Od Drzew Tablicowych do Sieciowych Tablic Decyzyjnych: ewolucja zmodularyzowanych reprezentacji wiedzy

Igor Wojnicki

AGH Akademia Górniczo-Hutnicza, Wydział EAIiE, Katedra Automatyki

Streszczenie: Stosując systemy regułowe, można wyróżnić dwie klasy problemów: efektywność procesu projektowania oraz wydajność procesu uruchamiania. Niniejszy artykuł przeglądowo opisuje metody projektowania reguł opracowane w Katedrze Automatyki AGH pod kątem rozwiązywania ww. problemów. Proponowane podejścia opierają się w dużej mierze na modularyzacji bazy wiedzy i odpowiedniej jej wizualizacji. Przedstawiono ich ewolucję, jak również wyniki najnowszych badań dotyczących wnioskowania kontekstowego.

Słowa kluczowe: reguły, wnioskowanie, reprezentacja reguł, kontekst, modularność

Reguły są powszechnie stosowaną metodą wyrażania logiki funkcjonowania systemów informatycznych. Znajdują zastosowanie zarówno w fazie projektowania tych systemów, jak i ich implementacji. Ponadto używane są w różnych dziedzinach, takich jak Logika Biznesowa (przykładowe systemy: ILOG Rules, Drools, LPA VisiRule, G2 Gensym), czy sterowanie (G2 Gensym, Tiger). W Logice Biznesowej reguły służą modelowaniu procesów zachodzących w przedsiębiorstwach oraz przepływu informacji z nimi związanych. Pozwalają na wyrażenie zasad takich przepływów w sposób zrozumiały dla kadry kierowniczej. Ponadto łatwość wizualizacji reguł zwiększa ich czytelność i możliwość późniejszej modyfikacji. W sterowaniu reguły znajdują zastosowanie wszędzie tam, gdzie wymagana jest przejrzystość zasad funkcjonowania lub potrzeba formalnej weryfikacji zdefiniowanych procedur sterujących.

Każdy system oparty na regułach (tzw. System Regułowy) składa się z bazy wiedzy, czyli reguł, oraz z ogólnego mechanizmu wnioskowania pozwalającego interpretować te reguły, tzw. maszyny wnioskującej. Programowanie takiego systemu sprowadza się zatem jedynie do wyspecyfikowania zbioru reguł, czyli deklaratywnego określenia warunków jego działania.

Wykorzystując podejście regułowe, można zidentyfikować dwie klasy problemów: skalowalność procesu projektowania (tworzenia bazy wiedzy) oraz wydajność procesu interpretacji samych reguł. W przypadku Logiki Biznesowej bardziej widoczna jest pierwsza klasa problemów, tj. efektywność procesu projektowania. W przypadku zastosowania w sterowaniu – druga.

Niniejszy artykuł przeglądowo opisuje badania naukowe podjęte w Katedrze Automatyki AGH, mające na celu rozwiązanie obu klas wspomnianych problemów.

1. Motywacja

Reguły są prostym, aczkolwiek silnym środkiem wyrazu do specyfikacji zachowania systemów informatycznych. W najprostszej formie pojedyncza reguła składa się z warunków oraz decyzji. Jeżeli warunki reguły są spełnione, maszyna wnioskująca podejmuje zdefiniowane decyzje. Zatem projektowanie reguł sprowadza się do określenia decyzji podejmowanych przy zaistnieniu określonych warunków.

