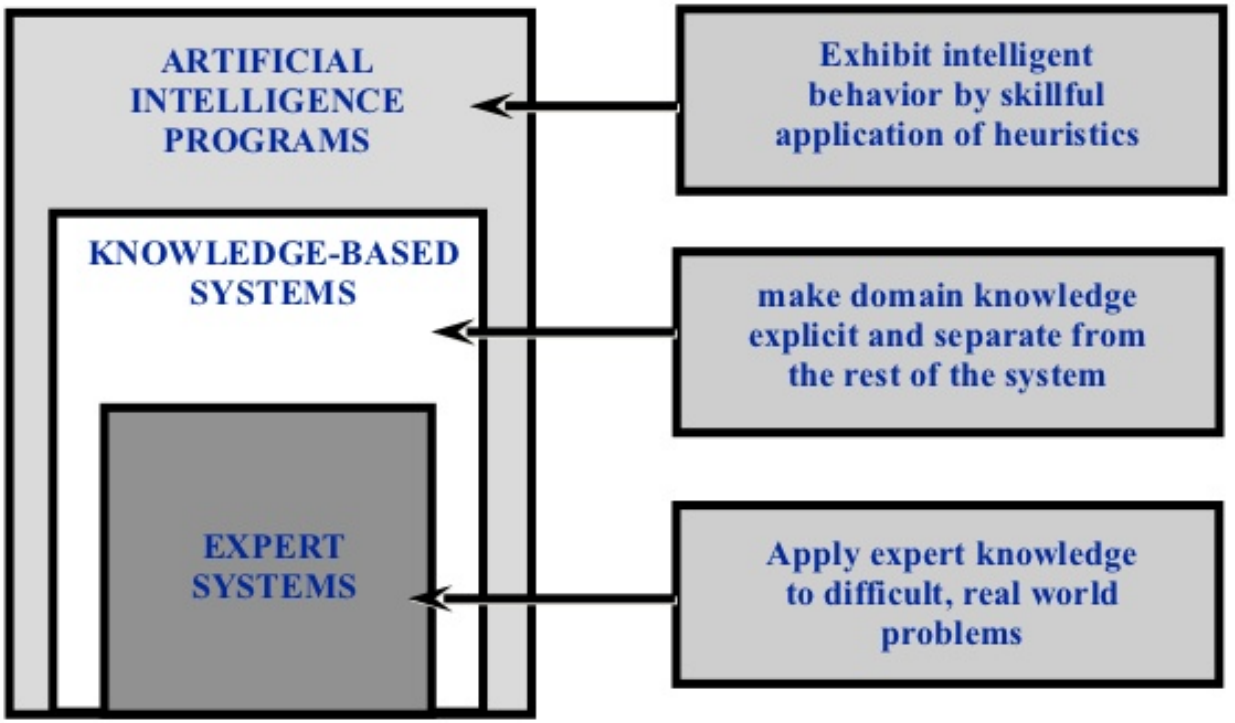
- Elementy języka

System oparty na wiedzy

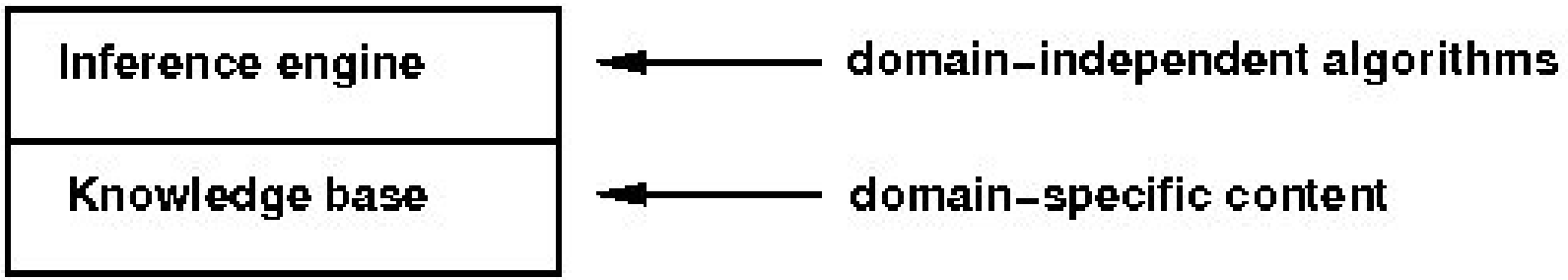
System oparty na wiedzy (KBS) to program komputerowy, który wnioskuje i wykorzystuje bazę wiedzy do rozwiązywania złożonych problemów. Termin ten jest szeroki i odnosi się do wielu różnych rodzajów systemów.
Wspólnym mianownikiem, który łączy systemy oparte na wiedzy, jest próba jednoznacznego przedstawienia wiedzy oraz system wnioskowania, który pozwala na uzyskiwanie nowej wiedzy. Tak więc system oparty na wiedzy ma dwie wyróżniające go elementy:

- Bazę wiedzy
- Silnik wnioskowania

Zależność AI KBS i ES
Schemat



Ogólny schemat KBS
Schemat



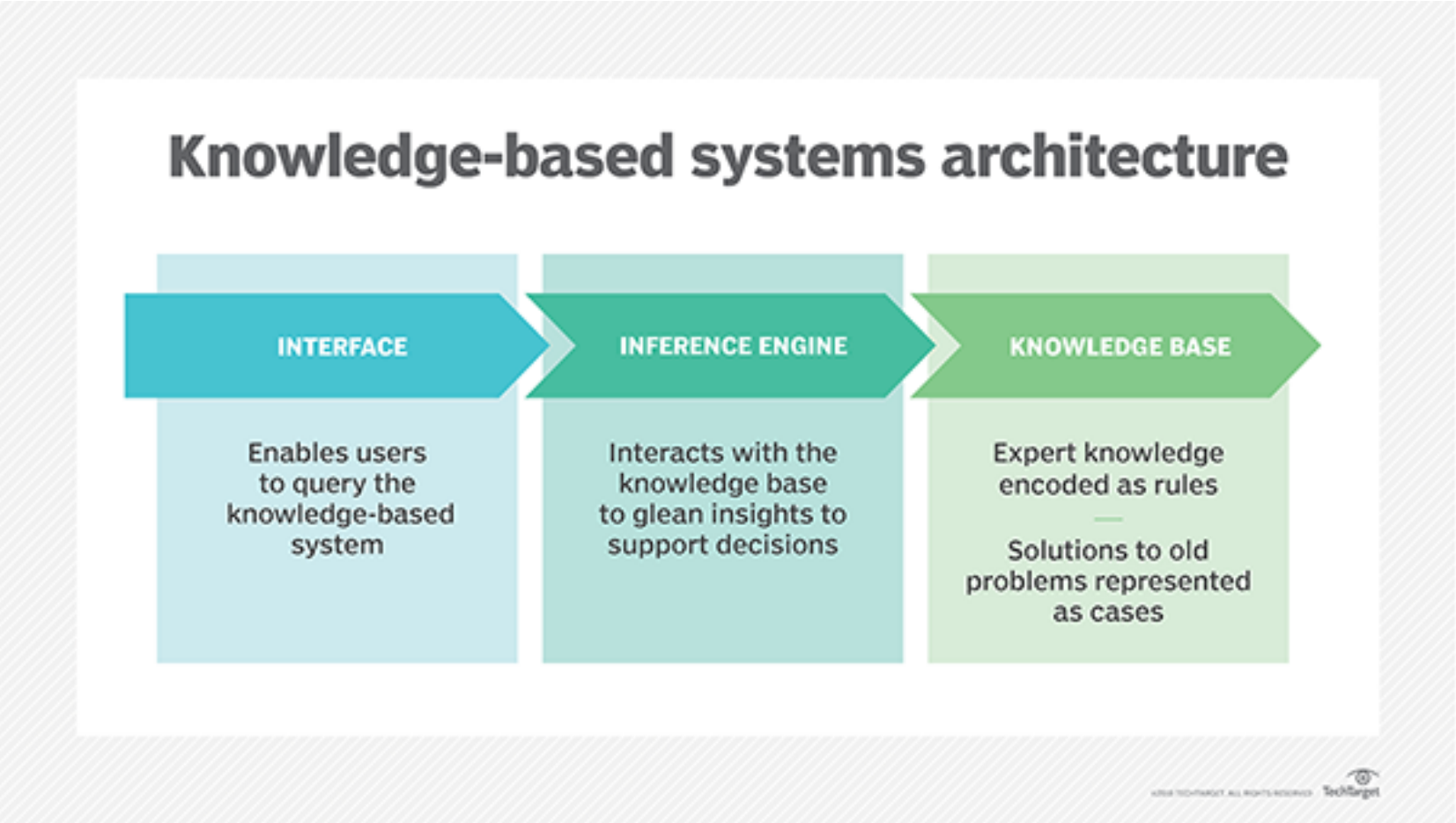
Baza wiedzy (KB)

Zbiór zdań w języku formalnym lub inaczej w języku reprezentacji wiedzy.

Silnik wnioskujący

Procedura dokonująca manipulacji na KB i generująca weryfikację hipotez.

KBS architektura
Schemat



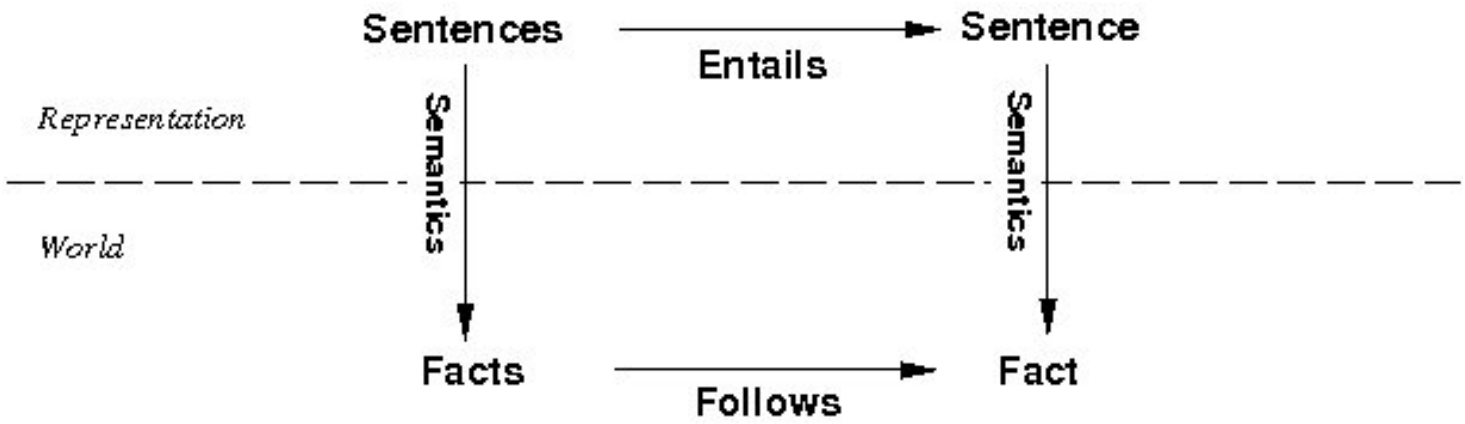
Baza wiedzy

Baza wiedzy

Reprezentuje fakty dotyczące świata, często w jakiejś formie ontologii (a nie osadzonej w kodzie proceduralnym, tak jak robi to konwencjonalnym programie komputerowym).
Do form reprezentacji można zaliczyć:

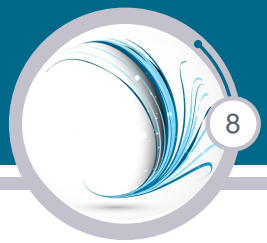
- postać regułowa (IF - THEN)
- drzewa decyzyjne
- tablice decyzyjne
- sieci semantyczne (ontologie)

Zależność pomiędzy reprezentacją i rzeczywistością



Representation — formalna reprezentacja wiedzy
World — rzeczywistość
Semnatics — znaczenie
Sentences — zdania reprezentujące wiedzę w jakimś języku formalnym
Facts — rzeczywiste zdarzenia, fakty
Follows — następstwo
Entails — konsekwencja

Silnik wnioskujący



Silnik wnioskujący

Silnik wnioskowania, pozwala wnioskować o nowej wiedzy. Najczęściej przybiera formę reguł IF-THEN w połączeniu z podejściami wnioskowania do przodu lub do tyłu. Inne podejścia obejmują:

- automatyczne dowodzenie - automated theorem provers
- programowanie logiczne
- systemy takie jak CHR (Constraint Handling Rules)

Sposoby reprezentacji wiedzy (Baza wiedzy)

Bazą wiedzy

może być zbiór definicji, faktów, pojęć i relacji między nimi.

