STUDIA INFORMATICA 2011
Volume 32 Number 2A (96)

Roman SIMIŃSKI, Sebastian JANUS
Uniwersytet Śląski, Instytut Informatyki

WIZUALIZACJA WNIOSKOWANIA W REGUŁOWYCH BAZACH WIEDZY Z WYKORZYSTANIEM SIECI PETRIEGO

Streszczenie. Artykuł przedstawia koncepcję wykorzystania sieci Petriego do zadania wizualizacji struktury regułowej bazy wiedzy oraz prezentuje funkcjonalne oraz implementacyjne aspekty systemu, wykorzystującego tę koncepcję do wizualizacji struktury regułowej bazy wiedzy oraz procesu wnioskowania dla takich baz. W artykule przedstawiono podstawowe zagadnienia związane z sieciami Petriego oraz wykorzystanie sieci do modelowania procesów wnioskowania oraz krótki opis realizacji systemu wykorzystującego tę koncepcję modelowania.

Słowa kluczowe: regułowe bazy wiedzy, wnioskowanie, sieć Petriego

VISUALISATION OF INFERENCE IN RULE KNOWLEDGE BASES USING PETRI NETS

Summary. The paper presents conception of utilization of Petri nets to visualization the structure of rule knowledge bases and the functional and implementation issues of system using this conception for visualization both the structure of knowledge bases and the inference process in such bases. The basic terms of Petri nets as well as the idea of using Petri nets as the modeling rule knowledge base have been presented in this paper. The methods of using Petri nets for modeling of the inference process has been also discussed in this paper together with description of realization features of proper software.

Keywords: rule knowledge bases, inference, Petri nets

1. Wprowadzenie – motywacja i cel artykułu

Systemy opierające swoje działanie na bazach wiedzy i algorytmach wnioskowania są od wielu lat stosowane jako skuteczne narzędzia wspomagania procesów decyzyjnych w dzie-

dzinach, w których istnieje dobrze określona wiedza, stanowiąca podstawę realizacji baz wiedzy. W ciągu ostatniej dekady można zaobserwować ponowny wzrost zainteresowania systemami z regułową bazą wiedzy, niewątpliwie jest on wynikiem rozwoju metod eksploracji danych. To właśnie generowane automatycznie zbiory reguł są najczęściej wynikiem eksploracji danych.

Podstawowymi zaletami reprezentacji regułowej są jej prostota i modularność [6]. Regułowe bazy wiedzy mogą być rozwijane przyrostowo, baza wiedzy może być stopniowo rozszerzana w miarę pozyskiwania wiedzy dziedzinowej. Poszukiwanie rozwiązania problemów stawianych przed systemami z bazą wiedzy odbywa się z wykorzystaniem wnioskowania, polegającego na wykorzystaniu odpowiednich reguł z bazy wiedzy, zgodnie z posiadanymi informacjami, opisującymi problem. Z racji zwykle jawnej reprezentacji wiedzy w bazie, otrzymujemy możliwość obserwowania nie tylko wyników działania systemu, ale również sposobu ich uzyskiwania — ta właściwość systemów z bazą wiedzy określana jest jako zdolność do udzielania wyjaśnień wyniku i sposobu rozwiązania problemu [1, 6].

Niestety w przypadku baz wiedzy liczących setki czy tysiące reguł możliwość weryfikacji wyników działania systemu z bazą wiedzy jest wyraźnie utrudniona. Klasyczne wyjaśnienia *jak*? oraz *dlaczego*? w przypadku wnioskowania na dużych i złożonych bazach wiedzy nie dostarczają wystarczająco użytecznych informacji o jego przebiegu, a analiza przebiegu wnioskowania jest istotnym elementem inżynierii wiedzy. Paradoksalnie, zalety reprezentacji regułowej — prostota i modularność — stają się problemem w przypadku dużych baz. Powiązania i zależności pomiędzy regułami są trudne do wychwycenia i mogą być przyczyną powstania anomalii, prowadzących do błędnego funkcjonowania systemu.

Problematyce wykrywania anomalii w regułowych bazach wiedzy poświęcono wiele prac [3]. W pracach [9, 12, 13] przedstawiono podejście bazujące na jednostkach decyzyjnych, ukierunkowane na statyczną analizę zawartości bazy wiedzy. Jednak wiele anomalii prowadzi do rzeczywistych problemów dopiero na etapie wnioskowania, co prowadzi do koncepcji dynamicznej weryfikacji baz regułowych. W pracach [9, 10, 11] zaproponowano powiązanie koncepcji jednostek decyzyjnych oraz sieci Petriego, w celu uzyskania spójnego i formalnego podejścia do dynamicznej analizy właściwości regułowej bazy wiedzy. Podejście to jest rozwijane i aktualnie trwają prace teoretyczne, a także implementacyjne poświęcone temu zagadnieniu. Niniejszy artykuł jest jednym z elementów prowadzonych prac.

Celem artykułu była realizacja systemu pozwalającego na wizualizację procesów wnioskowania w regułowych bazach wiedzy z wykorzystaniem aparatu oferowanego przez sieci Petriego. W przeciwieństwie do wspomnianych wcześniej prac nie przewidywano zastosowania jednostek decyzyjnych [8], a tworzone sieci Petriego stanowić miały bezpośredni model regułowej bazy wiedzy. Artykuł ma charakter wybitnie praktyczny, celem było zaprojektowa-

nie i implementacja systemu wspomagającego pracę inżyniera wiedzy, poprzez modelowanie regułowych baz wiedzy za pomocą sieci Petriego oraz wizualizację procesów wnioskowania.

