Weryfikacja modelowa
Logika LTL, pakiet nuXmv

Marcin Szpyrka

Katedra Informatyki Stosowanej
AGH w Krakowie

2015/16

Literatura

1. Christel Baier, Joost-Pieter Katoen: Principles of Model Checking. The MIT Press, 2008
2. Michael Huth, Mark Ryan: Logic in Computer Science. Modelling and Reasoning about Systems. Cambridge University Press, 2004.
3. Tomasz Szmuc, Marcin Szpyrka (red.): Metody formalne w inżynierii oprogramowania systemów czasu rzeczywistego, WNT, Warszawa, 2010.
4. Iwona Grobelna: Weryfikacja modelowa z NuSMV. Oficyna Wydawnicza Uniwersytetu Zielonogórskiego, Zielona Góra, 2011.
5. Marco Bozzano et al.: nuXmv 1.0 User Manual (pdf).
6. Paul Jackson: Model checking with NuSMV (pdf)

Logiki temporalne

- Logiki temporalne wprowadził do informatyki pod koniec lat 70. ubiegłego wieku Amir Pnueli: A. Pnueli: *The temporal logic of programs*. In: *Proc. of the 18th IEEE Symposium on Foundations of Computer Science*, pp. 46–67, Providence, Rhode Island, 1977. IEEE Computer Society Presss.
- Rozważane na wykładzie logiki temporalne są nadbudowane nad klasycznym rachunkiem zdań i umożliwiają rozważanie zależności czasowych bez wprowadzania czasu *explicite*. W szczególności, logiki te dostarczają różnorodnych operatorów (modalności) do opisywania i rozumowania o tym, jak prawdziwość zdań może zmieniać się w czasie.
- Struktura czasowa składa się ze zbioru punktów czasowych oraz binarnej relacji poprzedzania porządkującej te punkty (rozważamy czas dyskretny).
  Na tym wykładzie rozważana będzie liniowa struktura czasowa, tj. zakładamy, że każdy punkt czasowy ma dokładnie jeden następnik.
  LTL – Linear Temporal Logic

Amir Pnueli (1941-2009) izraelski matematyk, za wprowadzenie logiki temporalnej do informatyki oraz znaczący wkład w weryfikację modelową dostał w 1996 r. medal Turinga.

Składnia LTL

Niech $a \in AP$ i niech φ, φ_1 i φ_2 będą formułami logiki LTL nad zbiorem formuł atomowych AP. Wówczas formułami LTL są również:

$$true \mid false \mid a \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \Rightarrow \varphi_2 \mid \varphi_1 \Leftrightarrow \varphi_2 \mid$$
$$\mathsf{X}\varphi \mid \mathsf{F}\varphi \mid \mathsf{G}\varphi \mid \varphi_1 \,\mathsf{U}\, \varphi_2 \mid (\varphi)$$

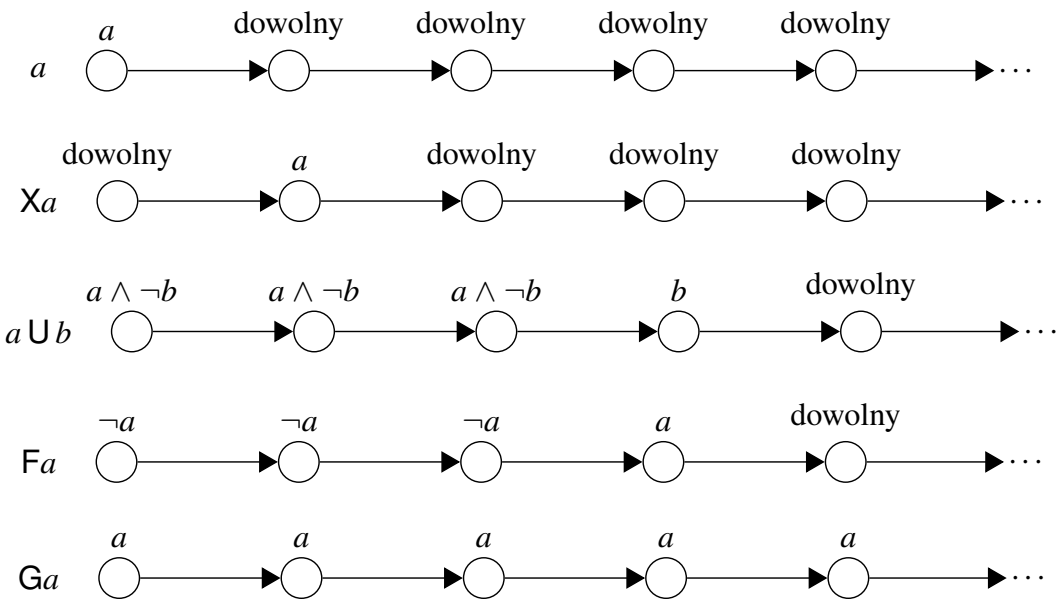
X – ne**X**t – w następnym stanie (punkcie czasowym)

