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

Roman KONIECZNY

JĘZYK PROGRAMOWANIA LOGLAN JAKO NARZĘDZIE OPISU MODELU SYSTEMU TRANSPORTOWEGO

Streszczenie. Artykuł niniejszy stanowi próbę odpowiedzi na pytanie: czy język programowania (a konkretniej: język LOGLAN) może być wykorzystany - nie tylko do notacji algorytmów procedur, czy większych projektów software'owych - ale również do opisu systemów. Uznając konwencję metaprogramistyczną jako najbardziej użyteczną przy tworzeniu projektów oprogramowania złożonych systemów zaproponowano konwencję notacyjną **P-LOGLAN** (bazującą na idei Pidgin ALGOLu oraz na Raporcie LOGLANu-82).

W artykule stwierdzono, że konwencja P-LOGLAN bardzo dobrze nadaje się do opisu formalnego złożonych systemów transportowych.

1. Uwagi wstępne

Artykuł niniejszy stanowi próbę odpowiedzi na pytanie: czy język programowania (a konkretniej: język LOGLAN) może być wykorzystany, nie tylko do notacji algorytmów procedur, czy większych projektów software'owych, ale również do opisu systemów. Model systemu transportowego (na danym szczeblu abstrakcji) stanowić tutaj może egzemplifikację tego interesującego problemu.

Uznając konwencję metaprogramistyczną, jako najbardziej użyteczną przy tworzeniu projektów oprogramowania złożonych systemów [4] - w pracy [5] zaproponowano konwencję notacyjną **P-LOGLAN** (bazującą na idei Pidgin ALGOLu [1] oraz na Raporcie LOGLANu-82 [6]).

2. Propozycja konwencji P-LOGLAN

Podstawowe postulaty tej konwencji są następujące [5]:

- konwencja P-LOGLAN stanowi nieformalne powiązanie różnych konwencji zapisu, stosowanych w językach wysokiego poziomu; jako pierwowzór tej konwencji może być uznany Pidgin ALGOL [1];
- konwencja **P-LOGLAN** bazuje na słowach kluczowych oraz wszystkich innych ustaleniach zawartych w Raporcie LOGLANu-82 [6] ;

- konwencja przeznaczona jest do tworzenia projektów oprogramowania, opisu systemów, a także różnego rodzaju (wymagających formalnego ujęcia) publikacji naukowych;
- konwencję notacyjną P-LOGLAN można również traktować jako zmodyfikowany ALGOL publikacyjny; może być także rozumiana jako język wysokiego poziomu, który może być zaimplementowany na dowolny komputer o odpowiednich zasobach;
- instrukcja w P-LOGLANie jest tekstem będącym mieszaniną słów kluczowych LOGLANu oraz <treści dodatkowej> podanej w konwencji: werbalistycznej, symbolicznej lub metaprogramistycznej [4] - względnie wszystkich razem (niedozwolona jest tylko konwencja graficzna);
- przy projektach oprogramowania we wstawkach <treści dodatkowej> obowiązuje stosowanie zapisów na bazie kodu ASCII (stosowanie innych symboli, jak i elementów grafiki dozwolone jest tylko w aneksach - opatrzonych odpowiednimi odsyłaczami);
- konwencja nie jest zamknięta i może być rozbudowywana o dodatkowe elementy.

Konwencja notacyjna P-LOGLAN może być szczególnie użyteczna wszędzie tam, gdzie mnogość języków, różnorodność komputerów oraz rozproszenie zespołów ludzkich stwarza sytuację wymuszającą konieczność ujednolicenia zapisów projektów oprogramowania.

