ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1989
Seria: TRANSPORT z. 13 Nr kol. 1034

Roman KONIECZNY

ZAGADNIENIE ROZSZERZENIA LOGLANOWSKICH ZASOBÓW SYMULACYJNYCH

Streszczenie. Artykuł niniejszy prezentuje wybrane aspekty dotyczące rozszerzenia zasobów symulacyjnych języka LOGLAN (nie uwzględnionych w klasie SIMULATION). Aspektami tymi są: generatory liczb pseudolosowych oraz zagadnienia symulacji zdarzeniowo-ciągłej.

Funkcjonalnie dodatkowe zasoby symulacyjne mogą być dołączone do LOGLANu jako niezależne pakiety nowych procedur źródłowych lub poprzez rozbudowę klasy SIMULATION.

Przedstawione w artykule zagadnienia sygnalizują zagadnienie rozszerzania loglanowskich zasobów symulacyjnych.

1. Uwagi wstępne

Standardowe zasoby symulacyjne języka LOGLAN zostały przedstawione w artykule [1]. Zasoby te mogą być rozszerzone o nowe elementy, wynikające z potrzeb realizacji przebiegów symulacyjnych. Klasycznymi takimi potrzebami są, np. generatory liczb pseudolosowych, histogramy, procedury analizy wyników... a także zagadnienia łączenia dwóch typów symulacji: dyskretnej (opartej na zdarzeniach) oraz ciągłej (w znaczeniu np. języka CSMP [4]). Artykuł niniejszy prezentuje wybrane aspekty dotyczące rozszerzenia zasobów symulacyjnych języka LOGLAN (nie uwzględnionych w klasie SIMULATION). Aspektami tymi są: generatory liczb pseudolosowych oraz zagadnienia symulacji zdarzeniowo-ciągłej.

Funkcjonalnie dodatkowe zasoby symulacyjne mogą być dołączone do LOGLANu jako niezależne pakiety nowych procedur źródłowych lub poprzez rozbudowę klasy SIMULATION.

2. Generatory liczb pseudolosowych

Standardowy loglanowski generator liczb pseudolosowych (funkcja RANDOM oraz procedura startowa RANSET - pokazane w przykładach [2]) nie wystarczają do prowadzenia symulacji z szerszym wariantowaniem danych wejściowych. Uznając tę potrzebę, w roku 1987 pod naukowym kierunkiem prof. A. Salwickiego, została wykonana praca [3], w której zrealizowano następujące moduły programowe:

UNIT *trial* : function (a : real) : boolean ;

Dla $0 < a < 1$ funkcja *trial* przyjmuje wartość TRUE z prawdopodobieństwem a oraz wartość FALSE z prawdopodobieństwem $1-a$.

Dla $a \geq 1$ funkcja przyjmuje wartość TRUE .

Dla $a \leq 0$ funkcja przyjmuje wartość FALSE .

UNIT *randint* : function (a , b : integer) : integer ;

Funkcja *randint* przyjmuje z jednakowym prawdopodobieństwem jedną z wartości $a, a+1, \ldots, b-1, b$ dla $a \leq b$.

Jeśli $b < a$, to funkcja zwróci wartość 0 oraz wypisze komunikat: "*Error in actual parameters in randint. The first one mustn't be greater than the second one.*" i wartości parametrów aktualnych.

UNIT *uniform* : function (a , b : real) : real ;

Funkcja *uniform* dla $a \leq b$ jest liczbą rzeczywistą z przedziału $[a , b)$ generowana zgodnie z rozkładem równomiernym.

Jeśli $b < a$, to funkcja zwróci wartość 0 oraz wypisze komunikat: "*Error in actual parameters in uniform. The first one mustn't be greater than the second one.*" i wartości parametrów aktualnych.

UNIT *normal* : function (m , s : real) : real ;

Wartością funkcji *normal* jest liczba rzeczywista generowana zgodnie z rozkładem normalnym $N(m,s)$ o wartości średniej m i odchyleniu standardowym s i gęstości równej:

$$f(x) = \frac{1}{\sqrt{2\pi}\, s} \exp\left[- \frac{(x-s)^2}{2s^2} \right] , \qquad -\infty < x < +\infty$$

UNIT *negexp* : function (c : real) : real ;

Dla $c > 0$ wartością funkcji *negexp* jest liczba generowana zgodnie z ujemnym rozkładem wykładniczym o dystrubuancie:

$$F_X(x) = \begin{cases} 1 - e^{-cx} & \text{dla } x \geq 0 \\ 0 & \text{dla } x < 0 \end{cases}$$

i wartości średniej $1/c$.

Dla $c \leq 0$ funkcja zwróci wartość 0 oraz wypisze komunikat: "*Wrong parameter in negexp. It must be positive.*" i wartość parametru aktualnego.

UNIT *Poisson* : function (c : real) : real ;

Dla $c > 0$ wartością funkcji *Poisson* jest liczba całkowita generowana zgodnie z rozkładem Poissona z wartoscią średnią równą c .

Dla $c \leq 0$ funkcja zwróci wartość 0 oraz wypisze komunikat: "*Wrong parameter in Poisson. It must be positive.*" i wartość parametru aktualnego.