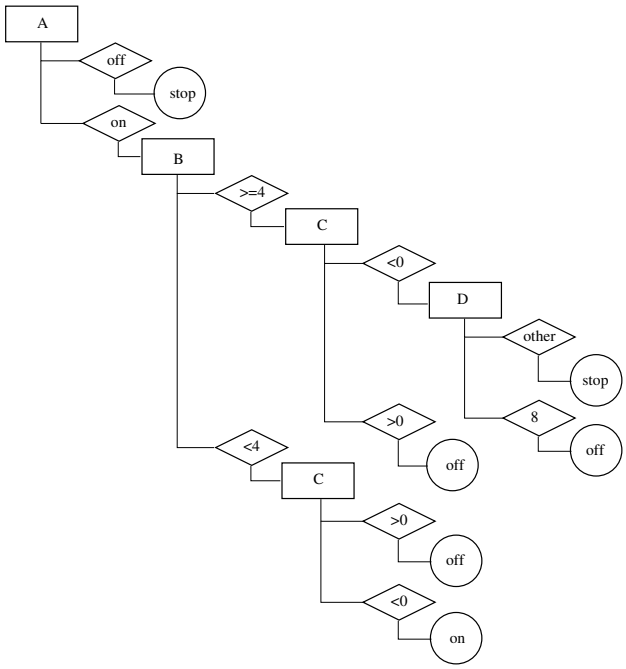
W celu zwiększenia czytelności procesu projektowania często używane są różne techniki wizualizacyjne mające na celu przedstawienie zbioru reguł w bardziej zrozumiałej dla projektanta formie. Często stosowane są tablice czy też drzewa decyzyjne, ilustrujące reguły odpowiednio w formie tabelarycznej bądź drzewa.

Tablice decyzyjne charakteryzują się dużą gęstością informacji, zatem mogą reprezentować relatywnie dużą liczbę reguł w stosunku do zajmowanej powierzchni, aczkolwiek reguły przedstawione w ten sposób często nie są przejrzyste. Wszystkie reguły znajdują się w pojedynczej tabeli, gdzie każda z kolumn (albo wierszy, w zależności od orientacji samej tabeli) reprezentuje pojedynczą regułę. Dla przykładu, na rys. 1 przedstawiono zbiór reguł, który podejmuje jedną z trzech decyzji reprezentowanych przez obiekt Act, tj. „on”, „off” lub „stop”, w zależności od warunków przekazywanych jako wartości obiektów A, B, C oraz D. Kolejne kolumny reprezentują kolejne reguły. Przykładowo reguła numer 2 definiuje, że jeżeli wartość A jest „on”, wartość B jest „< 4”, wartość C jest „> 0”, a wartość D jest dowolna, decyzją jest „off”.

nr reguły	1	2	3	4	5	6
warunki						
A	on	on	on	on	on	off
B	< 4	< 4	$>= 4$	$>= 4$	$>= 4$	
C	< 0	> 0	< 0	< 0	> 0	
D			8	other		
decyzje						
Act	on	off	off	stop	stop	stop

Rys. 1. Tablica decyzyjna
Fig. 1. A Decision Table

W porównaniu z tablicami decyzyjnymi drzewa decyzyjne bardziej przejrzyście przedstawiają strukturę reguł, ale gęstość informacji w takim drzewie jest niewielka. Przykładowe drzewo decyzyjne, reprezentujące te same reguły co poprzednio dla tablicy decyzyjnej, przedstawione jest na rys. 2.



Rys. 2. Drzewo decyzyjne
Fig. 2. A Decision Tree

O ile w przypadku programowania Logiki Biznesowej wydajność interpretacji reguł jest kwestią drugoplanową, o tyle dla systemów sterowania jest ona pierwszoplanowa. Stosując bazę wiedzy bez dodatkowej struktury, projektant ma niewielki wpływ na optymalizację i czas przeprowadzania procesu wnioskowania, gdyż wszystkie reguły, a dokładniej ich warunki, muszą być każdorazowo sprawdzane. Dodatkowo płaska struktura reguł utrudnia proces ich projektowania

Zatem główną motywacją do podjęcia badań była potrzeba stworzenia przejrzystej i informacyjnie pojemnej reprezentacji reguł oraz umożliwienie projektantowi podziału reguł na podzbiory (moduły), co z jednej strony powinno zwiększyć czytelność podczas samego projektowania, a z drugiej strony powinno polepszyć wydajność przy interpretacji reguł.

