Koncepcja narzędzi CAD/CASE do wspomagania projektowania systemów regułowych *

Antoni Ligęza, Igor Wojnicki, Grzegorz J. Nalepa
Katedra Automatyki AGH
ali@ia.agh.edu.pl, wojnicki@agh.edu.pl, gjn@agh.edu.pl

Streszczenie

Z implementacją systemów regułowych związane są dwie główne trudności. Pierwsza z nich to tzw. problem pozyskiwania wiedzy, druga natomiast dotyczy weryfikacji wiedzy. Artykuł przedstawia propozycję rozwiązania tych problemów poprzez wykorzystanie nowego narzędzia, tzw. *Modularnego Systemu Tabelowego*. Modularny System Tabelowy składa się z pewnej liczby tabel (modułów wiedzy). Każda tabela opisuje zestaw reguł działający w określonym kontekście. Połączenia pomiędzy tabelami określają możliwe ścieżki wnioskowania. Zaproponowane w niniejszej pracy podejście opiera się na koncepcji narzędzi wspomagających projektowanie (CAD/CASE) oraz graficznej i algebraicznej reprezentacji wiedzy. Pozwala to rozwiązać problem akwizycji wiedzy poprzez systematyczne projektowanie reguł z użyciem modułowej i hierarchicznej reprezentacji graficznej. Co więcej proponowane rozwiązanie umożliwia weryfikację własności projektowanych reguł takich jak zupełność, spójność i determinizm w trakcie projektowania.

1. Wstęp

Wiele systemów z bazą wiedzy oraz systemów przetwarzających informacje stosuje mechanizmy wnioskowania oparte na przetwarzaniu reguł. Jest to bardzo uniwersalny model przetwarzania wiedzy, a sam sposób wnioskowania wydaje się być najbardziej zbliżonym do rozumowania naturalnego. Systemy regułowe dostarczają prostego, a zarazem potężnego modelu programowania. Mogą być używane w dynamicznych bazach danych, dedukcyjnych bazach danych, systemach ekspertowych, systemach wspomagających podejmowanie decyzji, systemach monitorowania i nadzoru, oraz systemach inteligentnego przetwarzania informacji.

*Praca realizowana w ramach grantu KBN nr: 8 T11C 019 17.

Regułę wnioskowania R_i można przedstawić w postaci:

$$R_i : \; p_1 \wedge p_2 \wedge \ldots \wedge p_n \longrightarrow h, \tag{1}$$

gdzie $p_1, p_2, \ldots, p_n$ są warunkami aktywacji (*odpalenia* reguły; są to tzw. *pre-warunki*), a h jest wnioskiem. Zarówno warunki jak i wniosek mogą być wyrażone przy użyciu różnych języków służących do reprezentacji wiedzy. Systemy regułowe funkcjonalnie zbudowane są z reguł oraz podsystemu wnioskującego. Reguły tworzą program, zaś podsystem wnioskujący sprawuje nadzór nad jego realizacją. Więcej informacji na temat systemów regułowych oraz podsumowanie aktualnego stanu wiedzy można znaleźć w pracy [5].

Chociaż regułowy model programowania jest prosty pod względem formy, to jednak implementacja systemu regułowego jest zadaniem żmudnym i pracochłonnym. Powstają dwa główne problemy. Pierwszy z nich związany jest z trudnością w uzyskaniu precyzyjnej specyfikacji wiedzy. Drugi, to problem weryfikacji i walidacji wiedzy. Weryfikacja i walidacja sprowadza się do sprawdzenia określonych charakterystyk tworzonego systemu oraz jego poprawności. Zadanie to jest na tyle złożone, że w większości przypadków jest najbardziej kosztowną fazą wdrażania systemu.

