CADP

Sławomir Rudnicki

Niezawodność systemów współbieżnych i obiektowych

18 maja 2011

Wprowadzenie

CADP
(Construction and Analysis of Distributed Protocols)

- Język definiowania protokołów i systemów LOTOS,
- Zestaw narzędzi do manipulacji i weryfikacji systemów.

Program komercyjny, rozwijany przez zespół VASY w INRIA.
Darmowe licencje dostępne dla uczelni.

LOTOS

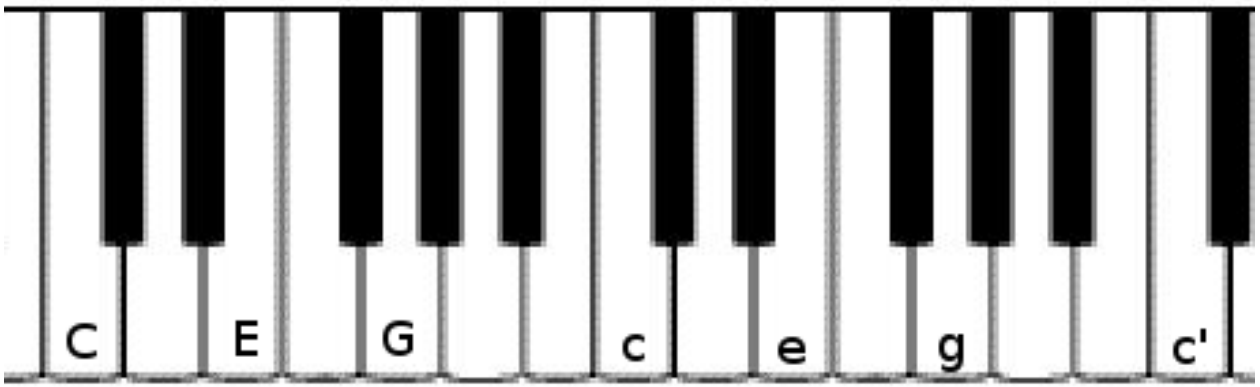
1. Algebra procesów wzorowana na:
   - CCS –
     Calculus of Communicating Processes, Milner 1980
   - CSP –
     Communicating Sequential Processes, Hoare 1978
2. Język do opisu abstrakcyjnych typów danych

LOTOS

- Rozwijany przez grupę ekspertów wewnątrz ISO w latach 1981–1987
- W roku 1989 uznany za międzynarodowy standard ISO 8807
- Sugerowane zastosowania:
  - Modelowanie protokołów dla poszczególnych elementów stosu OSI
  - Inne protokoły komunikacyjne
  - Systemy współbieżne przetwarzające informacje