2. Drzewa Tablicowe

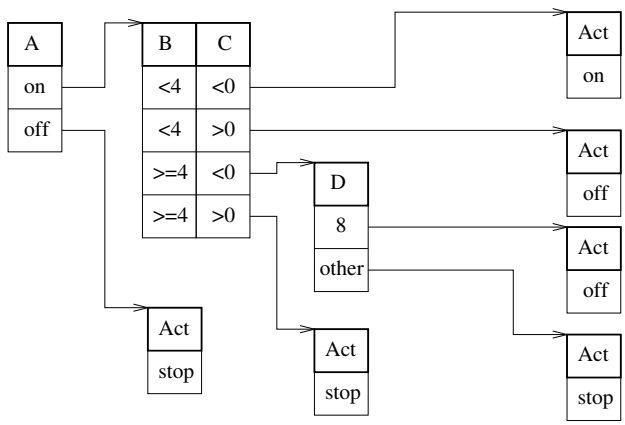
Drzewa Tablicowe (Reprezentacja Drzewiasto-Tablicowa, ang. *Tabular Trees* albo w skrócie *Tab-Trees*) powstały jako metoda projektowania reguł będąca rozwiązaniem dla pierwszej klasy wspomnianych na początku problemów, tj. skalowalności procesu projektowania [3, 5]. Podejście to jest zainspirowane koncepcją Ψ-drzew [3], które z kolei są rozwinięciem binarnych drzew decyzyjnych.

Bezpośrednim zastosowaniem Drzew Tablicowych, służącym do weryfikacji podejścia, było projektowanie reguł dla systemu ekspertowego czasu rzeczywistego Kheops [4]. Reguły systemu Kheops oparte są na atrybutach, lecz w ogólnym przypadku Drzewa Tablicowe mogą być zastosowane do modelowania bazy reguł opisanych innymi formalizmami, podobnie jak tablice czy drzewa decyzyjne.

Przykładowa baza wiedzy, ta sama co przedstawiona poprzednio za pomocą tablicy i drzewa decyzyjnego, pokazana jest na rys. 3. Reprezentacja składa się z połączonych tablic decyzyjnych tworzących strukturę drzewa. Przy czym, w przeciwieństwie do klasycznych tablic, warunki reguły reprezentowane są jako pojedynczy wiersz, a nie kolumna. Nagłówki kolumn reprezentują nazwy obiektów (w tym przypadku atrybutów). Pojedyncza tabela może zawierać warunki albo decyzje, w zależności od rodzaju obiektów występujących w nagłówkach. Każdy wiersz tabeli można łączyć z inną tabelą, co implikuje relację koniunkcji między warunkami albo warunkami i decyzjami. Jeżeli tabela zawiera więcej niż jedną kolumnę (np. tabela o nagłówkach B,C, rys. 3), to warunki definiowane przez wartości atrybutów przypisanych poszczególnym kolumnom traktowane są jako będące w relacji koniunkcji. Jeżeli tabela zawiera decyzję, określony w nagłówku atrybut przyjmuje wartości tej decyzji.

W porównaniu z drzewami decyzyjnymi Drzewa Tablicowe oferują znacznie większą gęstość informacji. Ponadto, w zestawieniu z tablicami decyzyjnymi przedstawiają reguły w sposób bardziej przejrzysty, eliminując powtórzenia warunków (np. powtórzenie wartości „on” dla reguł od 1 do 5, rys. 1).

Należy również zwrócić uwagę na modularność tego rozwiązania: wiele połączonych ze sobą tablic decyzyjnych. W procesie projektowania projektant skupia się tylko na pojedynczej tablicy, jednocześnie mając wgląd w całość projektowanej logiki. Reprezentacja taka wpływa zatem dodatnio na usprawnienie procesu projektowania reguł.



Rys. 3. Drzewo tablicowe (Reprezentacja Drzewiasto-Tablicowa)
Fig. 3. A Tabular Tree (Tab-Tree)

3. Rozszerzone Drzewa Tablicowe

Kolejnym krokiem rozwoju reprezentacji reguł stały się Rozszerzone Drzewa Tablicowe (ang. *eXtended Tabular Trees*, XTT). Główną modyfikacją warstwy wizualnej była integracja warunków i decyzji w ramach pojedynczej tabeli. XTT zostało zaprojektowana pod kątem konkretnego formalizmu reguł, tj. Logiki Atrybutowej [3], aczkolwiek inne formalizmy również mogą być użyte. W szczególności została wprowadzona obsługa nieatomicznych wartości atrybutów oraz wnioskowania niemonotonicznego. Przykładowa, modelowa tabela XTT przedstawiona jest na rys. 4.