Jak dotąd, proponowano różne metodologie oraz narzędzia usprawniające proces projektowania [3]. Najlepsze ze znanych rozwiązań to systemy (metodologie) KADS oraz CommonKADS [5]. Narzędzia te wspierają poszczególne etapy projektowania koncepcyjnego. Takie podejście umożliwia projektowanie złożonych systemów, jednak wsparcie dla fazy implementacji często ogranicza się jedynie do prostego sprawdzania poprawności składniowej. Aby zapewnić bezpieczeństwo i wydajność systemu regułowego należy przeprowadzić analizę wybranych własności utworzonych reguł [1, 2, 12, 13, 14]. Do istotnych własności należy zaliczyć zupełność, spójność oraz determinizm. Należy przy tym zwrócić uwagę, że analiza tych własności (po zaprojektowaniu reguł) pociąga za sobą olbrzymi nakład pracy, związany z ewentualnym wykryciem i usuwaniem błędów i braków w projekcie bazy wiedzy.

Niniejsza praca przedstawia kontynuację badań opisanych w [10] proponując następujące najważniejsze rozszerzenia:

- wprowadzono koncepcję modułowej reprezentacji wiedzy w oparciu rekonfigurowalne moduły w formie tabel,
- zaproponowano bardziej ogólną strukturę powiązań pomiędzy modułami, tzn. w formie grafu zamiast – jak poprzednio – drzewa,
- dopuszczono bardziej złożone wartości atrybutów (w szczególności mogą nimi być zbiory, przedziały oraz elementy tworzące strukturę kraty),
- zaproponowano koncepcję definiowania poprawności projektowanej bazy wiedzy w formie logicznych ograniczeń lokalnych, nakładanych na moduły i powiązania pomiędzy nimi, a realizowanych za pomocą środków algebraicznych.

Praca dotyczy zarówno wspomagania projektowania, jak i weryfikacji systemów regułowych, które można opisać za pomocą zaproponowanej tu koncepcji *Modu-*

larnego Systemu Tabelowego (MTS)[1]. Systemy takie to te, które funkcjonują na bazie rozszerzonych atrybutowych tablic decyzyjnych. Mogą one operować na faktach reprezentowanych jako atrybuty w relacji z ich wartościami. Przy czym, wartości atrybutów mogą mieć charakter nieatomiczny tzn. mogą to być również przedziały, zbiory, czy też elementy zbiorów. Modularny System Tabelowy składa się z pewnej liczby określonych tabel (modułów wiedzy). Każda tabela opisuje zestaw reguł działający w określonym kontekście. W ramach tabeli wszystkie reguły zbudowane są przy użyciu tych samych atrybutów (co implikuje identyczny schemat reguł). Połączenia pomiędzy tabelami określają możliwe ścieżki wnioskowania. Atrybuty mogą być porównywane z wartościami nieatomicznymi, włączając w to podzbiory oraz zakresy. Model MTS można zastosować dla wielu atrybutowych systemów regułowych. Stanowi on uniwersalną metodologię strukturalnej reprezentacji wiedzy.

Zaproponowane w niniejszej pracy podejście bazuje na koncepcji podobnej do narzędzi wspomagających projektowanie (CAD/CASE) oraz graficznej i algebraicznej reprezentacji wiedzy. Pozwala to rozwiązać problem akwizycji wiedzy poprzez systematyczne projektowanie reguł z użyciem modułowej i hierarchicznej struktury, której dostarcza MTS. Co więcej proponowane rozwiązanie umożliwia weryfikację własności projektowanych reguł takich jak zupełność, spójność i determinizm. Prowadzi to do przesunięcia weryfikacji reguł do fazy ich projektowania. Pozwala to na stopniowe udoskonalanie projektu systemu zamiast standardowego cyklu projektowanie-weryfikacja-modyfikacja.

Prototyp graficzno-algebraicznego narzędzia CAD/CASE ogólnego zastosowania, wspierającego projektowanie reguł i umożliwiającego weryfikację został zaimplementowany i jest również przedstawiony w niniejszej pracy. Narzędzie to używa nowej metodologii tworzenia reguł bazującej na koncepcji MTS o roboczej nazwie *drzewa tabelowe* oraz *grafy tabelowe* (inne nazwy to: tab-trees, tabular-trees, tree-tables; graph-tables). W dalszej części opisana jest koncepcja drzew tabelowych oraz zaprezentowany jest system *Osiris*. Jego aktualna wersja rozwojowa stanowi ilustrację koncepcji wspomagania budowy modułowej struktury bazy wiedzy z inkrementalną weryfikacją wybranych własności. Prezentowany system stanowi więc narzędzie o charakterze generycznym, a z uwagi na zastosowane rozwiązania może być łatwo dostosowany do szczegółowych wymagań projektowanego systemu regułowego.

