# Uniwersytet Zielonogórski

## Wydział Elektrotechniki, Informatyki i Telekomunikacji



**mgr inż. Jacek Tkacz**


# PROJEKTOWANIE UKŁADÓW STEROWANIA BINARNEGO WSPOMAGANE AUTOMATYCZNYM WNIOSKOWANIEM GENTZENA

Rozprawa doktorska

Promotor:
prof. dr hab. inż. Marian Adamski

Zielona Góra 2008

# Spis treści

Spis treści ..... 2
Spis ilustracji ..... 5
Spis tabel ..... 7
1. Wprowadzenie ..... 9
1.1. Motywacja ..... 10
1.2. Struktura pracy ..... 11
2. Cele, teza pracy i zadania ..... 13
2.1. Cele pracy ..... 13
2.2. Teza pracy ..... 13
2.3. Zadania ..... 16
3. Rachunek sekwentów Gentzena ..... 17
3.1. Wprowadzenie ..... 17
3.2. Eliminacja spójników logicznych ..... 20
3.2.1. Negacja ..... 20
3.2.2. Dysjunkcja ..... 21
3.2.3. Koniunkcja ..... 22
3.2.4. Implikacja ..... 23
3.2.5. Równoważność ..... 24
3.3. Tautologie, sekwenty spełniane i sekwenty sprzeczne ..... 25
3.4. Reguły dopuszczalne ..... 26
3.4.1. Skracanie ..... 27
3.4.2. Dominacja ..... 27
3.4.3. Cięcie ..... 27
3.4.4. Reguły dedykowane ..... 28
3.5. Odwracalność reguł wnioskowania ..... 30
4. Algorytm redukcji zbioru sekwentów ..... 32
4.1. Uporządkowana eliminacja spójników logicznych ..... 32
4.2. Wczesne wykrywanie tautologii ..... 33
4.3. Minimalizacja bazy rozwiązań ..... 34
4.4. Wykorzystanie elementów metody Thelena-Mathony'ego ..... 34

4.4.1. Zarys metody Thelena-Mathony'ego ..... 34
4.4.2. Metoda Thelena-Mathony'ego w ujęciu rachunku sekwentów ..... 36
4.5. Wykorzystanie metody rezolucji ..... 41
4.5.1. Zarys metody rezolucji ..... 42
4.5.2. Rezolucja w logice sekwentów ..... 42
5. Wykorzystanie komputerowego wnioskowania metodą Gentzena w projektowaniu układów cyfrowych ..... 44
5.1. Synteza kombinacyjnych układów cyfrowych ..... 44
5.1.1. Normalizacja i minimalizacja funkcji logicznych ..... 47
5.1.2. Weryfikacja układu kombinacyjnego względem częściowej lub pełnej specyfikacji ..... 55
5.2. Synteza sekwencyjnych układów cyfrowych ..... 60
5.2.1. Wyznaczanie funkcji wzbudzeń rejestrów funkcyjnych ..... 61
5.2.2. Normalizacja opisu sekwencyjnego układu sterującego opisanego tabelą przejść-wyjść ..... 63
5.3. Współpraca systemu automatycznego wnioskowania z kompilatorami języków opisu sprzętu ..... 65
5.4. Współpraca systemu automatycznego wnioskowania z systemami uniwersyteckimi ..... 74
6. Wykorzystanie komputerowego wnioskowania do analizy współbieżnych układów dyskretnych ..... 77
6.1. Analiza symboliczna grafów współbieżności ..... 77
6.2. Kolorowanie automatowe sieci Petriego i dekompozycja równoległa ..... 80
6.3. Badanie żywotności sieci Petriego ..... 86
7. Opis eksperymentalnego systemu wnioskowania „GENTZEN v.6" ..... 92
7.1. Założenia projektowe ..... 92
7.2. Ogólny opis systemu ..... 93
7.3. Interfejs użytkownika ..... 94
7.4. Struktura oprogramowania ..... 101
7.4.1. Diagram klas ..... 101
7.4.2. Diagram przypadków użycia ..... 104
7.4.3. Diagramy aktywności dla systemu wnioskującego ..... 105
7.4.4. Diagram stanów ..... 110
7.5. Testowanie oprogramowania ..... 111

8. Badania opracowanego algorytmu ..... 113
9. Podsumowanie ..... 120
9.1. Elementy nowatorskie i autorskie ..... 121
9.2. Plany dalszych prac ..... 122
[Literatura] ..... 124
Dodatek A – Spis przykładów ..... 129
Dodatek B – Struktura płyty CD ..... 130

# Spis ilustracji

Rys. 3-1 Proponowana reprezentacja drzewa dowodu ..... 19
Rys. 3-2 Przejście od sekwentów przesłanek do sekwentu wniosku ..... 30
Rys. 4-1 Graficzna reprezentacja normalizacji sekwentu ..... 32
Rys. 4-2 Reprezentacja graficzna reguły R1 ..... 35
Rys. 4-3 Reprezentacja graficzna reguły R2 ..... 35
Rys. 4-4 Reprezentacja graficzna reguły R3 ..... 35
Rys. 4-5 Reprezentacja graficzna reguły R4 ..... 36
Rys. 4-6 Reguła R1 w rachunku sekwentów Gentzena ..... 37
Rys. 4-7 Reguła R2 w rachunku sekwentów Gentzena ..... 37
Rys. 4-8 Konstrukcja stosu podręcznego ..... 38
Rys. 4-9 Reguła R3 w rachunku sekwentów Gentzena ..... 38
Rys. 4-10 Reguła R4 w rachunku sekwentów Gentzena ..... 39
Rys. 4-11 Połączenie wnioskowania z elementami metody Thelena-Mathony'ego ..... 40
Rys. 4-12 Rozwinięcie gałęzi 10 ..... 41
Rys. 4-13 Rozwinięcie gałęzi 28 ..... 41
Rys. 5-1 Schemat blokowy multipleksera ..... 45
Rys. 5-2 Schemat blokowy minimalizacji funkcji ..... 48
Rys. 5-3 Wyznaczanie pierwszego rozwiązania ..... 50
Rys. 5-4 Układ realizujący funkcję Y1 ..... 56
Rys. 5-5 Układ realizujący funkcję Y2 ..... 57
Rys. 5-6 Siatka dla wyników normalizacji Y1|-Y2 ..... 58
Rys. 5-7 Siatka dla wyników normalizacji Y2|-Y1 ..... 59
Rys. 5-8 Siatka dla wyników normalizacji sekwentu |-Y1<->Y2 ..... 59
Rys. 5-9 Schemat blokowy układu rejestrowego ..... 60
Rys. 5-10 Graf sekwencyjnego licznika mod4 ..... 61
Rys. 5-11 Układ sekwencyjny z wykorzystaniem rejestru ..... 63
Rys. 5-12 Schemat blokowy wykorzystania języków HDL ..... 66
Rys. 5-13 Oznakowana sieć działań ..... 67
Rys. 5-14 Układ sekwencyjny z wykorzystaniem rejestru ..... 69
Rys. 5-15 Układ rejestrowy ..... 73

Rys. 5-16 Symulacja w środowisku Active-HDL ..... 73
Rys. 6-1 Graf współbieżności ..... 77
Rys. 6-2 Przykład sieci pokolorowanej i jej graf współbieżności miejsc ..... 81
Rys. 6-3 System mieszania cieczy ..... 82
Rys. 6-4 Sieć Petriego dla systemu mieszania cieczy ..... 83
Rys. 6-5 a) Graf znakowań osiągalnych; b) Graf relacji współbieżności ..... 84
Rys. 6-6 Pokolorowana sieć Petriego dla systemu mieszania cieczy ..... 85
Rys. 6-7 Wycinek sieci Petriego ..... 86
Rys. 6-8 Przykład normalizacji sekwentów blokad i pułapek ..... 88
Rys. 6-9 Schemat badania żywotności sieci ..... 89
Rys. 6-10 Zmodyfikowany system mieszania cieczy ..... 90
Rys. 6-11 Sieć Petriego dla zmodyfikowanego systemu mieszania cieczy ..... 90
Rys. 7-1 Interfejs programu ..... 94
Rys. 7-2 Pasek narzędzi ..... 97
Rys. 7-3 Parametryzowanie programu ..... 97
Rys. 7-4 Kreator typów formuł ..... 99
Rys. 7-5 Wyniki procesu transformacji ..... 101
Rys. 7-6 Diagram klas ..... 103
Rys. 7-7 Diagram przypadków użycia ..... 104
Rys. 7-8 Diagram aktywności systemu ..... 106
Rys. 7-9 Diagram aktywności procesu normalizacji ..... 107
Rys. 7-10 Diagram aktywności wyszukiwania głównego spójnika ..... 108
Rys. 7-11 Algorytm rezolucji ..... 109
Rys. 7-12 Diagram stanów ..... 110
Rys. 8-1 Testowane typy grafów – a) pełne b) cykliczne ..... 114

# Spis tabel

Tab. 3-1 Specyfikacja bramki AND ..... 31
Tab. 5-1. Deklaracje zmiennych ..... 45
Tab. 5-2 Postać ogólna mikrooperacji ..... 45
Tab. 5-3 Specyfikacja multipleksera ..... 45
Tab. 5-4 Wyrażenia znormalizowane w postaci sumy iloczynów ..... 46
Tab. 5-5 Sekwent porównujący zbiory $F^0$ i $F^1$ ..... 48
Tab. 5-6 Implikanty funkcji ..... 49
Tab. 5-7 Tabela Quine'a pokrycia funkcji ..... 49
Tab. 5-8 Sekwent pokrycia funkcji ..... 49
Tab. 5-9 Minimalne pokrycia funkcji ..... 50
Tab. 5-10 Tabela decyzyjna rozpatrywanego układu ..... 52
Tab. 5-11 Specyfikacja behawioralna (założenie o otwartości świata) ..... 53
Tab. 5-12 Definicje typu zmiennych ..... 53
Tab. 5-13 Specyfikacja znormalizowana i uporządkowana ..... 53
Tab. 5-14 Symulacja logiczna dla wciśniętego przycisku pierwszego ..... 54
Tab. 5-15 Specyfikacja behawioralna z zabezpieczeniem ..... 55
Tab. 5-16 Tabela przejść ..... 61
Tab. 5-17 Specyfikacja w postaci sekwentowej dla przerzutnika D ..... 61
Tab. 5-18 Reprezentacja sekwentowa funkcji wzbudzeń ..... 62
Tab. 5-19 Specyfikacja w postaci sekwentowej dla przerzutnika JK ..... 62
Tab. 5-20 Reprezentacja sekwentowa funkcji wzbudzeń ..... 63
Tab. 5-21 Przykładowa tabela przejść-wyjść Mealey'ego ..... 64
Tab. 5-22 Reprezentacja sekwentowa tablicy przejść-wyjść ..... 64
Tab. 5-23 Znormalizowana reprezentacja sekwentowa dla tablicy przejść-wyjść ..... 65
Tab. 5-24 Tabela kodowania stanów ..... 68
Tab. 5-25 Reprezentacja sekwentowa sieci działań ..... 69
Tab. 5-26 Sekwenty reprezentujące poszczególne funkcje układu ..... 70
Tab. 5-27 Specyfikacja układu w języku VHDL ..... 71
Tab. 5-28 Specyfikacja układu w języku Verilog ..... 72
Tab. 5-29 Przykładowe wyniki syntezy układu ..... 74

Tab. 5-30 Specyfikacja układu w formacie ESPRESSO ..... 75
Tab. 5-31 Specyfikacja układu w formacie ESPRESSO po dodatkowej minimalizacji ..... 76
Tab. 6-1 Sekwent sąsiedztwa wierzchołków ..... 78
Tab. 6-2 Wyznaczone bazy grafu ..... 78
Tab. 6-3 Sekwent zbiorów wierzchołków przyległych ..... 79
Tab. 6-4 Sekwenty reprezentujące minimalne zbiory dominujące grafu ..... 79
Tab. 6-5 Sekwent dla grafu znakowań ..... 84
Tab. 6-6 Sekwent opisujący listę sąsiedztwa ..... 84
Tab. 6-7 Wynik procesu normalizacji ..... 85
Tab. 6-8 Sekwent dla blokad ..... 90
Tab. 6-9 Sekwent dla pułapek ..... 91
Tab. 6-10 Minimalne blokady i pułapki ..... 91
Tab. 7-1 Przykładowa dokumentacja procesu normalizacji ..... 96
Tab. 8-1 Testowanie algorytmu na wybranych zadaniach z logiki matematycznej ..... 113
Tab. 8-2 Testowanie algorytmu z wykorzystaniem grafów ..... 114
Tab. 8-3 Testy grafów generowanych losowo ..... 116
Tab. 8-4 Analiza sekwentów sąsiedztwa grafów losowych w postaci koniunkcyjnej ..... 117
Tab. 8-5 Testowanie algorytmu z wykorzystaniem przykładów zawartych w pracy ..... 118

# 1. Wprowadzenie

Rachunek sekwentów Gentzena przez wiele lat był niezauważony przez informatyków i matematyków. Pierwszą pracą w języku polskim była książka Adama Pawlaka [Pawl65], w której przedstawiono wykorzystanie rachunku sekwentów Gentzena do automatycznego wnioskowania metodą Wanga [Wang60]. Przełomem okazało się wydanie tłumaczenia prac Gentzena na język angielski przez Szabo [Gent69]. Pojawiły się pierwsze monografie dotyczące wykorzystania rachunku sekwentów Gentzena w informatyce [ErPa84][Gall86]. Wykorzystanie komputerowego wnioskowania do analizy i syntezy układów cyfrowych spowodowało rozwój logiki formalnej, stymulując powstanie szeregu nowych publikacji na ten temat [Krop99]. W języku polskim powstały między innymi następujące opracowania zwarte omawiające logikę Gentzena [Gent80][Pogo81][Jers81][Logi87][Lynd87][Adam90][BiWo93][Szaj96][Tiur98] [Bena05][Indr][Indr06]. Na uwagę zasługują prace ośrodka zielonogórskiego, w których automatyczne dowodzenie twierdzeń powiązano z metodami projektowania układów cyfrowych [Adam90][Szaj96].

Wykorzystanie rachunku sekwentów w projektowaniu układów cyfrowych wraz z obszerną bibliografią zostało przedstawione w monografii [Adam90]. Monografia [Adam90] skupiała się wokół zagadnień projektowania układów cyfrowych metodą systematyczną i dotyczyła głównie teorii automatycznej syntezy. Praca ta zawiera przegląd wcześniejszych publikacji dotyczących opisu układów w sekwentowej postaci regułowej. Na szczególną uwagę zasługują prace A. Zakrewskiego podsumowane w książce [Zakr81]. W monografii [Szaj96] przedstawiono szczegółowo realizację programu wnioskującego z wykorzystaniem Prologu. Inna wartościowa implementacja powstała na Uniwersytecie Śląskim [BiWo93]. Wykorzystanie metod formalnych z zastosowaniem rachunku Gentzena podsumowano między innymi w książce [AdKa05].

Jeden z pierwszych w Polsce systemów wnioskujących metodą Gentzena, dedykowany do syntezy układów cyfrowych, został zrealizowany w Wyższej Szkole Inżynierskiej w Zielonej Górze przez Mariana Adamskiego, Janusza Szajnę i Tomasza Kozłowskiego [Szaj96][Adam90]. Postęp technologiczny, wprowadzenie nowych systemów operacyjnych oraz dominacja nowych języków programowania, a także brak dostępu do źródeł archiwalnych i ich dokumentacji sprawiły, że prace programistyczne zaczęto ponownie od podstaw.

W chwili obecnej system Gentzena znajduje coraz większe wykorzystanie w rozwiązywaniu trudnych teoretycznych problemów logicznych metodami komputerowymi.

Rezultatem pracy autora jest oryginalne opracowanie dotyczące zagadnień komputerowej, symbolicznej analizy systemów dyskretnych pod kątem syntezy układów cyfrowych. W pracy skupiono się głównie na analizie struktur dyskretnych opisanych siecią Petriego. W odróżnieniu od monografii [Adam90][Szaj96] skupiono się zarówno na usprawnieniu systemu wnioskującego, jak i jego wykorzystaniu do rozwiązywania problemów kombinatorycznych występujących w projektowaniu układów sterowania binarnego. Rozprawa doktorska nie pretenduje do opracowania teoretycznego rachunków gentzenowskich. Przedstawiono tutaj jeden ze specyficznych wariantów, w ujęciu zbliżonym do Wanga [Wang60] i Pawlaka [Pawl65]. Oprócz oryginalnych prac Gentzena [Gent69] szczególnie wartościowe są książki [ErPa84][Gall86][Tiur98][Indr06] [Huza07]. W rozprawie szerzej nie omawiano symbolicznej metody projektowania układów cyfrowych metodą systematyczną [Adam90]. W pracy tej wykorzystano formalne, komputerowe dowodzenie twierdzeń do specyfikacji układów kombinacyjnych, jak i sekwencyjnych. Na uwagę zasługuje też wykorzystanie wnioskowania w rachunku sekwentów do formalnej specyfikacji i syntezy sterowników logicznych, realizowanych układowo za pośrednictwem języków opisu sprzętu HDL [Adam91][BaKu93][BiAd94] [AdKa05][AdWę07].

## 1.1. Motywacja

Z przeglądu literatury wynika, że na świecie powstało szereg uniwersyteckich opracowań systemów wnioskujących przeznaczonych do rozwiązywania problemów naukowo-technicznych. Niestety ich przydatność do komputerowego projektowania układów dyskretnych przez inżynierów jest dyskusyjna, a informacje o ewentualnych komputerowych implementacjach wyjątkowo skąpe. Programy te zostały pisane głównie przez logików w celu komputerowego dowodzenia ogólnych twierdzeń matematycznych. Z tego względu posiadają one szereg cech nadmiarowych, które nie zawsze są przydatne w procesie analizy i syntezy. Często charakteryzują się silną interaktywnością, a także wymagają szerokiej wiedzy matematycznej. Ponadto zespoły badawcze nie promują i na ogół nie udostępniają swojego oprogramowania konkurencyjnym zespołom badawczym.

Do komputerowego wnioskowania można wykorzystać narzędzia profesjonalne, na przykład oparte na grafach BDD [Shin96][Krop99][AdKa05]. Pozwalają one na rozwiązywanie szeregu problemów projektowych, w tym zwłaszcza dotyczących

formalnej weryfikacji układów cyfrowych. Ich wyraźne ukierunkowanie w stronę weryfikacji utrudnia wykorzystanie w badaniach naukowych, gdzie projektant musi mieć dużą swobodę w formułowaniu problemów logicznych.

Rozpoczęcie prac badawczych nad nowatorskim algorytmem wnioskowania symbolicznego Gentzena oraz jego zastosowaniem w syntezie i analizie układów dyskretnych wynika z następujących potrzeb:

- opracowanie sprawnego systemu wnioskującego i jego technicznej realizacji;
- wspomaganie badań naukowo-technicznych formalnym wnioskowaniem;
- wskazanie i przetestowanie przykładowych zastosowań metody wnioskowania w projektowaniu układów dyskretnych, a w szczególności układów sterowania binarnego;
- połączenie intuicji i doświadczeń praktycznych projektanta z metodami formalnymi przy rozwiązywaniu problemów kombinatorycznych techniki cyfrowej;
- zintegrowanie formalnego komputerowego wnioskowania z profesjonalnymi i uniwersyteckimi systemami CAD przeznaczonymi do projektowania układów cyfrowych (symulacja i synteza logiczna).

W rozprawie nie analizowano sprawności wcześniej opracowanych, komputerowych systemów wnioskowania logicznego, opierających się na metodach gentzenowskich. Większość z nich ma znaczenie historyczne, dostęp do nich jest praktycznie niemożliwy, a postęp inżynierii oprogramowania oraz technologii komputerowej utrudnia ich ewentualne uruchomienie na współczesnych platformach projektowych. Niepełny, ale wystarczający przegląd zawierają prace [Adam90][BiWo93][Szaj96][Indr06].

## 1.2. Struktura pracy

Praca została podzielona na dziewięć rozdziałów. Rozdział pierwszy stanowi wprowadzenie do tematyki przedmiotu, przedstawia motywację podjęcia tematu, a także omawia strukturę pracy.

W rozdziale drugim zdefiniowano tezę pracy oraz popierające ją tezy szczegółowe. Omówiono zasadnicze cele pracy oraz wynikające z nich zadania.

W rozdziale trzecim omówiono logikę sekwentów Gentzena, przedstawiono reguły wnioskowania oraz reguły dopuszczalne będące rozwinięciem reguł zasadniczych lub zaadoptowaniem do logiki sekwentów znanych praw logiki matematycznej. Rachunek

sekwentów Gentzena omówiono zarówno w ujęciu redukcyjnym (eliminacja spójników logicznych), jak i w ujęciu wprowadzania spójników logicznych.

W rozdziale czwartym omówiono autorski algorytm redukcji zbioru sekwentów, oparty na regułach tautologii, cięcia i dominacji. Przedstawiono metody Thelena-Mathony'ego i rezolucji oraz zaproponowano sposób zaadaptowania ich do rachunku sekwentów Gentzena.

W rozdziale piątym zaproponowano i omówiono przykładowe obszary zastosowania wnioskowania symbolicznego Gentzena w projektowaniu cyfrowych układów kombinacyjnych i sekwencyjnych. Zaproponowano sposób współpracy systemu wnioskującego z profesjonalnym oprogramowaniem do projektowania, symulacji i syntezy układów cyfrowych, poprzez transformację opisu specyfikacji z języku logiki sekwentów na języki opisu sprzętu (HDL). Omówiono także możliwość współpracy systemu wnioskującego z oprogramowaniem uniwersyteckim, przeznaczonym głównie do syntezy układów cyfrowych, za pośrednictwem formatu ESPRESSO.

W rozdziale szóstym omówiono i zaprezentowano proponowane obszary zastosowania wnioskowania symbolicznego Gentzena do analizy współbieżnych układów dyskretnych. Analiza pewnych własności grafów (np. grafów współbieżności), czy własności sieci Petriego z wykorzystaniem naturalnego wnioskowania pozwala na uzyskanie wielu cennych informacji o badanym układzie i jego funkcjonowaniu.

Rozdział siódmy stanowi dokumentację implementacji prototypowego autorskiego systemu wnioskującego. Omówione zostały założenia projektowe, opisano zaprojektowany interfejs użytkownika. Strukturę oprogramowania oraz ważniejsze algorytmy zaprezentowane zostały w formie diagramów UML. Opisany został także sposób przeprowadzonych testów poprawności implementacji.

W rozdziale ósmym zamieszczono i omówiono testy opracowanego algorytmu wnioskującego, prezentując wyniki z autorskiego programu do normalizacji sekwentów według reguł Gentzena. Prezentowane wyniki dotyczą zarówno samego systemu gentzenowskiego, jak i rozszerzonego o elementy zaproponowane przez autora, opisane w rozdziale czwartym.

W rozdziale dziewiątym podsumowano pracę badawczą oraz zrealizowane w ramach pracy zdefiniowane zadania. Wskazano sposób udowodnienia założonej tezy oraz określono sposób osiągnięcia sformułowanych celów pracy.

W dodatkach zamieszczono spis wykorzystanych w pracy przykładów oraz opis zawartości dołączonej do pracy płyty CD.

# 2. Cele, teza pracy i zadania

## 2.1. Cele pracy

Podstawowym celem pracy doktorskiej jest efektywna, oryginalna realizacja algorytmu sprowadzania sekwentów do postaci normalnej. W porównaniu z poprzednimi rozwiązaniami znanymi z literatury [Wang60][Pawl65] [Adam90][Szaj96] [BiWo93] wprowadzono szereg usprawnień zarówno o charakterze merytorycznym jak i technicznym, wynikającym z postępu nauki oraz technologii informacyjnej. Efektywniejsza implementacja została osiągnięta poprzez:

- zrównoleglenie realizacji niektórych operacji logicznych podczas sporządzania drzewa dowodu;
- wyraźne skrócenie drzewa dowodu poprzez wprowadzenie elementów metody Thelena-Mathony'ego i wcześniejsze wykrywanie tautologii, przed pełną normalizacją sekwentu;
- wprowadzenie do metody wnioskowania Gentzena elementów metody rezolucji;
- zastąpienie binarnego drzewa dowodu siecią Petriego oraz otwarcie możliwości przetwarzania współbieżnego.

Drugim, istotnym celem pracy jest zaproponowanie kilku możliwych obszarów zastosowań dla symbolicznego wnioskowania Gentzena w projektowaniu układów sterowania binarnego.

## 2.2. Teza pracy

Na podstawie przedstawionej motywacji podjęcia badań sformułowano następującą tezę pracy:

**AUTOMATYCZNE WNIOSKOWANIE METODĄ GENTZENA USPRAWNIA I UWIARYGADNIA PROCES FORMALNEGO PROJEKTOWANIA UKŁADÓW STEROWANIA BINARNEGO**

Projektowanie formalne polega na połączeniu procesu syntezy logicznej [Trac86] [Łuba01] z procesem dowodzenia jej poprawności [Rich84][Adam90][Krop99]. Drzewo dowodu poprawności projektu jest równocześnie dokumentacją toku projektowania, zapisaną w formalnym języku logiki matematycznej. Dowodzenie poprawności implementacji względem specyfikacji jest rekomendowanym sposobem

projektowania sterowników o podwyższonym stopniu bezpieczeństwa (safe, dependable controllers)[HaMa97][AdWę07].

W pracy doktorskiej postuluje się bezpośrednie przekazywanie wyników syntezy formalnej do profesjonalnych systemów komputerowego projektowania za pośrednictwem języków opisu sprzętu HDL (VHDL lub VERILOG) [Zwol07]. Drugą opcją jest wykorzystanie formatu ESPRESSO, akceptowanego przez większość systemów uniwersyteckich [Łuba01].

Zasadnicza teza pracy jest poparta tezami szczegółowymi, dotyczącymi różnorodnych etapów projektowania układów cyfrowych. W pracy ograniczono się do analizy i syntezy układów sterowania binarnego, zawierających cyfrowy sterownik logiczny i obiekt o charakterze mechatronicznym [AdCh00]. Zadanie projektowe można zawęzić na układy cyfrowe składające się z części sterującej, jak i części operacyjnej przetwarzającej dane cyfrowe [BaWę06][StCy07].

Ważną tezą szczegółową, dotyczącą samego systemu wnioskującego jest to, że: **wprowadzenie do systemu wnioskującego dodatkowych reguł dopuszczalnych opartych na intuicji projektanta, lecz formalnie udowodnionych, usprawnia proces normalizacji sekwentów, czego następstwem jest znaczne skrócenie drzewa dowodu**.

System wnioskowania Gentzena poszukuje wszystkich rozwiązań dokładnych, co powoduje często generowanie bardzo obszernego drzewa dowodu. Wykorzystując elementy znanych z literatury metod analizy drzew binarnych [Bena05] możliwe jest odpowiednio wczesne odrzucenie gałęzi, które nie mają już wpływu na wartość logiczną rozpatrywanego wyrażenia logicznego.

Metoda automatycznego wnioskowania Gentzena pozwala na efektywną analizę blokad i pułapek metodą symboliczną w sieciach Petriego [BaMi92], a tym samym badania zastojów, w układach sterowania binarnego. Program sterownika przedstawiony siecią SFC lub siecią Petriego stanowi podstawę do napisania jego formalnej specyfikacji w języku logiki sekwentów [Adam90]. Na jej podstawie możliwe jest wygenerowanie odpowiedniego równania charakterystycznego i jego rozwiązanie [Krop99]. W ten sposób badając spełnialność zbioru sekwentów, można sprawdzić relacje między pułapkami i blokadami w sieci Petriego [Węgr03][Kara07] w jednym kroku bez konieczności sprowadzania wyrażeń do postaci normalnej DNF (klauzulowej).

W pracy doktorskiej przedstawiono różnorodne przykłady zastosowania logiki sekwentów Gentzena w projektowaniu układów sterowania binarnego:

- projektowanie układów kombinacyjnych metodą symboliczną [Tkac04];
- minimalizacja funkcji logicznych, zwłaszcza słabo określonych o dużej liczbie argumentów [TkAd08];
- weryfikacja poprawności układu kombinacyjnego względem częściowej lub pełnej specyfikacji [TkAd05];
- wyznaczanie funkcji wzbudzeń układu sekwencyjnego (rozdz. 5.2.1);
- dekompozycja automatu współbieżnego na składowe automatowe [Tkac07];
- analiza i symulacja logiczna układów dyskretnych metodą symboliczną [Tkac06a];
- projektowanie klasycznych automatów opisanych tablicą przejść i wyjść [TkAd08a].

Ze względu na swoją uniwersalność sekwentowy system dowodzenia pozwala na komputerowe, symboliczne rozwiązywanie klasycznych problemów logicznych i kombinacyjnych. W wymienionych wyżej problemach technicznych wykorzystano następujące schematy rozwiązywania problemów o charakterze matematycznym [MoPa70][Jers81][Lynd87] [Logi87][Demi98][Tiur98][Indr05][Huza07]:

- logika matematyczna:
  - sprawdzanie, czy podana formuła jest tautologią;
  - wyznaczanie wartości zmiennych dla których funkcja jest spełniona;
  - sprawdzanie, czy podany zbiór dysjunktów jest sprzeczny;
  - wyszukiwanie zbioru implikantów funkcji logicznych;
  - wyszukiwanie zbioru implicentów funkcji logicznych;
  - udowadnianie konsekwencji logicznej – dowodzenie twierdzeń w rachunku zdań;
- analiza kombinatoryczna grafów metodami symbolicznymi [Deo80]:
  - kolorowanie grafu niezorientowanego;
  - wyszukiwanie maksymalnych niezależnych zbiorów wierzchołków grafu;
  - wyszukiwanie zbiorów dominujących;
  - wyznaczanie pokryć klikowych grafu.

## 2.3. Zadania

Aby zrealizować zamierzone cele pracy doktorskiej i udowodnić postawioną tezę, określono następujące zadania szczegółowe:

1. Zapoznanie się z literaturą przedmiotu dotyczącą wnioskowania symbolicznego Gentzena.
2. Opracowanie metody oraz projekt i implementacja komputerowego systemu wnioskującego według reguł Gentzena.
3. Zaproponowanie usprawnień systemu wnioskującego i ich implementacja.
4. Przeprowadzenie badań efektywnościowych wprowadzonych usprawnień.
5. Dostosowanie systemu wnioskującego do wspomagania projektowania układów sterowania binarnego.
6. Zaproponowanie obszarów wykorzystania systemu wnioskującego w projektowaniu układów sterowania binarnego.
7. Przedstawienie możliwości wykorzystania programu wnioskującego w uniwersyteckich dedykowanych programach CAD.
8. Opracowanie formy integracji programu wnioskującego z profesjonalnymi systemami do symulacji i syntezy układów cyfrowych.

Zadanie **1** dotyczy przeglądu literatury przedmiotu oraz poszukiwania istniejących systemów i narzędzi opracowywanych przez informatyków jak i logików, dotyczących komputerowej realizacji algorytmu wnioskowania symbolicznego Gentzena. Następstwem realizacji zadania **1** jest zadanie **2,** wynikiem, którego jest koncepcja oraz pierwsza implementacja systemu wnioskującego według reguł Gentzena. Na osiągnięcie głównego celu pracy pozwala realizacja zadania **3,** a jego potwierdzenie będzie stanowiło zadanie **4.** Zadania **5,6,7,8** pozwalają na osiągniecie drugiego celu pracy, a także uzasadniają postawioną tezę oraz potwierdzają popierające ją tezy szczegółowe.

# 3. Rachunek sekwentów Gentzena

Pojęcie rachunku sekwentów obecnie używa się w odniesieniu do obszernej grupy systemów dedukcyjnych o zbliżonym charakterze. Ze względu na nazwisko autora systemy te często nazywane są systemami gentzenowskimi i wywodzą się z badań Gentzena [Gent69][Gent80][ErPa84][Tiur98][Huza07] nad utworzeniem formalizmów alternatywnych do systemów aksjomatycznych. W wyniku prowadzonych przez Gentzena badań powstał system automatycznej dedukcji oraz system rachunku sekwentów. Poszukiwanie pierwszego było zasadniczym celem prac Gentzena, natomiast drugi stanowił tylko pomocnicze narzędzie przeznaczone do udowadniania systemu naturalnej dedukcji z systemem aksjomatycznym. Jednak udowodnione przez Gentzena twierdzenie o eliminacji cięcia oraz rachunek sekwentów stanowiły podstawę do skonstruowania pierwszych systemów automatycznego dowodzenia [Wang60][Pawl65] [Indr06] oraz otworzyły drogę do rozwoju nowoczesnej teorii dowodu.

## 3.1. Wprowadzenie

Rozpatrywane są sekwenty w logice zdań, dla których przyjęty został następujący alfabet (3-1):

$$\delta = \delta_1 \cup \delta_2 \cup \delta_3 \cup \delta_4 \qquad \textbf{(3-1)}$$

gdzie $\delta_1 = \{ P_0, P_1, \ldots \}$, $\delta_2 = \{ \neg, \vee, \wedge, \rightarrow, \equiv \}$, $\delta_3 = \{ (, ), , \}$, $\delta_4 = \{ \vdash \}$.

***Definicja 1 Sekwent***

Sekwentem nazywa się uporządkowaną parę $(\Gamma, \Delta)$ skończonych zbiorów formuł $\Gamma=\{A_1,A_2,\ldots,A_m\}$, $\Delta=\{B_1,B_2,\ldots,B_n\}$. W tradycyjnym ujęciu zbiory te uporządkowane są jako ustalone ciągi formuł. Zamiast pisać $(\Gamma, \Delta)$, stosuje się notację $\Gamma\vdash\Delta$. $\Gamma$ to poprzednik, a $\Delta$ to następnik sekwentu. Zarówno poprzednik jak i następnik mogą być ciągami pustymi. Intuicyjnie definiując sekwent $A_1,A_2,\ldots,A_m \vdash B_1,B_2,\ldots,B_n$ podaje się, że jest on spełniony dla wartościowań $v$, wtedy i tylko wtedy, gdy dla tych samych wartościowań formuła $(A_1 \wedge A_2 \wedge \ldots \wedge A_m) \rightarrow (B_1 \vee B_2 \vee \ldots \vee B_n)$ jest spełniona. W przyjętej notacji symbol „$\rightarrow$” oznacza implikację, symbol „$\vee$” dysjunkcję (alternatywę niewyłączającą), natomiast symbol „$\wedge$” jest symbolem koniunkcji.

***Definicja 2 Formuła***

Formuła jest częścią sekwentu składającą się z symboli zdaniowych i spójników logicznych. **1)** Każda zmienna zdaniowa jest formułą. **2)** Jeśli $A$ i $B$ są formułami, to $\neg A$, $(A \wedge B)$, $(A \vee B)$, $(A \rightarrow B)$, $(A \equiv B)$ też są formułami. **3)** Nie ma innych formuł oprócz skonstruowanych według (**1**) i (**2**).

Podformułą formuły $A$ nazywa się dowolne posłowo słowa $A$, samo będące formułą.

***Definicja 3 Formuła atomowa***

Szczególnym przypadkiem formuły jest formuła elementarna (atomowa), składająca się tylko z jednego symbolu zdaniowego.

***Definicja 4 Rachunek sekwentów***

Rachunek sekwentów składa się z jednego schematu sekwentu aksjomatycznego oraz zbioru reguł pozwalających na dedukcję nowego sekwentu (sekwentu-wniosku) z pary sekwentów lub pojedynczego sekwentu (sekwenty-przesłanki).

***Definicja 5 Schemat aksjomatów***

Ze względu na brak reguły osłabiania przyjęto następujący schemat aksjomatów: $\Lambda,A|\text{-}\Theta,A$. Poszczególne reguły wnioskowania szczegółowo opisano w rozdziale 3.2.

***Definicja 6 Reguła wnioskowania***

Regułą wnioskowania nazywa się wyrażenie postaci $\frac{\Sigma_1;\ldots;\Sigma_k}{\Sigma}$, gdzie $\Sigma_1,..\Sigma_k$, $\Sigma$ są dowolnymi sekwentami. $\Sigma$ nazywa się bezpośrednim wnioskiem z $\Sigma_1,..\Sigma_k$ według danej reguły wnioskowania.

***Definicja 7 Aksjomat***

Aksjomatem nazywamy wyrażenie otrzymane ze schematu aksjomatów poprzez podstawienie w miejsce symbolu $A$ konkretnej formuły.

***Definicja 8 Dowód w rachunku sekwentu***

Dowodem w rachunku sekwentów jest skończony ciąg sekwentów $\Sigma_1,..\Sigma_k$ taki, że dla każdego $i$ ($1 \leq i \leq k$) $\Sigma_i$ jest albo aksjomatem, albo bezpośrednim wnioskiem z sekwentów poprzedzających $\Sigma_i$, wedle reguł opisanych w rozdziale 3.2.

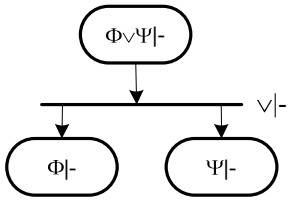
***Definicja 9 Sekwent wyprowadzalny***

Sekwent $\Sigma$ nazywa się wyprowadzanym w rachunku sekwentów, jeśli istnieje dowód w rachunku sekwentów, którego ostatnim elementem jest $\Sigma$.

W rachunku sekwentów zachodzi własność polegająca na odwracalności reguł wnioskowania, gdzie przesłanki dedukowane są z wniosku reguły. W praktyce dowód najczęściej konstruuje się zaczynając od sekwentu dowodzonego, systematycznie poszukując dla niego możliwych przesłanek.

W wyniku procesu wnioskowania, zgodnie z definicją 8, powstaje drzewo dowodu. W pracy doktorskiej proponuje się zastąpienie klasycznej reprezentacji drzewa dowodu interpretowalną siecią Petriego. Miejsce wejściowe dla tranzycji będzie reprezentował sekwent dowodzony, a miejsca wyjściowe będą reprezentowały sekwenty przesłanki.

Tranzycja odpowiada zastosowanej regule wnioskowania i dodatkowo będzie oznaczana symbolem sekwentu |- oraz zlokalizowanym względem niego głównym spójnikiem występującym w rozpatrywanym sekwencie (Rys. 3-1).




***Rys. 3-1 Proponowana reprezentacja drzewa dowodu***

W omawianej wersji rachunku sekwentów Gentzena rozpatruje się pięć spójników logicznych:

- Negacja „¬” „/”;
- Dysjunkcja „∨” „+”;
- Koniunkcja „∧” „*”;
- Implikacja „→” „->”;
- Równoważność „≡” „<->”.

Ze względu na komputerowe przetwarzanie sekwentów, obok tradycyjnej formy, w pracy stosuje się również alternatywny wariant oznaczeń spójników logicznych [Adam90], łatwiejszy do wprowadzania z klawiatury komputerowej.

Najważniejsze, wartościowe cechy systemu gentzenowskiego to:

- intuicyjność (forma zbliżona do reguł produkcji IF …THEN) [Lige03];
- skalowalność (system może zawierać podstawowy zbiór reguł i zostać rozszerzony o reguły specjalizowane) [ErPa84];
- sprawność (dla znacznej liczby problemów złożoność obliczeniowa zależy tylko liniowo lub wielomianowo od liczby zmiennych logicznych) [Indr06];
- rozproszenie (można opracować wersję na systemy obliczeniowe rozproszone) [Adam90];
- precyzyjny, intuicyjny, czytelny dla człowieka dowód twierdzeń [Adam90][Pawl65][Wang60][Jers81][Indr06][Huza07].

Główna zaleta dowodów w rachunku sekwentów jest następstwem własności podformuł [Indr][Indr06], ponieważ wszystkie formuły występujące w przesłance

dowolnej reguły są podformułami formuł występujących w konkluzji [Tiur98]. W dowodzie sekwentu $|-\Psi\varphi$ mamy do czynienia tylko z podformułami $\Psi$. Dla danej formuły $\Psi$, łatwiej jest znaleźć dowód w sensie Gentzena niż stosując inne systemy dowodzenia. Dlatego systemy zbliżone do rachunku sekwentów znajdują zastosowanie w komputerowym dowodzeniu twierdzeń.

## 3.2. Eliminacja spójników logicznych

W systemie Gentzena zastosowano dziesięć reguł wnioskowania [Adam90], które zostały omówione w niniejszym rozdziale. Dla każdego spójnika zostały podane dwie reguły jego eliminowania. Jedna reguła jest stosowana, gdy eliminowany spójnik występuje w poprzedniku, druga zaś, gdy eliminowany spójnik znajduje się w następniku.

Reguły stosowane są zawsze do spójnika głównego formuły nieelementarnej w redukowanym sekwencie. Proces eliminowania powtarzany jest tak długo, aż otrzymane zostaną same sekwenty znormalizowane, to jest sekwenty nie zawierające żadnych spójników logicznych. Sekwenty znormalizowane noszą również nazwę sekwentów atomowych [Indr06].

### 3.2.1. Negacja

***Eliminacja spójnika negacji w następniku***

Jeżeli głównym spójnikiem sekwentu jest negacja w następniku, to formułę będąca argumentem tej negacji zostaje przeniesiona do poprzednika (3-2).

$$\frac{\Lambda \mid\!- \Theta,\ \neg\Phi,\ \Psi}{\Lambda,\ \Phi \mid\!- \Theta,\ \Psi} \qquad \textbf{(3-2)}$$

Na przykład stosując powyższą regułę do sekwentu:

$$p \mid\!- \neg(r \wedge s),q$$

otrzymuje się wyrażenie:

$$p,(r \wedge s) \mid\!- q$$

W przedstawionym przykładzie $\Lambda$ jest ciągiem składającym się z jednej litery $p$, $\Theta$ – jest ciągiem pustym, $\Phi$ ma postać $(r \wedge s)$, oraz $\Psi$ jest również formułą elementarną $q$.