Tworzenie baz wiedzy

- automatycznie (np. z analizy danych)
- od człowieka - eksperta w dziedzinie na zasadzie interakcji.

Sposoby reprezentacji:

- reguły (if (PRZESŁANKA) then (KONKLUZJA), przy czym PRZESŁANKA jest wyrażeniem, które zbiera w sobie jeden lub więcej warunków).
- notacja obiektowa
- ramy
- sieci semantyczne

Typy wnioskowania (Układ wnioskujący)

W przód (Forward chaining), wnioskowanie z danych (przesłanek)

Metoda ta zaczyna od zestawu znanych faktów i wartości atrybutów i stosuje je w regułach, które zawierają takie dane w przesłance. Wszystkie odpalone reguły (z prawdziwą przesłanką), tworzą dodatkowe fakty, które są stosowane do odpalenia kolejnych reguł. Proces trwa do czasu, gdy żadne nowe fakty nie są produkowane, lub gdy uzyskana jest wartość celu. Sprawdza się, gdy przed rozpoczęciem wnioskowania i tak gromadzi się zbiór faktów.

Wstecz (Backward chaining), wnioskowanie z konkluzji

Proces zaczyna się od celu dowiedzenia (hipotezy) i próbuje uzyskać wartości wszystkich przesłanek i atrybutów użytych w części IF reguły, a następnie wstecznie sprawdzać dodatkowe reguły (o ile to konieczne), by otrzymać wartości atrybutów dla uzyskania celu. Metoda lepsza dla baz z regułami o rozbudowanych przesłankach w wieloma regułami, gdyż nie pyta użytkownika o wszystkie wartości atrybutów.

Wnioskowanie w przód

KNOWLEDGE BASE
(Initial State)

Fact :
F1 - A lathe is a machine tool

Rules :
R1 - If X is power driven Then X requires a power source
R2 - If X is a machine tool Then X has a tool holder
R3 - If X is a machine tool Then X is power driven

KNOWLEDGE BASE
(Intermediate State)

Fact :
F1 - A lathe is a machine tool
F2 - A lathe has a tool holder

Rules :
R1 - If X is power driven Then X requires a power source
R2 - If X is a machine tool Then X has a tool holder
R3 - If X is a machine tool Then X is power driven

Wnioskowanie w przód

KNOWLEDGE BASE
(Intermediate State)

Fact :
- **F1** - A lathe is a machine tool
- **F2** - A lathe has a tool holder
- **F3** - A lathe is power driven

Rules :
- **R1** - If X is power driven Then X requires a power source
- **R2** - If X is a machine tool Then X has a tool holder
- **R3** - If X is a machine tool Then X is power driven

KNOWLEDGE BASE
(Final State)

Fact :
- **F1** - A lathe is a machine tool
- **F2** - A lathe has a tool holder
- **F3** - A lathe is power driven
- **F4** - A lathe requires a power source

Rules :
- **R1** - If X is power driven Then X requires a power source
- **R2** - If X is a machine tool Then X has a tool holder
- **R3** - If X is a machine tool Then X is power driven

Wnioskowanie wstecz

KNOWLEDGE BASE
(Initial State)

Fact :
F1 -A lathe is a machine tool

Rules :
R1 - If X is power driven Then X requires a power source
R2 - If X is a machine tool Then X has a tool holder
R3 - If X is a machine tool Then X is power driven

GOAL STACK

Goal :	Satisfied
G1 - A lathe requires a power source	?

KNOWLEDGE BASE
(Intermediate State)

Fact :
F1 -A lathe is a machine tool

Rules :
R1 - If X is power driven Then X requires a power source
R2 - If X is a machine tool Then X has a tool holder
R3 - If X is a machine tool Then X is power driven

GOAL STACK

Goal :	Satisfied
G1 - A lathe requires a power source	?
G2 - A lathe is a power driven	?

Wnioskowanie wstecz

KNOWLEDGE BASE
(Intermediate State)

Fact :
F1 -A lathe is a machine tool

Rules :
R1 - If X is power driven Then X requires a power source
R2 - If X is a machine tool Then X has a tool holder
R3 - If X is a machine tool Then X is power driven

GOAL STACK

Goal :	Satisfied
G1 - A lathe requires a power source	?
G2 - A lathe is a power driven	?
G3 - A lathe is a machine tool	?

KNOWLEDGE BASE
(Intermediate State)

Fact :
F1 -A lathe is a machine tool

Rules :
R1 - If X is power driven Then X requires a power source
R2 - If X is a machine tool Then X has a tool holder
R3 - If X is a machine tool Then X is power driven

GOAL STACK

Goal :	Satisfied
G1 - A lathe requires a power source	?
G2 - A lathe is a power driven	?
G3 - A lathe is a machine tool	Yes

Wnioskowanie wstecz

KNOWLEDGE BASE
(Intermediate State)

Fact :
F1 -A lathe is a machine tool
F2 -A lathe is power driven

Rules :
R1 - If X is power driven Then X requires a power source
R2 - If X is a machine tool Then X has a tool holder
R3 - If X is a machine tool Then X is power driven

GOAL STACK

Goal :	Satisfied
G1 - A lathe requires a power source	?
G2 - A lathe is a power driven	Yes