2. Podstawy inżynierii wiedzy z wykorzystaniem narzędzi CAD/CASE

Projektowanie bazy wiedzy dla systemów regułowych sprowadza się do bezpośredniego użycia graficznych i algebraicznych narzędzi CASE. Narzędzia takie mogą wspomagać proces projektowania oraz pozyskiwania wiedzy udostępniając odpowiednie metody jej reprezentacji, bazujące na programowaniu graficznym. Warstwa logiczna, w oparciu o którą dokonywana jest specyfikacja wymaganych własności nie jest bezpośrednio widoczna dla użytkownika.

[1]MTS: Modular Tabular System – (ang.) Modularny System Tabelowy

W niniejszej pracy przedstawiony jest system CAD/CASE wspierający zarówno proces projektowania reguł jak również umożliwiający weryfikację. Narzędzie zostało zbudowane bazując na czterech nowych propozycjach z dziedziny systemów regułowych:

- moduły FBM [2] – są to podstawowe moduły logiczne, specyfikujące zestawy podobnych reguł,
- elementy IBP [3] – reprezentują bezpośrednio ścieżki wnioskowania systemu regułowego jako połączenia pomiędzy modułami FBM, które tworzą
- Tab-Trees lub Tab-Graphs, które reprezentują strukturę MTS oraz
- proste modele algebraiczne określające specyfikację logiki reguł w formie tabel decyzyjnych.

FBM umożliwia projektowanie złożonych systemów, reprezentując nawet bardzo skomplikowane reguły (operujące na wielu atrybutach). Ponadto moduł może być wykorzystany wielokrotnie (ponowne użycie wiedzy), połączony z innymi modułami, lub o ile będzie to konieczne – rozdzielony na kilka mniejszych modułów.

Bezpośrednie graficzne odwzorowanie IBP powoduje, że projektowanie złożonych reguł staje się prostsze i bardziej przejrzyste. Aby zdefiniować regułę użytkownik łączy odpowiednie moduły FBM. Zarówno wejście, wyjście jak i własności reguł są natychmiast widoczne i łatwe do zinterpretowania, a graficzny interfejs czyni projektowanie transparentnym.

Drzewa i grafy tablicowe (Tab-Trees) stanowią nowe graficzne narzędzie do reprezentacji wiedzy. Drzewo (graf) składa się z wielu modułów FBM połączonych za pomocą ścieżek IBP. Drzewa (grafy) tablicowe łączą w sobie zalety drzew decyzyjnych, atrybutowych tabel decyzyjnych oraz tzw. ψ-drzew [6, 7]. Więcej informacji na ten temat można znaleźć w [16].

Model algebraiczny w formie systemu tabelowego pozwala na efektywną weryfikację własności systemu. Podobieństwo do tabel relacyjnych baz danych i atrybutowych tabel decyzyjnych powoduje, że projektowanie i analiza stają się bardzo przejrzyste. Szczegóły techniczne odnośnie weryfikacji w systemach tabelowych można znaleźć w pracy [9].

3. Podstawy teoretyczne

3.1. Systemy tabelowe: projektowanie i weryfikacja

Jedną z głównych idei usprawniających projektowanie systemów regułowych jest przesunięcie weryfikacji do fazy projektowania, jak również wspomaganie samego projektowania za pomocą wygodnego i intuicyjnego graficznego środowiska programowania typu CAD/CASE. Aktualne badania są rozwinięciem tez przedstawionych w

[2] FBM: Flexible Branching Modules
[3] IBP: Inference Branching Paths

[6, 7], gdzie zaproponowano koncepcję tzw. ψ-drzewa jako sposób reprezentacji wiedzy, wspomagający proces logicznego projektowania zupełnych i deterministycznych systemów regułowych. W badaniach wykorzystane zostały również propozycje odnośnie systemów tabelowych [8, 9], które prezentowały podejście zbliżone do atrybutowych tabel decyzyjnych.