***Eliminacja spójnika negacji w poprzedniku***

Jeżeli głównym spójnikiem sekwentu jest negacja w poprzedniku, to formuła będąca argumentem tej negacji zostaje przeniesiona do następnika (3-3).

$$\frac{\Lambda,\ \neg\Phi,\ \Gamma \vdash \Theta,\ \Psi}{\Lambda,\ \Gamma \vdash \Theta,\ \Psi,\ \Phi} \qquad \textbf{(3-3)}$$

Jeżeli sekwent ma postać:

$$p,\neg(p \vee q) \vdash (r \wedge s)$$

w wyniku jego redukcji powstanie wyrażenie:

$$p \vdash (r \wedge s),(p \vee q)$$

### 3.2.2. Dysjunkcja

***Eliminacja spójnika dysjunkcji (alternatywy niewyłączającej) w następniku***

Jeżeli głównym spójnikiem sekwentu jest dysjunkcja w następniku, to zastępuje się ją przecinkiem (3-4).

$$\frac{\Lambda \vdash \Theta,\ \Phi \vee \Psi,\ \Gamma}{\Lambda \vdash \Theta,\ \Phi,\ \Psi,\ \Gamma} \qquad \textbf{(3-4)}$$

Na przykład stosując do sekwentu:

$$p \vdash r,(p \vee q),s$$

powyższą regułę, otrzymuje się:

$$p \vdash r,p,q,s$$

***Eliminacja spójnika dysjunkcji (alternatywy niewyłączającej) w poprzedniku***

Jeżeli głównym spójnikiem sekwentu jest dysjunkcja w poprzedniku, to sekwent ten zastępuje się dwoma sekwentami, z których pierwszy będzie zawierał lewy argument alternatywy, a drugi prawy argument alternatywy (3-5).

$$\frac{\Theta,\ \Phi \vee \Psi,\ \Gamma \vdash \Pi}{\Theta,\ \Phi,\ \Gamma \vdash \Pi;\quad \Theta,\ \Psi,\ \Gamma \vdash \Pi} \qquad (3\text{-}5)$$

Na przykład z sekwentu:

$$p,\ (p \vee q),(p \wedge s) \vdash r$$

po zastosowaniu powyższej reguły, uzyskuje się sekwent:

$$p,p,(p \wedge s) \vdash r$$

oraz:

$$p,q,(p \vee s) \vdash r$$

### 3.2.3. Koniunkcja

***Eliminacja spójnika koniunkcji w następniku***

Jeżeli głównym spójnikiem sekwentu jest koniunkcja w następniku, to sekwent ten zastępuje się dwoma sekwentami, z których pierwszy będzie zawierał lewy argument koniunkcji, a drugi prawy argument koniunkcji (3-6).

$$\frac{\Lambda \vdash \Theta,\ \Phi \wedge \Psi,\ \Gamma}{\Lambda \vdash \Theta,\ \Phi,\ \Gamma;\quad \Lambda \vdash \Theta,\ \Psi,\ \Gamma} \qquad (3\text{-}6)$$

Na przykład sekwent:

$$p,q \vdash r,((p \vee q) \wedge r),r$$

można zastąpić sekwentem:

$$p,q \vdash r,(p \vee q),r$$

oraz:

$$p,q \vdash r,r,r$$

***Eliminacja spójnika koniunkcji w poprzedniku***

Jeżeli głównym spójnikiem sekwentu jest koniunkcja w poprzedniku, to symbol koniunkcji zastępowany jest przecinkiem (3-7).

$$\frac{\Theta,\ \Phi \wedge \Psi,\ \Gamma \vdash \Pi}{\Theta,\ \Phi,\ \Psi,\ \Gamma \vdash \Pi} \qquad (3\text{-}7)$$

Na przykład na podstawie powyższej reguły z sekwentu:

$$p,(q \wedge r),(p \vee s) \vdash \neg(p \wedge s)$$

otrzymuje się:

$$p,q,r,(p \vee s) \vdash \neg(p \wedge s)$$

### 3.2.4. Implikacja

***Eliminacja spójnika implikacji w następniku***

Jeżeli głównym spójnikiem sekwentu jest implikacja w następniku, to pierwszy człon implikacji przenosi się do poprzednika (3-8).

$$\frac{\Lambda \vdash \Theta,\ \Phi \rightarrow \Psi,\ \Gamma}{\Lambda,\ \Phi \vdash \Theta,\ \Psi,\ \Gamma} \qquad (3\text{-}8)$$

Na przykład w sekwencie:

$$p \vdash r,p \rightarrow q,r$$

stosując powyższą regułę, otrzymuje się:

$$p,p \vdash r,q,r$$

***Eliminacja spójnika implikacji w poprzedniku***

Jeżeli głównym spójnikiem sekwentu jest implikacja w poprzedniku, to sekwent ten zastępuje się dwoma sekwentami, z których pierwszy zawiera drugi człon implikacji w poprzedniku, a drugi pierwszy człon implikacji w następniku (3-9).

$$\frac{\Theta,\ \Phi \rightarrow \Psi,\ \Gamma \vdash \Pi}{\Theta,\ \Psi,\ \Gamma \vdash \Pi;\quad \Theta,\ \Gamma \vdash \Pi,\ \Phi} \qquad (3\text{-}9)$$

Na przykład z sekwentu:

$$p, p \rightarrow q \vdash r$$

otrzymuje się:

$$p, q \vdash r$$

oraz:

$$p \vdash r, p$$

### 3.2.5. Równoważność

***Eliminacja spójnika równoważności w następniku***

Jeżeli głównym spójnikiem sekwentu jest równoważność w następniku, to sekwent ten zastępuje się dwoma sekwentami, z których pierwszy zawiera w poprzedniku drugi człon równoważności i w następniku pierwszy człon równoważności, a drugi zawiera w poprzedniku pierwszy człon równoważności i w następniku drugi człon równoważności (3-10).

$$\frac{\Lambda \vdash \Theta,\ \Phi \equiv \Psi,\ \Gamma}{\Lambda,\ \Psi \vdash \Theta,\ \Phi,\ \Gamma;\quad \Lambda,\ \Phi \vdash \Theta,\ \Psi,\ \Gamma} \qquad (3\text{-}10)$$

Na przykład stosując powyższą regułę do sekwentu:

$$p \vdash p \equiv q, r$$

otrzymuje się:

$$p, q \vdash p, r$$

oraz:

$$p, p \vdash q, r$$

***Eliminacja spójnika równoważności w poprzedniku***

Jeżeli głównym spójnikiem sekwentu jest równoważność w poprzedniku, to sekwent ten zastępuje się dwoma sekwentami, z których pierwszy zawiera w poprzedniku obydwa argumenty równoważności, a drugi w następniku te argumenty równoważności (3-11).

$$\frac{\Theta,\ \Phi \equiv \Psi,\ \Gamma \vdash \Pi}{\Theta,\ \Phi,\ \Psi,\ \Gamma \vdash \Pi;\quad \Theta,\ \Gamma \vdash \Pi,\ \Phi,\ \Psi} \qquad \textbf{(3-11)}$$

Na przykład sekwent:

$$p, p \equiv q \vdash r$$

po zastosowaniu powyższej reguły da w wyniku:

$$p, p, q \vdash r$$

oraz:

$$p \vdash r, p, q$$

## 3.3. Tautologie, sekwenty spełniane i sekwenty sprzeczne

Jedną z ważniejszych reguł systemu wnioskującego, dzięki której możliwe jest dowodzenie twierdzeń, jest reguła testowania relacji wynikania [Lynd87] pomiędzy formułami w sekwencie [Gent80][Gall86]. W wyniku normalizacji złożonego sekwentu, opisującego na przykład dyskretne zjawisko fizyczne lub proces w układzie sterowania binarnego, otrzymuje się równoznaczny zbiór rozwiązań, składający się z grupy prostszych sekwentów, definiujących różnorodne relacje w postaci aksjomatów teorii specyficznej [Evek85][Evek87].

Jeżeli w następniku i w poprzedniku danego wyrażenia występuje identyczna formuła, to całe wyrażenie jest tautologią (jest logicznie prawdziwe, niezależnie od wartości zmiennych) i dlatego nie musi być ono poddawane dalszej redukcji (3-12), aby to potwierdzić.

$$\Theta,\ \Phi,\ \Psi,\ \Gamma \vdash \Pi,\ \Psi; \qquad \textbf{(3-12)}$$

Przykłady:

$$p, (r \wedge s) \vdash p, q$$

$$p, (r \vee s) \vdash q,\ (r \vee s)$$

Badanie tautologii ma bardzo znaczący wpływ na cały proces normalizacji. Odpowiednio wczesne wykrycie zależności, które są zawsze prawdziwe i nie mają wpływu

na wartość logiczną badanego sekwentu, pozwala na znaczne skrócenie procesu wnioskowania.

Warto zwrócić uwagę, że podczas wykorzystania systemu wnioskującego do dowodzenia twierdzeń, otrzymanie już pierwszego rozwiązania (sekwentu znormalizowanego), które nie jest tautologią, stanowi formalny dowód, że analizowane twierdzenie nie jest zawsze prawdziwe.

## 3.4. Reguły dopuszczalne

Istnieje możliwość rozbudowywania systemu wnioskującego przez wprowadzanie nowych reguł wywodzących się z logiki matematycznej, opierających się na algebrze Boole'a, lub stanowiących złożenie znanych już wcześniej podstawowych reguł wnioskowania (3-13).

***Definicja 10 Reguła dopuszczalna***

Regułę $\frac{\Sigma_1;\ldots;\Sigma_k}{\Sigma}$ nazywamy dopuszczalną w rachunku sekwentów, jeśli z wyprowadzalności w rachunku sekwentów $\Sigma_1,..\Sigma_k$ wynika wyprowadzalność sekwentu $\Sigma$. Reguły dopuszczalne w formalny sposób zdefiniowano w pracach [BiWo93] [ŁaMa04][Indr06].

Przykłady reguł dopuszczalnych:

a)

$$\frac{\Gamma \vdash \Phi}{\Psi, \Gamma \vdash \Phi} \qquad \textbf{(3-13)}$$

b)

$$\frac{\Psi, \Psi, \Gamma \vdash \Phi}{\Psi, \Gamma \vdash \Phi} \qquad \textbf{(3-14)}$$

c)

$$\frac{\Gamma, \Psi, \Theta, \Gamma_1 \vdash \Phi}{\Gamma, \Theta, \Psi, \Gamma_1 \vdash \Phi} \qquad \textbf{(3-15)}$$

d)

$$\frac{\Gamma \vdash \Phi \quad \Phi, \Gamma_1 \vdash \Psi}{\Gamma, \Gamma_1 \vdash \Psi} \qquad (3\text{-}16)$$

Wymienione reguły inferencyjne można udowodnić ręcznie lub za pomocą opracowanego systemu komputerowego. Poniżej omówiono szczegółowo reguły dopuszczalne, które zostały wprowadzone do autorskiego systemu wnioskującego.

### 3.4.1. Skracanie

Reguła skracania umożliwia uproszczenie sekwentu poprzez złączenie grupy identycznych formuł w jedną. Jeżeli w poprzedniku lub w następniku sekwentu występuje kilka takich samych formuł, to wszystkie wystąpienia danej formuły zostaną zastąpione jedną formułą (3-17).

$$\frac{\Theta, \Phi, \Psi, \Gamma, \Psi \vdash \Pi, \Pi}{\Theta, \Phi, \Psi, \Gamma \vdash \Pi} \qquad (3\text{-}17)$$

Przykład:

$$p,(r \rightarrow s),p \vdash q \qquad \Rightarrow \qquad p,(r \rightarrow s) \vdash q$$

### 3.4.2. Dominacja

Jeżeli formuły poprzednika i następnika jednego z sekwentów (sekwentu dominującego) zawierają się w poprzedniku i następniku drugiego sekwentu (sekwentu zdominowanego), to sekwent zdominowany zostanie odrzucony z dalszego procesu normalizacji (3-18) [Adam90][BiWo93][Szaj96].

$$\frac{\Theta, \Phi, \Psi, \Gamma \vdash \Pi \qquad\qquad \Theta, \Gamma \vdash \Pi}{\Theta, \Gamma \vdash \Pi} \qquad (3\text{-}18)$$

Reguła dominacji redukuje bazę rozwiązań (zbiór aksjomatów specyficznych teorii [Evek85][Evek87][Adam90]) poprzez odrzucenie sekwentów nadmiarowych i pozostawienie sekwentów, które w pełni opisują dotychczas uwzględnione relacje w analizowanym modelu, odzwierciedlającym badane zjawisko w systemie dyskretnym.

### 3.4.3. Cięcie

Na podstawie dwóch sekwentów możliwe jest utworzenie jednego sekwentu prostszego, jeżeli dwa sekwenty zawierają tę samą formułę po przeciwnych stronach znaku

wynikania logicznego. Poprzednik nowego sekwentu będzie składał się ze wszystkich formuł wziętych z poprzedników dwóch sekwentów, z wykluczeniem wcześniej wymienionej formuły. Następnik zaś zbudowany będzie ze wszystkich formuł występujących w następnikach sekwentów, również z pominięciem formuły wspólnej (3-19).

$$\frac{\Gamma 1,\ \Phi \vdash \Theta 1 \qquad\qquad \Gamma 2 \vdash \Phi,\ \Theta 2}{\Gamma 1,\ \Gamma 2 \vdash \Theta 1,\ \Theta 2} \tag{3-19}$$

Formuła $\Phi$ nazywana jest formułą wycinaną. Sekwent $\Gamma 1,\ \Gamma 2 \vdash \Theta 1,\ \Theta 2$ nazywany consensusem jest przecięciem sekwentów $\Gamma 1,\ \Phi \vdash \Theta 1$ *i* $\Gamma 2 \vdash \Phi,\ \Theta 2$.

W teorii układów logicznych cięcie odpowiada sklejaniu lub uogólnionemu sklejaniu wyrażeń boolowskich [Adam90].

### 3.4.4. Reguły dedykowane

Ze względu na proponowane zastosowanie systemu wnioskowania do rozwiązywania problemów dotyczących techniki cyfrowej, zdecydowano się na wprowadzenie nowej, dodatkowej reguły eliminacji spójnika alternatywy wyłączającej *XOR*. Reguła ta jest afirmacją reguły równoważności odpowiadającej spójnikowi *XNOR* (3-20).

$$\frac{\vdash \Gamma \oplus \Phi}{\Gamma \equiv \Phi \vdash} \qquad \frac{\Gamma \oplus \Phi \vdash}{\vdash \Gamma \equiv \Phi} \tag{3-20}$$

Proponuje się, aby najczęściej spotykany w literaturze symbol spójnika *XOR* „$\oplus$”, ze względu na komputerowe przetwarzanie, przyjął akceptowalną formę tekstową „<+>”.

Wprowadzane reguły są przykładami użytecznych reguł dopuszczalnych zaproponowanych przez użytkownika systemu wnioskowania. Reguły tego rodzaju nazywane są również regułami wtórnymi [Indr06].

#### *Eliminacja alternatywy wyłączającej w następniku*

Jeżeli głównym spójnikiem sekwentu jest alternatywa wyłączająca w następniku, to sekwent ten zastępuje się dwoma sekwentami, z których pierwszy zawiera w następniku, a drugi w poprzedniku obydwa argumenty sumy wyłączającej oddzielone przecinkiem (3-21).

$$\frac{\Lambda \vdash \Theta,\ \Phi \oplus \Psi,\ \Gamma}{\Lambda \vdash \Theta,\ \Phi,\ \Psi,\ \Gamma;\quad \Lambda,\ \Phi,\ \Psi \vdash \Theta,\ \Gamma} \tag{3-21}$$

Na przykład stosując powyższą regułę do sekwentu:

$$p \vdash p \oplus q,r$$

otrzymuje się:

$$p \vdash p,\ q,\ r$$

oraz:

$$p,\ p,\ q \vdash r$$

***Eliminacja spójnika alternatywy wyłączającej i w poprzedniku***

Jeżeli głównym spójnikiem sekwentu jest alternatywa wyłączająca w poprzedniku, to sekwent ten zastępuje się dwoma sekwentami, z których pierwszy w poprzedniku zawiera drugi argument alternatywy wyłączającej, a w następniku pierwszy. Drugi sekwent w poprzedniku zawiera pierwszy argument alternatywy wyłączającej, a w następniku drugi (3-22).

$$\frac{\Theta,\ \Phi \oplus \Psi,\ \Gamma \vdash \Pi}{\Theta,\ \Psi,\ \Gamma \vdash \Pi,\ \Phi;\quad \Theta,\ \Phi,\ \Gamma \vdash \Pi,\ \Psi} \qquad \textbf{(3-22)}$$

Na przykład sekwent:

$$p,\ p \oplus q \vdash r$$

po zastosowaniu powyższej reguły da w wyniku:

$$p,\ q \vdash r,\ p$$

oraz:

$$p,\ p \vdash r,\ q$$

Dowód poprawności przeprowadzono przy pomocy systemu wnioskującego będącego systemem dowodzenia twierdzeń, poprzez przyjęcie założenie, że suma wyłączająca jest negacją równoważności (3-23).

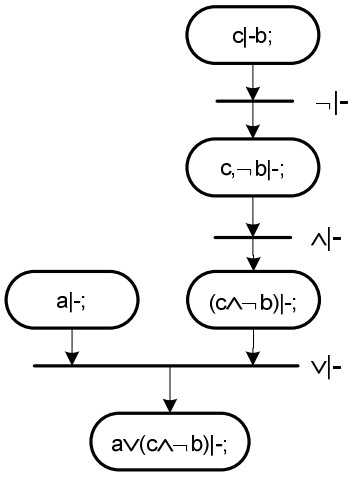
$$\vdash (\Phi \oplus \Psi) \equiv \neg(\Phi \equiv \Psi) \qquad \textbf{(3-23)}$$

W wyniku normalizacji według reguł Gentzena wszystkie otrzymane sekwenty są tautologiami (są zawsze prawdziwe), więc przyjęte założenie jest również zawsze prawdziwe. Dowód komputerowy zawiera 9 węzłów i trwa około 0.016 sekundy.

## 3.5. Odwracalność reguł wnioskowania

System Gentzena rozpatruje się także w ujęciu przeciwnym do eliminacji, czyli w ujęciu wprowadzania spójników logicznych [Adam90][ŁaMa04]. Takie zastosowanie umożliwia przejście z postaci znormalizowanej (aksjomatów) do postaci złożonej (wniosku). Wprowadzanie spójników logicznych wykonuje się poprzez odwrotne zastosowanie reguł eliminacji. Ujęcie to wykorzystywane jest w pracy do transformacji znormalizowanego opisu na postać koniunkcyjną, dysjunkcyjną lub implikacyjną.

Rozpatrując przykład, gdzie na podstawie uzyskanej bazy wyników z procesu normalizacji ($a|-;$ i $c|-b;$), pokazano jak przejść do standardowej postaci minimalnej zgodnie z Rys. 3-2, która będzie wyglądała następująco: $a \vee (c \wedge \neg b)|-;$ Proces wprowadzania spójników logicznych, udokumentowano w postaci interpretowanej sieci Petriego.




***Rys. 3-2 Przejście od sekwentów przesłanek do sekwentu wniosku***

Przykładowo, grupę sekwentów $A1,B1|-Y;$ $A2,B2|-Y;$ $A3,B3|-Y;$ opisujących zależności między sygnałami wejściowymi a sygnałem wyjściowym, dla tego samego modelu, można zastąpić jednym sekwentem, który przyjmie postać dysjunkcyjną $A1 \wedge B1 \vee A2 \wedge B2 \vee A3 \wedge B3|-Y$ [Adam90].

W pracy wykorzystuje się reguły wnioskowania do równoważnościowej transformacji specyfikacji układów – stopniowym zastępowaniu kilku sekwentów jednym sekwentem złożonym. Przykłady takiego postępowania zamieszczono w rozdziale 5.3, gdzie stosuje się transformację opisu sekwentowego na języki opisu sprzętu (HDL), a także na format ESPRESSO (rozdz. 5.4).

Jeśli dysponuje się relacyjnym opisem bramki AND w znormalizowanej formie bezspójnikowej (Tab. 3-1), to można wykorzystać zarówno postać koniunkcyjną, jak i dysjunkcyjną opisującą jej funkcjonowanie.

***Tab. 3-1 Specyfikacja bramki AND***

| Relacyjny opis bramki | Postać koniunkcyjna | Postać dysjunkcyjna |
|---|---|---|
| x1,x2\|-y;<br>y\|-x1;<br>y\|-x2; | x1*x2\|-y; | y\|-x1+x2; |

# 4. Algorytm redukcji zbioru sekwentów

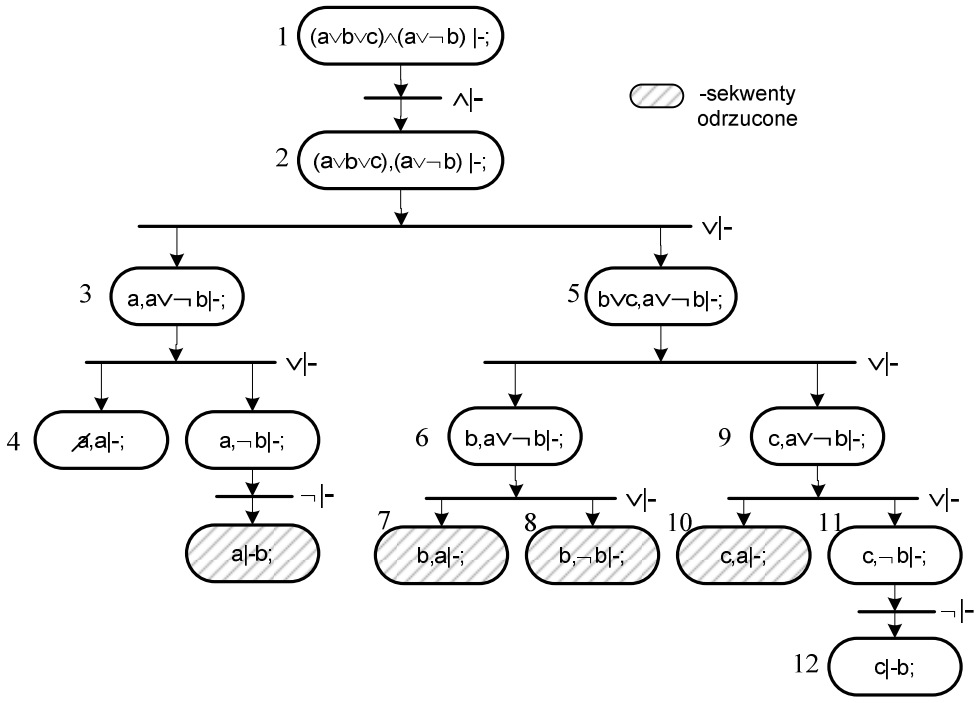
Algorytm Gentzena podczas wnioskowania wyznacza rozwiązania dokładne, falsyfikujące sekwent. Często wyznaczone relacje w badanym zbiorze rozwiązań mają charakter nadmiarowy i można je zastąpić nowymi, prostszymi relacjami lub po prostu odrzucić, gdyż nie mają one wpływu logicznego na analizowany sekwent. W szczególności, jedne relacje ze zbioru rozwiązań dominują nad innymi, co sprawia, że relacje zdominowane można pominąć.

## 4.1. Uporządkowana eliminacja spójników logicznych

Eliminacja spójników logicznych w systemie wnioskującym metodą Gentzena realizowana jest w porządku DFS [Deo80]. W każdym przypadku, gdy w wyniku eliminacji spójnika logicznego powstają dwa nowe sekwenty, to pierwszy sekwent będzie analizowany dalej, a drugi zostanie odłożony na stos i będzie oczekiwał tak długo, aż pierwszy zostanie całkowicie znormalizowany lub rozpatrywana gałąź zostanie odcięta.

Automatyczny system wnioskujący wykonuje nie tylko pojedyncze kroki redukcji, ale również makro-kroki Rys. 4-1, będące rezultatem łącznego zastosowania na przykład reguły eliminacji dysjunkcji, reguły skracania i reguły wykrywania tautologii.

Sekwent (4) z Rys. 4-1 powstał w wyniku eliminacji spójnika dysjunkcji, występującego po lewej stronie znaku wynikania logicznego w sekwencie 3 połączonej z jednoczesnym sklejeniem zmiennej „*a*”.




***Rys. 4-1 Graficzna reprezentacja normalizacji sekwentu***

Śledząc przebieg normalizacji, przedstawiony siecią Petriego na Rys. 4-1, można zaobserwować:

- zastosowanie reguły dominacji, gdy sekwenty „*a,b*|-;”, „*a,¬b*|-;”, „*b,a*|-;”, „*c,a*|-;” zostają zdominowane przez sekwent „*a*|-;” i odrzucone,
- skracanie sekwentu, gdy sekwent „*a,a*|-;” zastąpiony zostaje przez „*a*|-;”, po którym następuje zastosowanie dominacji w sąsiedniej gałęzi,
- tautologię „*b,¬b*|-;”, czyli sekwent „*b*|-*b;*”, który nie ma dalszego wpływu na spełnialność całego, badanego sekwentu złożonego.

Standardowy przebieg eliminacji spójników jest usprawniony przez priorytety eliminacji spójników ($\equiv$, $\otimes$, $\rightarrow$). Dzięki takiemu podejściu formuły szybciej przekształcone są do postaci koniunkcyjno-dysjunkcyjnej. Jest to korzystne podejście ze względu na wykorzystanie w systemie wnioskującym elementów metody Thelena-Mathony'ego [`Thel88`][`Math90`](rozdział 4.4.2).

## 4.2. Wczesne wykrywanie tautologii

Możliwe szybkie wykrywanie tautologii w analizowanym wyrażeniu ma znaczący wpływ liczbę uzyskanych rozwiązań, co jest następstwem odrzucenia wyrażeń zawsze prawdziwych, które nie mają wpływu na spełnialność logiczną rozpatrywanego wyrażenia. W rozwiązaniach opisywanych w literaturze zazwyczaj proponuje się testowanie tautologii dopiero na poziomie formuł atomowych. W pracy doktorskiej proponuje się odejście od analizy sekwentów tylko na poziomie formuł atomowych poprzez poszukiwanie tautologii praktycznie przy każdym zastosowaniu reguły eliminacji spójnika w rozpatrywanym sekwencie. Na przykład, w przypadku sekwentu częściowo znormalizowanego:

$$(A1 \wedge B1) \rightarrow Z,\ A2 \wedge B2,\ A3 \wedge B3 \mid\!- Y,\ (A1 \wedge B1) \rightarrow Z;$$

nie ma konieczności jego dalszej analizy, gdyż złożona formuła $(A1 \wedge B1) \rightarrow Z$ występuje po lewej i prawej stronie analizowanego sekwentu, co także odpowiada regule tautologii. Podczas próby znormalizowania przykładowego sekwentu do poziomu liści okaże się, że każdy następny potomny sekwent będzie również tautologią.

Podejście takie powoduje dużo wcześniejsze lokalizowanie złożonych tautologii, czego następstwem jest znaczne skrócenie drzewa dowodu, gdyż nie ma konieczności rozwijania gałęzi do poziomu liści (sekwentów znormalizowanych, nie zawierających spójników logicznych). Przykłady wyeliminowanych w ten sposób gałęzi zostały przedstawione w rozdziale 4.4.2.

## 4.3. Minimalizacja bazy rozwiązań

Wynikiem normalizacji sekwentów jest zbiór sekwentów atomowych. Minimalizacja bazy rozwiązań opiera się na dopuszczalnej regule dominacji i cięcia. W wyniku eliminacji spójników według reguł Gentzena sekwenty znormalizowane (nie zawierające spójników logicznych) trafiają do bazy rozwiązań. W momencie dodawania sekwent dodawany porównywany jest z sekwentami przechowywanymi w bazie. Jeżeli sekwent dodawany dominuje nad jakimś sekwentem lub sekwentami z bazy danych, to sekwenty zdominowane są odrzucane ze zbioru rozwiązań, a nowy sekwent jest do niego dodawany. Jeśli jakiś sekwent ze zbioru rozwiązań dominuje nad sekwentem dodawanym, to sekwent dodawany zostaje pominięty poprzez niedopisanie go do zbioru rozwiązań. Sposób ten sprawia, że w zbiorze rozwiązań będą znajdowały się sekwenty o możliwie najmniejszej liczbie formuł, jednak zdarza się, że nie będą one stanowiły minimalnej formy opisu rozpatrywanego problemu lub zjawiska opisanego relacjami logicznymi w formie sekwentów.

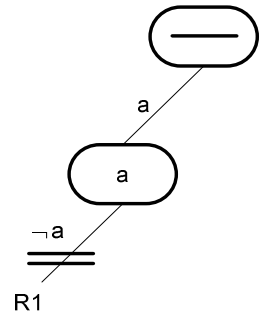
## 4.4. Wykorzystanie elementów metody Thelena-Mathony'ego

Metoda wyznaczania implikantów prostych Thelena [Thel88] bazuje na metodzie R. Nelsona [Nels55], który przedstawił, że możliwe jest wyznaczenie wszystkich implikantów prostych funkcji przy wykonaniu transformacji z postaci koniunkcyjnej do postaci dysjunkcyjnej. Metoda zaproponowana przez R. Nelsona jest bardzo czasochłonna, a także wymagająca dużej ilości pamięci, gdyż wszystkie pośrednie obliczenia muszą być przechowywane, a ich liczba rośnie wykładniczo [Nels55].

### 4.4.1. Zarys metody Thelena-Mathony'ego

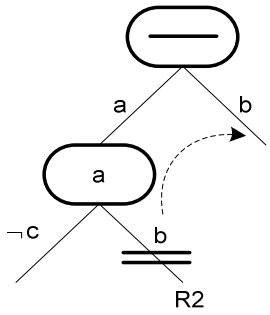
Thelen w swoim algorytmie zaproponował bardziej efektywne rozwiązanie, które również polega na transformacji funkcji z postaci koniunkcyjnej do postaci dysjunkcyjnej, jednakże zapotrzebowanie na pamięć jest liniowe. Wszystkie pośrednie obliczenia nie muszą być przechowywane w pamięci. Metoda ta polega budowaniu drzewa poszukiwań implikantów prostych, gdzie liście drzewa są implikantami prostymi lub implikantami, które zostaną pochłonięte przez wcześniej znalezione implikanty proste. Drzewo przeszukiwane jest według algorytmu DFS. W celu skrócenia drzewa poszukiwań stosuje się następujące reguły:

**R1** – łuk jest obcinany, jeżeli koniunkcja literałów dla poprzedniego wierzchołka zawiera uzupełnienie przypisane do danego łuku (Rys. 4-2);



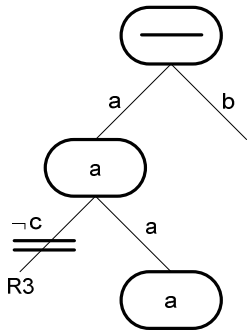

***Rys. 4-2 Reprezentacja graficzna reguły R1***

**R2** – łuk jest obcinany, jeżeli na wyższym poziomie istnieje łuk opisany tym samym literałem (Rys. 4-3);




***Rys. 4-3 Reprezentacja graficzna reguły R2***

**R3** – jeżeli z danego wierzchołka wychodzi łuk (łuki) opisany literałem występującym w koniunkcji opisującej dany wierzchołek, to obcinane są wszystkie, za wyjątkiem jednego, opisanego tym powtarzającym się literałem (Rys. 4-4).



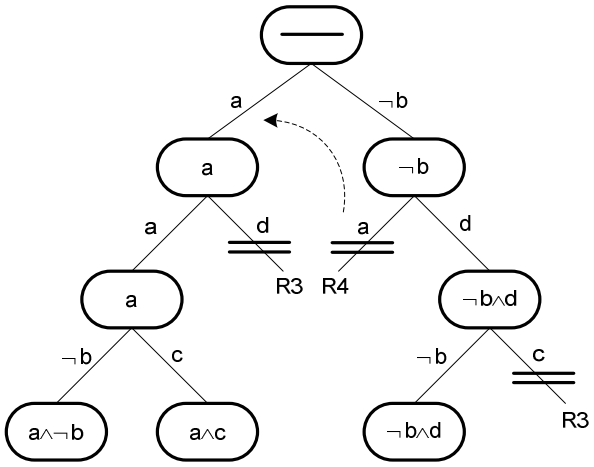

***Rys. 4-4 Reprezentacja graficzna reguły R3***

Reguły przedstawione powyżej bazują na następujących prawach algebry Boole'a (4-1):

$$
\begin{aligned}
a \wedge a &= a; & a \vee a &= a;\\
a \vee a \wedge b &= a; & a \wedge (a \vee b) &= a;\\
a \wedge \neg a &= 0; & a \vee \neg a &= 1;\\
a \wedge 0 &= 0; & a \vee 1 &= 1;\\
a \vee 0 &= a; & a \wedge 1 &= a;
\end{aligned}
\qquad \textbf{(4-1)}
$$

H.J. Mathony [Math90] zaproponował czwartą regułę skracania drzewa poszukiwań, która może zredukować drzewo nawet o 25%.

**R4** – łuk *j* jest obcinany, jeżeli rozwinięta jest już gałąź z łukiem *k* na wyższym poziomie, opisana takim samym literałem, oraz jeżeli reguła R2 nie została zastosowana w poddrzewie od łuku *k* w odniesieniu do łuku na tym samym poziomie co łuk *k*, który prowadzi do łuku *j* (Rys. 4-5).




***Rys. 4-5 Reprezentacja graficzna reguły R4***

Niestety reguła R4 w znacznym stopniu komplikuje algorytm, gdyż dodatkowo należy przechowywać informacje o wystąpieniu reguły R2. Zastosowanie niniejszej reguły zmniejsza prawdopodobieństwo pojawienia się w liściach drzewa implikantów niebędących implikantami prostymi. Ponadto nie gwarantuje ona, że implikanty takie nie będą się pojawiać [Kara07], więc nadal konieczne jest zastosowanie algorytmu sprawdzającego, tak jak ma to miejsce w przypadku użycia tylko trzech reguł.

Jako ważniejsze ograniczenia metody Thelena-Mathony'ego należy wymienić wymaganie postaci koniunkcyjnej dla analizowanego wyrażenia logicznego oraz brak możliwości stosowania innych spójników logicznych niż koniunkcja, dysjunkcja i negacja.

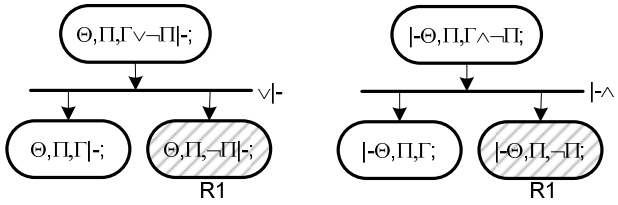
### 4.4.2. Metoda Thelena-Mathony'ego w ujęciu rachunku sekwentów

Znaczne skrócenie drzewa dowodu wnioskowania symbolicznego możliwe jest poprzez wprowadzenie do algorytmu elementów metody Thelena-Mathony'ego [Thel88][Math90]. Wnioskowanie symboliczne Gentzena, w przeciwieństwie do algorytmu Thelena-Mathony'ego, operuje na bardziej złożonych formułach, co jest następstwem większej ilości dopuszczalnych spójników logicznych, oraz brakiem wymagań odnośnie postaci normalizowanego wyrażenia. Każdy spójnik może zostać wyeliminowany na dwa sposoby, w zależności od jego umiejscowienia względem znaku wynikania logicznego „|-". Podczas eliminacji wyrażenie jest zstępowane jednym lub

maksymalnie dwoma wyrażeniami, dlatego drzewo dowodu jest niepełnym drzewem binarnym. W algorytmie Thelena analizuje się klauzule rozpatrując jeden lub więcej spójników jednocześnie, przez co budowane drzewo może mieć więcej niż dwa łuki wychodzące z jednego węzła. Różnice w konstrukcji drzew powodują najwięcej kłopotów w integracji obu metod. Problem integracji rozwiązano wprowadzając stosy podręczne. Reguły Thelena-Mathony'ego w postaci sekwentowej będą miały następującą postać:

**Reguła R1** – Jeżeli w wyniku eliminacji dowolnego spójnika, po lewej lub po prawej stronie sekwentu pojawi się formuła i jej afirmacja (Rys. 4-6), to można przerwać normalizację danego wyrażenia (4-2):

$$\Theta,\ \Phi,\ \Psi,\ \Gamma,\neg\Psi \vdash \Pi;\ \ lub\ \ \Theta,\ \Phi,\ \Gamma \vdash \Pi,\ \neg\Psi,\ \Psi; \tag{4-2}$$



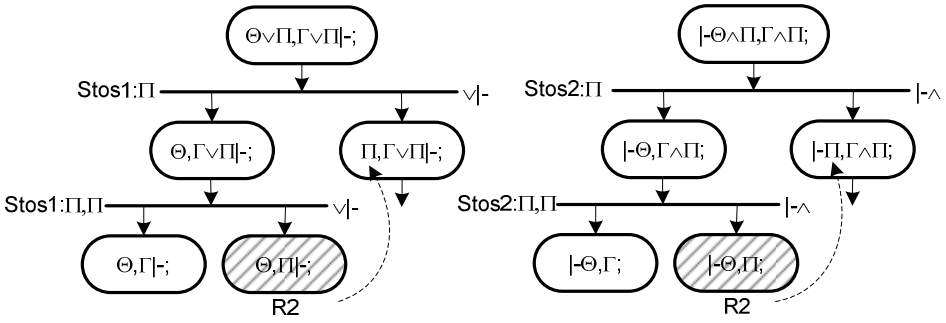

***Rys. 4-6 Reguła R1 w rachunku sekwentów Gentzena***

Można zauważyć, że w wyniku eliminacji ostatniego spójnika, w obu przypadkach powstanie sekwent:

$$\Theta,\ \Phi,\ \Psi,\ \Gamma \vdash \Pi,\ \Psi; \tag{4-3}$$

który odpowiada tautologii.

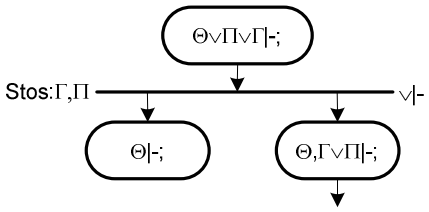
**Reguła R2** – Analogicznie do metody Thelena zapamiętywana jest na stosie druga formuła dysjunkcji lub koniunkcji, w zależności od usytuowania względem znaku wynikania logicznego. Jeśli generowana jest gałąź dla formuły zapisanej na stosie, to sekwent zostaje odrzucony z dalszego procesu normalizacji (Rys. 4-7).




***Rys. 4-7 Reguła R2 w rachunku sekwentów Gentzena***

Dla złożonych sekwentów, w których występuje jednocześnie strona lewa i prawa, należy budować dwa stosy podręczne, dla każdej strony sekwentu oddzielnie.

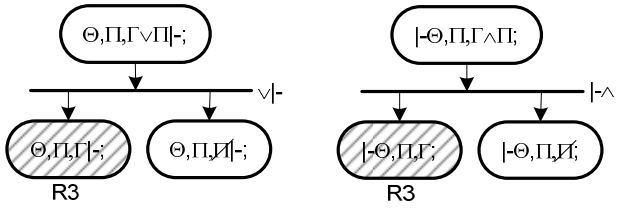
W przypadku reguły R2 problem występuje, gdy drugim członem eliminowanego spójnika jest formuła złożona. Różnice w strukturach drzew dla obu metod można rozwiązać odpowiednio konstruując stos podręczny (Rys. 4-8), poprzez pozyskanie elementów także ze złożonych formuł.




***Rys. 4-8 Konstrukcja stosu podręcznego***

W przypadku, gdy eliminowany spójnik jest inny niż koniunkcja lub dysjunkcja, informacja o aktualnym stosie przekazywana jest dalej. Reguła R2 tak jak w przypadku algorytmu Thelena sprawia, że pierwsze rozwiązania będą rozwiązaniami minimalnymi, dominującymi nad ewentualnie pojawiającymi się później rozwiązaniami nadmiarowymi.

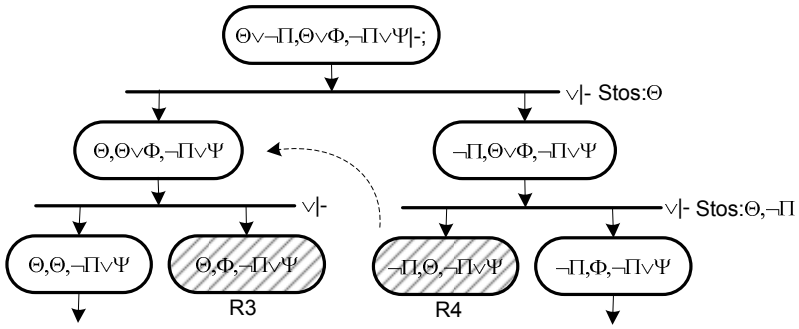
**Reguła R3** – Jeżeli w jednym z sekwentów, powstałych w wyniku eliminacji spójnika dysjunkcji po lewej stronie znaku wynikania logicznego, dojdzie do sklejenia formuł, to należy go pozostawić i poddać normalizacji (Rys. 4-9). W dalszym procesie normalizacji drugi sekwent należy pominąć. Analogicznie należy postąpić w przypadku eliminacji spójnika koniunkcji po prawej stronie sekwentu.




***Rys. 4-9 Reguła R3 w rachunku sekwentów Gentzena***

Aby w pełni zrealizować regułę R3 w logice sekwentów należy zbadać, czy z formułami w dyzjunkcji po lewej, czy też z formułami w koniunkcji po prawej nie ma jeszcze innych formuł, w których może dojść do sklejenia. Podejście takie pozwala na rozwiązanie problemu polegającego na różnicy w konstrukcji drzew w obu metodach.

**Reguła R4** – podobnie jak R2 musi generować stosy dla lewej i prawej strony sekwentu. Na stosie należy przechowywać formuły, które na wyższym poziomie drzewa były już normalizowane (Rys. 4-10). Dodatkowo, tak jak w przypadku algorytmu Thelena-Mathony'ego, konieczne jest przechowywanie informacji o ewentualnym wystąpieniu reguły R2.



