Modelowanie i Analiza Systemów Informatycznych

Logika Temporalna i Automaty Czasowe

(7)

Automaty czasowe NuSMV

wersja 2.4

Treść wykładu

NuSMV

- NuSMV – symboliczny weryfikator modelowy
- Do czego służy NuSMV
  - Modelowanie
  - Symulowanie
  - Weryfikowanie
- Przykład skryptu
- Jak uruchomić NuSMV

Treść wykładu

Język NuSMV

- Typy danych
- Makrodefinicje
- Operatory i predykaty
- Przypisanie i porównanie zmiennych
- Operacje na zbiorach
- Wyrażenie wyboru
- Moduły
- Przykład: wolny rynek
- Procesy i asynchroniczność

NuSMV

- NuSMV – symboliczny weryfikator modelowy
- Do czego służy NuSMV
  - Modelowanie
  - Symulowanie
  - Weryfikowanie
- Przykład pliku .smv
- Jak uruchomić NuSMV

NuSMV

NuSMV – symboliczny weryfikator modelowy

narzędzie	opis modelu systemu	logika temporalna
Kronos	język ET-LOTOS	CTL (TCTL)
NuSMV	**język SMV**	**LTL, CTL, RTCTL**
Spin	język PROMELA	LTL
UPPAAL	graficzny	CTL (TCTL)
Verus	język Verus	LTL, CTL, RTCTL, PRTCTL

- Rozszerzona i otwarta wersja SMV (*Symbolic Model Veryfier*).
- Specjalny język definicji automatów skończenie stanowych.
- Wykorzystanie BDD (binarnych diagramów decyzyjnych).

Do czego służy NuSMV

- Modelowanie, symulowanie i weryfikowanie skończenie stanowych systemów czasu rzeczywistego.
- Dla systemów deterministycznych i niedeterministycznych.
- Dla systemów synchronicznych i asynchronicznych.

NuSMV

Modelowanie

- Model systemu to skończenie stanowe automaty.
- Model może być modułowy i hierarchiczny.
- Definicja modelu przez skrypt w języku NuSMV.

Symulowanie

- Ręczna symulacja możliwych przebiegów automatów (ścieżek stanów).
- Możliwość wyboru symulowanej ścieżki stanów i jej długości.
- Ścieżka stanów jest prezentowana przez ich tekstowy opis.

NuSMV

Weryfikowanie

- Weryfikacja jest automatyczna.
- Formuły logiki temporalnej opisują specyfikację systemu.
- Dostępne logiki: LTL, CTL, LTL^{-}, RTCTL (z dolnym i górnym ograniczeniem operatorów temporalnych) i PSL.
- Dozwolone są wszystkie poprawnie zbudowane formuły.
- Każda formuła weryfikowana jest niezależnie od pozostałych.
- Weryfikacja formuły zwraca *true* lub *false*.
- Wynikowi *false* towarzyszy kontrprzykład (ścieżka stanów), jeśli można go wygenerować.
- Można wyliczyć długość minimalnej i maksymalnej ścieżki między dwoma określonymi stanami.

Przykład skryptu

```
--modelowanie pośrednie:

MODULE main

VAR   a : boolean;
      b : 0..4;

ASSIGN   init(a):= TRUE;
         next(a):= !a;
         init(b):= {0,2,4};
         next(b):= case
            next(a) : {0,2,4};
            !next(a) : {1,3};
         esac;

CTLSPEC AG(a -> b in {0,2,4})

INVARSPEC (!a -> b in {1,3})
```

```
--modelowanie bezpośrednie:

MODULE main

VAR   a : boolean;
      b : 0..4;

INIT  a = TRUE &
      b in {0,2,4};
TRANS next(a) = !a;
TRANS next(b) in case
         next(a) : {0,2,4};
         !next(a) : {1,3};
      esac;

CTLSPEC AG(a -> b in {0,2,4})

INVARSPEC (!a -> b in {1,3})
```

Jak uruchomić NuSMV

- Model systemu zapisany jest w pliku .smv.
- Praca w trybie automatycznym:

  `NuSMV` *`plik`*

- Praca w trybie interaktywnym:

  `NuSMV -int` *`plik`*

  - *Więcej w następnej prezentacji.*

Język NuSMV

- Typy danych
- Makrodefinicje
- Operatory i predykaty
- Przypisanie i porównanie zmiennych
- Operacje na zbiorach
- Wyrażenie wyboru
- Moduły
- Przykład: wolny rynek
- Procesy i asynchroniczność