KNOWLEDGE BASE
(Final State)

Fact :
F1 -A lathe is a machine tool
F2 -A lathe is power driven
F3 -A lathe requires a power source

Rules :
R1 - If X is power driven Then X requires a power source
R2 - If X is a machine tool Then X has a tool holder
R3 - If X is a machine tool Then X is power driven

GOAL STACK

Goal :	Satisfied
G1 - A lathe requires a power source	Yes

Niepewność w systemie

Wnioski nie zawsze są konstruowane ze 100% pewnością, czyli nie zawsze są binarne. Dodatkowo określa się pewną liczbę do przesłanki/konkluzji (najczęściej rzeczywistą) która mówi, o tym jakie jest przekonanie, że zdanie/fakt jest prawdziwe.
Metody wyrażania niepewności:

- certainty factors (współczynniki pewności)
- probability (prawdopodobieństwo)
- fuzzy logic (logika rozmyta)

Zastosowania KBS

- Opieka zdrowotna jest ważnym rynkiem dla KBS, które są określane jako systemy wspomagania decyzji klinicznych w kontekście nauk medycznych
  `http://symptoms.webmd.com/default.htm`;
- Analiza ścieżek lawinowych;
- Diagnostyka usterek urządzeń przemysłowych i innych
  `https://support.microsoft.com/en-us/contactus/`;
- Zarządzanie środkami finansowymi.

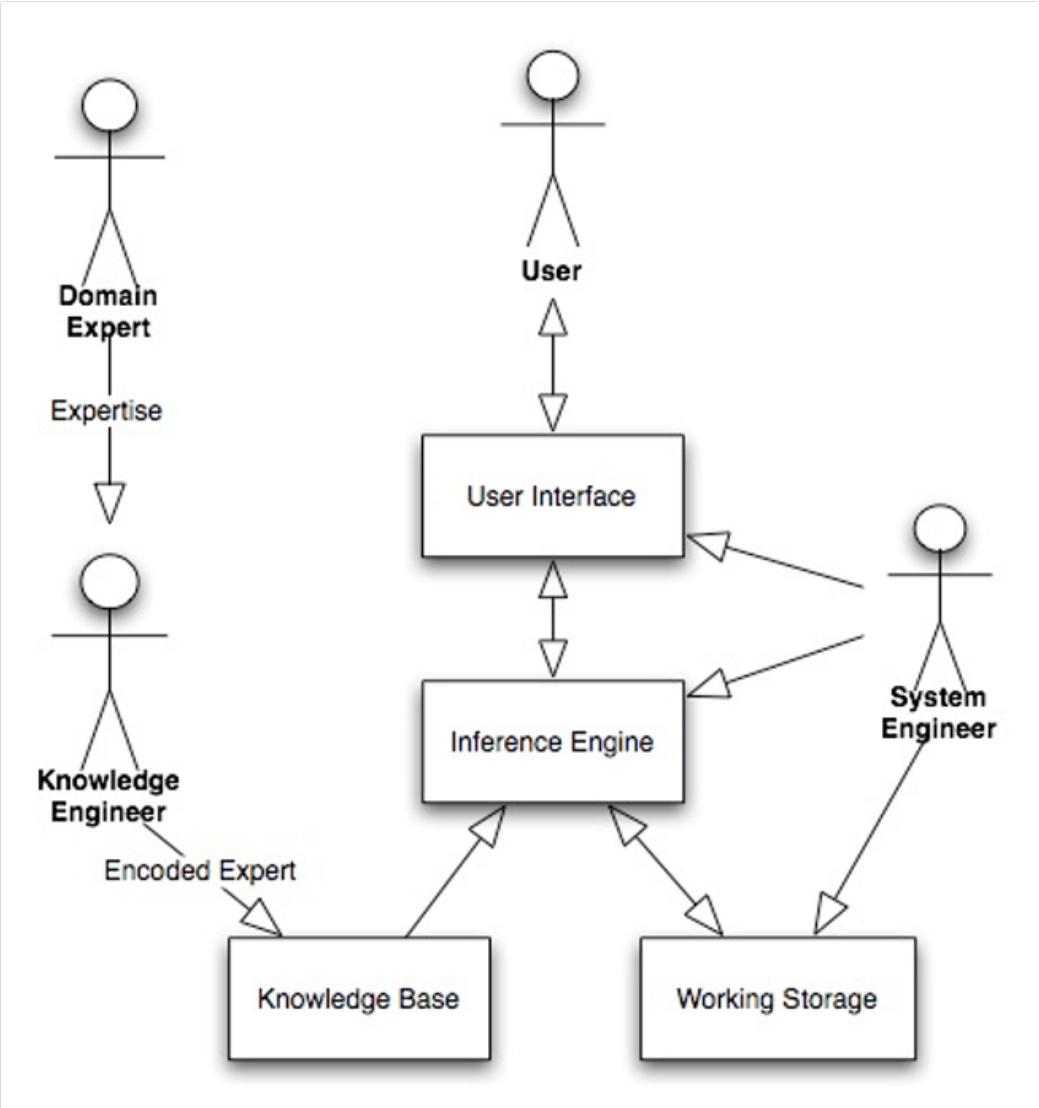
Systemy ekspertowe są oparte na wiedzy

Definicja

System ekspercki to system komputerowy, który naśladuje zdolności decyzyjne ludzkiego eksperta. Systemy eksperckie są przeznaczone do rozwiązywania złożonych problemów poprzez rozumowanie za pomocą baz wiedzy, reprezentowanych głównie jako reguły, a nie poprzez konwencjonalny kod proceduralny.
Cechy SE

- Wykorzystanie ludzkiej wiedzy do rozwiązywania problemów, które normalnie wymagałyby ludzkiej inteligencji.
- Zawiera pewną niealgorytmiczną wiedzę specjalistyczną.
- Reprezentowanie wiedzy specjalistycznej jako KB.