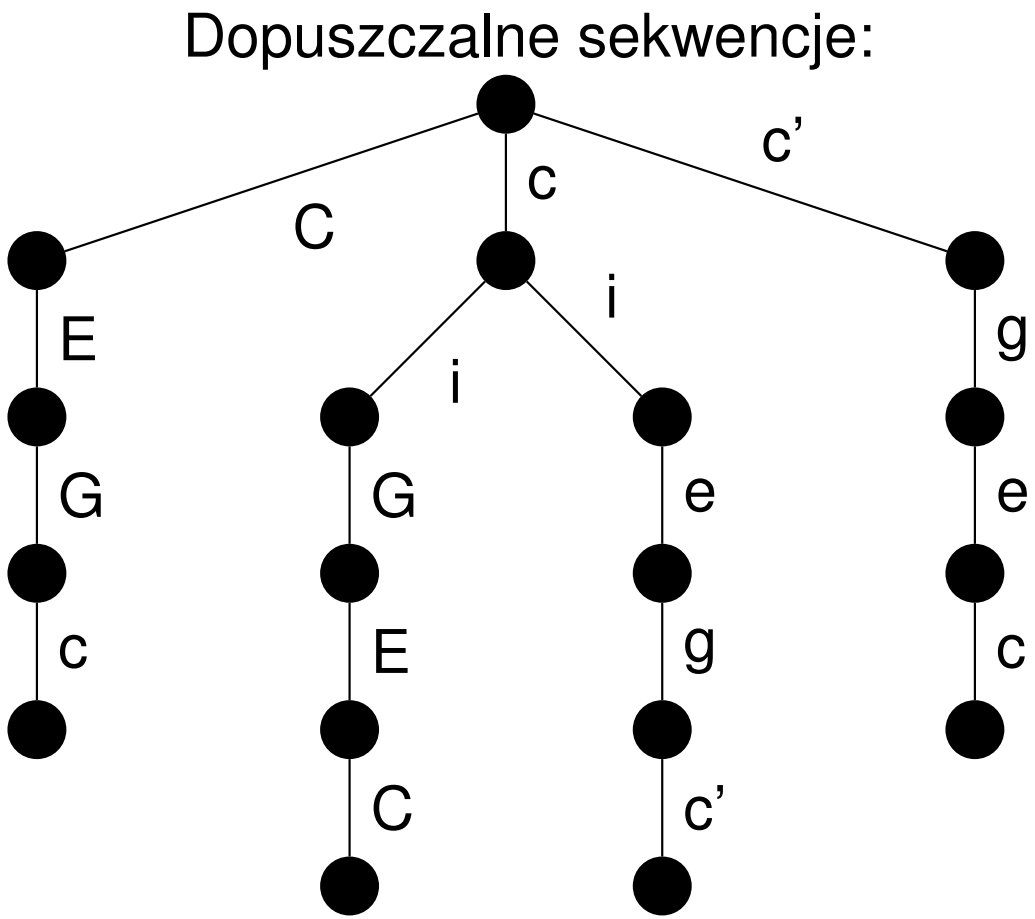
Analogia muzyczna

Protopianola



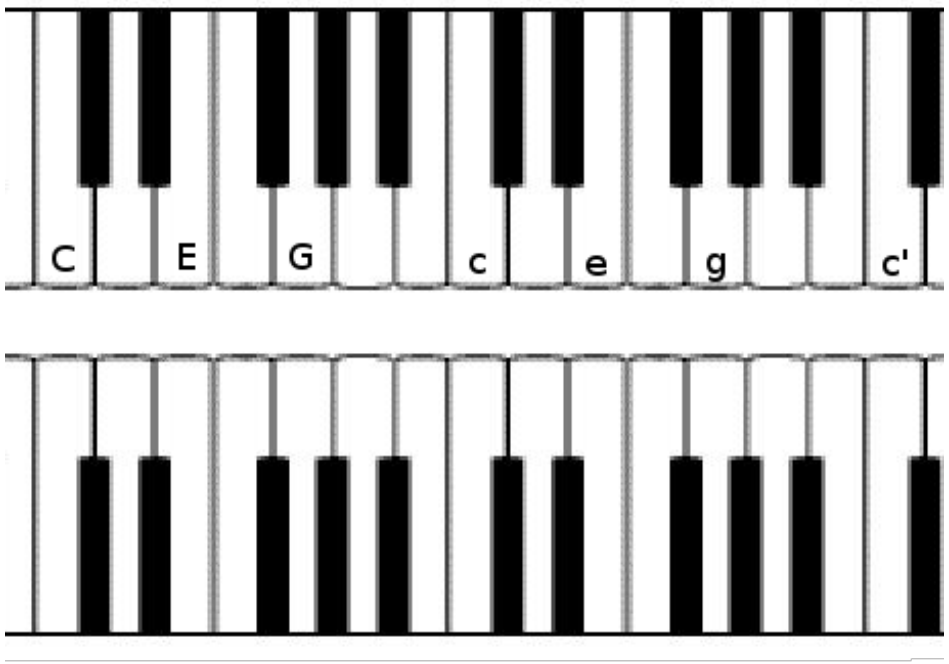
P1[C, E, G, c, e, g, c']