2. Regułowa baza wiedzy

Regułowa reprezentacja w bazach wiedzy ma zalety, a także ograniczenia. Wśród zalet wskazuje się naturalność i czytelność dla człowieka oraz jednoczesną łatwości formalizacji i przetwarzania komputerowego; modularność i łatwość przyrostowej rozbudowy. Jak zasygnalizowano we wprowadzeniu, te zalety stają się czasem wadą reprezentacji regułowej – łatwość przyrostowej rozbudowy bazy o stosunkowo niezależne elementy, jakimi są reguły często powoduje powstawanie w bazach błędów i anomalii [3, 11, 12]. Potrzebne są narzędzia wspomagające, wizualizujące strukturę bazy i przebieg wnioskowania w sposób graficzny. Realizacja takiego narzędzia była celem niniejszego artykułu. Powstała ona przy następujących założeniach odnośnie bazy wiedzy – zakłada się, że reguły są zapisane w postaci klauzul Horna, zapisanych z wykorzystaniem implikacji; baza wiedzy R jest zbiorem m reguł:

$$R = \{ r_1, r_2, \ldots, r_m \} \tag{1}$$

Każda reguła r ma postać:

$$r\colon l_1 \wedge l_2 \wedge \ldots \wedge l_n \rightarrow c, \tag{2}$$

gdzie:

$l_1, l_2, \ldots, l_n$ — literały będące warunkami reguły r,

c — literał będący konkluzją reguły r.

Niech A będzie niepustym zbiorem atrybutów warunkowych i decyzyjnych, dla każdego $a \in A$ definuje się zbiór wartości V_a:

$$A = \{ a_1, a_2, \ldots, a_m \} \tag{3}$$

$$V_a = \{ v_1^a, v_2^a, \ldots, v_k^a \} \tag{4}$$

Literały reguł zapisywane będą w postaci dwójek atrybut wartość (a, v_i^a), gdzie $a \in A$, $v_i^a \in V_a$, zapis ten będzie równoważny formie $a = v_i^a$. Każda reguła $r \in R$ będzie mogła mieć ogólną postać:

$$r\colon (a_1, v_1^{a1}) \wedge (a_2, v_1^{a2}) \wedge \ldots \wedge (a_n, v_1^{an}) \rightarrow (a_k, v_1^{ak}) \tag{5}$$

lub postać równoważną, wykorzystującą symbol '=':

$$r\colon a_1 = v_1^{a1} \wedge a_2 = v_1^{a2} \wedge \ldots \wedge a_n = v_1^{an} \rightarrow a_k, v_1^{ak} \tag{6}$$

Wnioskowanie na regułowych bazach wiedzy realizuje się głównie z wykorzystaniem dwóch podstawowych metod — wnioskowania *progresywnego*, zwanego inaczej wnioskowaniem *w przód* oraz wnioskowania *regresywnego*, zwanego inaczej wnioskowaniem *wstecz*, obie te metody są również łączone w algorytmie wnioskowania *mieszanego*.

Wnioskowanie w przód polega na kolejnym uaktywnianiu reguł spełnionych, a więc takich, których przesłanki są w zbiorze faktów. Uaktywnienie reguły powoduje dopisanie nowego faktu, co może spowodować, że spełniona (a potem uaktywniona) może zostać kolejna reguła. Proces ten będzie trwał tak długo, dopóki w bazie wiedzy da się odnaleźć reguły spełnione. Wnioskowanie w przód nie może odbyć się bez faktów. Mówi się, że jest ono *sterowane faktami* (ang. *data driven*).

Wnioskowanie wstecz polega na potwierdzeniu prawdziwości postawionej *hipotezy*, zwanej celem wnioskowania. Hipoteza jest potwierdzona wtedy, gdy istnieje reguła, której przesłanki są w bazie faktów, a konkluzja zgodna jest z hipotezą. Ustalenie prawdziwości przesłanek może powodować konieczność uaktywnienia wielu reguł. Wnioskowanie wstecz nie może odbyć się bez ustalonej hipotezy, stanowiącej cel wnioskowania. Mówi się, że jest ono *sterowane celem* (ang. *goal driven*).

Ze względu na ograniczone rozmiary niniejszego artykułu, szczegółowe przedstawienie algorytmów w ich wersjach klasycznych nie będzie możliwe. Są one jednak dobrze znane i opisywane w wielu publikacjach, np. [1, 4, 6].

3. Sieci Petriego a regułowa baza wiedzy

Podstawową formą wizualizacji struktury bazy wiedzy ma być sieć Petriego, a zachodzące procesy wnioskowania mają być modelowane poprzez propagację znaczników w takiej sieci. Rozdział ten prezentuje skrót podstawowych informacji, dotyczących sieci Petriego, ograniczony do niezbędnego minimum, koniecznego dla prezentacji dalszej części artykułu, więcej informacji na temat sieci Petriego odnaleźć można w pracach [2, 5, 7, 14, 15].

Sieć Petriego przyjmuje postać dwudzielnego, skierowanego grafu, składającego się z wierzchołków dwojakiego rodzaju. Pierwszy rodzaj wierzchołków to *miejsca* (ang. *places*), reprezentowane graficznie, zwykle poprzez okręgi. Drugi rodzaj wierzchołków to *przejścia*, zwane też *tranzycjami* (ang. *transitions*), zwykle graficznie reprezentowane przez prostokąty. Skierowane łuki łączą zawsze wierzchołki różnych rodzajów – zatem łączą miejsca z tranzycjami oraz tranzycje z miejscami.

