UPPAAL

Narzędzie do weryfikacji automatów czasowych

Aleksander Zabłocki

6 maja 2009

Plan

UPPAAL

- Modeluje układy (komunikujących się) automatów czasowych
- Weryfikuje podane własności wyrażane w uproszczonym CTL
- Powstaje na uniwersytetach w Uppsali oraz Aalborgu
- Pierwsza wersja: 1995, obecnie 4.0.7
- W internecie: www.uppaal.com

UPPAAL

- Modeluje układy (komunikujących się) automatów czasowych
- Weryfikuje podane własności wyrażane w uproszczonym CTL
- Powstaje na uniwersytetach w Uppsali oraz Aalborgu
- Pierwsza wersja: 1995, obecnie 4.0.7
- W internecie: www.uppaal.com

UPPAAL

- Modeluje układy (komunikujących się) automatów czasowych
- Weryfikuje podane własności wyrażane w uproszczonym CTL
- Powstaje na uniwersytetach w Uppsali oraz Aalborgu
- Pierwsza wersja: 1995, obecnie 4.0.7
- W internecie: www.uppaal.com

Funkcjonalność

Wielofunkcyjny interfejs graficzny:

- Definiowanie wejściowej sieci automatów
- Śledzenie wybranych ścieżek wykonania
- Weryfikacja podanych własności
- Znajduje kontrprzykłady, umożliwia ich śledzenie

UPPAAL?

- SWEden + DENmark = SWEDEN
  - REJECTED
- sweDEN + denMARK = DENMARK
  - REJECTED
- UPPsala + AALborg = **UPPAAL**
  - ACCEPTED

Plan

Języki ω-regularne

- JAO: skończone automaty rozpoznają skończone słowa
- Skończone automaty i nieskończone słowa — jak akceptować?
  - Büchi: Któryś ze stanów akceptujących ma się zdarzać ∞ razy
  - Muller: *Zbiór* stanów zdarzających się ∞ ma być którymś z *akceptujących*
- Siła automatów:
  Büchi det. $<$ (Büchi ndet. $=$ Muller ndet. $=$ Muller det.)
- W wersji czasowej Muller det. $<$ Muller ndet.!

Języki ω-regularne

- JAO: skończone automaty rozpoznają skończone słowa
- Skończone automaty i nieskończone słowa — jak akceptować?
  - Büchi: Któryś ze stanów akceptujących ma się zdarzać ∞ razy
  - Muller: *Zbiór* stanów zdarzających się ∞ ma być którymś z *akceptujących*
- Siła automatów:
  Büchi det. $<$ (Büchi ndet. = Muller ndet. = Muller det.)
- W wersji czasowej Muller det. $<$ Muller ndet.!

Języki ω-regularne

- JAO: skończone automaty rozpoznają skończone słowa
- Skończone automaty i nieskończone słowa — jak akceptować?
  - Büchi: Któryś ze stanów akceptujących ma się zdarzać ∞ razy
  - Muller: *Zbiór* stanów zdarzających się ∞ ma być którymś z *akceptujących*
- Siła automatów:
  Büchi det. $<$ (Büchi ndet. = Muller ndet. = Muller det.)
- W wersji czasowej Muller det. $<$ Muller ndet.!

Dokładamy czas

- Znak s_i w słowie σ pojawia się w momencie $t_i \in \mathbb{R}$
- *Słowem* jest para: (napis σ, ciąg czasów τ)
- Monotoniczność: $t_i \leq t_{i+1}$
- Rozbieżność: $t_i \longrightarrow \infty$
- Automat czasowy:
  - Pobyt w *lokacji* może trwać pewien czas
  - Przejście po krawędzi — natychmiastowe
  - Możliwe przejścia mogą zależeć od czasu

Dokładamy czas

- Znak s_i w słowie σ pojawia się w momencie $t_i \in \mathbb{R}$
- *Słowem* jest para: (napis σ, ciąg czasów τ)
- Monotoniczność: $t_i \leq t_{i+1}$
- Rozbieżność: $t_i \longrightarrow \infty$
- Automat czasowy:
  - Pobyt w *lokacji* może trwać pewien czas
  - Przejście po krawędzi — natychmiastowe
  - Możliwe przejścia mogą zależeć od czasu

Dokładamy czas

- Znak s_i w słowie σ pojawia się w momencie $t_i \in \mathbb{R}$
- *Słowem* jest para: (napis σ, ciąg czasów τ)
- Monotoniczność: $t_i \leq t_{i+1}$
- Rozbieżność: $t_i \longrightarrow \infty$
- Automat czasowy:
  - Pobyt w *lokacji* może trwać pewien czas
  - Przejście po krawędzi — natychmiastowe
  - Możliwe przejścia mogą zależeć od czasu

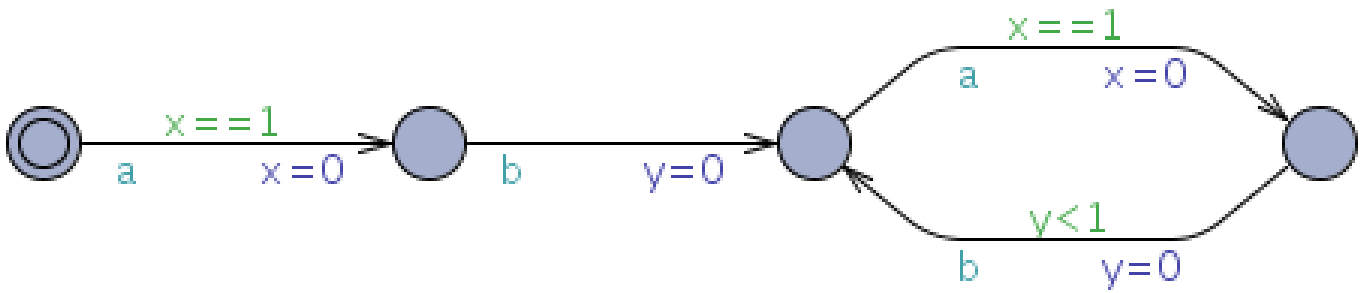
Automat czasowy — dokładniej

- Zbiór zegarów C o zgodnej prędkości (albo i nie...)
- Zegary można zerować przechodząc po krawędziach
- Ograniczenia czasowe na krawędziach:
  - Porównanie zegara z liczbą *wymierną*
    (to prawie jak z całkowitą...)
  - Operacje logiczne
- Determinizm:
  Różne reakcje na ten sam symbol są aktywne w różnych momentach