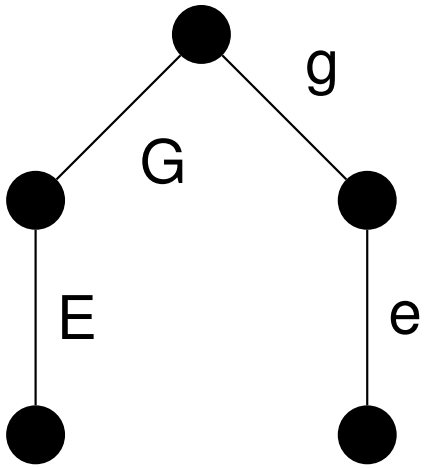
Analogia muzyczna



Analogia muzyczna

Dwie protopianole

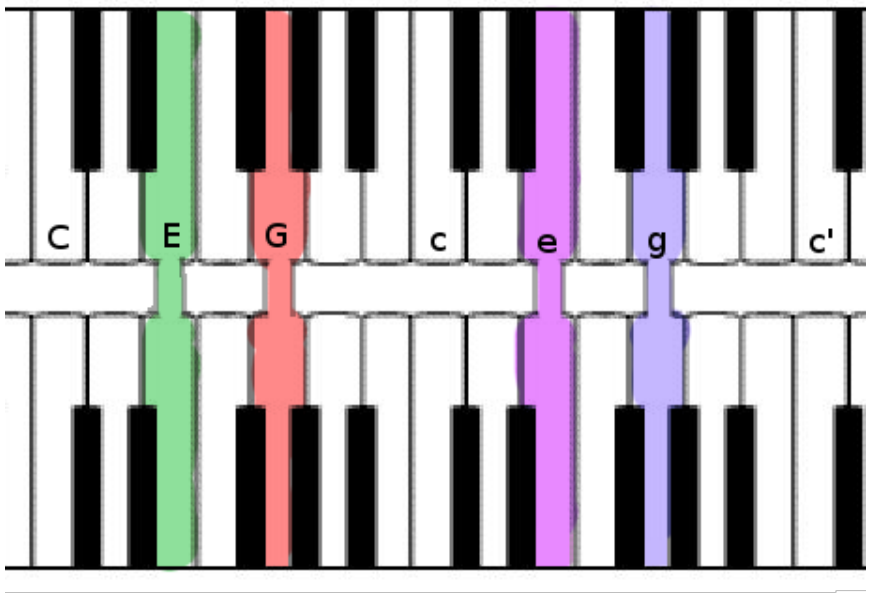


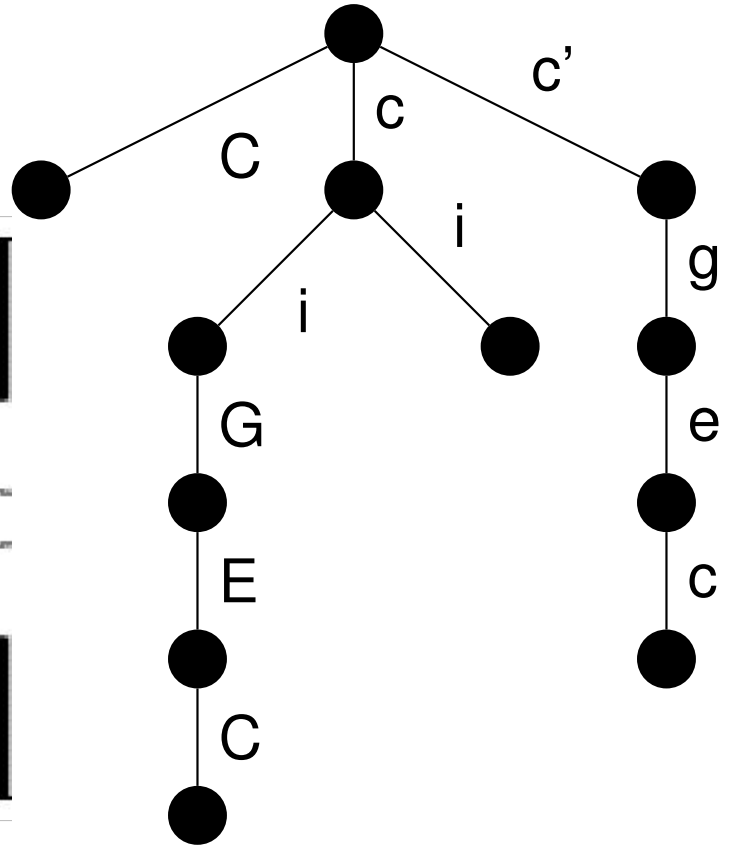


P1[C, E, G, c, e, g, c'] ||| P2[C, E, G, c, e, g, c']

Analogia muzyczna

Dwie połączone protopianole





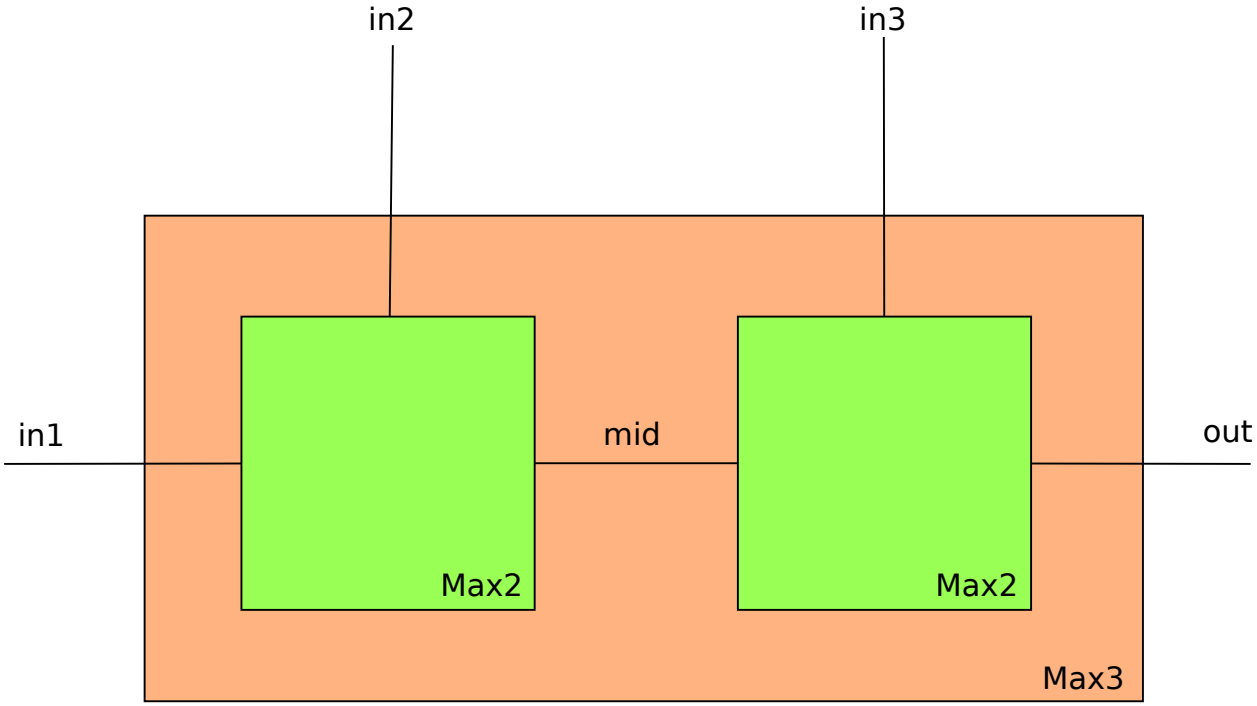
P1[C, E, G, c, e, g, c'] |[E, G, e, g]| P2[C, E, G, c, e, g, c']

Podstawowy LOTOS

- Algebra procesów bez wymiany danych na bramkach.
- Przykładowa definicja procesu:

```
process Max3[in1, in2, in3, out] :=
  hide mid in
    Max2[in1, in2, mid] |[mid]| Max2[mid, in3, out]
  where
    process Max2[a, b, c] :=
      a;b;c;stop
      []
      b;a;c;stop
    endproc
endproc
```

Ukryta bramka



Bramka `mid` jest ukryta – nie ma do niej dostępu z zewnątrz procesu `Max3`.

