Logika dla informatyków

Andrzej M. Borzyszkowski

Instytut Informatyki
Uniwersytet Gdański

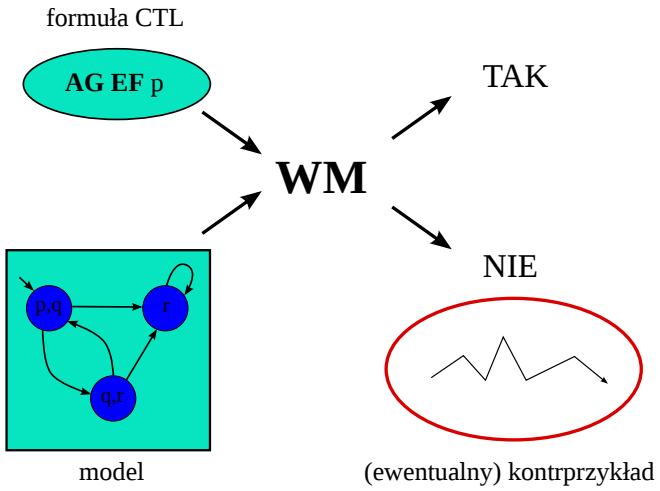
inf.ug.edu.pl/~amb

Algorytmy weryfikacji modelowej

Algorytmy weryfikacji modelowej

- semantyka operuje na systemach tranzycji (grafach przejść)
- algorytm dla logiki **CTL** polega na odpowiednim etykietowaniu wierzchołków grafu przejść
- algorytm dla logiki **LTL** jest nieco bardziej złożony
- dla każdej z logik możemy założyć, że spójniki należą do podzbioru wygodnego dla podania algorytmu

Weryfikacja modelowa CTL



Weryfikacja modelowa CTL

Dwie oczywiste „metody postępowania" w celu weryfikacji

$$\boxed{\mathcal{M}, s_0 \models \varphi}$$

1. $\mathcal{M} + s_0 + \varphi \quad \rightsquigarrow \quad$ TAK/NIE
2. $\mathcal{M} + \varphi \quad \rightsquigarrow \quad \mathrm{OK} = \{s \in S \mid \mathcal{M}, s \models \varphi\}$
   $s_0 \in \mathrm{OK}$?

Okazuje się, że metoda druga da się zrealizować efektywniej

Założenie: ograniczamy się do zestawu: $\bot$, $\neg$, $\wedge$, EX, AF, EU

Algorytm etykietowania wierzchołków grafu przejść

Niech $\mathcal{M} = (S, \to, L)$ będzie strukturą Kripkego.

Dla danej formuły φ logiki **CTL** chcemy znaleźć zbiór:

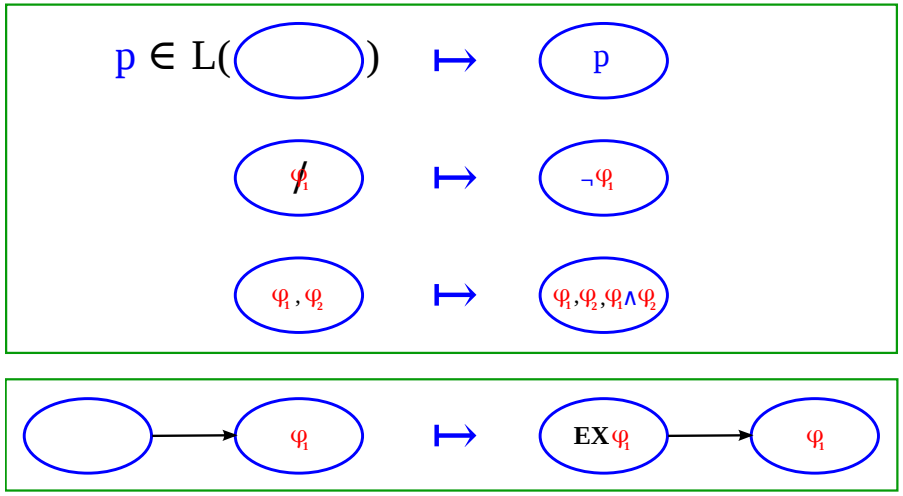
$$\{s \in S \mid \mathcal{M}, s \models \varphi\}$$

