**Uniwersytet Zielonogórski**
**Wydział Nauk Inżynieryjno-Technicznych**
**Instytut Sterowania i Systemów Informatycznych**

Praca magisterska
Kierunek: Informatyka
Specjalność: Inżynieria Systemów Informatycznych
Studia stacjonarne

# Modelowanie oraz analiza systemów produkcyjnych specyfikowanych siecią Petriego

# Modeling and analysis of production systems specified by Petri nets

inż. Dawid Konarczak

Promotor:
Prof. dr. hab. inż. Remigiusz Wiśniewski

Pracę akceptuję:

.....
(data i podpis promotora)

Zielona Góra, czerwiec 2025

## Streszczenie

Praca dyplomowa pt. *„Modelowanie oraz analiza systemów produkcyjnych specyfikowanych siecią Petriego”* poświęcona jest zagadnieniu modelowania i analizy systemów produkcyjnych przy wykorzystaniu formalizmu sieci Petriego. Sieci Petriego stanowią jedno z kluczowych narzędzi opisu procesów współbieżnych, umożliwiając graficzne i matematyczne odwzorowanie struktury i dynamiki systemów, w których zachodzi współistnienie, synchronizacja oraz konkurencja o zasoby. Ich właściwości sprawiają, że są powszechnie stosowane w modelowaniu procesów technologicznych, logistycznych i przemysłowych, gdzie istotna jest możliwość wykrywania błędów projektowych już na etapie wczesnego projektowania systemu. Celem niniejszej pracy jest opracowanie uproszczonej metody szybkiej weryfikacji modeli produkcyjnych opartych na sieciach Petriego, której zadaniem jest wczesne wykrywanie potencjalnych problemów takich jak zakleszczenia, brak żywotności, przekroczenie ograniczeń lub inne stany niepożądane. Klasyczne podejścia do analizy, takie jak analiza inwariantów czy budowa grafu osiągalności, mimo swojej dokładności, są często niewydajne w przypadku dużych i złożonych modeli. W pracy dokonano przeglądu literatury dotyczącej podstaw teoretycznych sieci Petriego, ich struktury, własności oraz zastosowań w systemach produkcyjnych. Na tej podstawie przedstawiono uproszczoną metodę weryfikacji, której skuteczność została oceniona zarówno teoretycznie, jak i poprzez badania eksperymentalne. Do eksperymentów wykorzystano środowisko PIPE, które pozwoliło na przeprowadzenie symulacji i analizę wybranych modeli. Praca zawiera także informacje o implementacji rozszerzeń umożliwiających analizę wsadową wielu modeli oraz przedstawia szczegółowe wyniki testów, zestawione w formie tabel. Wnioski wynikające z przeprowadzonych badań wskazują, że opracowane podejście może stanowić wartościowe uzupełnienie klasycznych metod weryfikacyjnych. Przedstawiona technika, dzięki swojej prostocie i niskiej złożoności obliczeniowej, znajduje zastosowanie w przypadkach, gdzie pełna analiza formalna jest nieopłacalna lub niemożliwa z uwagi na ograniczenia czasowe i sprzętowe.

**Słowa kluczowe:** sieci Petriego, systemy produkcyjne, modelowanie, weryfikacja, analiza, optymalizacja, PIPE.

# Spis treści

# Spis rysunków

# Spis tabel

# Rozdział 1

# Wstęp

## 1.1. Cel i zakres pracy

Celem niniejszej pracy jest opracowanie techniki modelowania oraz analizy systemów produkcyjnych przy wykorzystaniu sieci Petriego, ze szczególnym uwzględnieniem aspektów weryfikacji systemu na wczesnym etapie projektowym. Sieci Petriego są powszechnie stosowane w modelowaniu procesów współbieżnych i synchronicznych w systemach produkcyjnych. Kluczowym celem jest opracowanie metody weryfikacji systemu, która pozwoli na identyfikację potencjalnych błędów i nieprawidłowości na wczesnym etapie projektowania. Zakres pracy obejmuje:

1. Przegląd aktualnego stanu wiedzy w zakresie systemów produkcyjnych specyfikowanych siecią Petriego.
2. Dyskusję dotyczącą skuteczności oraz sprawności istniejących technik modelowania i analizy systemów produkcyjnych z użyciem sieci Petriego.
3. Opracowanie techniki modelowania i analizy systemów produkcyjnych specyfikowanych siecią Petriego pod kątem weryfikacji systemu na wczesnym etapie projektowym.
4. Przeprowadzenie badań eksperymentalnych w celu określenia skuteczności oraz sprawności opracowanej techniki.
5. Analizę oraz dyskusję uzyskanych wyników, opracowanie wniosków końcowych.

## 1.2. Motywacja i uzasadnienie wyboru tematu

Sieci Petriego, będące formalnym narzędziem opisu procesów współbieżnych i synchronicznych, stanowią fundament w modelowaniu systemów produkcyjnych, zwłaszcza w kontekście procesów, w których istotne jest odwzorowanie dynamiki współdzielonych zasobów. Zastosowanie sieci Petriego w systemach produkcyjnych jest kluczowe, ponieważ pozwala na dokładną analizę interakcji między poszczególnymi elementami systemu oraz umożliwia przewidywanie możliwych problemów, takich jak zakleszczenia czy brak synchronizacji. Temat pracy jest motywowany rosnącą potrzebą weryfikacji systemów na etapie ich projektowania, w szczególności w kontekście optymalizacji procesów i minimalizacji błędów na wczesnym etapie.

Istniejące metody weryfikacji i analizy, choć skuteczne, często są czasochłonne i kosztowne w przypadku złożonych systemów. Celem pracy jest opracowanie metody, która będzie łączyć prostotę, efektywność i dokładność, umożliwiając wczesną identyfikację problemów w modelach systemów produkcyjnych.

## 1.3. Krótki opis struktury pracy

Praca składa się z sześciu rozdziałów. W pierwszym rozdziale przedstawiono cel i zakres pracy, motywację oraz uzasadnienie wyboru tematu, a także ogólny opis struktury pracy. W drugim rozdziale omówiono teoretyczne podstawy sieci Petriego, ich własności oraz zastosowanie w systemach produkcyjnych. Rozdział trzeci poświęcony jest modelowaniu oraz analizie metod weryfikacji sieci Petriego, w tym omówieniu technik analitycznych oraz wyzwań związanych z ich zastosowaniem w praktyce. Czwarty rozdział opisuje opracowaną we współpracy z zespołem badawczym metodę szybkiej weryfikacji, jej mechanizm działania oraz ocenę skuteczności. W piątym rozdziale przedstawiono autorskie badania eksperymentalne, w tym opis bazy testowej, organizację badań oraz wyniki eksperymentów. W ostatnim, szóstym rozdziale zaprezentowano podsumowanie oraz wnioski końcowe z przeprowadzonych badań oraz możliwe drogi dalszego rozwoju.

# Rozdział 2

# Wprowadzenie teoretyczne

## 2.1. Podstawy sieci Petriego

### Wprowadzenie do sieci Petriego

Sieci Petriego stanowią jeden z najbardziej uznanych i powszechnie stosowanych formalizmów matematycznych wykorzystywanych do opisu, analizy oraz weryfikacji systemów złożonych, w których kluczową rolę odgrywa współbieżność, synchronizacja działań oraz współdzielenie ograniczonych zasobów [4]. Zostały zaprojektowane w latach sześćdziesiątych XX wieku przez Carla Adama Petri'ego i od tamtego czasu znalazły szerokie zastosowanie w wielu dziedzinach techniki i informatyki. Podstawową cechą, która wyróżnia sieci Petriego na tle innych narzędzi modelowania, jest ich zdolność do jednoczesnego ujęcia aspektu strukturalnego oraz dynamicznego systemu. Z jednej strony pozwalają one na przedstawienie wzajemnych zależności pomiędzy obiektami i zdarzeniami, a z drugiej umożliwiają symulację możliwych scenariuszy działania i identyfikację potencjalnych stanów niepożądanych. Sieci Petriego łączą w sobie dwie istotne cechy: formalizm matematyczny oraz intuicyjną reprezentację graficzną. Reprezentacja matematyczna pozwala na ścisłą analizę modelowanego systemu, podczas gdy reprezentacja graficzna ułatwia jego zrozumienie i interpretację, szczególnie w procesie projektowania i komunikacji zespołowej[5]. Dzięki temu formalizm ten jest jednocześnie dokładny i przystępny. Zdolność do odwzorowania współbieżności jest szczególnie ważna w systemach, w których wiele procesów zachodzi równocześnie i może wchodzić w interakcje. W sieciach Petriego

współbieżność jest modelowana w sposób naturalny, bez potrzeby wprowadzania sztucznego porządkowania zdarzeń. Dzięki temu możliwe jest wierne odtworzenie rzeczywistych zachowań systemu oraz przeprowadzenie dokładnej analizy jego dynamiki. Kolejnym ważnym aspektem jest możliwość formalnej weryfikacji różnych własności systemu, takich jak: żywotność, ograniczoność, bezpieczeństwo czy wolność od zakleszczeń. Własności te odgrywają kluczową rolę w ocenie poprawności i niezawodności projektowanych rozwiązań, zwłaszcza w przypadku systemów krytycznych, gdzie błędy mogą prowadzić do poważnych konsekwencji. Z praktycznego punktu widzenia, sieci Petriego znajdują zastosowanie w bardzo wielu dziedzinach. W informatyce służą do analizy programów współbieżnych, protokołów komunikacyjnych i systemów operacyjnych. W automatyce przemysłowej wykorzystywane są do projektowania układów sterowania i sekwencyjnych systemów logicznych[5]. W logistyce i zarządzaniu produkcją pozwalają na modelowanie przepływu materiałów, planowanie zadań oraz optymalizację wykorzystania zasobów. Podsumowując, sieci Petriego to uniwersalne narzędzie modelowania, które łączy formalizm matematyczny z przejrzystą formą graficzną. Ich wykorzystanie w procesie projektowania i analizy systemów złożonych pozwala na zwiększenie niezawodności, bezpieczeństwa oraz efektywności wdrażanych rozwiązań.

W swojej najprostszej, klasycznej formie struktura sieci Petriego definiowana jest formalnie jako trójka:

$$N = (P, T, F) \text{ [6]}$$

gdzie:

- $P$ jest niepustym, skończonym zbiorem miejsc;
- $T$ jest niepustym, skończonym zbiorem tranzycji;
- $F \subseteq (P \times T) \cup (T \times P)$ jest niepustym, skończonym zbiorem łuków;

Aby analizować dynamiczne zachowanie systemu, klasyczną sieć Petriego rozszerza się o oznaczenie początkowe (ang. *initial marking*) $M_0$. Powstaje wtedy oznakowana sieć Petriego, opisana formalnie czwórką[7]:

$$N = (P, T, F, M_0), [7]$$

gdzie:

$$M_0 : P \to \mathbb{N}.[7]$$

Każde miejsce może zawierać pewną liczbę tokenów, które reprezentują jednostki zasobów lub logiczne stany systemu. Oznaczenie początkowe określa początkowy stan systemu, przydzielając każdemu miejscu pewną liczbę tokenów. Oznaczenie w dowolnym momencie działania systemu określone jest funkcją:

$$M : P \to \mathbb{N}, \text{gdzie N jest zbiorem nieujemnych liczb całkowitych}[7]$$

Tranzycja może być odpalona, gdy wszystkie miejsca wejściowe tej tranzycji posiadają wystarczającą liczbę tokenów.

Po odpaleniu tranzycji $t_j$ zmienia się oznaczenie miejsc wejściowych i wyjściowych zgodnie z regułą:

$$M' = M + A \cdot u_j[3]$$

gdzie $M$ to aktualne oznaczenie (marking), $M'$ to oznaczenie po odpaleniu tranzycji $t_j$, $A$ to macierz incydencji sieci Petriego zdefiniowana jako:

$$A[p,t] = \begin{cases} 1 & \text{jeśli } (t,p) \in F \\ -1 & \text{jeśli } (p,t) \in F \\ 0 & \text{w przeciwnym razie} \end{cases}$$

a $u_j$ to wektor jednostkowy odpowiadający tranzycji $t_j$.[3]

Macierz incydencji $A$ ma wymiary $|P| \times |T|$ i odzwierciedla strukturę sieci – tzn. sposób połączenia miejsc z tranzycjami. Zmiana oznaczenia $M \to M'$ oznacza usunięcie znaczników z miejsc wejściowych tranzycji $t_j$ i dodanie ich do miejsc wyjściowych.