Zachowanie procesu: programy sekwencyjne

- Proces bezczynności:

  stop

- Zakończenie procesu:

  exit

- Złożenie sekwencyjne:

  $\mathbf{i}; P$

  $a; P$

- Wybór

  $P_1 [] P_2$

- Wywołanie procesu:

  $P[g_1, g_2, \ldots, g_n]$

  Pozwala uzyskać rekurencję lub iterację.

Zachowanie procesu: współbieżność

- Złożenie współbieżne z synchronizacją:

$$P_1 \,|[\mathbf{g_1}, \mathbf{g_2}, \ldots, \mathbf{g_n}]|\, P2$$

  Procesy synchronizują się na bramkach $g_1, \ldots, g_n$. Pozostałe akcje obu procesów są niezależne.
- Czysta współbieżność:

$$P_1 ||| P_2$$

  Bez synchronizacji
- Pełna synchronizacja:

$$P_1 || P_2$$

  Synchronizacja na wszystkich bramkach

Zachowanie procesu: relacje między procesami

- Wyzwolenie:

$$P_1 >> P_2$$

Proces P_2 może działać dopiero po zakończeniu się procesu P_1 Przykład:

$$(a; b; \mathbf{exit} ||| a; c; \mathbf{stop}) >> P[\ldots]$$

Nie zadziała!

- Blokowanie:

$$P \ [> \ B$$

Proces B może w każdym momencie przerwać działanie P.

Rozszerzony LOTOS

Do algebry procesów zostaną dodane wartości, które mogą być przekazywane pomiędzy procesami

- Deklaracja bramki:

$$g_i \langle \textbf{true}, \text{“}foo\text{“}, 3 \rangle$$

- Deklaracja wartości

$$!(x + 1)$$

- Deklaracja zmiennej

$$?x : nat$$

Komunikacja między procesami

- Synchronizacja:

  P_1 :: g $!E$

  P_2 :: g $!E$

- Przekazywanie wartości:

  P_1 :: g $!E$

  P_2 :: g $?x : t$

- Uzgadnianie wartości

  P_1 :: g $?x : t$

  P_2 :: g $?y : u$

Konstrukcje warunkowe

- Warunki wyboru:

$$g?x : nat[x < 3]$$

Przykład: negocjacja parametru

```
hide g in
  g ?x:nat[x<max]; B1(x)
  |[g]|
  g ?y:nat[y>min]; B2(y)
```

- Instrukcje dozorowane:

$$[x > 0] \rightarrow E$$

Przykład: wartość bezwzględna

```
    [x>0] -> sap !x; P[...](x, ...)
[] [x<=0] -> sap !-x; P[...](x, ...)
```

Wybór niedeterministyczny

- Wybór wartości danego typu:

$$\textbf{choice } x : t \text{ [] } \textbf{i}\,;\, B(x)$$

- Wybór bramki

$$\textbf{choice } g \textbf{ in } [a_1, \ldots, a_n] \text{ [] } P[g]$$

Składanie sekwencyjne z przekazywaniem wartości

- Przekazywanie wartości przez proces kończący się:

$$\textbf{exit}(x, y)$$

- Przejmowanie wartości przez nowy proces:

$$B_1 >> \textbf{accept}\ x_1 : t_1, \ldots, x_n : t_n\ \textbf{in}\ B_2$$

- Funkcjonalność procesu:

$$\textbf{process}\ P : \mathit{func} := \ldots$$

$$\textbf{noexit} \qquad \textbf{exit} \qquad \textbf{exit}(t_1, \ldots, t_n)$$

Abstrakcyjne typy danych

LOTOS pozwala na definiowanie typów danych przez konstruktory i aksjomaty.
Przykład: definicja liczb naturalnych z dodawaniem:

- Nośnik: $\mathcal{N}$
- Sygnatura: $\langle 0, s, + \rangle$
- Aksjomaty:

$$x + 0 = x$$

$$x + s(y) = s(x + y)$$

Abstrakcyjne typy danych

```
type Nat_numbers is
  sorts nat
  opns 0 : -> nat
       s : nat -> nat
       _+_ : nat, nat -> nat.
  eqns
       forall x, y : nat
       ofsort nat
         x + 0 = x;
         x + s(y) = s(x+y);
endtype
```

CAESAR

- Kompiluje opis zachowania procesów do:
  - C – do symulacji przez inne narzędzia
  - LTS (Labelled Transition System) – do weryfikacji
- Akceptuje rozszerzony język LOTOS bez rekurencji procesów po lewej stronie $>>$ i $[>$ oraz po obu stronach $|[\ldots]|$.
- W rozsądnym czasie buduje systemy o kilku milionach stanów.

CAESAR.ADT

- Kompilator opisów abstrakcyjnych typów danych do wykorzystania w C.
  - Typy definiowane w LOTOS-ie są przekształcane w typy C
  - Konstruktory są przekształcane w funkcje C

BCG

- BCG jest binarnym formatem przechowywania struktur Kripkego oraz zbiorem narzędzi do manipulacji nimi. Wybrane narzędzia:
  - BCG_DRAW
  - BCG_EDIT
  - BCG_MIN
  - BCG_INFO

OPEN/CAESAR

- Framework do tworzenia narzędzi do:
  - symulacji
  - uruchamiania
  - weryfikacji

  działające na różnych reprezentacjach przestrzeni stanów.

OPEN/CAESAR: wybrane moduły

- Symulatory:
  OCIS, XSIMULATOR, SIMULATOR
- GENERATOR:
  generator BCG z innych formatów z analizą osiągalności
- EXECUTOR:
  losowe wykonanie modelu
- EXHIBITOR:
  wyszukiwanie sekwencji

BISIMULATOR

- Weryfikuje odpowiedniość dwóch modeli względem zadanej relacji równoważności.

BISIMULATOR: relacje równoważności

- Silna odpowiedniość:

$$p \xrightarrow{b} p' \Rightarrow q \xrightarrow{b} q' \wedge p' \sim q'$$

- Równoważność gałęziowa:

$$p \xrightarrow{\tau} p' \Rightarrow p' \sim q$$

$$p \xrightarrow{a} p' \Rightarrow q \xrightarrow{\tau^*} q' \xrightarrow{a} q'' \wedge p \sim q' \wedge p' \sim q''$$

- Równoważność obserwacyjna:

$$p \xrightarrow{\tau} p' \Rightarrow q \xrightarrow{\tau^*} q' \wedge p' \sim q'$$

$$p \xrightarrow{a} p' \Rightarrow q \xrightarrow{\tau^* a \tau^*} q' \wedge p' \sim q'$$

- Równoważność $\tau^* a$:

$$p \xrightarrow{\tau^* a} p' \Rightarrow q \xrightarrow{\tau^* a} q' \wedge p' \sim q'$$

REDUCTOR

- Redukuje rozmiar przestrzeni stanów, zachowując równoważność względem wybranych relacji. Wybrane redukcje:
  - redukcja zduplikowanych tranzycji,
  - τ-dywergencja: zastąpienie silnie spójnych składowych z τ-przejściami pojedynczą pętlą,
  - τ-kompresja: usunięcie silnie spójnych składowych z τ przejściami,
  - $\tau^* a$: eliminacja wszystkich τ-przejść.

EVALUATOR

Umożliwia weryfikację własności definiowanych w regularnym μ-rachunku modalnym.