UNIT *Erlang* : function (a , b : real) : real ;

Dla $a,b > 0$ wartością funkcji *Erlang* jest liczba rzeczywista generowana zgodnie z rozkładem Erlanga z wartością średnią równą $1/a$ oraz odchyleniem standardowym równym $1/(a\sqrt{b})$.

Dla a lub $b \leq 0$ funkcja zwróci wartość 0 oraz wypisze komunikat: "*Error in parameters in Erlang. Both must be positive.*" i wartości parametrów aktualnych.

UNIT *discrete* : function (a : arrayof real) : integer ;

Wektor a interpretowany jest jako funkcja skokowa indeksów definiująca dystrybuantę skokową. Zakłada się, że $a(i) < a(j)$ dla $i < j$ oraz, że ostatni element wektora a jest równy 1 .

Jeśli wektor a jest zbudowany poprawnie, to wartością funkcji *discrete* jest liczba całkowita z przedziału $[d,g]$, gdzie d,g są odpowiednio dolnym i górnym ograniczeniem indeksów wektora a .
Liczba ta jest równa min (k: $a(k) > R$), gdzie R jest wartością losową z przedziału [0,1] .

Jeśli któryś z warunków określających poprawność wektora a nie jest spełniony, to funkcja zwróci wartość 0 oraz wypisze komunikat "*Wrong parameter in discrete.*" .

UNIT *linear* : function (a , b : arrayof real) : real ;

Wartością funkcji *linear* jest liczba rzeczywista generowana zgodnie z rozkładem, którego dystrybuanta F jest otrzymana przez liniową interpolację zadaną elementami tablic a i b takich, że $a(i) = F(b(i))$. Zakłada się, że a i b są wektorami tej samej długości. Pierwszy element wektora a jest równy 0, ostatni zaś równy 1 oraz zachodzi: $a(i) \geq a(j)$, $b(i) > b(j)$ dla $i > j$.

Jeśli któryś z tych warunków nie jest spełniony, to funkcja zwróci wartość 0 oraz wypisze komunikat: "*Error in parameters in linear.*" .

UNIT *Bernoulli* : function (n : integer , p : real) : integer ;
UNIT *binomial* : function (n : integer , p : real) : integer ;

Wartością obu funkcji dla $n > 0$ i $0 < p < 1$ jest liczba całkowita generowana zgodnie z rozkładem Bernoulliego z parametrami n, p. Dla parametrów nie spełniających tych ograniczeń funkcje zwrócą wartość 0 oraz wypiszą komunikat "*Error in parameters in Bernoulli (binomial). There must be* $n > 0$ *and* $0 < p < 1$." i wartości parametrów aktualnych.

Przydatność tych funkcji zależna jest od wielkości parametrów. Metoda zastosowana w *Bernoulli* pozwala na szybszą generację liczb o tym rozkładzie, ale dla ograniczonego zbioru parametrów, co spowodowane jest wynikającymi w trakcie obliczeń błędami zaokrągleń. Metoda użyta w *binomial* jest wolniejsza w wykonaniu, ale za to bardziej uniwersalna.

UNIT *geom* : function (p : real) : integer ;

Dla $0 < p < 1$ wartością funkcji *geom* jest liczba całkowita generowana zgodnie z rozkładem geometrycznym o parametrze p.

Dla p spoza tego przedziału funkcja zwróci wartość 0 oraz wypisze komunikat: "*Wrong parameter in geom. There must be* $0 < p < 1$." i wartość parametru aktualnego.

UNIT *beta* : function (a , b : real) : real ;

Dla $a, b > 0$ wartością funkcji *beta* jest liczba rzeczywista generowana zgodnie z rozkładem beta z parametrami a, b o gęstości zadanej wzorem:

$$f(x) = \begin{cases} \dfrac{1}{B(a,b)} x^{a-1}(1-x)^{b-1} & \text{dla } 0 \le x \le 1 \\ 0 & \text{dla } x < 0 \text{ lub } x > 1 \end{cases}$$

gdzie:

$B(a,b) = \Gamma(a) * \Gamma(b) * \Gamma(b) / \Gamma(a+b)$;

$\Gamma(a)$ jest funkcją gamma .

Dla a lub $b \le 0$ funkcja zwróci wartość 0 oraz wypisze komunikat: "*Error in parameters in beta. Both must be positive.*" i wartości parametrów aktualnych.

UNIT *gamma* : function (a : real) : real ;

Dla $a > 0$ wartością funkcji *gamma* jest liczba rzeczywista generowana zgodnie z rozkładem gamma z parametrem a o gęstości zadanej wzorem:

$$f(x) = \begin{cases} \dfrac{1}{\Gamma(a)} x^{a-1} e^{-x} & \text{dla } x > 0 \\ 0 & \text{dla } x \le 0 \end{cases}$$

gdzie:

$\Gamma(a)$ jest funkcją gamma .

Dla $a \le 0$ funkcja zwróci wartość 0 oraz wypisze komunikat: "*Wrong parameter in gamma. It must be positive.*" i wartość parametru aktualnego.