W sensie formalnym sieci Petriego bywają definiowane w różny sposób. Jeden z nich mówi, że sieć Petriego N to trójka:

$$N = (P, T, F), \tag{7}$$

gdzie:

- P – skończony zbiór miejsc,
- T – skończony zbiór tranzycji,
- F – relacja przepływu określająca zbiór łuków: $F \subseteq (P \times T) \cup (T \times P)$

oraz:

- $P \cap T = \varnothing$
- $P \cup T \neq \varnothing$

Dla $t \in T$ definiujemy:

- $\cdot t$ – zbiór miejsc wejściowych t: $\cdot t = \{ p \in P \mid (p, t) \in F \}$,
- $t \cdot$ – zbiór miejsc wyjściowych t: $\cdot t = \{ p \in P \mid (t, p) \in F \}$.

Sieć Petriego jest reprezentowana graficznie w postaci grafu:

- miejsca są reprezentowane w postaci okręgów,
- tranzycje w postaci odcinków lub prostokątów,
- żadne dwa miejsca i żadne dwie tranzycje nie są bezpośrednio połączone.

Miejsca i tranzycje opisują strukturę modelowanego systemu. Oczywiście w obrębie konkretnego zastosowania, miejsca i tranzycje sieci oznaczają konkretne składowe modelowanego systemu. Do modelowania dynamiki procesów w takich zachodzących systemach wykorzystuje się metody bazujące na ruchu *znaczników* (ang. *tokens*). Znaczniki są reprezentowane graficznie w postaci kropek umieszczanych w miejscach sieci. Znakowana sieć Petriego N' definiowana jest jako dwójka:

$$N' = (N, M_0), \tag{8}$$

gdzie:

- N – to sieć Petriego,
- M_0 – to znakowanie początkowe.

Znakowanie:

- $M: P \rightarrow \{ 0, 1, 2, \ldots \}$,
- $M(p)$ – liczba znaczników w miejscu p,

Gdy $\forall p \in P : M(p) \in \{ 0, 1 \}$ mówimy, że sieć jest znakowana binarnie. Znakowanie początkowe definiuje się następująco:

$$M_0 = \{ m_1, m_2, m_3, ..., m_n \}, \text{ gdzie } n \text{ – łączna liczba miejsc w sieci.} \tag{9}$$

Dynamiczne właściwości modelowanego systemu są reprezentowane poprzez *aktywowanie* (ang. *firing*) odpowiednich tranzycji. Aby ten proces nastąpił:

- tranzycja t musi być *wzbudzona* (ang. *enabled*): $\forall p \in \cdot t: M(p) > 0$,
- tranzycja t może się *uaktywnić* jedynie jeżeli jest wzbudzona.

Gdy tranzycja t uaktywnia się, znacznik jest usuwany z każdego miejsca wejściowego p tranzycji t oraz tranzycja t wstawia znacznik do każdego swojego miejsca wyjściowego p', powstaje nowe znakowanie m', co można pisać następująco:

$$\forall p \in P : m'(p) = M(p) + O(t, p) - I(p, t), \tag{10}$$

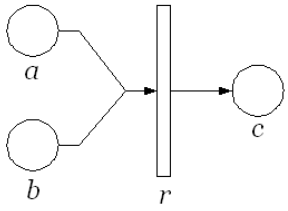
gdzie:

- $O(t, p)$ odwzorowuje $(P \times T) \rightarrow \{ 0, 1 \}$

- $I(p, t)$ odwzorowuje $(T \times P) \rightarrow \{ 0, 1 \}$

Przedstawienie regułowej bazy wiedzy w postaci sieci Petriego okazuje się intuicyjnie proste [9, 10, 11]. Przedstawiony niżej przykład wykorzystuje literały w postaci zmiennych zdaniowych na poziomie rachunku zdań – jest to uproszczenie poczynione dla klarowności dalszych prezentacji[1].

Załóżmy, że dana jest reguła r: $a \wedge b \rightarrow c$**.** Taka reguła r rozpatrywana na gruncie sieci Petriego może być tranzycją, literały warunkowe a i b stanowią miejsca wejściowe tranzycji, konkluzja c stanowi miejsce wyjściowe, co ilustruje rys. 1.



Rys. 1. Reguła r jako tranzycja
Fig. 1. The rule r as the transition

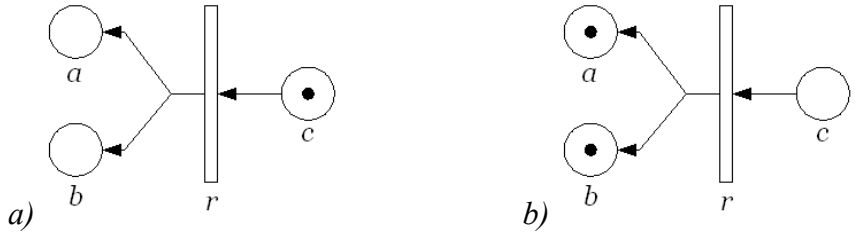
Załóżmy, że zbiór faktów dla wnioskowania w przód dla takiej reguły r zawiera dwa fakty: $\{ a, b \}$. Sytuację taką modelujemy umieszczeniem znaczników w obu miejscach wejściowych reguły r, prezentuje to rys. 2*a*. Zgodnie z przedstawionymi w poprzednim rozdziale informacjami na temat aktywowania tranzycji, możemy stwierdzić, że tranzycja reprezentująca regułę r jest *wzbudzona* i może zostać *aktywowana*. Aktywowanie tranzycji spowoduje usunięcie znaczników z miejsc wejściowych a i b oraz umieszczenie znacznika w miejscu wyjściowym c, co ilustruje rys. 2*b*. Stanowi to pojedynczy krok wnioskowania w przód.