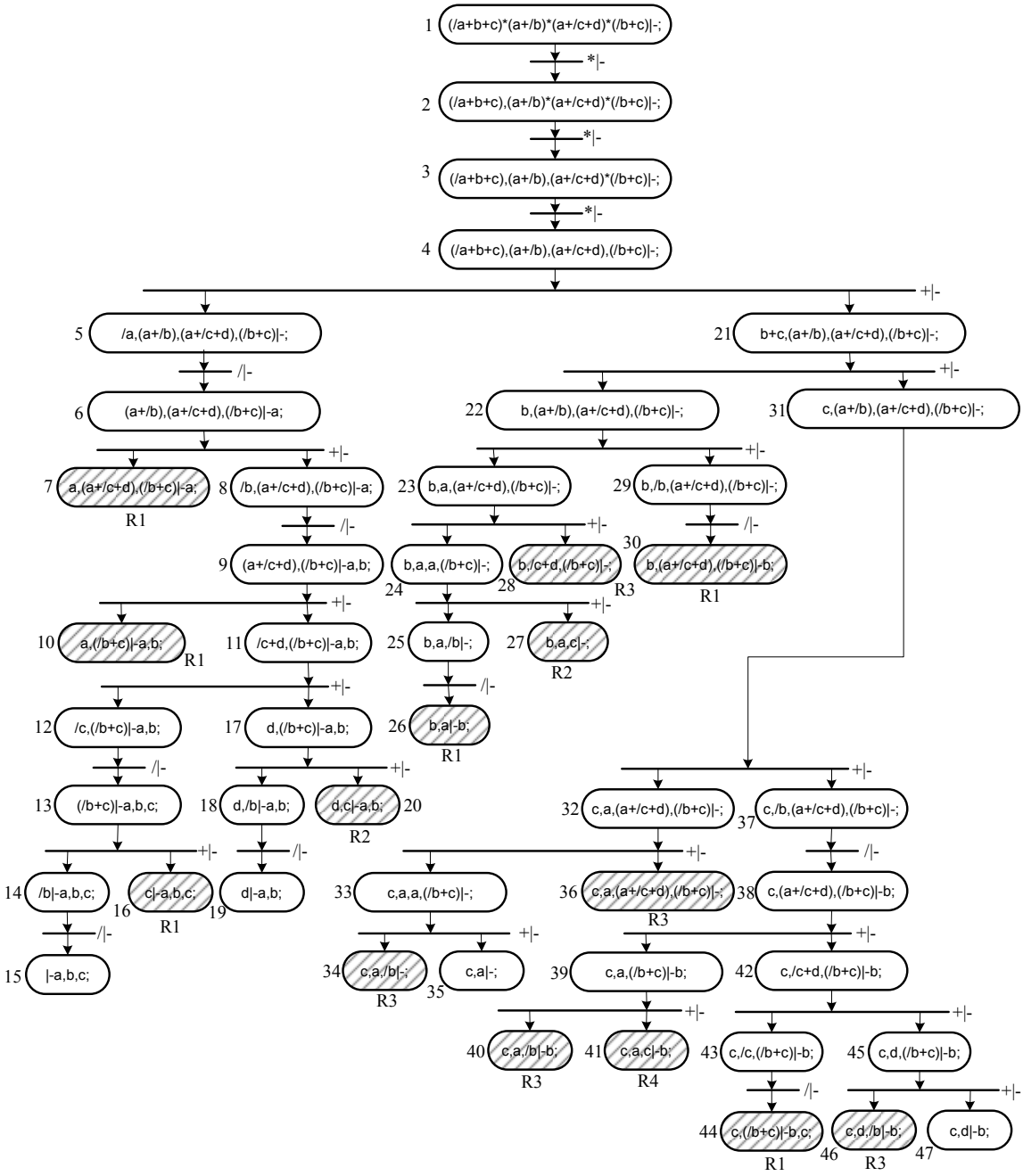

***Rys. 4-10 Reguła R4 w rachunku sekwentów Gentzena***

W wyniku połączenia metody wnioskowania symbolicznego Gentzena z metodą drzew Thelena-Mathony'ego, powstał złożony algorytm, który potrafi wygenerować implikanty proste przy przejściu z postaci koniunkcyjnej do postaci dysjunkcyjnej, a także implicenty proste przy transformacji z postaci dysjunkcyjnej do postaci koniunkcyjnej. Postać koniunkcyjną należy stawiać po lewej stronie sekwentu, a postać dysjunkcyjną po prawej stronie sekwentu.

Przy bardzo zawiłych formułach, przy braku ich oczywistej postaci koniunkcyjnej lub dysjunkcyjnej, proponowana metoda będzie nadal próbowała skrócić drzewo dowodu, choćby z wykorzystaniem tylko części reguł Thelena-Mathony'ego. Proponowane zintegrowane rozwiązanie podatne jest również na heurystyki związane bezpośrednio z algorytmem Thelena [`Kara07`], na przykład porządkowanie alfabetyczne formuł.

Reguła R4 nie została wykorzystana, gdyż jej funkcje całkowicie przejęły reguły R1-R3, wraz z dopuszczalną regułą dominacji. Ewentualne jej wprowadzenie w przypadku sekwentów z sześcioma spójnikami logicznymi (zamiast trzema, jak w przypadku metody Thelena-Mathony'ego) spowalnia proces analizy i nie jest opłacalne z punktu widzenia efektywności.

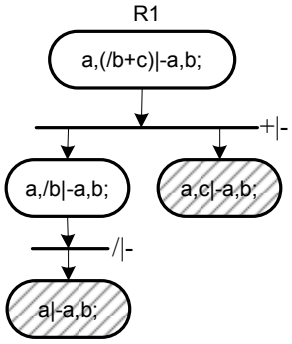
***Przykład 4-1 Łączne wykorzystanie wnioskowania Gentzena i algorytmu Thelena-Mathony'ego***




***Rys. 4-11 Połączenie wnioskowania z elementami metody Thelena-Mathony'ego***

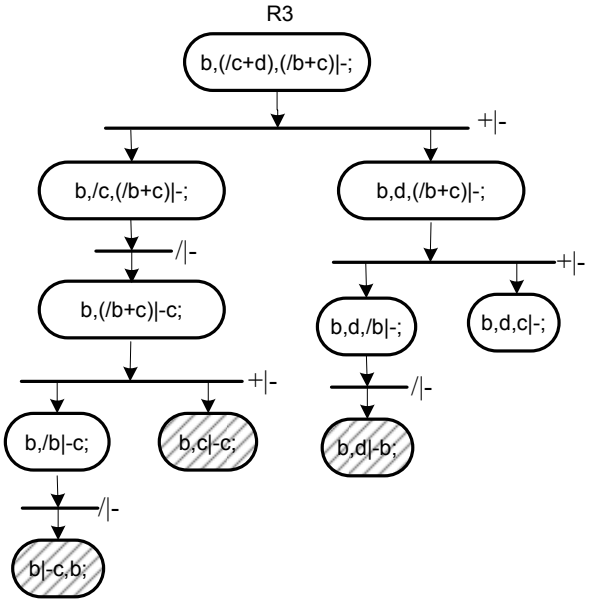
Na Rys. 4-11 przedstawiono łączne wykorzystanie systemu wnioskującego Gentzena z wprowadzonymi elementami metody Thelena-Mathony'ego. Oznakowania *R1-R4* informują o zastosowaniu reguły dopuszczalnej. Numery od *1* do *47* informują o kolejności rozpatrywania poszczególnych sekwentów. Sekwenty zakreślone, są sekwentami odrzuconymi z dalszego procesu normalizacji. W rozpatrywanym przykładzie zintegrowanie metod doprowadziło do skrócenia drzewa dowodu o około 40%.

Czternaście gałęzi zostało uciętych. Przykładowe rozwinięcia dwóch odciętych przedstawiono na Rys. 4-12 i Rys. 4-13. Rozwinięcie gałęzi 10, która została odcięta w wyniku zastosowania reguły **R1** (Rys. 4-12) doprowadziło do uzyskania liści, które również są tautologiami.




***Rys. 4-12 Rozwinięcie gałęzi 10***

Rozwinięcie gałęzi 28 (Rys. 4-13), która została odrzucona w następstwie zastosowania reguły **R3**, doprowadziło do wygenerowania czterech rozwiązań. Trzy z nich są tautologiami, a czwarte wprowadza nadmiarową relację do zbioru rozwiązań opisujących rozpatrywane zjawisko dyskretne.




***Rys. 4-13 Rozwinięcie gałęzi 28***

## 4.5. Wykorzystanie metody rezolucji

Rezolucja jest procedurą automatycznego dowodzenia twierdzeń opartą na generowaniu nowych klauzul, używaną przede wszystkim do wykazywania, że formuła w postaci klauzulowej jest niespełnialna. Jeśli w wyniku zastosowania metody uda się uzyskać niespełnialną klauzulę pustą, to będzie oznaczać, że pierwotny zbiór klauzul jest również niespełnialny [Gbur78][Gall86][ReCl90][Bena05][Indr05][Huza07].

### 4.5.1. Zarys metody rezolucji

Niech $C_1$, $C_2$ będą klauzulami takimi, że $l \in C_1$, $\bar{l} \in C_2$. Klauzule $C_1$, $C_2$ nazywamy klauzulami kolidującymi i mówimy, że kolidują względem komplementarnych literałów $l$, $\bar{l}$. Rezolwentą klauzul $C_1$ i $C_2$ nazywamy klauzulę $C$ postaci (4-4):

$$Rez(C_1,C_2) = (C_1-\{l\}) \cup (C_2 - \{\bar{l}\}). \qquad (4\text{-}4)$$

Klauzule $C_1$,$C_2$ nazywamy klauzulami macierzystymi dla C.

Przykładowe klauzule $C_1 = ab\neg c$ *i* $C_2 = bc\neg e$ kolidują względem literałów $c, \neg c$. Rezolwentą tych klauzul jest klauzula $C$ następującej postaci:

$$C = (ab\neg c - \{\neg c\}) \cup (bc\neg e - \{c\}) = ab \cup b\neg e = ab\neg e.$$

### 4.5.2. Rezolucja w logice sekwentów

Dla danych dwóch sekwentów można utworzyć trzeci sekwent, analogicznie do tworzenia rezolwenty dla klauzul, jeżeli sekwenty te opisują tę samą funkcję oraz posiadają w poprzednikach lub następnikach elementy komplementarne (pewien symbol występuje w jednym z nich bez negacji, a w drugim z negacją). Poprzednik i następnik nowego sekwentu powstaje przez połączenie poprzedników i następników rozpatrywanych sekwentów, z pominięciem elementów komplementarnych (4-5).

$$\frac{\Theta, \boldsymbol{\Phi}, \Gamma \vdash \Psi \quad \Theta, \neg\boldsymbol{\Phi}, \Gamma \vdash \Psi}{\Theta, \Gamma \vdash \Psi} \qquad (4\text{-}5)$$

Eliminując negację w jednym z sekwentów, poprzez zastosowanie reguły Gentzena, definicja przyjmie następującą postać opisaną wzorem (4-6).

$$\frac{\Theta, \boldsymbol{\Phi}, \Gamma \vdash \Psi \quad \Theta, \Gamma \vdash \Psi, \boldsymbol{\Phi}}{\Theta, \Gamma \vdash \Psi} \qquad (4\text{-}6)$$

dla której generowanie nowego sekwentu jest analogiczne do dopuszczalnej reguły cięcia przedstawionej w opisie systemu wnioskowania.

Nowo powstały sekwent nazywany jest sekwentem-consensusem, a sekwenty z których powstał sekwentami macierzystymi. Dzięki zastosowaniu niniejszej metody można badać spełnialność zbioru sekwentów lub dokonywać uproszczenia zbioru sekwentów. Aby dokonać minimalizacji zbioru sekwentów należy dodatkowo badać tautologie oraz możliwe pochłonięcia związane z nowo powstałym sekwentem.

Biorąc pod uwagę minimalizację zbioru sekwentów można wyróżnić następujące przypadki zastosowania elementów metody rezolucji:

- jeżeli consensus dwóch sekwentów jest tautologią, to można go pominąć w procesie dalszej analizy;
- jeżeli consensus dwóch sekwentów dominuje nad sekwentami macierzystymi, to sekwenty macierzyste można zastąpić sekwentem consensusu;
- jeżeli consensus dominuje nad innym sekwentem ze zbioru rozwiązań, to sekwent zdominowany można pominąć;
- jeżeli consensus nie dominuje nad żadnym sekwentem z bazy rozwiązań i nie jest tautologią, to można nie dodawać go do zbioru rozwiązań, gdyż stanowi rozwiązanie nadmiarowe.

Minimalizacji zbioru rozwiązań z wykorzystaniem metody rezolucji dokonuje się poprzez poszukiwanie sekwentów consensusu dominujących nad sekwentami z bazy rozwiązań. Procedurę poszukiwania sekwentów consensusu wykonuje się również dla nowo wprowadzonych do bazy sekwentów powstałych w wyniku zastosowania metody rezolucji.

Na przykład, w wyniku normalizacji sekwentu:

$$(\neg a \wedge \neg b \wedge c) \vee (\neg a \wedge b \wedge c) \vdash ;$$

według reguł Gentzena otrzymuje się dwa sekwenty w bazie rozwiązań:

$$c \vdash a,b; \quad oraz \quad b,c \vdash a;$$

W wyniku zastosowania reguły rezolucji otrzymuje się sekwent będący consensusem:

$$c \vdash a;$$

który zdominuje sekwenty macierzyste i sam pozostanie w bazie rozwiązań.

# 5. Wykorzystanie komputerowego wnioskowania metodą Gentzena w projektowaniu układów cyfrowych

Prace dotyczące wykorzystania logiki Gentzena w projektowaniu układów cyfrowych zostały zapoczątkowane przez prof. Mariana Adamskiego [Adam90]. Jednym ze sposobów projektowania układów cyfrowych jest zastosowanie logiki matematycznej i zastąpienie przekształceń tablicowych przekształceniami symbolicznymi oraz formalnym wnioskowaniem. Projektowanie układów cyfrowych w sposób naturalny wiąże się z metodami logiki matematycznej [Adam90][Szaj96].

Wykorzystanie logiki formalnej jako języka projektowania układów cyfrowych, a w szczególności jako narzędzia syntezy jest ciągle na poziomie badawczym.

## 5.1. Synteza kombinacyjnych układów cyfrowych

Odzwierciedlając spojrzenie na projektowany układ cyfrowy, można sporządzać specyfikację w logice sekwentów na kilka różnych sposobów. Zadaną funkcję logiczną $y = f(x_1,x_2,...,x_n)$ zastępuje się sekwentem $|-y<->f(x_1,x_2,...,x_n)$. Odpowiedniość pomiędzy funkcją boolowską a sekwentem została szczegółowo omówiona w pracy [ErPa84]. Postać tablicową funkcji zapisuje się w postaci podzbiorów sekwentów odpowiadających implikantom afirmacji tych funkcji lub ich negacji. Formułę implikanta zapisuje się w poprzedniku sekwentu, natomiast symbole zmiennych zależnych funkcji jako koniunkcje w następniku.

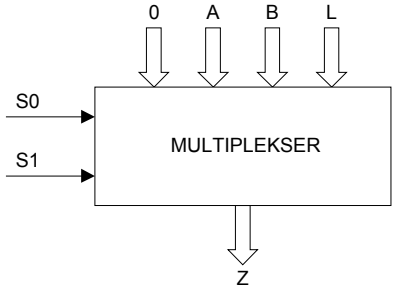
Główną zaletą specyfikowania układu za pomocą sekwentów jest możliwość stopniowego sporządzania opisów fragmentarycznych, a następnie ich rozszerzanie poprzez wprowadzanie dodatkowych aksjomatów dotyczących jego funkcjonowania.

***Przykład 5-1 Synteza kombinacyjnego układu cyfrowego***

Praktyczne wykorzystanie przekształceń Gentzena w syntezie układów cyfrowych omówione zostanie na przykładzie projektu multipleksera (Rys. 5-1) Przykład został zaczerpnięty z książki [Adam90]. Deklaracje wejść i wyjść układu przedstawiono w Tab. 5-1.

***Tab. 5-1. Deklaracje zmiennych***

| Wejścia | Wyjścia |
|---|---|
| A=[a3, a2, a1, a0]<br>B=[b3, b2, b1, b0]<br>L=[l3, l2, l1, l0] | Z=[z3, z2, z1, z0] |




***Rys. 5-1 Schemat blokowy multipleksera***

Zazwyczaj mikrooperacje bardziej złożonych bloków funkcjonalnych opisuje się zbiorami sekwentów, gdzie każdy z nich określa tylko pewien fragment działania układu. Mikrooperacje w postaci ogólnej, zostały przedstawione w Tab. 5-2.

***Tab. 5-2 Postać ogólna mikrooperacji***

```
/s0*/s1 |- [Z=0]  {Wyjście Z jest równe 0}
/s0*s1  |- [Z=A]  {Wyjście Z jest równe A}
s0*/s1  |- [Z=B]  {Wyjście Z jest równe B}
s0*s1   |- [Z=L]  {Wyjście Z jest równe L}
```

Specyfikacja multipleksera przy założeniu o otwartym świecie OWA [Adam90] przyjmie postać przedstawioną w Tab. 5-3. Definiuje się wówczas zarówno aktywne ($Z_i=1$), jak i pasywne ($Z_i=0$) stany wyjść.

***Tab. 5-3 Specyfikacja multipleksera***

```
/s0*/s1|- /z0*/z1*/z2*/z3;
/s0*s1|-(z0<->a0)*(z1<->a1)*(z2<->a2)*(z3<->a3);
s0*/s1|-(z0<->b0)*(z1<->b1)*(z2<->b2)*(z3<->b3);
s0*s1|-(z0<->l0)*(z1<->l1)*(z2<->l2)*(z3<->l3);
```

Po wykonaniu normalizacji sekwentów i określeniu typu każdej zmiennej (czy jest ona typu wejściowego, czy wyjściowego) i wprowadzając spójniki logiczne według reguł Gentzena (rozdz. 3.5), otrzymuje się wyrażenia opisujące wyjścia multipleksera, zarówno dla stanu *0* jak i *1*, w postaci sekwentowej uporządkowanej sumy iloczynów (Tab. 5-4):

***Tab. 5-4 Wyrażenia znormalizowane w postaci sumy iloczynów***

```
/s0*/s1 + s0*s1*/l0 + /s0*/a0 + /s1*/b0|-/z0;
/s0*/s1 + s0*s1*/l1 + /s0*/a1 + /s1*/b1|-/z1;
/s0*/s1 + s0*s1*/l2 + /s0*/a2 + /s1*/b2|-/z2;
/s0*/s1 + s0*s1*/l3 + /s0*/a3 + /s1*/b3|-/z3;
s1*a0*/s0 + s0*b0*/s1 + s0*s1*l0|-z0;
s1*a1*/s0 + s0*b1*/s1 + s0*s1*l1|-z1;
s1*a2*/s0 + s0*b2*/s1 + s0*s1*l2|-z2;
s1*a3*/s0 + s0*b3*/s1 + s0*s1*l3|-z3;
```

Na podstawie Tab. 5-4 określa się bezpośrednio zbiory $F^1$ jak i $F^0$. Dodatkowo, pośrednio można wyznaczyć zbiór $F^-$. Wygenerowane drzewo dowodu złożone jest 77 węzłów i zbudowane zostało w czasie 0,2s.

Stosując założenie o zamkniętym świecie (CWA) [Adam90], poprzez zmianę równoważności na implikację, możliwe jest uproszczenie specyfikacji. W tym przypadku określa się jedynie aktywne wyjścia, przyjmując, że niewymienione wyjścia są w stanie pasywnym. W wyniku normalizacji w ten sposób przygotowanej specyfikacji uzyska się tylko jeden zbiór rozwiązań (dla $F^1$ lub $F^0$ w zależności, czy zmienne funkcyjne implikują wyjścia, czy wyjścia implikują zmienne funkcyjne). Skróceniu także ulegnie drzewo dowodu.

Projektując układy kombinacyjne z wykorzystaniem rachunku sekwentów Gentzena możliwe jest również wprowadzanie do specyfikacji wyrażeń z obojętnymi, nieznanymi lub neutralnymi wartościami sygnałów wyjściowych. Nie będą miały one wpływu na sposób funkcjonowania syntezowanego układu, ale mogą mieć wpływ na złożoność znormalizowanego opisu układu. Mają one także wpływ na czas przetwarzania specyfikacji, gdyż będą brały udział w procesie minimalizacji.

Problem komputerowej dekompozycji i syntezy logicznej zaimplementowanego układu został rozwiązany poprzez transformację jego specyfikacji do postaci akceptowalnej przez system uniwersytecki „ESPRESSO” [ŁuJa97][Łuba01]. System ESPRESSO jest rozbudowanym, heurystycznym programem minimalizacji funkcji boolowskich, opracowanym w początkach lat osiemdziesiątych na Uniwersytecie Kalifornijskim w Berkeley. Swoją ogromną popularność zawdzięcza wolnej dystrybucji oraz dużej skuteczności procesów minimalizacji funkcji boolowskich. System ten, pierwotnie planowany do optymalizacji struktur PLA w uniwersyteckich systemach projektowania układów Full Custom, był powszechnie stosowany w komercyjnych kompilatorach układów PLD np. ABEL, LOGIC, CUPL [ŁuJa97]. Obecnie

wykorzystywany w uniwersyteckich systemach syntezy logicznej DEMAIN, SIS, MIS [Łuba01].

### 5.1.1. Normalizacja i minimalizacja funkcji logicznych

System wnioskowania Gentzena w pierwotnej formie jest mało przydatny do minimalizacji funkcji logicznych. Wprowadzenie reguł dopuszczalnych oraz proponowane w pracy połączenie wnioskowania symbolicznego z metodą Thelena-Mathony'ego (rozdz. 4.4.2) i metodą rezolucji (rozdz. 4.5.2) sprawia, że opracowany autorski system jest użyteczny do symbolicznej minimalizacji funkcji logicznych, zwłaszcza słabo określonych. Wzajemnie się uzupełniające metody rezolucji, dominacji i cięcia powodują, że sekwenty będące wynikiem normalizacji funkcji logicznej, opisanej w postaci sekwentowej, będą odpowiadały zbiorom implikantów lub implicentów prostych. Otrzymywany wynik uwarunkowany jest formą opisu funkcji logicznej w postaci sekwentowej.

Problem minimalnego pokrycia funkcji przez implikanty lub implicenty proste może zostać rozwiązany poprzez zaadoptowanie metody Petricka [Trac86] i ponowne wykorzystanie systemu wnioskującego.

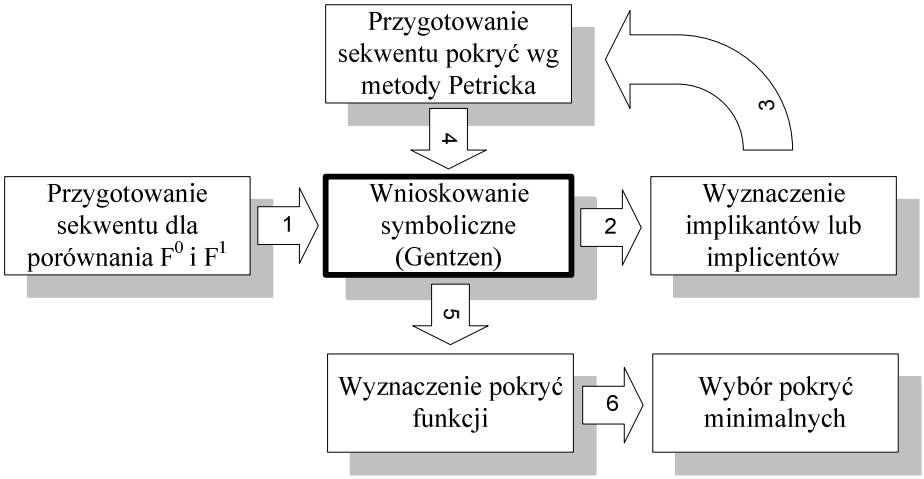
Stosując wnioskowanie symboliczne do minimalizacji funkcji logicznych należy pamiętać, że system gentzenowski poszukuje rozwiązań dokładnych, stąd w większości przypadków znane literatury specjalizowane metody heurystyczne i algebraiczne będą dawały szybsze rezultaty. Ważną zaletą systemu wnioskującego w zastosowaniu do minimalizacji funkcji logicznej jest możliwość tylko częściowego specyfikowania funkcji, jak również wprowadzanie wartości nieokreślonych, które będą brały udział w procesie minimalizacji.

#### *Minimalizacja funkcji silnie nieokreślonych*

Wielokrotne, kolejne zastosowanie algorytmu wnioskowania symbolicznego pozwala na automatyzację procesu dokładnej minimalizacji funkcji silnie nieokreślonych (Rys. 5-2).

Należy tu zwrócić uwagę, że proponowana komputerowa metoda minimalizacji jest ilustracją przydatności automatycznego wnioskowania do weryfikacji poprawności metod wykonywanych ręcznie lub komputerowo, metodami heurystycznymi.

***Przykład 5-2 Wnioskowanie symboliczne Gentzena w procesie minimalizacji funkcji silnie nieokreślonych.***




***Rys. 5-2 Schemat blokowy minimalizacji funkcji***

W pierwszym kroku, za pomocą wnioskowania Gentzena, na drodze symbolicznej następuje porównanie zbiorów $F^0$ (zbiór zer) i $F^1$ (zbiór jedynek) badanej funkcji (5-1) (Tab. 5-5), zakończone wyznaczeniem implikantów lub implicentów prostych (Tab. 5-6).

$$F^0 = \begin{Bmatrix} 00001 \\ 01010 \\ 10101 \\ 11100 \end{Bmatrix} \quad F^1 = \begin{Bmatrix} 00011 \\ 01100 \\ 01101 \\ 10000 \\ 11001 \\ 11101 \end{Bmatrix} \qquad \textbf{(5-1)}$$

W celu uzyskania postaci dysjunkcyjnej odpowiednie formuły dla zbiorów $F^0$ i $F^1$ umieszcza się po prawej stronie znaku wynikania logicznego w sekwencie, oddzielając je przecinkiem (Tab. 5-5). Umieszczając znak wynikania logicznego po formułach określa się postać koniunkcyjną.

***Tab. 5-5 Sekwent porównujący zbiory $F^0$ i $F^1$***

| **Sekwent zależności zbiorów** |
|---|
| `|-(/x1*/x2*/x3*x4*x5*y), (/x1*x2*x3*/x4*/x5*y),` |
| `(/x1*x2*x3*/x4*x5*y), (x1*/x2*/x3*/x4*/x5*y),` |
| `(x1*x2*/x3*/x4*x5*y), (x1*x2*x3*/x4*x5*y),` |
| `(/x1*/x2*/x3*/x4*x5*/y), (/x1*x2*/x3*x4*/x5*/y),` |
| `(x1*/x2*x3*/x4*x5*/y), (x1*x2*x3*/x4*/x5*/y);` |

W wyniku normalizacji otrzymane zostaną sekwenty informujące, dla jakich wartości funkcja przyjmuje wartość jeden, wartość zero, a także kiedy jest nieokreślona (Tab. 5-6)

***Tab. 5-6 Implikanty funkcji***

| $F^0$ | $F^1$ | $F^-$ |
|---|---|---|
| (x2*x1*/x5),(x3*x1*/x5),<br>(x5*x1*/x2),(x2*/x3*/x1),<br>(x2*/x3*/x5),(x4*x2),<br>(x3*/x2),(/x4*/x2*/x1),<br>(x5*/x4*/x2),<br>(/x4*/x3*/x1),<br>(/x5*/x3*/x1), (x4*/x5) | (x4*/x2), (x5*x4),<br>(x3*/x1),<br>(x2*/x1*/x4),<br>(/x4*/x1*/x5),<br>(x5*x2), (/x5*/x2),<br>(x1*/x3),<br>(/x4*/x3*/x5),<br>(x2*/x3*/x4) | (x2*x1*/x5*/x3),<br>(x4*x1),<br>(x5*x1*/x2*/x3),<br>(x2*/x3*/x1*/x4),<br>(x5*x2*/x3*/x1),<br>(x2*/x3*/x5*/x4),<br>(x5*x4*x2),<br>(x3*/x2*/x1),<br>(x3*/x2*/x5),<br>(x4*x3),<br>(/x4*/x3*/x1*/x5),<br>(/x5*/x2*/x1),<br>(x4*/x5*/x2) |

Drzewo dowodu dla rozpatrywanego przypadku posiada 5722 węzły i zostało wygenerowane za pomocą autorskiego systemu wnioskującego w czasie 3,7s.

Następnym etapem jest znalezienie minimalnego pokrycia funkcji. Oznaczając poszczególne implikanty na przykład dla $F^1$, symbolami od *A1,..,A10*, a poszczególne termy funkcji symbolami *a,..,f* możliwe jest zbudowanie tabeli Quine'a dla pokryć funkcji (Tab. 5-7). Symbol „*x*" oznacza, które termy funkcji są pokrywane przez dany implikanty.

***Tab. 5-7 Tabela Quine'a pokrycia funkcji***

| | **a** | **b** | **c** | **d** | **e** | **f** |
|---|---|---|---|---|---|---|
| **A1** | x | | | | | |
| **A2** | x | | | | | |
| **A3** | | x | x | | | |
| **A4** | | x | x | | | |
| **A5** | | x | | | | |
| **A6** | | | x | | x | x |
| **A7** | | | | x | | |
| **A8** | | | | x | x | |
| **A9** | | | | x | | |
| **A10** | | | | | x | |

Zgodnie z metodą Petricka [Łuba01][StCy07] sekwent definiujący wszystkie pokrycia funkcji przyjmie postać przedstawioną w tabeli (Tab. 5-8).

***Tab. 5-8 Sekwent pokrycia funkcji***

| **Sekwent pokrycia funkcji** |
|---|
| (A1+A2), (A3+A4+A5), (A3+A4+A6),<br>(A7+A8+A9), (A6+A8+A10), (A6)\|-; |

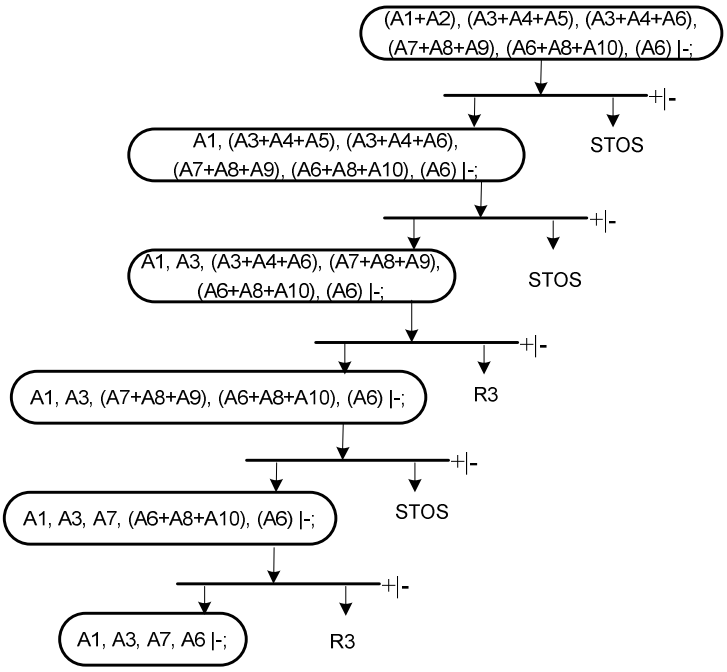
W wyniku normalizacji sekwentu (Tab. 5-8) wyznaczone zostaną wszystkie pokrycia funkcji (Tab. 5-9). W celu wyznaczenia minimalnego pokrycia zwykle wybiera się te, które zawierają najmniejszą liczbę implikantów prostych. W analizowanym przykładzie wszystkie pokrycia składają się z czterech implikantów, więc należy wybrać te o najmniejszej złożoności.

***Tab. 5-9 Minimalne pokrycia funkcji***

| **Pokrycia funkcji** |
|---|
| A1,A3,A7,A6\|-; A1,A3,A8,A6\|-; A1,A3,A9,A6\|-; |
| A1,A4,A7,A6\|-; A1,A4,A8,A6\|-; A1,A4,A9,A6\|-; |
| A1,A5,A6,A7\|-; A1,A5,A6,A8\|-; A1,A5,A6,A9\|-; |
| A2,A3,A7,A6\|-; A2,A3,A8,A6\|-; A2,A3,A9,A6\|-; |
| A2,A4,A7,A6\|-; A2,A4,A8,A6\|-; A2,A4,A9,A6\|-; |
| A2,A5,A6,A7\|-; A2,A5,A6,A8\|-; A2,A5,A6,A9\|-; |

Drzewo dowodu dla wyznaczenia minimalnych pokryć funkcji (Tab. 5-8) posiada 182 węzły, które zostały wygenerowane w czasie 0,1s.

Należy tutaj zwrócić uwagę, że w odróżnieniu od metod algebraicznych pierwsze pokrycie uzyskuje się już w pierwszych kilku krokach wnioskowania (w analizowanym przykładzie już po sześciu krokach), bez konieczności przetwarzania do końca całego wyrażenia (Rys. 5-3).




***Rys. 5-3 Wyznaczanie pierwszego rozwiązania***

Uzyskane pierwsze pokrycie niekoniecznie będzie minimalne, ale w wielu przypadkach wystarczające. Przykładowe minimalne pokrycie funkcji implikantami, dla zaznaczonego w (Tab. 5-9) sekwentu, będzie wyglądać następująco (5-2):

$$Y = (x5 * x4) + (x3 + / x1) + (x1 * / x3) + (x5 * x2) \qquad \textbf{(5-2)}$$

W samym procesie normalizacji wyrażeń opisujących badaną funkcję (sekwentów) pojawiają się też wyrażenia, reprezentujące wartości nieokreślone funkcji $F^-$ (kreski), które również mogą być ewentualnie wykorzystane w procesie minimalizacji lub dookreślone przez projektanta, jeśli są skutkiem nieuwzględnienia w specyfikacji układu kombinacyjnego pewnych istotnych stanów wejść.

***Przykład 5-3 Synteza i analiza cyfrowego układu sterującego z wykorzystaniem specyfikacji behawioralnej.***

W celu zademonstrowania przydatności proponowanej metody do celów badawczych, projektowych lub edukacyjnych zaadaptowano przykład zaczerpnięty z nowego wydania [SiMa03] klasycznego podręcznika, opracowanego przez zespół prof. J. Siwińskiego. Binarny, kombinacyjny układ sterujący opisany jest w następujący sposób (Zad.2.17): Zaprojektować układ przełączający, umożliwiający sterowanie pracą pomp *P1,P2,P3,P4* w zależności od sygnałów wejściowych *x1,x2,x3,r1,r2,r3,r4*. Stan pomp w zależności od tych sygnałów jest następujący:

- dla x1=1 - powinna pracować 1 pompa,
- dla x2=1 - powinny pracować 2 pompy,
- dla x3=1 - powinny pracować 3 pompy,
- dla x1=x2=x3=0 - żadna z pomp nie pracuje,
- dla r1=1 - odstawiona do remontu pompa P1,
- dla r2=1 - odstawiona do remontu pompa P2,
- dla r3=1 - odstawiona do remontu pompa P3,
- dla r4=1 - odstawiona do remontu pompa P4,
- dla r1=r2=r3= r4=0 - żadna z pomp nie jest odstawiona do remontu.

Należy przyjąć, że do remontu może być odstawiona co najwyżej jedna pompa. Silniki pomp *P1, P2, P3, P4* załączone są do sieci stycznikami *Z1, Z2, Z3, Z4*. Przy pojawieniu się sygnałów *x1, x2, x3* pompy powinny się włączać wg narastających indeksów (z pominięciem pompy odstawionej do remontu).

W podręczniku zastosowano opis w postaci klasycznej tablicy decyzyjnej (tablicy zespołu czterech funkcji logicznych), zamieszczając 20 zależności między wektorami wejściowymi i wyjściowymi oraz pomijając 108 wierszy, dla których sygnały wyjściowe są nieokreślone (Tab. 5-10).

***Tab. 5-10 Tabela decyzyjna rozpatrywanego układu***

| | x1 | x2 | x3 | r1 | r2 | r3 | r4 | z1 | z2 | z3 | z4 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 64 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 72 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 68 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 66 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 65 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 32 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 40 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 36 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 34 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| 33 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 16 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 24 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 20 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 18 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 17 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | - | - | - | - |
| pozostałe | | | | | | | | ⋮ | | | |
| 127 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | - | - | - | - |

W rozpatrywanym przypadku, po zastosowaniu klasycznej metody opisanej poprzednio, duża liczba wygenerowanych implikantów i ich selekcja (wyrażenie logiczne ma aż 11 zmiennych wejściowych i nie zawsze potrzebne jest wyznaczanie zbioru $F^-$) powoduje nadmierne wydłużenie się czasu obliczeń.

Jak pokazano w pracach [Adam90][TkAd08] w wielu przypadkach wygodniej jest przedstawić poszczególne zależności w układzie kombinacyjnym w zwarty sposób, odzwierciedlający opis słowny (Tab. 5-11), a następnie przekształcić je systemem wnioskującym do postaci zawsze znacznie uproszczonych sekwentów-klauzul (złożonych ze znacznie mniejszej liczby formuł atomowych).

***Tab. 5-11 Specyfikacja behawioralna (założenie o otwartości świata)***

| **Specyfikacja behawioralna (OWA)** |
|---|
| /x1,/x2,/x3\|-/z1*/z2*/z3*/z4; |
| x1\|-(/r1<->z1)*(r1<->z2)*/z3*/z4; |
| x2\|-(/r1<->z1)*(/r2<->z2)*((r1+r2)<->z3)*/z4; |
| x3\|-(/r1<->z1)*(/r2<->z2)*(/r3<->z3)*((r1+r2+r3)<->z4); |

Po zakończeniu procesu wnioskowania (przekształceniu sekwentów do postaci zbioru sekwentów-klauzul, niebędących aksjomatami logicznymi, przypisanych do liści drzewa dowodu) otrzymuje się uproszczony zbiór reguł typu *IF…THEN*, opisujący każdy sterownik pompy oddzielnie. Zastosowanie opisu behawioralnego sprawia, że drzewo dowodu składa się z 87 węzłów i jest wykonane w czasie 0,3s.

***Tab. 5-12 Definicje typu zmiennych***

| **Typy zmiennych** |
|---|
| IN:x1,x2,x3,r1,r2,r3,r4; |
| OUT:z1,z2,z3,z4; |

Wprowadzenie dodatkowego określenia typu zmiennych (wejściowe/wyjściowe) (Tab. 5-12) umożliwia uporządkowanie opisu i pogrupowanie znormalizowanych klauzul w grupy definiujące poszczególne wyjścia układu sterującego (Tab. 5-13).

***Tab. 5-13 Specyfikacja znormalizowana i uporządkowana***

| **Regułowy opis sterownika** |
|---|
| x1*/r1 + x2*/r1 + x3*/r1\|-z1; |
| x1*r1 + x2*/r2 + x3*/r2\|-z2; |
| x2*r1 + x2*r2 + x3*/r3\|-z3; |
| x3*r1 + x3*r2 + x3*r3\|-z4; |
| /x1*/x2*/x3 + r1*x1 + r1*x2 + r1*x3\|-/z1; |
| /x1*/x2*/x3 + x1*/r1 + r2*x2 + r2*x3\|-/z2; |
| /x1*/x2*/x3 + x1 + x2*/r1*/r2 + r3*x3\|-/z3; |
| /x1*/x2*/x3 + x1 + x2 + x3*/r1*/r2*/r3\|-/z4; |

Na podstawie podanej specyfikacji (Tab. 5-11) wyznacza się komputerowo dwie równoznaczne formy opisu układu (Tab. 5-13): dla wyjść aktywnych (*z1, z2, z3, z4*) - odpowiadająca $F^1$ oraz dla wyjść nieaktywnych (*/z1, /z2, /z3, /z4*) - odpowiadająca $F^0$. Taka forma ta jest bardzo dogodna do wprowadzenia do systemu ESPRESSO i umożliwia bezpośrednią współpracę programu wnioskującego z uniwersyteckimi programami do analizy i syntezy układów cyfrowych np. DOMAIN [Łuba01]. Przykład transformacji opisu na format ESPRESSO szczegółowo omówiony został w rozdziale 5.4.

### *Analiza i symulacja logiczna*

Stosując założenie o otwartym świecie (OWA) [Adam90] można wprowadzać do znormalizowanej specyfikacji układu dodatkowe założenia i badać jego zachowanie. Analiza jest bardzo przejrzysta dla człowieka gdyż odzwierciedla naturalny tok jego myślenia. Przykładowo, chcąc zbadać jak zachowa się układ po wciśnięciu przycisku *x1*, należy dodać do znormalizowanej specyfikacji wyrażenie (5-3).

$$|-x1*/x2*/x3; \qquad \textbf{(5-3)}$$

W wyniku ponownej normalizacji zbioru sekwentów systemem wnioskującym i uporządkowaniu wyrażeń ze względu na typy formuł otrzymuje się wyrażenia opisujące zachowanie układu (Tab. 5-14).

***Tab. 5-14 Symulacja logiczna dla wciśniętego przycisku pierwszego***

| **Wyznaczone zachowanie układu** | |
|---|---|
| x1*/r1\|-z1; | x1*r1\|-z2; |
| r1*x1\|-/z1; | x1*/r1\|-/z2; |
| x1\|-/z3; | x1\|-/z4; |
| /x1 + x2 + x3\|-; | |

Przyjęte założenie otwartego świata (OWA) w konsekwencji spowodowało wyznaczenie wszystkich możliwych przypadków, wynikających logicznie z pierwotnej specyfikacji kombinacyjnego układu sterującego, przy wciśniętym przycisku *x1* uruchamiającym tylko jedną pompę. Z analizy sekwentów opisujących zachowanie układu wynika, że jeżeli jest wciśnięty przycisk *x1* i pompa pierwsza *z1* nie jest odstawiona do remontu, to działać będzie pierwsza pompa (*x1*/r1|-z1*). Jeśli pierwsza pompa odstawiona jest do remontu, a przycisk pierwszy jest wciśnięty, to działać będzie pompa druga (*x1*r1|-z2*). Pompa pierwsza nie działa, jeśli przycisk pierwszy jest wciśnięty, a pompa jest remontowana (*r1*x1|-/z1*). Pompa druga nie działa, jeśli przycisk pierwszy jest wciśnięty i pompa druga jest remontowana (*x1*/r1|-/z2*). Jeżeli przycisk pierwszy jest wciśnięty, to pompa trzecia i czwarta nie działają (*x1|-/z3; x1|-/z4*). Ostatni sekwent (*/x1 + x2 + x3|-*) potwierdza przyjęte założenie (5-4).

$$\frac{|-x1*/x2*/x3;}{/x1+x2+x3|-;} \qquad \textbf{(5-4)}$$

Widać tutaj, że symulacja zgadza się ze specyfikacją i sterownik kombinacyjny działa prawidłowo.