Algorytm etykietuje każdy stan $s \in S$ podformułami formuły φ, które są w nim spełnione – oznaczenie $\mathbf{lab}(s)$

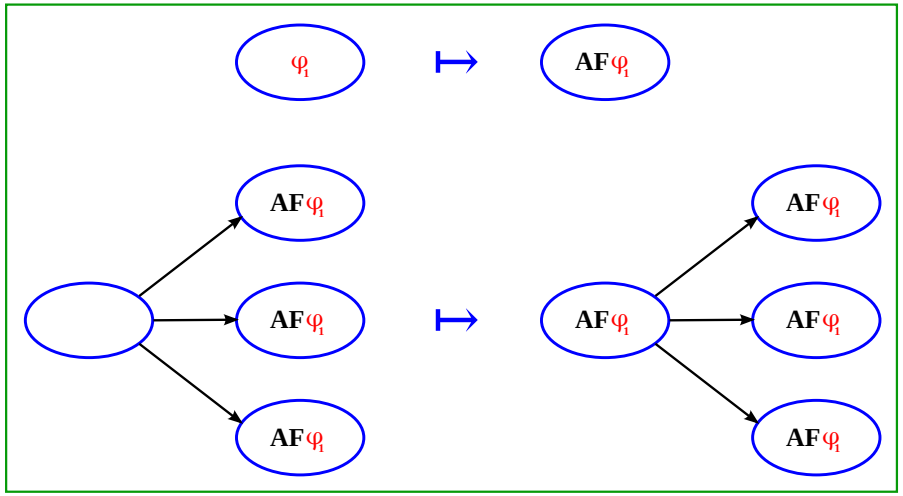
Etykietowanie początkowe: $\mathbf{lab}(s) \mathrel{\widehat{=}} L(s)$

Dla każdej podformuły sposób etykietowania zależy oczywiście od semantyki jej „spójnika głównego"

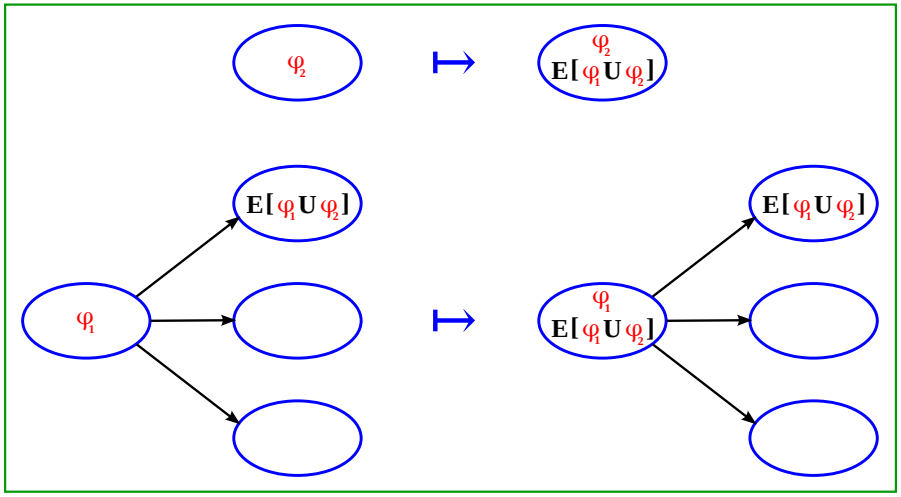
Etykietowanie: spójniki zdaniowe, formuły atomowe i EX



Etykietowanie: formuły AF



Etykietowanie: formuły EU



Warianty efektywniejsze

- zamiast ograniczać się do minimalnego zestawu spójników można jednak rozpatrywać inne też
- np. dla EG można zastosować podejście komplementarne, najpierw wszystkie stany zaetykietować $EG\,\varphi$ potem usuwać etykietę jeśli jest nieuzasadniona
- są jeszcze lepsze pomysły na sprawniejsze etykietowanie