Rys. 2. Aktywowanie reguły jako aktywowanie tranzycji
Fig. 2. The activation of the rule as rule's firing

Przeanalizujmy teraz tę samą regułę, rozpatrywaną na gruncie wnioskowania wstecz. Załóżmy, że celem wnioskowania jest ustalenie prawdziwości literału c oraz że, jak wyżej, istnieje zbiór faktów $\{ a, b \}$. Wnioskowanie wstecz jest sterowane celem i przebiega w odwrotnym kierunku niż wnioskowanie progresywne – możemy dokonać prostego zabiegu, polegającego na odwróceniu kierunku łuków w sieci modelującej rozpatrywaną regułę r. W przypadku wnioskowania wstecz znacznik zostanie umieszczony w tym miejscu sieci,

[1] W systemie opisywanym w dalszej części tego artykułu wykorzystuje się reguły z literałami w formie dwójek atrybut-wartość, przyjęte uproszczenie nie zmienia prezentowanej idei, lecz czyni prezentację klarowną.

które odpowiada założonemu celowi wnioskowania, a więc w miejscu *c* (rys. 3*a*). Zgodnie z regułami wzbudzania i uaktywniania tranzycji, tranzycja *r* jest teraz wzbudzona i może zostać aktywowana. Spowoduje to usunięcie znacznika z miejsca wejściowego *c* oraz umieszczenie znaczników w miejscach wyjściowych *a* i *b* tranzycji *r* (rys. 3*b*).



Rys. 3. Pojedynczy krok wnioskowania wstecz
Fig. 3. An elementary step of backward inference

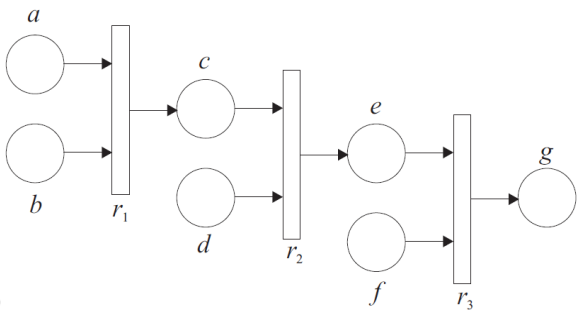
W tym momencie oznakowane miejsca reprezentują te przesłanki reguł, które muszą być spełnione, aby cel *c* mógł zostać potwierdzony. Istotnie, zbiór faktów { *a*, *b* } zawiera oznakowane miejsca *a* i *b*, zatem cel *c* wnioskowania może być potwierdzony.

Pojawienie się znacznika w miejscach sieci, w przypadku wnioskowania w przód i wstecz, interpretowane jest w różny sposób. W przypadku wnioskowania w przód, pojawienie się znacznika w miejscu odpowiadającym konkluzji reguły odpowiada pojawieniu się nowego faktu. Zatem zmiana (rozumiana jako przyrost) w stosunku do poprzedniego znakowania będzie reprezentowała pojawianie się nowych faktów. W przypadku wnioskowania wstecz, oznakowane miejsca wyjściowe tranzycji wskazują, jakie fakty należy dostarczyć, aby proces potwierdzania ustalonego celu wnioskowania mógł zakończyć się sukcesem. Jeżeli zbiór faktów jest wcześniej znany, należy sprawdzić czy oznakowane miejsca znajdują się w zbiorze faktów, co pozwala stwierdzić, że cel wnioskowania został potwierdzony. Jeżeli aktualnie wnioskowanie odbywa się bez z góry określonego zbioru faktów, a system jest interaktywny, oznakowane miejsca wskazują, jakie pytania należy zadać użytkownikowi, aby ustalić fakty niezbędne do potwierdzenia celu wnioskowania

Oczywiście rzeczywiste reguły zawierają wiele reguł, często ze sobą powiązanych. Takie reguły tworzą powiązane ze sobą tranzycie. Uaktywnienie ich powoduje odpowiednią propagację znaczników, a tym samym potencjalną możliwość uaktywnienia następnych reguł (tranzycji), zgodnie z zasadami omówionymi powyżej. Przykład bazy złożonej z 3 reguł oraz odpowiadającej jej sieci Petriego prezentują odpowiednio rys. 4*a* i 4*b*.

Ograniczone ramy niniejszego artykułu nie pozwalają na szczegółowe przedstawienie wnioskowania w rozbudowanych sieciach reprezentujących bazy wiedzy, szczegółowy opis znaleźć można w pracach [10, 11].

a)
$$r_1 : a \wedge b \rightarrow c$$
$$r_2 : c \wedge d \rightarrow e$$
$$r_3 : e \wedge f \rightarrow g$$

b)


Rys. 4. Powiązane reguły i odpowiadająca im sieć Petriego
Fig. 4. Related rules and the corresponding Petri net

Przedstawione powyżej sieci należą do najbardziej podstawowej klasy binarnych sieci Petriego. Pozwalają one na realizację wnioskowania jedynie na gruncie logiki dwuwartościowej. Takie sieci zostały zastosowane w proponowanym systemie. Ta prosta koncepcja budowania bazy wiedzy na bazie języka sieci Petriego niesie ze sobą wiele przypadków szczególnych, utrudniających modelowanie procesów wnioskowania i przetwarzania wiedzy. Dlatego z wykorzystaniem tego narzędzia zaproponowano inne klasy sieci Petriego, rozszerzające możliwości modelowania wiedzy. Opis tych sieci wykracza poza ramy niniejszego artykułu, interesujące opisy wykorzystania zaawansowanych klas sieci Petriego znaleźć można w pracach [14, 15, 16].