## Własności dynamiczne sieci Petriego

Sieci Petriego są szeroko analizowane pod kątem swoich właściwości dynamicznych, które pozwalają na ocenę poprawności oraz jakości działania zamodelowanego systemu. Podstawowymi własnościami są:

**Żywotność (ang. *liveness*)** Własność ta gwarantuje, że sieć będzie mogła kontynuować pracę bez końca. Tranzycja $t$ jest żywa w oznaczeniu początkowym $M_0$, jeśli spełniony jest warunek:

$$\forall M \in [M_0]\ \exists M' \in [M_0] \text{ takie, że } t \text{ jest aktywne w } M'[4]$$

Sieć N jest żywa wtedy i tylko wtedy, gdy:

$$\forall t \in T:\ t \text{ jest żywe}[4]$$

**Ograniczoność (ang. *boundedness*)** Sieć jest ograniczona, jeśli liczba tokenów w każdym miejscu jest zawsze ograniczona przez pewną stałą wartość $k$, formalnie:

$$\forall p \in P, \exists k \in \mathbb{N}, \forall M \in \mathcal{R}(M_0) : M(p) \leq k[3]$$

gdzie:

$$R(M_0) = \{M \mid M_0 \xrightarrow{*} M\}[4]$$

**Bezpieczeństwo (ang. *safeness*)** Jest to szczególny przypadek ograniczoności, gdy każde miejsce może mieć maksymalnie jeden token:

$$\forall p \in P, \forall M \in \mathcal{R}(M_0) : M(p) \leq 1[3]$$

**Wolność od zakleszczeń (ang. *deadlock-freeness*)** Sieć jest wolna od zakleszczeń, jeśli w każdym stanie osiągalnym istnieje przynajmniej jedna tranzycja możliwa do odpalenia:

$$\forall M \in \mathcal{R}(M_0), \exists t \in T : t \text{ jest aktywne w } M[3]$$

## Inwarianty w sieciach Petriego

Analiza inwariantów jest jedną z kluczowych metod badania strukturalnych właściwości sieci Petriego. Wyróżnia się dwa typy inwariantów: inwarianty miejsc (*P-inwarianty*) oraz inwarianty tranzycji (*T-inwarianty*).

**P-inwarianty[8]** Wektor $x \in \mathbb{N}^n$ jest *P-inwariantem* sieci, jeśli spełnia warunek:

$$A \cdot x = 0$$

Macierz $A$ oznacza macierz przepływu (incydencji), a $x$ odwzorowuje wagi przypisane miejscom. Inwarianty miejsc pozwalają identyfikować kombinacje miejsc, w których łączna liczba tokenów pozostaje stała – niezależnie od kolejności odpaleń tranzycji. Ich istnienie świadczy o zachowaniu zasobów i może być użyte do wykazania ograniczoności sieci.

**T-inwarianty[8]** Wektor $y \in \mathbb{N}^m$ jest *T-inwariantem* sieci, jeśli:

$$A^T \cdot y = 0$$

Taki wektor odpowiada liczbie odpaleń poszczególnych tranzycji, po których oznaczenie sieci wraca do stanu początkowego. T-inwarianty opisują możliwe cykliczne sekwencje działań, które nie zmieniają oznaczenia.

**Nośnik i minimalność** Dla inwariantu $x$ definiuje się jego *nośnik* jako zbiór miejsc, którym przypisano wartości dodatnie. Nośnik inwariantu jest *minimalny*, jeśli nie zawiera innego nośnika inwariantu poza samym sobą i zbiorem pustym. Każdy inwariant można zapisać jako liniową kombinację inwariantów o nośnikach minimalnych z nieujemnymi współczynnikami [8].

Opisane własności strukturalne stanowią fundament do dalszej analizy sieci, umożliwiając formalną weryfikację zachowania systemu oraz ocenę jego niezawodności i odporności na błędy projektowe, przez co są niezbędne dla dalszych analiz zawartych w tej pracy.

## 2.2. Zastosowanie sieci Petriego w systemach produkcyjnych

### Charakterystyka systemów produkcyjnych

Systemy produkcyjne stanowią złożone układy techniczno-organizacyjne, w których celem nadrzędnym jest przekształcanie surowców lub półproduktów w wyroby gotowe w sposób zorganizowany, efektywny i przewidywalny. Tego rodzaju systemy są obecne zarówno w produkcji dyskretnej (np. montaż podzespołów elektronicznych), jak i ciągłej (np. przemysł chemiczny czy spożywczy).

Współczesne systemy produkcyjne charakteryzują się wysokim stopniem złożoności. Często są to systemy rozproszone, w których wiele procesów zachodzi równocześnie, a ich wzajemne zależności, zależności czasowe oraz zasoby współdzielone powodują konieczność zastosowania narzędzi wspomagających projektowanie i analizę. Elementy takie jak buforowanie, kolejki, współdzielone urządzenia (np. podajniki, magazyny), oraz zależności międzyoperacyjne muszą być uwzględnione już na etapie projektowania systemu, aby uniknąć niepożądanych zjawisk, takich jak przestoje, kolizje, zakleszczenia czy niewydolność przepływu materiałowego.

W typowym systemie produkcyjnym wyróżnia się elementy takie jak:

- **Stanowiska robocze** – miejsca realizacji określonych operacji technologicznych,
- **Magazyny i bufory** – miejsca tymczasowego składowania półproduktów,
- **Transport wewnętrzny** – systemy przenośników, robotów AGV, suwnic,
- **Zasoby ludzkie lub robotyczne** – wykonujące konkretne zadania w procesie,

Zarządzanie takim systemem wymaga nie tylko kontroli sekwencji operacji, ale również uwzględnienia zależności czasowych, dostępności zasobów i możliwości występowania konfliktów. Tradycyjne diagramy przepływu lub schematy blokowe nie są w stanie uchwycić wszystkich aspektów zachowań dynamicznych systemu, zwłaszcza w przypadku współbieżności i zasobów ograniczonych.

## Powiązania modeli sieci z systemami produkcyjnymi

Sieci Petriego są doskonale przystosowane do odwzorowania rzeczywistego działania systemów produkcyjnych, dzięki swojej zdolności do modelowania zachowań współbieżnych, synchronizacji, zasobów i zdarzeń. Elementy sieci Petriego można z powodzeniem zinterpretować jako składniki systemów produkcyjnych:

- **Miejsca ($P$)** – mogą reprezentować dostępność zasobów, obecność produktu w określonym punkcie procesu (np. na stanowisku, w buforze),
- **Tranzycje ($T$)** – odwzorowują działania lub zdarzenia, takie jak rozpoczęcie bądź zakończenie operacji, przemieszczenie produktu,
- **Tokeny** – reprezentują konkretne jednostki produkcyjne (części, surowce, zamówienia), których przepływ przez system podlega analizie,
- **Łuki ($F$)** – odwzorowują relacje przyczynowo-skutkowe oraz przepływ informacji lub zasobów pomiędzy komponentami systemu.

**Przykład:** Rozważmy prosty model stanowiska montażowego, w którym produkt przemieszcza się przez trzy etapy: przygotowanie, montaż i kontrolę jakości. Można to odwzorować za pomocą trzech miejsc reprezentujących stany procesu oraz trzech tranzycji odpowiadających zakończeniu każdego z etapów. Token przemieszcza się przez te stany zgodnie z kolejnością operacyjną, odzwierciedlając fizyczny przepływ jednostki produkcyjnej przez linię.

**Współdzielenie zasobów:** Sieci Petriego umożliwiają także modelowanie współdzielonych zasobów. Przykładowo, jeśli dwa niezależne procesy korzystają z tego samego robota montażowego, można to odwzorować przez miejsce reprezentujące dostępność robota, a odpalenie każdej z konkurencyjnych operacji wymaga obecności tokena w tym miejscu. Pozwala to analizować ryzyko zakleszczeń lub przeciążeń.

Zalety stosowania sieci Petriego w systemach produkcyjnych obejmują:

- możliwość weryfikacji poprawności projektu już na etapie jego tworzenia,
- symulację przebiegu produkcji i identyfikację wąskich gardeł,
- sprawdzenie odporności systemu na nieprzewidziane zdarzenia (zakleszczenia, blokady),
- analizę obciążenia zasobów i synchronizacji operacji,
- możliwość modelowania dynamicznych zmian w strukturze procesu.

## Analiza problemów w kontekście produkcyjnym

W zastosowaniach przemysłowych szczególne znaczenie mają następujące problemy:

- **Zakleszczenia (ang. *deadlocks*)** – sytuacje, w których żadna operacja nie może być wykonana z powodu wzajemnych zależności zasobów.
- **Przeciążenie buforów (ang. *boundedness*)** – analiza ograniczoności miejsc pozwala ocenić, czy projektowana pojemność bufora jest wystarczająca.
- **Martwe stany (ang. *liveness*)** – sytuacje, w których niektóre operacje lub jednostki produkcyjne są nieskończenie długo blokowane.
- **Brak synchronizacji** – konieczność weryfikacji czy dane operacje są wykonywane w odpowiedniej kolejności.

Dzięki formalnej strukturze, sieci Petriego pozwalają na wykrycie powyższych problemów bez konieczności fizycznego wdrażania systemu. Możliwa jest analiza scenariuszy ekstremalnych i błędów projektowych jeszcze na etapie koncepcyjnym.

## Narzędzia wspierające modelowanie i analizę systemów z zastosowaniem sieci Petriego

Współcześnie, do modelowania systemów produkcyjnych przy użyciu sieci Petriego wykorzystuje się szereg narzędzi informatycznych, np:

- **PIPE (ang. *Platform Independent Petri net Editor*)** – środowisko do budowy, symulacji i analizy klasycznych sieci Petriego,
- **HIPPO-CPS** – platforma do modelowania, udostępniania i analizy sieci Petriego w kontekście systemów produkcyjnych, z naciskiem na ich weryfikację i walidację,
- **Microsoft Visio** – narzędzie ogólnego zastosowania, często wykorzystywane do tworzenia przejrzystych i estetycznych wizualizacji modeli sieci Petriego, w szczególności w kontekście prezentacji wyników analiz przypadków rzeczywistych,
- **IOPT-Tools** – zintegrowane środowisko webowe dostępne pod adresem `http://gres.uninova.pt/IOPT-Tools-V1.2/login.php`, umożliwiające modelowanie, symulację i analizę sieci typu Input-Output Place/Transition, szczególnie przydatne w projektowaniu systemów wbudowanych i sterowników cyfrowych.

W praktyce przemysłowej sieci Petriego są wykorzystywane m.in. do:

- modelowania linii montażowych i układów robotycznych,
- projektowania systemów sterowania produkcją (MES, SCADA),
- symulacji przepływu materiałowego w zakładach produkcyjnych,
- wspomagania decyzji przy planowaniu harmonogramów produkcyjnych.

# Rozdział 3

# Modelowanie i analiza sieci Petriego

Modelowanie oraz weryfikacja poprawności modeli sieci Petriego odgrywa kluczową rolę w ocenie bezpieczeństwa, niezawodności oraz wydajności projektowanych systemów współbieżnych i produkcyjnych. Poprawnie zbudowany model powinien gwarantować spełnienie określonych własności formalnych, które zapewniają stabilne i przewidywalne działanie systemu w każdych warunkach operacyjnych.

Niniejszy rozdział przedstawia metody modelowania oraz analizy systemów produkcyjnych specyfikowanych z zastosowaniem sieci Petriego. Zaprezentowana poniżej metoda modelowania została zaproponowana przez interdycyplinarny zespół kierowany przez Prof. Wiśniewskiego oraz Prof. Patalas-Maliszewską z Uniwersytetu Zielonogórskiego, w pracach którego uczestniczy autor niniejszej pracy magisterskiej. Natomiast analiza systemu produkcyjnego została omówiona szerzej. Dlatego też rozdziały 3.2 oraz 3.3 zawierają informacje odnośnie istniejących metod analizy systemów produkcyjnych specyfikowanych siecią Petriego, natomiast zaproponowaną metodę analizy przedstawiono szczegółowo w rodziale 4.