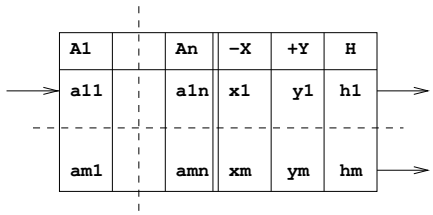
Integracja warunków i decyzji wpływa na zwiększenie gęstości informacyjnej nowej reprezentacji. Należy przy tym zwrócić uwagę, że zmienia się nieco interpretacja

połączeń pomiędzy tabelami (na rysunkach oznaczone jako strzałki). W przypadku Drzew Tablicowych połączenia symbolizowały koniunkcję warunków lub decyzji. Natomiast w przypadku XTT jest to raczej „przejście" do innej tabeli, lub nawet konkretnego wiersza w tabeli, a zatem określenie mechanizmu kontroli procesu wnioskowania, a nie tylko wizualizacji. O ile pojedyncza reguła w przypadku Drzew Tablicowych może być wizualizowana jako szereg połączonych tabel, o tyle w przypadku XTT będzie to pojedynczy wiersz w jednej tabeli.

Pozostałe rozszerzenia w znacznym stopniu wpływają na siłę wyrazu reprezentacji. Wprowadzenie obsługi wartości nieatomicznych powoduje, że pojedynczy atrybut może przyjmować nie tylko atomiczną wartość, ale również zbiór wartości. W konsekwencji powoduje to, że w warunkach mogą zostać użyte operacje na zbiorach. Samo nadanie więcej niż jednej wartości danemu atrybutowi dokonywane jest przez operacje typu dodaj/usuń (*assert/retract*) oznaczone odpowiednio znakiem „+" i „−" w nagłówku tabeli. Umożliwia to wnioskowanie niemonotoniczne.

Pojedyncza tabela XTT (rys. 4) podzielona jest na dwie części: lewą, warunkową oraz prawą, decyzyjną, gdzie separatorem jest podwójna linia. Przykładowo, pierwszy wiersz należy interpretować jako: „jeżeli $(A_1 \in a_{11}) \wedge \ldots (A_n \in a_{1n})$, to usuń $(X = x_1)$, dodaj $(Y = y_1)$, ustaw $(H = h_1)$", gdzie A_i, X, Y, H to atrybuty, a a_{ij}, x_1, y_1, h_1 konkretne wartości (w ogólnym przypadku może zostać wyspecyfikowane więcej kolumn w części decyzyjnej reprezentujących dowolną ilość operacji dodaj/usuń lub ustaw).

Proponowane podejście zostało z powodzeniem wykorzystane do modelowania klasycznych systemów regułowych [3].

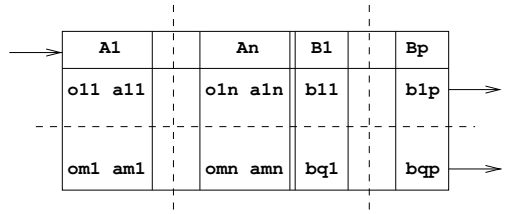


Rys. 4. Pojedyncza tabela Rozszerzonego Drzewa Tablicowego (XTT)
Fig. 4. A single XTT (eXtended Tabular Tree) table

Wprowadzając mechanizm kontroli nad procesem wnioskowania (połączenia między tablicami), utworzone zostało rozwiązanie dla drugiej klasy problemów wspomnianych na początku, tj. wydajności procesu interpretacji reguł. Projektant ma możliwość zadecydowania o ograniczeniu zbioru reguł, które maszyna wnioskująca musi brać pod uwagę, gdyż w danej chwili analizowane są warunki reguł co najwyżej dla pojedynczej tabeli. Sprzyja to zastosowaniu XTT do modelowania systemów wymagających determinizmu i optymalizacji czasu realizacji określonych zadań, takich jak systemy sterowania.

Pojedyncza tabela XTT często nazywana jest kontekstem, gdyż definiuje zachowanie całości systemu w określonym przypadku. Zatem, w porównaniu do Drzew Tablicowych, została jeszcze bardzie zaakcentowana koncepcja modularności: pojedyncza tabela (kontekst) staje się samodzielnym modułem wiedzy o zachowaniu się systemu w określonych warunkach.

XTT było wielokrotnie modyfikowane, uzupełniane i rozbudowywane [7]. W rezultacie badań powstała druga wersja XTT oznaczona jako XTT^2 albo XTT2.