4. System wizualizacji procesów wnioskowania

Celem systemu opisywanego w niniejszym artykule jest wspomaganie inżyniera wiedzy w analizie regułowych baz wiedzy, poprzez wizualizację dynamiki procesów wnioskowania w tychże bazach. Do realizacji tych postulatów system wykorzystuje binarne sieci Petriego, narzędzie łączące w naturalny sposób możliwości analizy statycznej struktury bazy wiedzy z analizą dynamiczną procesów wnioskowania. System powinien dostarczać sugestywnych i łatwych w zrozumieniu metod wizualizacji przebiegu wnioskowania oraz wyjaśniania jego wyników – zarówno w celach kontrolnych, jak i dydaktycznych.

4.1. Funkcje systemu

W swoim założeniu projektowany system nie jest systemem ekspertowym, przeznaczonym do rozwiązywania złożonych problemów oraz wykonywania procesów wnioskowania, ponieważ do tego celu istnieją już dobrze działające systemy szkieletowe. System natomiast stanowi narzędzie wspomagające realizację baz wiedzy, poprzez wygodny i prosty w zrozumieniu sposób prezentacji struktury i procesów wnioskowania w bazach wiedzy, w których procesy te bywają niejasne dla inżyniera wiedzy. System nie jest również interaktywnym edy-

torem regułowych baz wiedzy, to zagadnienie było elementem innego projektu. System obsługuje bazy wiedzy zapisane w postaci plików XML, tworzone i edytowane z wykorzystaniem zewnętrznych narzędzi. System umożliwia: (1) odczyt danych z pliku XML; (2) translację reguł zapisanych w pliku XML na strukturę reprezentującą sieć Petriego, (3) wizualizację struktury regułowej bazy wiedzy na grafie reprezentującym sieć Petriego, (4) modelowanie procesów wnioskowania progresywnego i regresywnego w regułowych bazach wiedzy, z wykorzystaniem propagacji znaczników. Wizualizacja obejmuje: możliwość dodawania faktów, wykorzystywanych w procesie wnioskowania; możliwość wybrania przez użytkownika dowolnego celu wnioskowania; krokową wizualizację dynamiki procesu wnioskowania na grafie reprezentującym sieć Petriego; prezentację aktualnie aktywnej reguły w bazie faktów oraz podgląd zbioru faktów dodawanych w trakcie procesu wnioskowania, informowanie użytkownika o osiągnięciu (lub nie) celu wnioskowania.

