Zastosowanie metod formalnych

Karol Kozlowski,

Wydzial elektryczny, Politechnika Warszawska

6 kwietnia 2025

Czym sa 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)

Wiecej na ten temat mozna znalezc w [?].

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.

Wiecej na ten temat mozna znalezc w [?].

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 [?].

```
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

Wiecej na ten temat mozna znalezc w [?].

Rozwiazanie PVS

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

Przyklad

Przyklad

```
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))
```

Przyklad

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

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


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


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

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 Lamporta.
- 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.

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.


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