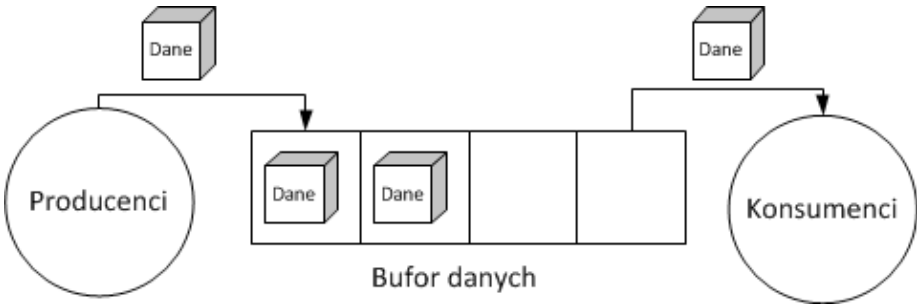
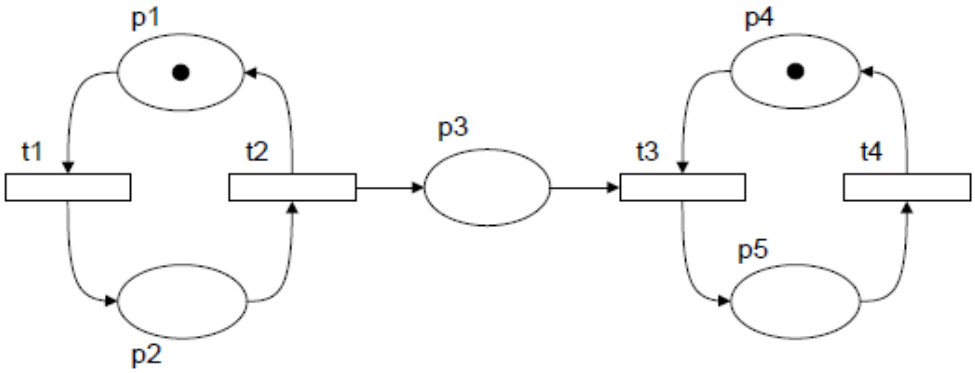
Mini-słownik pojęć: Weryfikacja modelowa

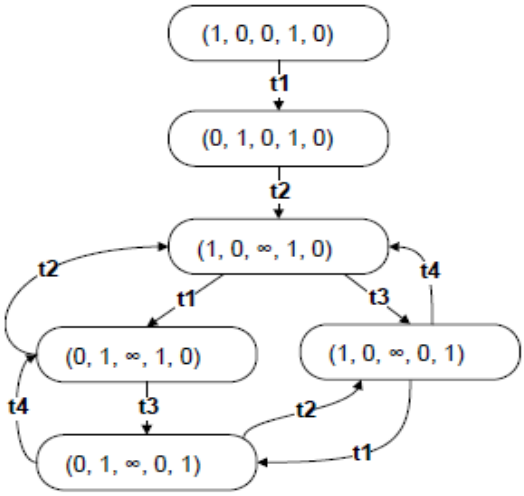
1. Problem (Producent-konsument [wiki])



