

# Zastosowanie metod formalnych

Karol Kozłowski, Katarzyna Mielęcka, Jan Gawroński, Piotr  
Głowacki

Wydział elektryczny, Politechnika Warszawska

6 kwietnia 2025

## Spis treści

- 1 Wprowadzenie do metod formalnych
- 2 Wykorzystanie metod formalnych w systemie PVS
  - Wprowadzenie do systemu PVS
  - Jaki problem rozwiązuje system PVS?
  - Rozwiązanie PVS
  - Formalna specyfikacja niezmienników
  - Automatyczna weryfikacja warunków spójności
  - Półautomatyczne dowodzenie poprawności
  - Podsumowanie
- 3 Zastosowanie metod formalnych: TLA+
- 4 Zastosowanie metod formalnych w kolejnictwie
  - Dlaczego warto?
  - Metoda B
  - Model Checking

# Czym są metody formalne?

## Definicja

Metody formalne to techniki matematyczne służące do:

- Specyfikacji - precyzyjnego opisu systemów za pomocą języków matematycznych (np. logika temporalna, rachunek procesów)
- Weryfikacji - dowodzenia poprawności systemów poprzez np. model checking (np. narzędzie SPIN) lub dowody twierdzeń (np. Coq, Isabelle)
- Automatyzacji analizy wykrywania sprzeczności lub luk w projektach na etapie modelowania (np. weryfikacja protokołów kryptograficznych)

## Kluczowe zalety

- Formalne gwarancje poprawności - zapewnienie matematycznie udowodnionej poprawności systemów, szczególnie w przypadku wymagań bezpieczeństwa.
- Precyzyjna specyfikacja wymagań - eliminacja niejednoznaczności dzięki matematycznym modelom i notacjom.
- Wykrywanie złożonych błędów - identyfikacja problemów takich jak zakleszczenia (deadlocks) czy warunki wyścigu (race conditions), które trudno wykryć tradycyjnymi metodami testowania.

## System PVS w pigułce

System PVS służy do opracowania i weryfikacji specyfikacji opisujących różne zagadnienia. PVS posiada rozbudowaną składnię i umożliwia operowanie w logice wyższego rzędu, definiowanie własnych typów i podtypów danych oraz tworzenie teorii parametryzowanych [6].

```
pointer_env [P: TYPE, T: TYPE]: THEORY
BEGIN
  pointer: TYPE = P + {nil}
  env: TYPE = [pointer -> (T + {undefined})]
END pointer_env
```

## Problem: Weryfikacja struktur dynamicznych

### Główne wyzwania

- **Złożoność struktur wskaźnikowych:**
  - Wycieki pamięci
  - Nieprawidłowe dereferencje
  - Zakleszczenia w systemach współbieżnych
- **Niejednoznaczność specyfikacji:**
  - Niewystarczalność logiki pierwszego rzędu
  - Potrzeba logiki wyższego rzędu (PVS)
- **Krytyczne zastosowania:**
  - Systemy sterowania (metro, koleje)
  - Aplikacje medyczne
  - Systemy awioniki i kosmiczne

## Rozwiązanie PVS

- Formalna specyfikacja niezmienników
- Automatyczna weryfikacja warunków spójności
- Półautomatyczne dowodzenie poprawności

## Przykład

### Przykład

```
accessed_disjoint?(accessed? : pred[pointer[P]]) :  
boolean =  
FORALL(v11,v12 : (valid_finseq_list?)) :  
  (accessed?(v11) AND accessed?(v12) AND v11  
  /= v12 =>  
    FORALL(1 : pointer[P]) : value?(1) =>  
      NOT list_member?(1,v11) OR NOT  
      list_member?(1,v12))
```



## Przykład

TCC1 dla niepustości listy, TCC2 dla zachowania typu w rekurencji,  
TCC3 dla warunku stopu rekurencji

```
last_TCC1: OBLIGATION
FORALL (v1: (valid_finseq_list?)): NOT empty?(v1)

last_TCC2: OBLIGATION
FORALL (v1: (valid_finseq_list?)):
    length(v1) > 1 IMPLIES valid_finseq_list?(tail(v1))

last_TCC3: OBLIGATION
FORALL (v1: (valid_finseq_list?)):
    length(v1) > 1 IMPLIES length(tail(v1)) < length(v1)
```

## Podsumowanie

- Metoda skuteczna dla list i drzew.
- Wymaga dużego nakładu pracy.
- Obiecujące wyniki dla systemów krytycznych.