Algorytmy dla logiki CTL – pseudokod

function $\mathtt{SAT}(\varphi)$ /* zbiór stanów w których zachodzi φ */
begin
  case
    $\varphi = \bot$: **return** $\emptyset$
    φ jest atomem : **return** $\{s \in S \mid \varphi \in L(s)\}$
    $\varphi = \neg\varphi_1$: **return** $S - \mathtt{SAT}(\varphi_1)$
    $\varphi = \varphi_1 \wedge \varphi_2$: **return** $\mathtt{SAT}(\varphi_1) \cap \mathtt{SAT}(\varphi_2)$
    $\varphi = EX\varphi_1$: **return** $\mathtt{SAT}_{\mathtt{EX}}(\varphi_1)$
    $\varphi = AF\varphi_1$: **return** $\mathtt{SAT}_{\mathtt{AF}}(\varphi_1)$
    $\varphi = E[\varphi_1\ U\ \varphi_2]$: **return** $\mathtt{SAT}_{\mathtt{EU}}(\varphi_1, \varphi_2)$
  end case
end function

Pseudokod dla formuł EX

function $\mathtt{SAT}_{\mathtt{EX}}(\varphi_1)$ /* $= \mathtt{SAT}(EX\varphi_1)$ */
local var X, Y
begin
  $X := \mathtt{SAT}(\varphi_1)$;
  $Y := \mathtt{pre}_\exists(X)$;
  return Y
end

- stany, których (niektóre) następniki spełniają φ_1
- $\mathtt{pre}_\exists(X) = \{s \in S \mid s \to s', s' \in X \text{ dla pewnego } s'\}$

Pseudokod dla formuł EU

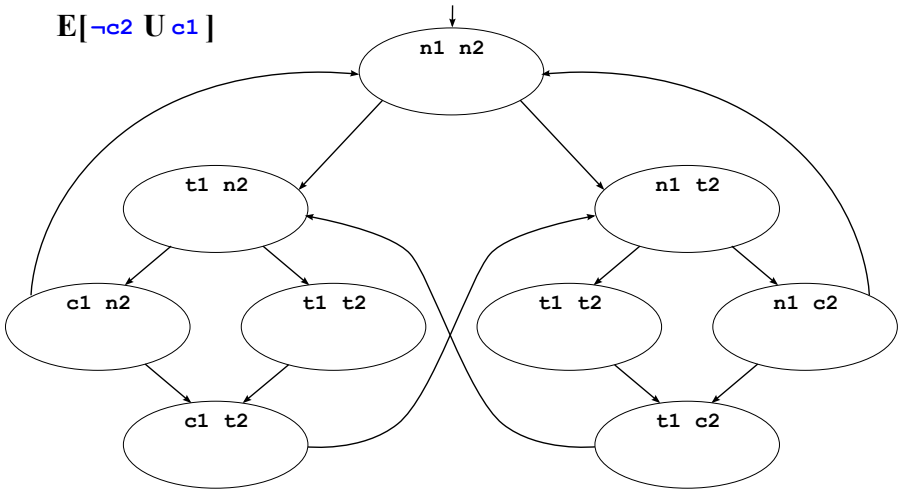
function $\mathtt{SAT}_{EU}(\varphi_1, \varphi_2)$ /* $= \mathtt{SAT}(E[\varphi_1\ U\ \varphi_2])$ */
local var W, X, Y
begin
  $W := \mathtt{SAT}(\varphi_1)$;
  $X := \emptyset$;
  $Y := \mathtt{SAT}(\varphi_2)$;
  repeat until $X = Y$
  begin
    $X := Y$;
    $Y := Y \cup (W \cap \mathtt{pre}_{\exists}(Y))$
  end
  return Y
end