Modularny System Tabelowy jest zbliżony do modelu relacyjnych baz danych (RDB) i posiada podobne zalety. System tablicowy może określać fakty albo reguły; w tym drugim przypadku część atrybutów odpowiada warunkom reguł, podczas gdy reszta odpowiada hipotezom i decyzjom. Takie systemy można tworzyć z wykorzystaniem struktury RDB, ale w tym przypadku wszystkie wartości atrybutów muszą być atomiczne. W przypadku MTS dopuszczalne są wartości nieatomiczne; mogą to być podzbiory, przedziały itp. System tabelowy może składać się z jednej lub więcej tabel połączonych w strukturę.

3.2. Kheops

Początkowo projekt *Osiris* był zorientowany jako narzędzie CAD/CASE do wspomagania tworzenia reguł dla systemu *Kheops*. *Kheops* [4] jest zaawansowanym systemem regułowym czasu rzeczywistego. Bazuje na metodologii wnioskowania w przód, oraz charakteryzuje się deterministycznym czasem reakcji (poniżej 15 milisekund) co umożliwia jego zastosowanie jako system czasu rzeczywistego. Szczegółowe informacje na temat Kheopsa można znaleźć w [4].

4. Własności systemów regułowych

4.1. Pochłanianie reguł

Rozważmy ogólny przypadek pochłaniania reguł; niektóre definicje znajdują się w [1, 12, 13] oraz z uwzględnieniem systemów tabelowych w [8, 9]. Reguła pochłania inną regułę jeżeli zachodzą warunki: warunek pierwszej reguły jest *słabszy* (bardziej ogólny) niż warunek reguły pochłanianej, wniosek pierwszej reguły jest *silniejszy* (bardziej szczegółowy) niż wniosek reguły pochłanianej.

Niech reguły $R:\ \phi \longrightarrow h$ i $R':\ \phi' \longrightarrow h'$, spełniają: $\phi' \models \phi$ i $h \models h'$. Reguła pochłaniana R' może zostać usunięta zgodnie z poniższym schematem:

$$\frac{\begin{array}{rccc} R: & \phi & \longrightarrow & h \\ R': & \phi' & \longrightarrow & h' \end{array}}{\begin{array}{rccc} R: & \phi & \longrightarrow & h \end{array}}$$

Intuicyjnie reguła pochłaniana może być usunięta ponieważ jej skutkiem są słabsze rezultaty i potrzebuje do zadziałania silniejszych warunków. Zatem ten sam rezultat może zostać osiągnięty z użyciem reguły pochłaniającej. Używając notacji tabelowej mamy:

reguła	A_1	A_2	$\dots$	A_j	$\dots$	A_n	H
R	t_1	t_2	$\dots$	t_j	$\dots$	t_n	h
R'	t'_1	t'_2	$\dots$	t'_j	$\dots$	t'_n	h'

reguła	A_1	A_2	$\dots$	A_j	$\dots$	A_n	H
R	t_1	t_2	$\dots$	t_j	$\dots$	t_n	h

Warunkiem pochłaniania, w przypadku powyższych tabel jest $t'_j \subseteq t_j$, dla $j = 1, 2, \dots, n$ i $h' \subseteq h$ (w większości przypadków $h' = h$ są wartościami atomicznymi). W aktualnej wersji systemu *Osiris* pochłanianie się reguł jest automatycznie eliminowane dzięki podziałowi wartości atrybutów na rozłączne przedziały.

4.2. Determinizm

Zbiór reguł jest *deterministyczny* jeżeli żadne dwie reguły nie są spełnione dla tego samego stanu. Zbiór reguł, który nie jest deterministyczny jest również określany jako *ambiwalentny* [12]. Determinizm można uzyskać eliminując nakładanie się (nawet częściowe) warunków stosowalności reguł.

Z czysto logicznego punktu widzenia system jest deterministyczny jeżeli prewarunki $\phi \wedge \phi'$ nie są nigdy spełnione łącznie. Obliczenie $\phi \wedge \phi'$ jest następujące: dla każdego atrybutu A_j jest wartość atomiczna postaci: $A_j = t_j$ w ϕ i $A_j = t'_j$ w ϕ', $i = 1, 2, \dots, n$. Jeżeli przecięcie t i t' dla pewnego j (przynajmniej jednego) jest puste (np. dwie różne wartości; bardziej ogólnie $t_j \cap t'_j = \emptyset$), wtedy reguły są rozłączne. W aktualnej wersji *Osirisa* determinizm może zostać osiągnięty za pomocą podziału dziedzin atrybutów na niepokrywające się przedziały (lub zbiory).