2. Model (Sieć Petriego [wiki][opis])



3. System tranzycyjny [System tranzycyjny] [wiki]



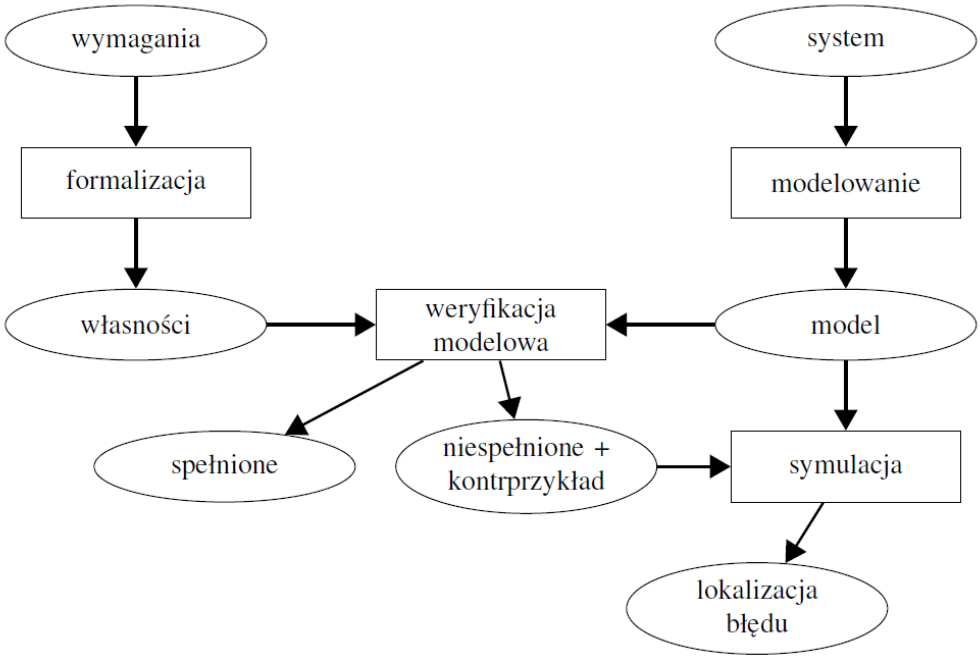
4. Weryfikacja [wiki]

$$XF\,p1 = 1 \;\wedge\; XF\,p1 = 0$$

$$XF\,p4 = 1 \;\wedge\; XF\,p4 = 0$$

Weryfikacja modelowa [wiki]

- Weryfikacja modelowa **opiera się na analizie modelu** przygotowanego z użyciem wybranej **metody formalnej[wiki][wiki_eng]** (model ma jednoznaczną semantykę).
- Dla przygotowanego modelu **analizowany jest zbiór wszystkich możliwych jego stanów**.
- Jednocześnie formalne wyrażenie specyfikacji systemu pozwala wyeliminować z niej błędy takie jak: niekompletność, dwuznaczność itp.



Możliwe wyniki weryfikacji:

1) Własność jest spełniona – Po pozytywnym sprawdzeniu wszystkich własności mamy pewność, że model jest zgodny ze specyfikacją.

2) Własność nie jest spełniona – Kontrprzykładem w takiej sytuacji jest stan, w którym badana własność nie jest spełniona oraz ścieżka pokazująca w jaki sposób można uzyskać ten stan zaczynając od stanu początkowego.

3) Model jest za duży, by możliwe było jego przeanalizowanie – Problem eksplozji stanów może uniemożliwić wyznaczenie zbioru wszystkich możliwych stanów ze względu na możliwości techniczne sprzętu komputerowego.

Rozwiązaniem w takim przypadku może być budowa modelu na wyższym poziomie abstrakcji (pomijamy szczegóły nieistotne z punktu widzenia analizowanych własności) lub zmiana metody weryfikacji [link].

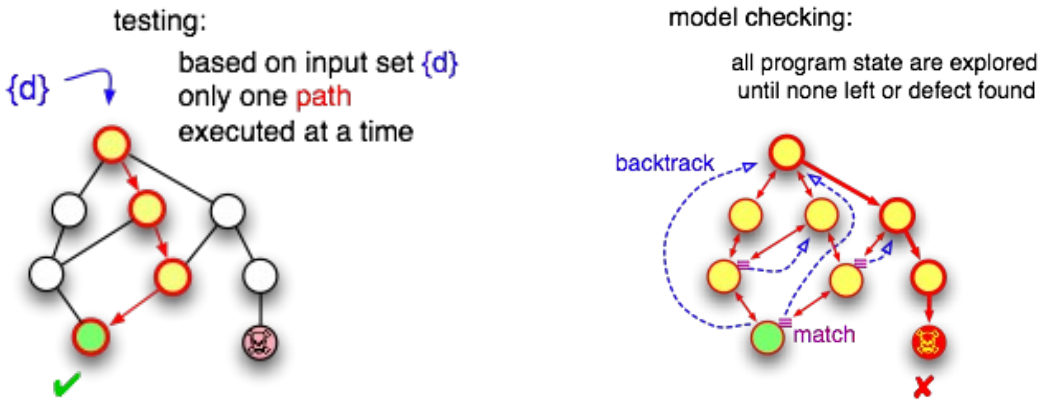
Weryfikacja a walidacja [wiki]:

- **Weryfikacja** – sprawdzenie, czy model jest zgodny ze specyfikacją – *check that we are building the thing right*. [wiki]
- **Walidacja** – sprawdzenie, czy model jest zgodny z jego nieformalnym opisem – *check that we are building the right thing* [wiki]

O ile dzięki **weryfikacji** projektant uzyskuje informacje o z**godności systemu symulacyjnego z jego założeniami**, o tyle **walidacja weryfikuje zgodność jego wizji z realnym światem**. Obie te fazy wzajemnie się uzupełniają i jako takie czasami przedstawiane są wspólnie jako faza oceny adekwatności modelu

Zalety weryfikacji modelowej:

- Jest to **uniwersalna technika**, którą można stosować dla szerokiego spektrum systemów informatycznych.
- Możliwa jest weryfikacja tyko wybranych własności (nie jest konieczna znajomość kompletnej specyfikacji).
- W przeciwieństwie do symulacji i testowania **sprawdzane są wszystkie możliwości**.



- W przypadku, gdy własność nie jest spełniona, dostarcza istotnych informacji pozwalających **znaleźć przyczynę** niepowodzenia.
- Jej stosowanie nie wymaga dużego zaangażowania użytkownika, ani też czasochłonnego przygotowania teoretycznego.
- Wzrasta zainteresowanie weryfikacją modelową ze strony dużych korporacji, czego efektem są m. in. komercyjne narzędzia wspierające jej stosowanie.
- Stosunkowo łatwo można zintegrować weryfikację modelową ze stosowanymi cyklami rozwoju oprogramowania. **Jej zastosowanie może skrócić czas wytwarzania oprogramowania i poprawić jego jakość.**

Wady weryfikacji modelowej:

- Nie może być stosowana (lub stosowana w ograniczonym zakresie) dla systemów z **nieskończoną liczbą stanów**
- **Weryfikacja dotyczy modelu, a nie samego systemu.**
- **Eksplozja stanów** może uniemożliwić weryfikację modelu (Weryfikacja modelowa lepiej sprawdza się dla systemów zorientowanych na przepływ sterowania niż na przepływ danych)[link].
- **Wymaga umiejętności i doświadczenia w budowaniu modeli** na odpowiednim poziomie abstrakcji i umiejętności zapisu wymagań jako formuł logicznych.

Podstawowy podział metod weryfikacji modelowej:

- **Weryfikacja jawnej reprezentacji** (Explicit representation model checking) – Stany i połączenia między nimi reprezentowane są jawnie. Podczas weryfikacji sprawdzane są dokładnie wszystkie stany. Problem eksplozji stanów [opis].

- **Weryfikacja symboliczna** (Symbolic model checking) – wewnętrzna reprezentacja zbiorów stanów i połączeń pomiędzy nimi jako formuł logicznych. Operacje wykonywane są na zbiorach stanów, a nie na pojedynczych stanach, co zdecydowanie przyspiesza weryfikację i zmniejsza ograniczenia związane z eksplozją stanów [opis][opis2].

- **Weryfikacja ograniczona** (Bounded model checking) – Sprawdzane są formuły na ścieżkach o ograniczonej długości [opis][opis2].