Rys. 5. Pojedyncza tabela XTT^2
Fig. 5. A Single XTT^2 Table

XTT^2 projektowano, podobnie jak XTT, pod katem Logiki Atrybutowej. Aczkolwiek sama reprezentacja jest bardziej elastyczna, łatwiej niż w przypadku XTT zastosować tu inny formalizm reprezentacji reguł.

Przykładowa tabela XTT przedstawiona jest na rys. 5. Strona wizualna została uproszczona. Usunięto oznaczenia „dodaj/usuń" w nagłówkach kolumn na rzecz operatorów sumy i różnicy zbiorów, które mogą być zastosowane w części decyzyjnej. Przejścia zostały ograniczone do przejść między konkretnym wierszem a docelową tabelą (przejście do konkretnego wiersza w docelowej tabeli zostało zabronione). Ponadto struktura drzewiasta została rozszerzona do struktury grafu, umożliwiając modelowanie bardziej skomplikowanych systemów – m.in. programowania aplikacji ogólnego przeznaczenia [9].

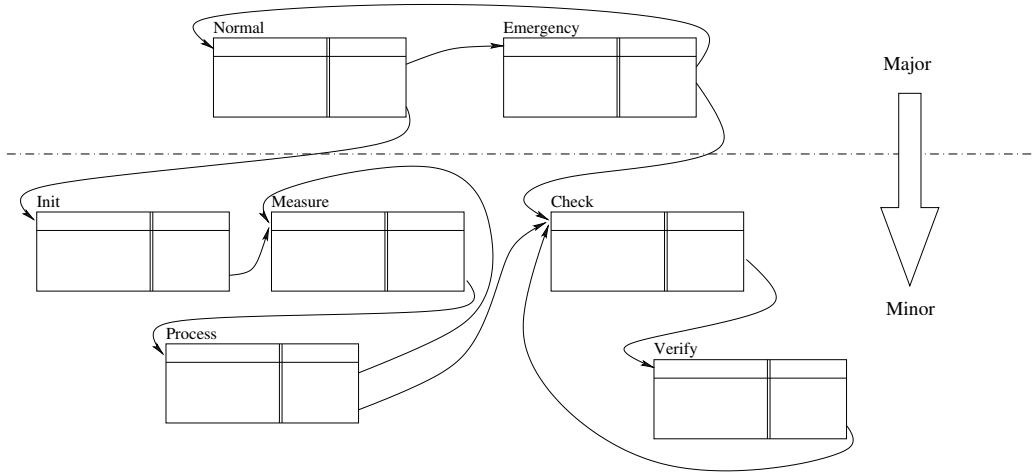
Pojedyncza reguła (pierwszy wiersz) jest interpretowana jako: „jeżeli $(A_1\ o_{11}\ a_{11}) \wedge \ldots (A_n\ o_{1n}\ a_{1n})$ to $(B_1 = b_{11} \wedge \ldots B_p = b_{1p})$", gdzie A_i, B_j to atrybuty, o_{ik} operatory logiczne, a_{ik} oraz b_{jk} to wartości lub wyrażenia obliczalne do wartości, przy czym wartość może być pojedynczym elementem (np. liczbą) albo zbiorem takich elementów. W ramach wyrażeń mogą zostać użyte zdefiniowane operatory, m.in. arytmetyczne lub operacji na zbiorach.

4. Sieciowe Tablice Decyzyjne

Sieciowe Tablice Decyzyjne (ang. *Networked Decision Tables*, NDeT) są rezultatem kontynuacji prac nad rozwojem XTT^2. Zmodyfikowano zasady procesu wnioskowania wprowadzając pojęcie aktywnego kontekstu. Pozwoliło to na eliminacje samoprzejść (przejście do tej samej tabeli), co wpłynęło na zwiększenie przejrzystości warstwy wizualnej i czytelności reguł.

Ponadto sprecyzowano pojęcie przełączania oraz aktywacji kontekstów, do czego skłoniły wcześniejsze badania prowadzone w tej dziedzinie [3, 6, 8]. Wprowadzono również możliwość zastosowania hierarchii kontekstów, czego rezultatem są Kontekstowe Sieciowe Tablice Decyzyjne (ang. *Contextual Networked Decision Tables*, CoNDeT). Koncepcja hierarchii kontekstów bazuje na Wnioskowaniu Kontekstowym (ang. *Context-Based Reasoning*, CxBR) [1].