Wskazane jest sprawdzenie sytuacji nieprzewidzianych w specyfikacji np. symulowanie układu, gdy dwa przyciski wciśnięte są jednocześnie. Analizując w taki

sposób układ można znaleźć błędy w specyfikacji lub wykryć w niej sytuacje, dla których projektowany układ jest niedospecyfikowany. W bardzo łatwy sposób można rozbudowywać też specyfikację (Tab. 5-11) wprowadzając dodatkowe reguły lub rozbudowując istniejące, które zabezpieczą układ przed nieprzewidzianym zachowaniem (Tab. 5-15).

***Tab. 5-15 Specyfikacja behawioralna z zabezpieczeniem***

| **Specyfikacja behawioralna (OWA)** |
|---|
| `/x1,/x2,/x3\|-/z1*/z2*/z3*/z4;` |
| `x1*/x2*/x3\|-(/r1<->z1)*(r1<->z2)*/z3*/z4;` |
| `x2*/x1*x3\|-(/r1<->z1)*(/r2<->z2)*((r1+r2)<->z3)*/z4;` |
| `x3*/x1*/x2\|-(/r1<->z1)*(/r2<->z2)*  (/r3<->z3)` |
| `*((r1+r2+r3)<->z4);` |

W rozbudowanej specyfikacji (Tab. 5-15) wprowadzono dodatkowe warunki zabezpieczające przed niedospecyfikowanymi skutkami wciśnięcia więcej niż jednego przycisku. W takiej sytuacji układ nie będzie uruchomiony. Całość zabezpieczenia można ponownie zweryfikować, wtórną normalizacją po dodaniu do znormalizowanej już specyfikacji (Tab. 5-15) sekwentu definiującego wciśnięcie więcej niż jednego przycisku (5-5).

$$|-x1*/x2*x3; \qquad \textbf{(5-5)}$$

W wyniku wnioskowania uzyskuje się tylko jeden sekwent informujący o zachowaniu układu w przypadku, gdy przyciski pierwszy i trzeci nie są wciśnięte (5-6).

$$|-/x1*x2*/x3; \qquad \textbf{(5-6)}$$

Brak jakichkolwiek rozwiązań informuje, że układ nie zadziała przy wciśniętym więcej niż jednym przycisku.

### 5.1.2. Weryfikacja układu kombinacyjnego względem częściowej lub pełnej specyfikacji

Kolejnym przykładem zastosowania automatycznego dowodzenia twierdzeń jest weryfikacja (porównanie) funkcji logicznych, realizowanych przez różne, ale zbliżone do siebie układy cyfrowe [TkAd05]. Wykorzystanie logiki formalnej do weryfikacji układów cyfrowych omówiono w pracy [Evek85][Evek87][Krop99].

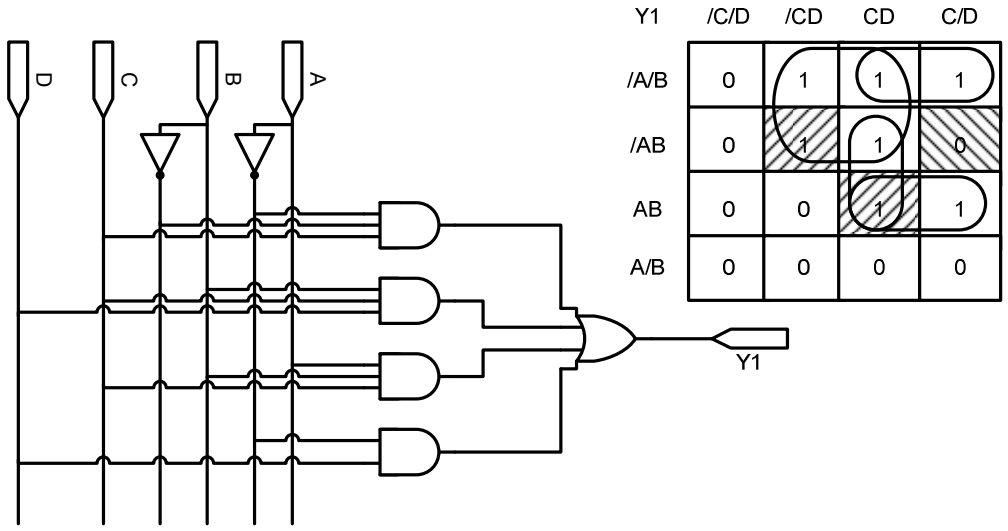
***Przykład 5-4 Weryfikacja układu kombinacyjnego względem częściowej lub pełnej specyfikacji.***

W poniższym przykładzie przedstawione zostaną zalety, jakie wykazuje system wnioskowania symbolicznego w identyfikacji istotnych różnic w układach kombinacyjnych. Dodatkowo przedstawione zostaną możliwości, jakie daje system

wnioskowania symbolicznego dla identyfikacji istotnych różnic w funkcjonowaniu układów kombinacyjnych przedstawionych na rysunkach (Rys. 5-4**,** Rys. 5-5). Dwa różne układy kombinacyjne realizują następujące funkcje logiczne (5-7):

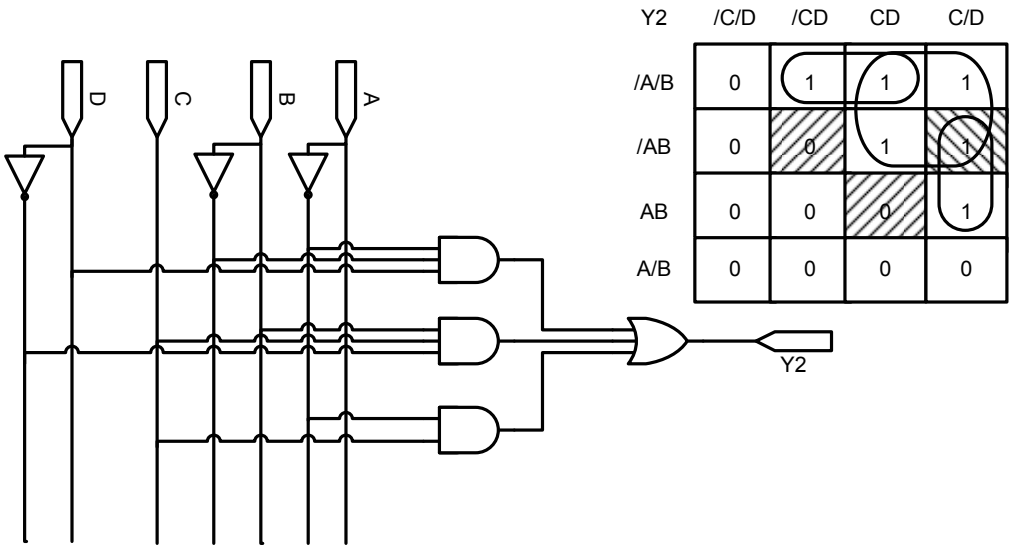
$$Y1=(/A*D)+(/A*/B*C)+(B*C*D)+(A*B*C)$$
$$Y2=(/A*C)+(/A*/B*D)+(B*C*/D) \qquad \textbf{(5-7)}$$

W celu poglądowego zobrazowania wyników porównań, na Rys. 5-4 i Rys. 5-5 zestawiono odpowiednie siatki Karnaugha, sporządzone dla rozpatrywanych funkcji.




***Rys. 5-4 Układ realizujący funkcję Y1***

Funkcja *Y1* nie jest postaci minimalnej, gdyż zawiera grupę nadmiarową *B*C*D*, służącą do eliminacji zjawiska hazardu statycznego. Funkcja *Y2* jest w dysjunkcyjnej postaci minimalnej. W siatkach Karnaugha zaznaczono kratki, odpowiadające kombinacjom sygnałów wejściowych, dla których rozpatrywane funkcje są różne.




***Rys. 5-5 Układ realizujący funkcję Y2***

W celu bardziej przejrzystego przedstawienia zagadnienia funkcje badanych układów przedstawione zostały w formie dysjunkcyjnej, choć system wnioskowania Gentzena nie wymaga wyłącznie tylko takiej postaci funkcji. Sposób przedstawienia opisu behawioralnego badanego układu nie ma w tym przypadku dużego znaczenia. Podczas badania wzajemnej relacji dwóch funkcji rozpatruje się następujące trzy przypadki (5-8):

$$\begin{gathered} 1)\ Y1 \vdash Y2; \\ 2) Y2 \vdash Y1; \\ 3)\ \vdash Y1 <-> Y2;\ (lub\ \vdash Y2 <-> Y1) \end{gathered} \qquad (5\text{-}8)$$

***Normalizacja sekwentu, gdy funkcja Y1 jest w poprzedniku, a Y2 w następniku wyrażenia.***

Rozpatruje się sytuację, w której zakłada się, że funkcja *Y1* implikuje funkcję *Y2* ($\vdash Y1 -> Y2$), co jest równoznaczne zapisowi (5-9):

$$Y1 \vdash Y2; \qquad (5\text{-}9)$$

Po podstawieniu odpowiednich formuł do wyrażenia (5-9) otrzymuje się sekwent, który poddawany jest stopniowej normalizacji:

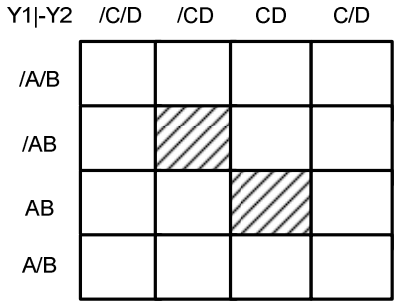
$$(/A*D)+(/A*/B*C)+(B*C*D)+(A*B*C) \vdash (/A*C)+(/A*/B*D)+(B*C*/D);$$

Po normalizacji otrzymano sekwenty:

$$/A,B,/C,D \vdash;$$

$$A,B,C,D \vdash;$$

Wskazują one te wartościowania funkcji, dla których implikacja ta nie zachodzi (Rys. 5-6). Wynikiem normalizacji są wszystkie kombinacje zmiennych wejściowych *ABCD*, dla których funkcja *Y1* przyjmuje wartość „*1*”, a funkcja *Y2* wartość „*0*”. W ten wyznacza się tylko część różnic pomiędzy funkcjami *Y1* i *Y2*. Różnice nie zostaną wykryte, jeżeli zbiór jedynek funkcji *Y1*, jest podzbiorem jedynek funkcji *Y2*.




***Rys. 5-6 Siatka dla wyników normalizacji Y1|-Y2***

Powyższa procedura wykorzystywana jest do wykrywania, które jedynki funkcji *Y1* nie są pokryte jedynkami funkcji *Y2*. Drzewo dowodu dla rozpatrywanej zależności funkcji posiada 60 węzły i generowane jest w czasie 0,03s.

***Normalizacja sekwentu, gdy funkcja Y2 jest w poprzedniku, a funkcja Y1 w następniku.***

W celu wyznaczenia pozostałych różnic konieczna jest analiza sekwentu odwrotnego (5-10).

$$Y2|\text{-}Y1; \qquad \textbf{(5-10)}$$

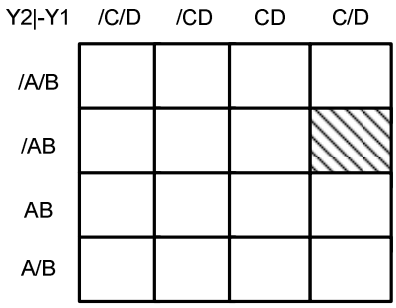
Po podstawieniu odpowiednich formuł dla wyrażenia (5-10) otrzymuje się sekwent, który poddaje się stopniowej normalizacji:

$$(/A*C)+(/A*/B*D)+(B*C*/D)|\text{-}(/A*D)+(/A*/B*C)+(B*C*D)+(A*B*C);$$

Po normalizacji otrzymuje się sekwent:

$$/A,B,C,/D|\text{-};$$

Wynikiem normalizacji są wszystkie kombinacje zmiennych wejściowych *ABCD* (Rys. 5-7), dla których funkcja *Y2* przyjmuje wartość „*1*”, a funkcja *Y1* wartość „*0*”.




***Rys. 5-7 Siatka dla wyników normalizacji Y2|-Y1***

Drzewo dowodu dla rozpatrywanej zależności funkcji posiada 43 węzły i generowane jest w czasie 0,03s

***Normalizacja sekwentu określającego równoważność pomiędzy funkcjami.***

Możliwe jest wyznaczenie wszystkich różnic w jednym procesie normalizacji bez konieczności analizy obydwu sekwentów, tak jak to miało miejsce w przypadku normalizacji według zasady (5-9) i (5-10), wystarczy poddać analizie jeden z poniższych sekwentów (5-11):

$$|\text{-}Y1 <\text{->} Y2; \quad |\text{-}Y2 <\text{->} Y1; \tag{5-11}$$

gdzie w wyniku podstawienia otrzymuje się:

$$|\text{-}(/A*D)+(/A*/B*C)+(B*C*D)+(A*B*C) <\text{->} (/A*C)+(/A*/B*D)+(B*C*/D);$$

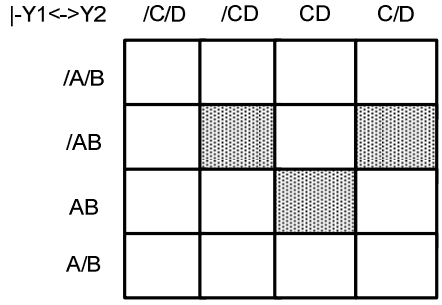
W wyniku normalizacji otrzymuje się sekwenty:

$$/A,B,/C,D|\text{-};$$

$$A,B,C,D|\text{-};$$

$$/A,B,C,/D|\text{-};$$

Wynikowe sekwenty reprezentują wszystkie kombinacje zmiennych, dla których funkcje przyjmują różne wartości (Rys. 5-8). Drzewo dowodu dla rozpatrywanej zależności (równoważności) funkcji posiada 104 węzły i generowane jest w czasie 0,05s




***Rys. 5-8 Siatka dla wyników normalizacji sekwentu |-Y1<->Y2***

Jeżeli z procesu normalizacji tak zbudowanego sekwentu nie powstanie żaden sekwent niebędący tautologią, oznaczało to będzie, że układy realizują tą samą funkcję logiczną.

Powyższy wynik można również uzyskać dzięki zastosowaniu nowego operatora „<+>" będącego zaprzeczeniem równoważności. W tym przypadku analizowane funkcje łączone przy pomocy operatora *XOR* należy umieścić po lewej stronie znaku wynikania logicznego, przygotowanego do normalizacji sekwentu (5-12):

$$Y1<+>Y2|\text{-}; \text{ lub } Y2<+>Y1|\text{-}; \qquad \textbf{(5-12)}$$

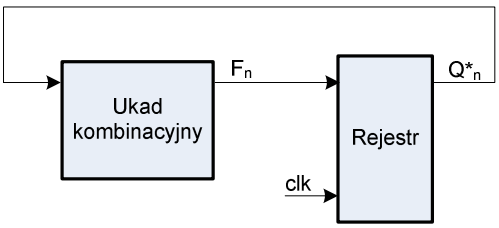
Przyjmując *Y1* za specyfikację układu kombinacyjnego, a *Y2* za jego implementację i stosując opisane powyżej przypadki można wskazać następujące ich zastosowania:

- badanie zgodności specyfikacji z implementacją (|-*Y1*<->*Y2*) lub (|-*Y2*<->*Y1*),
- weryfikacja, czy elementy specyfikacji zostały zrealizowane w implementacji (|-*Y1*->*Y2*) lub (*Y1*|-*Y2*), z uwzględnieniem wariantu, gdzie *Y1* może reprezentować tylko wybrane fragmenty specyfikacji,
- weryfikacja, czy implementacja lub jej część odpowiada założeniom zdefiniowanym specyfikacji (|-*Y2*->*Y1*) lub (*Y2*|-*Y1*).

Istotną zaletą zastosowania wnioskowania symbolicznego do weryfikacji zgodności specyfikacji z implementacją jest to, że system precyzyjnie wskazuje różnice w przypadku wystąpienia niezgodności.

## 5.2. Synteza sekwencyjnych układów cyfrowych

W pracy zakłada się, że synteza układów sekwencyjnych realizowana jest z wykorzystaniem układów rejestrowych. Układ rejestrowy składa się z układu kombinacyjnego i rejestru będącego zbiorem przerzutników (Rys. 5-9). Jest on realizacją sekwencyjnego bloku funkcjonalnego lub automatu sterującego, sekwencyjnego albo współbieżnego [SiMa03][Synt03][AdBa06][BaWę06][StCy07].

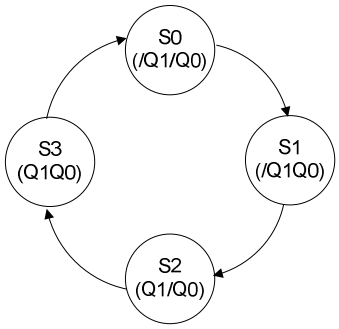



***Rys. 5-9 Schemat blokowy układu rejestrowego***

### 5.2.1. Wyznaczanie funkcji wzbudzeń rejestrów funkcyjnych

W specyfikacji występują zakodowane stany wewnętrzne oraz binarne wejścia i wyjścia. W celu dokładnego zobrazowania sposobu wyznaczania funkcji wzbudzeń przerzutników omówiony zostanie przykład prostego licznika modulo 4 opisanego grafem skierowanym przedstawionym na Rys. 5-10, w którym nie uwzględnia się binarnych wejść sterujących ani funkcji wyjść. Wyjścia układu będą wyprowadzone bezpośrednio z wyjść przerzutników *QN..Q0*.

***Przykład 5-5 Projekt sekwencyjnego licznika mod 4.***




***Rys. 5-10 Graf sekwencyjnego licznika mod4***

Po zakodowaniu stanów *S0..S3* kodem NKB wyznacza się tabelę przejść dla rozpatrywanego układu (Tab. 5-16). Wyznaczona tabela przejść zawiera informację niezbędne dla zbudowania układu z wykorzystaniem przerzutników *D* jak i *JK*.

***Tab. 5-16 Tabela przejść***

| **Stan akt.** | | **Stan nast.** | | **Przerzutniki** | |
|---|---|---|---|---|---|
| **Stan** | **Q1Q0** | **Stan** | **$Q1^*Q0^*$** | **D1 D0** | **J1K1 J0K0** |
| S0 | 0 0 | S1 | 0 1 | 0 1 | 0 - 1 - |
| S1 | 0 1 | S2 | 1 0 | 1 0 | 1 - - 1 |
| S2 | 1 0 | S3 | 1 1 | 1 1 | - 0 1 - |
| S3 | 1 1 | S0 | 0 0 | 0 0 | - 1 - 1 |

Zapisując sekwenty dla automatu według zależności (5-13):

$$\textit{Stan aktualny} \mid\!- \textit{Stan następny;} \qquad \textbf{(5-13)}$$

gdzie w przypadku przerzutnika *D* podstawiając za stan następny *Q*=D* otrzymuje się sekwenty przedstawione w Tab. 5-17.

***Tab. 5-17 Specyfikacja w postaci sekwentowej dla przerzutnika D***

| **Sekwenty definiujące automat** |
|---|
| `/Q1*/Q0|-/D1*D0;` |
| `/Q1*Q0|-D1*/D0;` |
| `Q1*/Q0|-D1*D0;` |
| `Q1*Q0|-/D1*/D0;` |

W wyniku normalizacji sekwentowego opisu układu, a następnie wprowadzenia spójników logicznych otrzymuje się sekwenty złożone, odzwierciedlające funkcje wzbudzeń oraz ich afirmacje dla poszczególnych przerzutników realizowanego układu (Tab. 5-18). Drzewo dowodu będące wynikiem procesu normalizacji posiada 29 węzłów i realizowane jest w czasie 0,01s.

***Tab. 5-18 Reprezentacja sekwentowa funkcji wzbudzeń***

| **Sekwenty wzbudzeń przerzutników D** |
|---|
| /Q1*/Q0 + Q1*Q0\|-/D1; |
| Q0*/Q1 + Q1*/Q0\|-D1; |
| /Q0\|-D0; |
| Q0\|-/D0; |

Równania wzbudzeń przerzutników D mają postać:

$$D0 = /Q0$$

$$D1 = Q0*/Q1 + Q1*/Q0 = Q0 \oplus Q1.$$

lub

$$/D0 = Q0$$

$$/D1 = /Q0*/Q1 + Q1*Q0 = Q0 \otimes Q1.$$

Dzięki uzyskaniu dwóch postaci (dla *D* i */D*) projektant może wybrać rozwiązanie optymalne, np. o mniejszej złożoności.

W przypadku przerzutnika *JK* za stan następny należy podstawić równanie dla stanu następnego tego przerzutnika (Tab. 5-19).

$$Q*=(/Q*J)+(Q*/K).$$

***Tab. 5-19 Specyfikacja w postaci sekwentowej dla przerzutnika JK***

| **Sekwenty definiujące automat** |
|---|
| /Q1*/Q0\|-/(((/Q1*J1)+(Q1*/K1)) * |
| ((/Q0*J0)+(Q0*/K0)); |
| /Q1*Q0\|-(((/Q1*J1)+(Q1*/K1)) * /(((/Q0*J0)+(Q0*/K0)); |
| Q1*/Q0\|-(((/Q1*J1)+(Q1*/K1)) * (((/Q0*J0)+(Q0*/K0)); |
| Q1*Q0\|-/(((/Q1*J1)+(Q1*/K1)) * /(((/Q0*J0)+(Q0*/K0)); |

W wyniku normalizacji i wprowadzania spójników logicznych uzyskuje się sekwenty odzwierciedlające realizowane przez projektowany układ funkcje przedstawione w Tab. 5-20. Drzewo dowodu będące wynikiem procesu normalizacji posiada 62 węzły i realizowane jest w czasie 0,04s.

***Tab. 5-20 Reprezentacja sekwentowa funkcji wzbudzeń***

| Sekwenty wzbudzeń przerzutników JK |
|---|
| /Q1*/Q0\|-/J1; |
| Q0*/Q1\|-J1; |
| Q1*/Q0\|-/K1; |
| Q1*Q0\|-K1; |
| /Q0\|-J0; |
| Q0\|-K0; |

Równania wzbudzeń przerzutników *JK*:

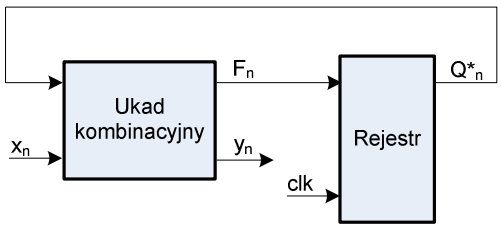
$$J1=Q0*/Q1 \quad K1=Q1*Q0$$

$$J0=/Q0 \quad K0=Q0$$

W przypadku opisu sekwentowego układu dla przerzutnika *JK* możliwe jest rozszerzenie jego specyfikacji polegające na dodaniu sekwentów opisujących stany nieokreślone. Sekwenty takie nie będą miały wpływu na działanie układu, natomiast będą miały wpływ na proces minimalizacji, dzięki czemu równania wzbudzeń mogą osiągnąć jeszcze lepszy stopień minimalizacji.

### 5.2.2. Normalizacja opisu sekwencyjnego układu sterującego opisanego tabelą przejść-wyjść

Specyfikacja sekwencyjnego układu cyfrowego polega na zastąpieniu funkcji przejść i funkcji wyjść regułami przedstawionymi w formie sekwentów [Adam90][AdBa06] [BaWę06]. Projektując układ sterujący wykorzystujący rejestr opierając się o jeden ze znanych typów automatów (Moore'a lub Mealey'ego) należy w bloku kombinacyjnym wyznaczającym funkcje wzbudzeń uwzględnić wejścia sterujące $x_n$ oraz wyjścia $y_n$ (Rys. 5-11).




***Rys. 5-11 Układ sekwencyjny z wykorzystaniem rejestru***

***Przykład 5-6 Realizacja sekwencyjnego układu cyfrowego opisanego tabelą przejść-wyjść.***

Dla celów demonstracyjnych przedstawiono tabelę przejść-wyjść dla przykładowego układu sterującego (Tab. 5-21) opisanego klasycznym automatem Mealey'ego.

***Tab. 5-21 Przykładowa tabela przejść-wyjść Mealey'ego***

| $a_m$ | $K(a_m)$ | $a_s$ | $K(a_s)$ | $X_h$ | $Y_h$ | $\Phi_h$ | h |
|---|---|---|---|---|---|---|---|
| $a_1$ | 000 | $a_2$ | 001 | $x_1$ | $y_1\ y_2$ | $D_3$ | 1 |
| | | $a_2$ | 001 | $/x_1\ /x_2$ | $y_2\ y_5$ | $D_3$ | 2 |
| | | $a_3$ | 010 | $/x_1\ x_2$ | $y_1\ y_3$ | $D_2$ | 3 |
| $a_2$ | 001 | $a_3$ | 010 | $x_3$ | $y_2\ y_5$ | $D_2$ | 4 |
| | | $a_1$ | 000 | $/x_3\ /x_4$ | $y_3\ y_6$ | - | 5 |
| | | $a_4$ | 011 | $/x_3\ x_4$ | $y_4$ | $D_2\ D_3$ | 6 |
| $a_3$ | 010 | $a_1$ | 000 | 1 | $y_3\ y_6$ | - | 7 |
| $a_4$ | 011 | $a_1$ | 000 | $x_3$ | $y_2\ y_3\ y_4$ | - | 8 |
| | | $a_5$ | 100 | $/x_3$ | - | $D_1$ | 9 |
| $a_5$ | 100 | $a_1$ | 000 | $x_4$ | $y_2\ y_3\ y_4$ | - | 10 |
| | | $a_1$ | 000 | $/x_4$ | y3 y6 | - | 11 |

Na podstawie posiadanej tabeli przejść wyjść sporządza się opis działania układu w postaci sekwentowej. Postępując według następującej reguły:

*Stan aktualny * Warunek |- Stan następny * Wyjścia* ***(5-14)***

poprzez zastosowanie jej do każdego wiersza tabeli przejść-wyjść, otrzymano następującą specyfikację układu (Tab. 5-22):

***Tab. 5-22 Reprezentacja sekwentowa tablicy przejść-wyjść***

| **Specyfikacja układu** |
|---|
| /Q1*/Q2*/Q3*x1\|-/D1*/D2*D3*y1*y2; |
| /Q1*/Q2*/Q3*/x1*/x2\|-/D1*/D2*D3*y2*y5; |
| /Q1*/Q2*/Q3*/x1*x2\|-/D1*D2*/D3*y1*y3; |
| #---------- |
| /Q1*/Q2*Q3*x3\|-/D1*D2*/D3*y2*y5; |
| /Q1*/Q2*Q3*/x3*/x4\|-/D1*/D2*/D3*y3*y6; |
| /Q1*/Q2*Q3*/x3*x4\|-/D1*D2*D3*y4; |
| #---------- |
| /Q1*Q2*/Q3\|-/D1*/D2*/D3*y3*y6; |
| #---------- |
| /Q1*Q2*Q3*x3\|-/D1*/D2*/D3*y2*y3*y4; |
| /Q1*Q2*Q3*/x3\|-D1*/D2*/D3; |
| #---------- |
| Q1*/Q2*/Q3*x4\|-/D1*/D2*/D3*y2*y3*y4; |
| Q1*/Q2*/Q3*/x4\|-/D1*/D2*/D3*y3*y6; |

Analogicznie do metody wyznaczania funkcji wzbudzeń rejestru pamięci opisanej w rozdziale 5.2.1 specyfikację poddaje się normalizacji, a następnie rozwiązania łączy się w sekwenty złożone wprowadzając spójniki logiczne. W wyniku otrzymuje się sekwenty odzwierciedlające równania wzbudzeń przerzutników uwzględniające warunki przejść oraz równania wyjść (Tab. 5-23). Równania wyjść również zależą od warunków przejść, co jest elementem charakterystycznym dla rozpatrywanego automatu Mealey'ego.

***Tab. 5-23 Znormalizowana reprezentacja sekwentowa dla tablicy przejść-wyjść***

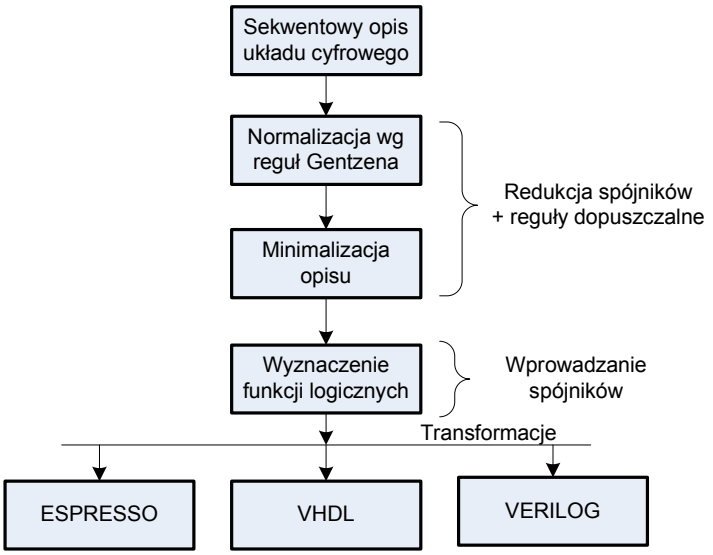
| **Specyfikacja układu** |
|---|
| Q2*Q3*/Q1*/x3\|-D1;<br>x2*/Q1*/Q2*/Q3*/x1 + Q3*x3*/Q1*/Q2 + Q3*x4*/Q1*/Q2\|-D2;<br>x1*/Q1*/Q2*/Q3 + Q3*x4*/Q1*/Q2*/x3 + /Q1*/Q2*/Q3*/x2\|-D3;<br><br>x1*/Q1*/Q2*/Q3 + x2*/Q1*/Q2*/Q3\|-y1;<br>x1*/Q1*/Q2*/Q3 + Q1*x4*/Q2*/Q3 + /Q1*/Q2*/Q3*/x2 +<br>Q3*x3*/Q1\|-y2;<br>Q3*/Q1*/Q2*/x3*/x4 + Q2*/Q1*/Q3 + x2*/Q1*/Q3*/x1 + Q2*x3*/Q1<br>+ Q1*/Q2*/Q3\|-y3;<br>Q3*x4*/Q1*/Q2*/x3 + Q2*Q3*x3*/Q1 + Q1*x4*/Q2*/Q3\|-y4;<br>/Q1*/Q2*/Q3*/x1*/x2 + Q3*x3*/Q1*/Q2\|-y5;<br>Q3*/Q1*/Q2*/x3*/x4 + Q2*/Q1*/Q3 + Q1*/Q2*/Q3*/x4\|-y6; |

Drzewo dowodu z procesu normalizacji zawiera 311 węzłów i jest realizowane w czasie 3,68s. Stosunkowo długi czas realizacji przy niewielkiej ilości węzłów spowodowany jest realizacją metody rezolucji, która po normalizacji wykonała dodatkową minimalizację wygenerowanego opisu.

## 5.3. Współpraca systemu automatycznego wnioskowania z kompilatorami języków opisu sprzętu

Współpraca systemu wnioskującego z kompilatorami języków opisu sprzętu HDL (***H**ardware **D**escription **L**anguage*) [Zwol07] umożliwia dodatkową symulację i weryfikację projektowanego układu, a także jego ewentualną syntezę i implementację na przykład w strukturach FPGA. Przy pomocy języka logiki sekwentów Gentzena można opisywać cyfrowe układy kombinacyjne jak i sekwencyjne. W wyniku normalizacji opisu, poprzez stosowanie reguł eliminacji spójników oraz dodatkowych reguł dopuszczalnych, otrzymuje się zbór relacji opisujących układ cyfrowy. Następnie poprzez zastosowanie reguł wprowadzania spójników wyznacza się sekwenty opisujące poszczególne funkcjonalności układu będące odpowiednikami funkcji logicznych realizowanych przez projektowany układ (Rys. 5-12). Bardzo podobny sposób wykonuje się transformację do formatu ESPRESSO. Format ten poza wykorzystaniem w programach minimalizacji

funkcji logicznych, często spotyka się w akademickich programach CAD służących do projektowania układów cyfrowych.

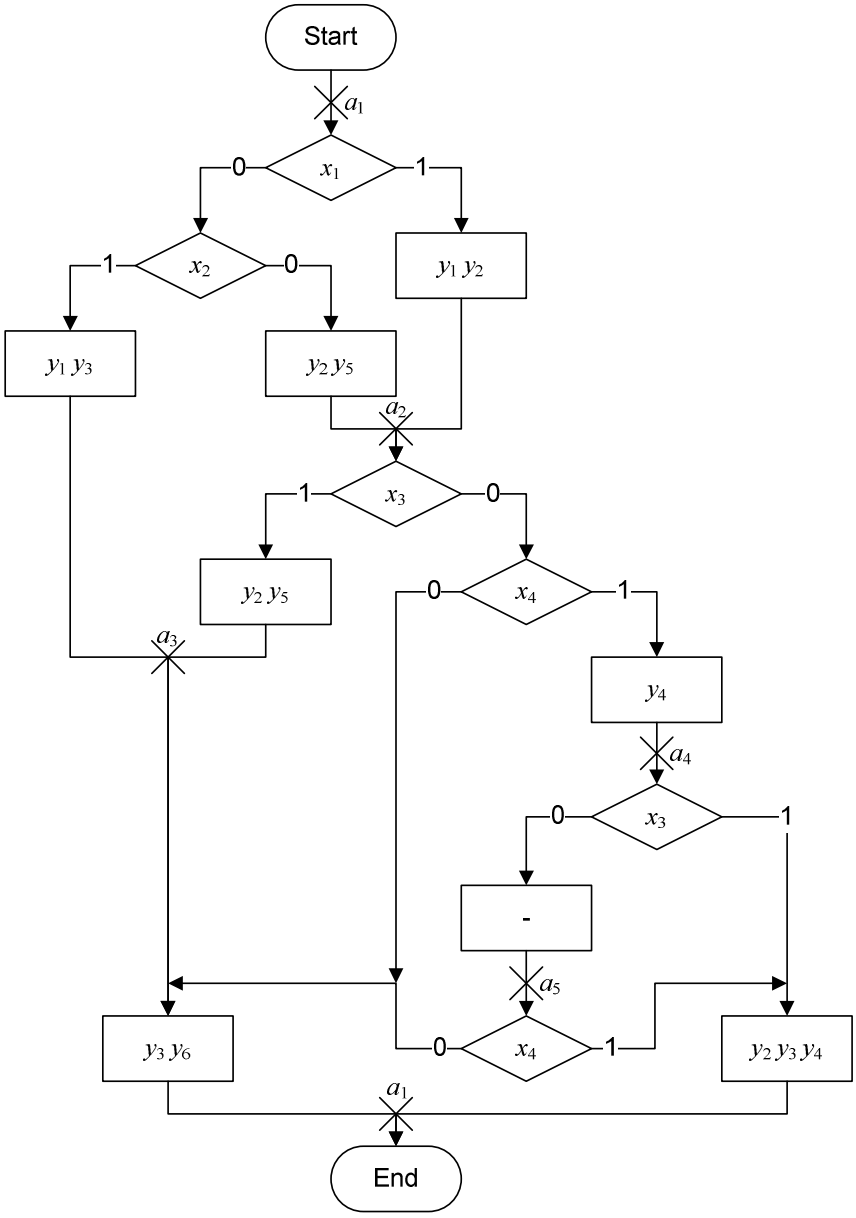



***Rys. 5-12 Schemat blokowy wykorzystania języków HDL***

***Przykład 5-7 Realizacja cyfrowego układu sekwencyjnego opisanego siecią działań.***

Jako przykład współpracy systemu wnioskującego z językami HDL zrealizowany zostanie układ sekwencyjny przedstawiony za pomocą oznakowanej sieci działań [Buko08] (Rys. 5-13). Zastosowanie sieci działań do opisu funkcjonowania układu sekwencyjnego opisano w literaturze [BaWę06].

Oznakowanie (*a1..a5*) zostało wprowadzone w celu wskazania miejsc odpowiadających stanom w układzie sekwencyjnym. Wyjścia układu będą generowane przy przejściach i zależeć będą od warunku przejścia, więc rozpatrywany układ będzie automatem Mealy'go. Celem przykładu jest pokazanie możliwości wstępnej weryfikacji formalnej metod opartych na intuicji, które następnie są uszczegóławiane i algorytmizowane, będąc podstawą do konstrukcji nowatorskich systemów CAD [AdBa06][BaWę06].




***Rys. 5-13 Oznakowana sieć działań***

Sieć działań można przedstawić w postaci tabeli przejść-wyjść, tak jak zostało to przedstawione w rozdziale 5.2.2, a następnie tabelę zapisać w postaci sekwentów i poddać normalizacji. W przypadku zastosowania logiki sekwentów tworzenie tablicy może zostać pominięte, gdyż sekwenty można utworzyć bezpośrednio z sieci działań według następujących schematów (3-15), (3-16):

*Stan aktualny * Warunek |- Stan następny*
*Stan aktualny |- Wyjścia* ***(5-15)***

lub:

$$Stan\ aktualny\ *\ Warunek\ |-\ Stan\ następny\ *\ Wyjścia \qquad \textbf{(5-16)}$$

W praktyce, dla sieci działań reguły przejść i wyjść zapisuje się łącznie w postaci blokowej (5-17):

$$Stan\ aktualny\ |-\ \prod(Warunek\ ->\ (Stan\ następny\ *\ Wyjścia)) \qquad \textbf{(5-17)}$$

Stan aktualny będzie iloczynem formuł zbudowanych na podstawie przyjętej tabeli kodowania stanów (Tab. 5-24).

***Tab. 5-24 Tabela kodowania stanów***

| Stan | Q1 | Q2 | Q3 |
|---|---|---|---|
| **A1** | 0 | 0 | 0 |
| **A2** | 0 | 0 | 1 |
| **A3** | 0 | 1 | 0 |
| **A4** | 0 | 1 | 1 |
| **A5** | 1 | 0 | 0 |

Stany następne należy opisać wyprowadzając je z równania przerzutnika. W omawianym przykładzie zastosowany zostanie przerzutnik D, dla którego równanie dla stanu następnego ma postać:

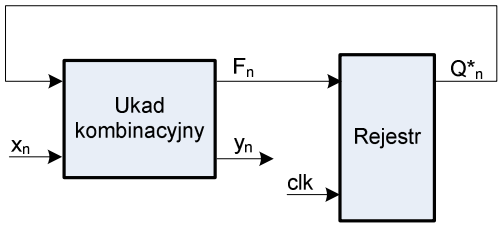
$$Q' = D$$

Budując sekwenty według zaproponowanej metody uzyska się reprezentację sekwentową dla rozpatrywanej sieci działań (Tab. 5-25). Niezbędne jest określenie typu formuł (Wejściowe/Wyjściowe), gdyż z definicji tej korzysta mechanizm wprowadzania spójników logicznych i składania sekwentów znormalizowanych w złożone opisy. Brak definicji typów formuł uniemożliwi transformację na języki opisu sprzętu, a także na format ESPRESSO.

***Tab. 5-25 Reprezentacja sekwentowa sieci działań***

| **Specyfikacja układu** |
|---|
| IN:Q1,Q2,Q3,x1,x2,x3,x4; |
| OUT:y1,y2,y3,y4,y5,y6,D1,D2,D3; |
| #---------- |
| /Q1*/Q2*/Q3\|-(x1->/D1*/D2*D3*y1*y2) * (/x1*/x2->/D1*/D2*D3*y2*y5) * (/x1*x2->/D1*D2*/D3*y1*y3); |
| #---------- |
| /Q1*/Q2*Q3\|-(x3->/D1*D2*/D3*y2*y5) * (/x3*/x4->/D1*/D2*/D3*y3*y6) * (/x3*x4->/D1*D2*D3*y4); |
| #---------- |
| /Q1*Q2*/Q3\|-/D1*/D2*/D3*y3*y6; |
| #---------- |
| /Q1*Q2*Q3\|-(x3->/D1*/D2*/D3*y2*y3*y4) * (/x3->D1*/D2*/D3); |
| #---------- |
| Q1*/Q2*/Q3\|-(x4->/D1*/D2*/D3*y2*y3*y4) * (/x4->/D1*/D2*/D3*y3*y6); |

W wyniku normalizacji specyfikacji wyznaczone zostaną sekwenty definiujące równania wzbudzeń przerzutników oraz ich afirmacje, a także równania dla poszczególnych wyjść układu. Dzięki wyznaczeniu równań wzbudzeń przerzutników, a także ich afirmacji projektant może wybrać lepsze rozwiązanie poprzez wybór tych funkcji, które składają się z mniejszej liczby termów lub termów o mniejszej liczbie zmiennych. W przypadku realizacji układu sekwencyjnego konieczne jest dołączenie do wygenerowanego układu rejestru (Rys. 5-14) analogicznie jak przy projektowaniu układów sekwencyjnych z wykorzystaniem tabeli przejść-wyjść (rozdział 5.2.2).




***Rys. 5-14 Układ sekwencyjny z wykorzystaniem rejestru***

W przypadku układów kombinacyjnych wygenerowany układ można bezpośrednio syntezować do struktury FPGA, bez definiowania rejestru.

## *VHDL*

***Przykład 5-8 Transformacja specyfikacji w języku sekwentów na język VHDL***

W wyniku normalizacji opisu układu w postaci sekwentowej i wprowadzeniu spójników logicznych powstałe złożone sekwenty odpowiadające funkcjom realizowanego układu stosunkowo łatwo można przekształcić do postaci języka VHDL. Nadmiarowości w postaci wyznaczonych $F^0$ lub w specyficznych przypadkach wartości nieokreślonych należy $F^-$ należy pominąć.

Dla sieci działań (Rys. 5-13), specyfikacji (Tab. 5-25) oraz po odrzuceniu ewentualnych elementów nadmiarowych ($F^0$ i $F^-$), w wyniku normalizacji i wprowadzania spójników logicznych, powstały sekwenty reprezentujące poszczególne funkcje układu (Tab. 5-26).