Automat czasowy — dokładniej

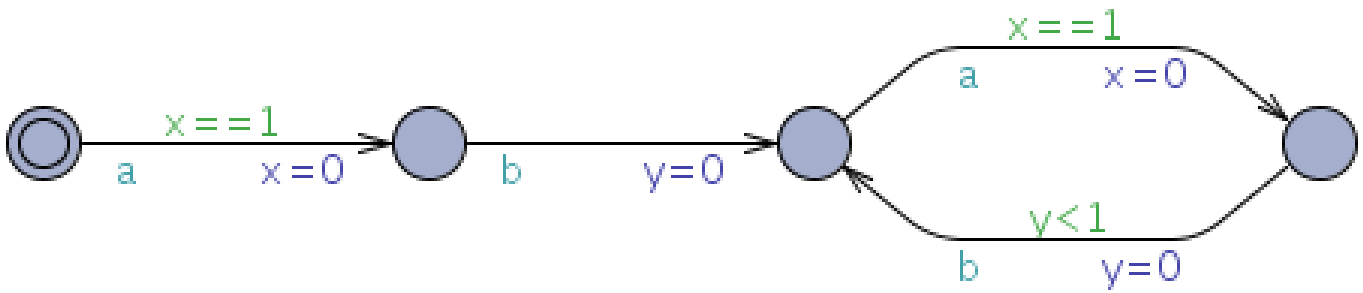
- Zbiór zegarów C o zgodnej prędkości (albo i nie...)
- Zegary można zerować przechodząc po krawędziach
- Ograniczenia czasowe na krawędziach:
  - Porównanie zegara z liczbą *wymierną*
    (to prawie jak z całkowitą...)
  - Operacje logiczne
- Determinizm:
  Różne reakcje na ten sam symbol są aktywne w różnych momentach

Jak $\mathbb{Z}$ wymusza zbieżność



(akceptuje stan po prawej)

Jak $\mathbb{Z}$ wymusza zbieżność



(akceptuje stan po prawej)

Słowo $((ab)^\omega, \tau)$ jest dobre $\Leftrightarrow$ $t_{2i-1} = i$ oraz ciąg $t_{2i} - t_{2i-1}$ maleje

Semantyka

- System przejść między *stanami* postaci:

$$(\text{lokacja } l, \text{wskazania zegarów } \vec{t})$$

- Upływ czasu:

$$(l, \vec{t}) \xrightarrow{\delta} (l, \vec{t} + \delta)$$

- Przejście po krawędzi:

$$(l, \vec{t}) \xrightarrow{a} (l', \vec{t}[Z \mapsto 0]),$$

o ile istnieje krawędź $l \xrightarrow{a} l'$ z ograniczeniami ϕ oraz zbiorem zerowanych zegarów Z, dla której zachodzi $\phi(\vec{t})$

- Oprócz *czasu* jest jeszcze *kolejność*

Semantyka

- System przejść między *stanami* postaci:

$$(\text{lokacja } l, \text{wskazania zegarów } \vec{t})$$

- Upływ czasu:

$$(l, \vec{t}) \xrightarrow{\delta} (l, \vec{t} + \delta)$$

- Przejście po krawędzi:

$$(l, \vec{t}) \xrightarrow{a} (l', \vec{t}[Z \mapsto 0]),$$

  o ile istnieje krawędź $l \xrightarrow{a} l'$ z ograniczeniami ϕ oraz zbiorem zerowanych zegarów Z, dla której zachodzi $\phi(\vec{t})$

- Oprócz *czasu* jest jeszcze *kolejność*

Semantyka

- System przejść między *stanami* postaci:

$$(\text{lokacja } l, \text{wskazania zegarów } \vec{t})$$

- Upływ czasu:

$$(l, \vec{t}) \xrightarrow{\delta} (l, \vec{t} + \delta)$$

- Przejście po krawędzi:

$$(l, \vec{t}) \xrightarrow{a} (l', \vec{t}[Z \mapsto 0]),$$

  o ile istnieje krawędź $l \xrightarrow{a} l'$ z ograniczeniami ϕ oraz zbiorem zerowanych zegarów Z, dla której zachodzi $\phi(\vec{t})$
- Oprócz *czasu* jest jeszcze *kolejność*

Plan

Automaty w UPPAAL-u

- Ograniczenia na pobyt w danej lokacji
- Zmienne lokalne i globalne
  - W ograniczeniach na lokacje/przejścia
  - W akcjach związanych z przejściami
- Zabraniamy alternatywy dla nierówności zegarowych (więc również negacji...)
- Można porównać różnicę dwóch zegarów z liczbą

Automaty w UPPAAL-u

- Ograniczenia na pobyt w danej lokacji
- Zmienne lokalne i globalne
  - W ograniczeniach na lokacje/przejścia
  - W akcjach związanych z przejściami
- Zabraniamy alternatywy dla nierówności zegarowych (więc również negacji...)
- Można porównać różnicę dwóch zegarów z liczbą

Automaty w UPPAAL-u

- Ograniczenia na pobyt w danej lokacji
- Zmienne lokalne i globalne
  - W ograniczeniach na lokacje/przejścia
  - W akcjach związanych z przejściami
- Zabraniamy alternatywy dla nierówności zegarowych
  (więc również negacji...)
- Można porównać różnicę dwóch zegarów z liczbą