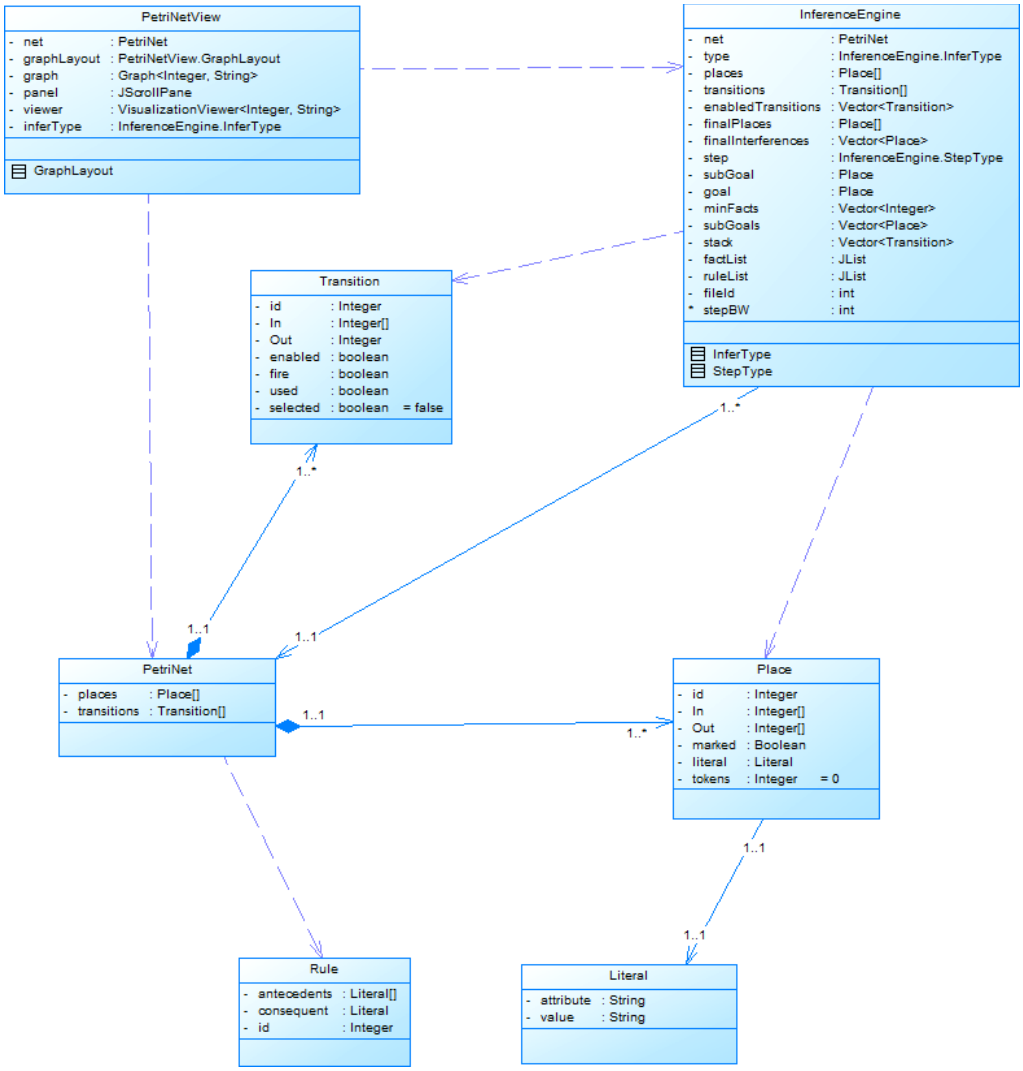
4.2. Architektura wewnętrzna

Do realizacji systemu będącego tematem niniejszego artykułu wybrano język Java oraz frameworka Swing Application Framework. Framework bazuje na technologii Java Swing. Aplikacja zaimplementowana została w środowisku programistycznym NetBeans. Struktura programu została podzielona na następujące pakiety: *Nets*, *XMLParser*, *Application, Application.resources.*

Pakiet *Nets* zawiera zbiór klas potrzebnych do opisania struktury sieci Petriego oraz algorytmy wnioskowania. Pakiet *XMLParser* zawiera tę część kodu aplikacji, odpowiedzialną za pobieranie i konwertowanie dostarczonego do programu pliku XML z regułową bazą wiedzy, na strukturę danych reprezentującą sieć Petriego. Pakiet *Application* zawiera główną klasę aplikacji oraz zbiór klas bezpośrednio związanych z obsługą interfejsu użytkownika. Zasoby niebędące kodem Javy, zgodnie z frameworkiem SAF, umieszczono w pakiecie *Application.resources*. Do wizualizacji regułowej bazy wiedzy w postaci sieci Petriego, wykorzystano (dostępną na licencji open source) bibliotekę o nazwie *JUNG* (*Java Universal Network*/*Graph Framework*), która dostarcza interfejs do modelowania, analizy i wizualizacji wszelkich danych, reprezentowanych za pomocą grafu.

Diagram hierarchii klas zaprojektowanych dla potrzeb programu przedstawia rys. 5. Obiekty klasy *Place* reprezentują miejsca sieci Petriego. Obiekty klasy *Transition* reprezentują tranzycje sieci Petriego. Klasa *PetriNet* reprezentuje całą sieć Petriego. Na strukturze tej operują algorytmy wnioskowania, jest ona również wykorzystywana przez klasy interfejsu użytkownika do rysowania i wizualizacji grafów sieci Petriego. Zadaniem klasy *PetriNetView* jest pośredniczenie pomiędzy interfejsem użytkownika, a strukturami danych, reprezentującymi sieć Petriego. Klasa pełni funkcję rozdzielającą warstwę prezentacji od warstwy logiki

systemu. Klasa *InferenceEngine* zaprojektowana została jako odpowiednik maszyny wnioskującej w klasycznych systemach ekspertowych. Głównym celem tej klasy jest udostępnienie interfejsu umożliwiającego przeprowadzanie operacji wnioskowania, wykorzystując struktury danych, reprezentujące sieć Petriego.