Rodzaje sprawdzanych własności:

- **poprawność funkcjonalna** - odpowiedź na pytanie *czy system spełnia swoje zadanie?*
- **osiągalność** - *czy istnieje ścieżka prowadząca do danego stanu systemu?*
- **bezpieczeństwo** - *nic złego nigdy się nie stanie*
- **żywotność** - *coś dobrego kiedyś się wydarzy*
- **sprawiedliwość** - *czy pod pewnymi warunkami określone wydarzenie jest powtarzalne?*
- **własności czasu rzeczywistego** - *czy system reaguje w czasie?*

Języki specyfikacji własności - Weryfikacja modelowa systemu informatycznego wymaga wyspecyfikowania jego własności w sposób dokładny i niedwuznaczny. Własności te są opisywane za pomocą języków specyfikacji własności, do których zaliczamy m. in.**:**

- **Logiki temporalne -** rozszerzenie tradycyjnego rachunku zdań zawierającym dodatkowo operatory odnoszące się do zachowania systemu w czasie, przy czym czas nie jest wprowadzany w sposób jawny, ważna jest wyłącznie kolejność osiągania poszczególnych stanów [wiki].

  Najpopularniejszymi logikami temporalnymi są:

  - LTL - Logika czasu liniowego [LTL][wiki_pl][wiki_eng]
  - CTL - Logika czasu rozgałęzionego [CTL][wiki_pl][wiki_eng]
  - CTL* - Rozszerzenie logiki CTL [CTL*][wiki]
  - RTCTL – Logika CTL wzbogacona o ograniczenia „czasowe” [RTCTL][wyklad]
  - Mu-calculus – Logika modalna zawierająca operatory najmniejszego i największego punktu stałego [wiki]

- **Asercje** [wiki] – możliwość specyfikowania pożądanych stanów systemu bezpośrednio w kodzie, za pomocą asercji.

- **Inne**:
  - PSL – język służący do specyfikacji własności i asercji.
  - Języki specyficzne dla konkretnych narzędzi.

Narzędzia:

Dziedziny zastosowań:

- **sprzęt:**
  - nuXmv (poprzednio znane jako NuSMV) – narzędzie do symbolicznej weryfikacji synchronicznych systemów o skończonej i częściowo również nieskończonej liczbie stanów [nuXmv - przykład],
  - CadenceSMV,
  - Murphi
- **protokoły, oprogramowanie systemowe, sterowniki:**
  - CADP - jest pakietem zawierającym szereg modułów funkcjonalnych pozwalających na specyfikację, symulację oraz weryfikację modeli systemów rozproszonych.
  - Spin
- **oprogramowanie – analiza kodu źródłowego:**
  - BLAST (C),
  - PathFinder (Java),
  - Bandera (Java),
  - CBMC (C/C++)
- **systemy zależne od czasu:**
  - Uppaal – zintegrowane środowisko do modelowania, walidacji i weryfikacji systemów czasu rzeczywistego, rozwijane na uniwersytecie w Uppsali.
  - Kronos,
  - DREAM (Distributed Real-time Embedded Analysis Method)
- **systemy stochastyczne:**
  - PRISM – umożliwia analizę systemów wykazujących zachowania losowe lub probabilistyczne.
- I wiele innych ([spis])

Uzupełnienie:

nuXmv - przykład:

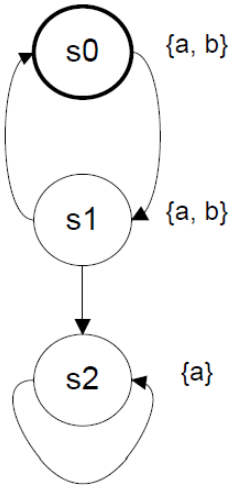
```
MODULE main

VAR
s: {s0, s1, s2};
a: boolean;
b: boolean;

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

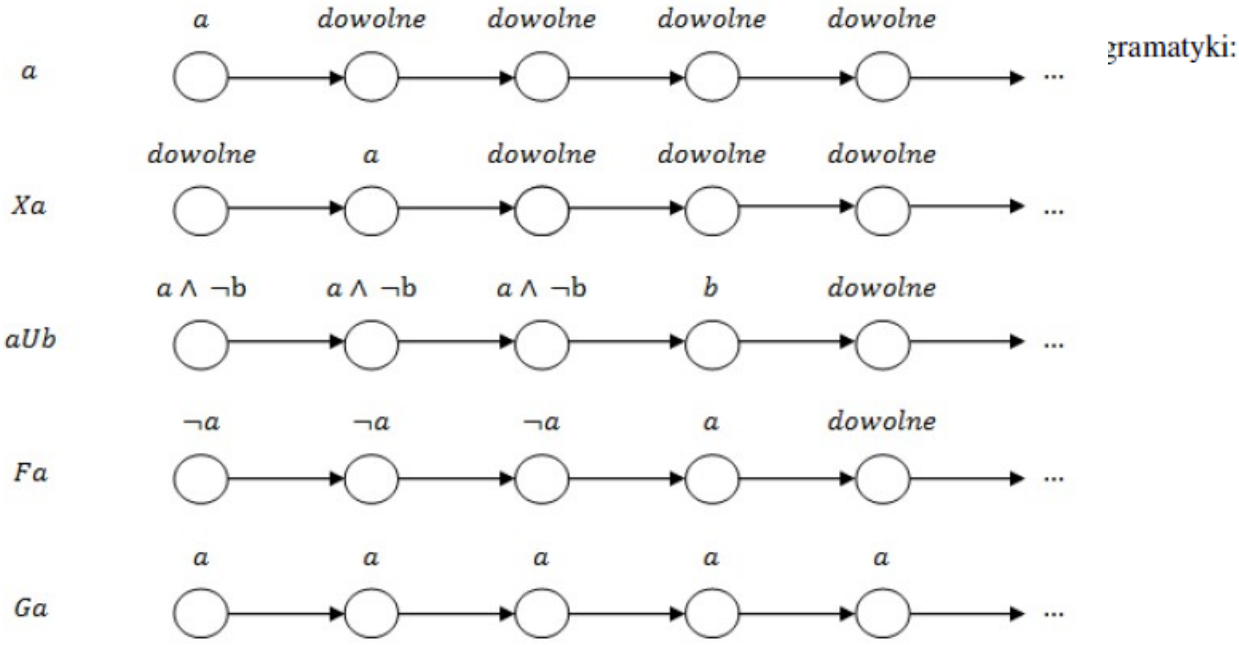
```
a := case
s = s0 : TRUE;
s = s1 : TRUE;
s = s2 : TRUE;
TRUE : FALSE;
esac;
b := case
s = s0 : TRUE;
s = s1 : TRUE;
TRUE : 0;
esac;