U – **U**ntil – aż do

F – **F**inally – kiedyś w przyszłości

G – **G**lobally – teraz i zawsze w przyszłości

Intuicyjne znaczenie operatorów LTL



Operatory LTL

Priorytet operatorów

- $\neg$, X, G, F
- U
- $\vee$, $\wedge$
- $\Rightarrow$
- $\Leftrightarrow$

Oznaczenia alternatywne

- $\bigcirc \varphi \equiv \mathsf{X}\varphi$
- $\Diamond \varphi \equiv \mathsf{F}\varphi$
- $\Box \varphi \equiv \mathsf{G}\varphi$

Przykłady

Stany procesu $Proc_i$: n_i (noncritical), w_i (waiting) i c_i (critical).
Rozważamy system złożony z procesów $Proc_1$ i $Proc_2$, których obowiązuje wzajemne wykluczanie w dostępie do sekcji krytycznej.

- $\mathsf{G}(\neg c_1 \vee \neg c_2)$ – zawsze jeden proces poza sekcją krytyczną;
- $\mathsf{GF}c_1 \wedge \mathsf{GF}c_2$ – każdy z procesów jest nieskończenie wiele razy w sekcji krytycznej („zawsze możliwe”);
- $(\mathsf{GF}w_1 \Rightarrow \mathsf{GF}c_1) \wedge (\mathsf{GF}w_2 \Rightarrow \mathsf{GF}c_2)$ – brak zagłodzenia;

Problem n jedzących filozofów: w_i – i-ty filozof ma jeden widelec i czeka na drugi, t_i – i-ty widelec jest zajęty.

$$\mathsf{G}\neg\left(\bigwedge_{0\leqslant i<n} w_i \;\wedge\; \bigwedge_{0\leqslant i<n} t_i\right)$$

Formuła opisuje brak zakleszczenia.

Światła drogowe: oznaczenia sygnałów r, g i y.

$$\mathsf{G}\,(r \Rightarrow \mathsf{X}\,(r\,\mathsf{U}\;(y \wedge \mathsf{X}\,(y\,\mathsf{U}\,g))))$$

Semantyka formuł LTL (1)

Niech φ będzie formułą LTL na zbiorem AP. LT-własnością indukowaną przez φ nazywamy zbiór

$$Words(\varphi) = \left\{\sigma \in \left(2^{AP}\right)^{\omega} : \sigma \models \varphi\right\},$$

gdzie relacja spełniania $\models \subseteq \left(2^{AP}\right)^{\omega} \times \text{LTL}$ jest najmniejszą relacją taką że:

- $\sigma \models \mathit{true}$
- $\sigma \models a$ wtw, gdy $a \in A_0$,
- $\sigma \models \varphi_1 \wedge \varphi_2$ wtw, gdy $\sigma \models \varphi_1$ i $\sigma \models \varphi_2$, (analogicznie dla $\vee, \Rightarrow, \Leftrightarrow$),
- $\sigma \models \neg\varphi$ wtw, gdy $\sigma \not\models \varphi$,
- $\sigma \models \mathsf{X}\varphi$ wtw, gdy $\sigma[1\ldots] \models \varphi$,
- $\sigma \models \mathsf{F}\varphi$ wtw, gdy $\exists_{j\geqslant 0}\ \sigma[j\ldots] \models \varphi$,
- $\sigma \models \mathsf{G}\varphi$ wtw, gdy $\forall_{j\geqslant 0}\ \sigma[j\ldots] \models \varphi$.
- $\sigma \models \varphi_1\,\mathsf{U}\,\varphi_2$ wtw, gdy $\exists_{j\geqslant 0}\ \sigma[j\ldots] \models \varphi_2$ i $\forall_{0\leqslant i<j}\ \sigma[i\ldots] \models \varphi_1$.

