Modelowanie i Analiza Systemów Informatycznych

Logika Temporalna i Automaty Czasowe

(6)

Modelowa weryfikacja systemu w UPPAAL

© Paweł Głuchowski, Politechnika Wrocławska

wersja 2.2

Treść wykładu

Wzorce modelowania

- Kontrolowanie przejścia zegarem
- Synchroniczne przekazanie treści między automatami
- Synchronizacja multicast z określoną liczbą odbiorców
- Pilne przejście
- Regulator czasowy
- Dobre praktyki

Treść wykładu

Przykłady

- Komar
- Trzech uciekinierów
- Algorytm Pettersona
- Dostęp do pliku
- Pociągi i szlaban

Wzorce modelowania

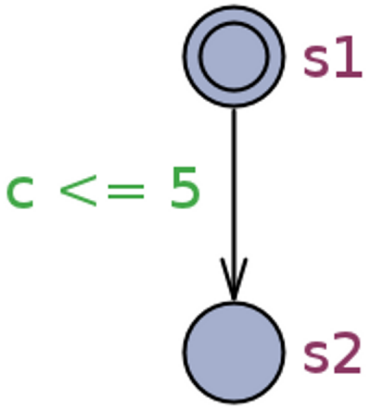
- Kontrolowanie przejścia zegarem
- Synchroniczne przekazanie treści między automatami
- Synchronizacja multicast z określoną liczbą odbiorców
- Pilne przejście
- Regulator czasowy
- Dobre praktyki

Wzorce modelowania

Kontrolowanie przejścia zegarem

Automaty A i B są kontrolowane zegarem *c*. Co je różni?

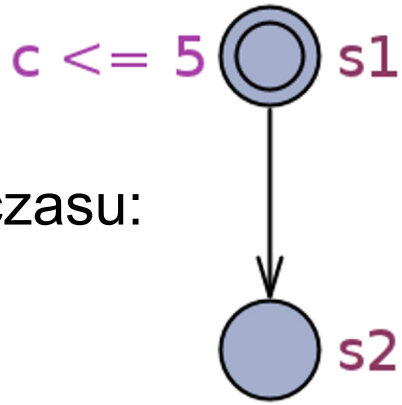
A: zegar kontroluje przejście między stanami s1 i s2:



- przejście s1→s2 jest <u>dostępne</u> póki nie minie 5 jednostek czasu, ale może nigdy nie nastąpić:

E<> A.s1 and c > 5	TAK
A<> A.s2	NIE
E<> A.s2	TAK

B: zegar kontroluje stan s1 (jako *invariant*):



- stan s1 <u>musi</u> zostać opuszczony zanim minie 5 jednostek czasu:

E<> B.s1 and c > 5	NIE
A<> B.s2	TAK

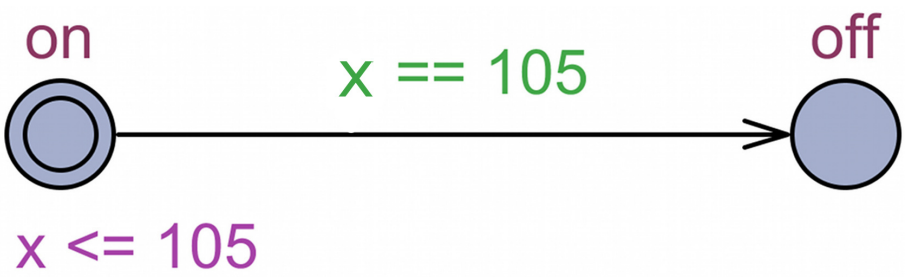
Wzorce modelowania

Kontrolowanie przejścia zegarem

Automat *Wykład* kontrolowany jest zmienną zegarową x.

- Wykład trwa (stan *on*) dokładnie 105 minut:

A<> Wyklad.off	TAK
A[] Wyklad.on imply x <= 105	TAK
A[] Wyklad.off imply x >= 105	TAK
E<> Wyklad.on and x==105	TAK
E<> Wyklad.off and x==105	TAK
Wyklad.on --> Wyklad.off	TAK



Wzorce modelowania

Synchroniczne przekazanie treści między automatami

(na podst. G. Behrmann et al. „A tutorial on UPPAAL”, 2006, str. 28—32)

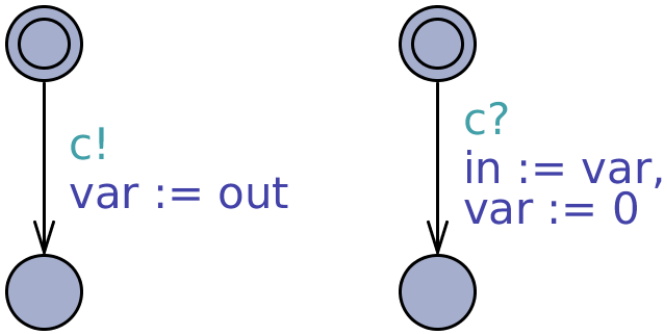
- Idea 1:
  - Synchronizacja automatów odbywa się przez kanały.
  - Dane przekazywane są przez zmienne globalne.
- Dostępne są 4 warianty:
  - synchronizacja jednokierunkowa bezwarunkowa,
  - synchronizacja jednokierunkowa warunkowa,
  - synchronizacja dwukierunkowa bezwarunkowa,
  - synchronizacja dwukierunkowa warunkowa.

Wzorce modelowania

Synchroniczne przekazanie treści między automatami

(na podst. G. Behrmann et al. „A tutorial on UPPAAL”, 2006, str. 28—32)

- Synchronizacja jednokierunkowa bezwarunkowa:
  - zmienna lokalna ***out*** (lewy automat) przekazywana jest do zmiennej lokalnej ***in*** (prawy automat) przez zmienną globalną ***var,***
  - przekazanie następuje przy synchronizacji kanałem ***c***.

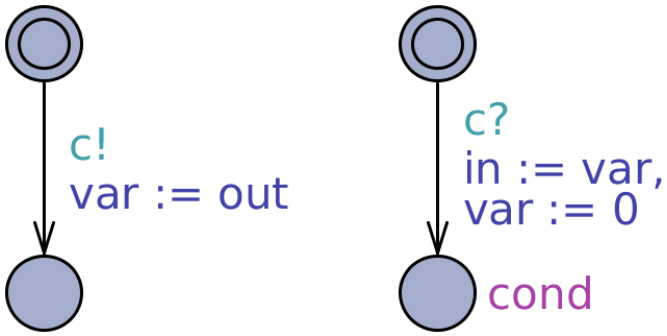


Wzorce modelowania

Synchroniczne przekazanie treści między automatami

(na podst. G. Behrmann et al. „A tutorial on UPPAAL”, 2006, str. 28—32)

- Synchronizacja jednokierunkowa warunkowa:
  - zmienna lokalna ***out*** (lewy automat) przekazywana jest do zmiennej lokalnej ***in*** (prawy automat) przez zmienną globalną ***var,***
  - przekazanie następuje przy synchronizacji kanałem ***c***,
  - synchronizacja następuje jeśli spełniony jest warunek (invariant) ***cond***.

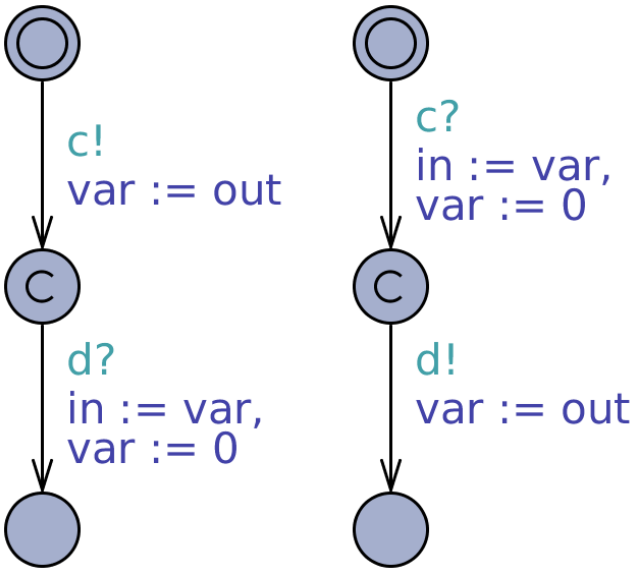


Wzorce modelowania

Synchroniczne przekazanie treści między automatami

(na podst. G. Behrmann et al. „A tutorial on UPPAAL”, 2006, str. 28—32)

- Synchronizacja dwukierunkowa bezwarunkowa:
  - zmienna lokalna ***out*** (lewy automat) przekazywana jest do zmiennej lokalnej ***in*** (prawy automat) przez zmienną globalną ***var,***
  - przekazanie następuje przy synchronizacji kanałem ***c***.
- Jednocześnie:
  - zmienna lokalna ***out*** (prawy automat) przekazywana jest do zmiennej lokalnej ***in*** (lewy automat) przez zmienną globalną ***var,***
  - przekazanie następuje przy synchronizacji kanałem ***d***.

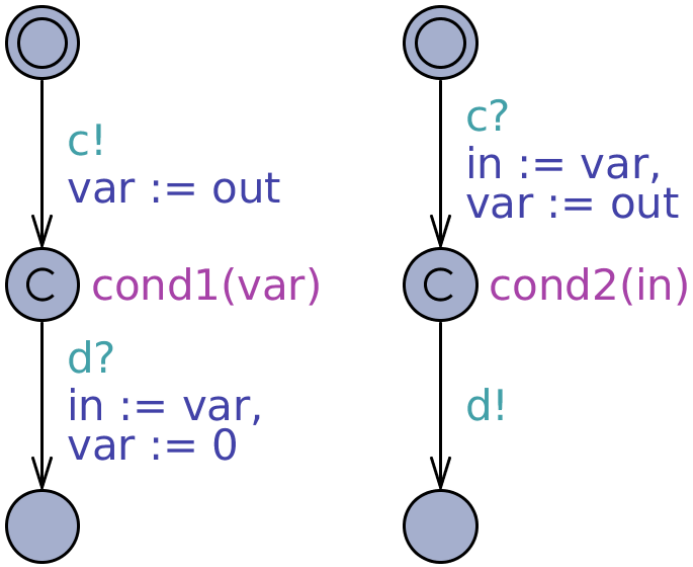


Wzorce modelowania

Synchroniczne przekazanie treści między automatami

(na podst. G. Behrmann et al. „A tutorial on UPPAAL”, 2006, str. 28—32)

- Synchronizacja dwukierunkowa warunkowa:
  - zmienna lokalna ***out*** (lewy automat) przekazywana jest do zmiennej lokalnej ***in*** (prawy automat) przez zmienną globalną ***var;***
  - przekazanie następuje przy synchronizacji kanałem ***c***, pod warunkiem ***cond1*** (prawy automat otrzymuje prawidłową wartość).
- Jednocześnie:
  - zmienna lokalna ***out*** (prawy automat) przekazywana jest do zmiennej lokalnej ***in*** (lewy automat) przez zmienną globalną ***var;***
  - przekazanie następuje przy synchronizacji kanałem ***d***, pod warunkiem ***cond2*** (lewy automat otrzymuje prawidłową wartość).



Wzorce modelowania

Synchroniczne przekazanie treści między automatami

(na podst. G. Behrmann et al. „A tutorial on UPPAAL”, 2006, str. 28—32)

- Idea 2:
  - Synchronizacja automatów odbywa się przez tablicę kanałów.
  - Mała liczba całkowita przekazywana jest przez odpowiedni kanał (jego numer jest tą liczbą).

Wzorce modelowania

Synchroniczne przekazanie treści między automatami

(na podst. G. Behrmann et al. „A tutorial on UPPAAL”, 2006, str. 28—32)

- Przekazywana jest liczba typu int z zakresu [min,max].
- Synchronizacja odbywa się tablicą kanałów:
  chan send[max-min+1];
- Nadawca wysyła liczbę: 2, 3 lub losowo wybraną (*random*).
- Zmienna v otrzymuje przesłaną wartość.

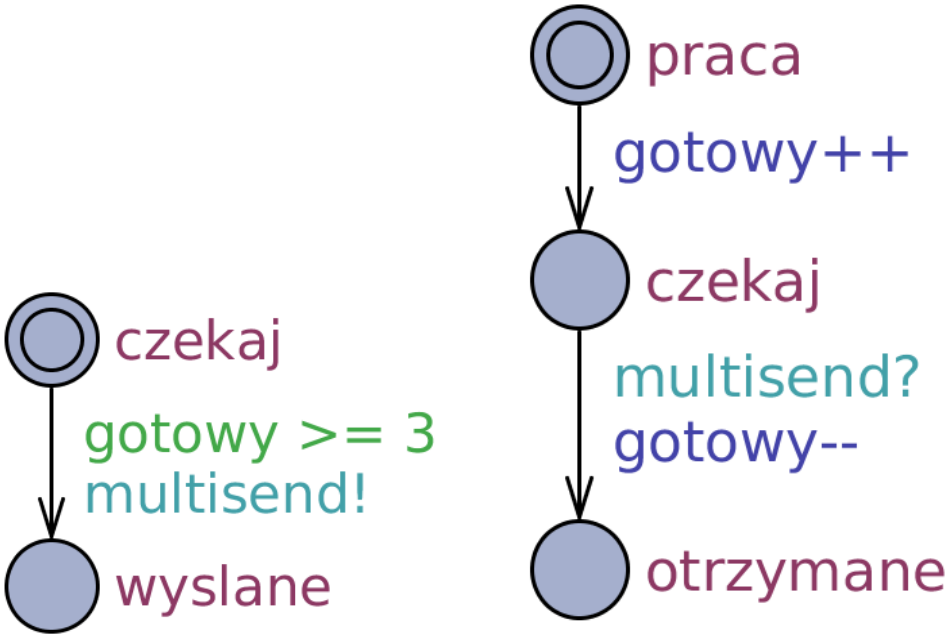
Weryfikacja dla dużej tablicy kanałów może być długotrwała.