LTLSPEC F !a
CTLSPEC AG EF !a
```



CADP Evaluator [src]- narzędzie pozwalające na weryfikację modelową podanego systemu LTS z wykorzystaniem własności zakodowanych przy pomocy logiki temporalnej *regular alternation-free μ-calculus.* Logika ta pozwala na łatwe specyfikowanie własności bezpieczeństwa, żywotności i sprawiedliwości. EVALUATOR pozwala na generowanie zarówno przykładów jak i kontrprzykładów dla weryfikowanych formuł pozwalających wyjaśnić dlaczego dana formuła jest lub nie jest spełniona.

LTL - (ang. Linear Temporal Logic) jest formalizmem dostosowanym do specyfikowania LT-własności. Logika zakłada liniową strukturę czasu. Czas jest zbiorem dyskretnym, w którym każdy punkt ma dokładanie jeden następnik.



gramatyki:

CTL - (ang. Computation Tree Logic) jest logiką czasu rozgałęzionego pozwalającą specyfikować własności systemu. Model czasu w tej logice jest strukturą drzewiastą, w której przyszłość nie jest

jednoznacznie określona. W przyszłości istnieje wiele potencjalnych ścieżek, z których każda może być wykonana.

CTL posiada dwa rodzaje formuł: formuły stanowe, będące asercjami atomicznych wyrażeń logicznych, dotyczących stanów oraz formuły ścieżkowe, określające własności temporalne ścieżek.

Formuły stanowe w CTL nad zbiorem AP są zdaniami atomicznymi, tworzonymi zgodnie z następującą gramatyką:

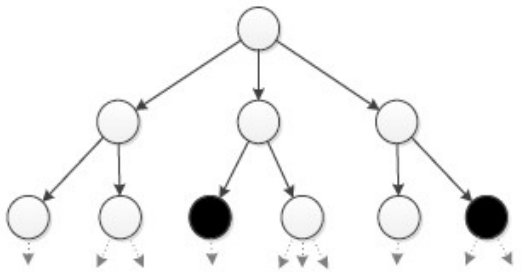
$$\Phi ::= true \mid a \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid \exists\varphi \mid \forall\varphi$$

gdzie $a \in AP$ oraz φ jest formułą ścieżkową.

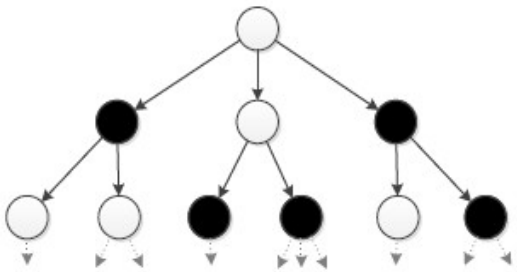
Formuły ścieżkowe w CTL są tworzone według następującej gramatyki:

$$\varphi ::= \mathrm{X}\Phi \mid \Phi_1 \,\mathrm{U}\, \Phi_2$$

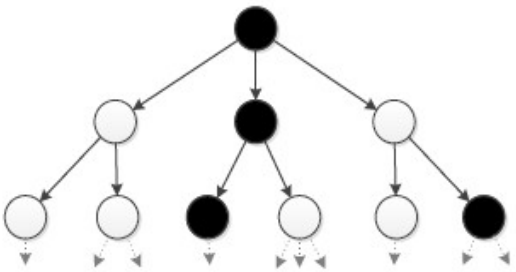
gdzi



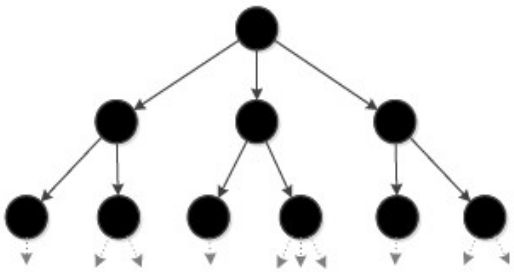
EF *czarny*



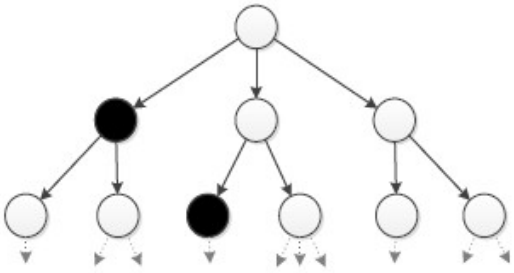
AF *czarny*



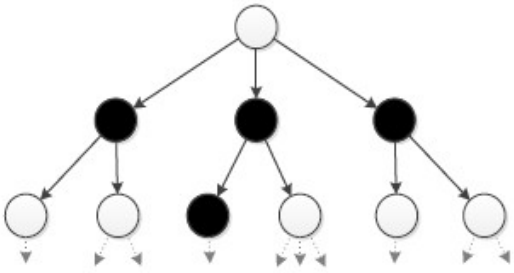