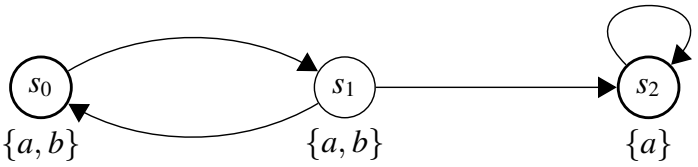
gdzie $\sigma = A_0A_1A_2\ldots \in \left(2^{AP}\right)^{\omega}$, $\sigma[j\ldots] = A_jA_{j+1}A_{j+2}\ldots$

Semantyka formuł LTL (2)

Niech $TS = (S, Act, \rightarrow, I, AP, L)$ będzie systemem tranzycyjnym bez stanów terminalnych i niech φ będzie formułą LTL na zbiorem AP.

- dla ścieżki $\pi \in Paths(TS)$, $\pi \models \varphi$ wtw, gdy $trace(\pi) \models \varphi$;
- Dla stanu $s \in S$, $s \models \varphi$ wtw, gdy $\forall_{\pi \in Paths(s)}\ \pi \models \varphi$;
- $TS \models \varphi$ wtw, gdy $Traces(TS) \subseteq Words(\varphi)$.

$I = \{s_0, s_2\}$

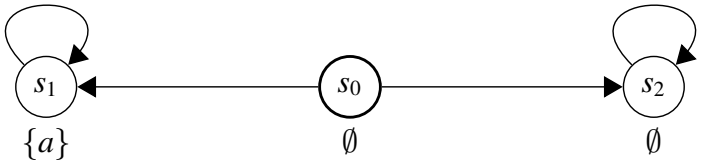


- $TS \models \mathsf{G}a$
- $s_0 \models \mathsf{X}(a \wedge b)$
- $s_2 \not\models \mathsf{X}(a \wedge b)$, stąd $TS \not\models \mathsf{X}(a \wedge b)$
- $TS \models \mathsf{G}(\neg b \Rightarrow \mathsf{G}(a \wedge \neg b))$
- $TS \not\models b\,\mathsf{U}\,(a \wedge \neg b)$

Semantyka negacji

Twierdzenie $\pi \in Paths(TS)$, $\quad \pi \models \varphi \ \Leftrightarrow\ \pi \not\models \neg\varphi$
$TS \models \neg\varphi \ \Rightarrow\ TS \not\models \varphi$ Nie ma równoważności!

Możliwe jest, że system tranzycyjny (lub stan) nie spełniają ani φ, ani $\neg\varphi$. Dzieje się tak, jeżeli istnieją ścieżki π_1 i π_2 takie, że $\pi_1 \models \varphi$ i $\pi_2 \models \neg\varphi$ (czyli $\pi_2 \not\models \varphi$). Wówczas $TS \not\models \varphi$ i $TS \not\models \neg\varphi$.



- $TS \not\models \mathsf{F}a$, bo $s_0 s_2^\omega \not\models \mathsf{F}a$
- $TS \not\models \neg\mathsf{F}a$, bo $s_0 s_1^\omega \not\models \neg\mathsf{F}a$

Równoważność formuł LTL (1)

Dwie formuły LTL φ_1, φ_2 są równoważne ($\varphi_1 \equiv \varphi_2$), jeżeli $Words(\varphi_1) = Words(\varphi_2)$.