***Tab. 5-26 Sekwenty reprezentujące poszczególne funkcje układu***

| **Sekwenty dla wzbudzeń przerzutników oraz wyjść układu** |
|---|
| `Q2*Q3*/Q1*/x3|-D1;`<br>`x2*/Q1*/Q2*/Q3*/x1 + Q3*x3*/Q1*/Q2 + Q3*x4*/Q1*/Q2|-D2;`<br>`x1*/Q1*/Q2*/Q3 + Q3*x4*/Q1*/Q2*/x3 + /Q1*/Q2*/Q3*/x2|-D3;`<br><br>`x1*/Q1*/Q2*/Q3 + x2*/Q1*/Q2*/Q3|-y1;`<br>`x1*/Q1*/Q2*/Q3 + Q1*x4*/Q2*/Q3 + /Q1*/Q2*/Q3*/x2 + Q3*x3*/Q1|-y2;`<br>`Q3*/Q1*/Q2*/x3*/x4 + Q2*/Q1*/Q3 + x2*/Q1*/Q3*/x1 + Q2*x3*/Q1 +`<br>`Q1*/Q2*/Q3|-y3;`<br>`Q3*x4*/Q1*/Q2*/x3 + Q2*Q3*x3*/Q1 + Q1*x4*/Q2*/Q3|-y4;`<br>`/Q1*/Q2*/Q3*/x1*/x2 + Q3*x3*/Q1*/Q2|-y5;`<br>`Q3*/Q1*/Q2*/x3*/x4 + Q2*/Q1*/Q3 + Q1*/Q2*/Q3*/x4|-y6;` |

Drzewo dowodu z procesu normalizacji zawiera 222 węzły i jest realizowane w czasie 3,62s. Na czas normalizacji znaczny wpływ ma metoda rezolucji wykonująca dodatkową minimalizację wygenerowanego opisu.

W wyniku proponowanej transformacji funkcji z postaci sekwentów na postać VHDL wygenerowano następujący kod w języku opisu sprzętu, zamieszony w Tab. 5-27.

***Tab. 5-27 Specyfikacja układu w języku VHDL***

**Plik „układ.vhdl”**

```
library IEEE;
use IEEE.std_logic_1164.all;

entity uklad is
port (
    Q1: in STD_LOGIC;
    Q2: in STD_LOGIC;
    Q3: in STD_LOGIC;
    x1: in STD_LOGIC;
    x2: in STD_LOGIC;
    x3: in STD_LOGIC;
    x4: in STD_LOGIC;
    D1: out STD_LOGIC;
    D2: out STD_LOGIC;
    D3: out STD_LOGIC;
    y1: out STD_LOGIC;
    y2: out STD_LOGIC;
    y3: out STD_LOGIC;
    y4: out STD_LOGIC;
    y5: out STD_LOGIC;
    y6: out STD_LOGIC
);
end uklad;

architecture uklad_arch of uklad is
begin
    D3<=(not Q1 and not Q2 and not Q3 and not x1 and not x2) or
        (Q3 and x4 and not Q1 and not Q2 and not x3);
    y2<=(not Q1 and not Q2 and not Q3 and not x1 and not x2) or
        (Q1 and x4 and not Q2 and not Q3) or
        (Q3 and x3 and not Q1);
    y5<=(not Q1 and not Q2 and not Q3 and not x1 and not x2) or
        (Q3 and x3 and not Q1 and not Q2);
    D2<=(x2 and not Q1 and not Q2 and not Q3 and not x1) or
        (Q3 and x4 and not Q1 and not Q2 and not x3);
    y1<=(x2 and not Q1 and not Q2 and not Q3 and not x1);
    y3<=(Q3 and not Q1 and not Q2 and not x3 and not x4) or
        (Q2 and not Q1 and not Q3) or
        (Q1 and not Q2 and not Q3 and not x4) or
        (x2 and not Q1 and not Q3 and not x1) or
        (Q2 and x3 and not Q1);
    y6<=(Q3 and not Q1 and not Q2 and not x3 and not x4) or
        (Q2 and not Q1 and not Q3) or
        (Q1 and not Q2 and not Q3 and not x4);
    y4<=(Q3 and x4 and not Q1 and not Q2 and not x3) or
        (Q2 and Q3 and x3 and not Q1) or
        (Q1 and x4 and not Q2 and not Q3);
    D1<=(Q2 and Q3 and not Q1 and not x3);
End uklad_arch;
```

## *VERILOG*

***Przykład 5-9 Transformacja specyfikacji w języku sekwentów na język VERILOG***

W wyniku proponowanej transformacji funkcji z postaci sekwentów na postać VERILOG, analogicznie jak w przypadku języka VHDL, wygenerowano następujący kod w języku opisu sprzętu Tab. 5-29.

***Tab. 5-28 Specyfikacja układu w języku Verilog***

**Plik „układ.v”**

```
module uklad(Q1,Q2,Q3,x1,x2,x3,x4,D1,D2,D3,y1,y2,y3,y4,y5,y6);

   input Q1,Q2,Q3,x1,x2,x3,x4;
   output D1,D2,D3,y1,y2,y3,y4,y5,y6;

   assign D3=(x1 & ~Q1 & ~Q2 & ~Q3) | (Q3 & x4 & ~Q1 & ~Q2 & ~x3) |
             (~Q1 & ~Q2 & ~Q3 & ~x2);
   assign y1=(x1 & ~Q1 & ~Q2 & ~Q3) | (x2 & ~Q1 & ~Q2 & ~Q3);
   assign y2=(x1 & ~Q1 & ~Q2 & ~Q3) | (Q1 & x4 & ~Q2 & ~Q3) |
             (~Q1 & ~Q2 & ~Q3 & ~x2) | (Q3 & x3 & ~Q1);
   assign y5=(~Q1 & ~Q2 & ~Q3 & ~x1 & ~x2) | (Q3 & x3 & ~Q1 & ~Q2);
   assign D2=(x2 & ~Q1 & ~Q2 & ~Q3 & ~x1) | (Q3 & x3 & ~Q1 & ~Q2) |
             (Q3 & x4 & ~Q1 & ~Q2);
   assign y3=(Q3 & ~Q1 & ~Q2 & ~x3 & ~x4) | (Q2 & ~Q1 & ~Q3) |
             (x2 & ~Q1 & ~Q3 & ~x1) | (Q2 & x3 & ~Q1) |
             (Q1 & ~Q2 & ~Q3);
   assign y6=(Q3 & ~Q1 & ~Q2 & ~x3 & ~x4) | (Q2 & ~Q1 & ~Q3) |
             (Q1 & ~Q2 & ~Q3 & ~x4);
   assign y4=(Q3 & x4 & ~Q1 & ~Q2 & ~x3) | (Q2 & Q3 & x3 & ~Q1) |
             (Q1 & x4 & ~Q2 & ~Q3);
   assign D1=(Q2 & Q3 & ~Q1 & ~x3);

endmodule
```

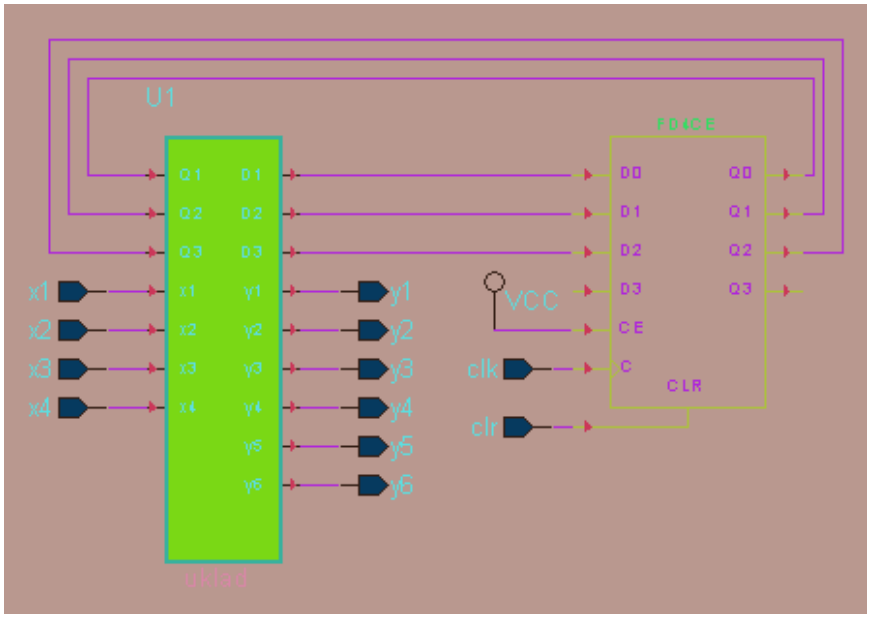
Przedstawione przykłady współpracy systemu wnioskującego według reguł Gentzena z kompilatorami języków opisu sprzętu mają na celu zaproponowanie sposobu współpracy obu systemów, natomiast sam wybór języka opisu sprzętu jest uzależniony od indywidualnych preferencji inżyniera-projektanta.

Opis behawioralny rejestrowego bloku funkcyjnego przedstawia się w postaci zbioru sekwentów, które po normalizacji automatycznie przedstawiane są w formie akceptowalnej przez język opisu sprzętu. Przed symulacją behawioralną można dokonać symulacji symbolicznej, tak jak to pokazano w rozdziale 5 (Przykład 5-3).

***Przykład 5-10 Symulacja i synteza układu rejestrowego***

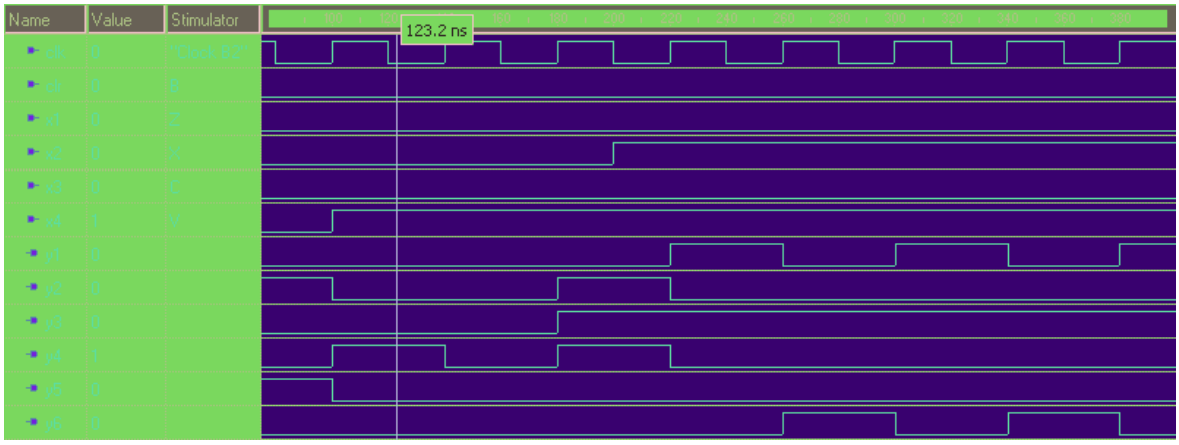
Do wykonania symulacji projektowanego układu wykorzystano narzędzie ActiveHDL v7.3 firmy ALDEC. W celu lepszego zobrazowania sposobu projektowania układu zdecydowano się na przedstawienie projektu w formie diagramu blokowego (Rys. 5-15), gdzie element „*U1-uklad*” jest opisany wygenerowanym w języku VHDL blokiem funkcyjnym, a element FD4CE rejestrem złożonym z czterech przerzutników typu D. Ten

sposób projektowania układu nie jest jednak wymogiem, ponieważ specyfikacja całego układu mogła zostać przygotowana na przykład tylko w języku HDL, bez użycia diagramu blokowego.




***Rys. 5-15 Układ rejestrowy***

Używając narzędzia do symulacji z pakietu Active-HDL przeprowadzono badanie poprawności funkcjonowania projektowanego układu. Na Rys. 5-16 przedstawiono fragment przebiegu symulacji. Wyniki symulacji są zgodne z założeniami zdefiniowanymi przy pomocy sieci działań. Na wygenerowanym przebiegu można zaobserwować przejście ze stanu *a1* do *a2* i wygenerowanie wyjść *y2* i *y5* dla sygnałów sterujących będących zerami. Następnie po pojawieniu się sygnału sterującego *x4* układ przechodzi do stanu *a4* generując wyjście *y4*. Dalej przechodzi do stanu *a5*, nie generując żadnych sygnałów wyjściowych. Ze stanu *a5* system wraca do stanu początkowego *a1* i generuje wyjścia *y2,y3* i *y4*. Od momentu pojawienia się kolejnego sygnału sterującego *x2*, układ przełącza stany *a1* i *a3*, generując na przemian wyjścia *y1* i *y3* oraz *y3* i *y6*.




***Rys. 5-16 Symulacja w środowisku Active-HDL***

Po dodaniu do schematu blokowego wymaganych buforów wejściowych i wyjściowych oraz globalnego bufora dla sygnału zegara przeprowadzona została przykładowa synteza i implementacja projektowanego układu. W procesie syntezy i implementacji wykorzystany został pakiet Xilinx ISE v9.1. Jako układ docelowy wybrano posiadany układ firmy Xilinx z rodziny spartan2e o następujących danych znamionowych:

- Device: XC2S200e;
- Package: PQ208;
- Speed: -6.

Wyniki przeprowadzonej próbnej syntezy zostały przedstawione w Tab. 5-29.

***Tab. 5-29 Przykładowe wyniki syntezy układu***

**Raport z procesu syntezy**

```
# IO Buffers                        : 12
#      IBUF                         : 5
#      IBUFG                        : 1
#      OBUF                         : 6
# Others                            : 1
#      FD4CE                        : 1
=========================================================================

Device utilization summary:
---------------------------

Selected Device : 2s200epq208-6

 Number of Slices:                      10  out of   2352     0%
 Number of 4 input LUTs:                19  out of   4704     0%
 Number of IOs:                         12
 Number of bonded IOBs:                 12  out of    142     8%
```

Kolejnym krokiem było przygotowanie pliku **.ucf* opisującego wyprowadzenia układu i przeprowadzenie procesu implementacji, w wyniku którego powstał plik „bitstream" odzwierciedlający strukturę projektowanego układu.

## 5.4. Współpraca systemu automatycznego wnioskowania z systemami uniwersyteckimi

Możliwość współpracy z systemami uniwersyteckimi została osiągnięta poprzez zastosowanie transformacji opisu znormalizowanego do standardu zapisu funkcji ESPRESSO, opracowanego przez uniwersytet Berkeley [ŁuJa97].

***Przykład 5-11 Transformacja specyfikacji w języku sekwentów na format ESPRESSO.***

Jako przykład przedstawiono transformację znormalizowanej specyfikacji sieci działań (Rys. 5-13) przedstawionej w Tab. 5-26 na format ESPRESSO. Wynik transformacji zawarto w Tab. 5-30.

***Tab. 5-30 Specyfikacja układu w formacie ESPRESSO***

**Format ESPRESSO**

```
.type f
.i 7
.o 9
.p 41
.ilb Q1 Q2 Q3 x1 x2 x3 x4
.ob D1 D2 D3 y1 y2 y3 y4 y5 y6

0001---  --1------
0001---  ---1-----
0001---  ----1----
00000--  -------1-
00001--  -1-------
001--1-  -1-------
001--1-  -------1-
001--00  -----1---
001--00  --------1
001--01  --1------
001--01  ------1--
010----  --0------
010----  -----1---
010----  --------1
011--1-  ------1--
011--0-  1--------
100---1  ----1----
100---1  ------1--
100---0  --------1
0-01---  -0-------
000-0--  --1------
000-1--  ---1-----
000-0--  ----1----
0-001--  --0------
0-001--  -----1---
001---1  -1-------
001---0  --0------
0-1--1-  --0------
0-1--1-  ----1----
0-1--00  -0-------
01---0-  --0------
01---1-  -----1---
100----  -0-------
100----  --0------
100----  -----1---
0-0-0--  -0-------
01-----  -0-------
0-0----  0--------
0----1-  0--------
-00----  0--------
00-----  0--------
.e
```

Format ten wykorzystywany jest głównie w uniwersyteckich systemach przeznaczonych do projektowania układów cyfrowych (DEMAIN, SIS, MIS), jednakże jego popularność sprawia, że pojawia się również w zastosowaniach komercyjnych. Ze względu na brak optymalizacji generowanego formatu ESPRESSO możliwa jest dodatkowa minimalizacja i optymalizacja opisu z wykorzystaniem systemu ESPRESSO (Tab. 5-31).

***Tab. 5-31 Specyfikacja układu w formacie ESPRESSO po dodatkowej minimalizacji***

| **Format ESPRESSO** |
|---|
| .i 7<br>.o 9<br>.ilb Q1 Q2 Q3 x1 x2 x3 x4<br>.ob D1 D2 D3 y1 y2 y3 y4 y5 y6<br>.p 11<br>001--00 000001001<br>001--01 011000100<br>00000-- 001010010<br>00001-- 010101000<br>011--0- 100000000<br>100---0 000001001<br>011--1- 000011100<br>100---1 000011100<br>001--1- 010010010<br>0001--- 001110000<br>010---- 000001001<br>.e |

Optymalizacja i minimalizacja opisu z wykorzystaniem systemu ESPRESSO stanowi dowód poprawności jego zbudowania. Udoskonalenie procesu transformacji i optymalizacja generowanego formatu ESPRESSO znajduje się w planach dalszych prac nad rozwojem systemu wnioskowania symbolicznego i możliwościami jego wykorzystania w procesach projektowania układów sterowania binarnego.

# 6. Wykorzystanie komputerowego wnioskowania do analizy współbieżnych układów dyskretnych

Ze względu na swój uniwersalny charakter wnioskowanie symboliczne znajduje także zastosowanie procesie projektowania układów współbieżnych. W niniejszym rozdziale przedstawionych kilka interesujących propozycji dotyczących głównie analizy układów współbieżnych, gdzie wiele znanych klasycznych metod algebraicznych zawodzi lub nie udostępnia wyczerpującej informacji wynikającej z samego procesu analizy.

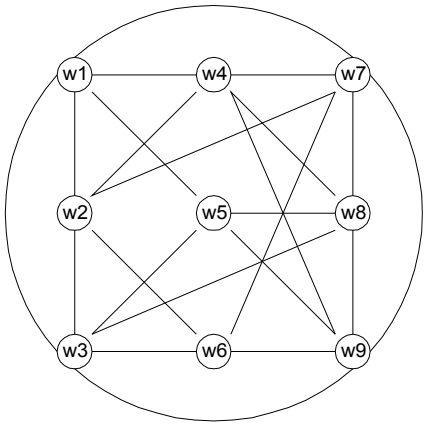
## 6.1. Analiza symboliczna grafów współbieżności

Teoria grafów często znajduje swoje zastosowanie w technice cyfrowej. Badanie własności grafów odzwierciedlających funkcjonowanie układu cyfrowego dostarcza dużo istotnych informacji o projektowanym układzie, jak i procesach w nim zachodzących.

W przypadku grafów niezorientowanych, naturalne wnioskowanie Gentzena można wykorzystać do wyszukiwania maksymalnych niezależnych zbiorów wierzchołków, które stanowi etap w procesie kolorowania grafu. W przypadku, gdy analizowany graf reprezentuje współbieżność między procesami w projektowanym systemie dyskretnym, wyznaczone kolorowania mogą posłużyć do dekompozycji układu na współbieżne (równoległe) składowe automatowe.

***Przykład 6-1 Analiza grafu współbieżności***

W celu przedstawienia problemu wyznaczania maksymalnych zbiorów wierzchołków niezależnych, przyjęto następujący graf (Rys. 6-1):




***Rys. 6-1 Graf współbieżności***

Poszczególne krawędzie grafu zapisano w postaci iloczynu sum (postać koniunkcyjna, po lewej stronie znaku wynikania logicznego), w formie częściowo znormalizowanej (Tab. 6-1) [Deo80].

***Tab. 6-1 Sekwent sąsiedztwa wierzchołków***

```
(w1+w2),(w1+w3),(w1+w4),(w1+w5),
(w1+w7),(w2+w3),(w2+w4),(w2+w6),
(w2+w7),(w3+w5),(w3+w6),(w3+w8),
(w3+w9),(w4+w7),(w4+w8),(w4+w9),
(w5+w8),(w5+w9),(w6+w7),(w6+w9),
(w7+w8),(w7+w9),(w8+w9)|-;
```

W wyniku przekształceń wyrażenia (Tab. 6-1) definiującego wszystkie krawędzie grafu przez wymienienie wierzchołków incydentnych, otrzymuje się sekwenty przedstawiające bazy grafu (Tab. 6-2). Drzewo dowodu dla rozpatrywanego sekwentu posiada 162 węzły i realizowane jest w czasie 0,12s.

***Tab. 6-2 Wyznaczone bazy grafu***

```
W1,w2,w3,w4,w8,w9,w6|-;
w1,w2,w3,w7,w8,w9|-;
w1,w2,w5,w6,w8,w9,w4|-;
w1,w2,w5,w6,w8,w9,w7|-;
w1,w3,w4,w6,w7,w5,w8|-;
w1,w3,w4,w6,w7,w5,w9|-;
w1,w3,w4,w6,w7,w8,w9|-;
w2,w3,w4,w5,w7,w6,w8|-;
w2,w3,w4,w5,w7,w9|-;
```

Dopełnienia poszczególnych wyrażeń wynikowych są maksymalnymi zbiorami wierzchołków niezależnych grafu:

*{1,6,8},{1,9},{2,5},{2,8},{2,9},{3,4},{3,7},{4,5,6},{5,7}*

Dodanie do któregokolwiek zbioru dodatkowego wierzchołka spowoduje, że zbiór przestaje być niezależny.

Bardzo ważnym aspektem dotyczącym wykorzystania wnioskowania gentzenowskiego do analizy symbolicznej grafów jest to, że pierwsze rozwiązanie często uzyskuje się już po kilku krokach normalizacji. Na przykład, dla rozpatrywanego grafu (Rys. 6-1) pierwsze rozwiązanie wyznaczone jest już po 47 krokach, w czasie 0,06s. W wielu przypadkach, a w szczególności zastosowań praktycznych, często pierwsze rozwiązanie bywa rozwiązaniem wystarczającym. W odróżnieniu od większości algorytmów heurystycznych w rozsądnym czasie można uzyskać wszystkie możliwe rozwiązania i formalny dowód ich poprawności.

Wyznaczanie maksymalnych zbiorów niezależnych jest etapem pośrednim w algorytmie kolorowania grafu [Odmi02](rozdz. 6.2).

W celu wyznaczenia minimalnych zbiorów dominujących sporządza się sekwenty, opisujące zbiory wierzchołków przyległych dla każdego poszczególnego wierzchołka (Tab. 6-3), w formie dysjunkcji.

***Tab. 6-3 Sekwent zbiorów wierzchołków przyległych***

```
(w1+w2+w3+w4+w5+w7),
(w2+w3+w1+w4+w6+w7),
(w3+w1+w2+w5+w6+w8+w9),
(w4+w1+w2+w7+w8+w9),
(w5+w1+w3+w8+w9),
(w6+w2+w3+w7+w9),
(w7+w1+w2+w4+w6+w8+w9),
(w8+w3+w4+w5+w7+w9),
(w9+w3+w4+w5+w6+w7+w8)|-;
```

W wyniku normalizacji sekwentu otrzymano opis w postaci znormalizowanej, gdzie poszczególne sekwenty odpowiadają minimalnym zbiorom dominującym (Tab. 6-4). Drzewo dowodu dla sekwentu sąsiedztwa grafu (Tab. 6-3) posiada 690 wierzchołków i realizowane jest w czasie 0,26s.

***Tab. 6-4 Sekwenty reprezentujące minimalne zbiory dominujące grafu***

```
w1,w6,w8|-;    w1,w9|-;
w2,w8|-;       w2,w9|-;
w3,w1|-;       w3,w2|-;
w3,w8|-;       w3,w9|-;
w4,w3|-;       w4,w2,w1|-;
w4,w6,w1|-;    w4,w8,w6|-;
w4,w9|-;       w5,w2|-;
w5,w6,w4|-;    w5,w6,w1|-;
w5,w6,w8|-;    w5,w6,w9|-;
w7,w3|-;       w7,w1|-;
w7,w5|-;       w7,w8|-;
w7,w9|-;
```

Minimalne zbiory dominujące mają następująca postać:

*{1,6,8},{2,8},{1,3},{3,8},{3,4},{1,4,6},{4,9},{4,5,6},{5,6,8},{3,7},{7,5},{7,9},*

*{1,9},{2,9},{2,3},{3,9},{1,2,4},{4,6,8},{2,5},{1,5,6},{5,6,9},{1,7},{7,8}*

Znajomość zbiorów dominujących może być wykorzystana w algorytmie kodowania miejsc sieci Petriego [Adam91][AdWę07].

## 6.2. Kolorowanie automatowe sieci Petriego i dekompozycja równoległa

Podczas projektowania sterownika cyfrowego często zdarza się, że jego rozmiary przekraczają fizyczne możliwości dostępnego elementu. Zjawisko takie stawia projektanta przed wyborem większego elementu cyfrowego lub zastosowaniem dekompozycji i podzieleniem projektowanego układu na szereg mniejszych podukładów. Dodatkowo, zdekomponowanie bardzo złożonego układu ułatwia proces projektowania. Łatwiej projektuje się kilka układów, ale za to o mniejszej złożoności. Rezultaty dekompozycji mogą być również wykorzystane do efektywnego kodowania lokalnych stanów wewnętrznych układu sterującego. Pokrycie sieci Petriego składowymi automatowymi świadczy o możliwości jej efektywnej implementacji układowej lub sprzętowej [BiAd94] [Mura89].

Regułowy opis funkcjonowania układu cyfrowego, w postaci sekwentów, w naturalny sposób odzwierciedla zależności przyczynowo-skutkowe w pewnym zbiorze zdarzeń, identyfikowanym ze współbieżnymi operacjami układu cyfrowego [Adam90][Adam91] [BaKu93][Synt03]. Efektywnym sposobem modelowania układów współbieżnych są sieci Petriego. Teoria sieci Petriego stanowi obszerną i szybko rozwijającą się dziedzinę nauki, która odgrywa ważną rolę w projektowaniu systemów informatycznych, a także w planowaniu i sterowaniu przepływem produkcji [Mura89][Szpy08]. Bogaty aparat matematyczny umożliwia badanie ich własności, takich jak żywotność, czy ograniczoność [SuSz99]. Analiza tych własności metodami symbolicznymi pozwala na wykrycie zastojów w sterownikach logicznych [SuSz99][PaCo01][Węgr03][AdWę07].

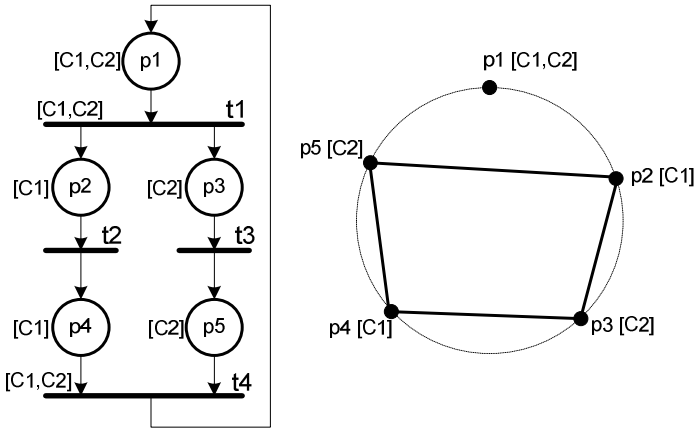
Jako przykład przedstawiony zostanie sposób dekompozycji sieci Petriego, opisującej współbieżny automat cyfrowy (współbieżną maszynę stanów), na podsieci typu automatowego z wykorzystaniem naturalnego wnioskowania Gentzena. Zakłada się, że wyznaczono już znanymi metodami graf znakowań osiągalnych sieci Petriego oraz to, że badana sieć jest żywa i ograniczona.

### *Kolorowanie automatowe sieci Petriego*

Kolorowanie automatowe sieci Petriego polega na przypisaniu miejscom sieci koloru w taki sposób, żeby:

- każde miejsce miało co najmniej jeden kolor;
- kolorowanie spełniało warunki opisane w pracach [AdKa05][BiAd94] [WęWo97].

W praktyce takie kolorowanie jest równoznaczne z pokryciem bezpiecznej (1-ograniczonej) sieci Petriego podsieciami typu automatowego. Tradycyjna metoda wyznaczania takiego pokrycia opiera się na określaniu inwariantów sieci [Mura89]. Inny sposób polega na kolorowaniu, w sensie matematycznym, grafu współbieżności miejsc (Rys. 6-2) [Adam91][BaKu93]. Zazwyczaj otrzymuje się nadmiarowe pokrycie sieci, co pozwala wykryć minimalną liczbę podsieci pokrywających całą sieć. Minimalna liczba podsieci jest równa maksymalnej liczbie numerów wierzchołków równocześnie oznakowanych.




***Rys. 6-2 Przykład sieci pokolorowanej i jej graf współbieżności miejsc***

Sposób wykorzystania kolorowanych sieci Petriego w projektowaniu sterowników można spotkać między innymi w pracach [BiAd94][WęWo97][PaCo01]. Modelowanie i symulacje takich sieci w środowisku CPN Jensena opisano w pracy [Węgr03].

### *Proponowany sposób kolorowania automatowego sieci metodą symboliczną*

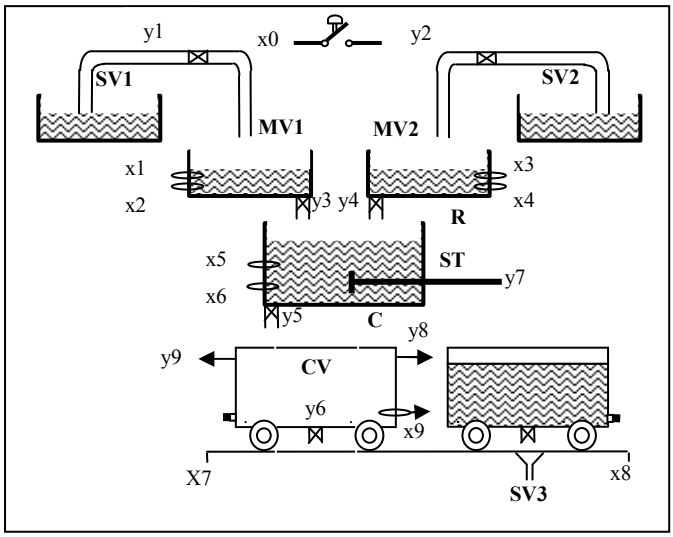
W pracy ograniczono się tylko do jednego sposobu analizy sieci. Wykorzystuje się tutaj twierdzenie zaczerpnięte z pracy [Gall86], z którego wynika, że normalizując sekwent Gentzena zawierający po lewej stronie formuły rachunku zdań, oddzielone przecinkiem (koniunkcję wyrażeń logicznych) można otrzymać równoważną formę w postaci dysjunkcyjnej.

Metoda polega na zwartym przedstawieniu relacji współbieżności między miejscami (lub makromiejscami) w sieci Petriego z wykorzystaniem równania charakterystycznego w logice zdań. Równanie charakterystyczne odpowiada zredukowanej liście sąsiedztwa, uzyskanej na przykład z grafu znakowań osiągalnych sieci Petriego. Po normalizacji sekwentów (pozbyciu się spójników logicznych) otrzymuje się sekwenty-klauzule. Nazwy miejsc niewystępujących w rozpatrywanej klauzuli opisują jedną z składowych automatowych sieci Petriego. Warto zauważyć, że w odróżnieniu od metod symbolicznych

znanych z literatury nie jest konieczne pełne przekształcenie skomplikowanego wyrażenia logicznego do postaci dysjunkcyjnej, tak jak w pracy Deo [Deo80] (rozdz. 6.1), aby uzyskać tylko jedną ze składowych kolorowanych podsieci. Każdy z kolorów definiuje jedną z podsieci typu automatowego. Inny wariant opiera się na analizie sekwentów, opisujących bezpośrednio relację równoległości między miejscami zawartymi w poszczególnych wierzchołkach grafu znakowań.

***Przykład 6-2 Kolorowanie automatowe sieci Petriego opisującej funkcjonowanie sterownika logicznego***

Jako przykład posłuży znany z literatury [Adam91] system sterowania procesem mieszania i transportu cieczy (Rys. 6-3). Przykład ten w kontekście przygotowania programów dla sterowników logicznych omówiono szczegółowo w pracach [AdCh00] [WęWo97][AdWę07].



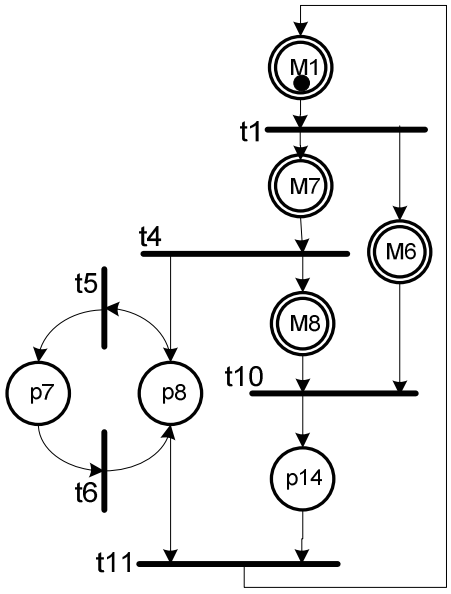

***Rys. 6-3 System mieszania cieczy***

System rozpoczyna działanie po naciśnięciu przycisku *x0*, co powoduje otwarcie zaworów *y1* i *y2*. Napełnione zostają zbiorniki *MV1* i *MV2* do poziomu kontrolowanego przez czujniki *x1* i *x3*. Naciśnięcie przycisku startującego proces powoduje również przemieszczenie wózka *C* ze zbiornikiem *CV* do pozycji początkowej, co sygnalizowane zostanie sygnałem *x7*. Po równoczesnym zamknięciu zaworów *y1* i *y2*, zostają otwarte zawory *y3* i *y4* i reaktor *R* zostaje napełniony cieczą z dwóch wcześniej napełnionych zbiorników *MV1* i *MV2* (sygnały *x2* i *x4*).

Gdy zbiornik reaktora zostanie napełniony do poziomu czujnika *x5*, uruchomione zostanie mieszadło *ST*. Opróżnienie zbiorników *MV1* i *MV2* powoduje zamknięcie zaworów *y3* i *y4*. Faza napełniania zbiornika *CV* nastąpi w momencie, gdy wózek *C* znajdzie się w lewym skrajnym położeniu, a zawór *y5* zostanie otwarty. Sygnał x6 informuje o opróżnieniu reaktora. Ciecz zostaje przetransportowana do zbiornika *SV3*

wózkiem *C*. Cały cykl zamyka się w stanie początkowym, a system czeka na ponowne naciśnięcie przycisku *x0*.

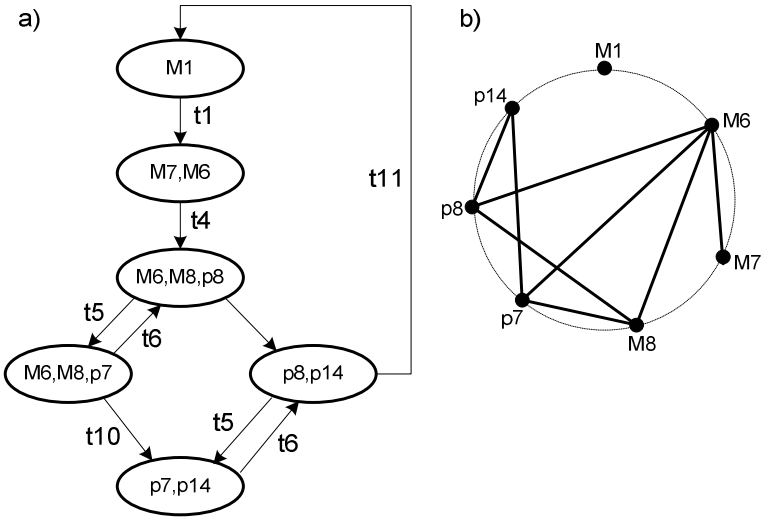
Ze względu na dużą objętościowo specyfikację behawioralną systemu, zbadana zostanie sieć w postaci zredukowanej – makrosieć (Rys. 6-4). W zawartej sieci celowo wyeksponowano w sposób szczegółowy funkcjonowanie mieszadła (miejsca *p7,p8*).




***Rys. 6-4 Sieć Petriego dla systemu mieszania cieczy***

Uproszczenie sieci zostało wykonane zgodnie z opisywanymi w literaturze algorytmami [BiAd94][SuSz99][Węgr03], co powoduje zachowanie jej pierwotnych własności takich jak żywotność, czy ograniczoność [Mura89]. Dzięki redukcji zmniejszają się rozmiary sekwentów oraz ich liczba, co ma znaczący wpływ na efektywność obliczeń analizowanych wyrażeń i powoduje znaczne skrócenie procesu normalizacji.

Zgodnie ze znanymi z literatury metodami wyznaczono graf znakowań (Rys. 6-5a) dla analizowanej makro sieci.




***Rys. 6-5 a) Graf znakowań osiągalnych; b) Graf relacji współbieżności***

Wybierając kolejne miejsca o najwyższym stopniu incydencji budowana jest zredukowana lista sąsiedztwa oraz odpowiadające im wyrażenia logiczne (Tab. 6-5). W odróżnieniu od pełnej listy sąsiedztwa [Deo80] uwzględniono każdą z krawędzi grafu sąsiedztwa (Rys. 6-5b) tylko jednokrotnie. Proponowane uporządkowanie jest zgodne z podstawami heurystycznych metod matematycznego kolorowania grafów niekierowanych [Odmi02].

***Tab. 6-5 Sekwent dla grafu znakowań***

| Zredukowana lista sąsiedztwa | Wyrażenia logiczne |
|---|---|
| M6: M7,M8,p7,p8 | M6+(M7*M8*p7*p8) |
| M8: p8,p7 | M8+(p8*p7) |
| p14: p8,p7 | p14+(p8*p7) |

Iloczyn wyrażeń logicznych dla poszczególnych elementów z listy sąsiedztwa tworzy równanie charakterystyczne (Tab. 6-6), które należy poddać normalizacji.

***Tab. 6-6 Sekwent opisujący listę sąsiedztwa***

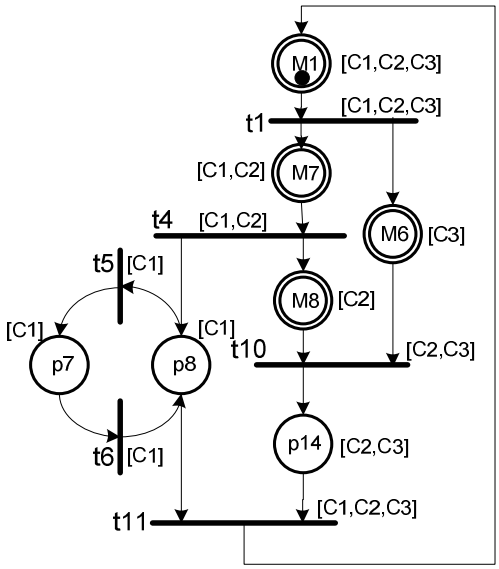
| Sekwent dla listy sąsiedztwa |
|---|
| (M6+(M7*M8*p7*p8)),(M8+(p8*p7)),(p14+(p8*p7))\|-; |

W wyniku normalizacji sekwentu przedstawionego w (Tab. 6-6), otrzymano sekwenty znormalizowane, których dopełnienie wyznacza podsieci automatowe (Tab. 6-7), a liczba sekwentów odpowiada minimalnej liczbie kolorów potrzebnej do pokrycia sieci. Drzewo dowodu składa się z 20 węzłów i realizowane jest w czasie 0,01s. W przypadku, gdy liczba podsieci przekraczałaby trzy (maksymalna liczba makromiejsc i miejsc współbieżnych), to do wyznaczania najkorzystniejszego pokrycia proponuje się wykorzystanie procedury Petricka, w wersji sekwentowej.

***Tab. 6-7 Wynik procesu normalizacji***

| Sekwenty znormalizowane | Dopełnienia | Kolory |
|---|---|---|
| M6,M8,p14\|-; | M1,M7,p7,p8 | C1 |
| M6,p8,p7\|-; | M1,M7,M8,p14 | C2 |
| M7,M8,p7,p8\|-; | M1,M6,p14 | C3 |

Poniższy rysunek (Rys. 6-6) przedstawia badaną sieć, która została pokolorowana przy pomocy wyznaczonych wyrażeń logicznych. Przez *{C1,C2,C3}* oznaczone są możliwe kolorowania poszczególnych miejsc.




***Rys. 6-6 Pokolorowana sieć Petriego dla systemu mieszania cieczy***

W rozpatrywanym konkretnym przypadku sieć Petriego została dekomponowana na trzy podsieci typu swobodnego wyboru [Mura89][Szpy08].

Należy tutaj zwrócić uwagę również na to, że przykład opisuje układ sterowania binarnego za pomocą makrosieci, wiec przy rozwinięciu jej do sieci pełnej liczba kolorów może wzrosnąć.

Algorytm wnioskowania symbolicznego Gentzena doskonale nadaje się analizy symbolicznej sieci Petriego, jak również do wyznaczania składowych automatowych. Warto tutaj zauważyć, że w rachunku sekwentów, w odróżnieniu od metod algebraicznych, każdy następny sekwent jest mniej złożony od wejściowego, a pierwsze rozwiązanie uzyskuje się bez konieczności przekształcania całej formuły. Dzięki temu możliwe jest analizowanie wyrażeń o dużej złożoności. Dodatkowo do algorytmu można wprowadzać heurystyki polegające na szeregowaniu formuł, czy też sortowaniu zmiennych, co może znacznie przyśpieszyć wyznaczanie poszukiwanych rozwiązań (np. składowych automatowych).

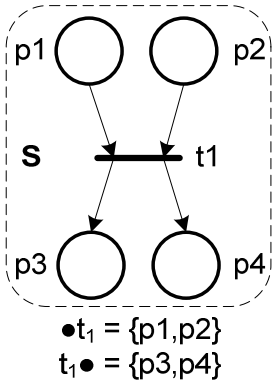
## 6.3. Badanie żywotności sieci Petriego

Podstawy symbolicznej analizy sieci Petriego pod kątem wykrywania blokad i pułapek omówiono w pracach [Mura89][BaMi92]. Szczegółowy przegląd bardziej zaawansowanych metod oraz oryginalne propozycje sposobów wyznaczania blokad i pułapek zawarto w pracach [Kara07] oraz [Węgr03].