4.3. Zupełność systemu regułowego

Intuicyjnie system regułowy jest *zupełny* jeżeli dla dowolnej, przewidzianej sytuacji wejściowej istnieje przynajmniej jedna reguła o spełnionych warunkach stosowalności. Istnieje kilka możliwości aby zdefiniować zupełność [1, 12, 13], a w przypadku systemów tabelowych [8, 9]. Logiczna zupełność jest rozważana dla zbioru reguł jako:

$$\begin{array}{llll} R_1: & \phi_1 & \longrightarrow & h_1 \\ R_2: & \phi_2 & \longrightarrow & h_2 \\ \vdots & \vdots & \vdots & \vdots \\ R_m: & \phi_m & \longrightarrow & h_m \end{array}$$

Powyższa propozycja pochodzi z czysto logicznej analizy [8, 9] i nie wymaga dodatkowego ekstensjonalnego wyliczania możliwych przypadków. Algebraiczny sposób przetwarzania wiedzy przedstawiony jest poniżej [9].

Rozważmy złączenie rozłącznych formuł warunków stosowalności reguły w postaci: $\Phi = \phi_1 \vee \phi_2 \vee \dots \phi_m$. Warunkiem logicznej zupełności powyższego systemu jest:

$$\models \Phi \tag{2}$$

co oznacza, że Φ jest tautologią. W proponowanym rozwiązaniu, zamiast dowodów logicznych użyta została metoda algebraiczna oparta na podziale dziedziny atrybutów [8, 9]. W obecnej wersji system *Osiris* umożliwia sprawdzanie zupełności reguł. Wyszukiwanie brakujących warunków reguł realizowane jest automatycznie i dostępne w każdym momencie projektowania systemu. Dzięki temu niezupełność systemu może być eliminowana etapami już w fazie projektowania.

5. Graficzna reprezentacja

Reprezentacja i pozyskiwanie wiedzy są wciąż otwartymi tematami w dziedzinie inżynierii wiedzy. Poniżej przedstawiono proponowane, nowe, uniwersalne podejście bazujące na Modularnych Systemach Tabelowych. Koncepcja MTS udostępnia przejrzysty i efektywny mechanizm reprezentacji struktur logicznych.

5.1. Tabele decyzyjne

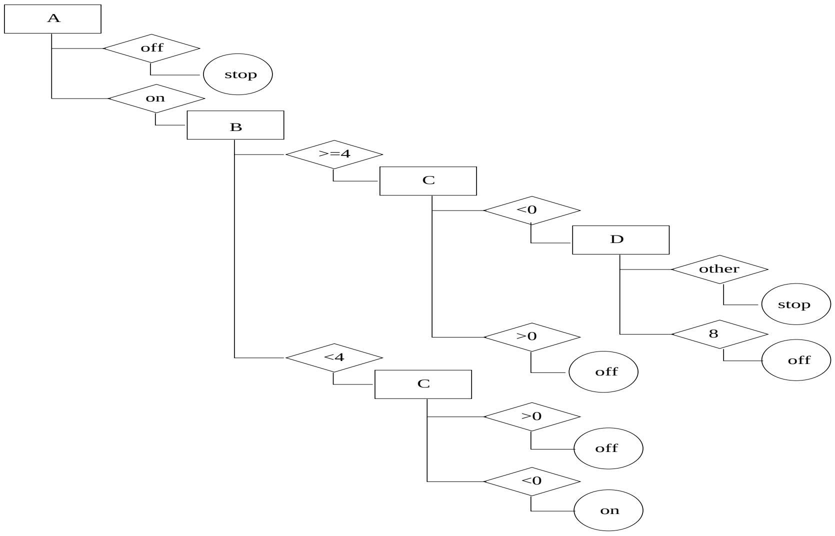
Tabele decyzyjne są sposobem reprezentacji reguł. Zarówno warunki jak i akcje są umieszczone w tabeli. Klasyczna tabela decyzyjna używa logiki rachunku zdań, rozszerzonej o wartość "nieistotne". Główną zaletą tabel decyzyjnych jest ich prostota oraz intuicyjna interpretacja. Główną wadą natomiast jest ograniczenie do logiki rachunku zdań. W niektórych przypadkach bardziej odpowiednie jest użycie wartości atrybutów. Rozszerzonymi tabelami decyzyjnymi są tabele OAV: Object-Attribute-Value. Przykład takiej tabeli pokazany jest w na rysunku 1. Wiersze określają dla jakich wartości atrybutów realizowane są określone akcje. Zarówno `v_ij` jak i `w_kl` mogą przyjmować różne wartości, nie tylko *prawda*, *fałsz*, *nieistotne*.