## Czym jest TLA+?

- TLA+ (Temporal Logic of Actions) - język specyfikacji formalnej opracowany przez Lesliego Lamport.
- Służy do opisu systemów współbieżnych i rozproszonych.
- System definiowany jako zbiór zmiennych i akcji - przejść między stanami.
- Deklaratywny - skupia się na zachowaniu, nie na implementacji.
- Bazuje na logice temporalnej i matematyce zbiorów.

## Narzędzie TLC i model checking

- Weryfikacja poprawności specyfikacji odbywa się z użyciem narzędzia TLC.
- TLC używa techniki model checking:
  - automatyczne przeszukiwanie przestrzeni stanów,
  - sprawdzanie właściwości bezpieczeństwa i ciągłości.

## Metoda 1: Modelowanie matematyczne (FIFO)

### Fragment specyfikacji bufora w TLA+:

VARIABLE queue

Init == queue = << >>

Enq(x) == queue' = Append(queue, x)

Deq == /\ queue # << >>  
      /\ queue' = Tail(queue)

- Bufor jako lista (queue).
- Akcje opisują przejścia między stanami.

## Metoda 2: Inwariant bezpieczeństwa

### Fragment definicji inwariantu w TLA+:

```
TypeInvariant ==  
  /\ queue \in Seq(Int)  
  /\ Len(queue) >= 0
```

```
Inv == TypeInvariant
```

- Bufor jest sekwencją liczb całkowitych.
- Długość bufora nie może być ujemna.
- Inwarianty są sprawdzane automatycznie przez TLC.
- Chronią przed błędami, np. usunięciem z pustej kolejki.

## Metoda 3: Własności temporalne

### Fragment specyfikacji własności temporalnych w TLA+:

Safety == [] (Len(queue) >= 0)

Liveness == <>(queue = << >>)

- Safety (bezpieczeństwo): długość bufora nigdy nie jest ujemna.
- Liveness (ciągłość): bufor w końcu się opróżni.
- Temporalne właściwości opisują zachowanie w czasie.

## Zastosowania TLA+

- Firmy: Amazon, Microsoft, Google.
- Projektowanie i weryfikacja systemów:
  - rozproszonych,
  - bazodanowych,
  - komunikacyjnych,
  - chmurowych.
- Identyfikacja błędów przed implementacją - kluczowa dla systemów krytycznych.



## Korzyści metod formalnych

- Gwarancja bezpieczeństwa
  - Wczesne wykrywanie błędów
- Redukcja ryzyka
  - Minimalizacja ludzkich pomyłek
- Spełnienie standardów
  - Normy CENELEC EN 50128 (SIL 3/4)
- Optymalizacja kosztów
  - Tańsze poprawianie na etapie projektowania
- Zarządzanie złożonością
  - Współbieżność, czas rzeczywisty, interakcje

# Metody formalne w sygnalizacji

## Główne podejścia

- Metoda B
- Model Checking
- Sieci Petriego
- Dowodzenie Twierdzeń

# Metoda B

- Specyfikacja (maszyny abstrakcyjne)
- Stopniowa refinacja
- Generacja kodu z weryfikacją

# Maszyny abstrakcyjne

- zmienne
- niezmienniki (inwarianty)
- operacje

## Specyfikacja maszyny abstrakcyjnej dla silni (Cz. 1/3)

### Machine FACTORIAL\_MAC

**Sets:**

*factorial*, *m*

**Constants:**

*factorial*

**Properties:**

$$m \in \mathbb{N} \wedge$$

$$factorial \in \mathbb{N} \leftrightarrow \mathbb{N} \wedge$$

$$f \in \mathbb{N} \leftrightarrow \mathbb{N} \wedge$$

$$0 \mapsto 1 \in f \wedge$$

## Specyfikacja maszyny abstrakcyjnej dla silni (Cz. 2/3)

### Machine FACTORIAL\_MAC

#### Properties (cont.):

$$\forall f. \begin{cases} \forall (n, fn). (n \mapsto fn \in f \Rightarrow n + 1 \mapsto (n + 1) \times fn \in f) \\ \Rightarrow \\ factorial \subseteq f \end{cases}$$