Wzorce modelowania

Synchronizacja multicast z określoną liczbą odbiorców

(na podst. G. Behrmann et al. „A tutorial on UPPAAL”, 2006, str. 32—33)

- Przykład: synchronizacja z przynajmniej 3 automatami przez kanał rozgłoszeniowy (*broadcast chan*) *multisend*:
  - Synchronizacja następuje dopiero, gdy przynajmniej 3 odbiorcy na nią czekają (*gotowy* >= 3).
  - Bez tego warunku synchronizacja mogłaby zajść bez odbiorców!

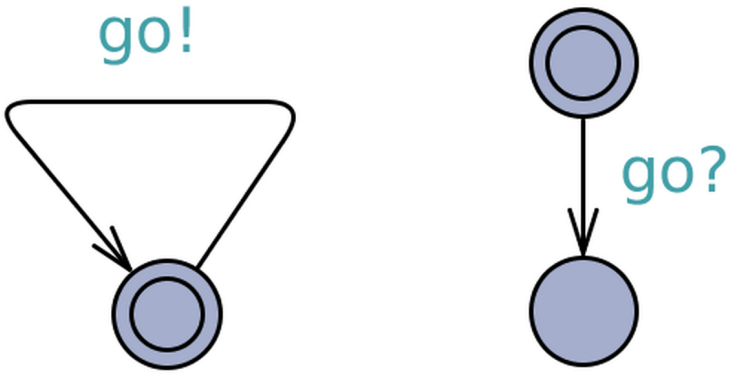


Wzorce modelowania

Pilne przejście

(na podst. G. Behrmann et al. „A tutorial on UPPAAL”, 2006, str. 34—35)

- **Założenie:** stan wyjściowy przejścia nie jest typu *urgent* lub *committed*.
- **Cel:** przejście zostanie wykonane natychmiast, gdy jego stan wyjściowy stanie się aktywny.
- Przejście inicjowane jest przez kanał *urgent chan* przez specjalny automat.



Wzorce modelowania

Regulator czasowy

(na podst. G. Behrmann et al. „A tutorial on UPPAAL”, 2006, str. 36—37)

- Regulator czasowy (*timer*) pozwala m.in.:
  - ograniczyć długość czasu aktywności danego stanu lub grupy stanów,
  - wyznaczyć względny moment synchronizacji automatów.
- Upływ czasu odliczany jest od zadanej wartości (zmiennej lub stałej) do zera.
- Regulator czasowy aktywowany jest w wybranym momencie.
- Osiągnięcie zera może powodować określoną reakcję automatu (np. przez synchronizację).

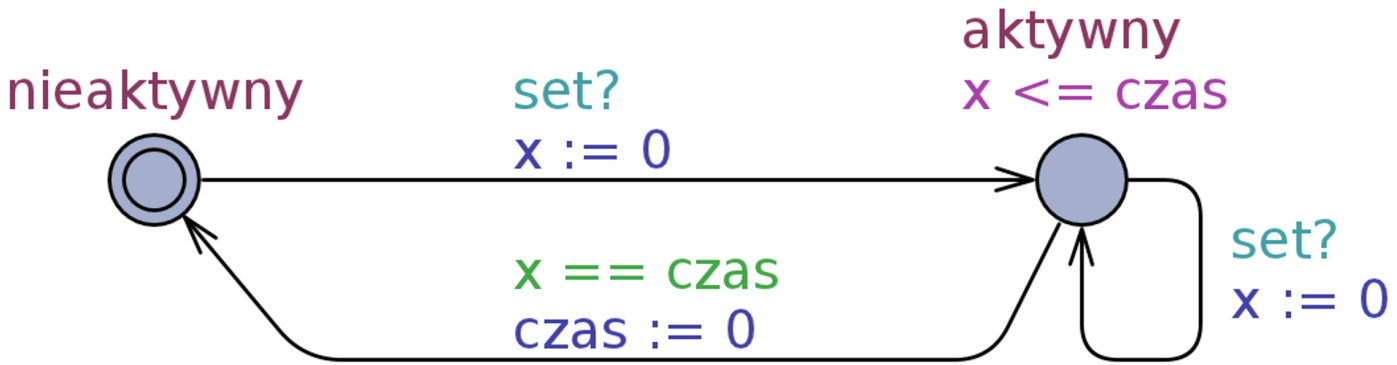
Wzorce modelowania

Regulator czasowy

(na podst. G. Behrmann et al. „A tutorial on UPPAAL”, 2006, str. 36—37)

Regulator czasowy ze zmiennym czasem działania:

- Regulator jest aktywowany kanałem (*chan*) *set*.
- W momencie aktywacji regulatora zerowany jest zegar (*clock*) *x*.
- Zmienna (*int*) *czas* określa długość czasu aktywności regulatora.
- W momencie dezaktywacji regulatora zmienna *czas* jest zerowana.
- Sprawdzenie, czy upłynął odliczany czas: *czas* == *0*.
- Możliwe jest zresetowanie regulatora podczas jego działania.



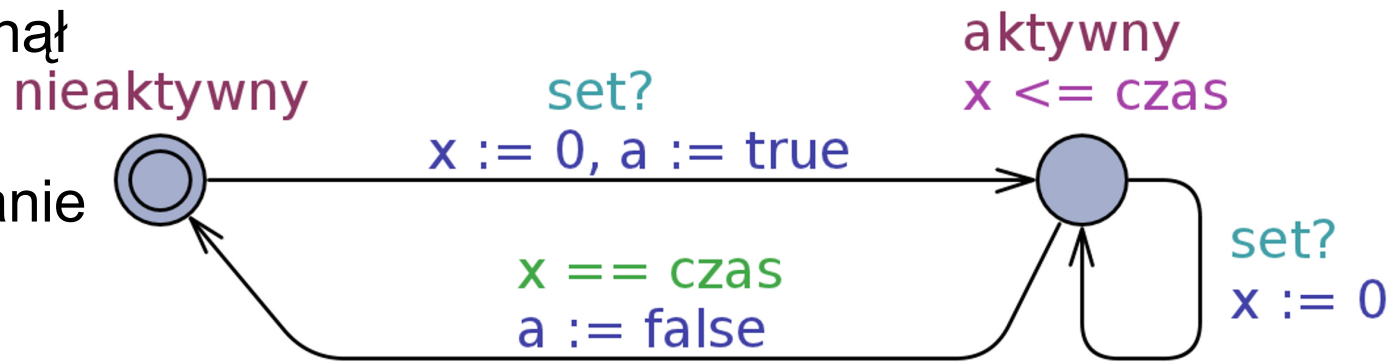
Wzorce modelowania

Regulator czasowy

(na podst. G. Behrmann et al. „A tutorial on UPPAAL”, 2006, str. 36—37)

Regulator czasowy ze stałym czasem działania:

- Regulator jest aktywowany kanałem (*chan*) *set*.
- W momencie aktywacji regulatora zerowany jest zegar (*clock*) *x*.
- Stała (*const int*) *czas* określa długość czasu aktywności regulatora.
- Zmienna (*bool*) *a* określa aktywność regulatora.
- W momencie dezaktywacji regulatora zmienna *a* staje się fałszywa.
- Sprawdzenie, czy upłynął odliczany czas: *!a*.
- Możliwe jest zresetowanie regulatora podczas jego działania.



Wzorce modelowania

Dobre praktyki

- W celu ograniczenia przestrzeni stanów w weryfikacji formuł zaleca się:
  - deklarować zmienne z maksymalnie ograniczonym zakresem wartości,
  - wyzerować zmienną lokalną, gdy jej wartość nie jest już potrzebna.

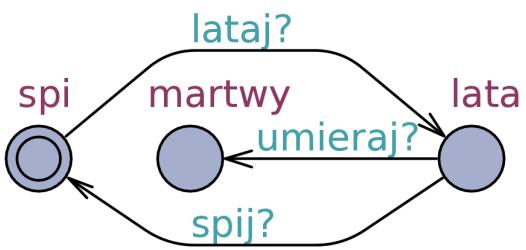
Przykłady

- Komar
- Trzech uciekinierów
- Algorytm Pettersona
- Dostęp do pliku
- Pociągi i szlaban

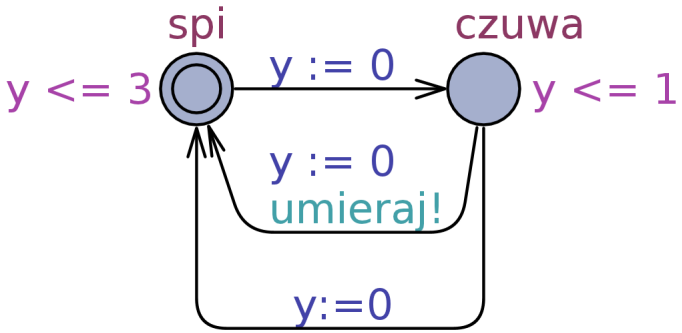
Przykłady

Komar

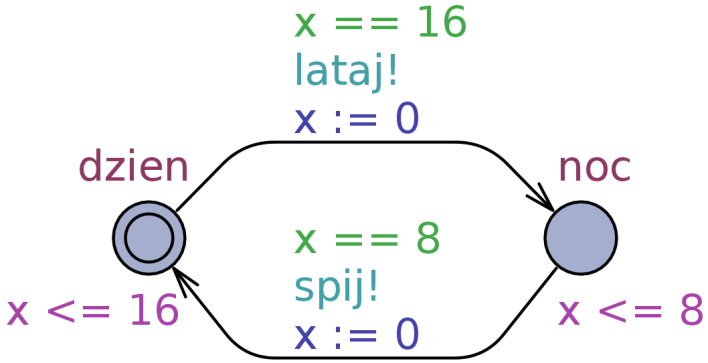
- Latem, kiedy dzień trwa 16 godzin, a noc trwa 8 godzin, komar poluje na krew zmęczonego człowieka.
- Człowiek śpi do 3 godzin albo czuwa do 1 godziny.
- Czas każe komarowi spać w dzień, a latać w nocy.
- Kiedy komar lata, a człowiek czuwa, komar może umrzeć.



Komar



Człowiek

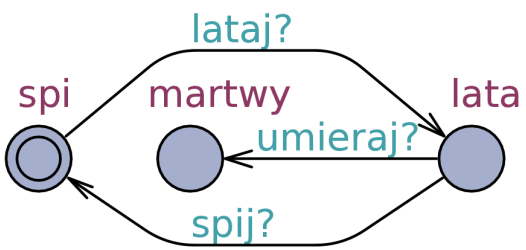


Czas

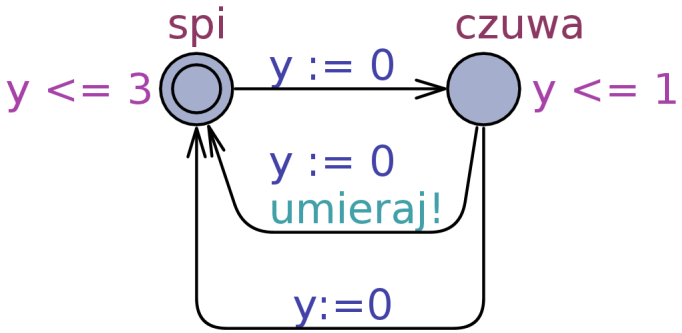
Przykłady

Komar

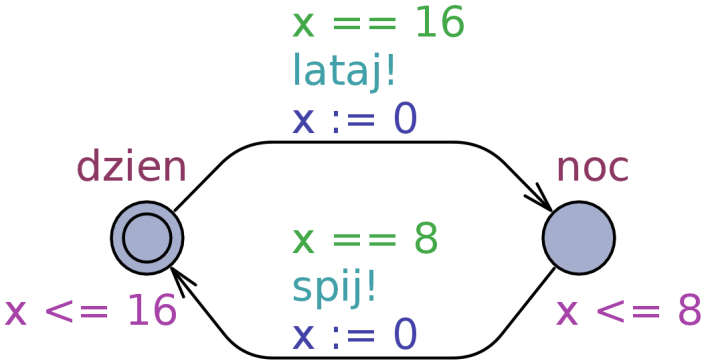
- Czy komar może zostać zabity?
  E<>Komar.martwy
- Czy komar na pewno zostanie zabity?
  A<>Komar.martwy
- Czy latający komar na pewno zostanie zabity?
  Komar.lata --> Komar.martwy
- Czy śpiący komar nie umiera?
  A[] Komar.spi imply not Komar.martwy