Aby rozpocząć proces interpretacji reguł, należy określić aktywny kontekst (tabelę). Warunki reguł są sprawdzane tylko dla aktywnego kontekstu. Aktywny kontekst może



Rys. 6. Kontekstowe Sieciowe Tablice Decyzyjne, CoNDeT
Fig. 6. Contextual Networked Decision Tables, CoNDeT

zmienić się na skutek uruchomienia reguły ze zdefiniowanym przejściem do innej tabeli. Kontekst może być również dezaktywowany, co kończy proces wnioskowania. Najwyżej jeden kontekst może być aktywny w tym samym czasie.

W przypadku CoNDeT każdy z kontekstów przynależy do odpowiedniej klasy. Klasy kontekstów tworzą hierarchię. Co najwyżej jeden kontekst z każdego poziomu takiej hierarchii może być aktywny. Podczas realizacji procesu wnioskowania najpierw sprawdzane są warunki reguł w aktywnym kontekście przynależącym do klasy najwyżej w hierarchii, następnie w aktywnym kontekście o klasę niższym itd. Reguła z przejściem przełącza aktywny kontekst, o ile kontekst docelowy należy do tej samej klasy, bądź też aktywizuje kontekst, jeżeli należy on do innej klasy. Praktyczne doświadczenia pokazały, że dwa poziomy hierarchii klas kontekstów są wystarczające w większości badanych przypadków.

Przykładowe konteksty wraz z hierarchią pokazane są na rys. 6. Można wyróżnić dwie klasy kontekstów: *Major* oraz *Minor*, gdzie *Major* jest klasą nadrzędną. W ramach klasy *Major* można wyróżnić dwa konteksty: *Normal* oraz *Emergency*. Dostarczają one reguł, które są zawsze egzekwowane w warunkach, odpowiednio, pracy normalnej oraz awaryjnej sterowanego urządzenia. W ramach klasy *Minor* zdefiniowano pięć kontekstów reprezentujących poszczególne etapy proces sterowania: inicjalizację (*Init*), pomiary (*Measure*), przetwarzanie (*Process*), sprawdzanie poprawności wyników cząstkowych (*Check*) oraz weryfikację (*Verify*). Odpowiednie przejścia pomiędzy kontekstami oznaczone są strzałkami. Dla zwiększenia czytelności schematu poszczególne reguły pominięto.

Hierarchia kontekstów okazała się przydatna przy zastosowaniu CoNDeT do wysokopoziomowego programowania urządzeń sterujących. Z dużą łatwością można oprogramować sytuacje wyjątkowe, takie jak awaryjne wyłączenie maszyny niezależne od aktualnego jej stanu, czy przejście w inny tryb pracy przy określonych parametrach.

Z punktu widzenia Inżynierii Oprogramowania posłużenie się hierarchią kontekstów jest podobne do koncepcji programowania aspektowego [2]. Konteksty należące do klas wyższych mogą być analogiem aspektów. Przy takim podejściu zamiast prostych warunków reguł należałoby dodatkowo wprowadzić tzw. pre- i post-warunki, sprawdzane odpowiednio przed oraz po warunkach reguł w kontekstach klas niższych. Zagadnienie to jest przedmiotem dalszych prac badawczych.

W przypadku CoNDeT mamy zatem do czynienia z wielowymiarową modularyzacją. Konteksty są pojedynczymi modułami, natomiast hierarchia klas kontekstów pozwala definiować związki zawierania pomiędzy nimi. Upraszczając, można powiedzieć, że konteksty wyższych klas mogą zawierać konteksty klas niższych, o ile istnieją odpowiednie reguły aktywujące te konteksty. Można zatem wyróżnić dwa poziomy modularyzacji specyfikowanej wiedzy o różnej granularności semantycznej. Bardziej szczegółowy poziom zapewnia modularyzację na poziomie zgrupowanych reguł, czyli kontekstów. Bardziej ogólny poziom zapewnia modularyzację na poziomie zgrupowanych kontekstów, czyli klas.

5. Podsumowanie i dalsze badania

Reguły są popularnym środkiem wyrazu logiki funkcjonowania systemów informatycznych. Badania przybliżone w niniejszym artykule dotyczą rozwiązywania problemów związanych z procesem ich projektowania oraz interpretacji.