Produkt automatów

- Intuicja: automaty pracują „niezależnie”
- Założenia:
  - Własne (rozłączne) zbiory zegarów
  - Własne (niekoniecznie rozłączne) zbiory symboli — ważne!
- Przepis na $A \times B$:
  - Mnożymy zbiory lokacji
  - Sumujemy zbiory zegarów
  - Sumujemy zbiory symboli
  - Przejście $(l, m) \xrightarrow{a} (l', m')$, jeśli:
    - $l \xrightarrow{a} l'$, $m \xrightarrow{a} m'$
    - $l \xrightarrow{a} l'$ i B nie zna a, lub na odwrót
  - Koniunkcja ograniczeń na przejścia/lokacje, suma zerowanych zegarów

Produkt automatów

- Intuicja: automaty pracują „niezależnie”
- Założenia:
  - Własne (rozłączne) zbiory zegarów
  - Własne (niekoniecznie rozłączne) zbiory symboli — ważne!
- Przepis na $A \times B$:
  - Mnożymy zbiory lokacji
  - Sumujemy zbiory zegarów
  - Sumujemy zbiory symboli
  - Przejście $(l, m) \xrightarrow{a} (l', m')$, jeśli:
    - $l \xrightarrow{a} l'$, $m \xrightarrow{a} m'$
    - $l \xrightarrow{a} l'$ i B nie zna a, lub na odwrót
  - Koniunkcja ograniczeń na przejścia/lokacje, suma zerowanych zegarów

Produkt automatów

- Intuicja: automaty pracują „niezależnie”
- Założenia:
  - Własne (rozłączne) zbiory zegarów
  - Własne (niekoniecznie rozłączne) zbiory symboli — ważne!
- Przepis na $A \times B$:
  - Mnożymy zbiory lokacji
  - Sumujemy zbiory zegarów
  - Sumujemy zbiory symboli
  - Przejście $(l, m) \xrightarrow{a} (l', m')$, jeśli:
    - $l \xrightarrow{a} l'$, $m \xrightarrow{a} m'$
    - $l \xrightarrow{a} l'$ i B nie zna a, lub na odwrót
  - Koniunkcja ograniczeń na przejścia/lokacje, suma zerowanych zegarów

Produkt automatów

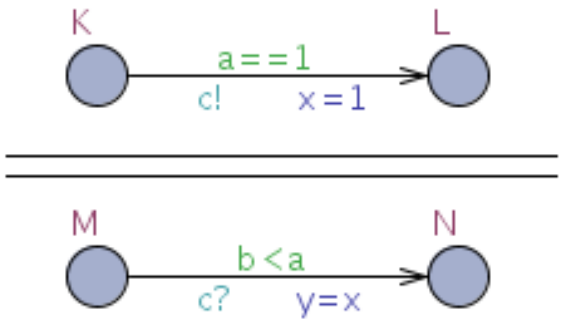
- Intuicja: automaty pracują „niezależnie”
- Założenia:
  - Własne (rozłączne) zbiory zegarów
  - Własne (niekoniecznie rozłączne) zbiory symboli — ważne!
- Przepis na $A \times B$:
  - Mnożymy zbiory lokacji
  - Sumujemy zbiory zegarów
  - Sumujemy zbiory symboli
  - Przejście $(l, m) \xrightarrow{a} (l', m')$, jeśli:
    - $l \xrightarrow{a} l'$, $m \xrightarrow{a} m'$
    - $l \xrightarrow{a} l'$ i B nie zna a, lub na odwrót
  - Koniunkcja ograniczeń na przejścia/lokacje, suma zerowanych zegarów

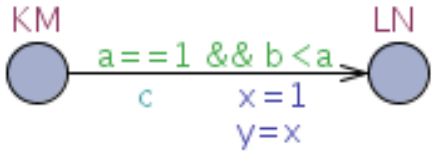
Sieci automatów

- Projekt w UPPAALU to układ automatów (*de facto* produkt)
- Synchronizacja — najprostszy przypadek:
- Synchronizacja binarna/broadcast

Sieci automatów

- Projekt w UPPAALU to układ automatów (*de facto* produkt)
- Synchronizacja — najprostszy przypadek:

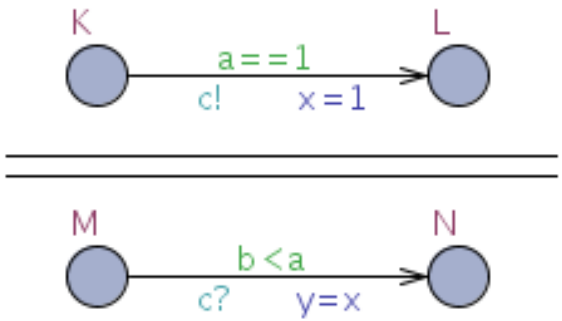


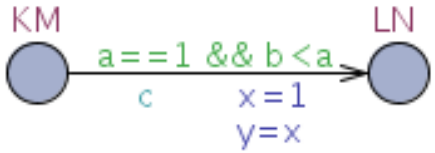


- Synchronizacja binarna/broadcast

Sieci automatów

- Projekt w UPPAALU to układ automatów (*de facto* produkt)
- Synchronizacja — najprostszy przypadek:





- Synchronizacja binarna/broadcast

Dalsze możliwości

- Selekcja — działa jak rodzina krawędzi z parametrem
- *Urgent locations* — nie pozwalają na upływ czasu
- *Committed locations* — j. w. i muszą być opuszczone jako pierwsze (czas vs kolejność...)
- *Urgent synchronisation* — nie można zwlekać, jeśli się da
- ...