## 3.1. Modelowanie systemów produkcyjnych

Modelowanie systemów produkcyjnych z wykorzystaniem sieci Petriego odgrywa istotną rolę w projektowaniu złożonych środowisk przemysłowych. Dzięki możliwo-

ściom oferowanym przez ten formalizm – zarówno graficznym, jak i matematycznym – możliwe jest przeprowadzenie weryfikacji, symulacji i analizy dynamicznych zachowań systemu jeszcze na etapie projektowania. Autorzy pracy [1] podkreślają znaczenie interpretowanych sieci Petriego (ang. *interpreted Petri nets*) jako narzędzia umożliwiającego odwzorowanie zależności między systemem produkcyjnym a jego otoczeniem [9]. Wskazują oni[1], że zastosowana koncepcja opiera się na interpretowanych sieciach Petriego, które łączą możliwość tworzenia modeli graficznych z solidnym zapleczem analitycznym. Pozwala to nie tylko na przejrzyste odwzorowanie struktury systemu, ale także na jego formalną weryfikację i wspomaganie procesu decyzyjnego z wykorzystaniem metod analizy matematycznej.

W klasycznym ujęciu sieci Petriego miejsca ($P$) reprezentują stany systemu lub dostępność zasobów, a tranzycje ($T$) odpowiadają operacjom technologicznym lub przejściom pomiędzy stanami. Rozszerzenie tego modelu do postaci interpretowanej sieci Petriego umożliwia powiązanie tranzycji z sygnałami wejściowymi, a miejsc z sygnałami wyjściowymi, co pozwala odwzorować logikę sterowania oraz sprzężenie systemu z danymi środowiskowymi. Jak wskazują autorzy [1], w interpretowanych sieciach Petriego przyjęto zasadę, zgodnie z którą sygnały wejściowe powiązane są z tranzycjami, natomiast sygnały wyjściowe przyporządkowuje się miejscom. Takie rozwiązanie pozwala na precyzyjne modelowanie logiki reakcji systemu na zmienne warunki zewnętrzne oraz odwzorowanie jego odpowiedzi operacyjnej. Z formalnego punktu widzenia, interpretowaną sieć Petriego definiuje się jako uporządkowaną szóstkę[10, 11, 12, 1]:

$$N = (P, T, F, M_0, X, Y),$$

gdzie:

- $P$ — zbiór miejsc (ang. places),
- $T$ — zbiór tranzycji (ang. transitions),
- $F \subseteq (P \times T) \cup (T \times P)$ — relacja przepływu (ang. flow relation),
- $M_0$ — znacznik początkowy (ang. initial marking),
- $X$ — zbiór sygnałów wejściowych (ang. input signals),
- $Y$ — zbiór sygnałów wyjściowych (ang. output signals).

Przypisanie elementów rzeczywistego systemu produkcyjnego do struktury sieci Petriego pozwala na precyzyjne odwzorowanie jego działania. Miejsca mogą reprezentować etapy procesu produkcyjnego, dostępność maszyn lub oczekiwanie na decyzję. Tranzycje realizują przejścia pomiędzy etapami lub wyzwalają określone działania. Sygnały wejściowe (np. opóźnienia dostaw, koszty energii, rotacja pracowników) sterują odpaleń tranzycji, natomiast sygnały wyjściowe (np. rozpoczęcie analizy, wyświetlenie komunikatu) są przypisywane do miejsc.

W pracy [1] przedstawiono rozbudowany przykład zastosowania interpretowanej sieci Petriego do modelowania systemu wspierającego decyzję o wdrożeniu technologii przyrostowych (AM) w przedsiębiorstwie przemysłowym. W modelu uwzględniono liczne czynniki zewnętrzne, mogące wpływać na opłacalność inwestycji, takie jak wzrost kosztów energii, rotacja pracowników czy zakłócenia w dostawach. Model umożliwia analizę wpływu tych czynników na decyzję inwestycyjną poprzez wykorzystanie mechanizmu logicznego sterowania siecią. Jak zaznaczają autorzy [1], zaproponowany algorytm weryfikacji został znacząco rozwinięty względem pierwotnej wersji — usunięto wcześniejsze ograniczenia związane z wykrywaniem potencjalnie nieograniczonych miejsc, a nowa wersja umożliwia dodatkowo identyfikację wielokrotnych wystąpień w zredukowanej macierzy strukturalnej.

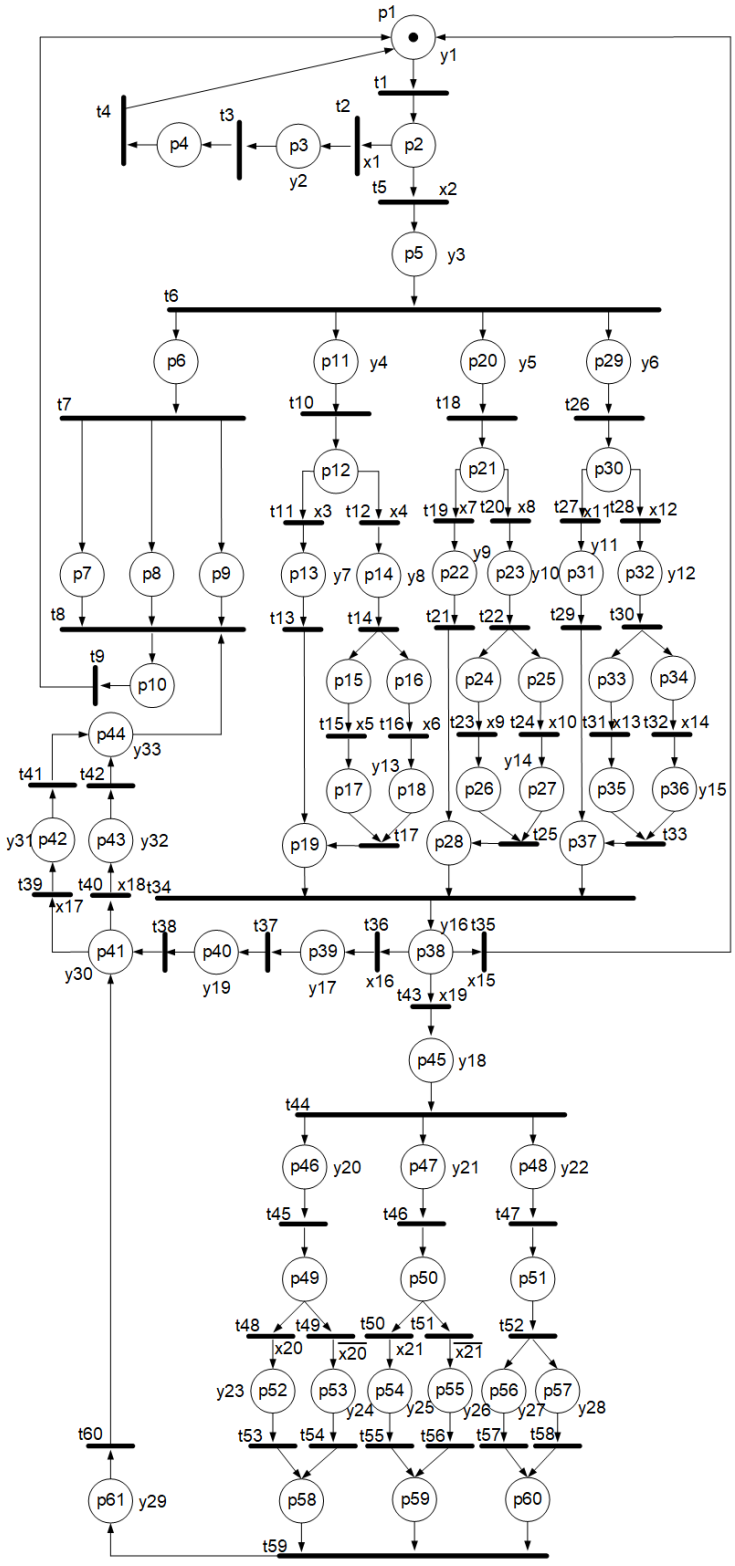
Zaprojektowana sieć zawierała 61 miejsc, 60 tranzycji oraz ponad 50 sygnałów wejściowych i wyjściowych, co umożliwiło wierne odwzorowanie złożonego procesu decyzyjnego. Do opracowania modelu wykorzystano środowisko PIPE, a jego graficzną postać przygotowano w programie Visio.

Na rysunku 3.1 zaprezentowano przykładowy fragment modelu, w którym zilustrowano sposób powiązania tranzycji z sygnałami wejściowymi, odzwierciedlającymi zmienne warunki zewnętrzne (np. wzrost kosztów energii). Z kolei odpowiednie miejsca odpowiadają decyzjom i działaniom wynikającym z analizy danych.

W celu pełniejszego odwzorowania działania modelu, w pracy zawarto również tabele opisujące przypisanie sygnałów wejściowych do tranzycji oraz sygnałów wyjściowych do miejsc. Są one przedstawione w tabelach 3.1 i 3.2.

Model ten został przetestowany na danych rzeczywistych pochodzących z przedsiębiorstwa z branży metalowej. Wynik analizy wskazywał na konieczność rozpoczęcia działań związanych z wdrożeniem technologii AM. Jak wskazano w pracy [1],

opracowane podejście do modelowania i analizy zostało zilustrowane na przykładzie rzeczywistego systemu decyzyjnego, który wspiera proces wdrożenia technologii przyrostowych w środowisku przemysłowym.




**Rysunek 3.1.** Przykładowa sieć produkcyjna [1].

| Place | Output | Task | Place | Output | Task |
|---|---|---|---|---|---|
| p1 | y1 | calculate: `result=xzyr` | p45 | y18 | Make additional analysis |
| p3 | y2 | print: current technology is optimal | p40 | y19 | Get result |
| p5 | y3 | print: more research needed | p46 | y20 | the possibility of using a different material |
| p11 | y4 | disruption analysis: material deliveries | p47 | y21 | reduction of employment |
| p20 | y5 | disruption analysis: high employee turnover | p48 | y22 | reduction of energy consumption costs |
| p29 | y6 | disruption analysis: increasing costs of energy consumption | p52 | y23 | GET: 0.3 |
| p13 | y7 | GET: 0 | p53 | y24 | GET:0 |
| p14 | y8 | GET: 0.25 | p54 | y25 | GET: 0.3 |
| p22 | y9 | GET: 0 | p55 | y26 | GET:0 |
| p23 | y10 | GET: 0.25 | p56 | y27 | GET: 0.3 |
| p31 | y11 | GET: 0 | p57 | y28 | GET: 0 |
| p32 | y12 | GET: 0.25 | p61 | y29 | ADD ALL OF THE RESULTS |
| p18 | y13 | DISPLAY: SIGNIFICANT INTERFERENCE LEVEL | p41 | y30 | Disruption result + additional analysis result |
| p27 | y14 | DISPLAY: SIGNIFICANT INTERFERENCE LEVEL | p42 | y31 | Print recommendation: tests and analysis |
| p36 | y15 | DISPLAY: SIGNIFICANT INTERFERENCE LEVEL | p43 | y32 | Print recommendation: analysis of other technology. |
| p38 | y16 | Result=S+E+A | p44 | y33 | Get recommendation |
| p39 | y17 | print: optimal | | | |