Prawa dualizmu

$$\neg \mathsf{X}\varphi \equiv \mathsf{X}\neg\varphi$$
$$\neg \mathsf{F}\varphi \equiv \mathsf{G}\neg\varphi$$
$$\neg \mathsf{G}\varphi \equiv \mathsf{F}\neg\varphi$$

Prawa idempotencji

$$\mathsf{F}\mathsf{F}\varphi \equiv \mathsf{F}\varphi$$
$$\mathsf{G}\mathsf{G}\varphi \equiv \mathsf{G}\varphi$$
$$\varphi \,\mathsf{U}\, (\varphi \,\mathsf{U}\, \psi) \equiv \varphi \,\mathsf{U}\, \psi$$
$$(\varphi \,\mathsf{U}\, \psi) \,\mathsf{U}\, \psi \equiv \varphi \,\mathsf{U}\, \psi$$

Prawa pochłaniania

$$\mathsf{F}\mathsf{G}\mathsf{F}\varphi \equiv \mathsf{G}\mathsf{F}\varphi$$
$$\mathsf{G}\mathsf{F}\mathsf{G}\varphi \equiv \mathsf{F}\mathsf{G}\varphi$$

Równoważność formuł LTL (2)

Prawa ekspansji

„rozwijanie na osi czasowej”

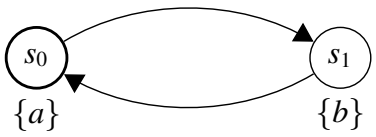
$$\varphi \,\mathsf{U}\, \psi \equiv \psi \vee (\varphi \wedge \mathsf{X}(\varphi \,\mathsf{U}\, \psi))$$
$$\mathsf{F}\varphi \equiv \varphi \vee \mathsf{X}\mathsf{F}\varphi$$
$$\mathsf{G}\varphi \equiv \varphi \wedge \mathsf{X}\mathsf{G}\varphi$$

Prawa rozdzielności

$$\mathsf{X}(\varphi \wedge \psi) \equiv \mathsf{X}\varphi \wedge \mathsf{X}\psi$$
$$\mathsf{X}(\varphi \vee \psi) \equiv \mathsf{X}\varphi \vee \mathsf{X}\psi$$
$$\mathsf{X}(\varphi \Rightarrow \psi) \equiv \mathsf{X}\varphi \Rightarrow \mathsf{X}\psi$$
$$\mathsf{X}(\varphi \,\mathsf{U}\, \psi) \equiv (\mathsf{X}\varphi) \,\mathsf{U}\, (\mathsf{X}\psi)$$
$$\mathsf{F}(\varphi \vee \psi) \equiv \mathsf{F}\varphi \vee \mathsf{F}\psi$$
$$\mathsf{G}(\varphi \wedge \psi) \equiv \mathsf{G}\varphi \wedge \mathsf{G}\psi$$
$$\varphi \,\mathsf{U}\, (\psi \vee \vartheta) \equiv \varphi \,\mathsf{U}\, \psi \vee \varphi \,\mathsf{U}\, \vartheta$$
$$(\varphi \wedge \psi) \,\mathsf{U}\, \vartheta \equiv \varphi \,\mathsf{U}\, \vartheta \wedge \psi \,\mathsf{U}\, \vartheta$$

UWAGA

$$\mathsf{F}(\varphi \wedge \psi) \not\equiv \mathsf{F}\varphi \wedge \mathsf{F}\psi$$
$$\mathsf{G}(\varphi \vee \psi) \not\equiv \mathsf{G}\varphi \vee \mathsf{G}\psi$$



$$TS \models \mathsf{F}a \wedge \mathsf{F}b$$
$$TS \not\models \mathsf{F}(a \wedge b)$$

„Typowe” formuły LTL

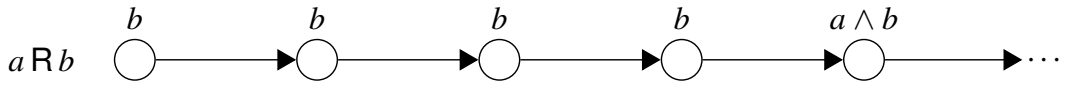
- Osiągalność ($\mathsf{F}\varphi$) oznacza, że pewna szczególna sytuacja może zajść, czyli że pewien stan jest osiągalny z wyróżnionego stanu początkowego;
- Bezpieczeństwo ($\mathsf{G}\neg\varphi$) oznacza, że pewna sytuacja nigdy nie zajdzie, czyli nigdy nie stanie się nic złego;
- Żywotność ($\mathsf{GF}\varphi$ lub $\mathsf{G}(\alpha \Rightarrow F\beta)$) oznacza, że pewna sytuacja kiedyś zajdzie, czyli że zawsze coś dobrego się wydarzy;
- Stabilizacja ($\mathsf{FG}\varphi$) oznacza, że od pewnego momentu w przyszłości pewna sytuacja (opisana przez φ) nie zmieni się; inaczej mówiąc, sytuacja opisana przez $\neg\varphi$ będzie trwała w systemie skończenie długo.