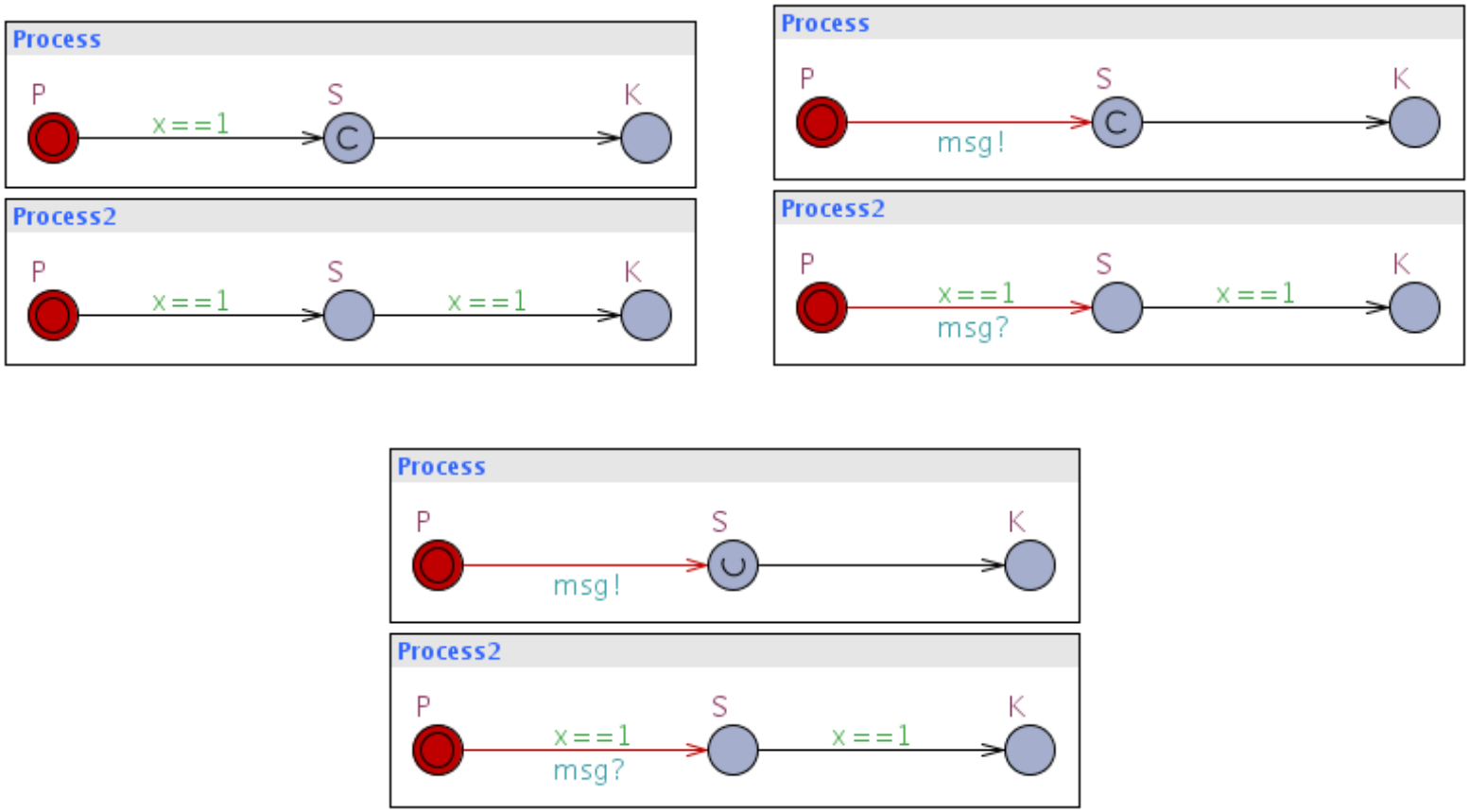
Dalsze możliwości

- Selekcja — działa jak rodzina krawędzi z parametrem
- *Urgent locations* — nie pozwalają na upływ czasu
- *Committed locations* — j. w. i muszą być opuszczone jako pierwsze (czas vs kolejność...)
- *Urgent synchronisation* — nie można zwlekać, jeśli się da
- ...

Dalsze możliwości

- Selekcja — działa jak rodzina krawędzi z parametrem
- *Urgent locations* — nie pozwalają na upływ czasu
- *Committed locations* — j. w. i muszą być opuszczone jako pierwsze (czas vs kolejność...)
- *Urgent synchronisation* — nie można zwlekać, jeśli się da
- . . .

Przykład


Process
P
x==1
S
C
K
Process2
P
x==1
S
x==1
K
Process
P
msg!
S
C
K
Process2
P
x==1
msg?
S
x==1
K
Process
P
msg!
S
U
K
Process2
P
x==1
msg?
S
x==1
K

Specyfikacja weryfikowanych własności

Uproszczony CTL na grafie przejść automatu

- Warunki stanowe:
  - Porównania zmiennych, zegarów, liczb całkowitych
  - Operacje logiczne
  - Kwantyfikatory (po skończonych zbiorach)
  - `A.L` — automat *A* jest w lokacji *L*
  - `deadlock` — nie ma i nie będzie dokąd pójść
- Warunki ścieżkowe: `A[], E[], A<>, E<>, -->`
  - `[]` = zawsze, `<>` = kiedyś
  - `p --> q` $\equiv$ `A[] (p imply A<> q)`
- Bez zagnieżdżania warunków ścieżkowych

Specyfikacja weryfikowanych własności

Uproszczony CTL na grafie przejść automatu

- Warunki stanowe:
  - Porównania zmiennych, zegarów, liczb całkowitych
  - Operacje logiczne
  - Kwantyfikatory (po skończonych zbiorach)
  - `A.L` — automat *A* jest w lokacji *L*
  - `deadlock` — nie ma i nie będzie dokąd pójść
- Warunki ścieżkowe: `A[], E[], A<>, E<>, -->`
  - `[]` = zawsze, `<>` = kiedyś
  - `p --> q` $\equiv$ `A[] (p imply A<> q)`
- Bez zagnieżdżania warunków ścieżkowych

Specyfikacja weryfikowanych własności

Uproszczony CTL na grafie przejść automatu

- Warunki stanowe:
  - Porównania zmiennych, zegarów, liczb całkowitych
  - Operacje logiczne
  - Kwantyfikatory (po skończonych zbiorach)
  - `A.L` — automat *A* jest w lokacji *L*
  - `deadlock` — nie ma i nie będzie dokąd pójść
- Warunki ścieżkowe: `A[], E[], A<>, E<>, -->`
  - `[]` = zawsze, `<>` = kiedyś
  - `p --> q` $\equiv$ `A[] (p imply A<> q)`
- Bez zagnieżdżania warunków ścieżkowych