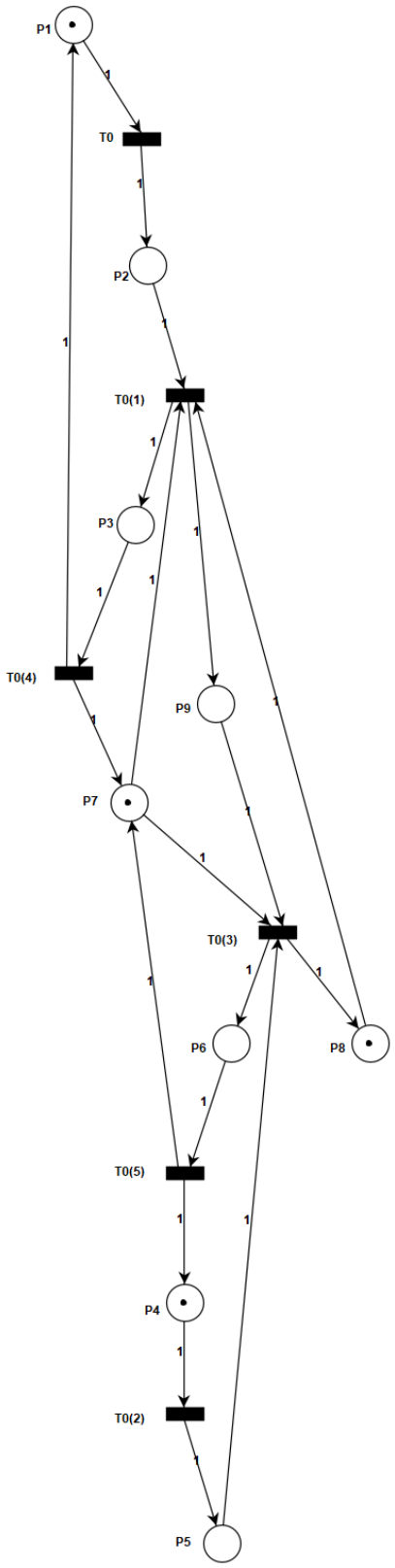
**Tabela 3.1.** Tabela sygnałów wyjściowych [1] dla przykładowej sieci produkcyjnej.

| Trans. | Input | Condition | Trans. | Input | Condition |
|---|---|---|---|---|---|
| t2 | x1 | RESULT=1 | t31 | x13 | NO MESSAGE |
| t5 | x2 | 4>RESULT>1 | t32 | x14 | DISPLAYED MESSAGE |
| t11 | x3 | A<20% | t35 | x15 | DISRUPTION RESULT>0.5 |
| t12 | x4 | A>20% | t36 | x16 | DISRUPTION RESULT<0.25 |
| t15 | x5 | NO MESSAGE | t39 | x17 | 0<DISRUPTION ANALYSIS<0.25 |
| t16 | x6 | DISPLAYED MESSAGE | t40 | x18 | 0.3 <DISRUPTION ANALYSIS <0.9 |
| t19 | x7 | S<1.2 | t43 | x19 | 0.25<DISRUPTION RESULT <0.5 |
| t20 | x8 | S>1.2 | t48 | x20 | DECISION: YES |
| t23 | x9 | NO MESSAGE | t49 | $\overline{x20}$ | DECISION: NO |
| t24 | x10 | DISPLAYED MESSAGE | t50 | x21 | DECISION: YES |
| t27 | x11 | E<190% | t51 | $\overline{x21}$ | DECISION: NO |
| t28 | x12 | E>190% | | | |

**Tabela 3.2.** Tabela sygnałów wejściowych [1] dla przykładowej sieci produkcyjnej.

Zastosowanie interpretowanej sieci Petriego w tym kontekście pozwoliło nie tylko odwzorować zachowanie systemu produkcyjnego, ale również wykorzystać metody weryfikacji formalnej (w tym analizę ograniczoności miejsc oraz wykrywanie niepokrytych miejsc) do identyfikacji potencjalnych błędów projektowych, takich jak źle zsynchronizowane ścieżki czy ukryte konflikty logiczne. Stanowi to przykład efektywnego połączenia narzędzi formalnych z praktycznym modelowaniem złożonych systemów przemysłowych, zwłaszcza w warunkach zakłóceń i niepewności otoczenia, takich jak kryzysy logistyczne, wahania cen energii czy wysoka rotacja pracowników.

Analiza opisana w [1] jednoznacznie wskazuje, że interpretowane sieci Petriego mogą stanowić realne i praktyczne wsparcie w procesie podejmowania decyzji inwestycyjnych w przedsiębiorstwach produkcyjnych. Ich uniwersalność i formalizacja matematyczna umożliwiają nie tylko odwzorowanie złożonej logiki procesów technologicznych i decyzyjnych, ale także weryfikację poprawności zaprojektowanego systemu już na etapie modelowania. Dzięki temu możliwe jest wcześniejsze wykrycie błędów, które mogłyby skutkować kosztownymi problemami operacyjnymi po wdrożeniu.




**Rysunek 3.2.** Sieć `multi_robot.pnh` [2]

W kontekście systemów cyber-fizycznych (CPS) istotnym zagadnieniem jest poprawne modelowanie oraz analiza bezpieczeństwa współbieżnych działań wielu jednostek wykonawczych [13]. Klasycznym przykładem tego typu systemu jest modelowanie zespołu robotów mobilnych współdzielących ograniczone zasoby, takie jak stanowiska robocze czy obszary manipulacyjne. Jednym z powszechnie przywoływanych modeli edukacyjnych jest sieć Petriego określana jako `multi-robot CPS`, przedstawiona m.in. w [14, 15].

Model `multi-robot CPS` odwzorowuje działanie dwóch lub więcej robotów poruszających się w przestrzeni i korzystających z ograniczonych wspólnych zasobów (np. przejść, stanowisk roboczych, narzędzi). Każdy robot realizuje własną sekwencję operacji, jednak dostęp do niektórych miejsc (obszarów zasobów) wymaga koordynacji, aby uniknąć kolizji i zakleszczeń.

W strukturze sieci wyróżnia się następujące komponenty:

- **Miejsca** reprezentujące stany jednostek (np. przebywanie w określonym obszarze) oraz dostępność zasobów (np. wolny/zajęty obszar przejścia),
- **Tranzycje** odpowiadające ruchowi robota lub zmianie stanu zasobu,
- **Tokeny** symbolizujące obecność robota w danym stanie lub dostępność zasobu.

Przykładowy schemat działania obejmuje sytuację, w której dwa roboty poruszają się po wyznaczonej trajektorii i napotykają wspólny obszar, który może być używany tylko przez jednego robota w danej chwili. Modelowanie tego scenariusza za pomocą sieci Petriego umożliwia formalną analizę potencjalnych błędów systemu, takich jak:

- **Deadlocks** — sytuacja, w której żaden z robotów nie może kontynuować ruchu,
- **Liveness** — operacja jednego z robotów zostaje na stałe zablokowana,
- **Safeness** — więcej niż jeden robot jednocześnie znajduje się w obszarze wymagającym wyłączności.

W pracy [15] zwrócono uwagę na znaczenie analizy własności bezpieczeństwa w kontekście systemów wielorobotowych. Opracowana metodologia umożliwia wykrywanie miejsc niebezpiecznych poprzez analizę przestrzeni stanów sieci lub wykorzystanie inwariantów miejscowych (*P-invariants*). Przykładowo, w modelu `multi-robot CPS` miejsca reprezentujące wspólne zasoby powinny być objęte ograniczeniami zapewniającymi, że nigdy nie znajdzie się tam więcej niż jeden token jednocześnie.

Dodatkowo w [14] podkreślono, że sieci Petriego stanowią nie tylko narzędzie opisu, ale również platformę do automatycznej syntezy nadzorców zapewniających poprawność działania systemu. Dzięki formalizmowi sieci możliwe jest nie tylko przewidywanie błędów projektowych, ale również automatyczne generowanie strategii naprawczych w czasie rzeczywistym.

## 3.2. Metody weryfikacji poprawności sieci

Przypomnijmy że podstawowymi własnościami podlegającymi weryfikacji w sieciach Petriego są: żywotność (ang. *liveness*), ograniczoność (ang. *boundedness*), bezpieczeństwo (ang. *safeness*) oraz wolność od zakleszczeń (ang. *deadlock-freeness*).

Żywotność (ang. *liveness*) odnosi się do zdolności sieci do kontynuowania działania w nieskończoności, przy założeniu nieograniczonego czasu i zasobów. Mówiąc ściślej, sieć jest żywa, jeżeli w każdym osiągalnym oznaczeniu możliwe jest w przyszłości odpalenie dowolnej tranzycji. Formalnie, tranzycja $t$ jest żywa w oznaczeniu początkowym $M_0$, jeżeli:

$$\forall M \in [M_0]\ \exists M' \in [M_0] \text{ takie, że } t \text{ jest aktywne w } M'[4]$$

Sieć N jest żywa wtedy i tylko wtedy, gdy:

$$\forall t \in T:\ t \text{ jest żywe}[4]$$

Gwarancja żywotności jest szczególnie istotna w systemach produkcyjnych i sterowania, w których pewne operacje muszą być zawsze możliwe do wykonania (np. zwolnienie bufora, obsługa alarmu).

Ograniczoność (ang. *boundedness*) sieci Petriego oznacza, że liczba tokenów w każdym miejscu jest ograniczona przez pewną stałą wartość. Formalnie sieć jest $k$-ograniczona, jeżeli:

$$\forall p \in P, \exists k \in \mathbb{N}, \forall M \in \mathcal{R}(M_0) : M(p) \leq k[3]$$

Przypadek szczególny ograniczoności występuje, gdy $k = 1$, co prowadzi do własności określanej jako bezpieczeństwo (ang. *safeness*). W sieci bezpiecznej każde miejsce może w danym momencie zawierać najwyżej jeden token. Bezpieczeństwo jest istotne w kontekście systemów, gdzie obecność więcej niż jednej jednostki w określonym stanie (np. na stanowisku roboczym, w przejściu, w zasobie krytycznym) jest niedopuszczalna.

Wolność od zakleszczeń (ang. *deadlock-freeness*) to własność gwarantująca, że system nigdy nie osiągnie stanu, w którym żadna tranzycja nie może zostać odpalona. Formalnie:

$$\forall M \in \mathcal{R}(M_0), \exists t \in T : t \text{ jest aktywne w } M[3]$$

Zakleszczenie jest sytuacją wysoce niepożądaną w systemach przemysłowych i informatycznych, ponieważ oznacza całkowite zatrzymanie procesów. Detekcja i eliminacja potencjalnych deadlocków jest zatem jednym z kluczowych celów analizy poprawności.

Weryfikacja wymienionych własności może być realizowana na różne sposoby, w zależności od rodzaju i rozmiaru analizowanej sieci. W przypadku niewielkich sieci często stosuje się bezpośrednie badanie przestrzeni stanów (ang. *reachability graph*), które pozwala na pełną analizę wszystkich możliwych przebiegów systemu. Dla większych i bardziej złożonych modeli preferowane są metody oparte na inwariantach miejsc i tranzycji, które umożliwiają dowodzenie własności bez konieczności eksploracji całej przestrzeni osiągalnych oznaczeń.

W kontekście żywotności, istotnym aspektem analizy jest sprawdzenie, czy istnieje oznaczenie, w którym dana tranzycja jest odpalalna oraz czy sieć jest odporna na sytuacje, w których pewne akcje stają się trwale zablokowane. Natomiast ograniczoność i bezpieczeństwo są weryfikowane poprzez analizę potencjalnego wzrostu

liczby tokenów w miejscach, co jest ściśle związane z analizą inwariantów miejscowych.

W przypadku systemów o wysokich wymaganiach niezawodnościowych, takich jak systemy produkcyjne klasy przemysłowej czy systemy cyber-fizyczne (CPS), każda z wymienionych własności musi być rygorystycznie sprawdzona. Ich niespełnienie może prowadzić nie tylko do degradacji wydajności systemu, ale również do poważnych zagrożeń bezpieczeństwa operacyjnego.

Ostatecznie, metody weryfikacji poprawności sieci Petriego stanowią niezbędne narzędzie w procesie walidacji modeli systemów współbieżnych, umożliwiając wykrywanie i eliminację błędów projektowych na wczesnym etapie rozwoju systemu. Dalsze sekcje pracy omawiają szczegółowo wybrane techniki analityczne oraz porównują ich efektywność i ograniczenia w praktyce [16].

W procesie analizy poprawności i własności modeli sieci Petriego stosuje się różne techniki analityczne, których celem jest formalna weryfikacja określonych cech systemu. W zależności od charakteru modelu, jego złożoności oraz pożądanej dokładności analizy, stosowane są zarówno techniki strukturalne (oparte na badaniu właściwości samej sieci), jak i techniki behawioralne (oparte na eksploracji przestrzeni stanów). Jedną z fundamentalnych metod strukturalnych jest analiza inwariantów miejscowych (*P-invariant*) oraz inwariantów tranzycyjnych (*T-invariants*)[5]. Formalnie, dla macierzy incydencji $A$ oraz wektora $x$:

$$A \cdot x = 0 \text{ [6]}$$

Inwarianty miejscowe pozwalają dowodzić ograniczoności sieci, wykrywać miejsca, które zawsze zachowują określoną liczbę tokenów, oraz sprawdzać warunki bezpieczeństwa. Metoda wyznaczania inwariantów może opierać się na rozwiązaniu układów równań liniowych lub metodach optymalizacyjnych.