3. Przykłady zapisu w konwencji P-LOGLAN

Poniżej podano przykład projektu programu zapisanego w konwencji P-LOGLAN:

```
PROGRAM  ODBIERAKI_PRĄDU;
(* PROBLEMATYKA:   Symulacja współpracy dynamicznej N odbieraków
                   prądu z siecią trakcyjną.

                   Zagadnienie istotnie ważne, gdy prędkość pociągu
                   elektrycznego przekracza 140 km/h.

   WEJŚCIE:    Układ (od 30 do 500) równań różniczkowo-różnicowych
               2-go rzędu, niestacjonarnych, nieliniowych, niejedno-
               rodnych (szczegółowo opisanych w publikacji:
               Roman KONIECZNY, Stanisław KRAWIEC "Zagadnienie oprog-
               ramowania modelu ST3+OP2 współpracy kilku odbieraków
               prądu z siecią trakcyjną" - Zeszyty Naukowe Politech-
               niki Śląskiej, Transport nr 5 / 1986 )

   WYJŚCIE:    Trajektorie punktu styku oraz przebiegi sił stykowych
               dla wyszczególnionych odbieraków; dodatkowo: po każdym
               przebiegu symulacyjnym obliczany jest współczynnik
               jakości współpracy do zestawienia zbiorczego.

   METODA:     Całkowanie numeryczne metodą Milne'a 5-go rzędu.    *)
```

```
LIBRARY ITEM INTO USERLIB;
EXTERNALS
   UNIT  MILNES:  PROCEDURE FROM USERLIB;
      COMPILE

UNIT  DANE_WE: PROCEDURE;  (* wprowadzenie danych *)
(* WEJŚCIE: plik tekstowy z danymi typu REAL,
   WYJŚCIE: tablica globalna DANE_WEJ            *)
BEGIN (* main DANE_WE *)
  wypełnienie listy LSO - stałych ograniczeń modelu;
  pytanie o nazwę pliku z danymi;
  IF plik nie istnieje THEN komunikat; pytanie o właściwą nazwę
                                  lub KONIEC PRACY        FI;
  wprowadzenie danych do tablicy DANE_WEJ;
  porównanie zawartości tablicy z ograniczeniami na liście LSO;
  IF różnice THEN komunikat, żądanie interakcyjnego poprawienia
                             błędnych danych lub KONIEC PRACY  FI;
  ......
END DANE_WE;

UNIT  INITIALS: PROCEDURE; (* inicjacja stałych, wartości pocz. *)
  (* WEJŚCIE: Tablica DANE_WEJ,
     WYJŚCIE: WEKTOR_ZS (zmienne stanu)      *)
BEGIN  (* main INITIALS *)
  obliczenie wartości początkowych na integratorach;
  obliczenie składowych aerodynamicznych
  oraz innych parametrów liczonych jednokrotnie;
END INITIALS;

UNIT  REJESTRACJA: PROCEDURE;  (* odnotowywanie wyników *)
  (* WEJŚCIE: numery wytypowanych odbieraków prądu,
              numery wytypowanych segmentów sieci trakcyjnej,
              WEKTOR_ZS,
     WYJŚCIE: tablice TRAJEKTORIE i SIŁY_STYKOWE             *)
BEGIN (* main REJESTRACJA *)
  ... bieżące zapisy do tablic TRAJEKTORIE i SIŁY_STYKOWE;
END REJESTRACJA;

UNIT  DYNAMIC:  PROCEDURE;  (* część dynamiczna modelu *)
  (*  WEJŚCIE: tablica DANE_WEJ, CZAS_SYMULACJI, PRĘDKOŚĆ_JAZDY,
      WYJŚCIE: WEKTOR_ZS                                     *)
  BEGIN (* main DYNAMIC *)
    ... CALL MILNES(..);
    ... CALL REJESTRACJA(..);
```

```
    .....
  END DYNAMIC;

VAR  (* zmienne globalne *)
  DANE_WEJ, WEKTOR_ZS, TRAJEKTORIE, SIŁY_STYKOWE, OK: ARRAYOF...
  CZAS_SYMULACJI, PREDKOŚĆ_JAZDY, KRYTERIUM_JAKOŚCI...

BEGIN (*   main  ODBIERAKI_PRĄDU   *)
  pytanie o prędkość jazdy i czas symulacji;
  pytanie o kryterium oceny jakości...(lista wymogów OK);
  ... CALL DANE_WE;  ...
  DO
    ... CALL INITIALS; ...
    WHILE czas <= CZAS_SYMULACJI
      DO
        ... CALL DYNAMIC;  ...
      OD;
    sekw. oceny jakości na podst. tab. TRAJEKTORIE i SIŁY_STYKOWE;
    IF KRYTERIUM_JAKOŚCI zgodne z wymogami OK
       THEN EXIT  ELSE REPEAT                FI;
  OD;
  końcowe zestawienie zbiorcze uzyskanych wyników; ...
END  ODBIERAKI_PRĄDU.
```

Powyższy przykład specyfikacji niezupełnej (zarysu programu) po dalszym rozwinięciu i uszczegółowieniu oraz dokładniejszym opisaniu powiązań modułów - stanowić już może podstawę do napisania rzeczywistego programu (praktycznie w dowolnym języku).

4. Zagadnienie rozszerzenia konwencji P-LOGLAN

Stosowanie konwencji P-LOGLAN ułatwia precyzowanie koncepcji oraz usuwanie ewentualnych błędów w założeniach - już na etapie opracowywania projektu programu. Dobrze zaprojektowany program jest łatwiejszy w uruchomieniu, a koszty jego wytworzenia są odpowiednio niższe.