Język NuSMV

Typy danych

Logiczny:

- wartości: FALSE i TRUE,
- przykład deklaracji:

```
a : boolean;
```

Ograniczony całkowitoliczbowy:

- wartości: od $-2^{31}+1$ do $2^{31}-1$,
- należy podać zakres wartości,
- przykład deklaracji:

```
b : 0..60;
```

Język NuSMV

Typy danych

Enumeracja:

- określone wartości: słowa (ciągi znaków) i liczby,
- kolejność wartości nie ma znaczenia,
- można mieszać słowa z liczbami,
- nie można używać wartości logicznych FALSE i TRUE,
- przykłady deklaracji:

```
c : {jeden, siedem, zero};
d : {1, 7, 0};
e : {jeden, siedem, 0};
```

Język NuSMV

Typy danych

Wektor bitów (wartości logicznych)**:**

- typy: *signed* i *unsigned*,
- wartości dla typu *signed*: od -2^{N-1} do $2^{N-1}-1$,
- wartości dla typu *unsigned*: od 0 do $2^{N}-1$,
  (N – rozmiar wektora)
- przykłady deklaracji:

```
f : unsigned word[4];
g : signed word[5];
```

Język NuSMV

Typy danych

Tablica:

- należy podać zakres indeksu i typ wartości,
- przykłady deklaracji:

```
h : array 0..5 of boolean;
i : array 10..20 of {tak, nie};
j : array 1..3 of array 1..2 of unsigned word[5];
```

Zmienne tworzy się poleceniem `VAR`, np:

```
VAR   a : boolean;
      b : 0..10;
```

Makrodefinicje

- Utworzenie makrodefinicji dla długiego wyrażenia skraca zapis.
- Przykład:

```
DEFINE c := a&b;
```

  - `c` jest tylko makrodefinicją dla `a&b`,
  - `c` można używać jak zmienną,
  - w BDD nie będzie zmiennej dla `c`,
  - w BDD zamiast `c` będzie wyrażenie `a&b`.

- W BDD istnieją tylko zmienne utworzone w ramach `VAR`.

Język NuSMV

Operatory i predykaty

- Operatory arytmetyczne:

  `+` suma,

  `–` różnica,

  `*` iloczyn,

  `/` iloraz (część całkowita),

  `mod` iloraz (reszta), Dzielenie przez 0 jest błędem!

  `<<` przesunięcie w lewo (`x << y` oznacza $x * 2^y$),

  `>>` przesunięcie w prawo (`x >> y` oznacza $x / 2^y$).

- Zależność między operatorami mnożenia i dzielenia:

  `y*(x/y) + (x mod y) = x`

Operatory i predykaty

- Operatory logiczne:

  `&` koniunkcja,

  `|` alternatywa,

  `xor` alternatywa wykluczająca,

  `!` negacja,

  `->` implikacja,

  `<->` równoważność.

- Wartości logiczne:

  `TRUE`,

  `FALSE`.

Operatory i predykaty

- Predykaty:

  = równe,

  != różne,

  < mniejsze,

  > większe,

  <= mniejsze lub równe,

  >= większe lub równe.

Język NuSMV

Przypisanie i porównanie zmiennych

- Wyrażenia można przypisywać do:
  - zmiennej w bieżącym stanie, np. `c := a+b`,
  - zmiennej w następnym stanie, np. `next(c) := a+b`,
  - zmiennej w początkowym stanie, np. `init(c) := a+b`.

- Wartości porównywać można ze:
  - zmienną w bieżącym stanie, np. `a = b`,
  - zmienną w następnym stanie, np. `next(c) = a+b`,
  - zmienną w początkowym stanie, np. `init(c) = a+b`.

Operacje na zbiorach

- Przypisanie do zmiennej dowolnej wartości ze zbioru, np.:

```
next(a) := {1,20,45};
```

- Porównanie zmiennej z dowolną wartością ze zbioru przy pomocy operatora `in`, np.:

```
b in {1,20,45};
```

- Utworzenie sumy zbiorów przy pomocy operatora `union`, np.:

```
init(c) := {s1,s2} union s3;
next(d) in 0 union 1;
```

Język NuSMV

Wyrażenie wyboru

- Przykład wyrażenia wyboru (*case*):

```
next(a) in case
        warunek1 : TRUE;
        warunek2 : {FALSE, TRUE};
        TRUE : a;  esac;
```

- Warunki porównywane są do `next(a)` w podanej kolejności.
- Uwzględniany jest tylko pierwszy spełniony warunek.
- Warunek `TRUE` jest zawsze spełniony.
- Należy uwzględnić wszystkie możliwe warunki.
- Po znaku : podano następną wartość dla `a`.
- Jeśli następną wartością jest `a`, to brak zmiany wartości.