Komar



Człowiek

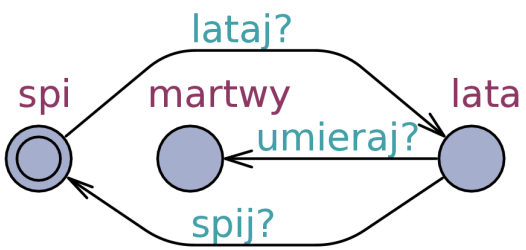


Czas

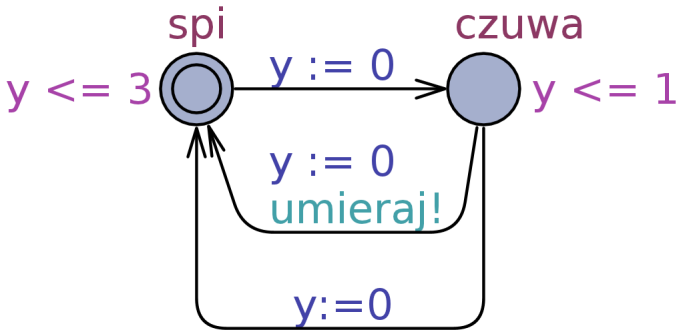
Przykłady

Komar

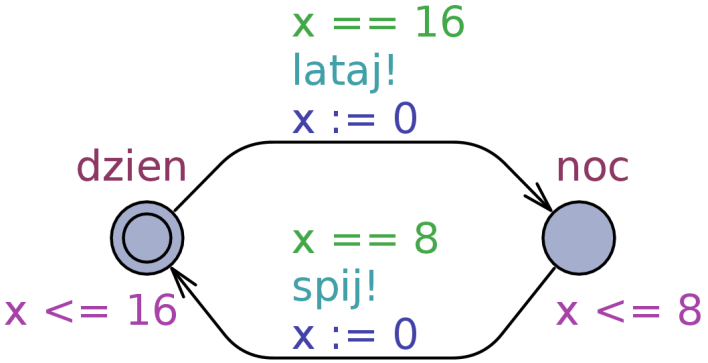
- Czy komar może zostać zabity?
  E<>Komar.martwy TAK
- Czy komar na pewno zostanie zabity?
  A<>Komar.martwy
- Czy latający komar na pewno zostanie zabity?
  Komar.lata --> Komar.martwy
- Czy śpiący komar nie umiera?
  A[] Komar.spi imply not Komar.martwy



Komar



Człowiek

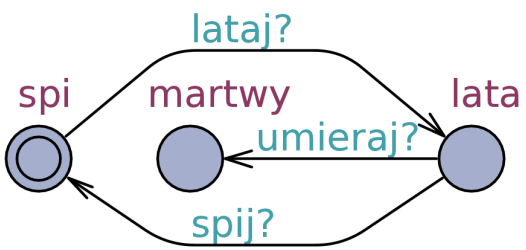


Czas

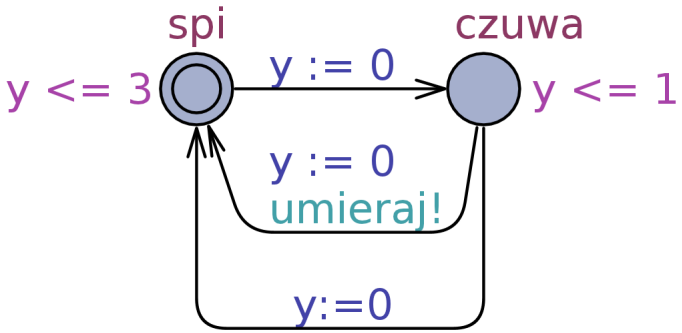
Przykłady

Komar

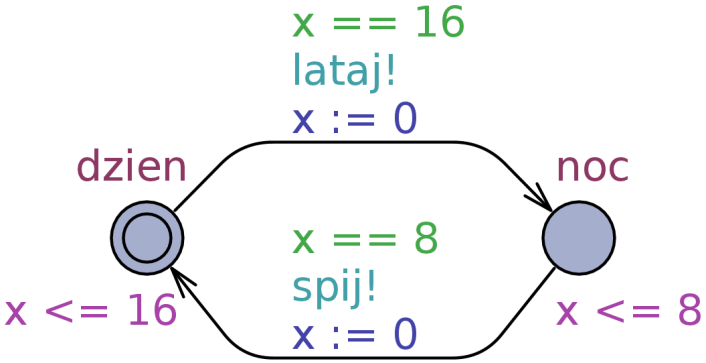
- Czy komar może zostać zabity?
  E<>Komar.martwy TAK
- Czy komar na pewno zostanie zabity?
  A<>Komar.martwy NIE
- Czy latający komar na pewno zostanie zabity?
  Komar.lata --> Komar.martwy
- Czy śpiący komar nie umiera?
  A[] Komar.spi imply not Komar.martwy



Komar



Człowiek

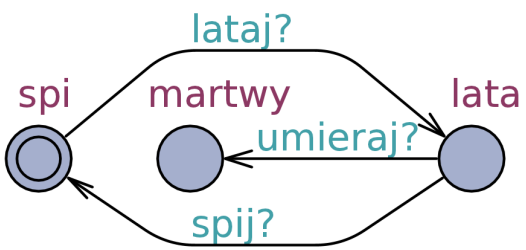


Czas

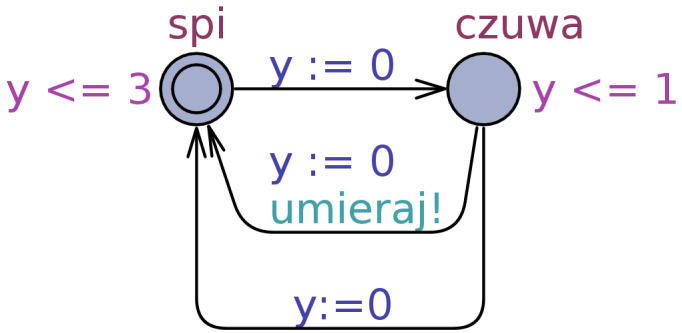
Przykłady

Komar

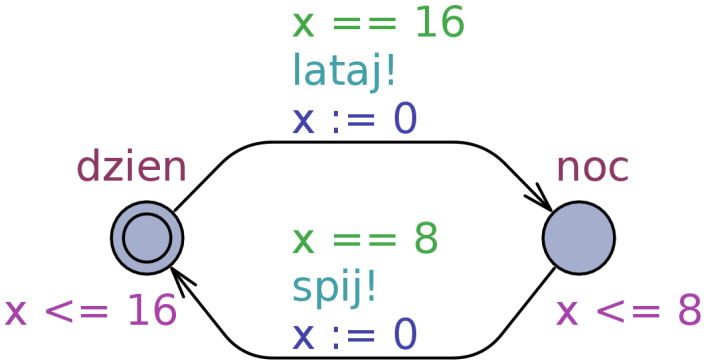
- Czy komar może zostać zabity?
  E<>Komar.martwy TAK
- Czy komar na pewno zostanie zabity?
  A<>Komar.martwy NIE
- Czy latający komar na pewno zostanie zabity?
  Komar.lata --> Komar.martwy NIE
- Czy śpiący komar nie umiera?
  A[] Komar.spi imply not Komar.martwy



Komar



Człowiek

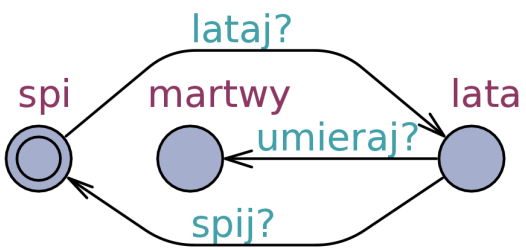


Czas

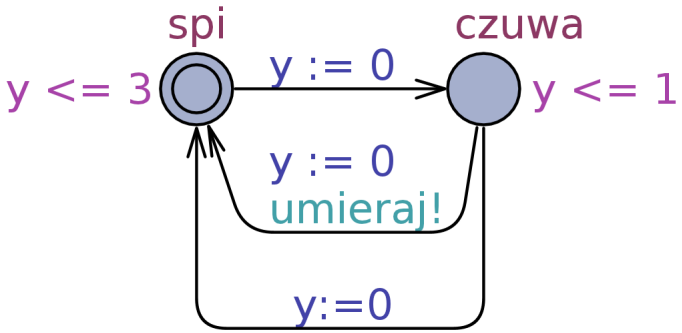
Przykłady

Komar

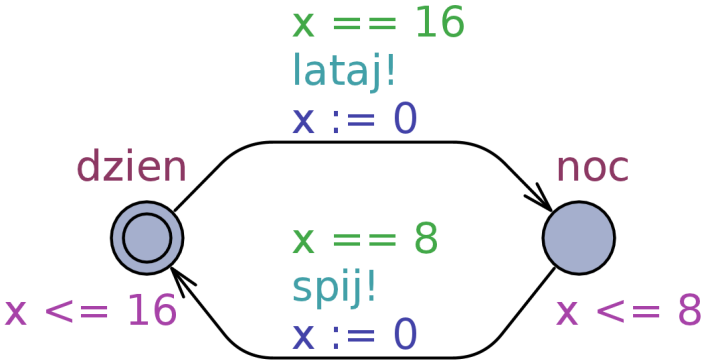
- Czy komar może zostać zabity?
  E<>Komar.martwy TAK
- Czy komar na pewno zostanie zabity?
  A<>Komar.martwy NIE
- Czy latający komar na pewno zostanie zabity?
  Komar.lata --> Komar.martwy NIE
- Czy śpiący komar nie umiera?
  A[] Komar.spi imply not Komar.martwy TAK



Komar



Człowiek



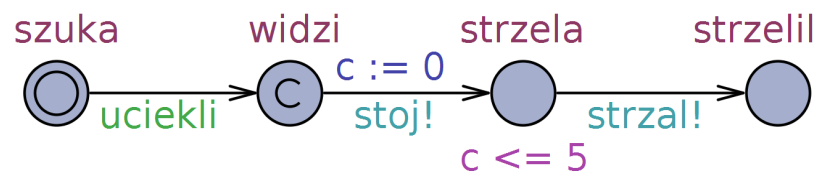
Czas

Przykłady

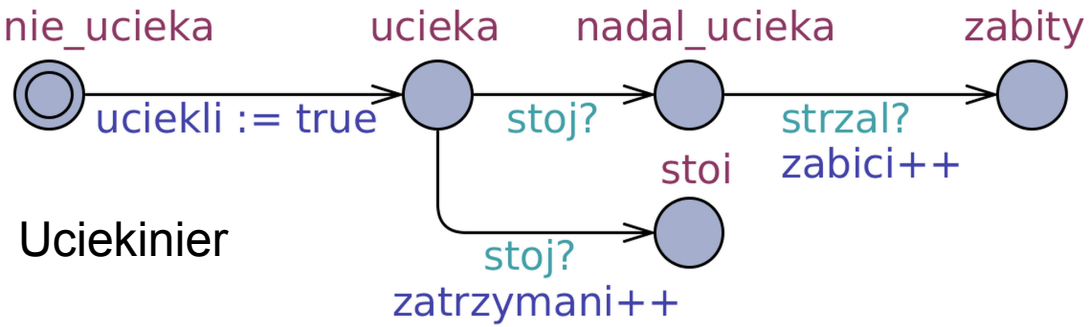
Trzech uciekinierów

Trzech uciekinierów uciekło z więzienia.

- Poszukuje ich jeden policjant.
- Gdy zobaczy pierwszego z nich, woła *stój!* (*broadcast chan*).
- Każdy z uciekających uciekinierów może się zatrzymać.
- Dokładnie jeden z nadal uciekających uciekinierów zostanie strzałem (*chan*) zabity przez policjanta.
- Policjant ma najwyżej 5 sekund na strzał.



Policjant



Uciekinier

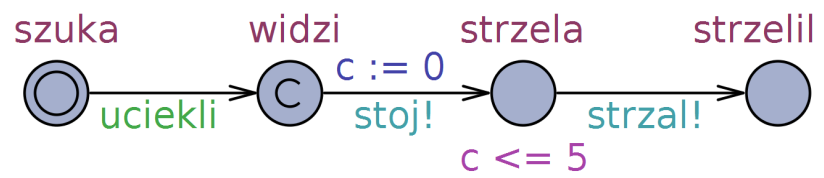
Przykłady

Trzech uciekinierów

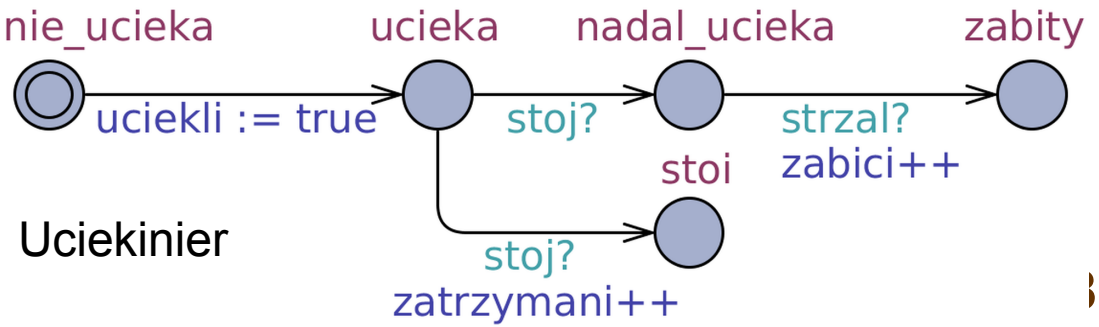
Trzech uciekinierów uciekło z więzienia.