Operatory W (słabe until) i R (release)

$$\varphi\,\mathsf{W}\,\psi = (\varphi\,\mathsf{U}\,\psi) \vee \mathsf{G}\varphi$$

W porównaniu do operatora U, operator W dopuszcza możliwość, że cały czas będzie zachodzić φ i ψ nigdy nie zajdzie. Operator U wymaga osiągnięcia stanu, w którym zachodzi ψ.

$$\varphi\,\mathsf{R}\,\psi = \neg(\neg\varphi\,\mathsf{U}\,\neg\psi)$$



Formuła $a\,\mathsf{R}\,b$ jest spełniona, jeżeli b zawsze zachodzi lub też b zachodzi do momentu, w którym również prawdziwe jest a.

Dla słowa $\sigma = A_0A_1A_2\ldots \in \left(2^{AP}\right)^{\omega}$,
$\sigma \models \varphi\,\mathsf{R}\,\psi$ wtw, gdy
$\forall_{j\geqslant 0}\ \sigma[j\ldots] \models \psi$ lub $(\exists_{i\geqslant 0}\ \sigma[i\ldots] \models \varphi$ i $\forall_{k\leqslant i}\ \sigma[k\ldots] \models \psi)$

Sprawiedliwość (1)

Niech Φ i Ψ będą formułami rachunku zdań nad zbiorem, AP.

- Ograniczenie bezwarunkowej sprawiedliwości jest formułą LTL postaci

$$ufair = \mathsf{GF}\Psi.$$

- Ograniczenie silnej sprawiedliwości jest formułą LTL postaci

$$sfair = \mathsf{GF}\Phi \Rightarrow \mathsf{GF}\Psi.$$

- Ograniczenie słabej sprawiedliwości jest formułą LTL postaci

$$wfair = \mathsf{FG}\Phi \Rightarrow \mathsf{GF}\Psi.$$

Założenie dotyczące sprawiedliwości jest koniunkcją ograniczeń sprawiedliwości w LTL danego typu (np. koniunkcja ograniczeń bezwarunkowej sprawiedliwości).

$$fair = ufair \wedge sfair \wedge wfair$$

Sprawiedliwość (2)

- $FairPaths(s) = \{\pi \in Paths(s) : \pi \models fair\}$, dla $s \in S$
- $FairTraces(s) = \{trace(\pi) : \pi \in FairPaths(s)\}$

Niech s będzie stanem systemu tranzycyjnego TS bez stanów terminalnych, φ formułą LTL i $fair$ założeniem LTL dotyczącym sprawiedliwości.

$$s \models_{fair} \varphi \quad \text{wtw, gdy} \quad \forall_{\pi \in FairPaths(s)}\ \pi \models \varphi$$
$$TS \models_{fair} \varphi \quad \text{wtw, gdy} \quad \forall_{s_0 \in I}\ s_0 \models_{fair} \varphi$$

Przyjmijmy, że w systemie złożonym z procesów $Proc_1$ i $Proc_2$ występuje jeszcze proces *Arbiter*, który losuje (rzut monetą), który proces może wejść w danym cyklu do sekcji krytycznej (stany *head* (reszka) i *tail*).

$$TS \not\models \mathsf{GF}c_1 \wedge \mathsf{GF}c_2$$
$$fair = \mathsf{GF}head \wedge \mathsf{GF}tail$$
$$TS \models_{fair} \mathsf{GF}c_1 \wedge \mathsf{GF}c_2$$

Twierdzenie

$$TS \models_{fair} \varphi \quad \text{wtw, gdy} \quad TS \models (fair \Rightarrow \varphi)$$