Moduły powyższe mogą być dołączone do programu symulacyjnego na zasadzie łączenia tekstów, według następującego schematu:

```
PROGRAM nazwa ;

(* ------------------------------------------ *)
   .......
   .......  grupa procedur pseudolosowych
   .......  skopiowana np. z pliku PP_LOS.LOG
   .......
(* ------------------------------------------ *)
   .......
   .......  treść klasy SIMULATION
   .......  skopiowana np. z pliku KLASSIM.LOG
   .......
(* ------------------------------------------ *)
   .......
   .......  dalej procedury
   .......  i program główny użytkownika
   .......

END .
```

3. Symulacja dyskretno-ciągła

Niekiedy w zagadnieniach symulacji systemów zachodzi potrzeba tzw. "mieszanego" (hybrydowego) podejścia do projektowania programów symulacyjnych. Sytuacja taka ma miejsce w przypadku, gdy chcemy połączyć w jedną całość dwa modele (submodele):

- pierwszy – bazujący na zdarzeniach (*wyróżniona oś czasu z planem zdarzeń*),
- drugi – bazujący na układach ciągłych opisanych równaniami różniczkowymi (*oś czasu nie jest wyróżniona explicite, zaznaczony jest tylko czas początku i końca przebiegu*).

Przykładem takiego modelu systemu może być LOKOMOTYWA ELEKTRYCZNA, gdzie elementami podsystemu dyskretnych zdarzeń będą: maszynista oraz otoczenie ruchowe (semafory, znaki kolejowe...); natomiast elementami podsystemu ciągłego będą: otoczenie zasilaniowe (np. napięcie w sieci trakcyjnej), otoczenie topograficzne (np. profil trasy) oraz podzespoły lokomotywy (np. grupa silników – opisana układem równań różniczkowych zwyczajnych). Zachowanie elementów podsystemu zdarzeniowego będzie miało wpływ ma pracę podsystemu ciągłego, i odwrotnie (*np. zmiana stanu semafora z sygnału STÓJ na sygnał WOLNA DROGA (zdarzenie) dla pociągu w stanie zatrzymania spowoduje przełączenie silników w fazę rozruchu a następnie w fazę pracy ustalonej (proces ciągły); zbyt niskie napięcie w sieci trakcyjnej (proces ciągły) – mimo sygnału WOLNA DROGA – może spowodować uniemożliwienie rozruchu lokomotywy (opóźnienie zdarzenia); pojawienie się nowego pociągu (zdarzenie) w danej sekcji zasilania będzie powodem obniżenia napięcia w sieci trakcyjnej (proces ciągły)*). Można ogólnie powiedzieć, że praca podzespołów LOKOMOTYWY ELEKTRYCZNEJ (podsystem procesów ciągłych) jest

sterowana przez podsystem procesów zdarzeniowych.

Podstawowa trudność w tego typu symulacji polega na tym, że konwencjonalne języki symulacji dyskretnych zdarzeń, jak np. GPSS czy CSL, nie przewidują możliwości włączenia procesów ciągłych. Podobnie w językach symulacyjnych przeznaczonych do klasycznej symulacji ciągłej, jak np. CSMP, MIMIC czy CEMMA, nie przewiduje się włączania procesów zdarzeniowych. Realizowanie symulacji mieszanej w ww. językach wymaga stosowania pewnych "chwytów" programowych, nie zawsze czytelnych z punktu widzenia opisu modelu.

Te dwie diametralnie różne techniki symulacyjne (dyskretno zdarzeniowa oraz ciągła) mogą być łatwo połączone w językach LOGLAN oraz SIMULA. Ogólne zasady takiego postępowania w języku LOGLAN mogą być następujące:

- wszystkie procesy zdarzeniowe są simprocesami, bazującymi na klasie SIMULATION;
- wszystkie procesy ciągłe są współprogramami, nie korzystającymi z zasobów klasy SIMULATION;
- procesy zdarzeniowe nie wymagają żadnych modyfikacji (w znaczeniu symulacji zdarzeń), poza włączeniem instrukcji aktywacji procesów ciągłych;
- planowanie zdarzeń odbywa się wyłącznie dla procesów zdarzeniowych, które z kolei uaktywniają w miarę potrzeby współpracujące z nimi procesy ciągłe;
- działanie procesów ciągłych polega na dokonywaniu obliczeń przebiegu swoich zmiennych stanu w czasie międzyzdarzeniowym (oznacza to, że każdy proces zdarzeniowy oddający lub przejmujący sterowanie powinien uruchomić wszystkie procesy ciągłe; uruchomienie to może być dokonane "w przód" tj. na okres czasu jaki upłynie do najbliższego zdarzenia lub "w tył" tj. na okres czasu jaki upłynął od zdarzenia poprzedniego do chwili bieżącej);
- wszystkie procesy ciągłe mogą być umieszczone na liście procesów ciągłych i uaktywniane przez specjalnie do tego celu przeznaczony współprogram;
- powiązanie procesów (współpraca) realizowane jest na ogólnych zasadach stosowanych w programowaniu quasi-równoległym.