Zaproponowana konwencja notacyjna P-LOGLAN, w chwili obecnej, gdy realnie rysuje się możliwość korzystania ze sprzętu wielokomputerowego o dużej mocy obliczeniowej oraz ze współpracujących z nim urządzeń dodatkowych, np. kamera TV, magnetowid, kanał przemysłowy, łączność satelitarna, użycie robotów etc. - wymaga jednak pewnej rozbudowy, tak aby P-LOGLAN mógł stać się czymś szerszym, np. LNSD-LOGLANem (Language for Notation of Software Designs).

Poniżej podano szkic niektórych postulatów-propozycji, rozszerzających konwencję P-LOGLAN:

1) Każdy moduł (UNIT) posiada następujaca strukturę:

```
UNIT <nazwa> :  typ modulu (lista parametrów formalnych);
  DECLARATIONS  ( część deklaracyjna )
     CONST ...
     VAR ...
     TYPE ...
     UNIT ...
  ENVIRONMENT  ( powiązanie z otoczeniem )
     IMPORT ...
     EXPORT ...
     DEVICES
        INPUT ...
        OUTPUT ...
     FLAG
  OPERATIONS   ( część operacyjna )
     INITIALS ...   ( instrukcje początkowego wykonania )
     DYNAMICS ...   ( instrukcje podstawowego wykonania )
END_OF_UNIT <nazwa>;
```

2) Moduły mogą być następującego typu:
LIBRARY - moduł będący zbiorem modułów,
BLOCK, PROCEDURE, FUNCTION, CLASS, COROUTINE, PROCESS - znaczenie, jak w Raporcie LOGLANu-82.

3) Podstawowe typy zmiennych :
REAL, INTEGER, BOOLEAN, CHARACTER, TEXT, BYTE, WORD[N-bits], WORD[N-bytes], ADDRESS, SET, FILE, SEMAPHORE, ARRAY_OF, ...

4) Możliwość tworzenia własnych typow zmiennych, typy wyliczeniowe, kontrola zakresu zmiennej , etc.

5) Komunikacja z otoczeniem:

a) IMPORT - każdy moduł może "zaimportować" inny moduł (tj. włączyć go do swego ciała) lub zmienną zewnetrzną, np.

```
IMPORT:  MODULE ODBIERAK FROM LIBRARY KOLEJOWA;
         VARIABLE X FROM MODULE INNY;
```

b) EXPORT - każdy moduł może udostępnić swoje zasoby innym modułom;

c) DEVICES - wyspecyfikowane są urządzenia, z którymi moduł współpracuje, np.

```
DEVICES  INPUT:  CAMERA_Tv, DOCUMENT READER, MOUSE, ...
         OUTPUT: VIDEO, LASER PRINTER, FLOPPY DISK,...
```

d) FLAG - każdy moduł stale udostępnia innym modułom wgląd do swojej "flagi", tj. wskaźnika stanu (SUSPENDED, ACTIVE, LOCKED, AWAITING, TERMINATED).

6) Na jednym komputerze może "rezydować" jeden lub więcej modułów.
Jeden moduł może także równocześnie (równolegle) pracować na wielu komputerach.

7) Lista instrukcji LNSD-LOGLANu może być sukcesywnie wzbogacana o dodatkowe możliwości, np. SAVE SCREEN, LOAD IMAGE FROM VIDEO, INCLUDE UNIT... FROM... , SOUND, VOICE, COLOUR, IMAGE, ...

8) Istnieje możliwość definiowania struktury (hierarchii) modułów oraz interface'ów międzymodułowych.

9) Przyjmuje się zasadę, że moduły mogą być niezależnie kompilowane.

Przedstawione powyżej postulaty mają, oczywiście, dyskusyjny charakter, a ich szersze omówienie i uzasadnienie wykracza poza ramy tematyczne niniejszego artykułu. Niemniej problematyka ta zasługuje na podjęcie odrębnych badań w tym kierunku - spolaryzowanych na specyfikę modelowania systemów transportowych.