EG *czarny*



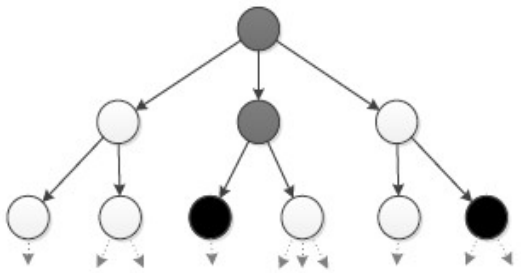
AG *czarny*

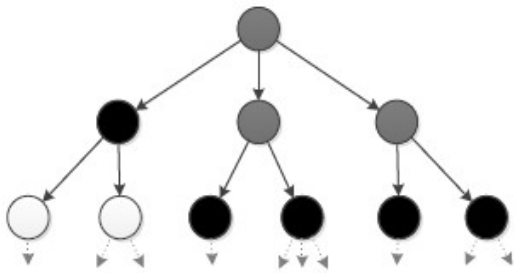


EX *czarny*

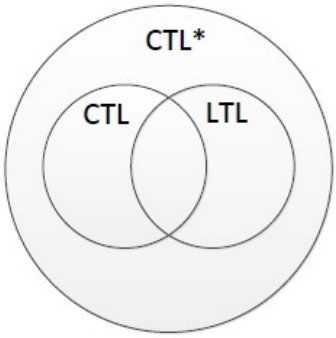


AX *czarny*

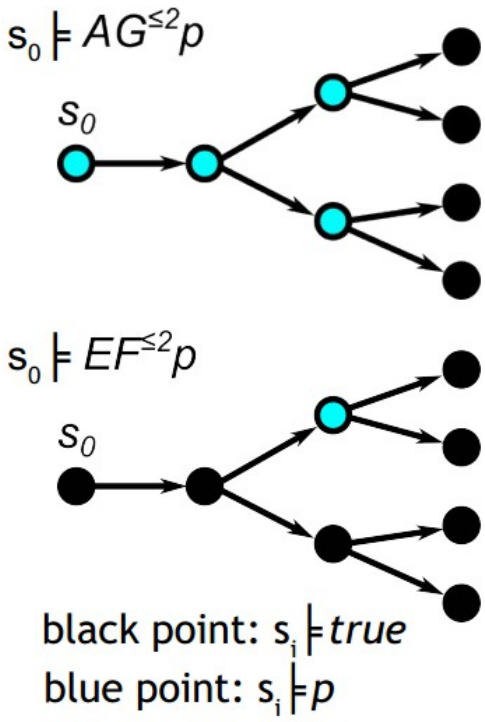




CTL* - Logika temporalna łącząca ekspresywność zarówno logiki LTL, jak i CTL poprzez zlikwidowanie ograniczenia CTL, mówiącego o tym, że przed każdym operatorem temporalnym (**X**, **U**, **F**, **G**) musi wystąpić operator ścieżkowy (**A**, **E**).



RTCTL – (ang. Real-Time Computation Tree Logic) rozszerzenie logiki temporalnej CTL, w którym każdy operator temporalny posiada ograniczenie. Pozwala to na określenie maksymalnego „dystansu czasowego" pomiędzy dwoma zdaniami na każdej/przynajmniej jednej ścieżce [info].



System tranzycyjny

System tranzycyjny TS jest krotką $(S,\ Act,\ \longrightarrow,\ I,\ AP,\ L)$, gdzie:

- S jest zbiorem stanów,
- Act jest zbiorem akcji,
- $\longrightarrow \subseteq\ S\ \times\ Act\ \times\ S$ jest relacją przejść,
- $I\ \subseteq\ S$ jest zbiorem stanów początkowych,
- AP jest zbiorem formuł atomowych,
- $L\colon S\ \longrightarrow\ 2^{AP}$ jest funkcją etykietującą stany.

TS jest nazywany skończonym, jeżeli S, Act oraz AP są skończone.

Literatura:

- http://en.wikipedia.org/wiki/Model_checking
- http://en.wikipedia.org/wiki/List_of_model_checking_tools
- M. Szpyrka – Wykłady do przedmiotu Wykład monograficzny z matematyki (weryfikacja modelowa), http://home.agh.edu.pl/~mszpyrka/doku.php?id=dydagh:mcdata:mc.
- C. Baier, J-P Katoen - Principles of Model Checking
- A. Biernacka, J. Biernacki - Zastosowanie technik weryfikacji modelowej do analizy własności sieci Petriego
- R. Mrówka – Modelowanie i systematyczna analiza poprawności behawioralnych artefaktów z wykorzystaniem algebry procesów.
- Wenkai Tan – Bounded Model Checking
- https://nuxmv.fbk.eu/
- http://staff.iiar.pwr.wroc.pl/pawel.gluchowski/wp-content/uploads/isa/isa_w6.pdf