Semantyka CTL w UPPAAL-u

- `E<> p`: istnieje *skończone* przejście, zakończone stanem spełniającym *p*
- `E[] p`: istnieje maksymalne *(nie)skończone* przejście po stanach spełniających *p*
- `A[] p` ≡ `not E<> not p`
- `A<> p` ≡ `not E[] not p`
- `p --> q` ≡ `A[] (p imply A<> q)`
  (prawa strona niepoprawna w UPPAAL-u)
- Wniosek: własności egzystencjalne w UPPAAL-u są słabsze niż w teorii

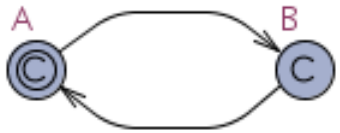
Semantyka CTL w UPPAAL-u

- `E<> p`: istnieje *skończone* przejście, zakończone stanem spełniającym *p*
- `E[] p`: istnieje maksymalne *(nie)skończone* przejście po stanach spełniających *p*
- `A[] p` ≡ `not E<> not p`
- `A<> p` ≡ `not E[] not p`
- `p --> q` ≡ `A[] (p imply A<> q)`
  (prawa strona niepoprawna w UPPAAL-u)
- Wniosek: własności egzystencjalne w UPPAAL-u są słabsze niż w teorii

Semantyka CTL w UPPAAL-u

- `E<> p`: istnieje *skończone* przejście, zakończone stanem spełniającym *p*
- `E[] p`: istnieje maksymalne *(nie)skończone* przejście po stanach spełniających *p*
- `A[] p` $\equiv$ `not E<> not p`
- `A<> p` $\equiv$ `not E[] not p`
- `p --> q` $\equiv$ `A[] (p imply A<> q)` (prawa strona niepoprawna w UPPAAL-u)
- Wniosek: własności egzystencjalne w UPPAAL-u są słabsze niż w teorii

Semantyka CTL w UPPAAL-u — przykład



- Według teorii automat nic nie akceptuje
- Według UPPAAL-a:

  TAK `E<> true`

  TAK `A --> B`

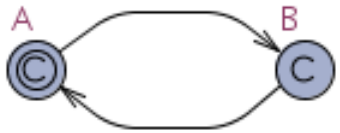
  TAK `B --> A`

  NIE `E<> x == 2`

  NIE `x == 0 --> x == 2`

  TAK `x == 2 --> x == 4`

Semantyka CTL w UPPAAL-u — przykład



- Według teorii automat nic nie akceptuje
- Według UPPAAL-a:

  TAK `E<> true`

  TAK `A --> B`

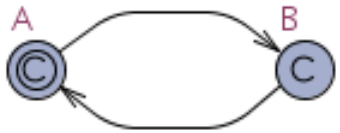
  TAK `B --> A`

  NIE `E<> x == 2`

  NIE `x == 0 --> x == 2`

  TAK `x == 2 --> x == 4`

Semantyka CTL w UPPAAL-u — przykład



- Według teorii automat nic nie akceptuje
- Według UPPAAL-a:

TAK `E<> true`
TAK `A --> B`
TAK `B --> A`
NIE `E<> x == 2`
NIE `x == 0 --> x == 2`
TAK `x == 2 --> x == 4`

Ustawienia weryfikacji

- Generowanie (kontr)przykładów, losowych bądź optymalnych
- Kolejność przeglądania stanów: wgłąb, wszerz, losowo wszerz
- Reprezentacja przestrzeni stanów:
  - DBM — szybkie, pamięciożerne
  - Compact Data Structure — wolniej, mniej pamięci (zwłaszcza przy wielu zegarach)
  - Over Approximation — bywa b. szybkie przy wielu zegarach
  - Under Approximation — pozwala ustalić ilość pamięci

Ustawienia weryfikacji

- Generowanie (kontr)przykładów, losowych bądź optymalnych
- Kolejność przeglądania stanów: wgłąb, wszerz, losowo wszerz
- Reprezentacja przestrzeni stanów:
  - DBM — szybkie, pamięciożerne
  - Compact Data Structure — wolniej, mniej pamięci (zwłaszcza przy wielu zegarach)
  - Over Approximation — bywa b. szybkie przy wielu zegarach
  - Under Approximation — pozwala ustalić ilość pamięci

Ustawienia weryfikacji

- Generowanie (kontr)przykładów, losowych bądź optymalnych
- Kolejność przeglądania stanów: wgłąb, wszerz, losowo wszerz
- Reprezentacja przestrzeni stanów:
  - DBM — szybkie, pamięciożerne
  - Compact Data Structure — wolniej, mniej pamięci (zwłaszcza przy wielu zegarach)
  - Over Approximation — bywa b. szybkie przy wielu zegarach
  - Under Approximation — pozwala ustalić ilość pamięci

Czas weryfikacji

- Test: model mostu kolejowego, 5 razy 14 własności
- CDS, DBM, Over — czasy 30-35 s
- Under: 2s (8B :)), 34s (128K), 93s (64M)

Czas weryfikacji

- Test: model mostu kolejowego, 5 razy 14 własności
- CDS, DBM, Over — czasy 30-35 s
- Under: 2s (8B :)), 34s (128K), 93s (64M)

Czas weryfikacji

- Test: model mostu kolejowego, 5 razy 14 własności
- CDS, DBM, Over — czasy 30-35 s
- Under: 2s (8B :)), 34s (128K), 93s (64M)

Plan

Osiągalność stanu

- Problem: `E<> p` (równoważnie: `not A[] not p`)
- To znaczy: czy da się dojść jakkolwiek do stanu spełniającego `p`?
- Osiągalny stan = osiągalny wierzchołek w grafie stanów
- Problemy:
  - Graf stanów „bardzo nieskończony”
  - Związek między osiągalnością stanu a niepustością języka
- Na ile istotne?