Celem usprawnienia procesu projektowania powstało szereg reprezentacji reguł, począwszy od Drzew Tablicowych, poprzez Rozszerzone Drzewa Tablicowe (XTT), aż po Kontekstowe Sieciowe Tablice Decyzyjne (CoNDeT). Zaproponowane sposoby wizualizacji reguł zwiększają czytelność bazy wiedzy. Modularyzacja natomiast usprawnia pod względem wydajności zarówno proces projektowania, jak i późniejszego uruchamiania reguł. Ponadto wprowadzenie hierarchii klas kontekstów (CoNDeT) umożliwia modelowanie problemów, w których wymagana jest efektywna obsługa sytuacji i zdarzeń wyjątkowych.

Aktualne prace badawcze skupiają się na wykorzystaniu CoNDeT do wysokopoziomowego sterowania, jak również modelowania aplikacji ogólnego przeznaczenia. Tworzony jest także zestaw narzędzi informatycznych do wspomagania projektowania bazy wiedzy oraz prototypowe środowisko uruchomieniowe dla efektywnej interpretacji reguł.

Bibliografia

1. Gonzalez A. J., Stensrud B. S., Barrett G. C. (2008): *Formalizing context-based reasoning: A modeling paradigm for representing tactical human behavior*, Int. J. Intell. Syst. 23(7), 822–847.
2. Kiczales G. (1996): *Aspect-Oriented Programming*, ACM Comput. Surv. 28(4es), 154.
3. Ligęza A. (2006): *Logical Foundations for Rule-Based Systems*. Springer-Verlag, Berlin, Heidelberg.
4. Ligęza A., Nalepa G., Wojnicki I. (2000): *Analysis of selected problems of design and implementation of real-time rule-based systems on the base of Kheops system (in Polish)*, [w:] Szmuc T., Klimek R. (red.), Real Time Systems 2000, 53–64, Institute of Automatics AGH, Cracow.
5. Ligęza A., Wojnicki I., Nalepa G. J. (2001): *Tab-Trees: a CASE tool for the design of extended tabular systems*, [w:] Proceedings of the 12th international conference, DEXA 2001, Database and expert systems applications, Munich.
6. Ligęza A. (2001): *Toward logical analysis of tabular rule-based systems*, International Journal of Intelligent Systems 16(3), 333–360, special issue on Verification and Validation Issues in Databases, Knowledge-Based Systems, and Ontologies.
7. Nalepa G. J., Wojnicki I. (2009): *Visual Generalized Rule Programming Model for Prolog with Hybrid Operators*, [w:] Seipel D., Hanus M., Wolf A. (red.), INAP2007/WLP 2007, *LNAI*, volume 5437, 178–194, Springer, Berlin Heidelberg.
8. Tzafestas S., Ligęza A. (1989): *Expert control through decision making*, Journal of Intelligent and Robotic Systems 1(4), 407–425.
9. Wojnicki I. (2011): *Implementing General Purpose Applications with the Rule-Based Approach*, [w:] Bassiliades N., Governatori G., Paschke A. (red.), RuleML Europe, *Lecture Notes in Computer Science*, volume 6826, 360–367, Springer.

From Tabular Trees to Networked Decision Tables: an Evolution of Modularized Knowledge-base Representations

Abstract: There are two common issues while dealing with rules and rule-based systems. These are efficiency of the design process and performance of rule interpretation. This paper discusses briefly several design approaches developed at the Department of Automatics, AGH which tackle these issues. The proposed solutions are mainly based on modularization and appropriate visualization of the knowledge base being designed. Evolution of selected approaches and results of recent research regarding application of context-based reasoning are presented as well.

Keywords: rules, reasoning, rule representation, context, modularity

dr inż. Igor Wojnicki

Wieloletni pracownik Katedry Automatyki AGH. Zajmuje się głównie zagadnieniami inżynierii wiedzy, a w szczególności procesami projektowania i wizualizacji reguł na pograniczu sztucznej inteligencji i inżynierii oprogramowania. Doświadczenie zdobywał zarówno w kraju, jak i za granicą. Preferuje Wolne Oprogramowanie (Free Software).
e-mail: wojnicki@agh.edu.pl