- Czy wszyscy uciekinierzy zatrzymają się?
  A<> forall (i:int[1,3]) Uciekinier(i).stoi
  A<> zatrzymani == 3

- Czy można zatrzymać wszystkich uciekinierów?
  E<> forall (i:int[1,3]) Uciekinier(i).stoi
  E<> zatrzymani == 3



Policjant



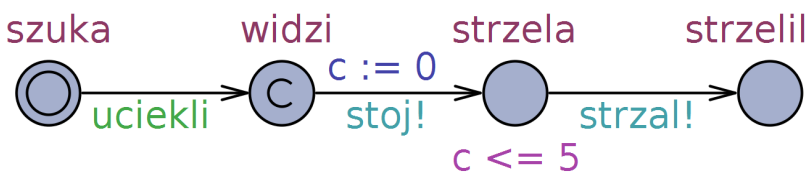
Uciekinier

Przykłady

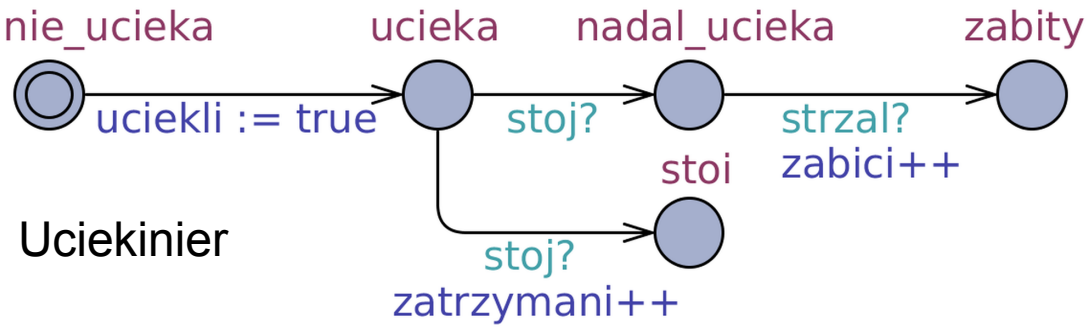
Trzech uciekinierów

Trzech uciekinierów uciekło z więzienia.

- Czy wszyscy uciekinierzy zatrzymają się?
  A<> forall (i:int[1,3]) Uciekinier(i).stoi
  A<> zatrzymani == 3
  NIE
- Czy można zatrzymać wszystkich uciekinierów?
  E<> forall (i:int[1,3]) Uciekinier(i).stoi
  E<> zatrzymani == 3



Policjant



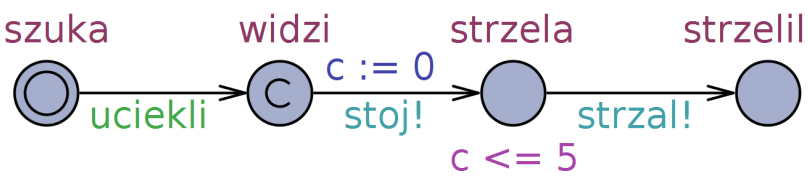
Uciekinier

Przykłady

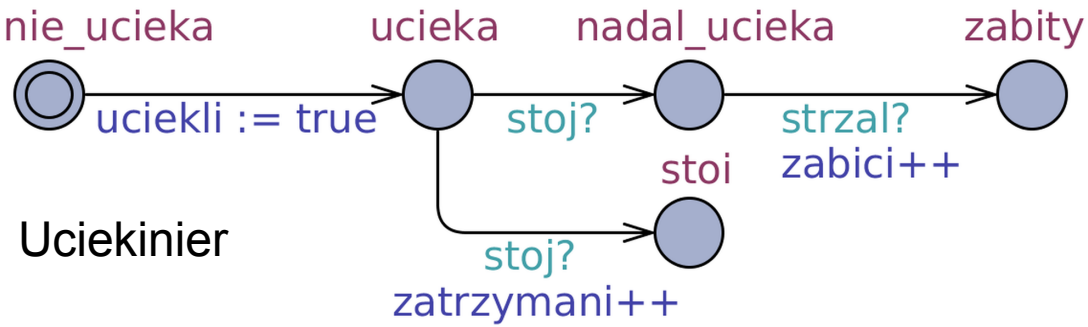
Trzech uciekinierów

Trzech uciekinierów uciekło z więzienia.

- Czy wszyscy uciekinierzy zatrzymają się?
  A<> forall (i:int[1,3]) Uciekinier(i).stoi
  A<> zatrzymani == 3
  NIE
- Czy można zatrzymać wszystkich uciekinierów?
  E<> forall (i:int[1,3]) Uciekinier(i).stoi
  E<> zatrzymani == 3
  TAK



Policjant



Uciekinier

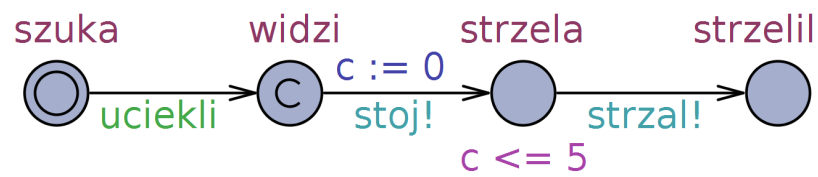
Przykłady

Trzech uciekinierów

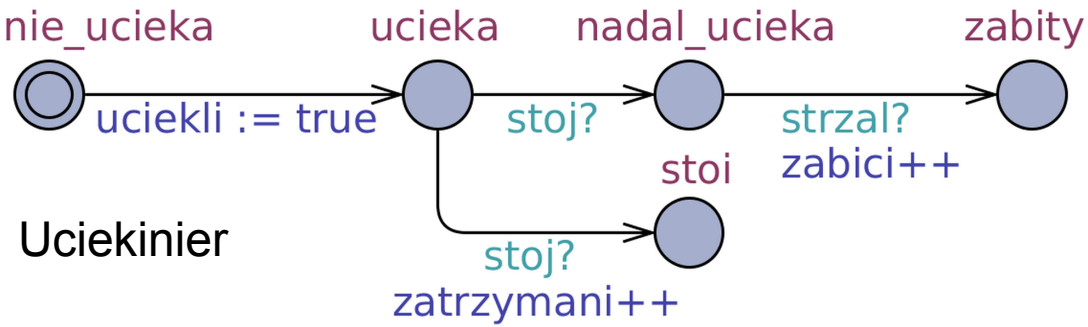
Trzech uciekinierów uciekło z więzienia.

- Czy można zastrzelić więcej niż jednego uciekiniera?
  A[] forall (i:int[1,3]) Uciekinier(i).zabity imply zabici == 1 kontrprzykład!
  E<> zabici > 1

- Czy można zastrzelić wszystkich uciekinierów?
  E<> forall (i:int[1,3]) Uciekinier(i).zabity
  E<> zabici == 3



Policjant



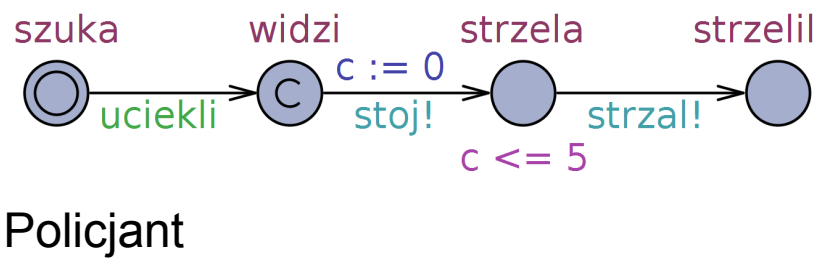
Uciekinier

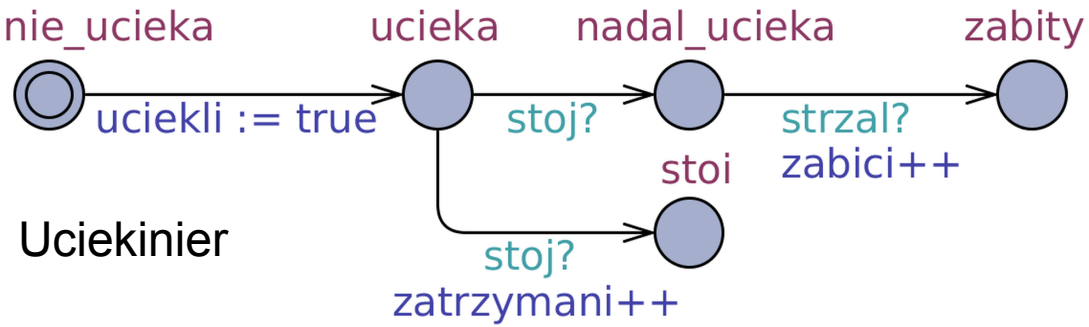
Przykłady

Trzech uciekinierów

Trzech uciekinierów uciekło z więzienia.

- Czy można zastrzelić więcej niż jednego uciekiniera?
  A[] forall (i:int[1,3]) Uciekinier(i).zabity imply zabici == 1 kontrprzykład!
  E<> zabici > 1
  NIE
- Czy można zastrzelić wszystkich uciekinierów?
  E<> forall (i:int[1,3]) Uciekinier(i).zabity
  E<> zabici == 3



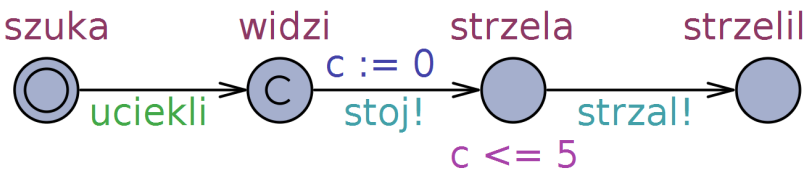


Przykłady

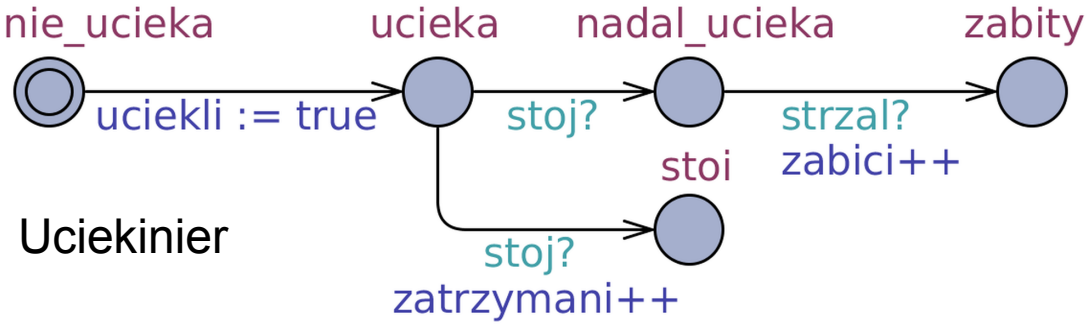
Trzech uciekinierów

Trzech uciekinierów uciekło z więzienia.

- Czy można zastrzelić więcej niż jednego uciekiniera?
  A[] forall (i:int[1,3]) Uciekinier(i).zabity imply zabici == 1 kontrprzykład!
  E<> zabici > 1
  NIE
- Czy można zastrzelić wszystkich uciekinierów?
  E<> forall (i:int[1,3]) Uciekinier(i).zabity
  E<> zabici == 3
  NIE



Policjant



Uciekinier

Przykłady

Algorytm Pettersona

(na podst. A. David et al. „UPPAAL 4.0: Small tutorial”, 2009)

Algorytm Pettersona dla wzajemnego wykluczania 2 procesów:

- **Proces P1:**

```
req1=1;                          // stan „idle”
turn=2;                          // stan „want”
while(turn!=1 && req2!=0);       // stan „wait”
req1=0;                          // stan „CS”
```

- **Proces P2:**

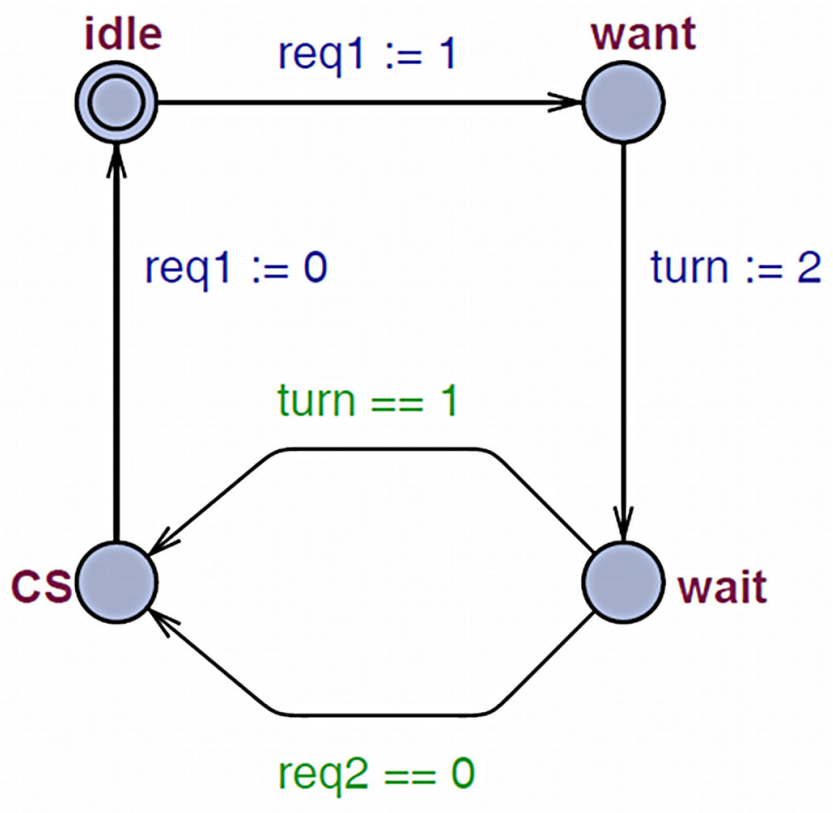
```
req2=1;                          // stan „idle”
turn=1;                          // stan „want”
while(turn!=2 && req1!=0);       // stan „wait”
req2=0;                          // stan „CS”
```