attrib_1	attrib_2	...	action_1	action_2	...
v_11	v_12	...	w_11	w_12	...
v_21	v_22	...	w_21	w_22	...
⋮	⋮	⋱	⋮	⋮	⋱

Rysunek 1: Tabela OAV

5.2. Drzewa i grafy decyzyjne

Reprezentacje drzewiaste są łatwe do zrozumienia. Korzeniem drzewa jest węzeł podstawowy. Od każdego węzła, na kształt gałęzi, rozchodzą się połączenia. Wybór połączenia zależy od warunku przyporządkowanego do węzła. Obliczenie warunku określa wybór połączenia. Drzewo jest przemierzane w kierunku z góry na dół, liście określają końcowe decyzje lub akcje. Przykład drzewa decyzyjnego pokazany jest na rysunku 2. Okręgi reprezentują akcje, prostokąty symbolizują atrybuty, a równoległoboki określają relacje i wartości. Istnieją również drzewa decyzyjne zwane ψ-drzewami [6, 7]; w takim przypadku pojedynczy węzeł może mieć więcej niż dwa



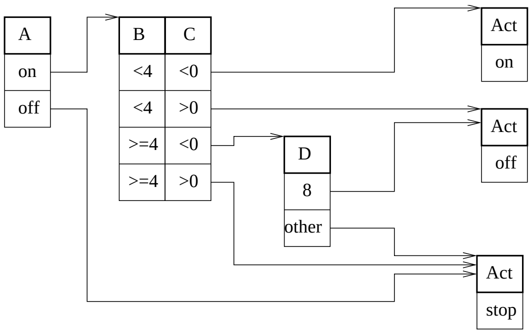
Rysunek 2: Przykład drzewa decyzyjnego

połączenia, co lepiej odzwierciedla proces podejmowania decyzji i pozwala porównywać atrybuty z wieloma różnymi wartościami w tym samym węźle. Struktura drzewa może być zastąpiona grafem skierowanym, co pozwala wielokrotnie wykorzystać wybrane moduły FBM.

5.3. Nowe podejście: reprezentacja drzewiasto-tabelowa i grafowo-tabelowa

Reprezentacja drzewiasto-tabelowa jest zbudowana jako hierarchia tabel FBM [6, 7]. Hierarchia ta jest reprezentowana za pomocą ψ-drzewa. Każdy wiersz tabeli FBM jest połączony z prawej strony z inną tabelą FBM. Takie połączenie stanowi równoważnik koniunkcji logicznej pomiędzy odpowiednimi wierszami tabel. Tabele FBM mogą być łatwo dzielone pionowo na mniejsze komponenty bądź łączone – stanowi to ich podstawową zaletę, umożliwia bowiem ponowne wykorzystanie już zdefiniowanych modułów. Złaczenie wyjść modułów FBM prowadzi do uzyskania oszczędniejszej reprezentacji grafowo-tabelowej.