Elementy i aktorzy w SE



Interfejs użytkownika — dialog

Użytkownik w SE prowadzi dialog. Dialog ten ma następujące cechy:

- Nie trzeba dawać odpowiedzi na każde pytanie.
- Rozmowa nie jest z góry zaplanowana.
- Nie ma ustalonego z góry sterowania.
- Dialog jest syntezowany na podstawie obecnie zgromadzonej wiedzy. Jeżeli zatem nie podana zostanie odpowiedź na pewne pytanie, to będzie to skutkowało kolejnymi zapytaniami.

Zastosowanie SE

- finanse (analiza ryzyka kredytowego czy inwestycyjnego, monitorowanie spłat kredytu, ...)
- medycyna (analiza wyników badań pacjenta, diagnoza choroby, ocena trafności leczenia, ...),
- kontrola procesów
- produkcja (planowanie produkcji, kontrola jakości, monitorowanie produkcji, ...)
- zarządzanie zasobami ludzkimi
- rolnictwo
- edukacja
- prawo (odtwarzanie norm prawnych)
- administracja (decyzje dotyczące wymiaru emerytur, monitorowanie decyzji urzędników)

Klasyfikacja SE

- interpretacyjne (np. do rozpoznawania mowy czy obrazów)
- predykcyjne (wnioskują o przyszłości)
- diagnostyczne (określają wady przedmiotu ekspertyzy)
- kompletowania (np. ustalanie konfiguracji komputera)
- planowania (np. planują ruchy robota dla osiągnięcia jakiegoś celu)
- monitorowania (porównują obserwacje z ograniczeniami)
- sterowania (kierują zachowaniem systemu)
- poprawiania (podają sposób postępowania z systemem)
- naprawy (harmonogram naprawy uszkodzenia)
- instruowania (np. systemy doskonalenia zawodowego)

CLIPS

Definicja

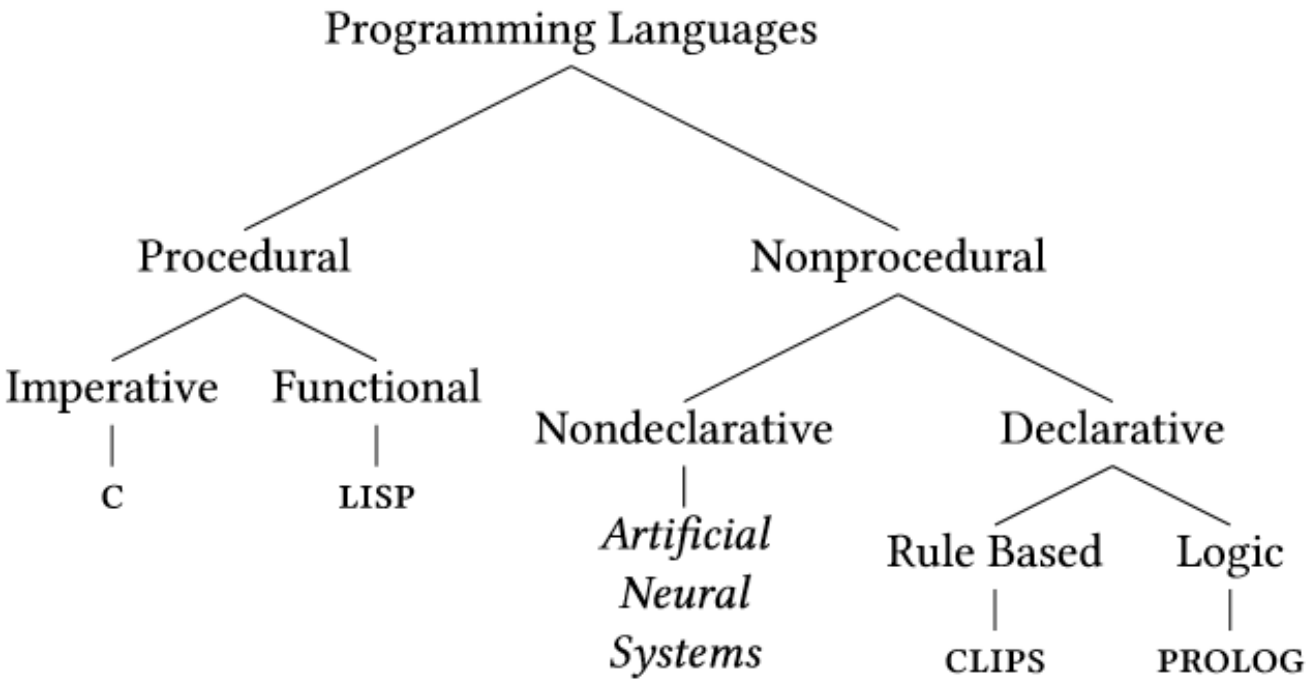
CLIPS (CLIPS = C Language Integrated Production System) - środowisko open source do tworzenia systemów ekspertowych/regułowych. Stworzone w NASA - Johnson Space Center.

Wnioskowanie

Mechanizmy wewnętrzne tego języka realizują wnioskowanie w przód / progresywne. Regułowy system ekspertowy zakodowany w języku CLIPS oparty jest o koncepcje wnioskowania forward chaining, data-driven inference.

CLIPS jako język programowania

CLIPS jest nieproceduralnym, deklaratywnym i opartym na regułach językiem programowania.