req1, req2, turn — zmienne kontrolne
CS — sekcja krytyczna

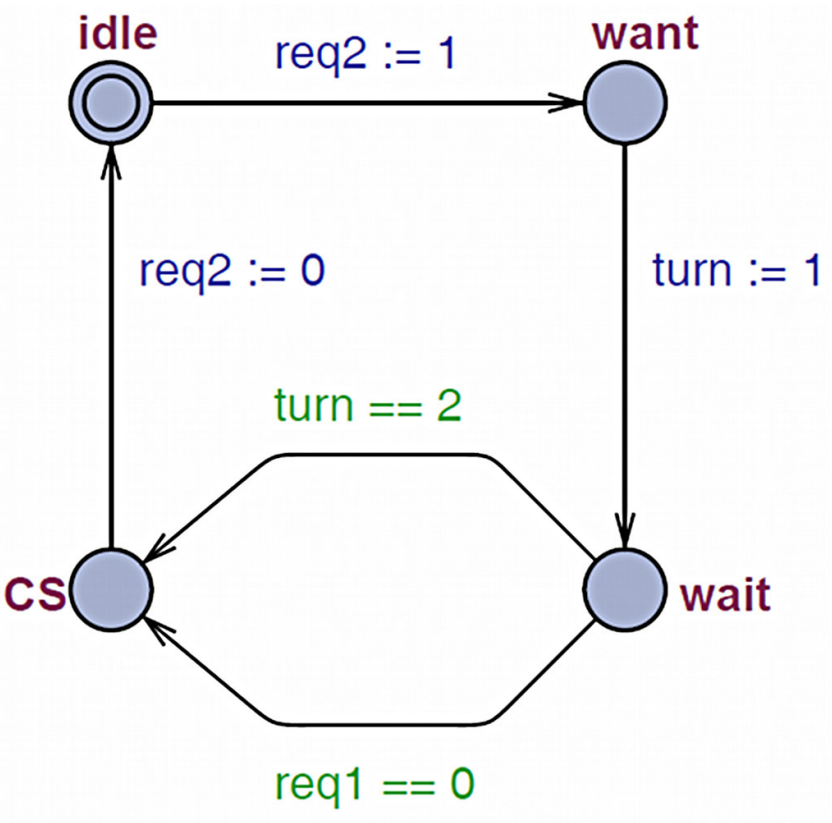
Przykłady

Algorytm Pettersona

(na podst. A. David et al. „UPPAAL 4.0: Small tutorial”, 2009)



P1



P2

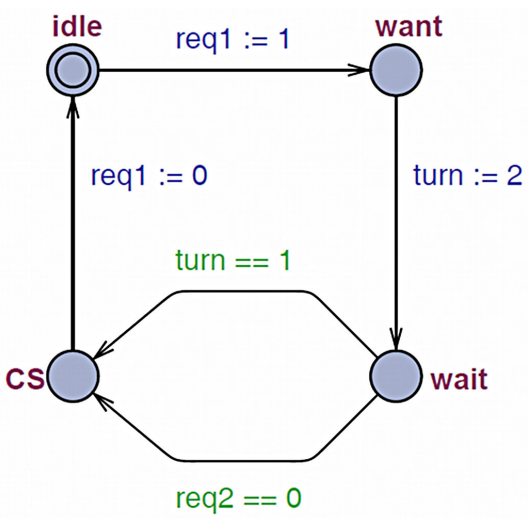
Przykłady

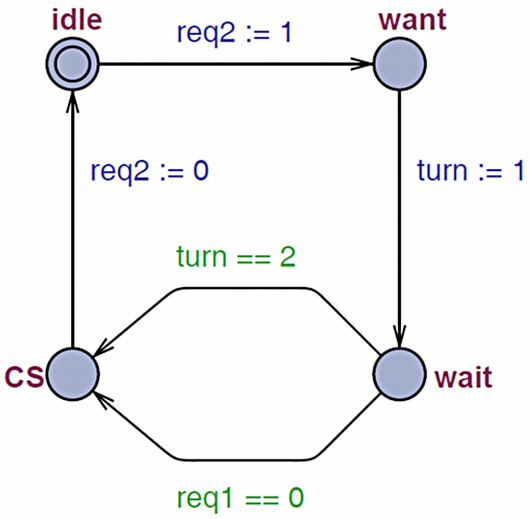
Algorytm Pettersona

(na podst. A. David et al. „UPPAAL 4.0: Small tutorial”, 2009)

Własności związane z sekcją krytyczną (CS):

- *bezpieczeństwo* — wzajemne wykluczanie obu procesów:
  A[] not (P1.CS and P2.CS)
- *osiągalność* — dostęp do sekcji krytycznej:
  E<>(P1.CS)
  E<>(P2.CS)



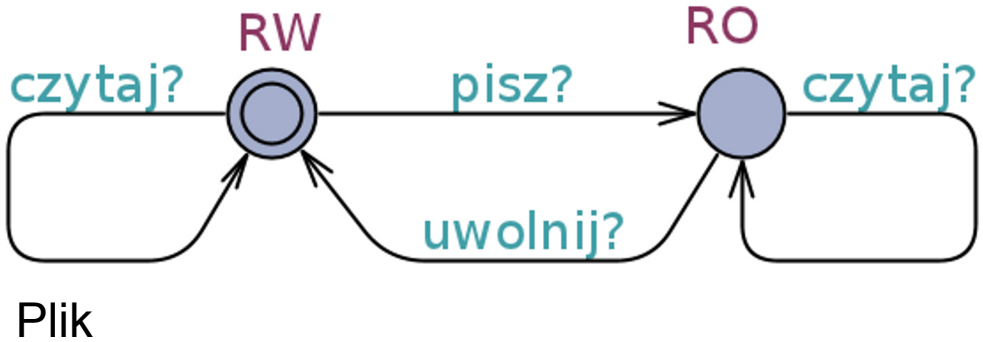


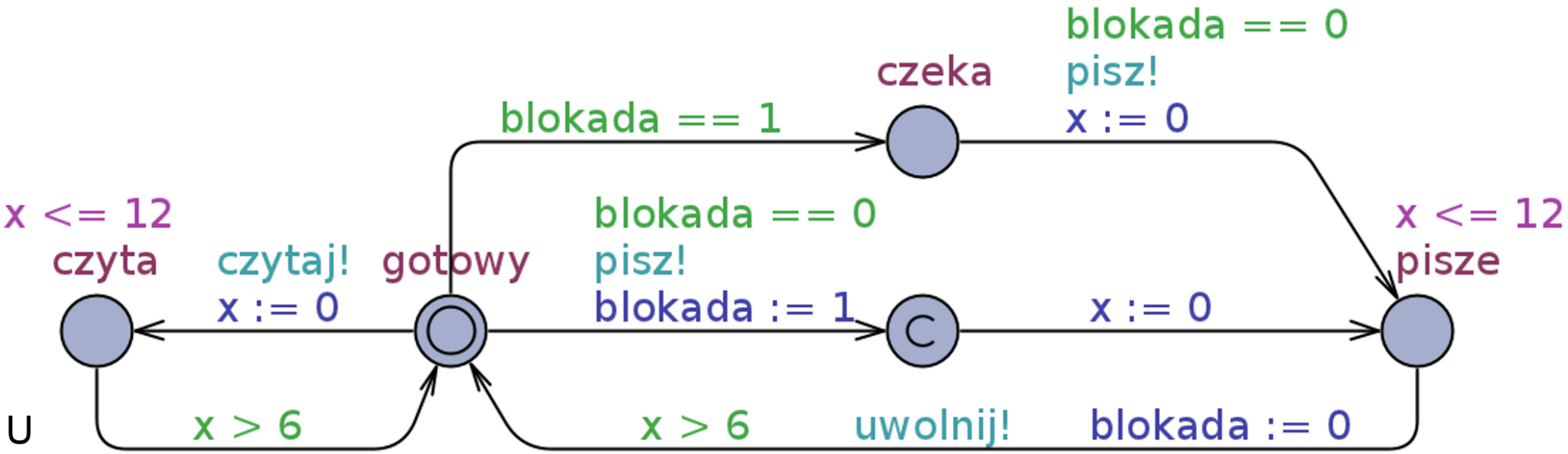
Przykłady

Dostęp do pliku

Użytkownicy *U(0)* i *U(1)* mogą chcieć dostępu do pliku:

- Plik jest zawsze dostępny do odczytu.
- Plik jest dostępny do zapisu tylko dla jednego użytkownika naraz. Drugi musi wówczas czekać na dostęp.

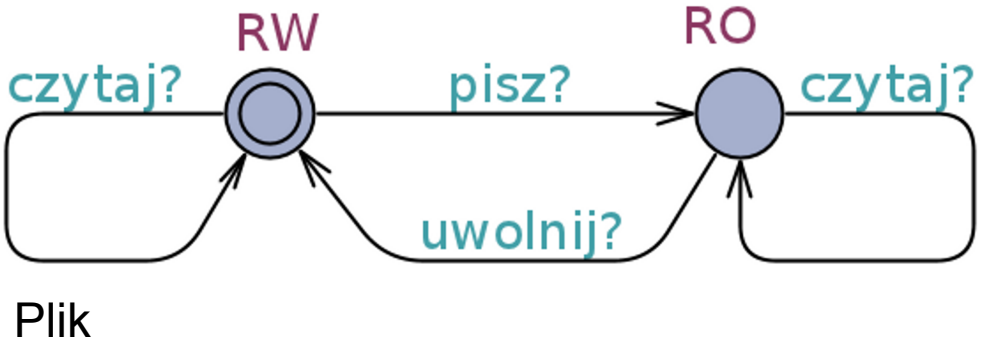




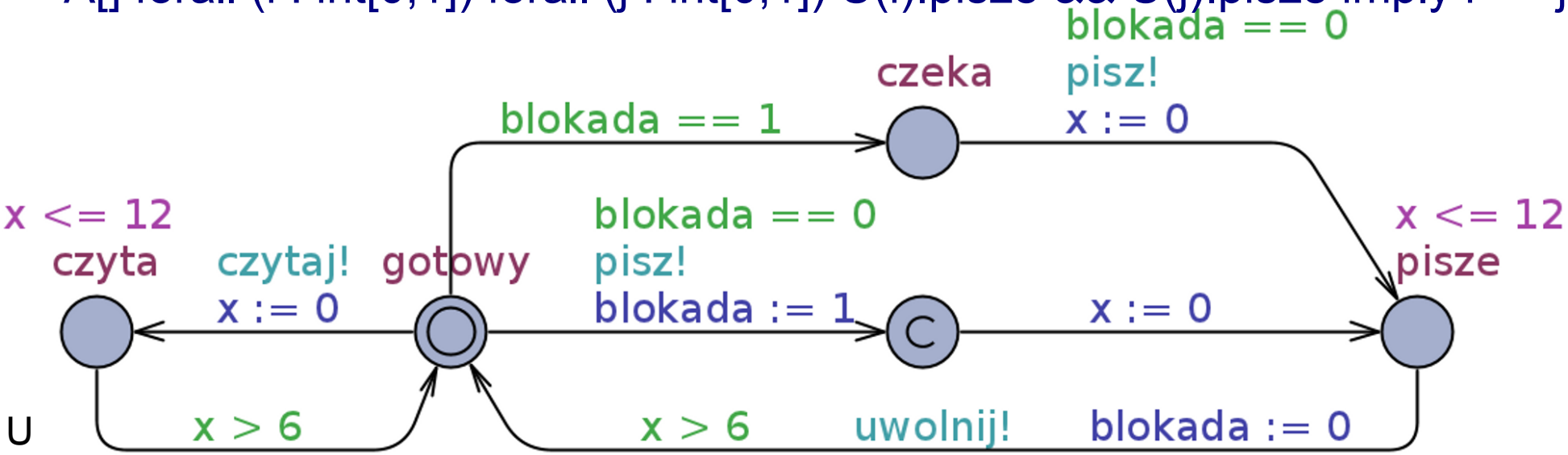
Przykłady

Dostęp do pliku

- Czy U(0) może pisać, podczas gdy U(1) czeka?
  E<> U(0).pisze and U(1).czeka



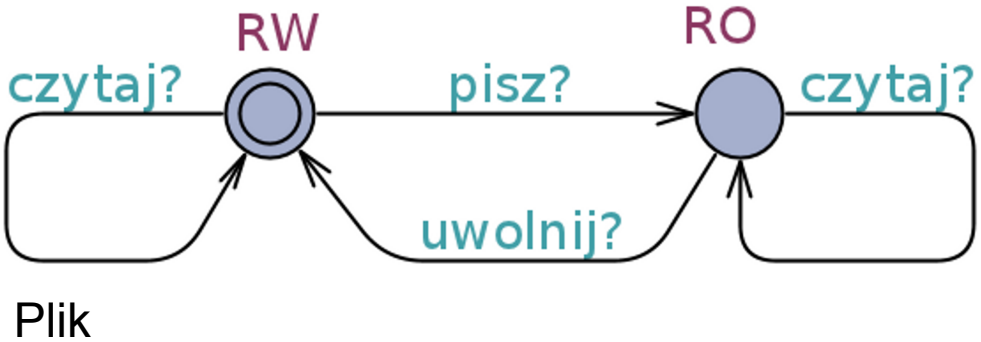
- Czy plik może być zapisywany przez więcej niż jednego użytkownika naraz?
  A[] forall (i : int[0,1]) forall (j : int[0,1]) U(i).pisze && U(j).pisze imply i == j