Moduły FBM używane w reprezentacji drzewiasto- lub grafowo-tabelowej podzielone są na dwa rodzaje: tabele atrybutów oraz tabele akcji. Tabele atrybutów są częścią atrybutową klasycznej tabeli OAV, natomiast tabele akcji są częścią akcyjną. Podczas gdy tabele atrybutowe mogą mieć dowolną liczbę wierszy (liczba kolumn zależy od liczby użytych atrybutów), tabele akcji mogą posiadać tylko jeden wiersz, oznacza to, że określona akcja (lub zbiór akcji, o ile w tabeli jest więcej niż jedna kolumna) może posiadać ustawioną tylko jedną wartość. Przykład reprezentacji drzewiasto-tabelowej jest pokazany na rysunku 3. Głównymi jej zaletami są: prostota, hierarchiczna struktura, efektywny sposób wizualizacji z dużym zagęszczeniem informacji, zalety tabel decyzyjnych, analogie do RDB.



Rysunek 3: Przykład reprezentacji drzewiasto-tabelowej

Obecnie, w stosunku do pierwotnego projektu (por. [10, 16]) reprezentacja drzewiasto-tabelowa została rozszerzona. Dotychczasową drzewiastą strukturę powiązań pomiędzy modułami zastąpiono grafem. Umożliwia to m.in. wykorzystanie tych samych modułów wiedzy na różnych ścieżkach wnioskowania (np. wielokrotne użycie modułów akcji `Act` na rys. 3). Modyfikacja ta w istotny sposób zmniejsza nakład pracy potrzebny do wykonania projektu systemu regułowego.

6. System Osiris

Głównym celem systemu Osiris było stworzenie interfejsu do wspomagania projektowania reguł dla systemów regułowych. Osiris jest tak zaprojektowany aby być środowiskiem uniwersalnym, niezależnym od systemu regułowego. Najistotniejsze komponenty systemu obejmują:

- graficzny edytor reguł bazujący na logice,
- MTS jako główną metodę projektowania,
- podsystem kontroli poprawności reguł (sprawdzanie zupełności),
- generator kodu dla systemu regułowego Kheops,
- moduł integracji z graficznym interfejsem użytkownika gKheops.

Osiris [16, 10] jest systemem wielomodułowym zaprojektowanym do pracy w środowisku Unix (testowany dla: Debian GNU/Linux i Solaris). Jego architektura zapewnia możliwość rozbudowy i modyfikacji (dla generacji kodu dla różnych systemów). Głównymi modułami systemu są: *Editor* oraz *Controller*. Pozostałe moduły mogą być dołączana w trakcie działania systemu. Czytelna wizualizacja tworzonych struktur logicznych jest możliwa dzięki graficznej reprezentacji drzewiasto-tabelowej. Należy zauważyć, że moduły do generacji kodu, weryfikacji jak również środowisko uruchomieniowe (odpowiednio Generator, Validator, gKheops) są oddzielnymi aplikacjami. Generator i Validator korzystają z teorii kompilatorów przetwarzając plik danych opisujących tworzone reguły; generują one kod dla systemu Kheops oraz umożliwiają sprawdzanie zupełności. Wszystkie moduły są zarządzane poprzez graficzny interfejs. Warto zauważyć, że moduł Controller może, w razie potrzeby, uruchomić dowolną aplikację do celów generacji kodu, bądź też sprawdzenia poprawności tworzonych reguł. Taka modularność umożliwia użycie Osirisa jako narzędzia uniwersalnego i niezależnego od systemu regułowego.

Do przechowywania projektowanych reguł Osiris używa pliku danych w formacie dokumentu XML. Format ten posiada dobrze zdefiniowaną gramatykę, co ułatwia generację kodu dla dowolnego systemu regułowego oraz ewentualną implementację, nawet złożonych algorytmów kontrolujących spójność reguł.

Jedną z najbardziej istotnych cech systemu Osiris jest fakt, że udostępnia on mechanizmy sprawdzania poprawności tworzonych reguł (moduł Validator). Przenosi to proces weryfikacji do fazy projektowania. Validator jest odrębną aplikacją, zatem ewentualne modyfikacje algorytmów kontroli poprawności reguł mogą być dokonane bez konieczności zmian w innych modułach systemu. Jako środowisko uruchomieniowe, kontrolujące wykonanie reguł, został wybrany gKheops [11, 10]. Wymaga podkreślenia, że środowiskiem uruchomieniowym może być dowolna aplikacja (system regułowy) kontrolująca proces odpalania reguł.