5. Opis modelu systemu transportowego w konwencji P-LOGLAN

Konwencja P-LOGLAN, oprócz podstawowego jej zastosowania - do notacji projektów software'owych, może być również wykorzystana do metaprogramistycznego opisu różnego rodzaju systemów (a dokładniej - modeli tych systemów). Wykorzystanie P-LOGLANu do opisu systemu transportowego (tj. modelu tego systemu na pewnym szczeblu abstrakcji) oparte zostało na pięciu podstawowych założeniach:

1) Jako definicję systemu przyjmuje się (według "Elementy teorii systemów i cybernetyki" - S.Mynarski, PWN Warszawa 1981, s.1), że: "System jest to celowo określony zbiór elementów oraz relacji zachodzących między tymi elementami i ich własnościami".

   Dla potrzeb niniejszych rozważań, określenie <element> zostaje zastąpione określeniem <obiekt> ; określenie <własności elementu> zostaje zastąpione określeniem <atrybuty obiektu> .

2) Symbole matematyczne (typu sigma, całka, kwantyfikator itd.) zostają przetransformowane na odpowiadające im klasy, względnie procedury w LOGLANie.

3) W oparciu o definicję p.1 - opis modelu wybranego systemu transportowego bazować będzie na takich pojęciach LOGLANu, jak: CLASS, PROCESS (zastępczo SIMPROCESS), FUNCTION, PROCEDURE, COROUTINE, NEW, KILL... a także innych słowach kluczowych.

4) Przyjmuje się zasadę, że za pomocą klas opisywane będą obiekty - rozumiane jako pasywne, natomiast za pomocą procesów obiekty aktywne. (Pojęcia: obiekty aktywne i pasywne można przez analogię interpretować podobnie jak obiekty obsługujące i obsługiwane).

5) Do opisu struktur oraz hierarchii systemów może być użyty mechanizm prefiksowania.

Do egzemplifikacji opisu systemu transportowego w konwencji P-LOGLAN wybrano model ruchu pojazdów w ruchu miejskim, zdefiniowany następująco:

Model ruchu pojazdów MRP jest piątką:

$$MRP \stackrel{def}{=} \langle SK, PŁ, PO, OT, ST \rangle$$

gdzie:

SK - jest zbiorem skrzyżowań,

PŁ - jest zbiorem połączeń między skrzyżowaniami,

PO - jest zbiorem pojazdów obecnych w systemie,

OT - jest zbiorem wejść i wyjść (połączenie z otoczeniem),

ST - jest zbiorem sterowań dla SK;

każde i-te SK jest trójką: $SK_i \stackrel{def}{=} \langle PR, PP, SYG \rangle$

gdzie:

PR - jest zbiorem pasów ruchu (kolejek),

PP - jest zbiorem przejść dla pieszych,

SYG - jest zbiorem sygnalizatorów;

każdy PR na skrzyżowaniu posiada następujące atrybuty:

$$PR = \langle NR, DŁ, ID, LP \rangle$$

gdzie:

NR - numer pasa na skrzyżowaniu,

DŁ - długość pasa w [m],

ID - informacje dodatkowe (np. obecność zielonej strzałki),

LP - aktualna liczba pojazdów na pasie NR;

każde PŁ posiada następujące atrybuty:

$$PŁ = \langle NP, DŁP, IDP, LPP \rangle$$

gdzie:

NP - numer połączenia,

DŁP - długość połączenia w [m],

IDP - informacje dodatkowe (np. dozwolona prędkość, możliwość wyprzedzania, itd.),

LPP - aktualna liczba pojazdów na połączeniu;

każdy PO scharakteryzowany jest poprzez piątkę atrybutów:

$$PO = \langle NRP, TYP, PARS, REF, V \rangle$$

gdzie:

NRP - numer pojazdu,

TYP - typ pojazdu (osobowy, ciężarowy, ciężarowy z przyczepą, autobus, autobus przegubowy, tramwaj, trolejbus, motocykl, rower, pojazd uprzywilejowany w akcji, ...)

PARS - parametry silnika,

REF - refleks kierowcy (tj. szybkość reakcji na wymuszenia zewnętrzne),
V - aktualna prędkość pojazdu;

każdy obiekt należący do zbioru OT posiada następujące atrybuty:

OT = (R, NO, IDO)

gdzie:
R - rodzaj kontaktu z otoczeniem (wlot, wylot),
NO - numer ,
IDO - informacje dodatkowe (np. charakterystyka rodzajowa pojazdów dla wlotu);

zbiór sterowań ST dla poszczególnych SK zawiera dane dotyczące nastaw poszczególnych sygnalizatorów; zbiór PR określa parametry pasów ruchu oraz zawiera dane o geometrii skrzyżowania; zbiór PP zawiera informacje o ruchu pieszych na danym przejściu; zbiór SYG przechowuje wartości stanów poszczególnych sygnalizatorów.