Wnioskowanie w przód podstawa działania

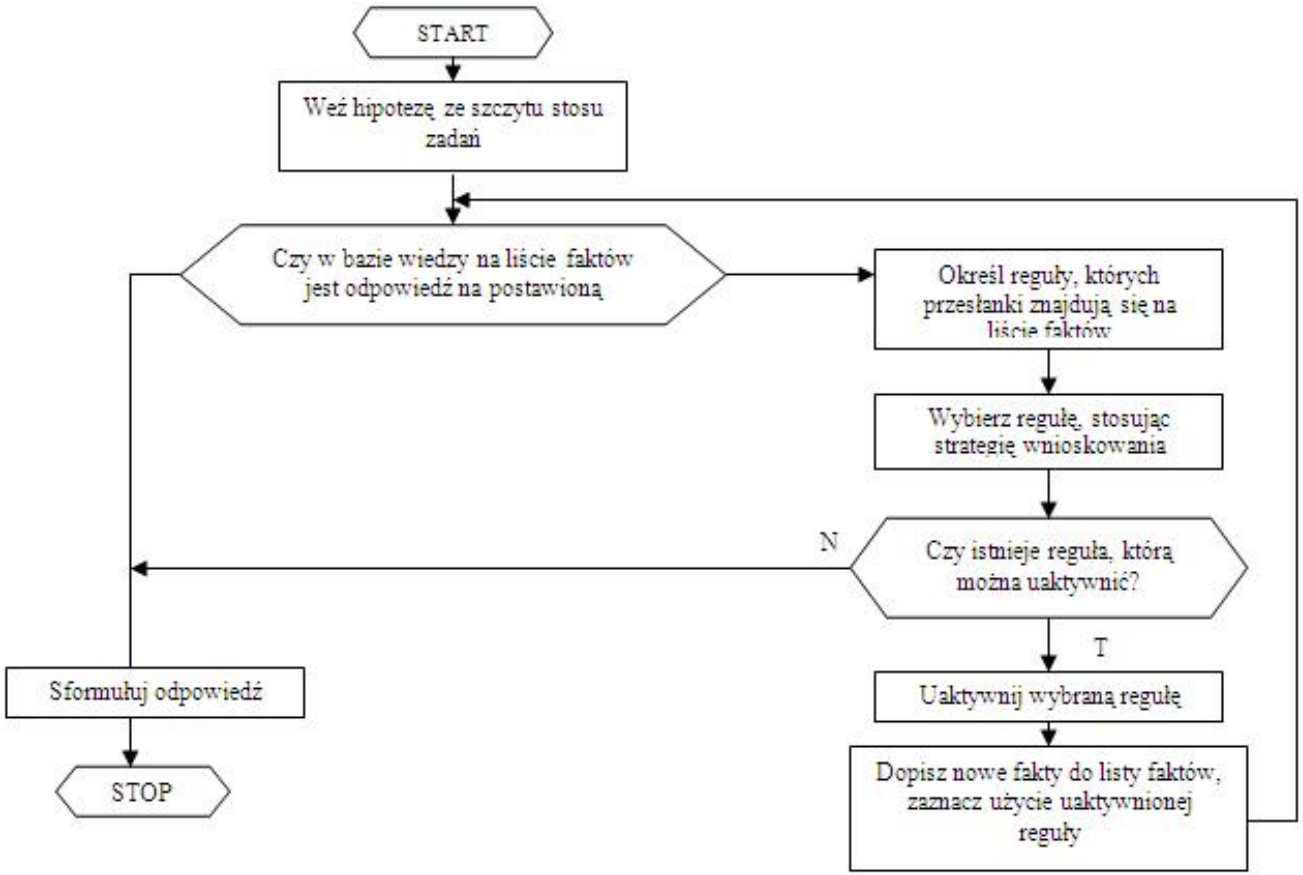
Algorytm wnioskowania w przód

- Dana jest KB = Wiedza początkowa / Fakty
- Dopóki osiągnięto cel lub nie można zastosować więcej reguł

1. Określ zbiór reguł C w bazie wiedzy KB(R), dla których są spełnione przesłanki.
2. Ze zbioru C wybierz regułę R na podstawie strategii sterowania wnioskowaniem.
3. KB = KB + Wynik uaktywnienia reguły R działający na KB plus KB(R).

Schemat wnioskowania w CLIPS

Schemat



Shell

Program napisany w języku CLIPS składa się z

- reguł
- faktów
- obiektów.

W terminologii języka słowem shell (powłoka) określa się tę jego część, która zawiera podstawowe elementy systemu ekspertowego:

1. Wykaz faktów (fact-list) i wykaz obiektów (instance-list) w pamięci roboczej.
2. Baza wiedzy (knowledge-base) w postaci zbioru reguł oraz agenda reguł.
3. Maszyna wnioskująca (inference-engine).

Można zatem powiedzieć, że system ekspertowy napisany w języku CLIPS jest programem sterowanym danymi, ponieważ fakty i obiekty są tymi danymi, które stymulują wykonanie poprzez maszynę wnioskującą.

Fakty (assert)

- Fakty są wprowadzane na listę faktów poleceniem (assert), np.

  (assert (kolor))

- Każdy fakt może być wpisany na listę faktów tylko jeden raz (Clips nie przyjmuje duplikatów).
- W faktach wykorzystuje się trojakie wartości atomowe:
  1. symboliczne – zaczynające się od litery, po której opcjonalnie następują: litery, cyfry, myślnik lub znaki podkreślenia
  2. ciągi znaków – zawierające dowolne znaki; zaczynają się one i kończą cudzysłowem, który również wchodzi w skład atomu
  3. numeryczne – liczby.

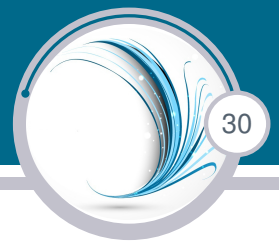
Fakty (deffacts)

W celu uniknięcia konieczności wielokrotnego wpisywania pojedynczych faktów można zdefiniować blok faktów, używając polecenia (deffacts), np.

(deffacts skrzyzowanie
(status stoj)
(status idz))

Po słowie kluczowym (deffacts) podaje się nazwę definiowanego bloku, a następnie fakty (w nawiasach). Cała instrukcja jest ujęta w nawiasy. Fakty zdefiniowane w ten sposób są wprowadzane na listę faktów przez polecenie (reset).