Schemat postępowania przy realizacji symulacji zdarzeniowo - ciągłej na bazie języka LOGLAN można zilustrować następującym przykładem:

Dany jest abstrakcyjny system składający się z pewnego zbioru bliźniaczych procesów zdarzeniowych oraz bliźniaczych procesów ciągłych. Każdy proces może być następującego typu:

- *typ 1 : proces izolowany (nie powiązany z innymi procesami),*
- *typ 2 : proces powiązany tylko z procesami tego samego rodzaju,*
- *typ 3 : proces dowolnie powiązany z procesami różnego rodzaju.*

UWAGA: Powiązanie z innymi procesami rozumiane jest tutaj, dla uproszczenia, tylko jako możliwość wglądu (odczytu) wyróżnionych zmiennych lokalnych wybranych procesów.
Wzorce procesów są następujące:

PROCES CIĄGŁY

Proces ten można sobie wyobrazić, jako pracę pewnego abstrakcyjnego

urządzenia sterowanego przez człowieka operatora (proces-zdarzeniowy). Przyjmijmy, dla uproszczenia, że w programie symulacyjnym proces ten będzie reprezentowany przez równanie różniczkowe zwyczajne $y'=f(t,y)$ rozwiązywane na bieżąco klasyczną metodą Rungego-Kutty IV rzędu ze stałym krokiem całkowania. W równaniu na y' będą obecne ponadto pewne współczynniki (parametry), których wartości wyznaczane będą w innych procesach.

PROCES ZDARZENIOWY

Proces ten może być wyobrażony, jako człowiek-operator pewnego urządzenia (procesu ciągłego). Przyjmijmy, dla uproszczenia, że proces ten nie będzie generował żadnych procesów potomnych oraz, że czas między zdarzeniami będzie określany w oparciu o odczytane wartości funkcji lokalnych związanych z nim procesów ciągłych (maksymalnie trzech).