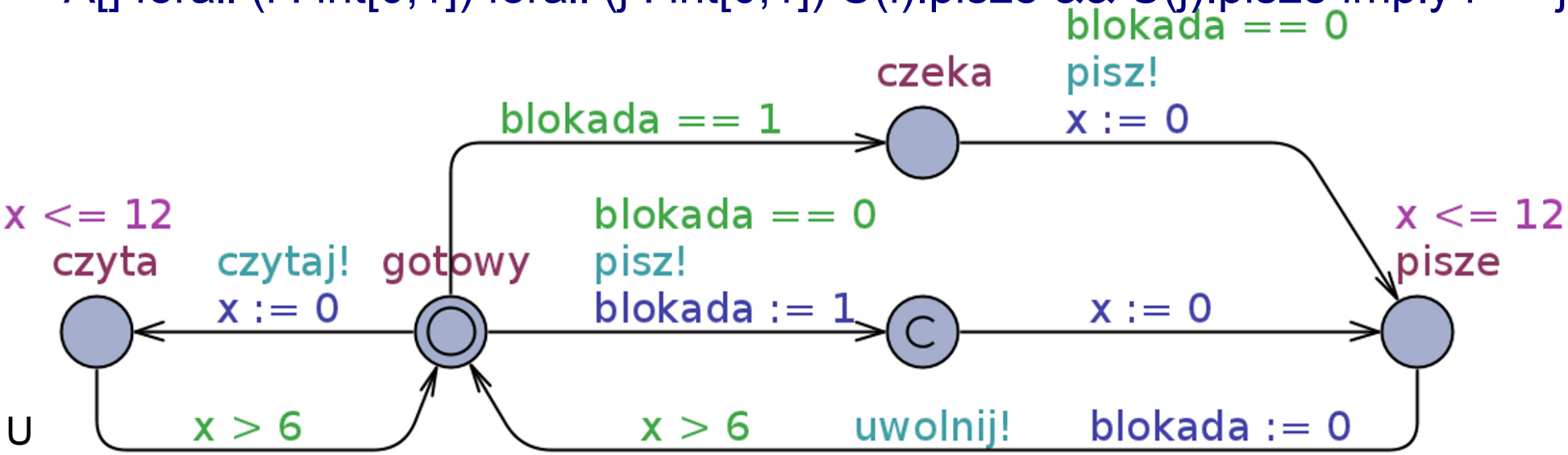
Przykłady

Dostęp do pliku

- Czy U(0) może pisać, podczas gdy U(1) czeka?
  E<> U(0).pisze and U(1).czeka
  TAK



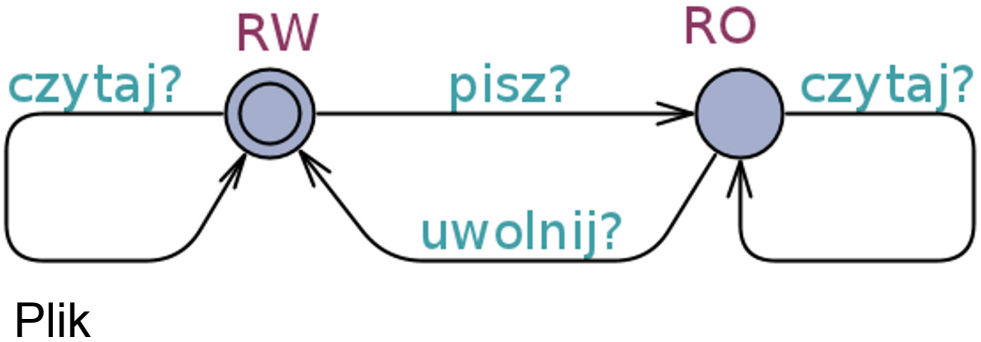
- Czy plik może być zapisywany przez więcej niż jednego użytkownika naraz?
  A[] forall (i : int[0,1]) forall (j : int[0,1]) U(i).pisze && U(j).pisze imply i == j



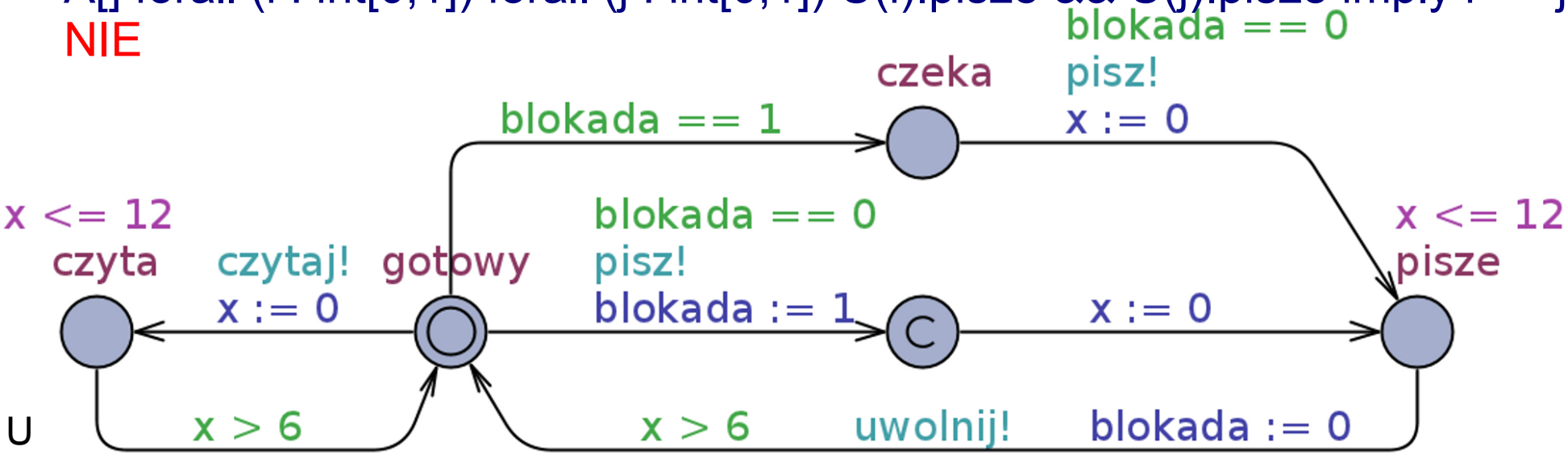
Przykłady

Dostęp do pliku

- Czy U(0) może pisać, podczas gdy U(1) czeka?
  E<> U(0).pisze and U(1).czeka
  TAK



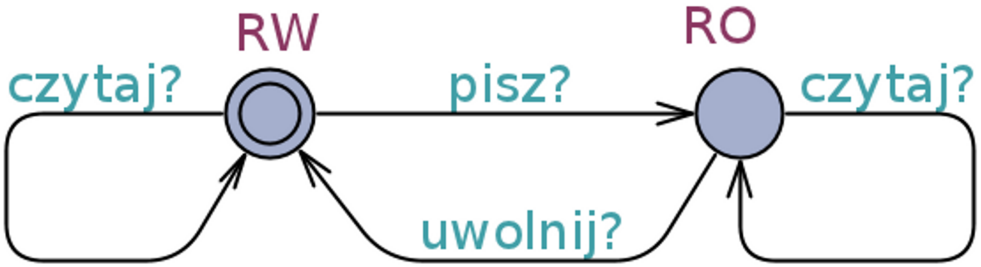
- Czy plik może być zapisywany przez więcej niż jednego użytkownika naraz?
  A[] forall (i : int[0,1]) forall (j : int[0,1]) U(i).pisze && U(j).pisze imply i == j
  NIE



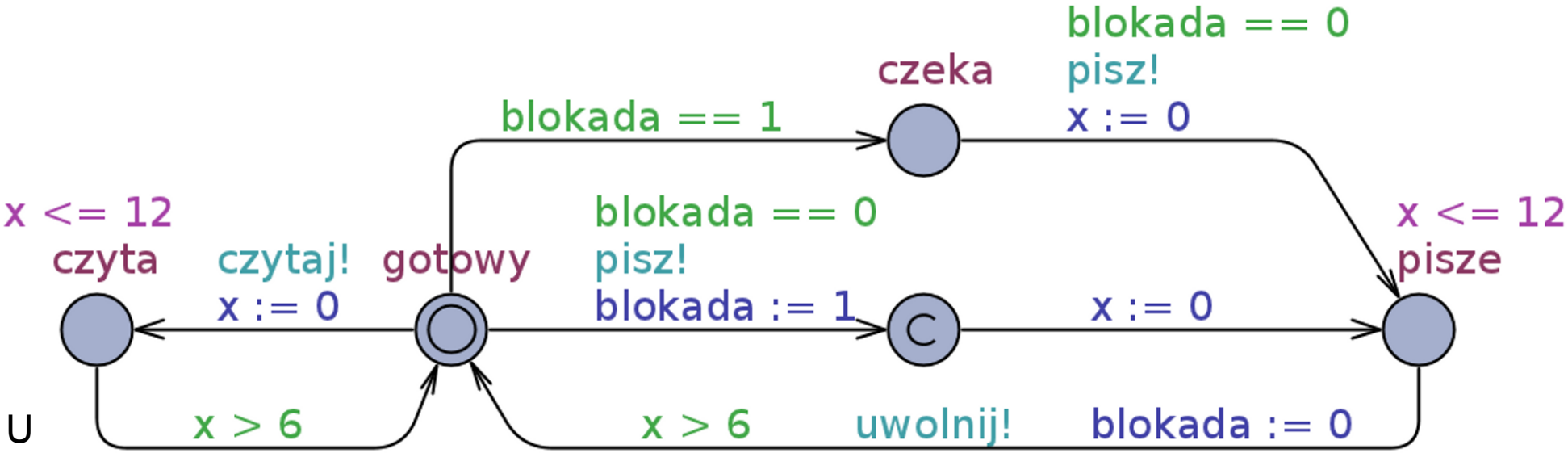
Przykłady

Dostęp do pliku

- Czy czekający użytkownik (np. U(0)) na pewno otrzyma dostęp do zapisu?
  U(0).czeka --> U(0).pisze



Plik

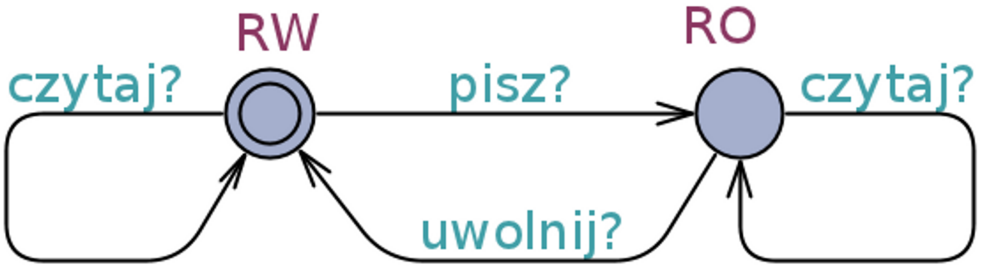


U

Przykłady

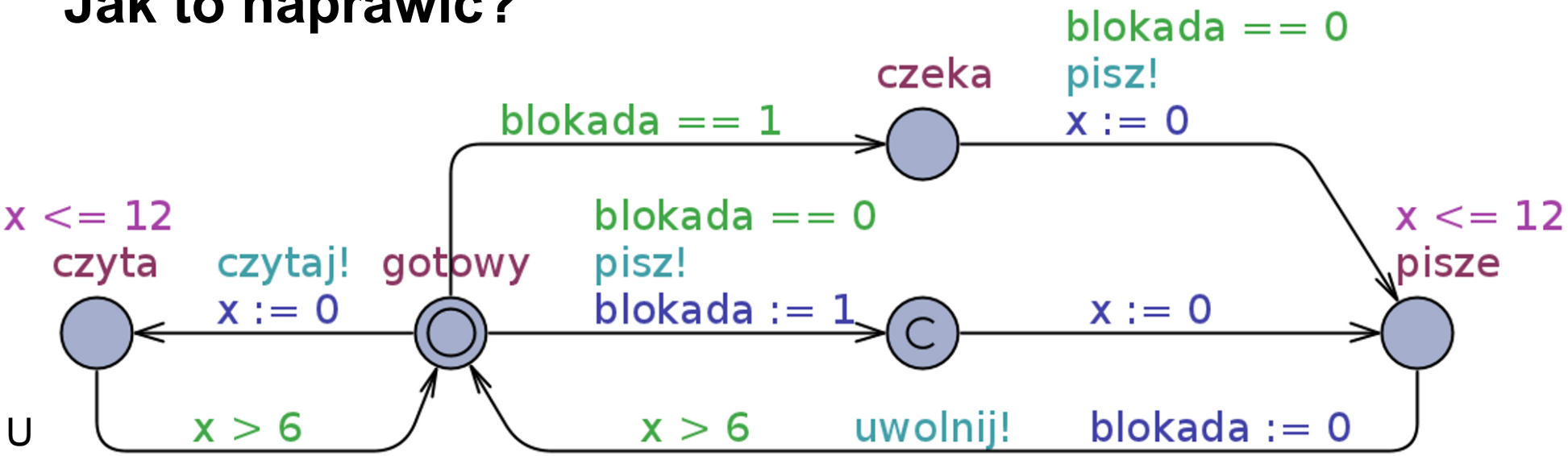
Dostęp do pliku

- Czy czekający użytkownik (np. U(0)) na pewno otrzyma dostęp do zapisu?
  U(0).czeka --> U(0).pisze
  NIE