Na podstawie definicji blokady dla fragmentu sieci przedstawionego na rysunku (Rys. 6-7) otrzymywane jest:

$$jeżeli\ p3 \in S,\ to\ p1 \in S\ lub\ p2 \in S\ oraz \tag{6-1}$$

$$jeżeli\ p4 \in S,\ to\ p1 \in S\ lub\ p2 \in S \tag{6-2}$$




***Rys. 6-7 Wycinek sieci Petriego***

Przedstawiając $p_i$ za pomocą zmiennej logicznej $x_i$ i stosując własność (6-3):

$$a+b->c \equiv (a->c) \wedge (b->c) \tag{6-3}$$

zależności (6-1) i (6-2) można zapisać w postaci sekwentu (6-4):

$$|-(x_3+x_4)->(x_1+x_2); \tag{6-4}$$

Po lewej stronie znaku implikacji znajduje się suma wszystkich zmiennych wyjściowych tranzycji *t*, natomiast po prawej stronie suma wszystkich zmiennych wejściowych (6-5):

$$|-\sum_{pi \in t\bullet} x_i -> \sum_{pj \in \bullet t} x_j; \tag{6-5}$$

Dla całej sieci wyrażenie opisujące wszystkie tranzycje przyjmie postać (6-6):

$$|-\prod_{t \in T} (\sum_{pi \in t\bullet} x_i -> \sum_{pj \in \bullet t} x_j); \tag{6-6}$$

Wszystkie wektory odpowiadające rozwiązaniom powyższego wyrażenia (6-6) odpowiadają blokadom w sieci Petriego.

W przypadku pułapek, na podstawie definicji pułapki, dla fragmentu sieci przedstawionego na rysunku (Rys. 6-7) otrzymywane jest:

$$jeżeli\ p_1 \in Q,\ to\ p_3 \in Q\ lub\ p_4 \in Q\ oraz \tag{6-7}$$

$$jeżeli\ p_2 \in Q,\ to\ p_3 \in Q\ lub\ p_4 \in Q \tag{6-8}$$

Przedstawiając $p_i$ za pomocą zmiennej logicznej $x_i$ i stosując własność (6-9):

$$a+b->c \equiv (a->c) \wedge (b->c) \tag{6-9}$$

zależności (6-7) i (6-8) można zapisać w postaci sekwentu (6-10):

$$|-(x_1+x_2)->(x_3+x_4); \tag{6-10}$$

Po lewej stronie znaku implikacji znajduje się suma wszystkich zmiennych wejściowych tranzycji $t$, natomiast po prawej stronie suma wszystkich zmiennych wyjściowych (6-11):

$$|-\sum_{pi \in {}^\bullet t} x_i -> \sum_{pj \in t^\bullet} x_j; \tag{6-11}$$

Dla całej sieci wyrażenie opisujące wszystkie tranzycje przyjmie postać:

$$|-\prod_{t \in T} (\sum_{pi \in {}^\bullet t} x_i -> \sum_{pj \in t^\bullet} x_j); \tag{6-12}$$

Wszystkie wektory odpowiadające rozwiązaniom powyższego wyrażenia (6-12) odpowiadają pułapkom w sieci Petriego.

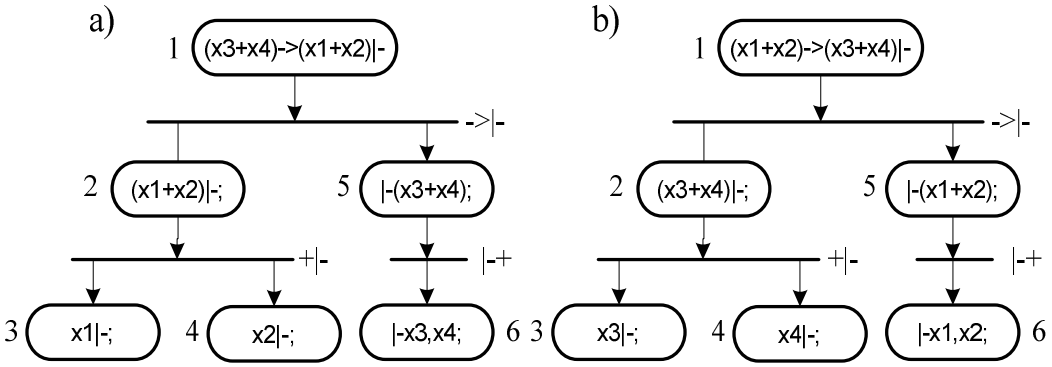
Struktura topologiczna sieci Petriego może być przedstawiona w postaci różnych wyrażeń logicznych, w tym sekwentów [Adam90]. Innym sposobem przedstawionym w literaturze jest opisanie pewnych własności strukturalnych sieci w postaci formuł Horna [Węgr03][AdKa05]. Zastosowanie efektywnych algorytmów Thelena-Mathony'ego [Thel88][Math90] do analizy tych własności zostało zaproponowane w pracy [Węgr03]. Właściwości systemu Gentzena umożliwiają analizę złożonych wyrażeń opisujących blokady i pułapki, bez konieczności dodatkowej transformacji ich do postaci formuł Horna.

Aby poddać normalizacji Gentzena wyrażenia opisujące blokady i pułapki, należy zapisać je po lewej stronie znaku wynikania logicznego. Po superpozycji lewych stron otrzymanych sekwentów otrzymuje się wyrażenie logiczne w postaci dysjunkcyjnej.

$$(x_3+x_4)->(x_1+x_2)|-; \tag{6-13}$$

$$(x_1+x_2)->(x_3+x_4)|-; \tag{6-14}$$

W sekwentach opisujących blokady i pułapki występują tylko trzy rodzaje spójników logicznych (dysjunkcja „+”, koniunkcja „*”, implikacja „->”), stąd wystarczy zastosować tylko trzy reguły wnioskowania. Wskazane jest użycie dodatkowych reguł wnioskowania, rozszerzających standardową logikę Gentzena, w celu doprowadzenia zespołu sekwentów do postaci minimalnej. Na Rys. 6-8 przedstawiono graficzne reprezentacje normalizacji sekwentów (6-13) oraz (6-14) według reguł Gentzena, opisujących blokady i pułapki dla przykładowego fragmentu sieci (rys. 1). Zakłada się, że wyrażenia opisujące blokady lub pułapki są fałszywe, a sieć nie zawiera blokad (rys. 2a) lub pułapek (rys. 2b). Proces normalizacji będzie dowodem postawionych założeń.




***Rys. 6-8 Przykład normalizacji sekwentów blokad i pułapek***

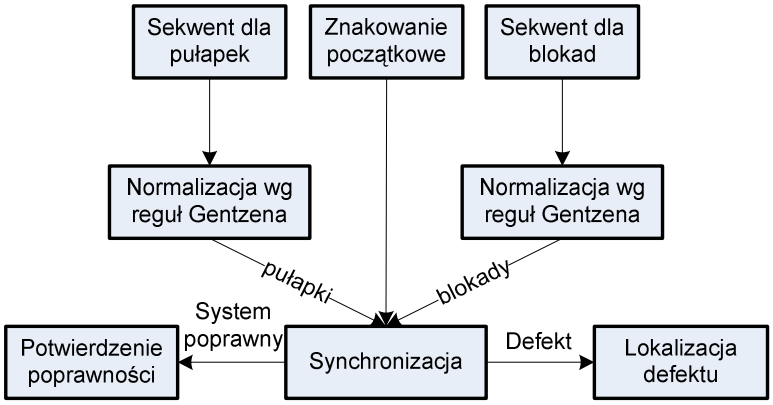
Sekwenty wynikowe będą stanowiły kontrprzykłady zdefiniowanych założeń. Blokady i pułapki w znormalizowanych sekwentach znajdują się po lewej stronie znaku wynikania logicznego (blokady: *x1* i *x2*; pułapki: *x3* i *x4*). Pułapki i blokady minimalne otrzymuje się na podstawie afirmacji zmiennych. Prawe strony sekwentów pomija się w dalszym procesie analizy.

### *Badanie żywotności sieci Petriego*

Żywotność sieci można określić jako możliwość utrzymania żywego znakowania, czyli dla każdej tranzycji można z każdego znakowania osiągalnego dojść do takiego znakowania, w którym ta tranzycja będzie mogła zostać zrealizowana.

Algorytm wyznaczania wszystkich blokad i pułapek nadaje się do sprawdzania żywotności sieci Petriego [Mura89][SuSz99]. Aby stwierdzić, czy dana sieć jest żywa, bada się zależności między blokadami i pułapkami. Po wyznaczeniu blokad i pułapek, należy sprawdzić zgodnie z własnością Commonera [Comm72], czy każda wyznaczona blokada zawiera oznakowaną pułapkę. W przypadku pułapek należy sprawdzić, czy każda wyznaczona pułapka zawiera oznakowaną blokadę.

Zastosowanie algorytmu Gentzena automatyzuje proces badania żywotności. Możliwe jest połączenie procesu wyznaczania blokad i pułapek z jednoczesnym badaniem zależności pomiędzy nimi.



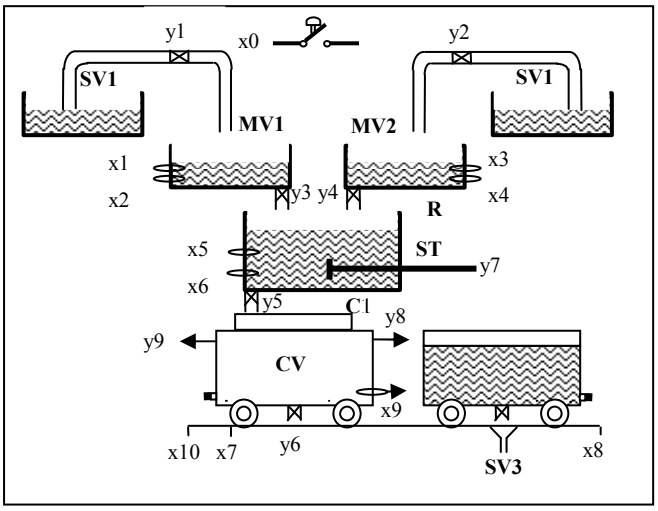

***Rys. 6-9 Schemat badania żywotności sieci***

Pełna automatyzacja badania żywotności możliwa jest przez wprowadzenie niewielkich zmian do algorytmu Gentzena. Sekwenty blokad jak i pułapek można normalizować współbieżnie (Rys. 6-9). Otrzymywane z procesów normalizacji sekwenty blokad i pułapek trafiają do procesu synchronizacji, który odrzuca prawe strony sekwentów, sprowadza je do postaci minimalnej poprzez odrzucenie rozwiązań nadmiarowych, sprawdza pokrycie blokad przez pułapki i pułapek przez blokady oraz weryfikuje znakowania początkowe. Pokryte wzajemnie pułapki i blokady zostają odrzucone. Jeżeli z całego procesu pozostaną blokady lub pułapki, to będą one wskazywały dokładnie miejsce defektu sieci Petriego opisującej projektowany system. Brak rozwiązań informuje o poprawnie zaprojektowanym systemie sterowania, w którym zastoje nie występują.

***Przykład 6-3 Analiza binarnego systemu sterowania***

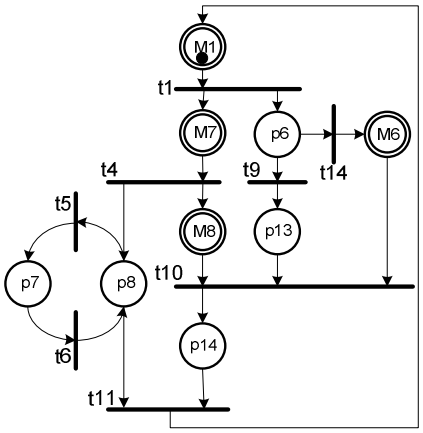
Jako przykład posłuży znany z literatury [Adam90] system sterowania procesem mieszania i transportu cieczy. W celu uwypuklenia problemu diagnostyki sieci zostanie przedstawiony zmodyfikowany system sterowania z błędnie zaprojektowaną siecią (Rys. 6-10), w której należy zlokalizować miejsca defektów [Węgr03].

Szczegółowe funkcjonowanie układu opisano w rozdziale 6.2 (Przykład 6-2). Celowa modyfikacja funkcjonowania systemu sterowania polega na tym, że wózek musi niekiedy przejechać poza pozycję *x7*, do pozycji *x10*.




***Rys. 6-10 Zmodyfikowany system mieszania cieczy***

Ze względu na dużą objętościowo specyfikację systemu, zbadana zostanie sieć w postaci zredukowanej - makrosieć (Rys. 6-11). Uproszczenie sieci zostało wykonane zgodnie z opisywanymi w literaturze algorytmami [SuSz99], co powoduje zachowanie jej pierwotnych własności takich jak żywotność, czy ograniczoność [Mura89]. Dzięki redukcji zmniejsza się rozmiary sekwentów, opisujących blokady i pułapki. Ma to znaczący wpływ na efektywność obliczeń analizowanych wyrażeń i powoduje znaczne skrócenie procesu normalizacji.




***Rys. 6-11 Sieć Petriego dla zmodyfikowanego systemu mieszania cieczy***

Stosując zmodyfikowany wzór (6-6), dla analizowanej sieci Petriego wyznaczono sekwent reprezentujący blokady (Tab. 6-8).

***Tab. 6-8 Sekwent dla blokad***

| **Sekwent dla blokad** |
|---|
| `(((M7+p6)->M1) * ((p8+M8)->M7) * (p13->p6) * (M6->p6)` |
| `* (p14->(M8+p13+M6)) * (p7->p8) * (p8->p7)` |
| `* (M1->(p8+p14)))|-;` |

W podobny sposób określa się sekwent reprezentujący pułapki (Tab. 6-9).

***Tab. 6-9 Sekwent dla pułapek***

| Sekwent dla pułapek |
|---|
| (((M1->(M7+p6)) * (M7->(p8+M8)) * (p6->p13) * (p6->M6) * ((M8+p13+M6)->p14) * (p8->p7) * (p7->p8) * ((p8+p14)->M1)))\|-; |

W wyniku normalizacji sekwentów otrzymano minimalne blokady i pułapki (Tab. 6-10). Wcześniej usunięte zostały zbyteczne, prawe strony sekwentów.

***Tab. 6-10 Minimalne blokady i pułapki***

| Blokady | Pułapki |
|---|---|
| M1,M7,M8,p14\|-; | M7,M8,p14,M1\|-; |
| M1,M7,p8,p7\|-; | M7,p8,p7,M1\|-; |
| M1,p6,p13,p14\|-; | p6,p13,M6,p14,M1\|-; |
| M1,p6,M6,p14\|-; | - |

W tabeli 3 zaznaczono dwie blokady (*M1,p6,p13,p14* i *M1,p6,M6,p14*), które nie są pokryte przez minimalne oznakowane w znakowaniu początkowym pułapki. Wskazują one dokładnie miejsce zastojów w projektowanym systemie sterowania.

# 7. Opis eksperymentalnego systemu wnioskowania „GENTZEN v.6”

W latach 1990-1991 w Wyższej Szkole Inżynierskiej w Zielonej Górze (obecnie Uniwersytet Zielonogórski) miały miejsce pierwsze próby implementacji algorytmu Gentzena z wykorzystaniem języka PROLOG. Duża złożoność obliczeń, brak prób optymalizacji algorytmu oraz wydajność ówczesnych komputerów nie dały satysfakcjonujących wyników, przez co prace nad badaniem i wykorzystaniem algorytmu zostały zarzucone [Adam90][Szaj96]. W roku 2000 w ramach pracy magisterskiej autora powstała kolejna implementacja algorytmu wnioskującego. Zastosowanie dedykowanego rozwiązania (bez warstwy pośredniej), dla konkretnego systemu operacyjnego, w tym przypadku MS Windows i jego implementacja w języku C++ [Stro94] pozwoliła na wyznaczenie rezultatów, których we wcześniejszych implementacjach w ogóle nie udawało się uzyskać. Przedstawiona w rozprawie wersja opiera się na podobnej koncepcji, która została znacznie zmodyfikowana i uzupełniona o rozwiązania opisane w rozdziałach 3.4 i 4.

## 7.1. Założenia projektowe

Podczas projektowania nowej aplikacji wnioskującej przyjęto następujące złożenia:

a) Opracowanie eksperymentalnego środowiska graficznego do normalizacji wyrażeń według reguł Gentzena, z możliwością generowania historii przebiegu wnioskowania.

b) Wprowadzenie do systemu Gentzena rozbudowanego nazewnictwa formuł logicznych. W literaturze [Pawl65] można znaleźć opisy słowne algorytmów (np. wyszukiwania głównego spójnika) bazujące na przestrzeni nazw formuł logicznych składających się z pojedynczych znaków. Przyjęto, że należy unowocześnić opisywane algorytmy, wzorując się na nowoczesnych językach programowania, co umożliwi sensowne nazywanie formuł oraz znacznie zwiększy przestrzeń ich generowania.

c) Zaproponowanie własnego, sprawnego algorytmu wyszukiwania głównego spójnika w zbiorach formuł logicznych. Rozwiązanie takie umożliwia wprowadzenie rozbudowanego nazewnictwa formuł logicznych, nadawanie priorytetów operatorom logicznym i pozwala na pewne uproszczenia semantyczne w definiowaniu formuł, przez co ułatwia użytkownikowi pracę z programem.

d) Opracowanie analizatora składni wprowadzanych wyrażeń. Projekt i implementacja prostego analizatora semantycznego i syntaktycznego weryfikującego poprawność wprowadzanych sekwentów, a także elementów dodatkowych (np. definicji typów formuł).

e) Wprowadzenie rozszerzeń dla podstawowego algorytmu Gentzena ułatwiających jego zastosowanie w technice cyfrowej. Możliwość definiowania typu formuł (Wejściowe/Wyjściowe).

f) Optymalizacja algorytmu, poprzez wprowadzenie reguł specyficznych bazujących na znanych z literatury algorytmach (Rezolucja, Thelen-Mathony).

g) Wprowadzenie możliwości transformacji wyrażeń wynikowych do różnych postaci bazujących na zdefiniowanych typach formuł.

h) Współpraca z innymi programami – w szczególności profesjonalnymi lub uniwersyteckimi programami CAD przeznaczonymi do projektowania układów cyfrowych.

## 7.2. Ogólny opis systemu

Pierwszym krokiem w procesie implementacji był wybór technologii. Nastawiając się głównie na implementację logiki algorytmu jako dodatkowe założenie przyjęto, że należy zredukować nakład pracy związany z budowaniem interfejsu obsługi w aplikacji. Dodatkowo poszukiwano technologii takich, w których występują zaawansowane funkcje przetwarzania danych tekstowych, mogących znacznie ułatwić implementację algorytmu wnioskowania symbolicznego. Wybrane zostało środowisko Borland C++ Builder [Reis03] ze względu na łatwość budowania interfejsów graficznych oraz dostępność zaawansowanych obiektów i metod obsługujących dane tekstowe. Aplikacje opracowane w tym środowisku są dedykowane dla systemu operacyjnego, w tym przypadku MS Windows, dzięki czemu efektywnie wykorzystują moc obliczeniową maszyny, na której zostaną uruchomione.

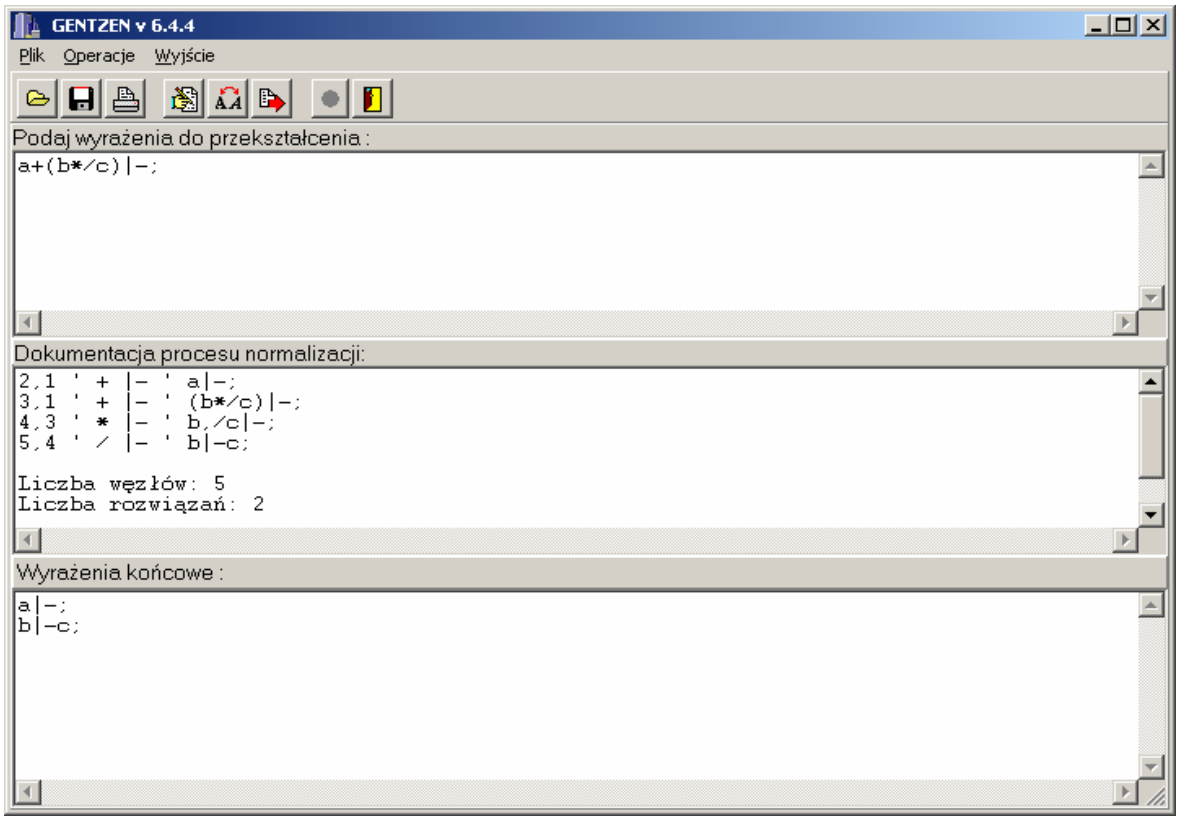
W wyniku implementacji oprogramowania, na podstawie przyjętych założeń, powstał eksperymentalny system wnioskujący według reguł Gentzena [Gent69][Gent80] (rozdz. 3.2). Prosty i zarazem intuicyjny interfejs programu umożliwia łatwą jego obsługę, a analizator składni czuwa nad poprawnością semantyczną oraz syntaktyczną wprowadzanych wyrażeń jak i elementów rozszerzeń. Dane do programu można odczytywać z pliku lub wprowadzać ręcznie, przy pomocy specjalnie do tego przygotowanego edytora. Dane wynikowe, dokumentację z przebiegu normalizacji, czy też

dane po transformacjach można zapisać w plikach tekstowych lub wydrukować. Istnieje także możliwość przetransformowania wyrażeń znormalizowanych do formatu ESPRESSO i wykonania ewentualnej, dodatkowej minimalizacji z wykorzystaniem zewnętrznego programu do minimalizacji funkcji boolowskich.

Ze względu na dużą ilość przetwarzanych danych jak i czas ich przetwarzania system wnioskujący dokonuje normalizacji wyrażeń, a także transformacji danych w niezależnych procesach systemu operacyjnego. System operacyjny sam przydziela odpowiednią ilość jednostek czasu poszczególnym procesom na wykonanie przydzielonych im zadań. Dzięki takiemu rozwiązaniu aplikacja nie odbiera całych zasobów systemowi operacyjnemu. Nawet przy długich i skomplikowanych obliczeniach można przerwać proces, a także korzystać z systemu operacyjnego przy realizacji innych zadań.

## 7.3. Interfejs użytkownika

Jednym z głównych założeń podczas projektowania systemu wnioskującego było opracowanie prostego i intuicyjnego interfejsu graficznego tak, aby niedoświadczony użytkownik również poradził sobie z jego obsługą.




***Rys. 7-1 Interfejs programu***

Wzorując się na interfejsach produktów opracowanych przez firmę Microsoft, które są uznawane za jedne z najlepszych na świecie, zaproponowano autorski interfejs (Rys. 7-1) do przetwarzania wyrażeń logicznych według reguł Gentzena.

Podstawowe elementy interfejsu użytkownika, które dostępne są w chwili uruchomienia programu można podzielić dwie zasadnicze części:

**a) część edycyjna** - gdzie użytkownik wprowadza wyrażenia logiczne w postaci sekwentów, które mają być poddane normalizacji. Umożliwia przeglądanie wyników, śledzenie dokumentacji z procesu normalizacji oraz analizę potencjalnych błędów składniowych wprowadzanych wyrażeń. Nad częścią edycyjną czuwa prosty analizator semantyczny i syntaktyczny. Do systemu wprowadza się sekwenty oddzielone średnikiem. Znakiem wynikania logicznego, charakterystycznym dla sekwentów jest symbol „|-”. System dopuszcza następujące spójniki logiczne opisane w formie znaków tekstowych: „+” – dysjunkcja, „*” – koniunkcja, „<->” – równoważność, „->” – implikacja, „/” – negacja, „<+>” – alternatywa wyłączająca. System nie wymaga od użytkownika wprowadzania nawiasów grupujących wszystkie formuły, tak jak na przykład jest zalecane w pracy Pawlaka [Paw165] i przyjęte w wielu podręcznikach logiki [ŁaMa04]. Jeżeli w sekwencie brakuje nawiasów, to system wnioskujący sam określa priorytety eliminowanych spójników logicznych według następującej kolejności (równoważność, alternatywa wyłączająca, implikacja, koniunkcja, dysjunkcja, negacja), gdzie równoważność jest spójnikiem o najwyższym priorytecie, a negacja o najniższym. Projektant może zmienić zdefiniowane priorytety spójnikom, odpowiednio wprowadzając do sekwentów nawiasy grupujące. Edytor i analizator składni dopuszczają również komentarze, gdzie komentarzem jest dowolny tekst, czy też sekwent poprzedzony znakiem „#”. Formuły w omawianym systemie wnioskującym mogą przyjmować dowolne nazwy zaczynające się od małych lub dużych znaków łacińskich. W dalszej części formuły mogą występować również cyfry (np. *zm10*). Na poziome edycji sekwentów możliwe jest określenie typu wprowadzanych formuł, rozróżnianych jako wejściowe i wyjściowe. Określenie typu wykonuje się za pomocą słów kluczowych *in* i *out*, gdzie po dwukropku wymienia się listę formuł oddzielonych przecinkami (7-1).

$$\begin{array}{c} in: x1,x2,x3; \\ out: y1,y2; \end{array} \qquad \textbf{(7-1)}$$

Analizator składni, dla celów badawczych, zezwala także na wprowadzanie sekwentów znormalizowanych lub częściowo znormalizowanych.

Tekstowa dokumentacja z procesu normalizacji, generowana automatycznie ma następującą postać (7-2):

$$A,B \text{ ‘ } operacja \text{ ‘ } wynik; \qquad \textbf{(7-2)}$$

gdzie *A* jest kolejnym wykonywanym przekształceniem licząc od sekwentów wprowadzonych do systemu, *B* jest numerem sekwentu rodzica, z którego powstaje, ‘ operacja ’ określa, która reguła eliminacji spójnika została wykonana i wskazuje po której stronie znaku wynikania logicznego dany spójnik się znajdował. Wynik przedstawia sekwent po zastosowaniu reguły eliminacji.

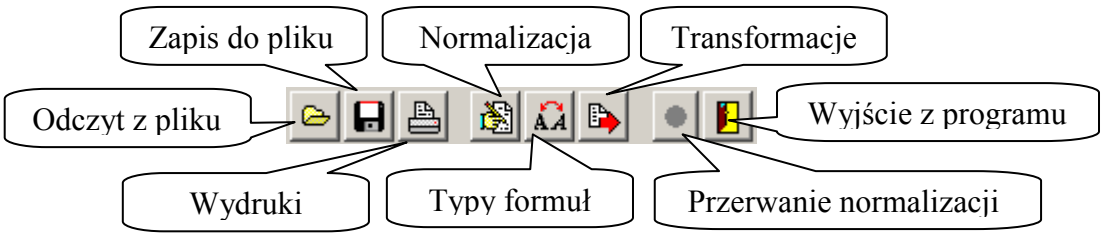
Przykładowa dokumentacja z procesu normalizacji przedstawiona została w Tab. 7-1.

***Tab. 7-1 Przykładowa dokumentacja procesu normalizacji***

| a+(b*/c)\|-; |
|---|
| 2,1 ' + \|- ' a\|-; |
| 3,1 ' + \|- ' (b*/c)\|-; |
| 4,3 ' * \|- ' b,/c\|-; |
| 5,4 ' / \|- ' b\|-c; |
| |
| Liczba węzłów: 5 |
| Liczba rozwiązań: 2 |
| Czas: 00:00:00.032 |

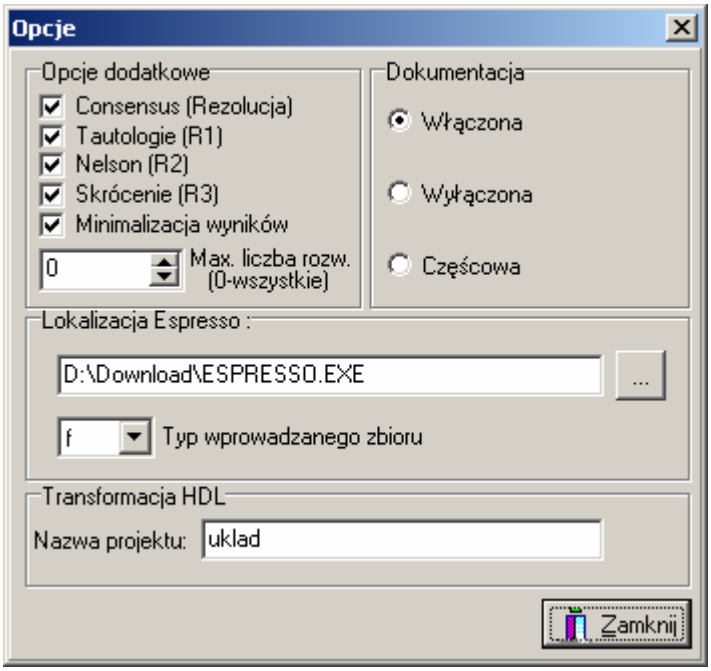
W procesie dokumentacji (Tab. 7-1), w wyniku eliminacji spójnika dysjunkcji w analizowanym sekwencie (po lewej stronie znaku wynikania logicznego) o domyślnym numerze pierwszym, powstały dwa sekwenty (2 i 3), gdzie sekwent (2) jest sekwentem znormalizowanym zapisywanym w zbiorze rozwiązań. Sekwent (3) poddawany będzie dalszemu procesowi normalizacji, gdyż posiada jeszcze spójniki logiczne, a żadna reguła dopuszczalna ani tautologia nie wyklucza go z procesu normalizacji. Sekwent (4) powstaje z sekwentu (3) w wyniku eliminacji spójnika koniunkcji występującego po lewej stronie sekwentu. Sekwent (5) powstaje w wyniku eliminacji spójnika negacji w sekwencie (4), także po lewej stronie sekwentu. Jest on również sekwentem znormalizowanym, który zostanie zapisany w zbiorze rozwiązań. Zawsze do dokumentacji dopisywana jest stopka podsumowująca zawierająca informacje o liczbie węzłów w drzewie dowodu, liczbie sekwentów zapisanych w zbiorze rozwiązań oraz czasie trwania procesu normalizacji. Zdarza się, że czasy analizy tego samego wyrażenia są minimalnie różne, co spowodowane jest oddaniem kontroli przydziału czasu, na realizowane zadanie, systemowi operacyjnemu.

**b) część sterująca** - w skład której wchodzą akcje menu oraz pasek narzędzi (Rys. 7-2) stanowiący tzw. szybki dostęp do najczęściej używanych funkcji programu.




***Rys. 7-2 Pasek narzędzi***

Podstawowe elementy interfejsu użytkownika obsługują przede wszystkim przekształcenia symboliczne zgodnie z regułami Gentzena. Procesem normalizacji steruje się za pomocą parametrów dostępnych z poziomu menu głównego programu (*Operacje->Opcje*). Dialogowe okno konfiguracyjne systemu wnioskującego zostało przedstawione na Rys. 7-3. Konfiguracja parametrów dla procesów normalizacji i transformacji zapisywana jest w pliku *GENTZEN.INI*, tworzonym w tej samej lokalizacji, w której znajduje się plik wykonywalny programu. Zapisanie konfiguracji w pliku usprawnia pracę z programem, gdyż zapisana konfiguracja jest wczytywana przy każdym uruchomieniu programu. Ponadto otwiera drogę dla innych aplikacji, które mogą wykorzystywać system wnioskujący jako podprogram w swoich zadaniach, zezwalając na sterowanie ważniejszymi funkcjami systemu.




***Rys. 7-3 Parametryzowanie programu***

**Lokalizacja ESPRESSO** wskazuje na umiejscowienie w systemie operacyjnym zewnętrznego programu do minimalizacji funkcji boolowskich, z którym aplikacja potrafi współpracować [ŁuJa97]. Typ wprowadzanego zbioru określa informację dla systemu ESPRESSO, w jakiej formie opisane jest rozpatrywane zjawisko dyskretne.

W reprezentacji logicznej funkcji w standardzie berkleyowskim pojawiają się pojęcia zbioru prostego (*ON-set*), zanegowanego (*OFF-set*) oraz nieistotnego (*DC-set*). *ON-set* stanowi zbiór argumentów, dla których funkcja przyjmuje wartość 1, *OFF-set* zbiór argumentów, dla których funkcja przyjmuje wartość 0, a *DC-set* zbiór wartości, dla których funkcja jest nieokreślona. W zależności od przygotowanej specyfikacji znormalizowany wynik z programu wnioskującego może być prezentowany przez jeden, dwa, czy też nawet trzy opisane powyżej zbiory. Aby dostosować postać znormalizowaną do formatu ESPRESSO należy określić, w jakiej kombinacji podawanych zbiorów, określonej w standardzie berkeleyowskim, znajduje się wynikowa postać znormalizowana:

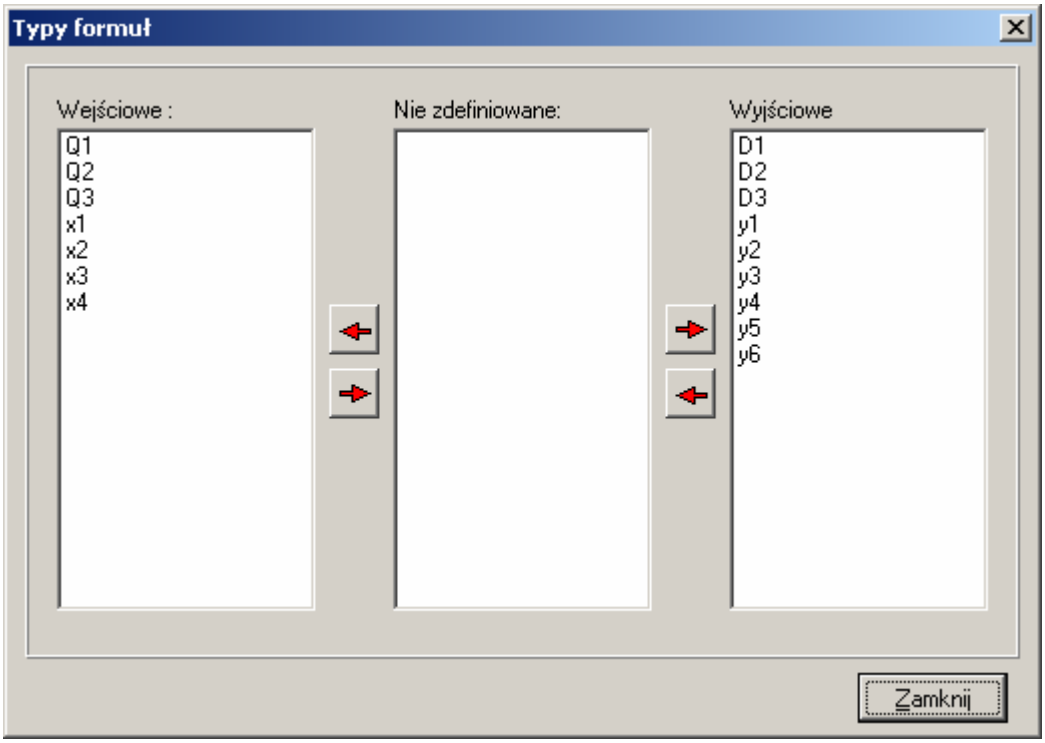
- **f** – zadany jest zbór *ON-set*. *DC-set* jest pusty, a *OFF-set* oblicza się jako uzupełnienie zbioru *ON-set*;
- **r** – zadany jest zbór *OFF-set*. *DC-set* jest pusty, a *ON-set* oblicza się jako uzupełnienie zbioru *OFF-set*;
- **fd** – zadane są zbiory *ON-set* i *DC-set*, przy czym mogą mieć część wspólną. Przyjmuje się wówczas, że część wspólna nie powinna należeć do zbioru *ON-set*. Zbiór *OFF-set* obliczany jest jako uzupełnienie sumy podanych zbiorów;
- **fr** – zadane są zbiory *ON-set* i *OFF-set*, przy czym muszą być one rozłączne. Zbiór *DC-set* obliczany jest jako uzupełnienie sumy podanych zbiorów;
- **dr** – zadane są zbiory *OFF-set* i *DC-set*. Ewentualna część wspólna traktowana jest jako zbiór *DC-set*. Zbiór *ON-set* obliczany jest jako uzupełnienie sumy podanych zbiorów;
- **fdr** – podane są wszystkie trzy zbiory, gdzie *ON-set* i *OFF-set* muszą być rozłączne. Ewentualne części wspólne *DC-set* i *ON-set* oraz *DC-set* i *OFF-set* są traktowane jako zbiór *DC-set*.

**Dokumentacja** definiuje poziom generowanej dokumentacji z procesu normalizacji. Stanowi bardzo ważny parametr, szczególnie przy analizie złożonych wyrażeń, gdzie proces dokumentacji przestaje być pomocny i czytelny dla użytkownika, a jego generowanie zajmuje dużo czasu oraz zasobów sprzętowych. Częściowa dokumentacja generuje tylko informację o czasach uzyskania poszczególnych wyników oraz dodatkowo informuje o każdorazowym osiągnięciu stu tysięcy węzłów w drzewie dowodu, w celu zasygnalizowania poprawnego działania algorytmu.

**Opcje dodatkowe** (Consensus, Tautologie (R1), Nelson (R2), Skrócenie (R3), Minimalizacja wyników) umożliwiają dowolne włączanie rozszerzeń algorytmu Gentzena

mających bezpośredni wpływ na skrócenie drzewa dowodu. Opcje oznakowane R1, R2 i R3 odpowiadają regułom dopuszczalnym wprowadzonym do systemu wnioskującego Gentzena, będące następstwem sekwentowej realizacji metody Thelena. Consensus jest implementacją metody rezolucji. Maksymalna liczba rozwiązań ogranicza poszukiwanie rezultatów do zadanej liczby. W wielu przypadkach wystarczające może być wyznaczenie jednego rozwiązania. Na przykład stosując system wnioskujący do dowodzenia twierdzeń znalezienie jednego kontrprzykładu udowadnia, że twierdzenie jest nieprawdziwe. Podobnie jest w przypadku wyznaczania pokryć funkcji boolowskich, gdzie znalezienie pierwszego lub kilku pierwszych rozwiązań może być w zupełności wystarczające np. do realizacji układu kombinacyjnego.

**Transformacja HDL** służy do wprowadzenia domyślnej, niezbędnej nazwy projektu generowanego w postaci języków opisu sprzętu (VHDL i VERILOG) na podstawie specyfikacji przygotowanej w formie sekwentowej.




***Rys. 7-4 Kreator typów formuł***

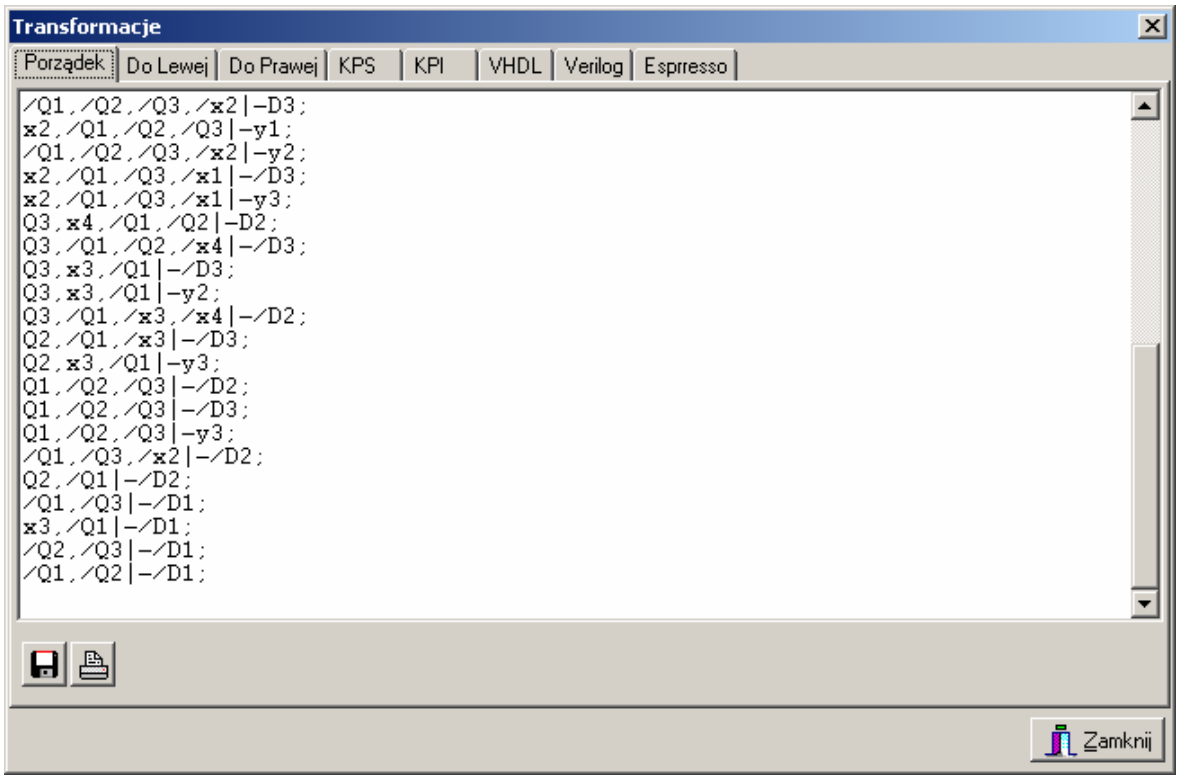
Transformacje znormalizowanego opisu możliwe są do wykonania dopiero w momencie, gdy zdefiniowane zostaną typy wprowadzonych do systemu formuł. Jeżeli definicje typów formuł nie zostaną podane w części edycyjnej, a użytkownik spróbuje wykonać transformację, to uruchomi się okienkowy kreator definicji typów formuł (Rys. 7-4). W kreatorze tym, przy pomocy myszki przenosi się formuły wejściowe na stronę lewą, a formuły wyjściowe na stronę prawą. Dodatkowym atutem jest to, że nawet w przypadku zdefiniowania formuł w części edycyjnej, można przy pomocy kreatora,

dostępnego również z poziomu paska narzędzi, przedefiniować typy wprowadzonych wcześniej formuł. Specyfikacja w części edycyjnej nie zostaje zmieniona, tylko na czas transformacji zmieniane są domyślne typy formuł.