Poniżej podano listing programu ZDACIAG realizującego przedstawione zagadnienie:

```
PROGRAM ZDACIAG;
(* Przykład symulacji mieszanej  -  zdarzeniowo-ciagłej *)

(*$L-*)
(* ........ tresc klasy SIMULATION *)
(*$L+*)

BEGIN
 PREF SIMULATION BLOCK;

 VAR
   TPOCZ  : REAL, (* poczatek podprzedziału przebiegu ciagłego *)
   TKONC  : REAL, (* koniec   podprzedziału przebiegu ciagłego *)
    (* TKONC-TPOCZ  wyznacza odcinek czasu miedzyzdarzeniowego *)
   LPC    : ARRAYOF PROCES_CIAGLY,
    (* lista procesow ciagłych - reprezentowana tablicowo      *)
   LPZ    : ARRAYOF PROCES_ZDARZENIOWY,
    (* lista procesow zdarzeniowych - reprezentowana tablicowo *)
   APC_   : APC,  (* referencja do wspołprogramu APC           *)
   PLIK   : FILE, (* referencja do pliku rejestrujacego wyniki *)
   I      : INTEGER;

 CONST
   L_PC=4,  (* liczba procesow ciagłych              *)
   L_PZ=6,  (* liczba procesow zdarzeniowych         *)
   H=1.0,   (* krok całkowania w procesach ciaglych *)
   CZAS_SYMULACJI=30.0;


 UNIT PROCES_CIAGLY: COROUTINE (P1,P2,P3,NUMER: INTEGER);
   (* Przykładowy wzorzec procesu ciagłego *)

   (* Parametry formalne P1,P2,P3 oznaczają numery współpracu-
      jacych procesow ciagłych lub zdarzeniowych; wartosc 0 przy
      generacji procesu oznacza brak wspołpracy, wartosc wieksza
      od 100 oznacza proces zdarzeniowy;
      parametr NUMER jest identyfikatorem procesu              *)

   VAR
    T           : REAL, (* czas, zmienna niezalezna przebiegu  *)
    Y           : REAL, (* wartosc biezaca całki y'=f(t,y)     *)
    K0,K1,K2,K3: REAL, (* wspołczynniki Rungego_Kutty IV. rz. *)
```

```
  Q1,Q2,Q3    : REAL; (* zm. pom. zalezne od innych procesow *)
  LOKALNA     : REAL; (* lokalna funkcja procesu uzalezniona
                      od przebiegu zmiennych Q1,Q2,Q3      *)

  UNIT F: FUNCTION (T,Y: REAL) : REAL;
  (* Przykładowa funkcja podcałkowa *)
  BEGIN
    RESULT:=SIN(Y)*EXP(-T)+COS(2*T);
  END F;

  BEGIN (* PROCES_CIAGLY *)

    (* okreslenie warunku poczatkowego *)
    Y:=0.5; LOKALNA:=1.0;
    RETURN;
    (*... główny cykl obliczeniowy ... *)
    DO
      T:=TPOCZ;
      DO
        K0:=F(T    ,Y       );
        K1:=F(T+H/2,Y+H*K0/2);
        K2:=F(T+H/2,Y+H*K1/2);
        K3:=F(T+H  ,Y+H*K2  );
        Y:=Y+(K0+2*K1+2*K2+K3)*H/6;
        T:=T+H;
        IF T>TKONC THEN EXIT
         (* koniec obliczeń dla podprzedziału [TPOCZ,TKONC] *)
        FI;
      OD;
      (* obliczenie wartosci funkcji lokalnej procesu,
         ktorej przebieg uzalezniony jest od innych procesow *)
      Q1:=POWIAZANIE(P1);
      Q2:=POWIAZANIE(P2);
      Q3:=POWIAZANIE(P3);
      LOKALNA:=Y+Q1+Q2+Q3;
      WRITELN(PLIK,TKONC:6:2,"  Proces ciagly numer: ",NUMER:4,
                   "   F.lokalna = ",LOKALNA:12:3);
      (* zwrot sterowania do wspołprogramu
         aktywacji procesow ciagłych       *)
      DETACH;
    OD;
  END PROCES_CIAGLY;


UNIT POWIAZANIE: FUNCTION (P: INTEGER) : REAL;
(* Przykład funkcji dokonujacej powiazania procesow *)
  BEGIN
    IF P=0 THEN   (* brak powiazania procesow *)
       RESULT:=0; RETURN FI;
    IF P>100 THEN (* powiazanie z procesem zdarzeniowym *)
       RESULT:=LPZ(P MOD 100).LOKALNA; RETURN FI;
    IF P>0 AND P<100 (* powiazanie z procesem ciagłym *)
       THEN RESULT:=LPC(P).LOKALNA; RETURN FI;
    RESULT:=1.0; (* pozostałe przypadki *)
  END POWIAZANIE;


UNIT APC: COROUTINE;
(* Wspołprogram aktywacji procesow ciagłych *)
  VAR I: INTEGER;
  BEGIN
    RETURN;
    DO
      TKONC:=TIME; (* patrzenie "w tył" ........     *)
      (* procesy ciagłe "doganiają" czas bieżący    *)
      FOR I:=1 TO L_PC DO ATTACH(LPC(I)) OD;
      TPOCZ:=TIME;
      (* zwrot sterowania do procesu zdarzeniowego *)
```

```
      DETACH;
    OD;
  END APC;

UNIT PROCES_ZDARZENIOWY:
     SIMPROCESS CLASS (P1,P2,P3,NUMER: INTEGER);
(* Przykładowy wzorzec procesu zdarzeniowego *)

(* Parametry formalne P1,P2,P3 oraz NUMER rozumiane sa
   tutaj analogicznie jak w PROCESIE_CIAGLYM          *)

  VAR
     Q1,Q2,Q3 : REAL, (* zmienne pomocnicze zalezne od
                         innych procesow              *)
     LOKALNA  : REAL; (* funkcja lokalna procesu
                         zalezna od Q1,Q2,Q3          *)

  BEGIN (* PROCES_ZDARZENIOWY *)
    DO
      IF TPOCZ<TIME THEN
        ATTACH(APC_); (* aktywacja procesow ciagłych *)
      FI;
      Q1:=POWIAZANIE(P1);
      Q2:=POWIAZANIE(P2);
      Q3:=POWIAZANIE(P3);
      LOKALNA:=TIME+Q1+Q2+Q3;
      WRITELN(PLIK,TIME:6:2,"  Proces zdarzeniowy : ",NUMER:4,
                  "   F.lokalna = ",LOKALNA:12:3);
      IF LOKALNA>Q1+Q2-Q3
        THEN CALL HOLD(15*RANDOM)
        ELSE CALL HOLD(25*RANDOM)
      FI;
    OD;
  END PROCES_ZDARZENIOWY;


BEGIN (* -------- Program główny ------------ *)

  (* Otwarcie pliku do rejestracji wynikow *)
  OPEN(PLIK,TEXT,UNPACK("WYNIKI.DAT"));
  CALL REWRITE(PLIK);
  (* Generowanie tablic - list procesow     *)
  ARRAY LPC DIM(1:L_PC);
  ARRAY LPZ DIM(1:L_PZ);
  (* Generowanie procesow i okreslenie wzajemnych powiazan *)
  APC_:=NEW APC;
  LPC(1):=NEW PROCES_CIAGLY       (  0, 101,   2,    1);
  LPC(2):=NEW PROCES_CIAGLY       (102, 103, 106,    2);
  LPC(3):=NEW PROCES_CIAGLY       (  0,   0,   0,    3);
  LPC(4):=NEW PROCES_CIAGLY       (  1,   2, 105,    4);
  LPZ(1):=NEW PROCES_ZDARZENIOWY (  1,   2,   3,  101);
  LPZ(2):=NEW PROCES_ZDARZENIOWY (  0,   0,   0,  102);
  LPZ(3):=NEW PROCES_ZDARZENIOWY (101, 102,   4,  103);
  LPZ(4):=NEW PROCES_ZDARZENIOWY (106,   0,   4,  104);
  LPZ(5):=NEW PROCES_ZDARZENIOWY (103,   3,   1,  105);
  LPZ(6):=NEW PROCES_ZDARZENIOWY (  3, 105, 104,  106);
  (* Wstepne zaplanowanie zdarzen *)
  FOR I:=1 TO L_PZ DO CALL SCHEDULE(LPZ(I),2) OD;
  TPOCZ:=TIME;
  WRITELN(PLIK,TIME:6:2,"  ... POCZATEK SYMULACJI ...");
  CALL HOLD(CZAS_SYMULACJI);
  WRITELN(PLIK,TIME:6:2,"  ... KONIEC SYMULACJI ... ");
  KILL(PLIK);  (* ... *)

END
END ZDACIAG.
```

Program ZDACIAG składa się z czterech podstawowych modułów: PROCES_CIAGLY, PROCES_ZDARZENIOWY, APC oraz POWIAZANIE.