Język NuSMV

Moduły

- Moduł jest komponentem systemu.
- Główny moduł:
  - nagłówek: `MODULE main`
  - jest obowiązkowy.
- Dodatkowy moduł:
  - nagłówek bez parametrów: `MODULE nazwa`
  - nagłówek z parametrami: `MODULE nazwa(param1, param2)`
  - parametr w module dostępny jest jak zwykła zmienna.
- Zmienną modułu może być instancja innego modułu, np.:

```
VAR   a : innyModuł(parametr)
```

Język NuSMV

Moduły

- Operator . pozwala na dostęp do wewnętrznej zmiennej modułu:

```
instancja_modułu.zmienna_modułu
```

- Przykład przekazywania wartości między modułami tego samego poziomu przez ich parametry:

```
MODULE main
VAR   a : coś(b.z);
      b : coś(a.z);

…

MODULE coś(parametr)
VAR   z : boolean;
```

Język NuSMV

Przykład: wolny rynek

- Założenia:
  - Dwaj handlarze synchronicznie modyfikują swoje ceny.
  - Każdy z nich chce mieć jak najwyższe ceny, ale mniejsze niż ma przeciwnik.
- Moduł główny:

```
MODULE main
VAR
   handlarz1 : handlarz(handlarz2.cena);
   handlarz2 : handlarz(handlarz1.cena);
DEFINE
   H1 := handlarz1.cena < handlarz2.cena;   --wygrywa H1
   H2 := handlarz2.cena < handlarz1.cena;   --wygrywa H2
   remis := handlarz1.cena = handlarz2.cena;
```

Język NuSMV

Przykład: wolny rynek

- Moduł handlarza:

```
MODULE handlarz(cena_konkurenta)
VAR
   cena : 0..25;
ASSIGN
   init(cena) := 10..20;
   next(cena) := case
         cena = 0 | cena = 25 : cena;
         cena < cena_konkurenta : cena + 1;
         cena >= cena_konkurenta : cena - 1;
         TRUE : cena;
      esac;
```

Przykład: wolny rynek

- Weryfikacja (w module głównym):

```
--skutki zwycięstwa handlarza1:
CTLSPEC AG(H1 -> AX(AF(H2 | remis)))     --prawda
CTLSPEC AG(H1 -> AX(AF(remis)))          --fałsz
CTLSPEC AG(H1 -> AX(AF(H2)))             --fałsz

--trwałość remisu:
CTLSPEC AG(remis -> AX(remis))           --prawda
CTLSPEC !remis -> AG(H1 | H2)            --fałsz
```

Język NuSMV

Procesy i asynchroniczność

- Proces to instancja modułu, która działa asynchronicznie z innymi procesami.
- Dla każdego procesu jego następny stan ustalany jest osobno, a nie jednocześnie.
- Proces zdefiniowany jest jak zwykły moduł.
- Instancja procesu zawiera słowo `process`:
  - bez parametrów: `VAR a : process nazwa;`
  - z parametrami: `VAR b : process nazwa(par1, par2);`

Modelowanie systemu

Procesy i asynchroniczność

- W modelu synchronicznym w jednym kroku:
  - następuje <u>jednocześnie</u> zmiana stanu każdego modułu
    - jednoczesna zmiana wartości zmiennych (wg specyfikacji) każdego modułu.
- W modelu asynchronicznym w jednym kroku:
  - następuje zmiana stanu jednego modułu (procesu)
    - zmiana wartości zmiennych (wg specyfikacji) jednego modułu.
  - Kolejność procesów jest losowa.
  - Zmienne pozostałych procesów pozostają w tym kroku niezmienione.
  - **Obecnie nie używa się procesów** (są „*deprecated*”)**.**

Koniec

Literatura:

- K.L. McMillan, „The SMV system”, 2001
- A. Cimatti et al. „NuSMV – a new symbolic model checker”
- R. Cavada et al. „NuSMV 2.5 User Manual”, 2010
- R. Cavada et al. „NuSMV 2.5 Tutorial”