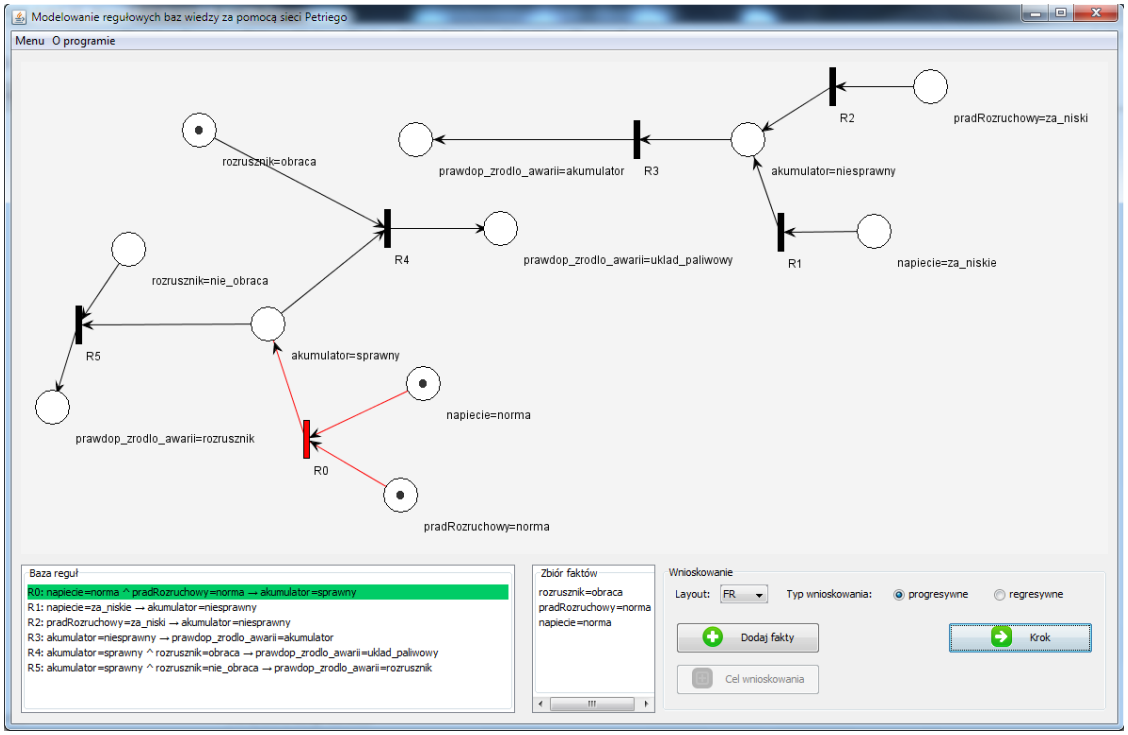


Rys. 5. Diagram klas systemu
Fig. 5. Class diagram of the system

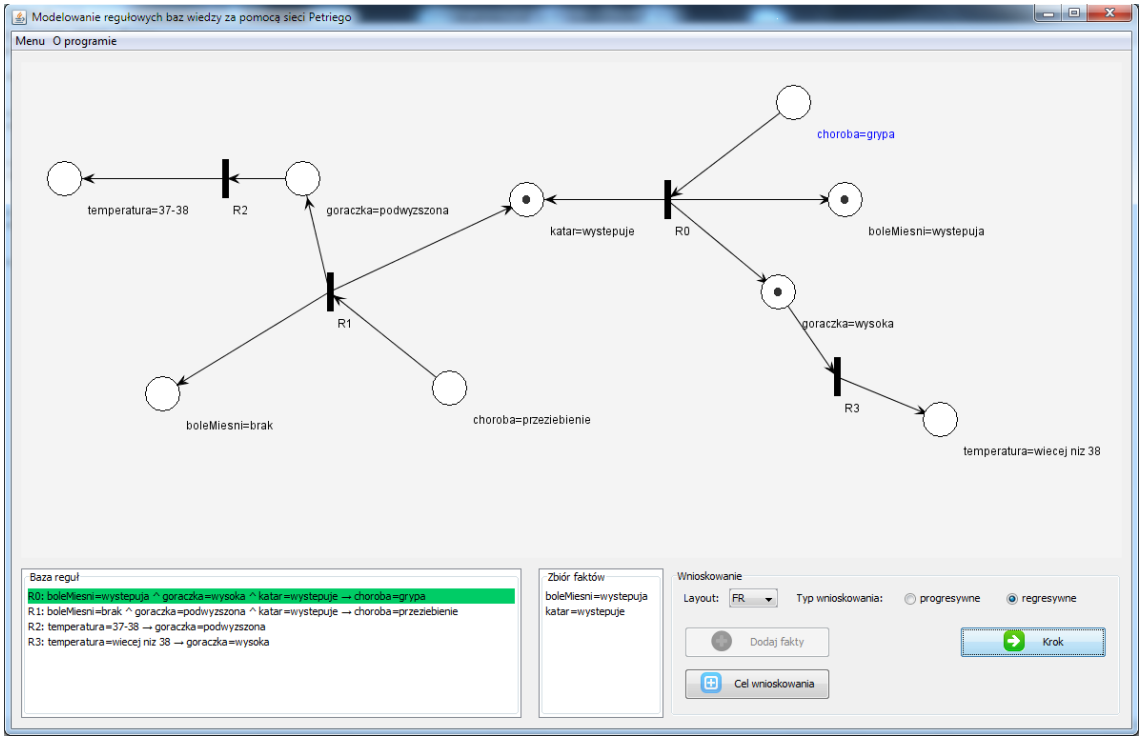
4.3. Interfejs użytkownika

Rysunki 6 i 7 przedstawiają przykłady wykorzystania zrealizowanego systemu w zadaniach wnioskowania w przód i wstecz. W obu przypadkach ekran został podzielony na dwie części – górną, zawierającą sieć Petriego, odpowiadającą wczytanej bazie wiedzy oraz dolną, zawierającą podgląd listy reguł, faktów oraz elementy pozwalające na wybór rodzaju wnioskowania oraz manipulowanie jego przebiegiem. Układ graficzny można zmieniać manual-

nie, poprzez zmianę ekranowej pozycji miejsc i tranzycji oraz poprzez wybór wśród predefiniowanych układów.



Rys. 6. Przykład wnioskowania w przód
Fig. 6. Forward inference example



Rys. 7. Przykład wnioskowania wstecz
Fig. 7. Backward inference example

5. Podsumowanie i wnioski końcowe

Wnioski płynące z prac implementacyjnych oraz testów praktycznych systemu prezentowanego w ramach tego artykułu jednoznacznie wskazują, że wykorzystanie sieci Petriego do modelowania procesów wnioskowania jest uzasadnione zarówno w warstwie teoretycznej, jak i praktycznej. Tak przyjęta konwencja prezentacji regułowej bazy wiedzy jest prosta i intuicyjnie zrozumiała. Wykorzystanie znaczników dobrze modeluje procesy wnioskowania, które stają się zrozumiałe dla szerokiego grona użytkowników. Należy pamiętać, że twórcy dziedzinowych baz wiedzy często nie mają przygotowania informatycznego. Regułowa baza wiedzy w postaci sieci Petriego to również nieocenione źródło możliwości związanych z weryfikacją poprawności bazy regułowej, tym bardziej, że istnieje możliwość wykorzystania wielu innych klas sieci Petriego [5, 7, 15], a nie tylko sieci binarnych, zastosowanych w prezentowanym systemie.