Po prawidłowej definicji formuł i wykonaniu procesu transformacji dla sekwentów znormalizowanych, prezentowane są różne postacie wygenerowanych wyników (Rys. 7-5):

- **Porządek** – porządkuje formuły w sekwentach przenosząc wejściowe na stronę lewą (do poprzednika), a wyjściowe na stronę prawą (do następnika), z ewentualnym wprowadzeniem spójnika negacji.
- **Do Lewej** – wszystkie formuły z następnika przenoszone są w formie zanegowanej do poprzednika.
- **Do Prawej** – wszystkie formuły z poprzednika przenoszone w formie zanegowanej są do następnika.
- **KPS** – wprowadzenie spójników logicznych i połączenie w kanoniczą postać sumy, dla sekwentów opisujących te same formuły wyjściowe.
- **KPI** – wprowadzenie spójników logicznych i połączenie w kanoniczą postać iloczynu, dla sekwentów opisujących te same formuły wyjściowe.
- **VHDL** – transformacja wygenerowanej specyfikacji układu w formie sekwentowej na język VHDL.
- **VERILOG** – transformacja wygenerowanej specyfikacji układu w formie sekwentowej na język VERILOG.
- **ESPRESSO** – transformacja wygenerowanej specyfikacji układu w formie sekwentowej formatu ESPRSSO.

Wszystkie wygenerowane opisy można zapisać w plikach tekstowych lub wydrukować. W przypadku transformacji do formatu ESPRESSO możliwe jest uruchomienie skojarzonego programu do minimalizacji funkcji boolowskich (ESPRESSO). Wyniki dodatkowej minimalizacji można również wydrukować lub zapisać w pliku tekstowym.




***Rys. 7-5 Wyniki procesu transformacji***

## 7.4. Struktura oprogramowania

Struktura oprogramowania zostanie przedstawiona i omówiona z wykorzystaniem diagramów UML w wersji 1.3 [WrMa05][BoRu02][PiPi07]. Diagramy zostały wykonane za pomocą systemu PowerDesigner v11 firmy SYBASE [PowDe]. Wybrane algorytmy zastosowane przy implementacji zostały przedstawione za pomocą diagramów aktywności. Strukturę aplikacji odzwierciedla diagram klas. Do wizualizacji relacji wewnątrz systemu i relacji pomiędzy użytkownikiem a systemem wnioskującym został wykorzystany diagram przypadków użycia. Wyróżnione, poszczególne stany, w jakich aplikacja może się znaleźć, przedstawiono przy użyciu diagramu stanów.

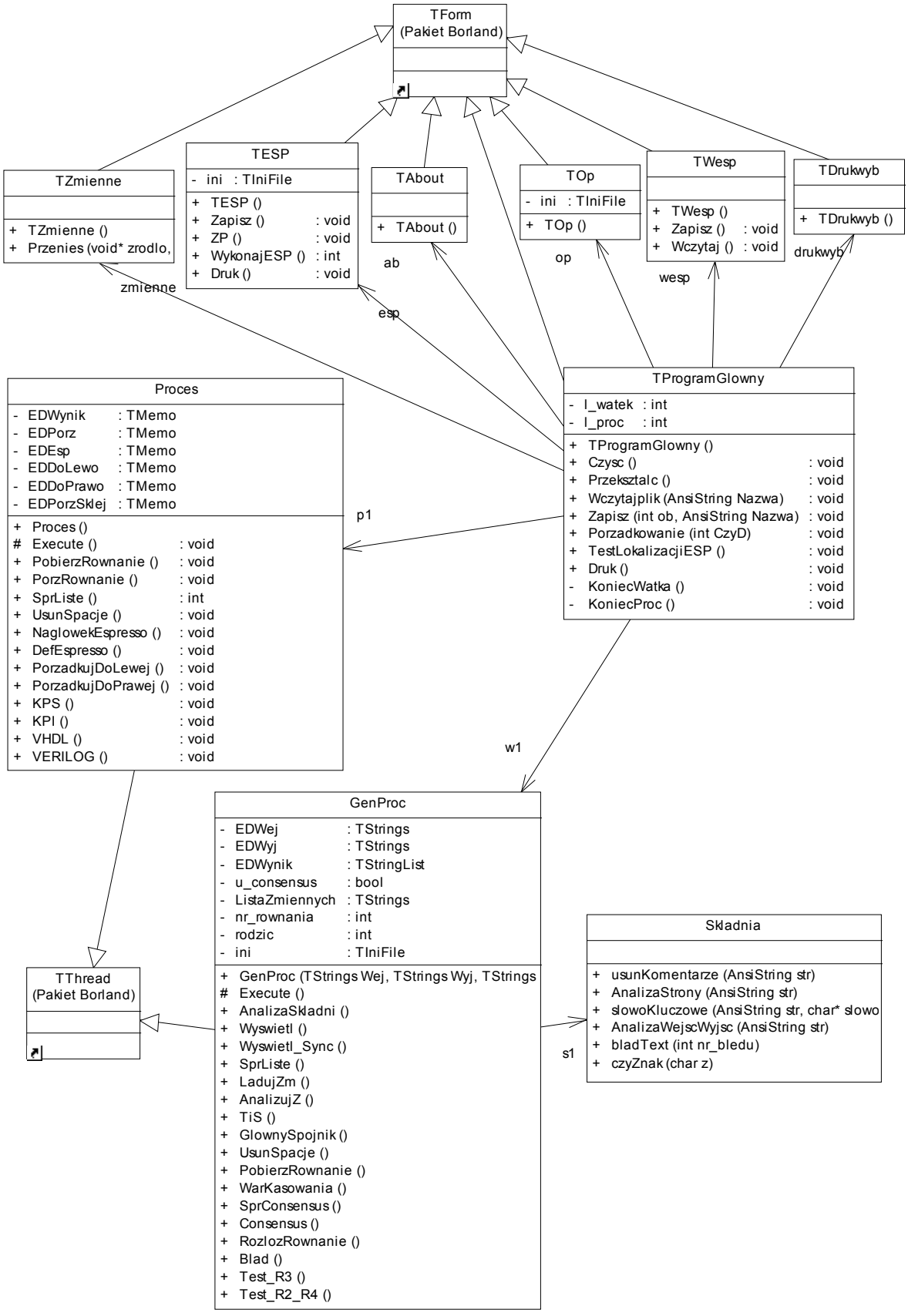
### 7.4.1. Diagram klas

Główną, a zarazem najważniejszą klasą systemu wnioskującego jest klasa *GenProc*. Dziedziczy ona własności po klasie systemowej *TThread* z pakietu Borland odpowiadającej za współbieżne przetwarzanie danych. Następstwem wykorzystania własności klasy *TThread* jest system przetwarzający sekwenty w odseparowanych wątkach systemu operacyjnego, gdzie system sam kontroluje przydział czasu na realizację zadania. Ponadto przy trudnych i długotrwałych obliczeniach system wnioskujący nie powoduje zakleszczeń systemu operacyjnego. Innym ważnym aspektem wykorzystania własności klasy *TThread* jest możliwość zrównoleglenia przetwarzania sekwentów, tak jak to zostało zaproponowane w przypadku modyfikacji systemu wnioskującego dla potrzeb badania

żywotności sieci Petriego (rozdz. 6.3). Uzupełnieniem klasy *GenProc* jest moduł *Skladnia* stanowiący bibliotekę funkcji przeznaczoną do weryfikacji semantycznej i syntaktycznej wprowadzanych do systemu sekwentów oraz definicji typów formuł. Własności i metody klasy *TThread* dziedziczy również klasa *Proces* odpowiedzialna za transformacje opisu znormalizowanego na różne formaty.

Pozostałe klasy dziedziczą własności po klasie *TForm* z pakietu Borland. Odpowiadają one za interfejs graficzny i przeznaczone są do obsługi oraz parametryzowania obiektów klas *GenProc* i *Proces*. Klasa *TProgramGłówny* dziedzicząca po *TForm* inicjuje obiekty wszystkich klas systemu wnioskującego i integruje je. Dodatkowo stanowi główny interfejs dla użytkownika programu (Wprowadzanie i edycja sekwentów, prezentowanie drzewa dowodu oraz sekwentów znormalizowanych). Klasa *TESP* służy do prezentacji wyrażeń znormalizowanych przekształconych na różne postacie (KPS, VHDL, VERILOG, ESPRESO itd.), oraz posiada funkcje do obsługi zewnętrznego programu ESPRESSO, z którego wyniki są przechwytywane przez obiekt klasy *TWEsp* i prezentowane użytkownikowi. Klasa *TZmienne* przeznaczona jest do obsługi kreatora typów formuł, natomiast klasa *TOp* odpowiada za graficzny interfejs obsługi parametrów programu.

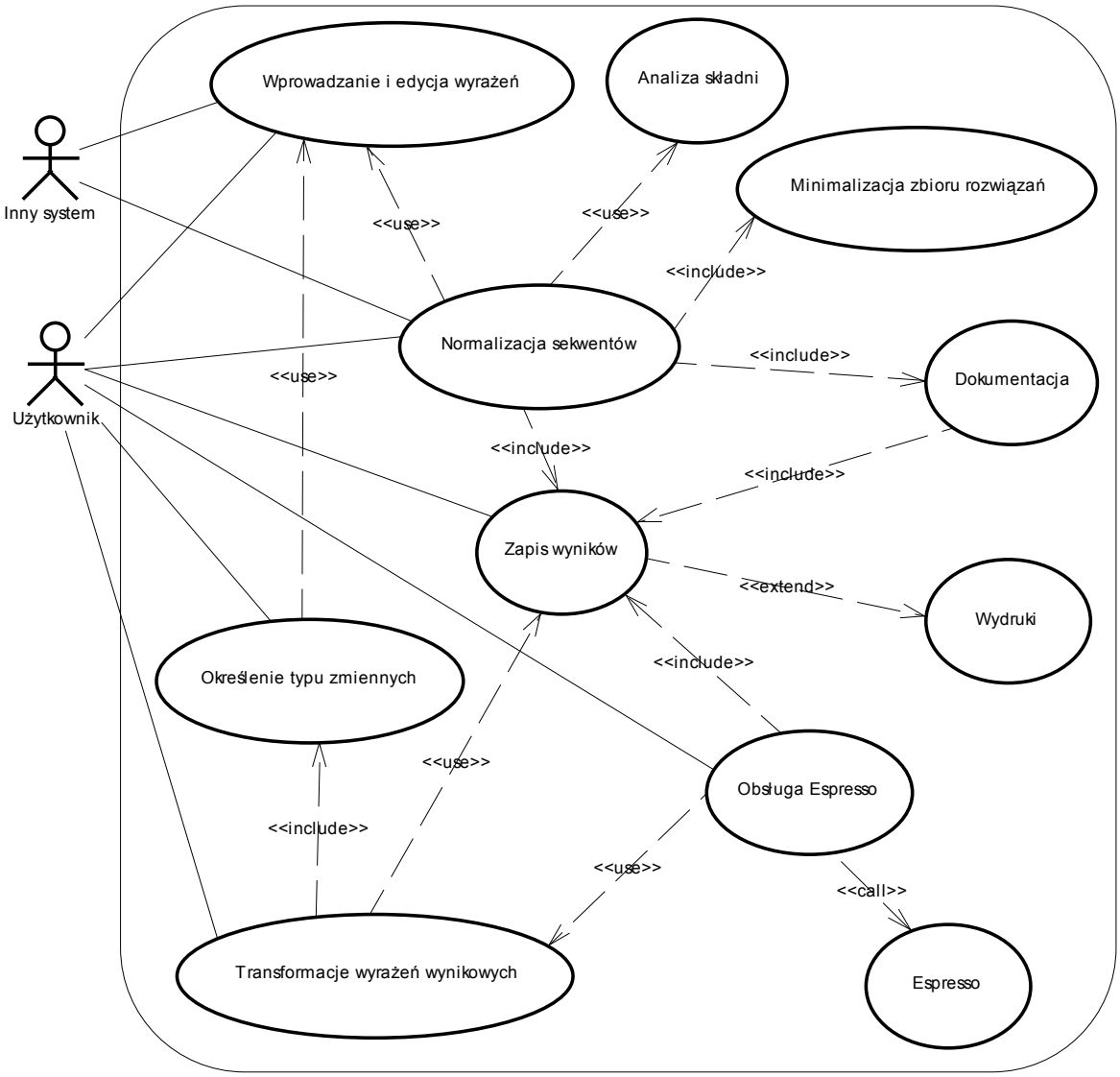
Diagram klas wraz z relacjami został przedstawiony na Rys. 7-6.




***Rys. 7-6 Diagram klas***

### 7.4.2. Diagram przypadków użycia

Na poniższym diagramie przedstawione zostały przypadki użycia systemu wnioskującego Rys. 7-7. W pierwotnej formie przewidzianych było dwóch aktorów systemu – *użytkownik* oraz *inny system* wykorzystujący system wnioskujący jako podprogram do realizacji swoich zadań. Inne systemy mogą wykorzystywać system wnioskujący za sprawą modułowej budowy programu. Stosunkowo proste jest wydzielenie klasy *GenProc* wraz z modułem weryfikacji składni i wykorzystanie go w innych programach. Przykładem będzie tutaj praca inżynierska zrealizowana na Uniwersytecie Zielonogórskim wykorzystująca system wnioskujący do minimalizacji binarnych tablic decyzyjnych [Lige03].




***Rys. 7-7 Diagram przypadków użycia***

### 7.4.3. Diagramy aktywności dla systemu wnioskującego

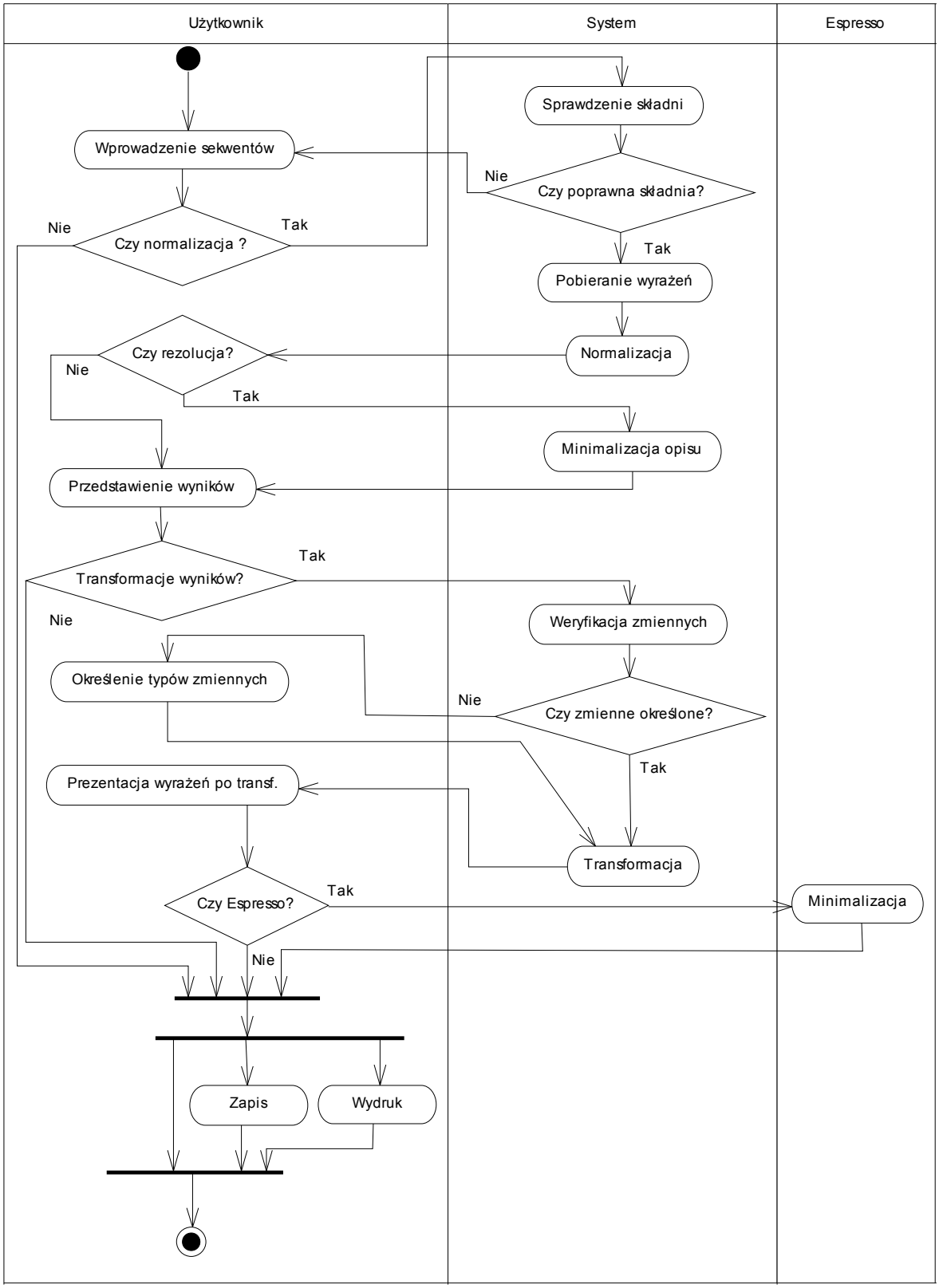
Główny algorytm programu przedstawiony został na Rys. 7-8 w formie diagramu aktywności [WrMa05][BoRu02]. Na diagramie wyróżniono trzy obszary aktywności: *Użytkownika*, *Systemu* wnioskującego i zewnętrznego programu *ESPRESSO*.

Użytkownik wprowadza do programu dane w postaci sekwentów i podejmuje decyzję o wykonaniu normalizacji. Następnie system weryfikuje poprawność składni. W przypadku błędów kontrola zwracana jest użytkownikowi, a gdy sekwenty są poprawne, to pobierane są one do procesu normalizacji i sprowadzane do postaci znormalizowanej. Po zakończeniu procesu normalizacji prezentowane jest użytkownikowi drzewo dowodu i sekwenty w postaci znormalizowanej. W przypadku, gdy ustawiony był parametr odpowiadający za uruchomienie metody rezolucji dochodzi do dodatkowej minimalizacji sekwentów znormalizowanych. Wyniki normalizacji i drzewo dowodu mogą zostać zapisać pliku lub wydrukowane.

W kolejnym kroku użytkownik podejmuje decyzję o wykonaniu transformacji opisu znormalizowanego. System sprawdza definicję typów formuł, a w przypadku ich braku prosi użytkownika o ich określenie. Po wykonaniu transformacji wyniki prezentowane są użytkownikowi i użytkownik podejmuje decyzję o ewentualnym ich zapisaniu lub wydrukowaniu.

W przypadku transformacji do formatu ESPRESSO użytkownik ma możliwość uruchomienia dodatkowej minimalizacji z wykorzystaniem zewnętrznego programu ESPRESSO. Warunkiem koniecznym jest posiadanie heurystycznego programu minimalizacji funkcji boolowskich ESPRESSO i wcześniejsze skojarzenie go z systemem wnioskującym poprzez okno opcji programu. Wyniki z programu ESPRESSO prezentowane są w oknie systemu wnioskującego, w którym znajdują się także przyciski przeznaczone do opcjonalnego zapisu oraz wydruku.

W celu lepszego zobrazowania algorytmu uszczegółowieniu poddane zostaną normalizacja (Rys. 7-9), metoda rezolucji (Rys. 7-11), a także algorytm wyszukiwania głównego spójnika, będące elementami składowymi procesu normalizacji (Rys. 7-10).




***Rys. 7-8 Diagram aktywności systemu***

Proces normalizacji przedstawiony na diagramie aktywności Rys. 7-9 rozpoczyna się od zlokalizowania głównego spójnika w rozpatrywanym sekwencie. W przypadku braku spójników logicznych sekwent znormalizowany dodawany jest do bazy rozwiązań

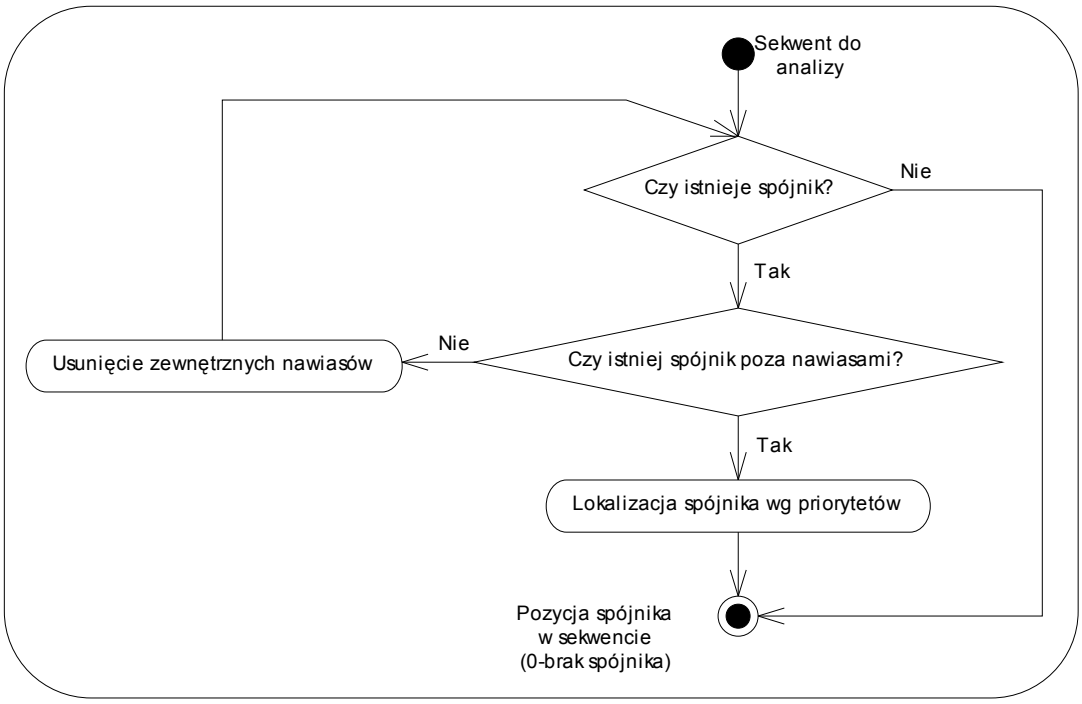
z zastosowaniem minimalizacji bazującej na dopuszczalnych regułach cięcia i dominacji. Jeżeli spójnik zostanie zlokalizowany, to rozpoczyna się proces jego eliminacji z wykorzystaniem reguł Gentzena. W wyniku eliminacji powstaje jeden lub dwa sekwenty wynikowe, które poddawane są testowi tautologii. Następnie wykonywane są reguły dopuszczalne, a także reguły wyprowadzone z metody Thelena-Mathony'ego. W przypadku dwóch sekwentów wynikowych pierwszy rozpatrywany jest dalej, natomiast drugi zostaje odłożony na stos w celu późniejszej jego normalizacji. Jeśli rozpatrywany sekwent nie jest tautologią, lub nie odrzuciła go żadna z reguł dopuszczalnych, to wraca on do procesu normalizacji i ponownie system próbuje zlokalizować w nim główny spójnik. Jeżeli sekwent zostaje odrzucony z procesu normalizacji, to pobierany jest sekwent odłożony na stosie, a gdy stos jest pusty, to pobierany jest kolejny sekwent z listy sekwentów wprowadzonych do systemu.


***Rys. 7-9 Diagram aktywności procesu normalizacji***

Wyszukiwanie głównego spójnika w rozpatrywanym sekwencie, przedstawione za pomocą diagramu Rys. 7-10, jest jedną z ważniejszych funkcji programu, gdyż od niego zależy sposób konstruowania drzewa dowodu. Od algorytmu tego zależy także komfort

obsługi programu, gdyż dzięki niemu możliwe jest wprowadzanie wygodnych nazw formuł, nie ma konieczności wprowadzania wszystkich nawiasów w sekwentach oraz dopuszczalne jest wprowadzanie sekwentów częściowo znormalizowanych.

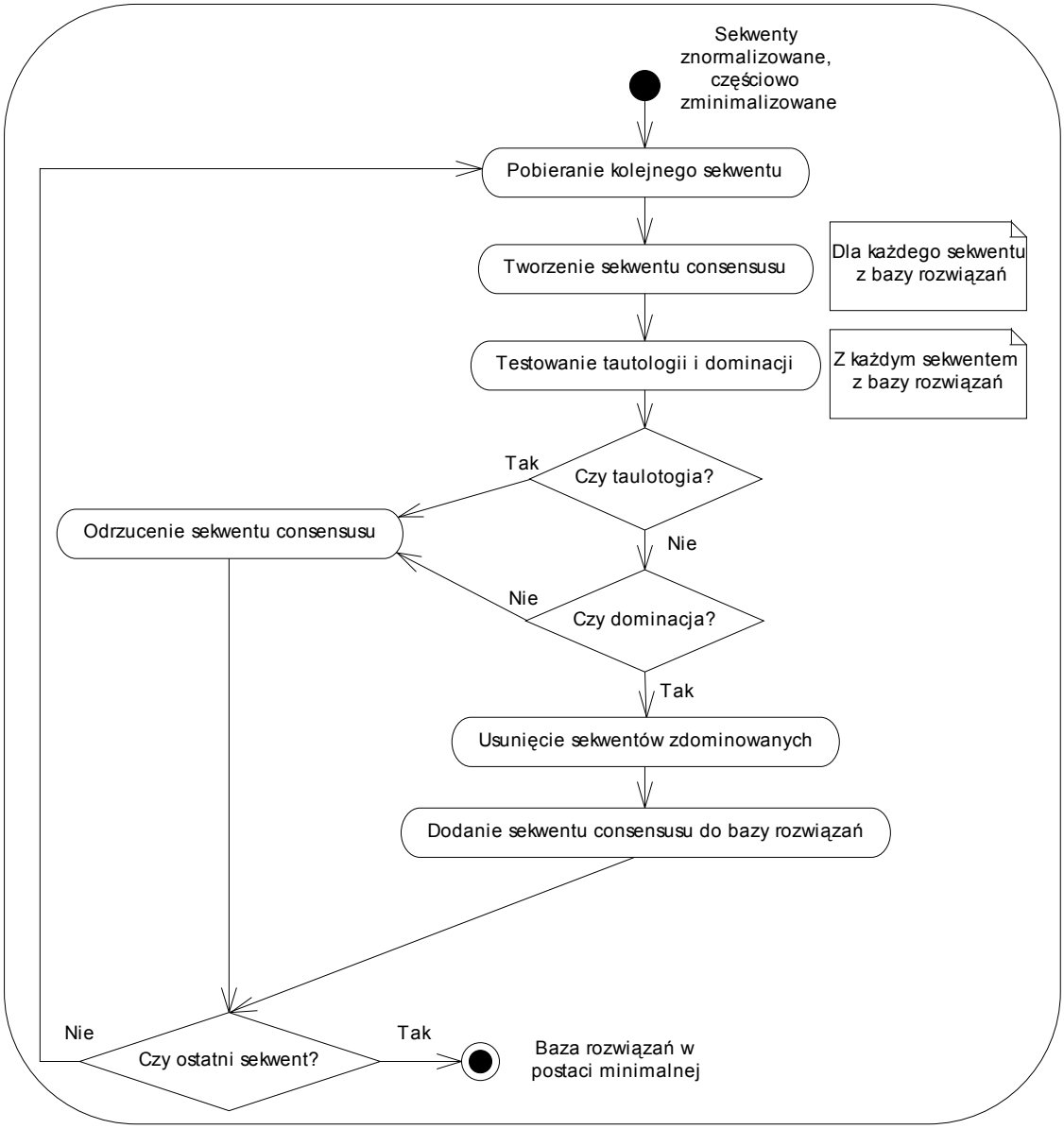
W pierwszym etapie algorytmu sprawdzane jest, czy w sekwencie występuje jakikolwiek spójnik logiczny. W przypadku braku spójnika algorytm kończy działanie i generuje informację o braku spójników logicznych (sekwent jest znormalizowany). Jeżeli spójnik występuje w rozpatrywanym sekwencie, to poszukuje się spójników występujących poza nawiasami. W przypadku braku spójników poza nawiasami, usuwa się pierwsze napotkane nawiasy i sekwent wraca do ponownej analizy. Jeżeli spójnik lub grupa spójników występuje poza nawiasami to wybiera się pierwszy o najwyższym priorytecie. W przypadku kilku spójników o tym samy priorytecie, wybierany jest pierwszy od lewej strony. Wynikiem działania algorytmu jest pozycja spójnika w sekwencie, która przekazywana jest dalej do procesu normalizacji.




***Rys. 7-10 Diagram aktywności wyszukiwania głównego spójnika***

Niekiedy zdarza się, że w przypadku normalizowania sekwentów o bardzo zawiłych, zagnieżdżonych formułach, sekwenty wynikowe nie zostaną sprowadzone do postaci minimalnej. Przyjęte założenie, że system wnioskujący będzie normalizował dowolne sekwenty i sprowadzał je do postaci minimalnej, bez względu na budowę formuł i zastosowane w nich spójniki logiczne, spowodowało, że do systemu wnioskowania został wprowadzony dodatkowy algorytm oparty na metodzie rezolucji (Rys. 7-11), do ewentualnego dalszego upraszczania otrzymanych wyrażeń symbolicznych. Z drugiej strony, w wielu przypadkach praktycznych, rezolucja realizowana jest prawie

natychmiastowo, bez konieczności wcześniejszej zawiłej normalizacji skomplikowanych wyrażeń boolowskich.



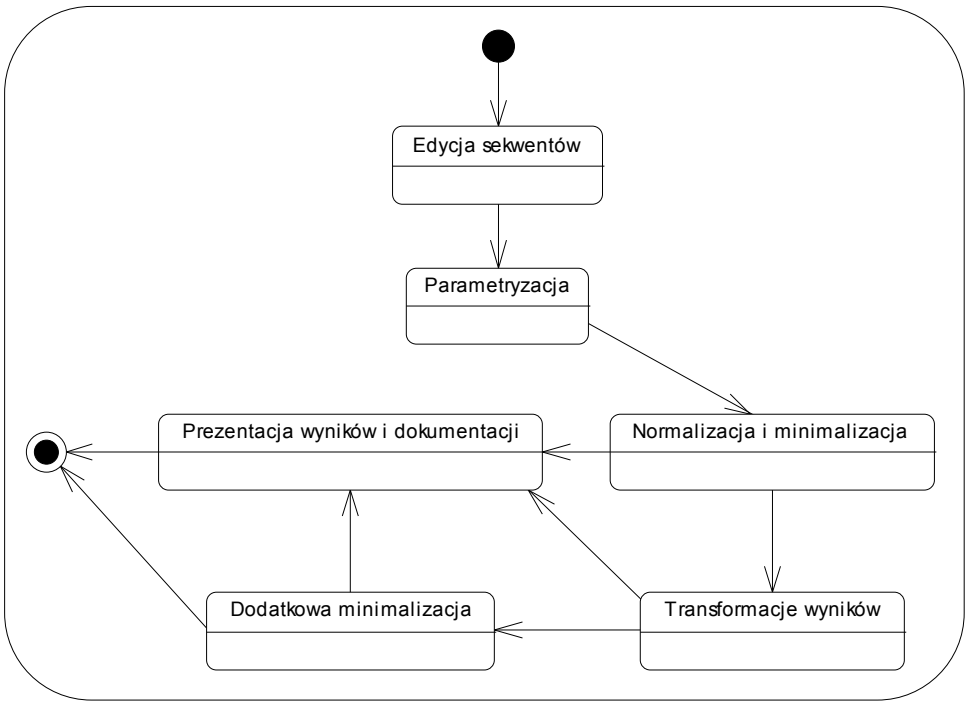

***Rys. 7-11 Algorytm rezolucji***

Algorytm ten polega na kolejnym pobieraniu sekwentów znormalizowanych i próbie tworzenia sekwentów consensusu z każdym innym sekwentem z bazy rozwiązań opisującym to samo zjawisko dyskretne. Jeżeli utworzony sekwent consensusu jest tautologią lub nie dominuje nad żadnym sekwentem z bazy rozwiązań, to jest pomijany. Sekwent consensusu, który nie dominuje nad żadnym innym sekwentem wyznacza nową, poprawną relację również opisującą rozpatrywane zjawisko dyskretne. Jednak w procesie minimalizacji będzie sekwentem nadmiarowym, a relacja którą wyznacza będzie opisana przez inne, prostsze sekwenty znajdujące się już w bazie rozwiązań. Sekwent consensusu, który dominuje nad jakimkolwiek sekwentem z bazy rozwiązań dodaje się do niej.

Sekwenty zdominowane usuwa się z bazy rozwiązań. Dla nowo dodanego do bazy sekwentu cała procedura musi zostać powtórzona.

### 7.4.4. Diagram stanów

W projektowanej aplikacji można wyróżnić sześć podstawowych stanów przedstawionych na poniższym diagramie (Rys. 7-12). Edycja, gdzie użytkownik wprowadza i edytuje wyrażenia (sekwenty) przygotowywane do normalizacji. Parametryzacja jest stanem, w którym użytkownik definiuje, jakie elementy algorytmu mają być aktywne.




***Rys. 7-12 Diagram stanów***

*Normalizacja i minimalizacja*, są stanem przetworzenia opisu zjawiska dyskretnego na postać znormalizowaną. *Transformacje wyników*, to stan przejścia z opisu znormalizowanego na różne inne postacie (formaty). *Dodatkowa minimalizacja* jest stanem, w którym wykonuje się minimalizację wygenerowanego opisu w formacie ESPRESSO przy użyciu zewnętrznego programu minimalizacji funkcji boolowskich. Ostatnim stanem jest *prezentacja wyników* użytkownikowi. Stany *transformacji wyników* oraz *dodatkowej minimalizacji* w procesie działania aplikacji mogą zostać pominięte.

## 7.5. Testowanie oprogramowania

W celu przeprowadzenia testów poprawności implementacji określono cztery obszary testowe dotyczące głównych funkcjonalności systemu:

- poprawność wnioskowania symbolicznego metodą Gentzena;
- skuteczność wprowadzenia elementów metody Thelena-Mathony’ego;
- efektywność wprowadzenia metody rezolucji;
- zespolenie wyżej wymienionych metod;
- testowanie parametryczne (skalowalne).

Testy poprawności implementacji algorytmu Gentzena przeprowadzono realizując rozwiązane wcześniej przykłady zaczerpnięte z literatury przedmiotu [Adam90][Szaj96]. Porównanie wyników potwierdziło poprawność implementacji algorytmu. Dodatkowo przeprowadzono formalnie dużą liczbę dowodów znanych twierdzeń z logiki, co było pierwotnym przeznaczeniem systemu Gentzena [ŁaMa04]. Dowody przeprowadzano zarówno przez potwierdzenie jak i zaprzeczenie, definiując założenia po odpowiedniej stronie znaku wynikania logicznego.

Testy poprawności implementacji algorytmu Thelena-Mathony’ego w logice sekwentów Gentzena przeprowadzono dla przykładów zaczerpniętych z literatury [Thel88][Math90][Kara07] dotyczących realizacji samego algorytmu, jak i badań nad występującymi w nim heurystykami.

Testy algorytmu rezolucji zaadoptowanego do logiki sekwentów Gentzena przeprowadzono, wprowadzając do systemu termy funkcji w postaci znormalizowanej bez wcześniejszej minimalizacji. Wyniki z zaimplementowanego procesu rezolucji porównywano z wynikami otrzymywanymi ręcznie z wykorzystaniem siatek Karnaugha, a także w przypadku bardziej złożonych funkcji, z wynikami wygenerowanymi przy pomocy programów do minimalizacji funkcji logicznych (ESPRESSO, BEM).

Testy poprawności zespolenia omawianych metod przeprowadzono minimalizując funkcje logiczne. Wyniki porównywano z rezultatami otrzymywanymi z programu do minimalizacji funkcji logicznych BEM (Boolean Expression Manipulator) korzystającego w procesie minimalizacji diagramów BDD [Shin96]. Zdecydowano się na użycie oprogramowania BEM ze względu na występujące w nim spójniki logiczne odpowiadające występującym w algorytmie wnioskowania symbolicznego (poza typowymi, najczęściej

spotykanymi, występują w nim również: implikacja, równoważność oraz alternatywa wyłączająca). Warto tutaj przypomnieć, że system BEM podaje tylko jedno rozwiązanie, otrzymane w sposób heurystyczny i nie przedstawia formalnego dowodu na to, że wynikowe wyrażenie logiczne jest poprawne.

Testowanie parametryczne przeprowadzono na podstawie skalowalnych grafów [Goga], zwiększając stopniowo liczbę ich wierzchołków. Warto zwrócić uwagę, że w odróżnieniu od metod heurystycznych, z wykorzystaniem których uzyskuje się pierwsze akceptowane rozwiązanie, przebadano również możliwość uzyskiwania wszystkich rozwiązań (pełny przegląd i wskazanie rozwiązań optymalnych).

# 8. Badania opracowanego algorytmu

Algorytm wnioskowania Gentzena wyznacza rozwiązania dokładne metodą symboliczną, więc nie można wprost porównywać wydajności jego pracy oraz przydatności praktycznej bezpośrednio z innymi algorytmami przybliżonymi (heurystycznymi). Z tego względu jakość wprowadzonych modyfikacji mierzona będzie liczbą węzłów w drzewie dowodu i porównywana z wynikami pochodzącymi z algorytmu wnioskującego z wyłączonymi autorskimi, zaproponowanymi w pracy usprawnieniami.

Pierwsze testy poprawności wnioskowania przeprowadzone zostały poprzez rozwiązywanie nietrywialnych zadań z logiki matematycznej. Ich wyniki zostały przedstawione w Tab. 8-1. Zadania zaczerpnięto z literatury [ŁaMa04].

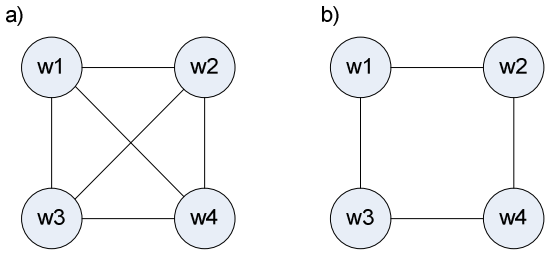
***Tab. 8-1 Testowanie algorytmu na wybranych zadaniach z logiki matematycznej***

| Nr testu | Liczba węzłów (Gentzen) | Czas (mm:ss.dst) | Liczba węzłów (Gentzen - nowa koncepcja | Czas (mm:ss.dst) | Skrócenie drzewa (%) |
|---|---|---|---|---|---|
| 1 | 32 | 00:00.032 | 17 | 00:00.016 | 47 |
| 2 | 239 | 00:00.141 | 89 | 00:00.063 | 63 |
| 3 | 16 | 00:00.016 | 15 | 00:00.015 | 7 |
| 4 | 14 | 00:00.015 | 14 | 00:00.015 | 0 |
| 5 | 76 | 00:00.032 | 38 | 00:00.015 | 50 |
| 6 | 60 | 00:00.031 | 40 | 00:00.031 | 34 |
| 7 | 63 | 00:00.047 | 41 | 00:00.015 | 35 |
| 8 | 157 | 00:00.094 | 66 | 00:00.047 | 58 |
| 9 | 19 | 00:00.016 | 16 | 00:00.015 | 16 |
| 10 | 9 | 00:00.016 | 9 | 00:00.015 | 0 |
| 11 | 13 | 00:00.016 | 11 | 00:00.015 | 16 |
| 12 | 13 | 00:00.016 | 13 | 00:00.015 | 0 |
| 13 | 15 | 00:00.015 | 15 | 00:00.015 | 0 |
| 14 | 15 | 00:00.016 | 15 | 00:00.016 | 0 |
| 15 | 18 | 00:00.016 | 17 | 00:00.016 | 6 |

Zadania *1-4* polegały na sprawdzeniu, dla jakich wartości zmiennych badana funkcja logiczna jest fałszywa. Zadania *5-9* dotyczą badania równoważności dwóch funkcji logicznych. Zadania *10-15* dotyczą wykorzystania rachunku sekwentów do dowodzenia twierdzeń w logice zdań. Z otrzymanych wyników można wywnioskować, że wprowadzone udoskonalenia nigdy nie pogarszają efektywności systemu wnioskującego, a w większości przypadków znacznie skracają drzewo dowodu. Rozwiązywane zadania nie

stanowiły problemu technicznego dla opracowanego systemu wnioskującego, a czasy realizacji poszczególnych zadań w większości mieściły się w 0,1s.

Badanie parametryczne opracowanego systemu wykonano za pomocą wybranych dwóch typów grafów (Rys. 8-1).




***Rys. 8-1 Testowane typy grafów – a) pełne b) cykliczne***

Grafy do testów zostały wybrane z opublikowanej biblioteki [Goga], a wyniki przeprowadzonych testów przedstawiono w Tab. 8-2.