PROCES CIAGLY jest współprogramem, w którym zaraz po wygenerowaniu określany jest warunek początkowy dla zmiennych stanu. Proces ten zorganizowany jest jako pętla nieskończona, wewnątrz której znajduje się pętla iteracyjna dla przedziału czasu [TPOCZ, TKONC]. Proces ten uaktywniany jest ze współprogramu APC, do którego - po obliczeniu wartości funkcji lokalnej - zwracane jest sterowanie.

PROCES ZDARZENIOWY pracuje w konwencjonalny dla tego typu procesów sposób. Istotną różnicę stanowi pierwsza instrukcja pętli nieskończonej, jaką jest wywołanie współprogramu aktywacji procesów ciągłych (APC). Instrukcja warunkowa IF TPOCZ<TIME... zapobiega niepożądanej reaktywacji procesów ciągłych w przypadku zdarzeń jednoczesnych.

APC jest współprogramem aktywacji procesów ciągłych na podstawie listy procesów ciągłych LPC. W programie istnieją dwie zmienne odnotowujące ten sam czas symulacyjny: zmienna TIME (w klasie SIMULATION) oraz zmienna T (w procesie ciągłym). Współprogram APC określa odcinki czasu międzyzdarzeniowego dla zmiennej T. Nowa wartość zmiennej TPOCZ (po przypisaniu TPOCZ:=TIME;) służy przy okazji, jako zabezpieczenie przed niepożądanym ponownym wywołaniem współprogramu APC przez kolejny proces zdarzeniowy o tym samym zawiadomieniu co poprzedni (przypadek zdarzenia jednoczesnego).

POWIAZANIE jest funkcją realizującą uproszczony schemat powiązania procesów poprzez wzajemny wgląd do ich własnych funkcji lokalnych.

Działanie części głównej programu polega na wygenerowaniu instancji wszystkich procesów, wstępnego zaplanowania zdarzeń (które czasem jest potrzebne dla "wybiegu" procesów ciągłych, celem uzyskania przebiegów "ustalonych") oraz przekazania sterowania do procesów na czas trwania symulacji. Wartości liczbowe parametrów aktualnych w instrukcjach generacji procesów mogą być interpretowane jako elementy tablicy powiązań procesów. Przykładowo: proces zadrzeniowy nr 5 (105) powiązany jest z procesem zdarzeniowym nr 3 (103) oraz z procesami ciągłymi nr 3 i nr 1.

Poniżej podano zawartość pliku WYNIKI.DAT utworzonego i zapisanego podczas wykonywania przebiegu symulacyjnego:

```
 0.00  ... POCZATEK SYMULACJI ...
 2.00  Proces ciagly numer:    1   F.lokalna =        2.145
 2.00  Proces ciagly numer:    2   F.lokalna =        1.145
 2.00  Proces ciagly numer:    3   F.lokalna =        1.145
 2.00  Proces ciagly numer:    4   F.lokalna =        4.434
 2.00  Proces zdarzeniowy :  103   F.lokalna =        6.434
 2.00  Proces zdarzeniowy :  101   F.lokalna =        6.434
 2.00  Proces zdarzeniowy :  106   F.lokalna =        3.145
 2.00  Proces zdarzeniowy :  102   F.lokalna =        2.000
 2.00  Proces zdarzeniowy :  105   F.lokalna =        1.723
 2.00  Proces zdarzeniowy :  104   F.lokalna =        3.579
 4.76  Proces ciagly numer:    1   F.lokalna =        9.947
 4.76  Proces ciagly numer:    2   F.lokalna =       12.947
 4.76  Proces ciagly numer:    3   F.lokalna =        1.368
 4.76  Proces ciagly numer:    4   F.lokalna =       34.985
```