Polecenie (reset)



- (reset) usuwa wszystkie fakty z listy, a następnie wprowadza fakty zdefiniowane przez (deffacts) oraz fakt (initial-fact) z indeksem f-0.
- (initial-fact) jest użyteczny przy startowaniu programów, ponieważ jest często umieszczony w części warunkowej reguły.
- Fakty zdefiniowane przez (deffacts) i wprowadzone na listę faktów, przez (reset) mogą być w prosty sposób usunięte z tej listy przez użycie słowa kluczowego (undeffacts) wraz z nazwą bloku faktów np.
  (undeffacts skrzyzowanie)
  Polecenie to jednocześnie usuwa zadeklarowany blok faktów

Polecenie (retract) i (clear)

- Służy do usuwania faktu z listy faktów.
- Fakt można usunąć jedynie przez podanie jego indeksu, nie można natomiast usunąć przez podanie samego faktu.
- Aby usunąć z listy fakt (filtr) o indeksie f-1, należy wprowadzić *(retract 1)*.
- Polecenie (clear) usuwa wszystkie fakty z listy faktów. Polecenie to usuwa jednak również reguły.

CLIPS sesja
cz. 1


```
CLIPS> (assert (initial fact))
<Fact-0>
CLIPS> (assert (day monday))
<Fact-1>

CLIPS> (facts)
f-0     (initial fact)
f-1     (day monday)
For a total of 2 facts.

CLIPS> (retract 1)
CLIPS> (facts)
f-0     (initial fact)
For a total of 1 fact.
```

CLIPS sesja
cz. 2


```
CLIPS> (retract 1)
CLIPS> (facts)
f-0     (initial fact)
For a total of 1 fact.

CLIPS> (watch facts)
CLIPS> (assert (day monday))
==> f-2     (day monday)
<Fact-2>

CLIPS> (facts)
f-0     (initial fact)
f-2     (day monday)
For a total of 2 facts.
```

CLIPS sesja
cz. 3


```
CLIPS>(clear)
CLIPS>(facts)
CLIPS>(unwatch facts)
CLIPS>(assert (rainy day))
CLIPS>(defrule weather (rainy day) => (assert (carry umbrella)))
CLIPS>(facts)
f-0     (rainy day)

CLIPS>(run)
CLIPS>(facts)
f-0     (rainy day)
f-1     (carry umbrella)

CLIPS>(exit)
```

Reguły

W CLIPSie reguły mają konstrukcję zbliżoną do instrukcji
(IF ... THEN ...),
używanych w powszechnie stosowanych językach programowania.
Słowo IF dla uproszczenia jest pomijane, słowo THEN jest zastępowane symbolem =>. Zgodnie z tym zdanie:
IF jest filtr
THEN wprowadź akcję
jest reprezentowane w CLIPSie przez poniższą regułę:

```
(defrule stop ”Regula 1”
(kolor czerwony)
=>
(assert (status stoj)))
```

Konstrukcja reguły

1. Nawiasy. Cała reguła musi być umieszczona w nawisach.
2. Słowo kluczowe defrule. Oznacza, że następujących dalej ciąg znaków definiuje regułę.
3. Obligatoryjna nazwa reguły. Użycie po raz drugi tej samej nazwy powoduje usunięcie poprzednio wprowadzonej reguły o tej nazwie. Nazwą reguły może być tylko atom zaczynający się od litery.
4. Opcjonalny komentarz (w cudzysłowie).
5. Pole warunków. Może ono zawierać jeden lub więcej warunków. Jeśli wszystkie warunki są spełnione, to reguła przechodzi do zbioru reguł zwanego agendą. Tylko reguły z tego zbioru mogą być później uaktywnione.
6. Symbol => reprezentuje w regule słowo THEN.
7. Lista działań, jakie zostaną podjęte w chwili uaktywnienia reguły.

Część reguły znajdująca się po lewej stronie symbolu => nosi nazwę LHS (left hand side), a po prawej stronie RHS (right hand side).

CLIPS składnia reguły

(if ... then ... else)
(if (<warunek>) then (<akcja1>) else (<akcja2>))

```
(defrule rule_name

   (pattern_1)   ; IF Condition 1
   (pattern_2)   ; And Condition 2
   .
   .
   (pattern_N)   ; And Condition N
   =>   ; THEN
   (action_1) ; Perform Action 1
   (action_2) ; And Action 2
   .
   .
   (action_N)) ; And Action N
```

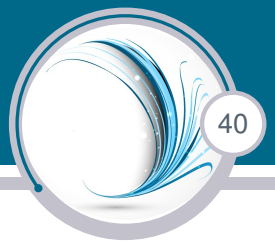
Polecenie (run)

- (run) uruchamia wykonanie programu. Powinno być poprzedzone komendą (reset).
- Użycie polecenia (run) z liczbą np. (run 25) powoduje uaktywnienie tylko 25 reguł.
- Tylko te reguły są uaktywniane, które znajdują się w agendzie. Czyli takie, gdy są spełnione wszystkie warunki znajdujące się w polu warunków reguły (LHS).
- W polu warunków musi wystąpić przynajmniej jeden fakt lub jedna zmienna
- Reguła może być uaktywniona tylko przez te fakty z listy faktów, które zostały wprowadzone na tę listę po zdefiniowaniu reguły.
- Reguła po uaktywnieniu nie może być powtórnie uaktywniona przez te same fakty. Aby jakaś reguła została ponownie uaktywniona, użyte przez nią fakty muszą być usunięte z listy faktów, po czym jeszcze raz za nią wprowadzone.

Operacje na regułach

- (ppdefrule) — podgląd treści reguły.
- (rules) — lista nazw zdefiniowanych reguł.
- (excise) — usuwanie reguły, np.
  (excise stop)

Wnioskowanie w przód podstawa działania



Algorytm wnioskowania w przód

- pattern: fakt, który może być dopasowany do wzorca
- action: potwierdzenie faktów
- wszystkie zmienne mają prefiksem ? (np., ?x)
- zmienne nie muszą być jednoznacznie zadeklarowane, ale muszą być zainicjowane przed użyciem
- wszystkie operacje są wyrażone w zapisie prefiksowym
- gdy porównania są warunkami należy je użyć w następujący sposób: (test (> 5 0))

Zmienne

Nazwy zmiennych używanych w CLIPSie rozpoczynają się od znaku zapytania, np.:
?x
?nazwisko

Nazwami zmiennych mogą być tylko atomy symboliczne.
Przykład:

(defrule pisz
(nazwisko ?x)
=>
(printout t ?x crlf))

(assert (nazwisko Kowalski) (nazwisko Nawrocki) (Jan Kabacki))

Do zmiennej ?x będą przypisywane kolejno fakty (Kowalski), (Nawrocki). Po wprowadzeniu faktów system uaktywniania regułę dla faktu (Nawrocki) i po raz drugi dla faktu (Kowalski).