Przemieszczanie się pojazdów w badanym systemie odbywa się według zasad wynikających z Kodeksu Drogowego (wzbogaconych o elementy kultury na jezdni).

Znając specyfikę ruchu miejskiego oraz dodatkowe szczegóły podane m.in. w pracach [2] oraz [3] - można ww. system opisać w konwencji P-LOGLANu w sposób bardziej precyzyjny, tworząc jednocześnie projekt programu symulacyjnego:

```
PROGRAM MODEL_RUCHU_MIEJSKIEGO;
  (* PROBLEMATYKA:   Badania symulacyjne ruchu pojazdów
                     w dużych aglomeracjach miejskich.
     WEJŚCIE:        Dane o topologii sieci ulic i skrzyżowań
                     z sygnalizacją świetlną.
                     Charakterystyka rodzajowa pojazdów
                     na poszczególnych wlotach do sieci.
     WYJŚCIE:        Wskaźniki jakości ruchu (średni czas przejazdu
                     pojazdu przez sieć, liczba zatrzymań, zużycie
                     paliwa, zawartość spalin w otoczeniu... )
     METODA:         Symulacja dyskretna w oparciu o system procesów
                     quasi-równoległych.                          *)
UNIT ZGŁOSZENIA_ZEWNĘTRZNE: PROCESS (NUMER_WLOTU: INTEGER);
  (* WEJŚCIE: tablica CHARAKTERYSTYKA_WLOTÓW,
     WYJŚCIE: wygenerowanie obiektu POJAZD i umieszczenie go
              na odpowiadającemu danemu wlotowi połączeniu.    *)
  VAR (* atrybuty *)
```

```
    N_WLOTU:        INTEGER, P: ARRAY_OF POJAZD, ...
    ODSTEP_CZASU:   REAL,
    N_KOLEJNY:      INTEGER, ...
  BEGIN (* main ZGŁOSZENIA_ZEWNĘTRZNE *)
    ......
    DO  (* pętla nieskończona *)
      w oparciu o dane w tablicy CHARAKTERYSTYKA_WLOTÓW
      losowanie typu pojazdu, parametrów silnika,
      prędkości początkowej, ... ;
      P(N_KOLEJNY) := NEW POJAZD (wylosowane wartości atrybutów,
                                  numer połączenia odpowiadającego
                                  wlotowi  NUMER_WLOTU);
      losowanie wartości ODSTĘP_CZASU do następnego zgłoszenia;
      WAIT(ODSTĘP_CZASU);
    OD;
  END ZGŁOSZENIE_ZEWNĘTRZNE;

UNIT  SYGNALIZATOR: PROCESS (NUMER_SYG: INTEGER,
                             STER_ZD:   BOOLEAN);
  (* WEJŚCIE: tablica CZASY_NASTAW lub STEROWANIE_ZDALNE,
     WYJŚCIE: efektywny czas trwania światła zielonego
                                          i czerwonego *)
  VAR
    T_ZIELONE, T_CZERWONE: REAL,
    N_SYG: INTEGER, STEROW_ZDALNE: BOOLEAN, ...
  BEGIN (* main SYGNALIZATOR *)
    ... STEROW_ZDALNE := STER_ZD; ...
    DO
      IF STEROW_ZDALNE THEN określenie T_ZIELONE, T_CZERWONE
                            w oparciu o dane z tablicy
                            STEROWANIE_ZDALNE;
                       ELSE ... w oparciu o dane z tab.
                            CZASY_NASTAW;        FI;
      IF światło zielone THEN
                              WAIT(T_CZERWONE)
                         ELSE
                              WAIT(T_ZIELONE)    FI;
      ......
    OD
  END SYGNALIZATOR;

UNIT PRZEJŚCIE_DLA_PIESZYCH: CLASS (NR_SKRZ, NR_PRZEJŚCIA:
                                    INTEGER);
  (* WEJŚCIE: tablica DANE_O_PRZEJŚCIACH,
     WYJŚCIE: rozmieszczenie pieszych na przejściu,
     UWAGI:   wartości atrybutów obiektu tej klasy
              uaktualnia procedura SYTUACJA_NA_PRZEJŚCIU  *)
```

```
  VAR
      LICZBA_STREF: INTEGER,
      LICZBA_PIESZYCH (* w poszczególnych strefach przejścia *):
      ARRAY_OF INTEGER, A: LICZBA_PIESZYCH, ...

  BEGIN (* main PRZEJŚCIE_DLA_PIESZYCH *)

    określenie wartości LICZBA_STREF na podstawie
    tablicy DANE_O_PRZEJŚCIACH;

    ARRAY A DIM (1 : LICZBA_STREF);

    ......

  END PRZEJŚCIE_DLA_PIESZYCH;

UNIT SYTUACJA_NA_PRZEJŚCIU: PROCEDURE (NR_SKRZ, NR_PRZEJŚCIA:
                                       INTEGER);
  (* WEJŚCIE: stan sygnalizatora związanego z przejściem...,
              tablica DANE_O_PRZEJŚCIACH,
              sytuacja na połączeniu (tj. dane o zbliżających
              się pojazdach),

     WYJŚCIE: uaktualnione wartości atrybutów obiektu
              PRZEJŚCIE_DLA_PIESZYCH                          *)

  VAR  P: PRZEJŚCIE_DLA_PIESZYCH, ...

  BEGIN (* main SYTUACJA_NA_PRZEJŚCIACH *)

    ......

    w oparciu o stan sygnalizatora związanego z danym przejściem,