Pseudokod dla formuł AF

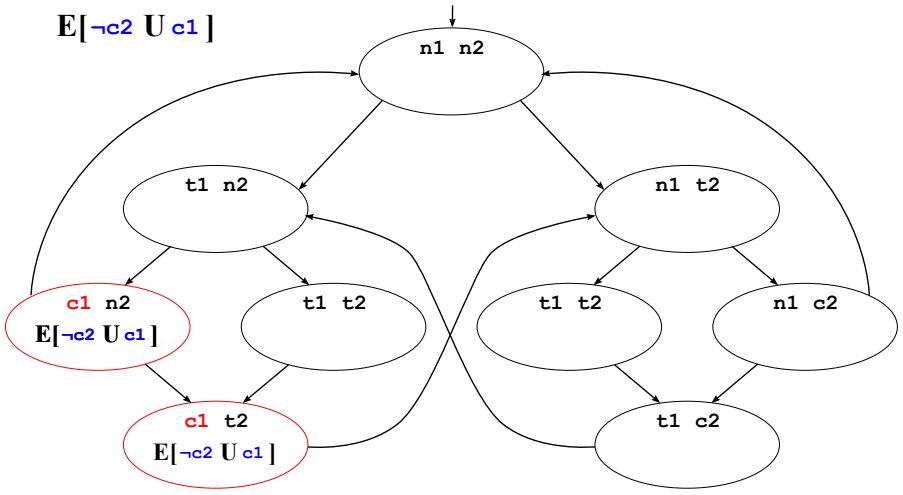
function $\mathtt{SAT}_{AF}(\varphi_1)$ /* $= \mathtt{SAT}(AF\varphi_1)$ */
local var X, Y
begin
  $X := \emptyset$;
  $Y := \mathtt{SAT}(\varphi_1)$;
  repeat until $X = Y$
  begin
    $X := Y$;
    $Y := Y \cup \mathtt{pre}_{\forall}(Y)$
  end
  return Y
end

$$\mathtt{pre}_{\forall}(X) = \{s \in S \mid s \to s' \text{ implikuje } s' \in X\}$$