Przedstawiony system stał się nieocenionym źródłem doświadczeń i spostrzeżeń, wykorzystanych przy opracowaniu założeń nowej wersji systemu *kbBuilder* [13], w którym zaplanowano połączenie koncepcji sieci Petriego oraz koncepcji jednostek decyzyjnych [9]. Połączenie to powinno rozwiązać podstawową słabość reprezentacji bazy regułowej w postaci sieci Petriego — jest nim utrata czytelności wraz ze wzrostem rozmiaru bazy. Niestety, próba wizualizacji dużych baz wiedzy w przedstawionym systemie dowiodła, że odwzorowanie dużej bazy regułowej bezpośrednio w sieć Petriego nie sprawdza się w warunkach praktycznych. Dzieje się tak już przy kilkudziesięciu regułach. Sieć staje się nieczytelna, a wizualizacja procesów wnioskowania chaotyczna. W nowej wersji systemu *kbBuilder* wizualizacja struktury oraz modelowanie procesów zachodzących w bazach regułowych zorientowane będą właśnie na jednostki decyzyjne [12, 13]. Umożliwiają one na budowanie hierarchiczne zorganizowanych struktur, które pozwalają na efektywne zarządzanie nawet dużymi bazami reguł. Prace nad realizacją systemu *kbBuilder* trwają, ich ukończenie zaplanowane jest na koniec 2011 roku.

BIBLIOGRAFIA

1. Jackson P.: Introduction to Expert Systems, Addison-Wesley, New York, USA 1999.
2. Murata T.: Petri Nets: Properties, Analysis and Aplications. Proceedings of IEEE, Vol. 77, No. 4, April 1989.
3. Nazareth D. L.: Investigating the applicability of Petri nets for rule-based system verification. IEEE Transactions on Knowledge and Data Engineering, 4 (3), 1992.

4. Nowak A., Simiński R., Wakulicz-Deja A.: Inference algorithms for hierarchical knowledge bases. Kłopotek M. A., Przepiórkowski A., Wierzchoń S. T., Trojanowski K. (red.): Recent Advances in Intelligent Information Systems. Academic Publishing House EXIT.
5. Peterson J. L.: Petri Net Theory and the Modeling of Systems. Prentice Hall, Englewood Cliffs, NJ, 1981.
6. Reichgelt H.: Knowledge Representation: An AI Perspective. Ablex Publishing Corporation, New Jersey 1991.
7. Rozenberg G., Thiagarajan P. S.: Petri nets: basic notations, structure, behavior. Current Trends in Concurency. Lecture Notes in Computer Science, Vol. 224, Springer-Verlag, Berlin 1986.
8. Simiński R.: Decision units approach in knowledge base modeling, [in:] Kłopotek M. A., Przepiórkowski A., Wierzchoń S. T., Trojanowski K. (eds.): Recent Advances in Intelligent Information Systems, Academic Publishing House EXIT, 2009, s. 597÷606.
9. Simiński R.: Extending decision units conception using Petri nets. Advances in Soft Computing, Intelligent Information Processing and Web Mining, Springer-Verlag, 2006, s. 413÷420.
10. Simiński R.: Modelowanie procesów wnioskowania z wykorzystaniem sieci Petriego. Materiały Konferencji Naukowej Systemy Wspomagania Decyzji, Zakopane 2006, s. 127÷137.
11. Simiński R.: Petri net and matrix representation of rule knowledge base for verification task. Intelligent Information Processing and Web Mining, Advances in Soft Computing, Springer-Verlag, 2005.
12. Simiński R., Wakulicz-Deja A.: Decision units as a tool for rule base modeling and verification. Intelligent Information Processing and Web Maining, Advances in Soft Computing, Physica-Verlag, 2003.
13. Simiński R., Wakulucz-Deja A.: Application of Decision Units in Knowledge Engineering. Rough Sets and Current Trends in Computing, Lecture Notes in Artificial Intelligence, Springer-Verlag, 2004.
14. Suraj Z.: Rough Set Methods for the Synthesis and Analysis of Concurrent Processess. ICS PAS Reports, 893, Warszawa December 1999.
15. Suraj Z., Fryc B., Matusiewicz Z., Pancerz K.: A Petri Net System: an Overview, [in:] Flasiński M., Nawarecki E., Polkowski L., Schaefer R., Stefanowski J., Suraj Z. (eds.): Special volume with selected papers from TASC 2004 workshop, Fundamenta Informaticae, Vol. 71, No. 1, IOS Press, Amsterdam 2006, s. 101÷119.

Recenzenci: Dr inż. Robert Brzeski
Prof. dr hab. inż. Henryk Rybiński

Wpłynęło do Redakcji 30 stycznia 2011 r.

Abstract

The paper presents idea of utilization of Petri nets to visualization the structure of rule knowledge bases and the functional and implementation issues of system using this conception for visualization both the structure of knowledge bases and the inference process in such bases. The basic terms of Petri nets as well as the idea of using Petri nets as the modeling rule knowledge base have been presented in this paper. The method of using Petri nets for modeling of the inference process has been also discussed in further part of this paper. Short discourse has been included in the summary to this paper on foreseen directions of Petri nets usage in extending the properties of decision units conception.

Adresy

Roman SIMIŃSKI: Uniwersytet Śląski, Instytut Informatyki, ul. Będzińska 39,
41-200 Sosnowiec, Polska, roman.siminski@us.edu.pl.
Sebastian JANUS: Uniwersytet Śląski, Instytut Informatyki, ul. Będzińska 39,
41-200 Sosnowiec, Polska, sebbbastian@gmail.com.