CLIPS przykład kodu


```
(defrule factorial
   (fact_run ?x)
   =>
   (assert (fact ?x 1)))

(defrule fact_helper
   (fact ?x ?y)
   (test (> ?x 0))
   =>
   (assert (fact (- ?x 1) (* ?x ?y))))

(assert (fact_run 5))
```

CLIPS działanie


```
CLIPS> (assert (fact_run 5))
<Fact-0>

CLIPS> (facts)
f-0 (fact_run 5)
For a total of 1 fact.

CLIPS> (run)
CLIPS> (facts)
f-0     (fact_run 5)
f-1     (fact 5 1)
f-2     (fact 4 5)
f-3     (fact 3 20)
f-4     (fact 2 60)
f-5     (fact 1 120)
f-6     (fact 0 120)
For a total of 7 facts.
```

Opis algorytmu

Działanie

- Reguła fact_helper sprawdza istnienie faktu „fact”, po którym następują dwie wartości.
- Poprzez dopasowanie wzorca, ?x i ?y są zainicjowane.
- Gdy ?x i ?y mają wartość, nie mogą być zmienione w ramach tego samego dopasowania (ale mogę być zmienione przy kolejnych dopasowaniach do reguły).
- Zmienne mogą być również definiowane globalnie przy użyciu słowa kluczowego defglobal.
- Zmienne globalne mogą być przypisane w części akcji reguły.
- Uwaga: nie jest to dokładnie rekurencja. Raczej deklarowany jest nowy fakt, który pasuje do stanu reguły powodując ponowne wykonanie akcji danej reguły.

Kontrola wykonywania programów

Polecenie (watch) umożliwia kontrolę wykonywania programu.
Polecenia włączają sygnalizację podczas:

- (watch facts) — wprowadzenia faktów i usuwania faktów
- (watch rules) — wykonywania reguł
- (watch activations) - uaktywniania reguł

Polecenie (unwatch) wraz ze spcecyfikacją
(unwatch facts)
(unwatch rules)
(unwatch activations)
anuluje działanie polecenia (watch).

Inne przykłady

(defrule nazwiska
(nazw ?x)
=>
(assert (pan ?x)))

(defrule podwojne)
(?x ?x)
=>
(printout t „Imie nazwisko pana” ?x „sa jednakowe” crlf))

(defrule redukcja
?wyrzuc <- (Jan ?x)
=>
(printout t „Z listy został usunięty pan” ?wyrzuc crlf))
(retract ?wyrzuc))

Zmienna anonimowa

- Zmienna anonimowa (wildcard), reprezentowana jest przez znak zapytania.
- Ze zmienną anonimową może być związany tylko jeden atom.
- Wykorzystuje się ją tam, gdzie konieczne jest zaznaczenie obecności atomu, ale jego wartość nie ma znaczenia

  (defrule imie
  (Jan ?)
  =>
  (printout „Został znaleziony Jan” crlf))
- Wielokrotna zmienna anonimowa (multi–field–wildcard) jest symbolizowana przez $? (znak dolara i znak zapytania bez żadnej nazwy).
- Można do niej przypisać dowolną liczbę (włącznie z zerem) atomów.

CLIPS - szablony

Algorytm wnioskowania w przód

- Szablony są używane do powiązania danych w jednym pakiecie.
- Szablony są kontenerami z faktami, gdzie każdy fakt jest miejscem (slot) w szablonie.
- Szablony mogą być użyte podobnie jak fakty do tworzenia baz; reguły mogą być używane do zapytań do takiej bazy.

CLIPS szblony


```
(deftemplate car
   (slot make
     (type SYMBOL)
     (allowed-symbols
      truck compact)
     (default compact))
   (multislot name
      (type SYMBOL)
      (default ?DERIVE)))

(deffacts cars
   (car (make truck)
        (name Tundra))
   (car (make compact)
        (name Accord))
   (car (make compact)
        (name Passat)))

(defrule compactcar
   (car (make compact) ;select only cars that are compact
         (name ?name))
   =>
   (printout t ?name crlf))
```

Wyrażenia matematyczne

- Liczby bez kropki to long long a z kropką to float.
- Elementarne operatory arytmetyczne są następujące: + dodawania, - odejmowania, * mnożenia, / dzielenia, ** potęgowania.
- Operatory nie mają priorytetów.
- Wyrażenia arytmetyczne mają postać prefiksową
  (operator arg arg)
  (+ ?x ?y)
  (- ?x ?y ?z)
  jest równoważne odejmowaniu ?x - ?y - ?z

Wprowadzenie wyniku do listy faktów


```
(defrule iloczyn
(liczby ?x ?y)
=>
(assert (=(* ?x ?y))))
```

Po wprowadzeniu tej reguły oraz faktu (liczby 5 6) i uruchomieniu programu, na liście faktów znajdą się:

f-1 (liczby 5 6)
f-2 (30)

CLIPS funkcje zdefiniowane przez użytkownika


```
(deffunction increment (?i)
    (+ ?i 1)
)
(increment 6)

(deffunction fibbonacci (?f)
    (if (or (= ?f 1) (= ?f 0) ) then
        1
    else
        (+ (fibbonacci (- ?f 1) )
            (fibbonacci (- ?f 2) )
        )
    )
)
(fibbonacci 5)
```

Źródła

- https://searchcio.techtarget.com/definition/knowledge-based-systems-KBS
- http://edu.pjwstk.edu.pl/wyklady/adn/scb/wyklad10/w10.htm
- S.J. Russel, P. Norvig, Artificial Intelligence. A modern approach.
- http://perugini.cps.udayton.edu/teaching/courses/Spring2017/cps499/Languages/notes/CLIPS.html

Dziękuję za uwagę
Czas na pytania ????