#### Variables:

- *result*

#### Invariant:

$$result \in \mathbb{N}$$

## Specyfikacja maszyny abstrakcyjnej dla silni (Cz. 3/3)

### Machine FACTORIAL\_MAC

#### Assertions:

$$factorial \in \mathbb{N} \rightarrow \mathbb{N};$$

$$factorial(0) = 1;$$

$$\forall n. (n \in \mathbb{N} \Rightarrow factorial(n+1) = (n+1) \times factorial(n))$$

#### Initialisation:

$$result := \mathbb{N}$$

#### Operations:

$$computation = \text{begin } result := factorial(m) \text{ end}$$

# Zastosowania i problemy

## Zastosowania

- Metro paryskie
- Metro kopenhaskie
- Sygnalizacja kolejowa

## Wyzwania

- Wymaga ekspertyzy
- Czasochłonne



# Model Checking

- Automatyczna weryfikacja modelu
- Graf stanów i własności systemu
- Narzędzia: NuSMV, UPPAAL

## Przykładowe zastosowanie

- System ERTMS
- Weryfikacja bezpiecznych odległości

## Ograniczenia

Problem eksplozji stanów

## Porównanie metod

Metoda	Zalety	Wady
Metoda B	Pełna weryfikacja, generacja kodu	Wymaga ekspertyzy, czasochłonne
Model Checking	Automatyzacja, szybkość	Problem eksplozji stanów
Sieci Petriego	Wizualizacja współbieżności	Słaba skalowalność
Dowodzenie	Precyzja dla złożonych systemów	Czasochłonne, manualne

## Podsumowanie

- Metody formalne - niezbędne narzędzie w kolejnictwie
- Kluczowe dla spełnienia standardów bezpieczeństwa
- Wybór metody zależy od:
  - Złożoności systemu
  - Dostępnych zasobów
  - Wymagań standardów

### Przyszłe kierunki

Integracja metod (np. Model Checking + Sieci Petriego)

# Metody formalne w systemach wbudowanych

- Metody formalne są kluczowe dla niezawodności i bezpieczeństwa systemów wbudowanych.
- Systemy wbudowane działają w środowiskach krytycznych, takich jak motoryzacja, medycyna czy lotnictwo.
- Wykorzystanie metod formalnych umożliwia:
  - Precyzyjne modelowanie zachowań systemów.
  - Weryfikację zgodności z rygorystycznymi normami.
  - Walidację poprawności działania w różnych scenariuszach.
- Zapobiega awariom, które mogą prowadzić do poważnych konsekwencji.

## Metoda RT-EFSM - Wprowadzenie

### Czym jest RT-EFSM?

Rozszerzona Maszyna Stanów Skończonych Czasu Rzeczywistego (RT-EFSM) to formalny model stosowany do testowania oprogramowania wbudowanego czasu rzeczywistego, łączący elementy tradycyjnych FSM i EFSM z aspektami czasowymi.

### Kluczowe cechy modelu RT-EFSM

- Stany ( $S^*$ )
- Zdarzenia wejściowe i wyjściowe ( $I, O$ )
- Warunki strażnicze ( $C$ )
- Zmienne środowiskowe ( $V$ )
- Globalny zegar ( $L$ )

# Zastosowanie RT-EFSM w Testowaniu Systemów Wbudowanych

## Praktyczne zastosowania

Metoda RT-EFSM została skutecznie zastosowana w projektach takich jak testowanie systemu nawigacji bezwładnościowo-GPS dla lotnictwa, poprawiając jakość i niezawodność oprogramowania.

## Proces testowania RT-EFSM

- Tworzenie klas równoważności przejść z ograniczeniami czasowymi.
- Generowanie drzew scenariuszy testowych i sekwencji testowych.
- Definiowanie przypadków testowych według kryteriów pokrycia.

## Sieci Petriego – Podstawy i Zalety

### Czym są Sieci Petriego?

Sieci Petriego (Petri Nets) to formalny model matematyczny używany do modelowania i weryfikacji systemów współbieżnych oraz czasowo-zależnych. Składają się z:

- **Miejsc (Places)** – reprezentujących stany lub zasoby.
- **Tranzycji (Transitions)** – przedstawiających zdarzenia lub akcje.
- **Łuków (Arcs)** – określających relacje pomiędzy miejscami i tranzycjami.
- **Tokenów** – wskazujących aktualny stan systemu.