    dane w tablicy DANE_O_PRZEJŚCIACH, dane o sytuacji na połą-
    czeniu - uaktualnienie atrybutów obiektu P(NR_skrz,N_przej);

    ......

  END SYTUACJA_NA_PRZEJŚCIU;

UNIT AKT_STANU_PRZEJŚĆ:  PROCESS (CYKL_AKT: INTEGER);
  (* cykliczne uaktualnianie stanu przejść *)
  VAR  CYKL: INTEGER, ...
  BEGIN (* main AKT_STANU_PRZEJŚĆ *)
    CYKL := CYKL_AKT;

    ... ...

    DO
      FOR wszystkie przejścia
        DO   CALL SYTUACJA_NA_PRZEJŚCIU(...)  OD;
      WAIT(CYKL);

    OD; ...

  END AKT_STANU_PRZEJŚĆ;


UNIT  POJAZD:  PROCESS (NRP: INTEGER, TYP, PARS...,
                        REF, V: REAL );
  (* WEJŚCIE: dane z tablicy SYTUACJA_NA_POŁĄCZENIU,
              dane z tablicy SYTUACJA_W_KOLEJKACH,
              dane o najbliższych obiektach: SYGNALIZATOR
              i PRZEJŚCIE_DLA_PIESZYCH,
```

```
      WYJŚCIE: uaktualnione wartości atrybutów obiektu POJAZD  *)

  VAR
    NUMER:        INTEGER,
    LOKALIZACJA:  INTEGER, (* na połączeniu, w kolejce,
                            w trakcie opuszczania skrzyżowania *)
    PRĘDKOŚĆ:     REAL, ...
  UNIT JAZDA_NA_POŁĄCZENIU: PROCEDURE (...);
    (* WEJŚCIE: wartości atrybutów obiektu POJAZD,
                dane z tab. SYTUACJA_NA_POŁĄCZENIU,
       WYJŚCIE: czas dojazdu do najbliższego skrzyżowania  *)
    BEGIN (* main JAZDA_NA_POŁĄCZENIU *)
      obliczenie czasu dojazdu do najbliższego skrzyżowania;
      ...
END JAZDA_NA_POŁĄCZENIU;

UNIT PRZYBYCIE_DO_KOLEJKI: PROCEDURE (...);

(* WEJŚCIE: dane z tablicy SYTUACJA_W_KOLEJKACH,
       WYJŚCIE: wybór kolejki,
                określenie dalszego kierunku jazdy  *)
    END PRZYBYCIE_DO_KOLEJKI;


  UNIT OPUSZCZANIE_SKRZYŻOWANIA: PROCEDURE (...);

    (* WEJŚCIE: dane o geometrii skrzyżowania,
                wartości atrybutów obiektu POJAZD,
                dane o sygnalizacji i zajętości skrzyżowania,
       WYJŚCIE: numer następnego połączenia, prędkość pocz.
                lub opuszczenie sieci                          *)
    END OPUSZCZENIE_SKRZYŻOWANIA;

  BEGIN (* main POJAZD *)
    nadanie wartości początkowych atrybutom;
    DO
      IF pojazd na połączeniu THEN CALL JAZDA_NA_POŁĄCZENIU(...);
                                   WAIT(CZAS_DOJAZDU_DO_KOLEJKI);
                                   CALL PRZYBYCIE_DO_KOLEJKI(...);
      FI; ... ...
      IF pojazd pierwszy w kolejce
         THEN CALL OPUSZCZANIE_SKRZYZOWANIA(...);
              WAIT(CZAS_OPUSZCZANIA);
      FI;
      ... ...
    OD;
    ... ...
  END POJAZD;
```

```
VAR (* zmienne globalne *)
    CHARAKTERYSTYKA_WLOTÓW, CHARAKTERYSTYKA_POŁACZEŃ,
    GEOMETRIA_SKRZYŻOWAŃ, CZASY_NASTAW, STEROWANIE_ZDALNE,
    DANE_O_PRZEJŚCIACH, SYTUACJA_NA_POŁĄCZENIU,
    SYTUACJA_W_KOLEJKACH : ARRAY_OF ... ,
    ZZ:                    ARRAY_OF ZGŁOSZENIE_ZEWNĘTRZNE,
    SG:                    ARRAY_OF SYGNALIZATOR,
    PP:                    ARRAY_OF PRZEJŚCIE_DLA_PIESZYCH,
    PO:                    ARRAY_OF POJAZD, ...
    ASP:              AKT_STANU_PRZEJŚĆ, LICZBA_WLOTÓW, LICZBA_POŁĄCZEŃ,
                      LICZBA_SKRZYŻOWAŃ: INTEGER, ... , ST_ZD: BOOLEAN, ...

  BEGIN (* main MODEL_RUCHU_MIEJSKIEGO *)
    wypełnienie tablic topologicznych danymi z plików wejściowych;
    FOR wszystkie wloty
      DO ZZ(Numer_wlotu):=NEW ZGŁOSZENIE_ZEWNĘTRZNE(Numer_wlotu)
      OD;
    ST_ZD:=TRUE;

    FOR wszystkie sygnalizatory
      DO SG(Numer_syg):=NEW SYGNALIZATOR(Numer_syg, ST_ZD)  OD;
    FOR wszystkie przejścia
      DO PP(Numer_przej):=NEW PRZEJŚCIE_DLA_PIESZYCH(Nr_sk,
                              Numer_przej)                  OD;

    ASP:= NEW AKT_STANU_PRZEJŚĆ(5 (* [sek] *) );
    uaktywnienie kolejno wszystkich procesów: SYGNALIZATOR,
    PRZEJŚCIE_DLA_PIESZYCH, ZGŁOSZENIE_ZEWNĘTRZNE ;

    DO
       PROCEDURA SYNCHRONIZACJI PROCESÓW;
       bieżąca rejestracja stanu systemu;
       obliczenie wskaźników jakości ruchu;
       ocena wskaźników jakości;
       procedura obliczająca nowe wartości
       elementów tablicy STEROWANIE_ZDALNE;
       IF koniec pracy systemu
          THEN EXIT  ELSE REPEAT FI;
    OD;
    analiza przebiegów...

END MODEL_RUCHU_MIEJSKIEGO.
```