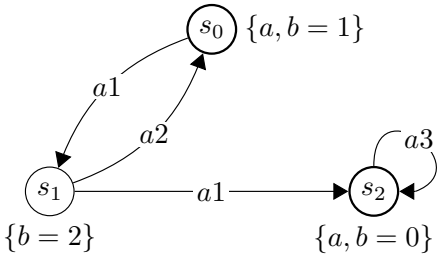
- nuXmv (następca NuSMV) – narzędzie dostarcza język opisu systemów skończenie i nieskończenie stanowych oraz narzędzia do weryfikacji modelowej takich systemów.
- Darmowa licencja dla użytku niekomercyjnego: `https://nuxmv.fbk.eu`
- Specyfikacja własności dostępna dla logik LTL i CTL.
- Zachowano kompatybilność z poprzednimi wersjami (NuSMV).

nuXmv 1.0 User Manual

Język nuXmv (bez etykiet łuków)

```
MODULE main
VAR
  s : {s0, s1, s2};
  a : boolean;
  b : 0 .. 2;
ASSIGN
  init(s) := {s0, s2};

  next(s) := case
    s = s0 : s1;
    s = s1 : {s0, s2};
    s = s2 : s2;
  esac;

  a := case
    s = s0 : TRUE;
    s = s2 : TRUE;
    TRUE   : FALSE;
  esac;

  b := case
    s = s0 : 1;
    s = s1 : 2;
    TRUE   : 0;
  esac;
```

} set of states
} atomic propositions given implicitly
} initial states
} transition relation
} labelling function

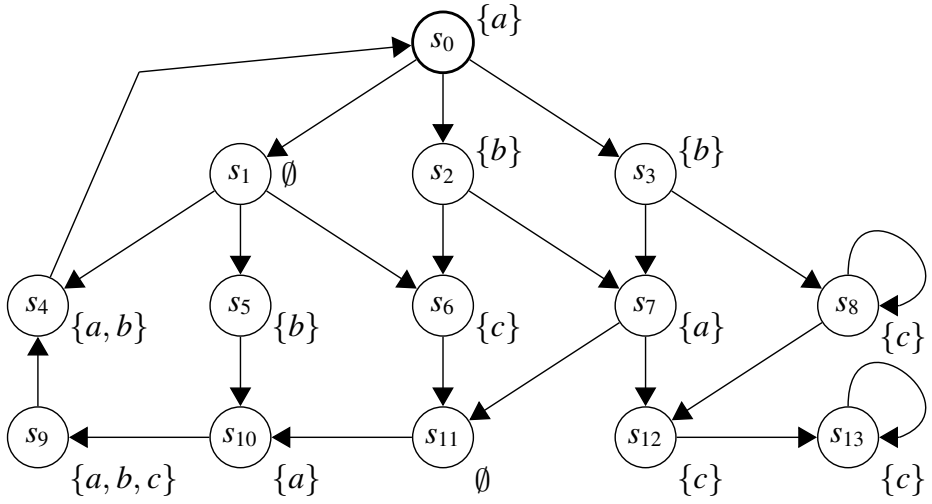


Identyfikatory, typy danych, operatory, wyrażenia

- Nazwy zmiennych mogą zawierać wyłącznie małe i wielkie litery alfabetu łacińskiego, cyfry oraz znaki $, #, – i _. Nazwa musi rozpoczynać się od litery lub znaku podkreślenia i nie może się pokrywać z żadnym ze słów kluczowych.
- Linie zaczynające się od znaków −− są traktowane jako komentarze.
- Podstawowe typy danych: **boolean** (TRUE, FALSE), **integer**, **typy wyliczeniowe** (np. {on, off}, {1, 2, none} – nie mogą zawierać wartości logicznych).
- Operatory logiczne: `!` (negacja), `&` (koniunkcja), `|` (alternatywa), **`xor`** (alternatywa wykluczająca), **`xnor`** (negacja **`xor`**), `->` (implikacja), `<->` (równoważność).
- Operatory relacyjne: `=`, `!=`, `<`, `>`, `<=`, `>=`.
- Operatory arytmetyczne: `-`, `+`, `*`, `/`, **`mod`**.
- Wyrażenie warunkowe: `? :`.
- Wyrażenie **`case`**:

```
next(s) := case
  s = s0 : s1;
  s = s1 : {s1, s2};
  s = s2 : s2;
esac;
```