Alternatywną i bardziej bezpośrednią techniką analityczną jest budowa drzewa osiągalności (ang. *reachability tree*) [6]. Drzewo osiągalności reprezentuje wszystkie możliwe oznaczenia sieci, jakie mogą zostać osiągnięte z oznaczenia początkowego poprzez sekwencyjne odpalanie tranzycji. Formalnie, każdy wierzchołek drzewa odpowiada oznaczeniu $M$, a każda krawędź — odpalanemu przejściu. Budowa drzewa

osiągalności pozwala w pełni przeanalizować dynamikę systemu [6], w tym:

- wykryć stany martwe (dead states),
- zidentyfikować potencjalne zakleszczenia,
- sprawdzić ograniczoność miejsc,
- ocenić żywotność tranzycji.

Jednakże technika ta cierpi na poważne ograniczenie znane jako problem eksplozji przestrzeni stanów (ang. *state space explosion*) [6]. W przypadku dużych lub nieskończenie rozwijających się sieci, drzewo osiągalności może być nieskończone lub niepraktyczne do analizy. Dlatego w praktyce stosuje się uproszczenia lub łączy analizę drzewa z analizą strukturalną.

W kontekście analizy zachowania sieci Petriego stosuje się także podejście behawioralne. Analiza behawioralna polega na badaniu możliwych sekwencji odpaleń tranzycji oraz zmian oznaczeń sieci w czasie. W przeciwieństwie do metod strukturalnych, które opierają się na właściwościach statycznych sieci (takich jak jej topologia czy inwarianty), analiza behawioralna uwzględnia rzeczywiste scenariusze działania systemu.

Do głównych zalet analizy behawioralnej należą:

- możliwość pełnej detekcji zakleszczeń,
- możliwość analizy konkretnych sekwencji zdarzeń,
- uwzględnianie dynamicznych warunków sterowania.

Jednak analiza behawioralna posiada także istotne wady:

- wysokie wymagania obliczeniowe dla dużych sieci,
- trudności w analizie sieci nieskończonych lub o zmiennej strukturze,
- brak bezpośredniego powiązania z właściwościami strukturalnymi sieci.

W praktyce, skuteczna analiza modeli systemów produkcyjnych i współbieżnych wymaga zastosowania kombinacji metod strukturalnych i behawioralnych. Wstępna analiza strukturalna pozwala szybko wykryć oczywiste błędy konstrukcyjne (np. brak ograniczoności, naruszenie bezpieczeństwa), natomiast analiza behawioralna umożliwia pełną ocenę zachowania systemu w konkretnych scenariuszach operacyjnych.

Zastosowanie odpowiednich technik analitycznych, dostosowanych do charakteru badanego systemu oraz jego krytyczności operacyjnej, jest kluczowe dla zapewnienia wysokiej jakości projektowanych rozwiązań przemysłowych i cyber-fizycznych.

## 3.3. Podsumowanie analizy metod weryfikacji

Weryfikacja poprawności modeli sieci Petriego jest kluczowym elementem procesu projektowania systemów współbieżnych, produkcyjnych i cyber-fizycznych. Stosowane w tym celu techniki analityczne można podzielić na dwie główne kategorie: metody strukturalne oraz metody behawioralne. Każda z tych grup oferuje istotne zalety, jednak w praktyce również wiąże się z ograniczeniami, które należy brać pod uwagę przy doborze narzędzi analitycznych.

Metody strukturalne, takie jak analiza inwariantów miejscowych i tranzycyjnych czy wyznaczanie ograniczeń logicznych bez generowania przestrzeni stanów, charakteryzują się wysoką efektywnością obliczeniową. Umożliwiają one szybkie wykrycie podstawowych błędów projektowych, takich jak brak ograniczoności czy naruszenie bezpieczeństwa miejsc. Co istotne, techniki strukturalne pozwalają na analizę bardzo dużych sieci, których pełna eksploracja przestrzeni stanów byłaby niemożliwa lub niepraktyczna.

Główne ograniczenie metod strukturalnych polega jednak na ich niepełności. Analiza strukturalna dostarcza tylko informacji o pewnych aspektach sieci — przede wszystkim tych związanych ze stałymi własnościami matematycznymi, takimi jak suma tokenów lub równowaga przepływów. Nie pozwala ona natomiast na pełne odwzorowanie dynamiki zachowania systemu, szczególnie w sytuacjach, gdy właściwości zachowania wynikają z sekwencji zdarzeń, a nie wyłącznie z lokalnych własności miejsc i tranzycji. W konsekwencji metoda ta może nie wykryć subtelnych błędów,

takich jak zakleszczenia wynikające z konkretnego przebiegu wykonania. Z kolei metody behawioralne, oparte na analizie drzewa osiągalności lub grafu osiągalności, oferują pełny wgląd w dynamiczne zachowanie sieci. Pozwalają one wykryć martwe stany, zakleszczenia, brak żywotności oraz ocenić wszystkie możliwe scenariusze pracy systemu. Dzięki temu analiza behawioralna jest narzędziem niezastąpionym w procesie formalnej weryfikacji poprawności modeli, zwłaszcza w systemach krytycznych. Podstawową wadą podejścia behawioralnego jest jednak problem eksplozji przestrzeni stanów (ang. *state space explosion*). Nawet relatywnie proste modele mogą generować ogromne grafy osiągalności, których analiza staje się niepraktyczna lub wręcz niemożliwa. Problem ten narasta w przypadku sieci dynamicznych, nieskończonych lub posiadających duże liczby miejsc i tokenów. Dodatkowo, generowanie i analiza pełnej przestrzeni stanów są procesami bardzo wymagającymi obliczeniowo i pamięciowo, co ogranicza ich stosowalność w dużych modelach. W praktyce inżynierskiej, skuteczna analiza modeli sieci Petriego polega na łączeniu obu podejść: wstępne badanie strukturalne pozwala na szybkie wychwycenie oczywistych błędów projektowych i ograniczenie liczby potencjalnych problemów, a następnie selektywna analiza behawioralna umożliwia dogłębną weryfikację dynamiki działania systemu.

Podsumowując, żadna z metod weryfikacji nie jest wystarczająca samodzielnie dla wszystkich typów sieci i zastosowań. Dobór techniki analitycznej powinien być dostosowany do charakteru modelowanego systemu, jego wielkości, wymagań bezpieczeństwa oraz dostępnych zasobów obliczeniowych. Tylko komplementarne wykorzystanie metod strukturalnych i behawioralnych pozwala na uzyskanie pełnej i wiarygodnej oceny poprawności modeli sieci Petriego stosowanych w rzeczywistych systemach przemysłowych i cyber-fizycznych.

# Rozdział 4

# Proponowana metoda szybkiej weryfikacji

Niniejszy rozdział przedstawia autorską metodę szybkiej weryfikacji systemu produkcyjnego specyfikowanego z zastosowaniem sieci Petriego. Algorytm został zaproponowany przez prof. Wiśniewskiego oraz dr. Wojnakowskiego, we współpracy z zespołem prof. Patalas-Maliszewskiej, a autor pracy współuczestniczył w pracach związanych z eksperymentalną weryfikacją skuteczności i sprawności opracowanej metody.

Kluczowym czynnikiem w projektowaniu systemów specyfikowanych siecią Petriego jest szybkość weryfikacji modelu, zwłaszcza w kontekście dynamicznie rozwijających się systemów produkcyjnych, cyber-fizycznych oraz osadzonych [13]. Klasyczne podejścia, choć matematycznie ścisłe i pełne, nie są skalowalne dla modeli, których liczba miejsc i tranzycji przekracza kilkaset lub kilka tysięcy. Stąd też konieczność opracowania szybszych metod walidacji.

## 4.1. Opis metody

Nowatorski algorytm służący do wykrywania niepokrytych tranzycji w systemach współbieżnych modelowanych za pomocą sieci Petriego przedstawiono w pracy[3]. Główną zaletą tej metody jest jej złożoność obliczeniowa – algorytm działa w czasie wielomianowym, a same obliczenia są realizowane z użyciem operacji macierzowych o złożoności sześciennej, co zostaje formalnie uzasadnione w dalszej części publikacji.

Proponowana metoda opiera się na analizie macierzy incydencji danego systemu i przebiega w dwóch głównych etapach. W pierwszym kroku dokonuje się przekształcenia macierzy incydencji do postaci schodkowej zredukowanej (RREF), przy użyciu podstawowych operacji algebraicznych takich jak dodawanie, odejmowanie, mnożenie czy dzielenie. Po uzyskaniu zredukowanej macierzy, druga faza algorytmu polega na identyfikacji niepokrytych tranzycji w sieci.

Omawiane podejście znacząco różni się od tradycyjnych metod, które koncentrują się na pełnym pokryciu tranzycji przez t-inwarianty. W tym przypadku algorytm wyszukuje te tranzycje, które nie mogą być elementem żadnego poprawnego t-inwariantu. Zgodnie z definicją, t-inwariant jest wektorem spełniającym równanie $A^T y = \vec{0}$ [7]. Algorytm wykrywa naruszenia tego warunku, analizując wiersze (miejsca) zawierające wartości nieujemne i identyfikując kolumny (tranzycje), które nie mają kompensujących wartości ujemnych. Jeśli dla danej tranzycji brak jest takiego równoważenia, nie może ona spełnić warunku tworzenia poprawnego t-inwariantu.

Dodatkowo, w celu zwiększenia skuteczności działania i skrócenia czasu analizy, uwzględniane są właściwości wynikające z analizy postaci RREF, takie jak np. pozycja i występowanie wiodących jedynek.

## 4.2. Mechanizm działania i szczątkowe szczegóły implementacyjne

Algorytm wykrywania niepokrytych tranzycji w sieci Petriego można opisać w następujących krokach:

1. **Inicjalizacja:**
   - Wczytywana jest macierz incydencji $A_{|T|\times|P|}$ dla analizowanego modelu sieci Petriego.
   - Tworzony jest pusty zbiór `uncoveredTransitions`, który będzie zawierał indeksy tranzycji uznanych za niepokryte.
2. **Transpozycja macierzy incydencji:**
   - Macierz $A$ jest transponowana do postaci $A^T$ o wymiarach $|P| \times |T|$.
3. **Redukcja macierzy do postaci schodkowej (RREF):**
   - Dla każdego wiersza $i$ poszukiwany jest pierwszy niezerowy element (tzw. *leading one*).
   - W razie potrzeby wykonywana jest zamiana wierszy, aby ustawić wiodącą jedynkę na właściwej pozycji.
   - Wiersz jest dzielony przez wartość tej jedynki w celu normalizacji.
   - Następnie przeprowadzana jest eliminacja Gaussa w pozostałych wierszach poprzez odpowiednie kombinacje liniowe, aby wyeliminować wartości w tej samej kolumnie.
4. **Identyfikacja niepokrytych tranzycji:**
   - Analizowane są wiersze zawierające wyłącznie nieujemne wartości.
   - Dla każdego takiego wiersza wybierana jest pierwsza niezerowa wartość.
   - Sprawdzane są kolejne kolumny w zredukowanej macierzy – jeśli dla wybranej kolumny wszystkie odpowiadające wiersze również mają wartości nieujemne (brak kompensujących wartości ujemnych), dana tranzycja jest uznawana za niepokrytą i dodawana do zbioru `uncoveredTransitions`.

5. **Zakończenie działania i zwrócenie wyników:**

    - Jeżeli zbiór `uncoveredTransitions` nie jest pusty, algorytm wypisuje komunikat i zwraca jego zawartość.
    - W przeciwnym przypadku zwracany jest zbiór pusty oraz informacja o braku niepokrytych tranzycji.

Prototyp algorytmu został zaimplementowany w języku Java. W testach praktycznych niektóre sieci o rozmiarze rzędu kilkuset miejsc były analizowane w czasie krótszym niż 160 milisekund, co stanowiło redukcję czasu analizy o rząd wielkości (przy większości sieci) w porównaniu do klasycznych podejść.