5. Uwagi końcowe

Oceniając powyższy opis przykładowego systemu transportowego, dokonany w konwencji P-LOGLAN, można odnotować następujące zalety tej konwencji:

- konwencja ułatwia nie tylko opis formalny, ale także zrozumienie

działania danego systemu;
- możliwe jest dokonywanie opisów na dowolnych poziomach uszczegółowienia; poziomy uszczegółowienia mogą być różne dla różnych modułów;
- konwencja umożliwia, oprócz opisu systemów, tworzenie pseudoprogramów, które mogą być potem przenoszone na dowolny język programowania;
- poszczególne moduły pseudoprogramu mogą być projektowane równolegle przez różnych specjalistów (np. od sterowania sygnalizacją, od badania zachowań pieszych na przejściach, od techniki jazdy, itd.);
- poprzez swoją w pewnym sensie "nieformalność" konwencja ta jest łatwa do opanowania przez osoby różnych zawodów, co ma poważne znaczenie przy tworzeniu projektów interdyscyplinarnych (dla osób nie znających LOGLANu wymagane jest tylko stworzenie minisłownika z podstawowymi definicjami).

Oczywiście, konwencja metaprogramistyczna wprowadza pewien "narzut" w postaci pojęć informatycznych dodatkowo występujących w opisie, nie występujących natomiast w rzeczywistości. Przykładowo: używając pojęć typu "proces", "klasa", "współbieżność"... ułatwia zbliżenie opisu do rzeczywistości, natomiast takie pojęcia informatyczne jak "synchronizacja procesów", "zmienna referencyjna", "plik" itd. - z punktu widzenia stricto opisu systemu są czymś sztucznym. Jest to jednak cena jaką się "płaci" za zbliżenie opisu do projektu programu. Można powiedzieć, że niejako "wadą" tej konwencji jest fakt - iż narzuca (wymusza) ona projekty programów. Naturalnie, wybór konwencji notacyjnej zależy od celu jakiemu ma służyć dany opis (w niniejszych rozważaniach obowiązuje milczące założenie, iż celem nadrzędnym jest taki opis, który ułatwia opracowanie modelu a następnie programu symulacyjnego).