Przykład (1)




```
s : {s0, s1, s2, s3, s4, s5, s6, s7, s8, s9, s10, s11, s12, s13};
a : boolean;
b : boolean;
c : boolean;
```

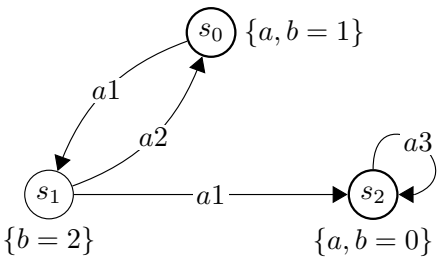
Przykład (2)

```
MODULE main
VAR
  s : {s0, s1, s2, s3, s4, s5, s6,
   s7, s8, s9, s10, s11, s12, s13};
  a : boolean;
  b : boolean;
  c : boolean;
ASSIGN
  init(s) := s0;
  next(s) := case
    s = s0 : {s1, s2, s3};
    s = s1 : {s4, s5, s6};
    s = s2 : {s6, s7};
    s = s3 : {s7, s8};
    s = s4 : s0;
    s = s5 : s10;
    s = s6 : s11;
    s = s7 : {s11, s12};
    s = s8 : {s8, s12};
    s = s9 : s4;
    s = s10 : s9;
    s = s11 : s10;
    s = s12 : s13;
    s = s13 : s13;
  esac;

a := case
  s = s0 : TRUE;
  s = s4 : TRUE;
  s = s7 : TRUE;
  s = s9 : TRUE;
  s = s10 : TRUE;
  TRUE : FALSE;
esac;

b := case
  s = s2 : TRUE;
  s = s3 : TRUE;
  s = s4 : TRUE;
  s = s5 : TRUE;
  s = s9 : TRUE;
  TRUE : FALSE;
esac;

c := case
  s = s6 : TRUE;
  s = s8 : TRUE;
  s = s9 : TRUE;
  s = s12 : TRUE;
  s = s13 : TRUE;
  TRUE : FALSE;
esac
```


Język nuXmv (z etykietami łuków)

```
MODULE main
IVAR
  action: {a1,a2,a3};
VAR
  s : {s0, s1, s2};
  a : boolean;
  b : 0 .. 2;
ASSIGN
  init(s) := {s0, s2};

  next(s) := case
    s = s0 & action = a1: s1;
    s = s1 & action = a2: s0;
    s = s1 & action = a1: s2;
    s = s2 & action = a3: s2;
    TRUE: s;
  esac;

  a := case
    s = s0 : TRUE;
    s = s2 : TRUE;
    TRUE   : FALSE;
  esac;

  b := case
    s = s0 : 1;
    s = s1 : 2;
    TRUE   : 0;
  esac;
TRANS s = s0 -> (action = a1)
TRANS s = s1 -> (action = a1
                 | action = a2)
TRANS s = s2 -> (action = a3)
```

} set of actions
} set of states
} atomic propositions given implicitly
} initial states
} transition relation
} labelling function
} transitions' labels



nuXmv – symulacja (1)

```
-- uruchomienie w trybie interaktywnym
nuXmv -int

-- wczytanie modelu i iniicjalizacja
read_model -i mc02.smv
go

-- wybór stanu początkowego (losowo)
pick_state -r
print_current_state -v
-- Current state is 1.1
-- s = s0
-- a = TRUE
-- b = FALSE
-- c = FALSE

-- losowa symulacja (10 kroków) i wyświetlenie ścieżki
simulate -r -k 10
show_traces -v

-- symulacja począwszy od wskazanego stanu
goto_state 1.8
simulate -r -k 12
show_traces 2
```


nuXmv – symulacja (2)