***Tab. 8-2 Testowanie algorytmu z wykorzystaniem grafów***

| **Nr testu** | **liczba wierzcołków** | **Liczba węzłów (Gentzen)** | **Czas (mm:ss.dst)** | **Liczba węzłów (Gentzen - nowa koncepcja)** | **Czas (mm:ss.dst)** | **Skrócenie drzewa (%)** |
|---|---|---|---|---|---|---|
| 1 | 4 | 21 | 00:00.016 | 15 | 00:00.015 | 29 |
| 2 | 5 | 45 | 00:00.017 | 24 | 00:00.016 | 47 |
| 3 | 6 | 93 | 00:00.032 | 35 | 00:00.016 | 63 |
| 4 | 7 | 189 | 00:00.062 | 48 | 00:00.031 | 75 |
| 5 | 8 | 381 | 00:00.093 | 63 | 00:00.047 | 84 |
| 6 | 9 | 765 | 00:00.203 | 80 | 00:00.062 | 90 |
| 7 | 10 | 1533 | 00:00.469 | 99 | 00:00.078 | 94 |
| 8 | 4 | 18 | 00:00.016 | 12 | 00:00.015 | 54 |
| 9 | 5 | 35 | 00:00.016 | 18 | 00:00.016 | 49 |
| 10 | 6 | 68 | 00:00.016 | 26 | 00:00.016 | 62 |
| 11 | 7 | 133 | 00:00.031 | 36 | 00:00.016 | 73 |
| 12 | 8 | 262 | 00:00.078 | 49 | 00:00.031 | 82 |
| 13 | 9 | 519 | 00:00.172 | 65 | 00:00.047 | 88 |
| 14 | 10 | 1032 | 00:00.390 | 87 | 00:00.047 | 92 |
| 15 | 14 | 16587 | 00:14.742 | 541 | 00:00.687 | 97 |
| 16 | 11 | 2218 | 00:00.995 | 193 | 00:00.156 | 92 |

W testach parametrycznych wykorzystano dwa rodzaje grafów. Testy 1-7 przeprowadzone zostały dla grafów pełnych, gdzie każdy wierzchołek połączony jest krawędzią z każdym innym wierzchołkiem tego grafu (Rys. 8-1a). W testach 8-14

wykorzystano graf cykliczny, gdzie każdy wierzchołek ma tylko dwóch sąsiadów, przy czym pierwszy i ostatni wierzchołek są połączone krawędzią (Rys. 8-1b). Badania zostały przeprowadzone poprzez stopniowe zwiększanie liczby wierzchołków w rozpatrywanych grafach. Dla każdego grafu utworzono sekwent sąsiedztwa, odpowiadający wszystkim krawędziom grafu i poddano go normalizacji. Dodatkowo, w sposób losowy wybrano dwa złożone grafy z dostępnej biblioteki testowej [Goga]. Test 15 przeprowadzono dla grafu Heawood'a, a test 16 dla grafu Herschel'a.

Z przeprowadzonych testów wynika, że proponowana w pracy koncepcja systemu wnioskującego znacznie poprawia efektywność systemu wnioskującego opracowanego przez Gentzena. Należy w tym przypadku pamiętać również o tym, że istnieją grafy, dla których drzewo dowodu nie zostanie skrócone. Jednak w wielu przypadkach, a szczególnie w przypadku grafów opisujących rzeczywiste zjawiska dyskretne, takie skrócenia będą występowały.

Kolejną grupę testów przeprowadzono dla grafów losowych. W ramach testu opracowany został program wspomagający losowe generowanie macierzy sąsiedztwa. Na podstawie wylosowanej macierzy, dla zadanej liczby wierzchołków, wyznaczane są dwa rodzaje sekwentów sąsiedztwa. Program testowy umieszczony został na załączonej do pracy płycie CD, której strukturę opisano w dodatku B.

Pierwszy test dla gafów losowych przeprowadzono budując sekwenty według następującej zależności (8-1). Sekwent sąsiedztwa reprezentuje grupę iloczynów, gdzie argumenty stanowią sumy kolejnych wierzchołków z iloczynami wierzchołków sąsiadujących. Dodatkowo sekwent (8-1) definiujący sąsiedztwa uproszczono poprzez zastosowanie częściowej normalizacji, zastępując główne iloczyny przecinkami.

$$(w1 + (w2*w3*w_i)), (w2+(w3*w5*w_i)), \ldots, (w_n+(w_x*\ldots*w_y))|-; \qquad \textbf{(8-1)}$$

Testowe grafy przygotowywano zwiększając, co 5 liczbę wierzchołków i losując krawędzie pomiędzy nimi. Wyniki przeprowadzonych testów zawarte zostały w Tab. 8-3 i zawierają, poza wartościami charakteryzującymi graf, liczbę uzyskanych rozwiązań, liczbę węzłów występujących w drzewie dowodu oraz czas jego realizacji. Dodatkowo przedstawiony został czas uzyskania pierwszego rozwiązania.

Przeprowadzając testy dla losowych grafów zaobserwować można skuteczność algorytmu przy bardzo złożonych wyrażeniach logicznych. Opracowany algorytm radzi sobie z wyrażeniami reprezentującymi sąsiedztwa grafów losowych, złożonymi z około 50-55 zmiennych logicznych.

***Tab. 8-3 Testy grafów generowanych losowo***

| Nr testu | Liczba wierzchołków | Liczba krawędzi | Liczba węzłów | Liczba rozwiązań | Czas analizy [mm:ss.dst] | Pierwsze rozwiązanie [mm:ss.dst] |
|---|---|---|---|---|---|---|
| 1 | 5 | 7 | 25 | 4 | 00:00.017 | 00:00.015 |
| 2 | 10 | 31 | 119 | 9 | 00:00.078 | 00:00.031 |
| 3 | 15 | 48 | 797 | 30 | 00:00.891 | 00:00.047 |
| 4 | 20 | 96 | 2574 | 48 | 00:04.328 | 00:00.125 |
| 5 | 25 | 154 | 3110 | 84 | 00:09.812 | 00:00.234 |
| 6 | 30 | 202 | 13821 | 204 | 01:06.493 | 00:00.391 |
| 7 | 35 | 282 | 26667 | 311 | 03:20.040 | 00:00.640 |
| 8 | 40 | 375 | 52584 | 499 | 08:57.868 | 00:01.094 |
| 9 | 45 | 483 | 77581 | 711 | 18:10.531 | 00:01.532 |
| 10 | 50 | 610 | 91751 | 986 | 31:56.025 | 00:02.109 |

Niestety, wyznaczone pierwsze rozwiązanie dla sekwentu zbudowanego według zależności (8-1) może nie być rozwiązaniem minimalnym (w sensie liczby zmiennych). Będzie to rozwiązanie prawidłowe, lecz krótsze, inne rozwiązanie, może pojawić się w dalszym procesie normalizacji. Rozwiązanie krótsze zredukuje wszystkie mniej korzystne rozwiązania, które uzyskano wcześniej, co omówione zostało w rozdziale 4.3

Mając na celu wyznaczenie od razu minimalnego pierwszego rozwiązania, należy sekwent opisujący sąsiedztwa grafu przygotować w klasycznej postaci koniunkcyjnej lub dysjunkcyjnej, analogicznie jak zostało to opisane w przypadku analizy grafów współbieżności (rozdz 6.1). Konstrukcja sekwentu, w postaci koniunkcyjnej na pewno spowoduje natychmiastowe wykorzystanie zaimplementowanej reguły *R2* z algorytmu Thelena-Mathony'ego (rozdz. 4.4.2), w wyniku użycia której już pierwsze uzyskane rozwiązania będą rozwiązaniami minimalnymi.

$$(w1+w2),\ (w1+w_i),(\ w2+w_j),\ldots,\ (w_m+w_n)|\text{-};\qquad \textbf{(8-2)}$$

Sekwent sąsiedztwa buduje się według zależności (8-2) tak, że będzie on iloczynem sum sąsiadujących ze sobą wierzchołków, gdzie każda krawędź wymieniona jest tylko jeden raz. Proponowany sekwent sąsiedztwa (8-2) będzie zapisany w formie częściowo znormalizowanej, czego następstwem jest zastąpienie iloczynów przecinkami. Tak przygotowany sekwent będzie posiadał znacznie bardziej rozbudowane drzewo dowodu w stosunku do sekwentu zdefiniowanego w poprzednim teście, gdzie sąsiedztwa opisano zależnością (8-1). Większe drzewo dowodu spowoduje zwiększenie się czasu normalizacji, w przypadku poszukiwania wszystkich rozwiązań. Spowodowane jest to występowaniem większej ilości sum po lewej stronie sekwentu (w poprzedniku), gdzie w procesie

normalizacji każda suma rozwijana jest w dwóch osobnych poddrzewach. Zarówno w przypadku normalizacji sekwentu, dla przykładowego grafu, przygotowanego według zależności (8-1), jaki i (8-2) wynik pełnego procesu normalizacji będzie identyczny. Chcąc uzyskać pierwsze minimalnym rozwiązanie należy zbudować sekwent sąsiedztwa grafu według zależności (8-2).

Badanie przeprowadzono dla tych samych grafów, które zostały wylosowane do poprzedniego testu i rozszerzono je o dwa dodatkowe, nadal zwiększając liczbę wierzchołków. Wyniki zawarte w Tab. 8-4 prezentują czas oraz liczbę węzłów do momentu uzyskania pierwszego minimalnego rozwiązania.

***Tab. 8-4 Analiza sekwentów sąsiedztwa grafów losowych w postaci koniunkcyjnej***

| **Nr testu** | **Liczba wierzchołków** | **Liczba krawędzi** | **Pierwsze minimalne rozwiązanie** | |
|---|---|---|---|---|
| | | | **Liczba węzłów** | **Czas analizy [mm:ss.dst]** |
| 1 | 5 | 7 | 11 | 00:00.015 |
| 2 | 10 | 31 | 32 | 00:00.062 |
| 3 | 15 | 48 | 49 | 00:00.156 |
| 4 | 20 | 96 | 97 | 00:00.625 |
| 5 | 25 | 154 | 156 | 00:01.609 |
| 6 | 30 | 202 | 204 | 00:02.922 |
| 7 | 35 | 282 | 308 | 00:07.046 |
| 8 | 40 | 375 | 552 | 00:21.045 |
| 9 | 45 | 483 | 588 | 00:27.668 |
| 10 | 50 | 610 | 612 | 00:49.779 |
| 11 | 55 | 756 | 852 | 01:40.538 |
| 12 | 60 | 867 | 1135 | 03:08.078 |

W wielu przypadkach uzyskanie pierwszego minimalnego rozwiązania będzie wystarczające. W przypadku analizy grafów nieskierowanych jego dopełnienie będzie jednym z minimalnych zbiorów niezależnych.

Z testów wynika, że proponowany algorytm potrafi wyznaczyć pierwsze minimalne rozwiązanie w przypadku losowych wyrażeń złożonych z około 60 zmiennych logicznych.

Należy tutaj nadmienić, że zasadniczym celem systemu wnioskującego jest wspieranie procesu projektowania, przeprowadzanego przez człowieka, a nie konkurowanie z wyrafinowanymi programami heurystycznymi, udoskonalanymi przez wielu matematyków i informatyków.

W tabeli Tab. 8-5 umieszczono wyniki testów normalizacji sekwentów dla przykładów zawartych w pracy.

***Tab. 8-5 Testowanie algorytmu z wykorzystaniem przykładów zawartych w pracy***

| Nr testu | Liczba węzłów (Gentzen) | Czas (mm:ss.dst) | Liczba węzłów (Gentzen - nowa koncepcja | Czas (mm:ss.dst) | Skrócenie drzewa (%) |
|---|---|---|---|---|---|
| 1 | 55 | 00:00.032 | 31 | 00:00.015 | 44 |
| 2 | 77 | 00:00.141 | 77 | 00:00.141 | 0 |
| 3 | 486 | 00:00.063 | 182 | 00:00.079 | 63 |
| 4 | 87 | 00:00.047 | 87 | 00:00.047 | 0 |
| 5 | 61 | 00:00.016 | 54 | 00:00.031 | 12 |
| 6 | 127 | 00:00.036 | 116 | 00:00.031 | 9 |
| 7 | 74 | 00:00.031 | 68 | 00:00.031 | 9 |
| 8 | 311 | 00:00.172 | 311 | 00:00.172 | 0 |
| 9 | 222 | 00:00.156 | 222 | 00:00.156 | 0 |
| 10 | 16777216 | 32:14:945 | 162 | 00:00.110 | 99 |
| 11 | 22226400 | 38:24.352 | 690 | 00:00.265 | 99 |
| 12 | 25 | 00:00.016 | 20 | 00:00.016 | 20 |
| 13 | 584 | 00:00.109 | 420 | 00:00.125 | 29 |
| 14 | 390 | 00:00.124 | 390 | 00:00.124 | 0 |

Test 1 przeprowadzono dla przykładu (Przykład 4-1) z rozdziału 4.4.2, w którym zaprezentowano połączenie algorytmu wnioskującego z metodą Thelena-Mathony'ego. Test 2 dotyczy przykładu (Przykład 5-1) z rozdziału 5.1, w którym omawiana jest synteza układu kombinacyjnego z wykorzystaniem rachunku sekwentów. Test 3 dotyczy sekwentowej realizacji metody Petricka (Przykład 5-2) – rozdział 5.1.1. Test 4 przeprowadzono dla przykładu (Przykład 5-3) z rozdziału 5.1.1 i dotyczy normalizacji behawioralnego opisu specyfikacji układu sterowania. W testach 5 i 6 weryfikowano implementację układu kombinacyjnego z częściową lub pełną specyfikacją (Przykład 5-4) – rozdział 5.1.2. Test 7 przeprowadzono podczas wyznaczania wzbudzeń przerzutnika JK dla układu licznika synchronicznego mod4 (Przykład 5-5) – rozdział 5.2.1. Test 8 przeprowadzono dla specyfikacji sekwentowej wygenerowanej z klasycznej tabeli przejść wyjść (Przykład 5-6) – rozdział 5.2.2. Test 9 przeprowadzony został podczas normalizacji specyfikacji w formie sekwentowej, która przygotowana została na podstawie sieci działań (Przykład 5-7) – rozdział 5.3. Testy 10 i 11 wykonane zostały podczas analizy grafu współbieżności (Przykład 6-1) – rozdział 6.1, gdzie test 10 przeprowadzono dla wyznaczania maksymalnych zbiorów niezależnych, a test 11 dla poszukiwania

minimalnych zbiorów dominujących. Test 12 wykonany został dla kolorowania automatowego sieci Petriego (Przykład 6-2) – rozdział 6.2. Testy 13 i 14 zrealizowano podczas wyznaczania blokad i pułapek w sieci Petriego (Przykład 6-3) – rozdział 6.3.

Z przeprowadzonych testów wynika, że dla bardzo złożonych przykładów, wprowadzenie do algorytmu Gentzena proponowanych rozszerzeń i elementów innych metod daje nieocenioną poprawę jego efektywności. Szczególnie w testach 10 i 11 z Tab. 8-5 można zaobserwować znaczne skrócenie drzewa dowodu (około 99%) W wielu przypadkach, gdzie do skrócenia nie dochodzi, warto zastanowić się innym alternatywnym sposobem opisu tego samego zjawiska dyskretnego lub jego specyficznym charakterem. Projektant, korzystając z bazy swoich doświadczeń może opracować różne wersje opisów, dla różnych specyficznych badań, które mogą prowadzić do uzyskania lepszych rezultatów.

Wyniki testów nie uwidaczniają, jeszcze jednej bardzo istotnej cechy opracowanego algorytmu. W wielu przypadkach wynik normalizacji nie jest wyznaczany w formie minimalnej, tylko jako następstwo dowolnej, nieraz nadmiernie zawiłej konstrukcji sekwentów oraz stosowania skomplikowanych głęboko zagnieżdżonych spójników logicznych (implikacja, równoważność, alternatywa wyłączająca). Algorytm Gentzena, w pierwotnej formie, przy zastosowaniu klasycznych reguł dopuszczalnych, znormalizuje, uprości do zwartej postaci, ale nie sprowadzi sekwentów do optymalnej postaci minimalnej. Pomocna, w tym przypadku staje się zaproponowana w pracy i wprowadzona do systemu wnioskującego metoda rezolucji, która uruchomiona po zakończeniu procesu normalizacji sprowadzi znormalizowany opis do postaci minimalnej.

Omawiany prototypowy system komputerowego wnioskowania skutecznie i poprawnie. Optymalizacja otrzymywanych wyników oraz skrócenie czasu obliczeń poprzez udoskonalanie metody i algorytmów będzie przedmiotem dalszych prac autora.

# 9. Podsumowanie

Jednym z najważniejszych rezultatów pracy doktorskiej jest koncepcja autorskiego systemu wnioskowania o dużej wartości użytkowej, nie mająca swojego odpowiednika na rynku informatycznym. Zaproponowane połączenie wnioskowania symbolicznego Gentzena z metodą Thelena-Mathony'ego oraz metodą rezolucji znacznie zwiększa walory użytkowe proponowanego autorskiego sytemu wnioskowania. Zintegrowanie systemu wnioskowania z profesjonalnymi i uniwersyteckimi systemami CAD umożliwia praktyczne wykorzystanie metod formalnych w komputerowym projektowaniu układów cyfrowych, zgodnie z koncepcjami zawartymi w pracach [Adam90][Evek87][Krop99].

Możliwość komputerowego udowadniania twierdzeń z logiki zdań, może mieć duże znaczenie z dydaktycznego punktu widzenia [ŁaMa04][Indr05][Indr06].

W wyniku opracowanej koncepcji i prowadzonych badań zrealizowany został w języku C++ autorski system wnioskujący ukierunkowany na wspomaganie projektowania układów sterowania binarnego. Po raz pierwszy otwarto system komputerowego wnioskowania na współpracę ze znanymi uniwersyteckimi programami przeznaczonymi do syntezy układów cyfrowych np. DEMAIN [Łuba01]. Uzyskano to przez zastosowanie transformacji wyrażeń znormalizowanych do formatu ESPRESSO. W ten sposób dokonano integracji systemu wnioskującego z uniwersyteckim heurystycznym systemem dekompozycji i minimalizacji funkcji logicznych opracowanym na Politechnice Warszawskiej.

Opisany format jest również akceptowany przez systemy VIS i SIS opracowane na Uniwersytecie Berkeley. Wprowadzono możliwość symulacji i prototypowej syntezy specyfikowanych formalnie układów cyfrowych w układach FPGA za pośrednictwem języków opisu sprzętu(VHDL i VERILOG) [Zwol07].

W pracy doktorskiej zostały opracowane i przedstawione przykładowe metody projektowania, analizy oraz symulacji logicznej układów sterowania binarnego z wykorzystaniem automatycznego wnioskowania Gentzena.

Podstawowy cel pracy osiągnięto poprzez realizację zadań **1-3** zdefiniowanych w rozdziale 2.3. Nowatorska koncepcja systemu wnioskującego (rozdz. 3 i 4), jego projekt i implementacja (rozdz. 7) stanowią udokumentowanie realizacji zadań **1-3**. Badania efektywnościowe algorytmu (rozdz. 8) potwierdzają realizację zadania **4**, a także uwiarygodniają postawioną tezę szczegółową, dotyczącą wprowadzania do systemu dodatkowych reguł dopuszczalnych, w celu usprawnienia algorytmu i skrócenia drzewa dowodu. Drugi cel pracy osiągnięto podczas realizacji zadania **6**, w wyniku którego

zaproponowano szereg przykładowych obszarów zastosowania wnioskowania symbolicznego Gentzena, gdzie projektant może wspierać się wnioskowaniem symbolicznym podczas projektowania układów sterowania binarnego (rozdz. 5 i 6).

Teza pracy została udowodniona na drodze teoretycznej (logika matematyczna – rozdz 5.1.1, teoria grafów – rozdz. 6.1) oraz eksperymentalnej (przykłady wykorzystania systemu w rozwiązywaniu konkretnych problemów technicznych – rozdz. 5 i 6). Na udowodnienie tezy składają się realizacje zadań **5** (rozdz. 7)**, 7** i **8** (rozdz. 5.3 i 5.4), w wyniku których powstał oryginalny system wnioskujący według reguł Gentzena, dostosowany do wspomagania prac inżyniera, projektanta systemów cyfrowych, w szczególności systemów sterowania binarnego.

Opracowany system pozwoli zintensyfikować badania nad metodami formalnego opisu i komputerowej syntezy silnie współbieżnych sterowników cyfrowych [Adam90][Kara07][AdWę07].

## 9.1. Elementy nowatorskie i autorskie

Nowatorskimi elementami pracy są:

- opracowanie i zautomatyzowanie sekwentowej wersji algorytmu Thelena-Mathony'ego;
- opracowanie i zautomatyzowanie sekwentowej wersji algorytmu rezolucji;
- zespolenie algorytmu wnioskowania symbolicznego Gentzena z opracowanymi wersjami sekwentowymi algorytmów Thelena-Matchonego oraz rezolucji;
- wprowadzenie szeregu usprawnień do algorytmu wnioskującego zwiększających jego efektywność (wprowadzenie reguł dopuszczalnych i wczesne wykrywanie tautologii), jak i przydatność do projektowania układów cyfrowych;
- zaproponowanie wykorzystania rachunku sekwentów Gentzena, w ujęciu wprowadzania spójników logicznych (syntezy sekwentu), do transformacji sekwentów znormalizowanych na różne formaty akceptowalne przez uniwersyteckie i profesjonalne programy CAD, przeznaczone do projektowania, syntezy i analizy układów cyfrowych.

Ponadto wyróżnić można również autorskie osiągnięcia praktyczne pracy, którymi są:

- opracowanie składni formatu dla języka sekwentów przeznaczonego do wykorzystania przy wspomaganiu projektowania układów cyfrowych;

- utworzenie prototypu systemu wnioskującego, zgodzenie z koncepcją opracowaną w ramach pracy doktorskiej, wspomagającego projektowanie, analizę i weryfikację układów cyfrowych, a w szczególności układów sterowania binarnego;
- utworzenie zmodyfikowanej wersji prototypu systemu wnioskującego przeznaczonego do badania zastojów w układach sterowania binarnego.

## 9.2. Plany dalszych prac

Elastyczność i wszechstronność metody wnioskowania symbolicznego zachęca do prowadzenia dalszych badań nad wykorzystaniem metod formalnych w informatyce i elektronice [Krop99].

- **Poszukiwanie kolejnych usprawnień eliminujących nadmiarowość przetwarzanych danych.** W przypadku zawiłych formuł i braku ich jakiegokolwiek uporządkowania, a także przy zastosowaniu wielu spójników logicznych, pojawiać się będą nadmiarowe, niepotrzebnie analizowane poddrzewa w drzewie dowodu, które należałoby wykryć i wyeliminować z procesu analizy.
- **Badania nad podatnością algorytmu na heurystyki.** Ze wstępnych obserwacji wynika, że po wprowadzeniu do metod symbolicznych Gentzena korzystnych elementów heurystyki, własności te w wielu przypadkach będą występowały również w nowo opracowanym algorytmie.
- **Zaproponowanie kolejnych rozszerzeń dla języka logiki sekwentów zwiększających użyteczność systemu wspomagającego prace projektowe.** Kolejne rozszerzenia mogłyby doprowadzić do realizacji zintegrowanego środowiska projektowego służącego do projektowania, analizy i symulacji układów cyfrowych, gdzie specyfikacja dla złożonych systemów cyfrowych przygotowywana byłaby za pomocą opracowanego języka logiki sekwentów.

Doświadczenia zdobyte podczas projektowania, programowania i testowania gentzenowskiego systemu wnioskowania skłania autora do podjęcia się nowego wyzwania, polegającego na implementacji dedukcji naturalnej w klasycznym rachunku zdań w ujęciu zaproponowanym w monografii [Indr06] Motywacją do próby zautomatyzowania wnioskowania wspieranego tablicami semantycznymi jest ich coraz większa popularność [Bena05] wśród informatyków.

Uzupełnienia wymagają badania porównawcze systemu gentzenowskiego z innymi formami wnioskowania komputerowego [Indr06]. Jak wynika z niektórych badań, wnioskowanie symboliczne może z powodzeniem konkurować z modnym obecnie i wdrożonym do przemysłu sposobem wykorzystującym diagramy OBDD [BaSc01].

Można przypuszczać, że jego zalety uwidocznią się zwłaszcza w przypadku symbolicznej analizy sieci Petriego [PaCo01] oraz przy projektowaniu cyfrowych systemów osadzonych [Dvor07].

Warte rozważenia jest zaadoptowanie systemu wnioskowania do generowania dowodów sekwentowych w logice liniowej [RiFe05].

# [Literatura]


[Adam90] Adamski M.: *Projektowanie układów cyfrowych systematyczną metodą strukturalną*, Wydawnictwo Wyższej Szkoły Inżynierskiej w Zielonej Górze, Zielona Góra 1990.

[Adam91] Adamski M.: *Parallel Controller Implementation using Standard PLD Software. FPGAs*. Abingdon, England 1991, pp 296-304.]

[AdCh00] Adamski M., Chodań M.: *Modelowanie układów sterowania dyskretnego z wykorzystaniem sieci SFC.* Wydawnictwo Politechniki Zielonogórskiej, Zielona Góra 2000.

[AdKa05] Adamski M, Karatkevich A., Węgrzyn M.: *Design of embedded control systems.* - New York : Springer, 2005, s. 267

[AdBa06] Adamski M, Barkalov A.: *Architectural and Sequential Synthesis of Digital Devices.* University of Zielona Góra Press. Zielona Góra 2006.

[AdWę07] Adamski M., Węgrzyn M., Węgrzyn A.: *Safe Reconfigurable Logic Controllers Design*. W: Measurements Models Systems and Design. Ed. Józef Korbicz. WKiŁ, Warszawa 2007. pp 343-370.

[BaSc01] Baldamus M., Schneider K.: *The BDD Space Complexity of Different Forms of Concurrency.* Proceedings of the Second Conference on Application of Concurrency to System Design, 2001 IEEE.

[BaKu93] Banaszak Z., Kuś J., Adamski M.: *Sieci Petriego. Modelowanie, Sterowanie i Synteza Systemów Dyskretny.* Wydawnictwo Wyższej Szkoły Inżynierskiej, Zielona Góra 1993.

[BaWę06] Barkalov A., Węgrzyn M.: *Design of control units with programmable logic.* University of Zielona Góra Press, Zielona Góra 2006.

[BaMi92] Barkaoui K., Minoux M.: *A polynomial-time graph algorithm to decide liveness of some basic classes of bounded Petri nets.* Lecture Notes in Computer Science, Berlin 1992:Springer Verlag, Vol.616,ss.62-75.

[Bena05] Ben-Ari M.: *Logika matematyczna w informatyce,* Wydawnictwa Naukowo Techniczne, Warszawa 2005.

[BiWo93] Biela A., Wojtylak M.: *Automatyczne dowodzenie twierdzeń.* Uniwersytet Śląski, Katowice 1993.

[BiAd94] Biliński K., Adamski M., Saul J. M., Dagless E. L.: *Petri-net-based algo-rithms for parallel-controller synthesis*. - IEE Proceedings - Computers and Digital Techniques .- 1994, Vol. 141, no 6, ss. 405-412

[BoRu02] Booch G., Rumbaugh J., Jacobson I.: *UML przewodnik użytkownika*. Wydawnictwa Naukowo-Techniczne, Warszawa 2002

[Buko08] Bukowiec A.: *Automata Synthesis System – Sample FSMs*. Publikacja internetowa: http://willow.iie.uz.zgora.pl/~abukowie/AS/as.htm

[Comm72] Commoner F.: *Deadlocks in Petri Nets.* - Wakefield 1972: Applied Data Res. Inc

[Demi98] De Micheli G.: *Synthesis and Optimization of Digital Circuits.* McGraw-Hill, New York, 1994. Również tłumaczenie polskie: *Synteza i optymalizacja układów cyfrowych.* WNT, Warszawa 1998.

[Deo80] Deo N.: Teoria grafów w technice i informatyce. PWN, Warszawa 1980.

[Dvor07] Dvorak V.: *Time- and Space-Efficient Evaluation of Sparse Boolean Functions in Embedded Software.* Engineering of Computer-Based Systems, 2007 IEEE, pp. 178-185.

[ErPa84] Ershov Yu.L, Palyutin E.A.: *Mathematical Logic.* Mir Publisher, Moscow 1984.

[Evek85] Eveking H.: *The application of CHDL's to the abstract specification of hardware.* Computer Hardware Description Languages and their Applications. C.J. Koomen and T. Moto-oka (eds) Elsevier Science Publishers. B.V. (North-Holland) IFIP 1985, pp. 167-178.

[Evek87] Eveking H.: *Verification synthesis and correctness – preserving transformations – cooperative approaches to correct hardware design.* From HDL Descriptions to Guaranteed Correct Circuits Design. D. Borrione (editor) Elsevier Science Publishers. B.V. (North-Holland) IFIP 1987, pp. 229-239.

[Goga] *Gallery of graphs and algorithms*. Internet source on page: http://www.hamline.edu/~lcopes/SciMathMN/gallery.html

[Gall86] Gallier J.H.: Logic of Computer Science. Foundations of Automatic Theorem Proving. Hapv&Row Publisher, New York 1986

[Gbur78] Gburzyński P.: *Automatyczne dowodzenie twierdzeń z wykorzystaniem zasady rezolucji*. Prace IPI PAN – ICS Pas Reports 1978, No.319.

[Gent69] Gentzen G.: *The Collected Papers of Gerhard Gentzen.* Edited by M.E. Szabo. North-Holland Publishing Company, Amsterdam 1969.

[Gent80] Gentzen G.: *Badania nad wniskowaniem logicznym*. Tłumaczenie z języka niemieckiego dr Kazimierz Trzęsicki. Sekcja Wydawnicza Filii UW w Białymstoku, Białystok 1980.

[HaMa97] Halang W., Adamski M.: *A programmable electronic system for safety related control applications*. Advances in safety and reliability : proceedings of the International Conference - ESREL '97 . Vol. 1 .- Oxford : Pergamon, 1997 - pp. 349--355

[Huza07] Huzar Z.: *Elementy logiki i teorii mnogości dla informatyków*. Oficyna Wydawnicza politechniki Wrocławskiej, Wrocław 2007.

[Indr] Indrzejczak A.: *Wprowadzenie do rachunku sekwentów- zagadnienia metodoogiczne, zastosowania.* Książka internetowa. http://www.filozof.uni.lodz.pl/prac/ai/Gentzen.pdf

[Indr05] Indrzejczak A.: *Elementy logiki.* Wyższa Szkoła Humanistyczno-Ekonomiczna w Łodzi, Łódź, 2005

[Indr06] Indrzejczak A.: *Hybrydowe systemy dedukcyjne w logikach modalnych.* Wydawnictwo Uniwersytetu Łódzkiego, Łódź, 2006

[Jers81] Jerszow A.P: *Wprowadzenie do teorii programowania.* WNT, Warszawa 1981

[Kara07] Karatkevich A.: *Dynamic Analysis of Petri Net-Based Discrete Systems*. Springer Verlag, Berlin 2007

[Krop99] Kropf T.: *Introduction to Formal Hardware Verification*. Springer Verlag, Berlin 1999

[Lige03] Ligenza A.: *Logical Foundations for Rule-Based Systems*. Uczelniane Wydawnictwo Naukowo-Dydaktyczne Akademii Górniczo-Hutniczej, Kraków 2003.

[Logi87] *Logika formalna.* Zarys encyklopedyczny z zastosowaniem do informatyki i lingwistyki. Praca zbiorowa pod red. W. Marciszewskiego. PWN, Warszawa 1987.

[Lynd87] Lyndon R.C.: *O logice matematycznej*. PWN, Warszawa 1987, (tłumaczenie).

[ŁaMa04] Ławrow I., Maksimowa Ł.: *Zadania z teorii mnogości, logiki matematycznej i teorii algorytmów*. Wydawnictwo Naukowe PWN, Warszawa 2004.

[ŁuJa97] Łuba T., Jasiński K., Zbierzchowski B.: *Specjalizowane układy cyfrowe w strukturach PLD i FPGA*, Wydawnictwa Komunikacji i Łączności, Warszawa 1997.

[Łuba01] Łuba T.: *Synteza układów logicznych*, Wyższa Szkoła Informatyki Stosowanej i Zarządzania, Wyd. 2, poprawione i rozszerzone, Warszawa 2001.

[Majc95] Majczak, A:. *Praktyczne programowanie w C++*, Intersoftland, Warszwa 1995

[Math90] Mathony H.J.: *Uniwersal logic design algorithm and its application the synthesisof two-lewel switching circuits*, IEE Proceedings Letters, Elsevier Science Publishers (North Holland),Vol.29,1990,pp. 195-210.

[MoPa70] Mostowski A.W. Pawlak Z.: *Logika dla inżynierów*. PWN, Warszawa, 1970.

[Mura89] Murata T.: *Petri Nets: Properties, Analysis and Applications*. Proceedings of the IEEE 1989, Vol.77, No 4, ss. 541-580.

[Nels55] Nelson R.: *Simplest normal truth functions*. Journal of Symbolic Logic, 20(2):105-108, 1955.

[Odmi02] *Optymalizacja dyskretna. Modele i metody kolorowania grafów*. Red. M. Kubale, WNT, Warszawa 2002.

[Pawl65] Pawlak Z.: *Automatyczne dowodzenie twierdzeń*, Państwowe Zakłady Wydawnictw Szkolnych, Warszawa 1965.

[PaCo01] Pastor E., Cortadella J., Roig O.: *Symbolic Analysis of Bounded Petri Nets*. IEEE Transitions on Computers, Vol. 50 No 5, 2001 pp 432-448.

[PiPi07] Pilone D., Pitman N.: *UML 2.0. Almanach*. Helion, Gliwice 2007

[Pogo81] Pogorzelski W.A.: *Klasyczny rachunek kwantyfikatorów*. PWN, Warszawa 1981.

[PowDe] PowerDesigner - Enterprise architecture modeling and design solution. http://www.sybase.com/products/modelingmetadata/powerdesigner

[ReCl90] Reeves S., Clarke M.: *Logic for computer science*. The Bath Press, Avon 1990.

[Reis03] Reisdorph K.: *C++ Builder 6 dla każdego*. Helion, Warszawa 2003

[RiFe05] Ribeiro O., Fernandes J., Pinto L.: *Model Checking Embedded Systems with PROMELA*. Proceedings of 12th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems, 2005.

[Rich84] Richards T.J.: *Formal Logic. The Automation of Reasoning*. La Trobe University, Dept. of Philosophy, 1984.

[Shin96] Shin-ichi M.: *Binary decision diagrams and applications for VLSI CAD*. Kluwer Academic Publishers, Massachusetts 1996.

[SiMa03] Siwiński J., Małysiak H. (red.): *Zbiór zadań z układów przełączających*. Wydawnictwo Politechniki Śląskiej. Gliwice 2003.

[StCy07] Stańczyk U., Cyran K., Pochopień B.: *Theory of Logic Circuits*. Publishers of the Silesian University of Technology, Gliwice 2007.

[Stro94] Stroustrup B.: *Język C++*, Wydawnictwa Naukowo-Techniczne, Warszawa 1994.

[Szaj96] Szajna J.: *Projektowanie układów cyfrowych z wykorzystaniem programowania logicznego*, Wydawnictwo Politechniki Zielonogórskiej, Zielona Góra 1996.

[Szpy08] Szpyrka M.: *Sieci Petriego w modelowaniu i analizie systemów współbieżnych*. Wydawnictwa Naukowo - Techniczne, Warszawa 2008

[Synt03] *Synteza układów cyfrowych*. Praca zbiorowa pod redakcją Prof. Tadeusza Łuby. Wydawnictwa Komunikacji i Łączności, Warszawa 2003.

[SuSz99] Suraj Z., Szpyrka M.: *Sieci Petriego I PN-Tools*. Wydawnictwo Wyższej Szkoły Pedagogicznej, Rzeszów 1999.

[Thel88] Thelen B.: Investigation of Algorithms for Computer-Aided Logic Design of Digital Circuits. PhD thesis, Universität Karlsruche, 1988 (in German).

[Tiur98] Tiuryn J.: *Wstęp do teorii mnogości I logiki* Uniwersytet Warszawski, Wydział Matematyki, Informatyki I Mechaniki, Warszawa 1998.

[Tkac02] Tkacz J.: *Zastosowanie naturalnego wnioskowania Gentzena w projektowaniu układów cyfrowych.* Pierwsza Krajowa Konferencja Elektroniki - KKE 2002. Wydział Elektroniki Politechniki Koszalińskiej. Koszalin 2002.

[TkAd03] Tkacz J., Adamski M: *Wykorzystanie rachunku sekwentów Gentzena do minimalizacji symbolicznej cyfrowych układów kombinacyjnych*, Materiały VI Konferencji Naukowej RUC Szczecin 2003 ss45-50.

[Tkac04] Tkacz J.: *Wykorzystanie logiki sekwentów Gentzena do projektowania kombinacyjnych układów cyfrowych.* Oficyna Wydawnicza Uniwersytetu Zielonogórskiego, s. 86-90. Zielona Góra, 2004.

[Tkac05] Tkacz J.: *Wykorzystanie logiki sekwentów Gentzena do symbolicznej analizy sieci Petriego.* Informatyka - sztuka czy rzemiosło - KNWS' 05 : materiały II konferencji naukowej. Złotniki Lubańskie 2005.

[TkAd05] Tkacz J., M. Adamski: *Wykorzystanie logiki sekwentów Gentzena do weryfikacji kombinacyjnych rekonfigurowalnych układów cyfrowych.* Reprogramowalne Układy Cyfrowe - RUC 2005 : materiały VIII krajowej konferencji naukowej. Szczecin, 2005.

[Tkac06] Tkacz J.: *Gentzen system calculus implementation for symbolic minimization of complicated logical expressions.* Discrete-Event System Design - DESDes '06 : a proceedings volume from the 3rd IFAC Workshop. International Federation of Automatic Control by University of Zielona Góra Press, 2006, s. 53-56. Rydzyna 2006.

[Tkac06a] Tkacz J.: Wykrywanie zastojów w sterownikach logicznych metodą symbolicznego wnioskowania Gentzena. Pomiary, Automatyka, Kontrola .- 2006, nr 6, wyd. spec., s. 11-13.

[Tkac07] Tkacz J.: *Kolorowanie automatowe sieci Petriego metodą wnioskowania symbolicznego.* Pomiary, Automatyka, Kontrola .- 2007, nr 5, s. 120-122.

[TkAd08] Tkacz J., Adamski M.: *Wykorzystanie komputerowego wnioskowania w projektowaniu kombinacyjnych układów sterowania*. Przegląd Telekomunikacyjny nr 6, 2008, s 728-730.

[TkAd08a] Tkacz J, Adamski M.: *Projektowanie sekwencyjnych układów cyfrowych z wykorzystaniem logiki sekwentów Gentzena.* Publikacja w przygotowaniu. 2008.

[Trac86] Traczyk W.: *Układy cyfrowe. Podstawy teoretyczne i metody syntezy.* Wydawnictwa Naukowo-Techniczne, Warszawa 1982, 1986.

[Wang60] Wang Hao: *Toward Mathematical Mathematics*. IBM. J. Res. Devel, Vol.4, 1960, No 1 pp.2-22.

[Węgr03] Węgrzyn A.: *Symboliczna analiza układów sterowania binarnego z wykorzystaniem wybranych metod analizy sterowania binarnego*. Oficyna Wydawnicza Uniwersytetu Zielonogórskiego, Zielona Góra 2003.

[WęWo97] Węgrzyn M., Wolański P., Adamski M., Monteiro J.L.: *Coloured Petri net model of application specific Logic Controller programs*. W: Proceedings of the IEEE International Symposium on Industrial Electronics - ISIE '97. Guimaraes, Portugalia, 1997 .- Piscataway, 1997 .- Vol. 1, s. 158—163

[WrMa05] Wrycza S., Marcinkowski B., Wyrzykowski K.: *Język UML w modelowaniu systemów informatycznych*, Helion, Gliwice 2005

[Zakr81] Закревский А.Д.: *Логический синтез каскадных схем.* Наука, Москва 1981.

[Zwol07] Zwoliński Mark: *Projektowanie układów cyfrowych z wykorzystaniem języka VHDL.* Wydawnictwa Komunikacji i Łączności, Warszawa 2007.

# Dodatek A – Spis przykładów

Przykład 4-1 Łączne wykorzystanie wnioskowania Gentzena i algorytmu Thelena-Mathony'ego ..... 40
Przykład 5-1 Synteza kombinacyjnego układu cyfrowego ..... 44
Przykład 5-2 Wnioskowanie symboliczne Gentzena w procesie minimalizacji funkcji silnie nieokreślonych. ..... 48
Przykład 5-3 Synteza i analiza cyfrowego układu sterującego z wykorzystaniem specyfikacji behawioralnej. ..... 51
Przykład 5-4 Weryfikacja układu kombinacyjnego względem częściowej lub pełnej specyfikacji. ..... 55
Przykład 5-5 Projekt sekwencyjnego licznika mod 4. ..... 61
Przykład 5-6 Realizacja sekwencyjnego układu cyfrowego opisanego tabelą przejść-wyjść. ..... 63
Przykład 5-7 Realizacja cyfrowego układu sekwencyjnego opisanego siecią działań. ..... 66
Przykład 5-8 Transformacja specyfikacji w języku sekwentów na język VHDL ..... 70
Przykład 5-9 Transformacja specyfikacji w języku sekwentów na język VERILOG ..... 72
Przykład 5-10 Symulacja i synteza układu rejestrowego ..... 72
Przykład 5-11 Transformacja specyfikacji w języku sekwentów na format ESPRESSO ..... 75
Przykład 6-1 Analiza grafu współbieżności ..... 77
Przykład 6-2 Kolorowanie automatowe sieci Petriego opisującej funkcjonowanie sterownika logicznego ..... 82
Przykład 6-3 Analiza binarnego systemu sterowania ..... 89

# Dodatek B – Struktura płyty CD

**[ROZPRAWA]** – *zawiera rozprawę doktorską w dwóch znanych, popularnych formatach.*

+ DoktoratJT2008.doc

+ DoktoratJT2008.pdf

**[PROTOTYP]** – *opracowany w ramach pracy prototypy aplikacji.*

[GENTZEN6] – *system normalizacji sekwentów wg reguł Gentzena.*

+ GENTZEN6.EXE

+ GENTZEN6.INI

[PROJEKT_DLA_C++BUILDER6.0]

[DIAGRAMY_UML] – *dokumentacja UML ze środowiska PowerDesigner.*

[GENTZEN_PETRI] – *badanie zastojów w sieciach Petriego.*

+ GENTZEN6_PETRI.EXE

[PROJEKT_DLA_C++BUILDER6.0]

[PRZYKŁADY] – *wybrane przykłady testowe.*

**[DODATKI]** – *uniwersyteckie programy wykorzystywane w pracy.*

[BEM] - *binary expresion manimulator.*

+ Bem_Cads.exe

[ESPRESSO] – *berkeleyowski system do minimalizacji funkcji boolowskich.*

+ ESPRESSO.zip

[PROJEKT_AHDL] – *przykładowa symulacja w środowisku AHDL*

[TEST_ GRAF] – *aplikacja do generowania losowych grafów testowych*

+ GraphTest.exe