Dla tak sformułowanego celu konwencja P-LOGLAN jest najlepsza. (W dobie rewolucji informatycznej można postawić tezę, że każda notacja powinna "asymptotycznie" dążyć do notacji metaprogramistycznej.)

Wysnuwając generalny wniosek, można stwierdzić, że konwencja P-LOGLAN bardzo dobrze nadaje się do opisu modeli systemów transportowych.

LITERATURA

[1] AHO A.V., HOPCROFT J.E., ULLMAN J.D.: Projektowanie i analiza algorytmów komputerowych. PWN Warszawa 1983.

[2] Instytut Transportu Politechniki Śląskiej (praca zbiorowa pod kier. dr inż. B. Wojciechowskiego): Zasady projektowania i wdrażania systemów sterowania ruchem ulicznym pod kątem minimalizacji energochłonności transportu. Katowice 1985.

[3] KONIECZNY R., KRAWIEC S.: Komputerowy model ruchu pojazdów w sieci skrzyżowań z sygnalizacją świetlną. Zeszyty Naukowe Politechniki Śląskiej, Transport nr 5, 1987.

[4] KONIECZNY R.: O pewnych konwencjach notacji algorytmów programów komputerowych - Zeszyty Naukowe Politechniki Śląskiej, Transport nr 6 / 1988 .

[5] KONIECZNY R. (praca zbiorowa): Zastosowanie języka LOGLAN do modelowania dużych systemów transportowych na przykładzie modelu ruchu pociągów - praca naukowo-badawcza NB-277/RT/87 wykonana w ramach programu RI.P.09 - Instytut Transportu Politechniki Śląskiej, Katowice 1987 .

[6] Report on the LOGLAN 82 programming language. PWN, Warszawa - Łódź, 1984.

LOGLAN PROGRAMMING LANGUAGE AS AN INSTRUMENT FOR DESCRIPTION OF TRANSPORT SYSTEM MODEL

Summary

The present article makes an attempt of answering the question: if a programming language (and precisely - LOGLAN language) can be used not only for the procedure algorithms notation or larger software designs but also for system specification.

Admitting the metaprogrammistic convention to be the most useful when designing complex systems software, P-LOGLAN notational convention (based on the idea of ALGOL Pidgin and on the LOGLAN-82 report) has been suggested.

It has been stated in the paper that P-LOGLAN convention is very suitable for the formal specification of complex transport systems.

DIE PROGRAMMIERSPRACHE LOGLAN ZUR BESCHREIBUNG DES MODELLS DES TRANSPORTSYSTEMS

Zusammenfassung

Vorliegender Aufsatz bildet eine Probe der Antwart auf die Frage: Soll eine Programmiersprache (konkret: die Sprache LOGLAN) nur zur Notation von Prozeduralgorithmen oder größeren Software-Projekten genutzt werden oder kann sie auch zur Systembeschreibung eingesetzt werden ?

Auerkennend die mataprogramistische Konvention als die nützliche bei der Entwicklung von Software-Projekten der Komplizierten Systemen wurde eine P-LOGLAN-Notationskonvention vorgeschlagen, die auf der Pidgin Idee von

ALGOL und auf dem LOGALN-82-Raport basiert.

Im Aufsatz wurde fastgestellt, daß die P-LOGLAN konvention sich sehr gut für formale Beschreibung Komplizierter Transportsysteme eignet.

ЯЗЫК ПРОГРАММИРОВАНИЯ LOGLAN КАК СРЕДСТВО ОПИСАНИЯ МОДЕЛИ ТРАНСПОРТНОЙ СИСТЕМЫ

Резюме

В статье дается попытка ответить на вопрос может ли язык программирования (а конкретно - язык LOGLAND) быть использован не только для записи алгоритмов процедур, или-же более обширных софтвер проектов, но также для описания систем.

Признавая метапрограмистическую конвенцию как наиболее полезной при проектировании отпраграммирования сложных систем, предложена конвенция по записи P-LOGLAN (базирующая на идеи Pidgin ALGOL а также на рапорте LOGLANa-82).

В статье показано, что конвенция P-LOGLAN очень полезна для описании сложных систем.