4.76	Proces zdarzeniowy :	103	F.lokalna =	48.175
6.56	Proces ciagly numer:	1	F.lokalna =	21.206
6.56	Proces ciagly numer:	2	F.lokalna =	55.144
6.56	Proces ciagly numer:	3	F.lokalna =	1.825
6.56	Proces ciagly numer:	4	F.lokalna =	89.899
6.56	Proces zdarzeniowy :	105	F.lokalna =	77.764
10.44	Proces ciagly numer:	1	F.lokalna =	63.529
10.44	Proces ciagly numer:	2	F.lokalna =	55.270
10.44	Proces ciagly numer:	3	F.lokalna =	1.950
10.44	Proces ciagly numer:	4	F.lokalna =	198.512
10.44	Proces zdarzeniowy :	103	F.lokalna =	217.386
10.83	Proces ciagly numer:	1	F.lokalna =	62.812
10.83	Proces ciagly numer:	2	F.lokalna =	223.640
10.83	Proces ciagly numer:	3	F.lokalna =	1.109
10.83	Proces ciagly numer:	4	F.lokalna =	365.324
10.83	Proces zdarzeniowy :	101	F.lokalna =	298.390
11.38	Proces ciagly numer:	1	F.lokalna =	522.472
11.38	Proces ciagly numer:	2	F.lokalna =	222.974
11.38	Proces ciagly numer:	3	F.lokalna =	0.443
11.38	Proces ciagly numer:	4	F.lokalna =	823.654
11.38	Proces zdarzeniowy :	102	F.lokalna =	11.385
12.62	Proces ciagly numer:	1	F.lokalna =	522.663
12.62	Proces ciagly numer:	2	F.lokalna =	233.215
12.62	Proces ciagly numer:	3	F.lokalna =	1.299
12.62	Proces ciagly numer:	4	F.lokalna =	834.940
12.62	Proces zdarzeniowy :	101	F.lokalna =	769.797
13.87	Proces ciagly numer:	1	F.lokalna =	1003.841
13.87	Proces ciagly numer:	2	F.lokalna =	232.746
13.87	Proces ciagly numer:	3	F.lokalna =	0.830
13.87	Proces ciagly numer:	4	F.lokalna =	1315.180
13.87	Proces zdarzeniowy :	104	F.lokalna =	1332.194
15.62	Proces ciagly numer:	1	F.lokalna =	1003.275
15.62	Proces ciagly numer:	2	F.lokalna =	232.648
15.62	Proces ciagly numer:	3	F.lokalna =	0.733
15.62	Proces ciagly numer:	4	F.lokalna =	1314.419
15.62	Proces zdarzeniowy :	101	F.lokalna =	1252.272
15.79	Proces ciagly numer:	1	F.lokalna =	1486.233
15.79	Proces ciagly numer:	2	F.lokalna =	233.228
15.79	Proces ciagly numer:	3	F.lokalna =	1.312
15.79	Proces ciagly numer:	4	F.lokalna =	1798.536
15.79	Proces zdarzeniowy :	103	F.lokalna =	3077.983
16.90	Proces ciagly numer:	1	F.lokalna =	1486.300
16.90	Proces ciagly numer:	2	F.lokalna =	3093.313
16.90	Proces ciagly numer:	3	F.lokalna =	0.800
16.90	Proces ciagly numer:	4	F.lokalna =	4658.178
16.90	Proces zdarzeniowy :	106	F.lokalna =	1427.663
17.49	Proces ciagly numer:	1	F.lokalna =	4345.566
17.49	Proces ciagly numer:	2	F.lokalna =	4517.011
17.49	Proces ciagly numer:	3	F.lokalna =	-0.020
17.49	Proces ciagly numer:	4	F.lokalna =	8940.321
17.49	Proces zdarzeniowy :	103	F.lokalna =	10221.464
19.69	Proces ciagly numer:	1	F.lokalna =	5769.400
19.69	Proces ciagly numer:	2	F.lokalna =	11660.628
19.69	Proces ciagly numer:	3	F.lokalna =	0.117
19.69	Proces ciagly numer:	4	F.lokalna =	17507.909
19.69	Proces zdarzeniowy :	102	F.lokalna =	19.691
19.81	Proces ciagly numer:	1	F.lokalna =	12912.259
19.81	Proces ciagly numer:	2	F.lokalna =	11668.175
19.81	Proces ciagly numer:	3	F.lokalna =	-0.642
19.81	Proces ciagly numer:	4	F.lokalna =	24657.555
19.81	Proces zdarzeniowy :	105	F.lokalna =	23152.887
22.95	Proces ciagly numer:	1	F.lokalna =	12919.095
22.95	Proces ciagly numer:	2	F.lokalna =	11667.465
22.95	Proces ciagly numer:	3	F.lokalna =	-1.352
22.95	Proces ciagly numer:	4	F.lokalna =	47738.094
22.95	Proces zdarzeniowy :	106	F.lokalna =	24506.683

```
27.41  Proces ciagly numer:    1   F.lokalna =    12917.613
27.41  Proces ciagly numer:    2   F.lokalna =    34745.714
27.41  Proces ciagly numer:    3   F.lokalna =       -2.124
27.41  Proces ciagly numer:    4   F.lokalna =    70814.089
27.41  Proces zdarzeniowy :  104   F.lokalna =    95348.187
28.31  Proces ciagly numer:    1   F.lokalna =    35996.498
28.31  Proces ciagly numer:    2   F.lokalna =    34746.350
28.31  Proces ciagly numer:    3   F.lokalna =       -1.488
28.31  Proces ciagly numer:    4   F.lokalna =    93894.247
28.31  Proces zdarzeniowy :  101   F.lokalna =    70769.666
28.58  Proces ciagly numer:    1   F.lokalna =   105514.940
28.58  Proces ciagly numer:    2   F.lokalna =    34746.762
28.58  Proces ciagly numer:    3   F.lokalna =       -1.076
28.58  Proces ciagly numer:    4   F.lokalna =   163413.514
28.58  Proces zdarzeniowy :  106   F.lokalna =   118528.579
30.00  ... KONIEC SYMULACJI ...
```

Zrealizowany w programie ZDACIAG schemat przeprowadzania symulacji mieszanej w języku LOGLAN umożliwia konstruowanie różnego typu procesów zdarzeniowych oraz ciągłych współpracujących ze sobą. Schemat ten może być wykorzystany do modelowania systemów o dowolnym stopniu złożoności.