Osiągalność stanu

- Problem: `E<> p` (równoważnie: `not A[] not p`)
- To znaczy: czy da się dojść jakkolwiek do stanu spełniającego `p`?
- Osiągalny stan = osiągalny wierzchołek w grafie stanów
- Problemy:
  - Graf stanów „bardzo nieskończony”
  - Związek między osiągalnością stanu a niepustością języka
- Na ile istotne?

Osiągalność stanu

- Problem: `E<> p` (równoważnie: `not A[] not p`)
- To znaczy: czy da się dojść jakkolwiek do stanu spełniającego `p`?
- Osiągalny stan = osiągalny wierzchołek w grafie stanów
- Problemy:
  - Graf stanów „bardzo nieskończony”
  - Związek między osiągalnością stanu a niepustością języka
- Na ile istotne?

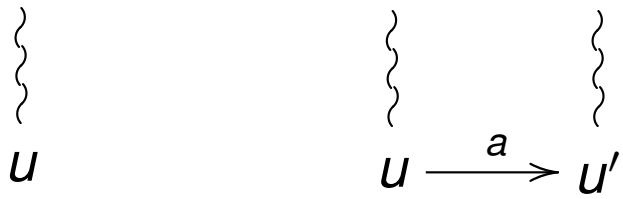
Dobre sklejenia

- Szukamy sklejającej relacji równoważności $\sim$ o dobrych własnościach
  - *Stabilność*: jeśli $\begin{array}{ccc} q & \xrightarrow{a} & q' \\ \wr & & \\ u & & \end{array}$, to $\begin{array}{ccc} q & \xrightarrow{a} & q' \\ \wr & & \wr \\ u & \xrightarrow{a} & u' \end{array}$
  - *Wrażliwość*: przy ustalonym wektorze $\vec{t}$, $\sim$ nie skleja lokacji akceptujących z nieakceptującymi
- Osiągalność stanów akceptujących w grafie jest równoważna osiągalności w ilorazie grafu przez $\sim$

Dobre sklejenia

- Szukamy sklejającej relacji równoważności $\sim$ o dobrych własnościach
  - *Stabilność*: jeśli $q \xrightarrow{a} q'$, to $q \xrightarrow{a} q'$, $q \sim u$, $q' \sim u'$ oraz $u \xrightarrow{a} u'$
  - *Wrażliwość*: przy ustalonym wektorze $\vec{t}$, $\sim$ nie skleja lokacji akceptujących z nieakceptującymi
- Osiągalność stanów akceptujących w grafie jest równoważna osiągalności w ilorazie grafu przez $\sim$

Dobre sklejenia

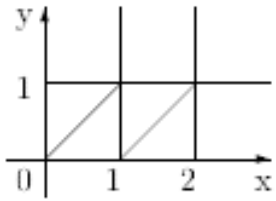
- Szukamy sklejającej relacji równoważności $\sim$ o dobrych własnościach
  - *Stabilność*: jeśli $q \xrightarrow{a} q'$, to
    

  - *Wrażliwość*: przy ustalonym wektorze $\vec{t}$, $\sim$ nie skleja lokacji akceptujących z nieakceptującymi
- Osiągalność stanów akceptujących w grafie jest równoważna osiągalności w ilorazie grafu przez $\sim$

Dobre sklejenia

- Szukamy sklejającej relacji równoważności $\sim$ o dobrych własnościach
  - *Stabilność*: jeśli $q \xrightarrow{a} q'$, to $q \xrightarrow{a} q'$, $q \sim u$, $q' \sim u'$, $u \xrightarrow{a} u'$
  - *Wrażliwość*: przy ustalonym wektorze $\vec{t}$, $\sim$ nie skleja lokacji akceptujących z nieakceptującymi
- Osiągalność stanów akceptujących w grafie jest równoważna osiągalności w ilorazie grafu przez $\sim$

Obszary wektorów czasowych



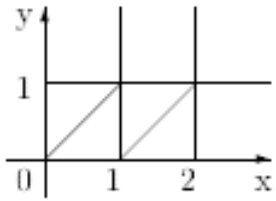
6 Corner points: e.g. $[(0,1)]$

14 Open line segments: e.g. $[0 < x = y < 1]$

8 Open regions: e.g. $[0 < x < y < 1]$

- Skalujemy czas tak, by wszystkie jego ograniczenia były całkowite
- Rozważamy minimalne zbiory, które można określić w dozwolonych formułach
- Wrażliwość — oczywista
- Stabilność — widać na obrazku
- Skończenie wiele!

Obszary wektorów czasowych



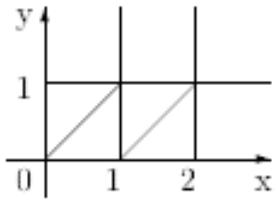
6 Corner points: e.g. $[(0,1)]$

14 Open line segments: e.g. $[0 < x = y < 1]$

8 Open regions: e.g. $[0 < x < y < 1]$

- Skalujemy czas tak, by wszystkie jego ograniczenia były całkowite
- Rozważamy minimalne zbiory, które można określić w dozwolonych formułach
- Wrażliwość — oczywista
- Stabilność — widać na obrazku
- Skończenie wiele!

Obszary wektorów czasowych



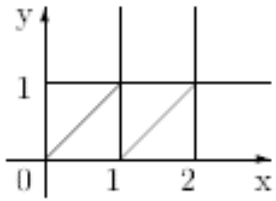
6 Corner points: e.g. $[(0,1)]$

14 Open line segments: e.g. $[0 < x = y < 1]$

8 Open regions: e.g. $[0 < x < y < 1]$

- Skalujemy czas tak, by wszystkie jego ograniczenia były całkowite
- Rozważamy minimalne zbiory, które można określić w dozwolonych formułach
- Wrażliwość — oczywista
- Stabilność — widać na obrazku
- Skończenie wiele!