7. Podsumowanie

W pracy zaprezentowano koncepcję narzędzia CAD/CASE pracującego w środowisku graficznym, które wspiera proces projektowania systemów regułowych bazując na metodologii MTS. Głównym celem proponowanego rozwiązania jest stworzenie intuicyjnego, przyjaznego dla użytkownika środowiska, które usprawniałoby procesy pozyskiwania, projektowania i weryfikacji wiedzy. W aktualnej fazie prototypu system umożliwia sprawdzanie zupełności projektowanych reguł, natomiast problemy determinizmu, i pochłaniania reguł rozwiązane są na poziomie reprezentacji wiedzy.

System Osiris bazuje na nowej reprezentacji wiedzy tzw. reprezentacji drzewiasto-tabelowej lub grafowo-tabelowej, która zbudowana jest z tabel FBM połączonych za pomocą modułów IBP. Jest to zbiór narzędzi ogólnego zastosowania o generycznym charakterze, przeznaczonych głównie dla ludzi zaznajomionych z relacyjnymi bazami danych i tabelami decyzyjnymi. Należy zauważyć, że istnieje niewiele narzędzi wspierających w podobny sposób proces projektowania reguł. Połączone systemy Osiris i gKheops mogą potencjalnie znaleźć zastosowanie, jako uniwersalne i rozwojowe środowisko CASE dla wielu różnych systemów regułowych.

Bibliografia

[1] Ed P. Andert. Integrated knowledge-based system design and validation for solving problems in uncertain environments. *Int. J. of Man-Machine Studies*, 36:357–373, 1992.

[2] F. Coenen. Validation and verification of knowledge based systems: Report on eurovav'99. *Knowledge Engineering Review 15:2*, pages 187–196, 2000.

[3] R. de Hoog. *[5]*, chapter Methodologies for Building Knowledge-Based Systems: Achievements and Prospectus.

[4] Jean-Paul Gouyon. Kheops users's guide. *Report of Laboratoire d'Automatique et d'Analyse des Systemes*, (92503), 1994.

[5] J. Liebowitz. *The Handbook of Applied Expert Systems.* CRC Press, Boca Raton, 1998.

[6] A. Ligęza. Towards design of complete rule-based control systems. In R. Karba J. Kocijan, editor, *IFAC/IMACS International Workshop on Artificial Intelligence in Real-Time Control*, pages 189–194. IFAC, Bled, Slovenia, 1995.

[7] A. Ligęza. Logical support for design of rule-based systems. reliability and quality issues. In M.C. Rousset, editor, *ECAI-96 Workshop on Validation, Verification and Refinment of Knowledge-based Systems*, volume W2, pages 28–34. ECAI'96, Budapest, 1996.

[8] A. Ligęza. Towards logical analysis of tabular rule-based systems. *In [15]*, pages 30–35, 1998.

[9] A. Ligęza. Toward logical analysis of tabular rule-based systems. *International Journal of Intelligent Systems*, 16:333–360, 2001.

[10] Antoni Ligęza, Igor Wojnicki, and Grzegorz J. Nalepa. Design and implementation support of rule-based systems (in polish). In K. Zieliński, editor, *II Polish National Conference on Software Engineering*, pages 229–236. AGH, Zakopane/Cracow, 2000.

[11] Grzegorz Nalepa. Graphical user interface to kheops rule-based expert system. Master's thesis, Academy of Mining and Metallurgy, 1999.

[12] Alun D. Preece. A new approach to detecting missing knowledge in expert system rule bases. *Int. J. of Man-Machine Studies*, 38:161–181, 1993.

[13] A. Vermesan. *A chapter in [5]*, chapter Foundation and Application of Expert System Verification and Validation.

[14] A. Vermesan and F. Coenen (Eds.). *Validation and Verification of Knowledge Based Systems – Theory, Tools and Practice.* Kluwer Academic Publishers, Boston, 1999.

[15] R. R. (ed.) Wagner. Database and expert systems applications. In *Proceedings of the Ninth International Workshop, Vienna; IEEE Computer Society*, Los Alamitos, CA, 1998.

[16] Igor Wojnicki. Design and implementation of a graphical user interface for computer aided logical design of kheops knowledge based system. Master's thesis, Academy of Mining and Metallurgy, 2000.