## 4.3. Ocena skuteczności i efektywności proponowanej metody (porównanie z metodą klasyczną)

Skuteczność oraz sprawność opracowanego algorytmu zweryfikowano eksperymentalnie. Wyniki badań zestawiono i porównano z rezultatami uzyskanymi przez klasyczną metodę analizy (Martinez-Silva), bazującą na pełnym rozwiązaniu układów równań macierzowych związanych z inwariantami miejsc.

Metoda Martinez-Silva charakteryzuje się złożonością obliczeniową rzędu $O(n^3)$, gdzie $n$ oznacza liczbę miejsc lub tranzycji w analizowanej sieci. Choć podejście to zapewnia formalną poprawność wyników, jego zastosowanie w przypadku dużych modeli sieci Petriego staje się czasochłonne i niepraktyczne ze względu na gwałtowny wzrost czasu obliczeń wraz ze wzrostem rozmiaru modelu.

Z kolei opracowana metoda szybkiej weryfikacji charakteryzuje się złożonością obliczeniową rzędu $O(|P|^2 \cdot |T|)$, co zostało wykazane i omówione w pracy [3]. Oznacza to, że algorytm działa w czasie wielomianowym względem rozmiaru sieci i może być efektywnie stosowany nawet w przypadku modeli zawierających setki miejsc i tranzycji.

W trakcie przeprowadzonych testów eksperymentalnych zaobserwowano następujące zależności:

- dla modeli zawierających kilkaset miejsc i tranzycji, takich jak

*decision_making_system_v2*, pełna analiza metodą Martinez-Silva trwała ponad 800 milisekund [3],

- w tym samym przypadku metoda szybkiej weryfikacji zwracała wynik w czasie krótszym niż 15 milisekund,
- dla bardziej złożonych modeli, takich jak *photovoltaics_single_module* (139 miejsc, 162 tranzycje), metoda Martinez-Silva nie była w stanie zwrócić wyniku w czasie poniżej jednej godziny, co zostało oznaczone jako przekroczenie limitu czasowego (*timeout*) [3].
- W tym samym przypadku metoda szybkiej weryfikacji podała wynik w czasie krótszym niż 70 milisekund

Choć metoda szybkiej weryfikacji nie gwarantuje wykrycia wszystkich potencjalnych problemów w sieci, w tym subtelnych zależności występujących w rzadko aktywowanych fragmentach modelu, to jednak w praktyce inżynierskiej okazuje się wystarczająco skuteczna. Jej największym atutem jest możliwość szybkiego wykrywania typowych błędów projektowych, takich jak niepokryte tranzycje, problemy z ograniczonością miejsc czy potencjalne pułapki synchronizacyjne.

Co więcej, niska złożoność obliczeniowa metody umożliwia jej dynamiczne zastosowanie w procesie iteracyjnego projektowania – model może być weryfikowany wielokrotnie w miarę jego rozwijania, co znacząco redukuje ryzyko kumulacji błędów i pozwala na ich eliminację na wczesnym etapie cyklu projektowego.

Podsumowując, metoda szybkiej weryfikacji opracowana przez zespół badawczy w ramach którego opracowane i przedstawione zostały badania efektywności algorytmu [3], stanowi skalowalne, wydajne i praktyczne narzędzie wspomagające analizę dużych modeli sieci Petriego, skutecznie łącząc ograniczony czas analizy z wysoką jakością wyników.

# Rozdział 5

# Autorskie badania eksperymentalne (porównanie z narzędziem PIPE)

Kolejnym krokiem badań eksperymentalnych było porównanie opracowanej metody w celu praktycznego potwierdzenia skuteczności oraz efektywności opracowanej metody szybkiej weryfikacji. Ich celem było nie tylko porównanie wydajności względem klasycznych podejść (w szczególności metody Martinez-Silva), ale również weryfikacja przydatności algorytmu w analizie rzeczywistych i syntetycznych modeli systemów produkcyjnych oraz cyber-fizycznych. W eksperymentach wykorzystano różnorodną bazę modeli, środowisko PIPE oraz autorskie narzędzia umożliwiające wsadową analizę wielu przypadków testowych.

Rozdział ten przedstawia szczegółowy opis użytej bazy testowej, organizacji procesu badawczego, zastosowanej procedury eksperymentalnej, jak również modyfikacji środowiska narzędziowego i prezentacji uzyskanych wyników. Szczególny nacisk położono na analizę porównawczą czasów działania oraz trafności wykrywania błędów, co pozwoliło na kompleksową ocenę praktycznych walorów opracowanego rozwiązania.

## 5.1. Opis bazy testowej

W celu przeprowadzenia badań metody szybkiej weryfikacji opracowano zróżnicowaną bazę testową modeli sieci Petriego. Zbiór ten obejmuje zarówno sieci syntetyczne, jak i modele oparte na rzeczywistych systemach produkcyjnych i systemach cyber-fizycznych, które zostały zaprojektowane, zebrane lub zmodyfikowane w ramach współpracy zespołu badawczego.

Baza testowa została podzielona na trzy główne grupy:

1. **Modele literaturowe** – pochodzące z publikacji naukowych, w tym m.in. interpretowane sieci Petriego opisane w [1] oraz sieci użyte w [3]. Modele te reprezentują rzeczywiste systemy decyzyjne i produkcyjne o średniej i dużej złożoności strukturalnej.

2. **Modele przemysłowe** – opracowane na podstawie danych reprezentujących rzeczywiste scenariusze przemysłowe (np. zakłady branży metalowej i elektrotechnicznej). Modele te odzwierciedlają autentyczne scenariusze związane z zarządzaniem zasobami, reakcją na zakłócenia oraz planowaniem produkcji.

Łącznie baza testowa zawierała:

- **386 modeli testowych**, obejmujących różnorodne systemy oparte na sieciach Petriego oraz modele cyber-fizyczne (CPS),
- rozmiary sieci: od **7 do 200 miejsc** i od **6 do 170 tranzycji**,
- zróżnicowaną strukturę topologiczną: sieci acykliczne, cykliczne, z miejscami buforującymi, synchronizującymi i decyzyjnymi.

Dzięki tak dobranej bazie testowej możliwe było nie tylko przeprowadzenie pomiarów wydajności i skuteczności samego algorytmu szybkiej weryfikacji, ale również porównanie jego działania w kontekście różnych klas modeli – od prostych systemów logistycznych, przez złożone linie montażowe, po interpretowane sieci decyzji inwestycyjnych i modele agentowe.

Baza testowa została przetestowana zarówno w środowisku PIPE, jak i w ramach zaimplementowanego narzędzia analitycznego umożliwiającego przetwarzanie wsadowe (bulk analysis) modeli w formacie PNML.

## 5.2. Organizacja badań

Badania eksperymentalne zostały przeprowadzone w warunkach kontrolowanych, przy użyciu zunifikowanego środowiska sprzętowego oraz spójnych procedur testowych. Ich celem była jednoczesna ocena skuteczności (trafności wykrywania niepokrytych tranzycji) oraz efektywności (czasu wykonania analizy) opracowanego algorytmu w odniesieniu do klasycznych metod analizy sieci Petriego.

Eksperymenty przeprowadzono w ramach prac zespołu badawczego, przy współudziale autorów algorytmu. Proces badawczy obejmował przygotowanie środowiska analitycznego, konwersję modeli do odpowiedniego formatu wejściowego oraz realizację pomiarów w trybie wsadowym. Analizą wyników i interpretacją rezultatów zajmował się zespół badaczy odpowiedzialny za implementację i ocenę metody.

Wszystkie testy zostały uruchomione na jednolitym zestawie sprzętowym, obejmującym procesor *Intel(R) Core(TM) i7-9750H* (2.60 GHz), 16 GB pamięci operacyjnej oraz system operacyjny Windows 64-bit. Taka konfiguracja została wybrana ze względu na zapewnienie powtarzalności pomiarów oraz porównywalności wyników z wcześniejszymi badaniami prezentowanymi w literaturze.

Badania zostały zorganizowane w taki sposób, aby każda z trzech metod porównawczych (metoda Martinez–Silva, narzędzie PIPE oraz opracowany w zespole badawczym algorytm) analizowała dokładnie te same modele, w tej samej kolejności i na identycznych parametrach środowiskowych. Dodatkowo, dla zapewnienia uczciwego porównania, każdej metodzie przydzielono maksymalny czas analizy równy jednej godzinie. W przypadku przekroczenia tego czasu przypadek testowy był oznaczany jako *timeout*, co zgodne było z przyjętymi wcześniej kryteriami eksperymentalnymi.

Zarówno procedury pomiarowe, jak i struktura danych wynikowych były zdefiniowane tak, aby umożliwić automatyczne porównanie wyników i ocenę trafności detekcji błędów.

## 5.3. Procedura eksperymentów

Procedura eksperymentalna została zaprojektowana tak, aby umożliwić obiektywne porównanie opracowanej metody szybkiej weryfikacji z klasycznymi podejściami opartymi na pełnej analizie strukturalnej. W szczególności skoncentrowano się na dwóch kryteriach oceny: trafności wykrywania niepokrytych tranzycji oraz czasie działania.

Każdy model z bazy testowej został poddany analizie za pomocą trzech metod:

- klasycznej metody Martinez–Silva,
- narzędzia PIPE,
- opracowanej metody szybkiej weryfikacji.

Aby zapewnić porównywalność wyników, wszystkie analizy przeprowadzono na tym samym środowisku sprzętowym, z identycznymi parametrami wykonania i ograniczeniem czasowym wynoszącym 1 godzinę na pojedynczy przypadek testowy.

| Benchmark | Places | Transitions | Reference Method | | PIPE Tool | | Proposed Method | |
|---|---|---|---|---|---|---|---|---|
| | | | Result | Time [ms] | Result | Time [ms] | Result | Time [ms] |
| SHA_decoder | 200 | 170 | true | 10.06 | true | 65,385 | true | 150.35 |
| cn_crr25 | 200 | 51 | true | 1.42 | true | 34.00 | true | 16.00 |
| photovoltaics_single_module | 139 | 162 | n/a | timeout | n/a | timeout | true | 69.06 |
| manufacturing_AM_deadlock | 75 | 73 | false | 854.1 | false | 29,131 | false | 11.97 |
| decision_making_system_v2 | 75 | 73 | false | 899.8 | false | 9697 | false | 12.21 |
| lnet_p8n1 | 51 | 40 | false | 0.53 | n/a | timeout | false | 5.59 |
| RSA_v1 | 49 | 58 | n/a | timeout | n/a | timeout | true | 5.79 |
| RSA_v2 | 49 | 58 | true | 22.05 | n/a | timeout | true | 6.34 |
| effectiveness_AM_v2_unsafe | 48 | 52 | n/a | timeout | n/a | timeout | false | 5.64 |
| state_space48 | 48 | 48 | true | 0.67 | true | 3282 | true | 3.20 |
| zuberek5 | 41 | 31 | true | 0.80 | true | 2.00 | true | 2.12 |
| five_food_preparation_lines | 37 | 33 | true | 0.38 | true | 1153 | true | 2.48 |
| crossroadSM_FPGA | 32 | 12 | true | 0.12 | n/a | timeout | true | 0.50 |
| airplane_flight_procedure | 28 | 17 | n/a | timeout | n/a | timeout | true | 0.80 |
| HAN | 24 | 40 | true | 255,822 | n/a | timeout | true | 1.26 |
| two_lanes_long | 21 | 40 | n/a | timeout | n/a | timeout | true | 1.93 |
| 66_greenhouse_automation | 17 | 22 | true | 8.36 | true | 859.00 | true | 0.70 |
| automation3 | 17 | 16 | false | 0.206 | false | 277.000 | true | 0.831 |
| bause1 | 11 | 11 | false | 0.10 | false | 134.00 | false | 0.87 |
| esparza1 | 11 | 10 | false | 0.064 | false | 261.000 | true | 0.397 |
| credit_procedure | 10 | 8 | false | 0.16 | false | 93.00 | false | 0.90 |
| bridge | 8 | 6 | true | 0.03 | true | 309.00 | true | 0.23 |
| 47_two_vehicles | 7 | 6 | true | 0.03 | true | 406.00 | true | 0.24 |

**Tabela 5.1.** Tabela wynikowa przeprowadzonych badań[3]

Wyniki eksperymentów zostały przedstawione w formie liczbowej w tabeli 5.1, z podziałem na czas wykonania oraz skuteczność wykrywania niepokrytych tranzycji.