# Zastosowania Sieci Petriego w systemach wbudowanych

## Przykłady zastosowań

- **Kontrolery systemów krytycznych** – np. kontroler świateł drogowych, który wymaga zapewnienia, że zielone światła nie świecą się jednocześnie w przecinających się kierunkach.
- **Weryfikacja bezpieczeństwa** – narzędzia jak CPN Tools pozwalają sprawdzić zgodność systemu z zadanymi właściwościami bezpieczeństwa.
- **Wczesna weryfikacja** – możliwość formalnego potwierdzenia poprawności na wczesnych etapach projektowania systemu, co znacząco redukuje koszty.



## Zastosowanie metody B w systemach wbudowanych

- Metoda B znajduje szerokie zastosowanie w projektowaniu i weryfikacji systemów wbudowanych, szczególnie tych wymagających wysokiej niezawodności i bezpieczeństwa.
- Przykłady zastosowań:
  - Projektowanie jądra separacyjnego (Separation Kernel) dla systemów operacyjnych w bezpiecznych systemach wbudowanych.
  - Formalna specyfikacja komponentów systemowych, takich jak zarządzanie pamięcią czy interfejsy komunikacyjne.
  - Automatyczna weryfikacja poprawności systemu poprzez generowanie dowodów matematycznych.
  - Minimalizacja kodu krytycznego w jądrze, co ułatwia formalną weryfikację i zwiększa bezpieczeństwo.

## Korzyści zastosowania metody B

- Zapewnienie izolacji komponentów systemu oraz spełnienie rygorystycznych wymagań bezpieczeństwa.
- Tworzenie systemów wbudowanych spełniających najwyższe standardy jakości i bezpieczeństwa.
- Zastosowania w przemyśle lotniczym, medycznym oraz w systemach o znaczeniu krytycznym.
- Formalna weryfikacja zwiększa niezawodność i spójność oprogramowania.

## Bibliografia



Leslie Lamport, *Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers*, Addison-Wesley, 2002.



Chris Newcombe et al., *How Amazon Web Services Uses Formal Methods*, Communications of the ACM, 2015.



Igor Konnov, Jure Kukovec, Thanh-Hai Tran, *TLA+ Model Checking Made Symbolic*, CAV 2019.



Chris Newcombe, Tim Rath, Fan Zhang, Bogdan Munteanu, Marc Brooker, Michael Deardeuff, *How Amazon Web Services Uses Formal Methods*, Communications of the ACM, Vol. 58, No. 4, pp. 66–73, 2015.



Hillel Wayne, *Practical TLA+: Planning Driven Development*, Lospinato Books, 2018.



S. Poreda, *Wykorzystanie metod formalnych do specyfikacji struktur wskaźnikowych*, Uniwersytet Warszawski, 2023.

## Bibliografia (cd.)



Sławomir Lasota, *Weryfikacja protokołu Needhama-Schroedera przy użyciu narzędzi SPIN i UPPAAL*, Wydział Matematyki, Informatyki i Mechaniki, Uniwersytet Warszawski, [https://www.mimuw.edu.pl/~sl/teaching/03\\_04/WPKWK/PREZENTACJE-SPIN\\_UPPAAL/NS/](https://www.mimuw.edu.pl/~sl/teaching/03_04/WPKWK/PREZENTACJE-SPIN_UPPAAL/NS/).



Igor Wojnicki, *Weryfikacja własności systemów współbieżnych z użyciem metod formalnych*, Praca doktorska, Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie, 2019, <https://winntbg.bg.agh.edu.pl/rozprawy2/10463/full10463.pdf>.



J.R Abrial. *Modeling in Event-B*, Cambridge University Press, 2013.



Abrial, J.R., Lee, M.K.O., Neilson, D.S., Scharbach, P.N., Sørensen, I.H. *The B-method*. VDM '91 Formal Software Development Methods, 1991. <https://doi.org/10.1007/BFb0020001>



Dominique Cansell, Dominique Méry. *Tutorial on the event-based B method*. IFIP FORTE 2006, Paris, 2006.



A. Abdullah, *CENELEC EN 50128: Railway Applications - Communication, signaling and processing systems, Software for railway control and protection*