Plik

- **Oczekiwanie na dostęp może być nieskończone. Jak to naprawić?**

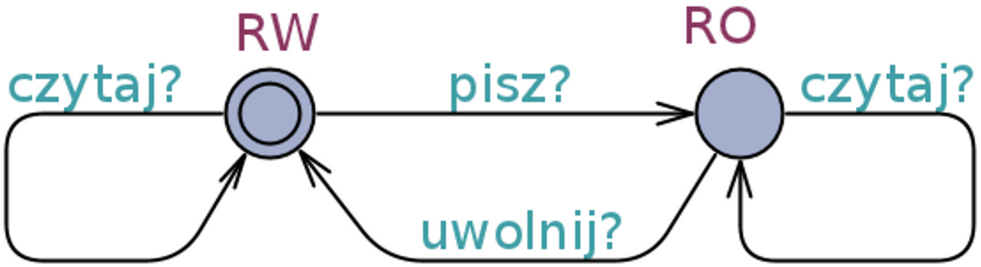


U

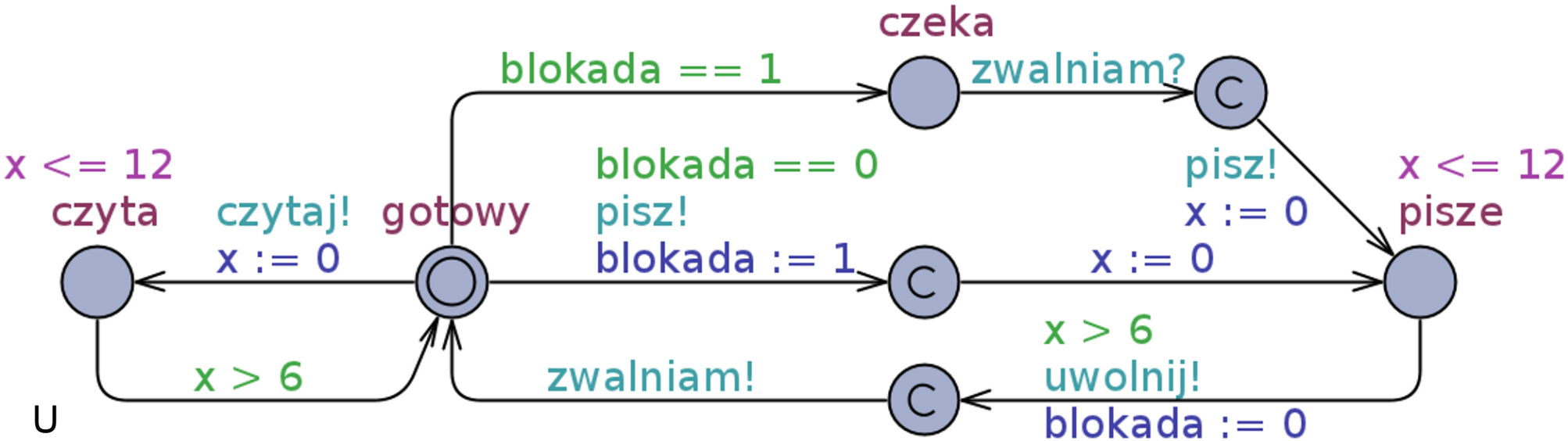
Przykłady

Dostęp do pliku

- Czy czekający użytkownik (np. U(0)) na pewno otrzyma dostęp do zapisu?
  U(0).czeka --> U(0).pisze
  TAK



Plik



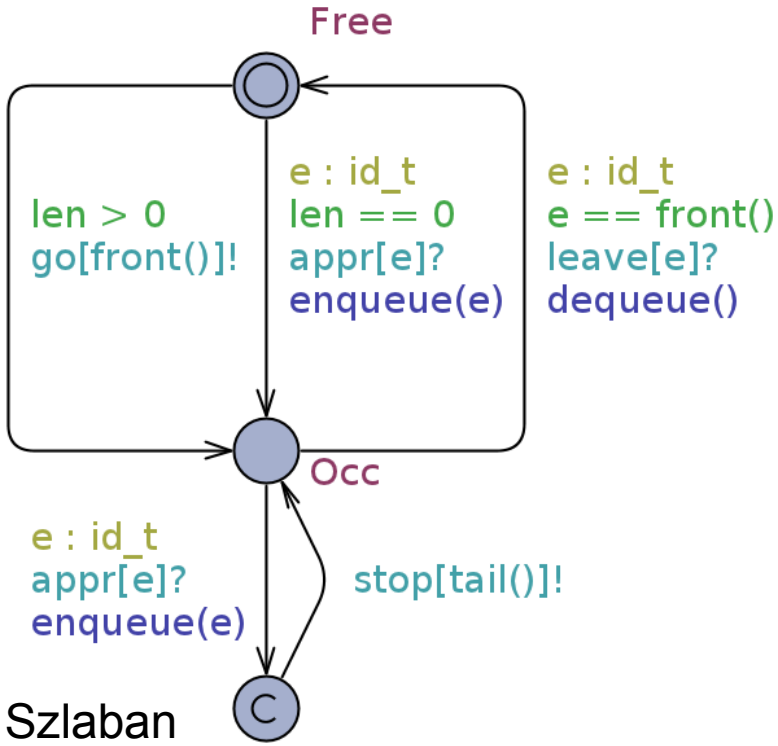
U

Przykłady

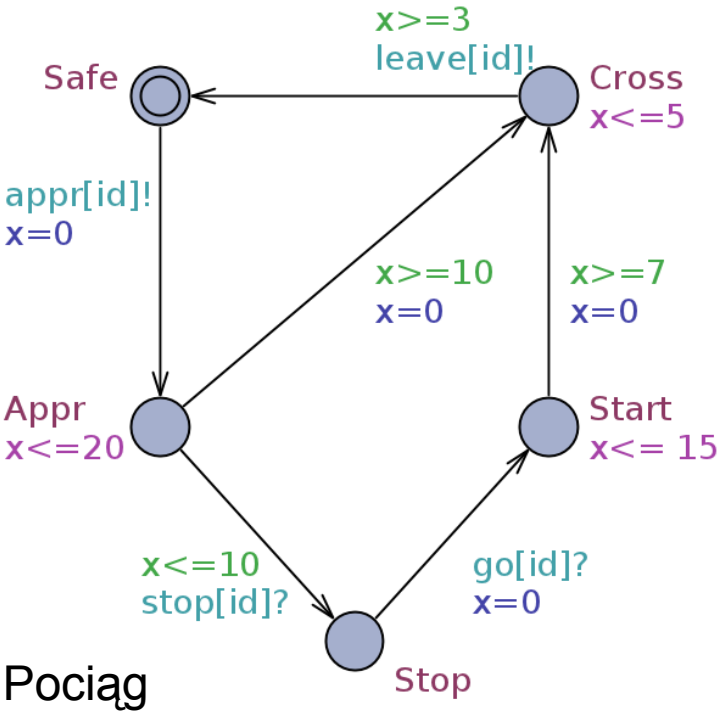
Pociągi i szlaban

(na podst. W. Yi et al. „Automatic Verification of Real-Time Communicating Systems by Constraint Solving”,1994.)
Ten model znajduje się w katalogu demo programu UPPAAL.

- N pociągów zbliża się jednocześnie do przejazdu przez most.
- Szlaban wpuszcza na most tylko jeden pociąg naraz, każąc pozostałym czekać na swoją kolej.



Szlaban



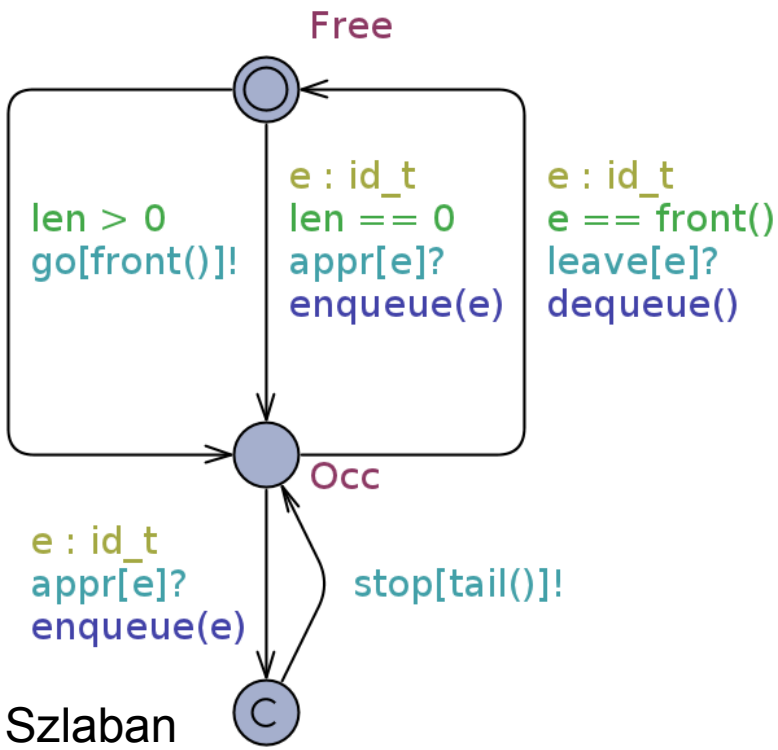
Pociąg

Przykłady

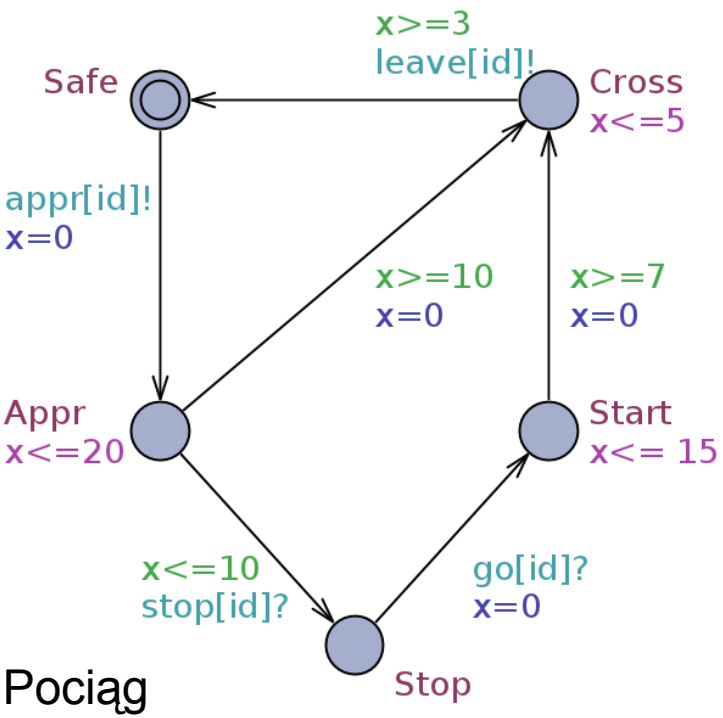
Pociągi i szlaban

(na podst. W. Yi et al. „Automatic Verification of Real-Time Communicating Systems by Constraint Solving”,1994.)
Ten model znajduje się w katalogu demo programu UPPAAL.

- Zmiany stanów pociągu zajmują pewien czas
  (x – zmienna zegarowa)



Szlaban



Pociąg

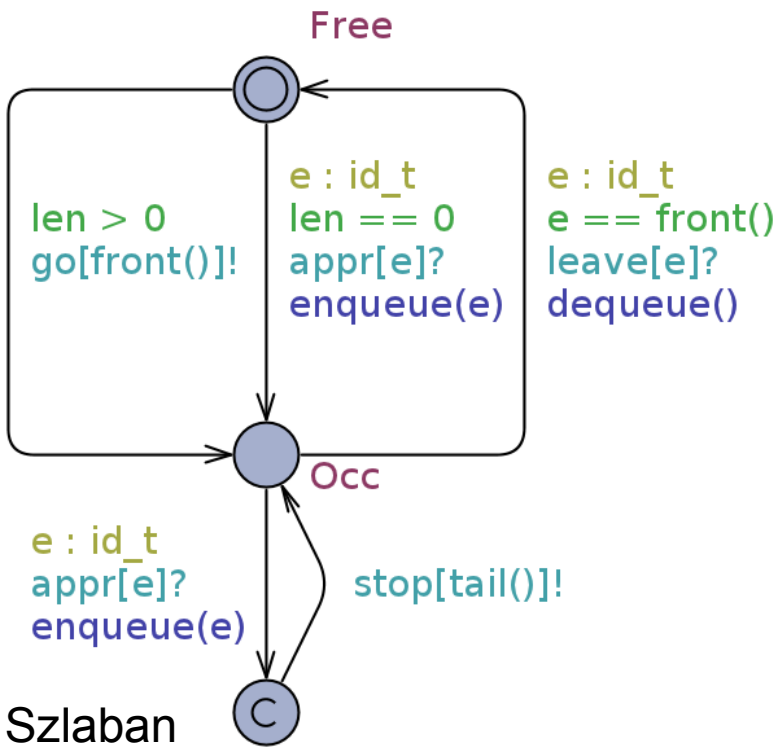
Przykłady

Pociągi i szlaban

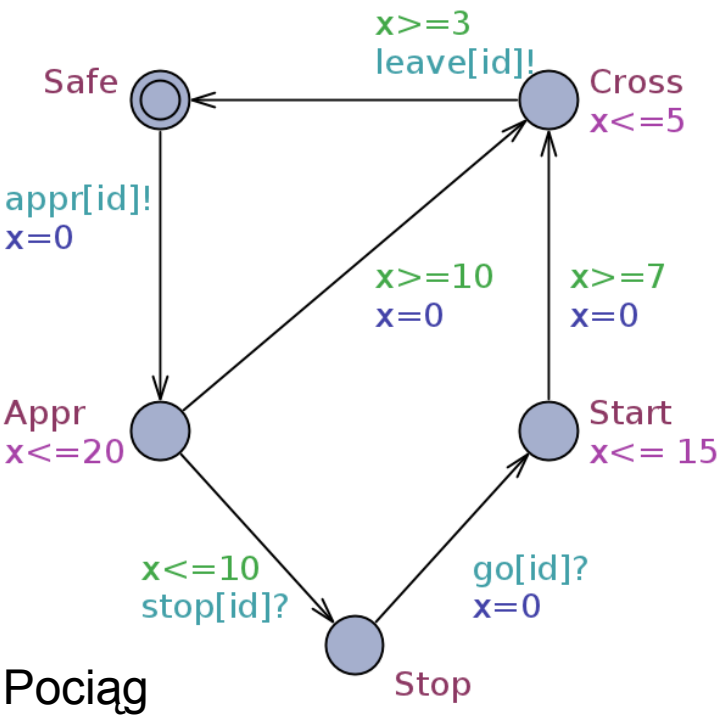
(na podst. W. Yi et al. „Automatic Verification of Real-Time Communicating Systems by Constraint Solving”,1994.)
Ten model znajduje się w katalogu demo programu UPPAAL.