EVALUATOR: język opisu własności

- $\mathcal{A}$ – Formuła logiczna zbudowana z wyrażeń regularnych opisujących akcje, np.:

true

"$a\,!\,TRUE$" **or** "$b\,!\,FALSE$"

"$ba(b|c)(a*)$" **or** "$.*!3$"

- $\mathcal{R}$ – Wyrażenia regularne nad $\mathcal{A}$ z operacjami

$$R_1.R_2 \quad R_1|R_2 \quad R* \quad R^+$$

EVALUATOR: język opisu własności

- $\mathcal{F}$ – Formuły logiki modalnej:

$$\textbf{true} \quad \textbf{false} \quad \neg F \quad F_1 \overset{\wedge \vee}{\Rightarrow} F_2$$

$$\langle R \rangle F \quad [R] F$$

$$@R$$

$$\mu X.F \quad \nu X.F$$

EVALUATOR: przykłady

- Brak zakleszczeń:

$$[\mathbf{true}^*]\,\langle \mathbf{true} \rangle$$

- Bezpieczeństwo:

$$[\mathbf{true}^*.\mathit{Error}]\,\mathbf{false}$$

- Żywotność:

$$\langle \mathbf{true}^*.\mathit{Ok} \rangle\,\mathbf{true}$$

$$\mu X.\,(\langle \mathbf{true} \rangle\,\mathbf{true}\ \mathbf{and}\ [\mathbf{not}\ \mathit{Ok}]\,X)$$

XTL

EXecutable Temporal Language – funkcyjny język opisu własności przestrzeni stanów z akcjami.

- Pozwala na wiązanie w zmienne danych zawartych w etykietach tranzycji (np. przekazywanych pomiędzy procesami) i manipulację nimi. Można napisać:

```
Box (SEND ?M:Msg, EF_A (true, Dia (RECV !M, true)))
```

(“każda wiadomość, która zostaje wysłana, może zostać odczytana”)

XTL: przykłady

```
def Dia (A:labelset, F:stateset) : stateset =
    { S:state where
        exists T:edge among out (S) in
            (label (T) among A) and (target (T) among F)
        end_exists
    }
end_def
```

$$Dia(A, F) \equiv \langle A \rangle\, F$$

XTL: przykłady

```
def EF_A (A:labelset, F:stateset) =
    for in X:stateset
        while X <> (F or Dia (A, X))
        apply or
        from false
        to F or Dia (A, X)
    end_for
end_def
```

$$\mathrm{EF\_A}(A, F) \equiv \mu X.\,(F \vee \langle A \rangle\, X) \equiv \mathbf{EF}_A(F)$$

XTL – biblioteki

Istnieją biblioteki, które nadbudowują nad XTL inne logiki modalne (np. CTL, ACTL) umożliwiając tworzenie zwięzłych formuł:

```
INIT implies AG_A (not (PUT), Box (GET, false))
```

(“Na początku nie można osiągnąć akcji GET przed akcją PUT”)

I więcej...

CADP udostępnia również:

- Analizę łańcuchów Markowa z czasem ciągłym,
- Rozpraszanie obliczeń na wiele stacji roboczych,
- Testowanie pod względem zadanych scenariuszy,
- Graficzny interfejs użytkownika.

Case studies

Na stronie projektu CADP znajduje się lista ponad 130 studiów przypadku z lat 1990–2010, w których narzędzie zostało wykorzystane.

- Algorytmy współbieżne:
  - wzajemne wykluczanie,
  - wybór lidera,
- Protokoły komunikacyjne i kryptograficzne:
  - CHAP,
  - TCP,
  - protokół Needhama-Schrödera,
- Zastosowania biznesowe:
  - chilijski system przesyłania faktur,
  - holenderski system wyborów on-line,
  - Air Traffic Control – system kontroli lotów.

Bibliografia

- Bolognesi T., Brinksma E., *Introduction to the ISO Specification Language LOTOS*, Computer Networks and ISDN Systems, 1987.
- Mateescu R., Garavel H., *XTL: A Meta-Language and Tool for Temporal Logic Model-Checking*, 1998.
- CADP Manual Pages