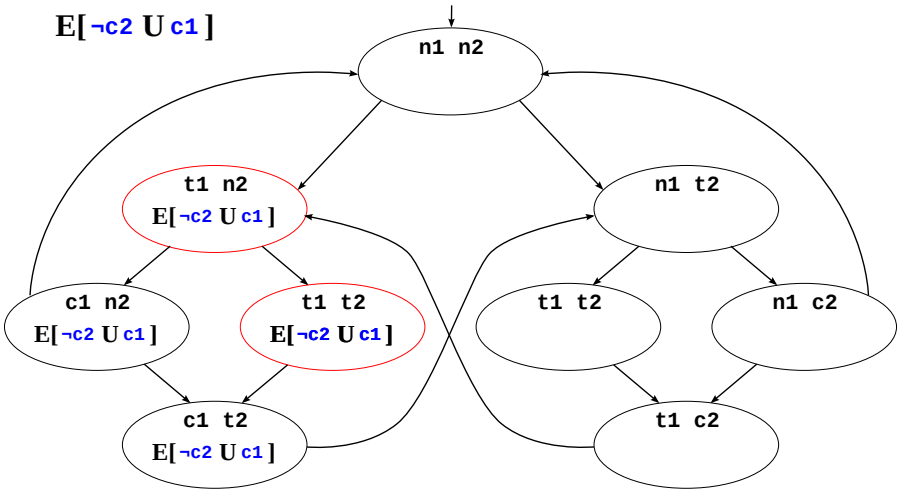
Przykład: *wzajemne wykluczanie*



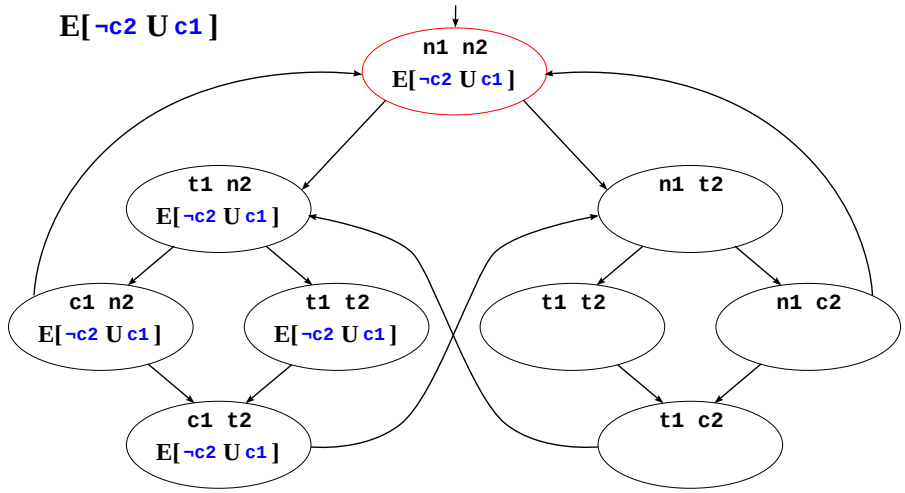
Przykład: *wzajemne wykluczanie*



Przykład: *wzajemne wykluczanie*



Przykład: *wzajemne wykluczanie*



LTL

Weryfikacja własności wyrażonych w LTL

Aby sprawdzić, czy

$$\mathcal{M}, s \models \varphi$$

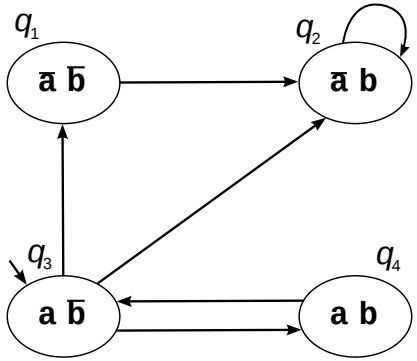
gdzie φ jest własnością wyrażoną w logice **LTL**, nie wystarcza badanie pojedynczych stanów – trzeba rozważać całe ścieżki

W tym celu

- budujemy automat $A_{\neg\varphi}$ kodujący potencjalne naruszenia własności φ
- rozważamy synchroniczne wykonanie $\mathcal{M}$ oraz $A_{\neg\varphi}$

Automat $A_{\neg\varphi}$ „monitoruje" działanie $\mathcal{M}$. Jeśli istnieje obliczenie akceptowane przez „złożenie" $\mathcal{M}$ oraz $A_{\neg\varphi}$, wówczas demonstruje ono naruszenie własności φ.

Przykład specyfikacji niespełnionej



Weryfikujemy $\mathcal{M} \models \neg(\mathbf{a}\ U\ \mathbf{b})$, $\mathcal{M}$ jest powyższym systemem tranzycji

$\neg(\mathbf{a}\ U\ \mathbf{b})$ nie zachodzi dla ścieżek zaczynających się od $q_3, q_2, \ldots$ albo od $q_3, q_4, \ldots$

Będziemy budować automat kodujący zaprzeczenie tej formuły A_{aUb}

Konstrukcja automatu akceptującego ślad

- dana formuła φ zawierająca spójniki temporalne U oraz X
- $\mathcal{C}(\varphi)$ jest zbiorem podformuł φ oraz ich zaprzeczeń
- stany są pozbiorami $q \subseteq \mathcal{C}(\varphi)$ takimi, że
  - dla $\varphi_1 \in \mathcal{C}(\varphi)$ albo $\varphi_1 \in q$ albo $\neg\varphi_1 \in q$
  - $\varphi_1 \vee \varphi_2 \in q$ w.t.w. $\varphi_1 \in q$ lub $\varphi_2 \in q$
  - jeśli $\varphi_1\ U\ \varphi_2 \in q$, to $\varphi_1 \in q$ lub $\varphi_2 \in q$
  - jeśli $\neg(\varphi_1\ U\ \varphi_2) \in q$, to $\neg\varphi_2 \in q$
- stany początkowe zawierają formułę φ
- $q \to q'$ w.t.w.
  - $X\,\varphi_1 \in q$ pociąga $\varphi_1 \in q'$
  - $\neg(X\,\varphi_1) \in q$ pociąga $\neg\varphi_1 \in q'$
  - $\varphi_1\ U\ \varphi_2 \in q$ i $\neg\varphi_2 \in q$ pociąga $\varphi_1\ U\ \varphi_2 \in q'$
  - $\neg(\varphi_1\ U\ \varphi_2) \in q$ i $\varphi_1 \in q$ pociąga $\neg(\varphi_1\ U\ \varphi_2) \in q'$