Przypadki, dla których czas wykonania przekroczył limit jednej godziny, zostały oznaczone jako *timeout*, zgodnie z przyjętymi założeniami eksperymentalnymi.

Dla każdej z metod zarejestrowano informację o:

- nazwie badanej sieci,
- liczbie miejsc,
- liczbie tranzycji,
- wynikach poszczególnych metod analizy oraz czasu trwania tej analizy

Uzyskane dane umożliwiły pełną ocenę zarówno skuteczności (trafności), jak i efektywności (czasowej) proponowanej metody.

## 5.4. Szczegóły badań w środowisku PIPE

W celu umożliwienia wydajnej analizy dużej liczby modeli sieci Petriego w narzędziu PIPE (*Platform Independent Petri net Editor*), dokonano istotnych modyfikacji kodu modułu odpowiedzialnego za analizę inwariantów.

Bazowa implementacja modułu `InvariantAnalysis` dostępna w oficjalnej wersji PIPE umożliwia analizę pojedynczego modelu sieci poprzez interfejs graficzny. W ramach współopracowanego artykułu, przygotowano nowy moduł `InvariantAnalysisBulk`, który rozszerza funkcjonalność oryginału o możliwość przeprowadzania analiz wsadowych.

Główna modyfikacja polegała na:

- wyodrębnieniu logiki analizy pojedynczego pliku z oryginalnej klasy i nadaniu jej parametrów umożliwiających wielokrotne wywołania w pętli,
- wczytywaniu wszystkich plików z wybranego folderu źródłowego,
- automatycznej weryfikacji poprawności formatu pliku (.pnh lub .xml),
- wykonaniu analizy tylko dla poprawnych modeli (pozostałe są pomijane),
- rejestrowaniu wyników analizy (czas wykonania, pokrycie inwariantami P i T) w formacie tabelarycznym,

- zapisywaniu wyników do pliku `execTimes.xlsx`.

Dodatkowo moduł umożliwia analizę w trybie uproszczonym – jedynie czas wykonania wraz z flagami informującymi o pokryciu inwariantami P i T – co znacząco przyspiesza testy dużych zbiorów danych.

Dla realizacji funkcjonalności eksportu danych wykorzystano bibliotekę `Apache POI`, a wstępna konwersja plików z formatu `.pnh` do `.xml` realizowana była za pomocą klasy pomocniczej `HippoConverter`. Wynikiem działania algorytmu jest plik `execTimes.xlsx` zawierający nazwę modelu, czas analizy w mikrosekundach, oraz informacje o pokryciu inwariantami.

Tak przystosowane środowisko pozwoliło na szybkie i powtarzalne przeprowadzenie eksperymentów na zbiorze 386 modeli testowych bez konieczności ręcznego uruchamiania każdego przypadku.

## 5.5. Modyfikacje kodu

W celu umożliwienia automatycznej analizy wielu modeli sieci Petriego w ramach jednego uruchomienia programu, konieczne było wprowadzenie istotnych zmian do istniejącego modułu analizy inwariantów w narzędziu PIPE (*Platform Independent Petri net Editor*). Oryginalna wersja modułu `InvariantAnalysis` pozwalała jedynie na interaktywną analizę pojedynczego modelu poprzez GUI, bez możliwości przetwarzania wielu modeli w sposób zautomatyzowany.

Zaprojektowana przez autora niniejszej pracy modyfikacja — nowy moduł `InvariantAnalysisBulk` — umożliwia analizę wsadową całych folderów zawierających pliki modeli, a następnie zapis wyników do pliku Excela. Poniżej przedstawiono szczegółowy opis wprowadzonych zmian.

### 5.5.1. Modularizacja logiki analitycznej

Pierwszym krokiem było wydzielenie logiki odpowiedzialnej za analizę pojedynczego modelu. Oryginalnie kod analityczny był powiązany bezpośrednio z interfejsem graficznym. W nowej wersji kluczowa funkcjonalność została przeniesiona do metody:

```
private String[] analyse(PetriNetView pnmlData, Boolean timeAnalyzeOnly)
```

Metoda ta przyjmuje jako argument obiekt typu `PetriNetView` (wczytany model sieci) oraz flagę trybu uproszczonego. Zwraca tablicę tekstową zawierającą:

- czas wykonania analizy (w mikrosekundach),
- informację o pokryciu inwariantami miejsc (P),
- informację o pokryciu inwariantami przejść (T).

Dzięki temu metoda ta mogła być wielokrotnie wywoływana w pętli analizującej kolejne pliki w katalogu wejściowym.

### 5.5.2. Obsługa folderów i plików wejściowych

W nowym module dodano możliwość przetwarzania całych folderów zawierających pliki z modelami sieci. W tym celu wprowadzono komponent GUI `PetriNetsChooserPanel`, który umożliwia wybór folderu z poziomu aplikacji.

Następnie w metodzie wywołującej analizę, zastosowano iterację po wszystkich plikach:

```
for (File file : Objects.requireNonNull(xmlFilesDir.listFiles())) {
    ...
}
```

Dla każdego pliku przeprowadzana była kontrola poprawności formatu. Pliki ‘.pnh‘ były konwertowane do ‘.xml‘ za pomocą klasy pomocniczej `HippoConverter`. Jeśli plik nie zawierał obiektów sieci (miejsc lub przejść), był automatycznie pomijany.

### 5.5.3. Konwersja plików PNH do XML

Oryginalne pliki wejściowe dostępne w projekcie HIPPO zapisane były w formacie ‘.pnh‘. W celu umożliwienia ich użycia w PIPE, konieczna była konwersja do formatu ‘.xml‘, zgodnego z wymaganiami edytora. Operację tę realizowano za pomocą klasy `HippoConverter`:

```
String pipeFile = hippoConverter.generateXMLPipe(file.toPath());
Files.write(path, pipeFile.getBytes(StandardCharsets.UTF_8));
```

Nowo wygenerowany plik XML był następnie analizowany przy użyciu standardowego parsera stanu sieci.

### 5.5.4. Zapis wyników do pliku Excela

Wyniki analizy dla każdego modelu zapisywane są do pliku `execTimes.xlsx`. W tym celu zaimplementowano dodatkowe metody zapisujące dane zebrane podczas analizy:

```
editAndSaveToExcel(ArrayList<String[]> executionTimes, String pathFile)
```

Każdy wpis zawiera następujące dane:

- nazwę analizowanego pliku,
- czas wykonania (kolumna 5/6),
- wynik pokrycia inwariantami P i T (kolumny 7 i 8),
- ścieżkę do pliku (kolumna 9).

Jeśli plik Excela już istnieje, dane są uzupełniane w odpowiednim wierszu — dzięki funkcji:

```
findRowByColumnValue(String filePath, String searchValue)
```

Zastosowano bibliotekę Apache POI do operacji na plikach ‘.xlsx‘.

### 5.5.5. Tryb uproszczony (czas + pokrycie)

Aby przyspieszyć eksperymenty, wprowadzono dodatkową opcję interfejsu użytkownika w postaci pola typu `JCheckBox`, oznaczonego jako „Analyse (time only)”. Po jej zaznaczeniu program pomija generowanie raportów HTML i zapisuje tylko dane minimalne: czas wykonania oraz pokrycie P- i T-inwariantami.

Fragment logiki obsługującej tryb uproszczony:

```
if(timeAnalyzeOnly){
    out = new String[]{String.valueOf(elapsedTimeMillis),
        String.valueOf(pInvariantsCovered),
        String.valueOf(tInvariantsCovered)};
}
```

### 5.5.6. Kontrola błędów i optymalizacje

W module dodano liczne zabezpieczenia przed błędami:

- sprawdzenie, czy plik XML istnieje i ma rozmiar większy niż zero,
- pominięcie modeli bez zdefiniowanych miejsc i przejść,
- obsługa błędów pamięci i błędów wykonania w blokach `try-catch`,
- manualne czyszczenie pamięci (`System.gc()`) w przypadku błędu `OutOfMemoryError`.

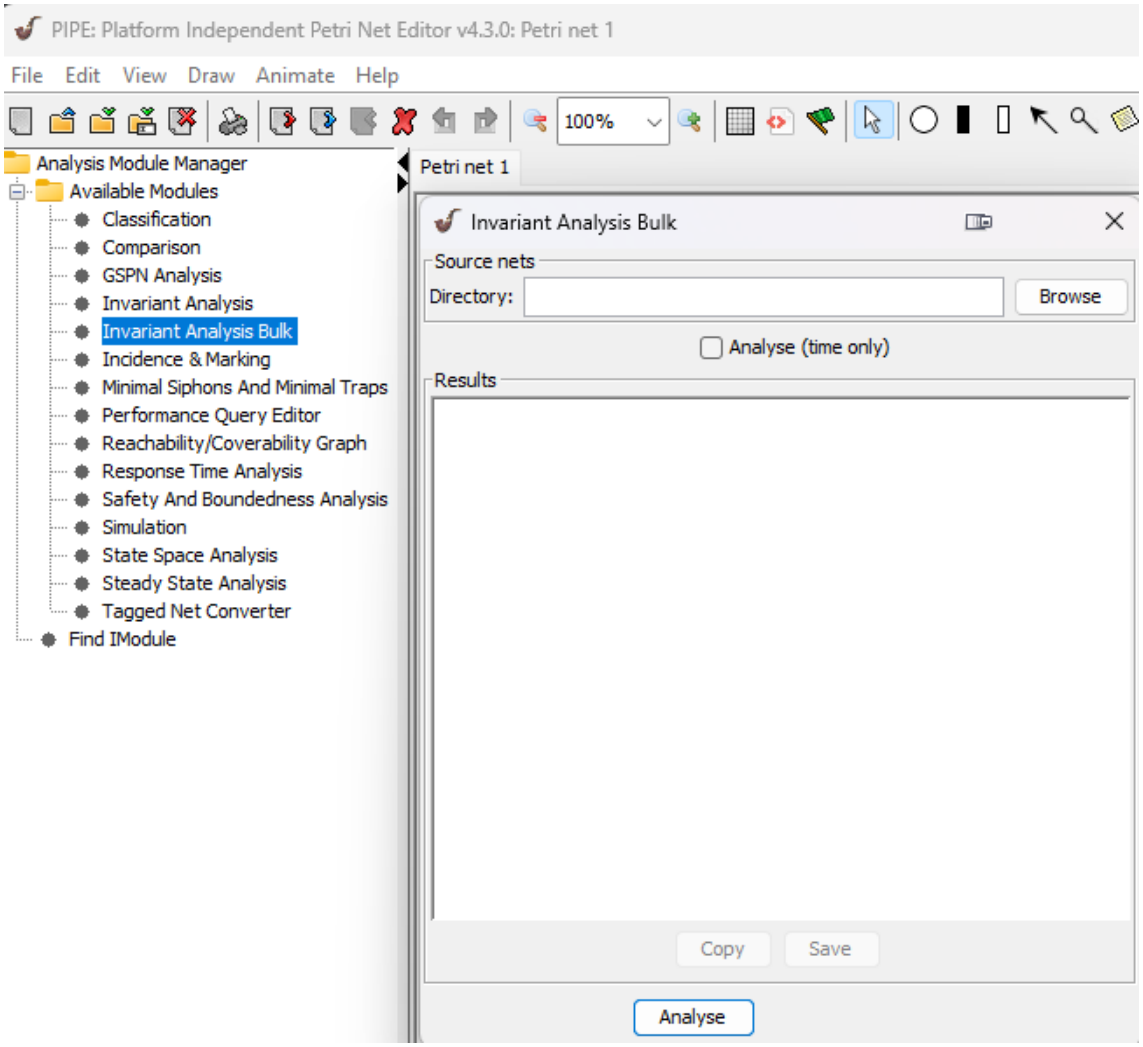
### 5.5.7. Podsumowanie

Zmodyfikowany moduł `InvariantAnalysisBulk` znacząco zwiększa funkcjonalność środowiska PIPE w kontekście analizy dużych zbiorów modeli sieci Petriego. Rozszerzenie to umożliwia pełną automatyzację procesu eksperymentalnego, eliminując konieczność ręcznego uruchamiania analiz dla każdego pliku osobno, co w przypadku analizy setek lub tysięcy sieci stanowi istotne usprawnienie. Dzięki zastosowaniu tego podejścia możliwe jest wykonywanie wielkoskalowych eksperymentów w sposób efektywny czasowo oraz w pełni powtarzalny, co bezpośrednio przekłada się na wiarygodność uzyskiwanych wyników.