Obszary wektorów czasowych



6 Corner points: e.g. $[(0,1)]$

14 Open line segments: e.g. $[0 < x = y < 1]$

8 Open regions: e.g. $[0 < x < y < 1]$

- Skalujemy czas tak, by wszystkie jego ograniczenia były całkowite
- Rozważamy minimalne zbiory, które można określić w dozwolonych formułach
- Wrażliwość — oczywista
- Stabilność — widać na obrazku
- Skończenie wiele!

Osiągalność stanu a niepustość języka

- Sprowadzamy osiągalność do niepustości: dodatkowa lokacja, jedyna akceptująca, można wejść jeśli zachodzi p, nie można wyjść
- Sprowadzamy (prawie) niepustość do osiągalności:
  - Nowy automat jest skończony, więc kiedyś się pętlimy
  - Wystarczy pętlić się w stanie akceptującym...
  - ... pod warunkiem, że pętla nie mieści się w jednym obszarze czasowym (?)

Osiągalność stanu a niepustość języka

- Sprowadzamy osiągalność do niepustości: dodatkowa lokacja, jedyna akceptująca, można wejść jeśli zachodzi p, nie można wyjść
- Sprowadzamy (prawie) niepustość do osiągalności:
  - Nowy automat jest skończony, więc kiedyś się pętlimy
  - Wystarczy pętlić się w stanie akceptującym...
  - ... pod warunkiem, że pętla nie mieści się w jednym obszarze czasowym (?)

Osiągalność stanu a niepustość języka

- Sprowadzamy osiągalność do niepustości: dodatkowa lokacja, jedyna akceptująca, można wejść jeśli zachodzi p, nie można wyjść
- Sprowadzamy (prawie) niepustość do osiągalności:
  - Nowy automat jest skończony, więc kiedyś się pętlimy
  - Wystarczy pętlić się w stanie akceptującym...
  - ... pod warunkiem, że pętla nie mieści się w jednym obszarze czasowym (?)

Osiągalność stanu a niepustość języka

- Sprowadzamy osiągalność do niepustości: dodatkowa lokacja, jedyna akceptująca, można wejść jeśli zachodzi p, nie można wyjść
- Sprowadzamy (prawie) niepustość do osiągalności:
  - Nowy automat jest skończony, więc kiedyś się pętlimy
  - Wystarczy pętlić się w stanie akceptującym...
  - ... pod warunkiem, że pętla nie mieści się w jednym obszarze czasowym (?)

Strefy

- Podejście alternatywne wobec idei obszarów
- *Strefa* — zbiór wypukły, opisany przez ograniczenia współrzędnych i ich różnic
- Trzy podstawowe operacje: przekrój, rzutowanie, „rozpóźnienie”

Strefy

- Podejście alternatywne wobec idei obszarów
- *Strefa* — zbiór wypukły, opisany przez ograniczenia współrzędnych i ich różnic
- Trzy podstawowe operacje: przekrój, rzutowanie, „rozpóźnienie”

Automat stref

- Stanami są wszystkie możliwe pary (lokacja l, strefa S)
- Stan $\approx$ zbiór możliwych stanów wyjściowego automatu po danych przejściach
- Przejście: jeśli w oryginalnym automacie $l \xrightarrow{a} l'$ pod warunkiem γ z zerowaniem C, to

$$(l, S) \xrightarrow{a} (l', \pi_C(S\uparrow \cap \gamma))$$

- Automat skończony
- Olbrzymi w porównaniu z automatem obszarów, ale zazwyczaj mniej stanów osiągalnych

Automat stref

- Stanami są wszystkie możliwe pary (lokacja l, strefa S)
- Stan $\approx$ zbiór możliwych stanów wyjściowego automatu po danych przejściach
- Przejście: jeśli w oryginalnym automacie $l \xrightarrow{a} l'$ pod warunkiem γ z zerowaniem C, to

$$(l, S) \xrightarrow{a} (l', \pi_C(S\uparrow \cap \gamma))$$

- Automat skończony
- Olbrzymi w porównaniu z automatem obszarów, ale zazwyczaj mniej stanów osiągalnych

Automat stref

- Stanami są wszystkie możliwe pary (lokacja l, strefa S)
- Stan $\approx$ zbiór możliwych stanów wyjściowego automatu po danych przejściach
- Przejście: jeśli w oryginalnym automacie $l \xrightarrow{a} l'$ pod warunkiem γ z zerowaniem C, to

$$(l, S) \xrightarrow{a} (l', \pi_C(S\uparrow \cap \gamma))$$

- Automat skończony
- Olbrzymi w porównaniu z automatem obszarów, ale zazwyczaj mniej stanów osiągalnych

Automat stref

- Stanami są wszystkie możliwe pary (lokacja l, strefa S)
- Stan $\approx$ zbiór możliwych stanów wyjściowego automatu po danych przejściach
- Przejście: jeśli w oryginalnym automacie $l \xrightarrow{a} l'$ pod warunkiem γ z zerowaniem C, to

$$(l, S) \xrightarrow{a} (l', \pi_C(S\uparrow \cap \gamma))$$

- Automat skończony
- Olbrzymi w porównaniu z automatem obszarów, ale zazwyczaj mniej stanów osiągalnych

Osiągalność

- Osiągalność zbioru stanów $\approx$ osiągalność strefy nierozłącznej ze strefą akceptującą
  (można sprowadzić po prostu do osiągalności strefy akceptującej)
- Stosujemy zwykłe wyszukiwanie w głąb w automacie stref
- Optymalizacja: pomijamy podstany już przejrzanych stanów

Osiągalność

- Osiągalność zbioru stanów $\approx$ osiągalność strefy nierozłącznej ze strefą akceptującą
  (można sprowadzić po prostu do osiągalności strefy akceptującej)
- Stosujemy zwykłe wyszukiwanie w głąb w automacie stref
- Optymalizacja: pomijamy podstany już przejrzanych stanów

Osiągalność

- Osiągalność zbioru stanów $\approx$ osiągalność strefy nierozłącznej ze strefą akceptującą
  (można sprowadzić po prostu do osiągalności strefy akceptującej)
- Stosujemy zwykłe wyszukiwanie w głąb w automacie stref
- Optymalizacja: pomijamy podstany już przejrzanych stanów