Automat akceptujący c.d.

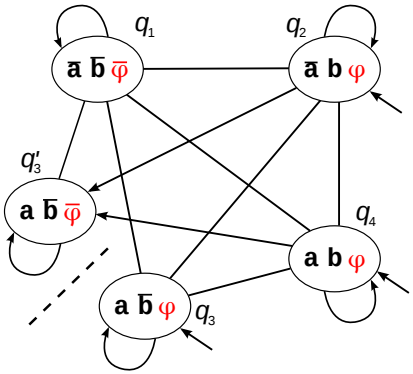
- powyższymi metodami nie da się wykluczyć nieskończonej ścieżki, dla której formuła $\varphi_1\ U\ \varphi_2$ będzie fałszywa, bo ciągle będzie $\varphi_1 \in q$ oraz $\neg\varphi_2 \in q$
- dodatkowo trzeba zażądać, że ślad jest akceptowany dla formuły φ, jeśli istnieje ścieżka, dla której dla wszystkich podformuł $\varphi_1\ U\ \varphi_2$ nieskończenie wiele razy okaże się, że $\neg(\varphi_1\ U\ \varphi_2) \vee \varphi_2 \in q$

Przykład: automat A_{aUb}

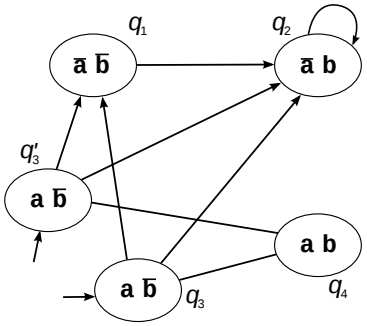
- dla formuły $\varphi = a\ U\ b$ są trzy podformuły $\mathcal{C}(\varphi) = \{a, b, \varphi\}$
- z ośmiu stanów, trzy są wykluczone przez definicję stanu
- wykluczone są też niektóre tranzycje
- ścieżka, która akceptuje ślad nie może na stałe zawierać φ i $\neg b$

Przykład: automat A_{aUb}

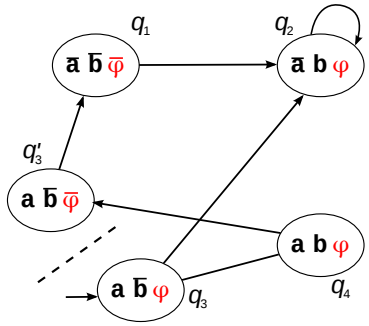
- automat A_{aUb} ma stany dla każdego wartościowania zmiennych a, b
  (brak strzałek oznacza przejście w każdą stronę)
- dla wartościowania $a\bar{b}$ ma dwa stany – jeszcze nie wiadomo, czy formuła φ już zaszła
- φ oznacza, że formuła jest postulowana, $\bar{\varphi}$, że już wiadomo, że nie zachodzi
- żądamy też, by niedozwolone było nieskończone przebywanie w stanie $q_3 = a\bar{b}\varphi$



Przykład: produkt modelu i automatu negującego sprawdzaną formułę



kopia oryginalnego modelu z rozdwojonym stanem początkowym



produkt dwóch systemów tranzycji

(**a** U **b**) zachodzi dla ścieżek zaczynających się od $q_3, q_2, \ldots$ albo od $q_3, q_4, \ldots$ czyli $\neg$(**a** U **b**) nie jest spełniona dla tych ścieżek.