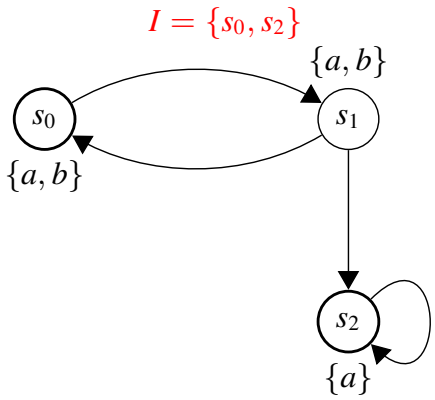
```
-- interaktywny wybór stanu początkowego
pick_state -i
-- wyświetla wszystkie możliwe stany początkowe

-- symulacja interaktywna
simulate -i -a -k 3
-- na każdym etapie wyświetlana jest lista następników do wyboru

-- symulacja interaktywna z ograniczeniem dla następników
goto_state 1.1
simulate -c "c = FALSE" -i -a -k 7
-- na każdym etapie wyświetlana jest lista następników do wyboru,
-- symulacja jest przerywana, jeżeli nie istnieje następnik
-- spełniający warunek

-- zakończenie pracy
quit
```

NuMSV – weryfikacja

```
NuSMV -int mc02.smv
go

-- wyznaczenie zbioru osiągalnych stanów
compute_reachable

-- wydruk listy osiągalnych stanów
print_reachable_states -v

-- sprawdzenie występowania stanów martwych
check_fsm

-- sprawdzenie własności
NuSMV > check_ltlspec -p "G F a"
-- specification  G ( F a)  is false
-- as demonstrated by the following execution sequence
Trace Description: LTL Counterexample
Trace Type: Counterexample
-> State: 1.1 <-
-- ...

-- wykonanie poleceń zawartych w pliku
source mc02ltl.txt
```


Specyfikowanie własności w pliku z modelem

```
1  MODULE main
2  VAR
3    s : {s0, s1, s2};
4    a : boolean;
5    b : boolean;
6  ASSIGN
7    init(s) := {s0, s2};
8    next(s) := case
9      s = s0 : s1;
10     s = s1 : {s0, s2};
11     s = s2 : s2;
12   esac;
13
14   a := TRUE;
15   b := case
16     s = s0 : TRUE;
17     s = s1 : TRUE;
18     TRUE : FALSE;
19   esac;
20
21 LTLSPEC G a
22 LTLSPEC X (a & b)
23 LTLSPEC G (!b -> G(a & !b))
24 LTLSPEC b U (a & !b)
```




```
nuXmv ex4.smv
```

Operatory przeszłości

O – **O**nce – kiedyś w przeszłości

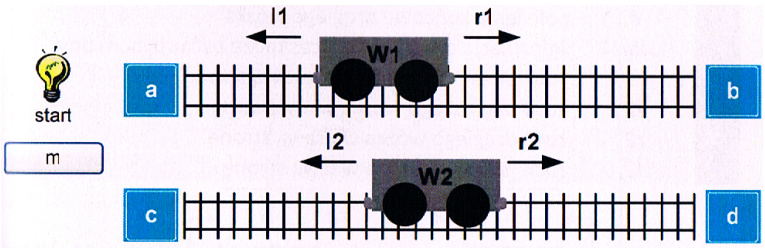
H – **H**istorically – zawsze w przeszłości

S – **S**ince – od

Y – **Y**esterday – w poprzednim stanie

Op	kiedyś w przeszłości p było spełnione
Hp	zawsze w przeszłości p było spełnione
p S q	p zachodzi od czasu, gdy ostatnio zachodziło q (od następnego stanu w odniesieniu do tego, w którym było spełnione q)
Yp	p było spełnione w poprzednim stanie

Przykład



W chwili początkowej wózki znajdują się odpowiednio w punktach a i c. Po naciśnięciu przycisku m rozpoczynają (równolegle) przejazd w prawą stronę. Poruszają się tak długo, aż osiągną odpowiednio punkty b i d (niekoniecznie w tym samym czasie). Następnie pierwszy wózek rozpoczyna ruch w lewą stronę. Gdy dotrze do punktu a, powrót rozpoczyna drugi wózek.

- $s0$ – stan początkowy,
- $s1$ – ruch obu wózków w prawo,
- $s2$ – ruch wózka $w1$ w prawo (wózek $w2$ w punkcie d),
- $s3$ – ruch wózka $w2$ w prawo (wózek $w1$ w punkcie b),
- $s4$ – ruch wózka $w1$ w lewo,
- $s5$ – ruch wózka $w2$ w lewo.

wozki.smv