Oprócz sekwencji całkowania numerycznego do programu symulacyjnego mogą być w łatwy sposób dołączone inne funkcje języka CSMP. Daje to możliwość skonstruowania w oparciu o LOGLAN języka symulacyjnego przeznaczonego do symulacji zdarzeniowo-ciągłej. Stworzenie takiego języka mogłoby być dokonane poprzez przeróbkę klasy SIMULATION. Np. uaktywniania procesów ciągłych mogłaby dokonywać procedura CHOICEPROCESS (*zagadnienie to wymaga oddzielnego opracowania*).

Na zakończenie niniejszego podrozdziału należy dodać, że stosowana w procesie ciągłym stałokrokowa metoda całkowania numerycznego wprowadza pewną niedokładność, w przypadkach gdy czas między -zdarzeniowy nie jest krotnością H lub gdy jest on mniejszy od H. Problem ten można łatwo rozwiązać poprzez zastosowanie metody zmiennokrokowej, w której na początku cyklu iteracyjnego krok H byłby równy TKONC-TPOCZ, a potem byłby odpowiednio zmniejszany (w zależności od przyjętej dokładności obliczeń).

UWAGA UZUPEŁNIAJĄCA: W przypadku stosowania w symulacji technik animacji obrazu, z uwagi na niejednakowe absorbowanie czasu procesora przez akcje procesów aktywnych - uaktualnianie obrazu nie może odbywać się poprzez stałokrokowy proces zdarzeniowy odnotowywany na osi czasu symulacyjnego, tylko w oparciu o zegar maszynowy - tak aby zmiany obrazu na ekranie przebiegały w sposób płynny i nie deformowały rzeczywistego obrazu symulacji.

4. Uwagi końcowe

Przedstawione w artykule zagadnienia sygnalizują jedynie problematykę rozszerzania loglanowskich zasobów symulacyjnych. Bardzo interesującym zagadnieniem jest prowadzenie symulacji na wielu komputerach połączonych w sieć. Podstawowym problemem w tego typu symulacji byłaby kwestia synchronizacji oraz komunikacji procesów pogrupowanych na różnych komputerach. Uwzględnienie symulacyjnej osi czasu w programowaniu współbieżnym jest zagadnieniem niekonwencjonalnym, którego omówienie

wykracza poza ramy niniejszego artykułu.

LITERATURA

[1] KONIECZNY R.: Zasoby symulacyjne języka LOGLAN - *(niniejszy zeszyt)*

[2] KONIECZNY R.: Przykłady rozwiązania problemów symulacyjnych w języku LOGLAN - *(niniejszy zeszyt)*.

[3] KOSTON H.: Generatory rozkładów pseudolosowych w LOGLANie - praca magisterska, Instytut Informatyki Uniwersytetu Warszawskiego, Warszawa 1987

[4] ZIĘTARA G.: Język symulacji układów ciągłych - IBM System 360 - Continuous System Modeling Program (CSMP). - Wyd. ZETO ZOWAR, Warszawa 1973.

PROBLEMS OF EXTENDING LOGLAN SIMULATION RESOURCES

Summary

The present article shows selected aspects which refers to extending LOGLAN simulation resources (not included in the SIMULATION class).

These aspects are: pseudo-random number generators and problems of event-continuous simulation.

Functionaly additional simulation resources can be added to LOGLAN as separate packs of new source procedures or by developing the SIMULATION class .

The problems presented in the paper signal the problem of extending the LOGLAN simulation resources.

ERWEITERUNGSPROBLEM VON LOGLAN-SIMULATIONSVORRÄTEN

Zusammenfassung

Vorliegender Aufsatz stellt ansgewählte Aspekte dar, die die Erweiterung der Simulationsvorräte der Sprache LOGLAN betreffen und die in der klasse SIMULATION nicht berücksichtig worden sind.

Zu diesen Aspekten gehören: Pseudozufallszahlengeneratoren sowie Probleme der ereignis-stetigen Simulation.

Funktionsmäßig zasätzliche Simulationsvorräte können an LOGLAN angeschlosen werden. Zwei Wege sind hier möglich: als unabhängige Pakete neuer Quellprozeduren oder durch Ausbau der Klasse SIMULATION.

Die im Aufsatz vorgestellten Fragen signalisieren das Problem der Erweiterung von LOGLAN-Simulationsvorräten.

ВОПРОСЫ ПАСШИРЕНИЯ ЛОГЛАНОВСКИХ СИМУЛЬЦИОННЫХ РЕСУРСОВ

Резюме

В данной статье представляются избранные аспекты, касающиеся расширения симуляционных ресурсов языка LOGLAN (не учтенных в классе SIMULATION).

Этими аспектами являются: генераторы псевдослучайных чисел и вопросы гибридной имитации. В качестве функциональных дополнительные симуляционные ресурсы могут быть присоединены к LOGLANу как независимые пакеты новых основных процедур или-же путем расширения класса SIMULATION.

Представленные в статье проблемы ставят вопросы расширения логлановских симуляционных ресурсов.