DBM

- Reprezentujemy strefę jako graf
  - Wierzchołkami są zegary oraz 0
  - Krawędź $x \xrightarrow{a} y$ oznacza, że $y - x \leq a$
  - Dopuszczamy krawędzie ujemne!
- Domknięcie: stosujemy nierówność trójkąta
- Redukcja: usuwamy niepotrzebne krawędzie $\longrightarrow$ postać kanoniczna

DBM

- Reprezentujemy strefę jako graf
  - Wierzchołkami są zegary oraz 0
  - Krawędź $x \xrightarrow{a} y$ oznacza, że $y - x \leq a$
  - Dopuszczamy krawędzie ujemne!
- Domknięcie: stosujemy nierówność trójkąta
- Redukcja: usuwamy niepotrzebne krawędzie $\longrightarrow$ postać kanoniczna

DBM

- Reprezentujemy strefę jako graf
  - Wierzchołkami są zegary oraz 0
  - Krawędź $x \xrightarrow{a} y$ oznacza, że $y - x \leq a$
  - Dopuszczamy krawędzie ujemne!
- Domknięcie: stosujemy nierówność trójkąta
- Redukcja: usuwamy niepotrzebne krawędzie $\longrightarrow$ postać kanoniczna

Operacje na DBM

- Niesprzeczność $\Leftrightarrow$ brak ujemnych cykli (czyli brak ujemnych pętli po domknięciu)
- Przekrój: sumujemy krawędzie (i redukujemy)
- Rzutowanie: piszemy zera na odp. krawędziach (i redukujemy)
- Rozpóźnienie:
  - Najpierw domykamy
  - Potem usuwamy krawędzie od zera do zegarów (inne zostawiamy)
  - (i redukujemy)

Operacje na DBM

- Niesprzeczność $\Leftrightarrow$ brak ujemnych cykli
  (czyli brak ujemnych pętli po domknięciu)
- Przekrój: sumujemy krawędzie (i redukujemy)
- Rzutowanie: piszemy zera na odp. krawędziach
  (i redukujemy)
- Rozpóźnienie:
  - Najpierw domykamy
  - Potem usuwamy krawędzie od zera do zegarów (inne zostawiamy)
  - (i redukujemy)

Operacje na DBM

- Niesprzeczność $\Leftrightarrow$ brak ujemnych cykli
  (czyli brak ujemnych pętli po domknięciu)
- Przekrój: sumujemy krawędzie (i redukujemy)
- Rzutowanie: piszemy zera na odp. krawędziach
  (i redukujemy)
- Rozpóźnienie:
  - Najpierw domykamy
  - Potem usuwamy krawędzie od zera do zegarów (inne zostawiamy)
  - (i redukujemy)

Redukcja DBM

- Krawędź jest *redundatntna*, jeśli da się dojść krócej
- Usunąć wszystkie redundantne?
  Tak, jeśli nie ma ujemnych *lub zerowych* cykli
- Zerowy cykl oznacza równość wszystkich zegarów
- Metoda:
  - Znajdujemy klasy równoważności
  - Między dwoma klasami zostawiamy ≤ 1 krawędź
  - W każdej klasie zostawiamy jeden rozpinający cykl

Redukcja DBM

- Krawędź jest *redundatntna*, jeśli da się dojść krócej
- Usunąć wszystkie redundantne?
  Tak, jeśli nie ma ujemnych *lub zerowych* cykli
- Zerowy cykl oznacza równość wszystkich zegarów
- Metoda:
  - Znajdujemy klasy równoważności
  - Między dwoma klasami zostawiamy ≤ 1 krawędź
  - W każdej klasie zostawiamy jeden rozpinający cykl

Redukcja DBM

- Krawędź jest *redundatntna*, jeśli da się dojść krócej
- Usunąć wszystkie redundantne?
  Tak, jeśli nie ma ujemnych *lub zerowych* cykli
- Zerowy cykl oznacza równość wszystkich zegarów
- Metoda:
  - Znajdujemy klasy równoważności
  - Między dwoma klasami zostawiamy ≤ 1 krawędź
  - W każdej klasie zostawiamy jeden rozpinający cykl

Redukcja DBM

- Krawędź jest *redundatntna*, jeśli da się dojść krócej
- Usunąć wszystkie redundantne?
  Tak, jeśli nie ma ujemnych *lub zerowych* cykli
- Zerowy cykl oznacza równość wszystkich zegarów
- Metoda:
  - Znajdujemy klasy równoważności
  - Między dwoma klasami zostawiamy ≤ 1 krawędź
  - W każdej klasie zostawiamy jeden rozpinający cykl

Redukcja DBM

- Krawędź jest *redundatntna*, jeśli da się dojść krócej
- Usunąć wszystkie redundantne?
  Tak, jeśli nie ma ujemnych *lub zerowych* cykli
- Zerowy cykl oznacza równość wszystkich zegarów
- Metoda:
  - Znajdujemy klasy równoważności
  - Między dwoma klasami zostawiamy ≤ 1 krawędź
  - W każdej klasie zostawiamy jeden rozpinający cykl

Literatura

- R. Alur, Timed Automata
  `http://www.cis.upenn.edu/~alur/Cav99.ps`
  szersza wersja: `http://www.cis.upenn.edu/~alur/Nato97.ps`
- Strona Uppaala: `http://www.uppaal.com`
  - G. Berman, A. David, K. G. Larsen, A Tutorial on Uppaal
    `http://www.it.uu.se/research/group/darts/papers/texts/new-tutorial.pdf`
  - K. G. Larsen, Formal Methods for Real Time Systems (. . .) (slajdy)
    `http://www.cs.aau.dk/~kgl/ARTES/index.htm`
- R. Alur, D. L. Dill, A Theory of Timed Automata
  (odwrócony porządek stron)
  `http://www.cis.upenn.edu/~alur/Icalp90.ps`