- Szlaban może otrzymywać (i przechowywać w kolejce) komunikaty od nadjeżdżających pociągów:

E<> Szlaban.Occ



Szlaban



Pociąg

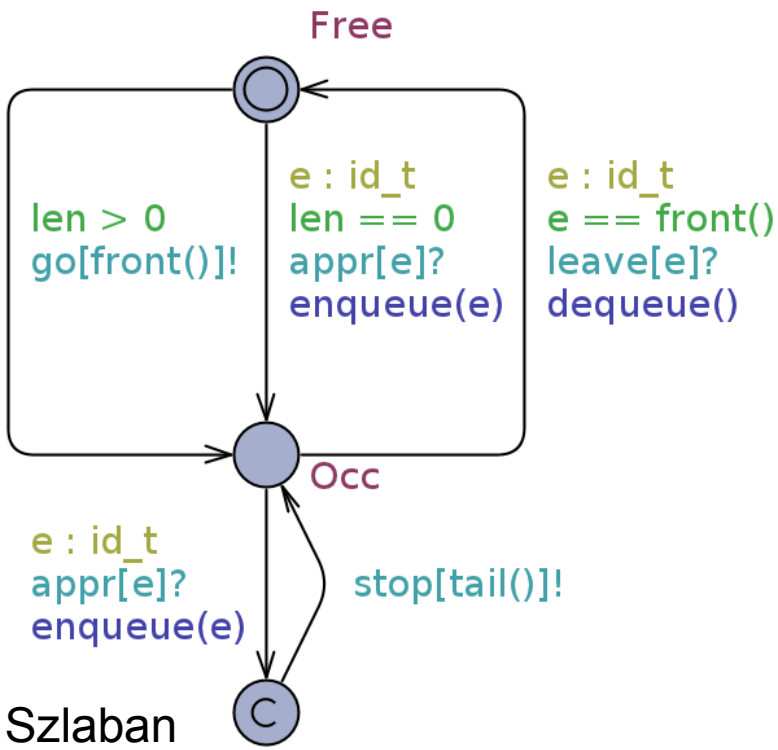
Przykłady

Pociągi i szlaban

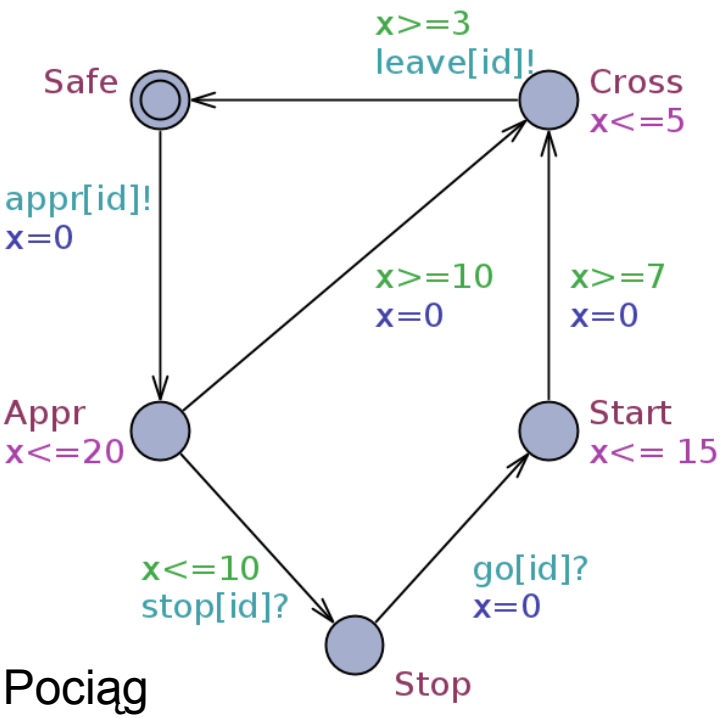
(na podst. W. Yi et al. „Automatic Verification of Real-Time Communicating Systems by Constraint Solving”,1994.)
Ten model znajduje się w katalogu demo programu UPPAAL.

- Pociąg 0 może przejechać przez most, podczas gdy pozostałe pociągi oczekują na przejazd:

E<> Pociag(0).Cross and (forall (i : id_t) i != 0 imply Pociag(i).Stop)



Szlaban



Pociąg

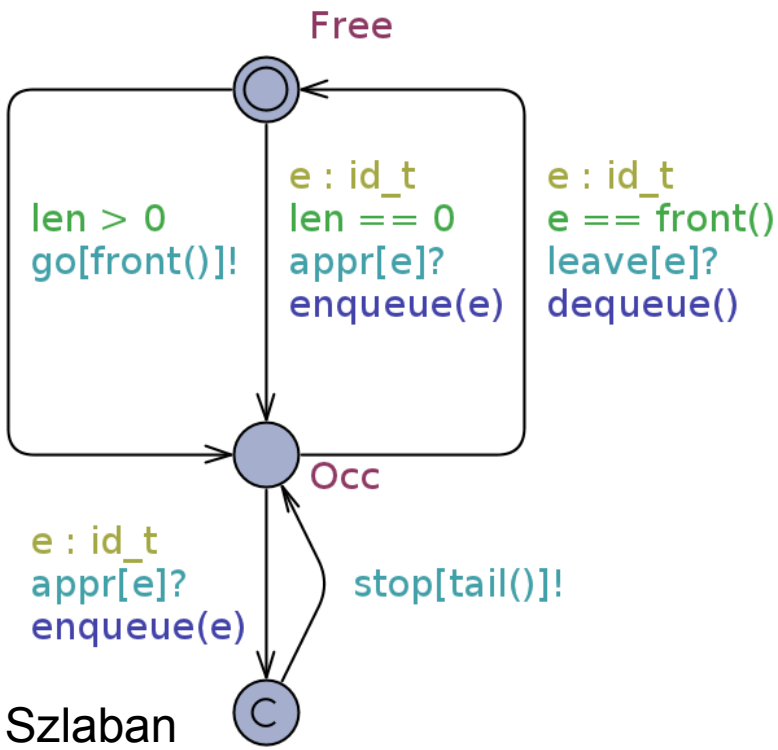
Przykłady

Pociągi i szlaban

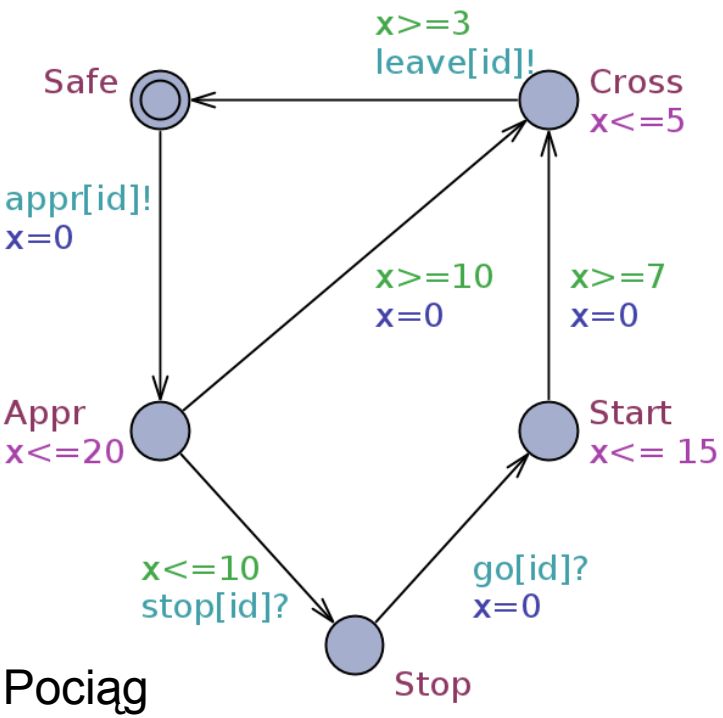
(na podst. W. Yi et al. „Automatic Verification of Real-Time Communicating Systems by Constraint Solving”,1994.)
Ten model znajduje się w katalogu demo programu UPPAAL.

- Nigdy więcej niż jeden pociąg nie przejeżdża przez most naraz:

A[] forall (i : id_t) forall (j : id_t) Pociag(i).Cross && Pociag(j).Cross imply i == j



Szlaban



Pociąg

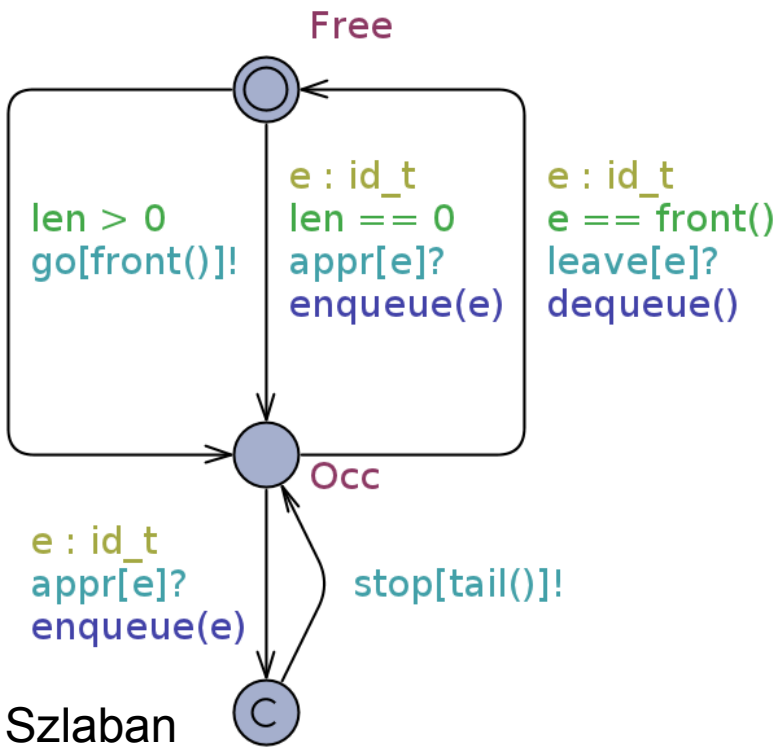
Przykłady

Pociągi i szlaban

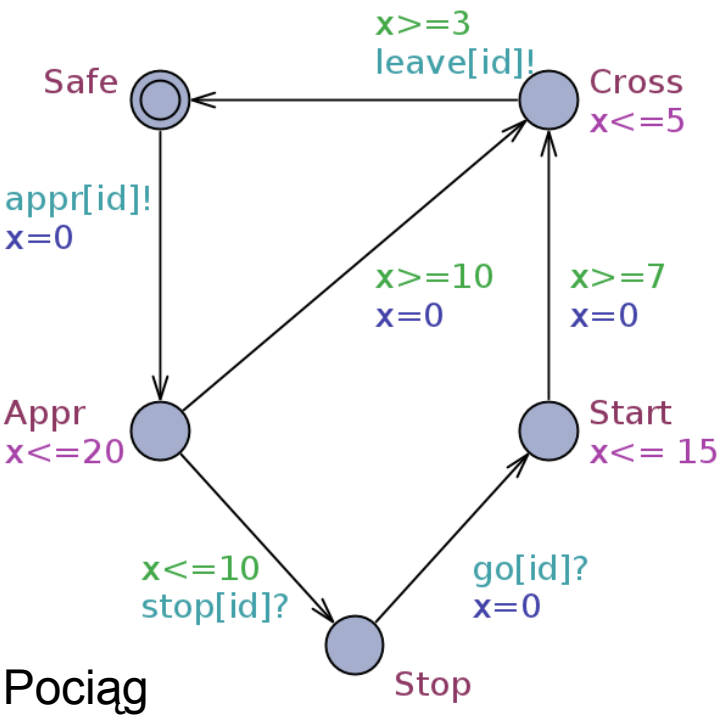
(na podst. W. Yi et al. „Automatic Verification of Real-Time Communicating Systems by Constraint Solving”,1994.)
Ten model znajduje się w katalogu demo programu UPPAAL.

- Kiedykolwiek pociąg (np. 0) zbliża się do mostu, w końcu go przejedzie:

  Pociag(0).Appr --> Pociag(0).Cross



Szlaban



Pociąg

Koniec

Literatura:

- G. Behrmann et al. „A tutorial on UPPAAL”, 2006, at: www.uppaal.com
- A. David et al. „UPPAAL 4.0: Small tutorial”, 2009, at: www.uppaal.com
- „UPPAAL Language Reference”, http://www.uppaal.com/index.php?sida=217&rubrik=101