W kontekście niniejszej pracy, moduł ten odegrał kluczową rolę w realizacji badań empirycznych — zarówno tych przeprowadzonych na potrzeby wspólnej publikacji naukowej, jak i badań będących integralną częścią tej pracy dyplomowej. Automatyzacja analizy inwariantów, a także możliwość wyboru formatu wyników (pełny lub minimalny) pozwoliły nie tylko na redukcję czasu obliczeń, lecz także na precyzyjne dopasowanie struktury danych wyjściowych do potrzeb dalszej obróbki i porównań.

Na rysunku 5.1 zaprezentowano zrzut ekranu przedstawiający zaktualizowany interfejs użytkownika środowiska PIPE. Widoczna jest nowa zakładka odpowiadająca

za analizę inwariantów w trybie wsadowym, której obecność potwierdza zintegrowanie zmodyfikowanego modułu z główną częścią systemu. W centralnej części okna programu umieszczony został specjalny panel umożliwiający konfigurację parametrów analizy, natomiast poniżej widoczny jest checkbox, który pozwala użytkownikowi zadecydować, czy wynik analizy ma zawierać wszystkie wykryte inwarianty, czy jedynie ich minimalny reprezentatywny podzbiór. Tego rodzaju opcja znacząco ułatwia zarówno analizę jakościową, jak i porównywanie wyników między różnymi sieciami.




**Rysunek 5.1.** Interfejs użytkownika środowiska PIPE z widocznym zmodyfikowanym modułem analizy inwariantów w trybie wsadowym - opracowanie własne

# Rozdział 6

# Podsumowanie i wnioski

Praca dyplomowa pt. „Modelowanie oraz analiza systemów produkcyjnych specyfikowanych siecią Petriego" miała na celu opracowanie techniki modelowania oraz analizy systemów produkcyjnych przy użyciu sieci Petriego, kładąc szczególny nacisk na aspekty weryfikacji systemu już na wczesnym etapie projektowym. Zgodnie z założonym zakresem, w drugim rozdziale pracy dokonano przeglądu aktualnego stanu wiedzy z zakresu systemów produkcyjnych specyfikowanych siecią Petriego oraz podstaw teoretycznych tych sieci, wskazując ich istotne własności i zastosowania w praktyce inżynierskiej. Trzeci rozdział został poświęcony szczegółowej dyskusji dotyczącej skuteczności i sprawności istniejących technik modelowania i analizy systemów produkcyjnych, prezentując zarówno metody strukturalne, jak i behawioralne, oraz wskazując ich mocne strony oraz ograniczenia. Rozdział czwarty pracy przedstawia opracowanie nowatorskiej techniki modelowania i analizy systemów produkcyjnych, w szczególności w kontekście weryfikacji systemów na wczesnym etapie projektowym oraz szczegółowo opisuje opracowaną metodę szybkiej weryfikacji, w tym mechanizm działania algorytmu, szczegóły implementacyjne oraz przeprowadzone porównania jej efektywności względem klasycznych podejść analitycznych. Piąty rozdział przedstawia autorskie badania eksperymentalne mające na celu określenie skuteczności oraz sprawności zaproponowanej techniki weryfikacyjnej. Badania przeprowadzono na szerokiej bazie testowej, wykorzystując do tego środowisko PIPE, w którym dodatkowo dokonano autorskich modyfikacji usprawniających przeprowadzanie analiz wsadowych. Ostatni rozdział pracy obejmuje kompleksową analizę oraz dyskusję uzyskanych wyników, jak również opracowanie końcowych wniosków, wskazujących

na potencjał proponowanej metody w praktyce projektowej. Przedstawiono także propozycje dalszych kierunków badań, które pozwolą na rozszerzenie i pogłębienie osiągniętych rezultatów.

## 6.1. Kluczowe wnioski wynikające z przeprowadzonych analiz

W toku badań opracowano wersję algorytmu umożliwiającą selektywną analizę lokalnych struktur sieci w celu detekcji niepokrytych tranzycji, co stanowi sygnał potencjalnych błędów projektowych, takich jak blokady, zakleszczenia lub niespełnione warunki aktywacji. Badania metody w środowisku PIPE pozwoliła na przetestowanie jej działania na bazie 386 modeli systemów rzeczywistych i referencyjnych.

Kluczowe obserwacje płynące z badań obejmują:

- opracowana metoda zwracała wyniki dla 100% przypadków testowych (386/386),
- czas wykonania analizy był średnio od kilku do kilkudziesięciu razy krótszy niż w przypadku klasycznych metod strukturalnych,
- skuteczność w wykrywaniu niepokrytych tranzycji osiągnęła poziom 99,48% (384/386 przypadków zgodnych z pełną analizą),
- metoda wykazuje największą przewagę czasową w przypadku dużych modeli (powyżej 100 miejsc i tranzycji), gdzie klasyczne metody często przekraczały limit czasowy (1 godzina),
- nowy tryb wsadowy oraz automatyczny eksport wyników dla algorytmów programu PIPE umożliwiły pełną automatyzację procesu zbierania danych z programu PIPE do dalszej analizy.

## 6.2. Porównanie opracowanej metody z metodami klasycznymi

W toku badań porównano opracowaną metodę szybkiej analizy z dwiema popularnymi alternatywami: klasycznym algorytmem Martinez–Silva oraz narzędziem PIPE wykorzystującym analizę stanu. W każdym przypadku zastosowano identyczny zestaw danych, środowisko sprzętowe i limit czasowy, co pozwoliło na rzetelną ocenę porównawczą. Klasyczna metoda Martinez–Silva cechuje się wysoką formalnością oraz pełnym pokryciem przestrzeni inwariantów, jednak jej koszt obliczeniowy rośnie wykładniczo względem rozmiaru sieci. W praktyce skutkuje to bardzo długim czasem analizy – w 7 przypadkach algorytm ten nie zwrócił wyniku w czasie poniżej jednej godziny. Z kolei narzędzie PIPE, mimo że wyposażone w wiele funkcji analitycznych, również nie poradziło sobie z 47 przypadkami testowymi w zadanym czasie. W badaniu wydajnościowym najdłuższe przypadki (np. `photovoltaics_single_module`, `airplane_flight_procedure`) nie zostały poprawnie przeanalizowane (w wyznaczonym czasie) ani przez Martinez–Silva, ani przez PIPE.

Opracowana metoda zwracała wyniki dla wszystkich 386 przypadków, a jej złożoność obliczeniowa $O(|P|^2 \cdot |T|)$ pozwalała na błyskawiczne wykonanie analizy — w wielu przypadkach poniżej 10 ms. Co istotne, skuteczność metody utrzymywała się na bardzo wysokim poziomie, mimo braku pełnej eksploracji stanów.

Porównanie można podsumować następująco:

- **Czas działania:** średnio od 10 do 100 razy szybszy dla dużych modeli niż metody referencyjne,
- **Zasięg analizy:** niepełny (heurystyczny), lecz trafny – 99,48% zgodności z wynikami referencyjnymi,
- **Złożoność:** wielomianowa, skalowalna – istotna przewaga nad metodami klasycznymi w modelach średnich i dużych.

Z uwagi na swoje właściwości, metoda doskonale nadaje się do zastosowań inżynierskich, w których istotny jest czas analizy, a pełna gwarancja poprawności nie jest bezwzględnie wymagana na etapie wczesnego projektowania.

## 6.3. Wskazanie kierunków dalszych badań

Wyniki uzyskane w ramach przeprowadzonych eksperymentów potwierdziły skuteczność opracowanego algorytmu szybkiej weryfikacji, co stanowi silną motywację do jego dalszego rozwoju. Choć skuteczność metody przekracza 99%, planowane są prace mające na celu dokładne zbadanie przypadków, w których analiza nie zidentyfikowała niepokrytych tranzycji.

Analiza tych dwóch wyjątków ma na celu lepsze zrozumienie ograniczeń obecnej heurystyki oraz identyfikację warunków, które powodują jej nieskuteczność. Jednym z rozważanych kierunków jest modyfikacja strategii przekształcania macierzy incydencji do postaci schodkowej (RREF). Poprzez eksperymenty z alternatywnymi porządkami kolumn możliwe może być zwiększenie liczby poprawnie wykrywanych niepokrytych tranzycji — a w konsekwencji być może uzyskanie pełnego pokrycia wszystkich przypadków. Dodatkowo, rozważana jest integracja proponowanego podejścia z analizą inwariantów miejsc. Połączenie informacji wynikających z inwariantów P oraz heurystyk strukturalnych wykorzystywanych przez omawiany algorytm może prowadzić do synergii w ocenie zarówno miejsc, jak i tranzycji w modelu sieci Petriego, co może skutkować poprawą trafności i spójności analizy.

Podsumowując, praca potwierdziła, że możliwe jest znaczące przyspieszenie analizy modeli sieci Petriego bez istotnej utraty jakości wyników. Opracowana metoda łączy w sobie prostotę, efektywność oraz praktyczną użyteczność, czyniąc ją potencjalnym standardem wstępnej walidacji modeli w systemach współbieżnych i cyberfizycznych.

# Bibliografia


[1] Remigiusz Wiśniewski, Justyna Patalas-Maliszewska, Marcin Wojnakowski, and Marcin Topczak. Modelling and analysis of a petri net-based system supporting implementation of additive manufacturing technologies. *IEEE Transactions on Automation Science and Engineering*, pages 546–556, 2023.

[2] Uniwersytet Zielonogórski. *HIPPO*. `https://hippo.uz.zgora.pl`, 2020.

[3] Marcin Wojnakowski, Maxim Maliński, Remigiusz Wiśniewski, Andrzej Obuchowicz, Zhiwu Li, and Dawid Konarczak. A polynomial-time algorithm for detection of uncovered transitions in a petri net-based concurrent system. *Applied Sciences*, 15(2):680, 2025.

[4] Wolfgang Reisig. Nets consisting of places and transistions. In Wolfgang Reisig, editor, *Petri Nets: An Introduction*, EATCS Monographs on Theoretical Computer Science, pages 62–76. Springer, Berlin/Heidelberg, Germany, 1985.

[5] Claude Girault and Rüdiger Valk. *Petri Nets for Systems Engineering: A Guide to Modeling, Verification, and Applications*. Springer, Berlin/Heidelberg, Germany, 2003.

[6] James L. Peterson. *Petri Net Theory and the Modeling of Systems*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1981.

[7] Tadao Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[8] J. Martínez and M. Silva. A simple and fast algorithm to obtain all invariants of a generalised petri net. In C. Girault and W. Reisig, editors, *Application*

*and Theory of Petri Nets*, pages 301–310. Springer, Berlin/Heidelberg, Germany, 1982.

[9] Eike Best, Régis Devillers, and Maciej Koutny. *Petri Net Algebra*. Monographs in Theoretical Computer Science. An EATCS Series. Springer, Berlin/Heidelberg, Germany, 2001.

[10] R. Wisniewski, I. Grobelna, and A. Karatkevich. Determinism in cyber-physical systems specified by interpreted petri nets. *Sensors*, 20(19), 2020.

[11] R. Wiśniewski. *Prototyping of Concurrent Control Systems Implemented in FPGA Devices*. Springer International Publishing, 2017.

[12] A. Ramirez-Trevino, I. Rivera-Rangel, and E. Lopez-Mellado. Observability of discrete event systems modeled by interpreted petri nets. *IEEE Transactions on Robotics and Automation*, 19(4):557–565, August 2003.

[13] Remigiusz Wiśniewski. Prototyping of concurrent control systems implemented in fpga devices. In *Advances in Industrial Control*. Springer International Publishing, Cham, Switzerland, 2017.

[14] Roman Zurawski and MengChu Zhou. Petri nets and industrial applications: A tutorial. *IEEE Transactions on Industrial Electronics*, 41(6):567–583, 1994.

[15] Remigiusz Wiśniewski, Zhiwu Li, and Justyna Patalas-Maliszewska. Analysis of safeness in a petri-net based cps model. *International Journal of Applied Mathematics and Computer Science*, 31(4):591–602, 2021.

[16] Zhiwu Li and MengChu Zhou. *Deadlock Resolution in Automated Manufacturing Systems*. Advances in Industrial Control. Springer, London, UK, 2009.