STUDIA UNIV. BABEŞ-BOLYAI, MATHEMATICA, XXXVII, 3, 1992

ALGEBRAIC SPECIFICATION OF PSP

Ilie PARPUCEA* and Bazil PÂRV**

Received: March 1, 1993
AMS Subject Classification: 68Q05

REZUMAT. - Specificarea algebrică a RSP. În lucrare se prezintă specificarea algebrică a tipurilor de date folosite în procesorul simbolic Poisson PSP ([Pâr89]). Se foloseşte un model ierarhic, bazat pe semantica denotaţională (algebrică).

0. Introduction. Algebraic specification of PSP (Poisson Symbolic Processor, see [Pâr89]) data types is made by means of an hierachical model, from simple to complex types. The abstract data structure of PSP is viewed as a hierachy of abstract data types.

A **Poisson series** have the form:

$$S = \sum_{i=0}^{\infty} C_i \, x_1^{j_1} x_2^{j_2} \ldots x_m^{j_m} \, {}^{\sin}_{\cos}(k_1 y_1 + k_2 y_2 + \ldots + k_n y_n), \tag{1}$$

where: C_i are *numerical coefficients*; $x_1, x_2, \ldots, x_m$ are monomial variables; $y_1, y_2, \ldots, y_n$ are trigonometric variables; $j_1, j_2, \ldots, j_m$ and $k_1, k_2, \ldots, k_n$ are exponents, and, respectively, coefficients. The summation index i covers the set of all possible combinations of the exponents j and coefficients k ($j \in \mathbb{Z}^m, k \in \mathbb{Z}^n$).

The form (1) of Poisson series can be briefly written as follows:

$$S = \sum_{i=0}^{\infty} T_i, \tag{2}$$

in which T_i is a *term* of this series:

* *"Babeş-Bolyai" University, Department of Economic Sciences 3400 Cluj-Napoca, Romania*

** *"Babeş-Bolyai" University, Faculty of Mathematics and Computer Science, 3400 Cluj-Napoca, Romania*

$$T_i = C_i \ P_i \ F_i \ ,$$

where the *polynomial part* P_i has the form:

$$P_i = x_1^{j_1} x_2^{j_2} \ldots x_m^{j_m}, \tag{3}$$

while the *trigonometric part* F_i is:

$$F_i = {\sin \atop \cos}(k_1 y_1 + k_2 y_2 + \ldots + k_n y_n) \,. \tag{4}$$

In practice, one does not operate with Poisson series, but with partial sums of these ones, called *Poisson expressions*, of the form:

$$S = \sum_{i=0}^{N} T_i, \quad N \in \mathbf{N}. \tag{5}$$

PSP operates with Poisson expressions of the form (5). The hierarchical model of its algebraic specification consists of five levels:

1) numerical coefficients specification;
2) trigonometric part specification;
3) polynomial part specification;
4) term specification;
5) Poisson expression specification.

1. Numerical coefficients specification. The coefficients C_i from (1) are considered rational numbers, of the form M/N, with $M,N \in \mathbf{Z}$. The definition of the abstract data type RAT follows the chain:

BOOL --> NAT --> INT --> RAT

where:

BOOL - represents the primitive boolean type;

NAT - represents the hierarchical natural type (including zero value);

INT - represents the hierarchical integer type;

RAT - represents the hierarchical rational type.

In the specification of the NAT type we use the NAT* subtype of NAT, which corresponds to the natural numbers without zero value. The above listed types are specified as follows:

1.1. The primitive BOOL type

```
type BOOL =
     SORT bool,
     CONS
          true: --> bool,
          false: --> bool,
     OPNS
          .not.: bool --> bool,
          .and.: bool, bool --> bool,
          .or.:  bool, bool --> bool,
     VARS
           X,Y: bool,
     AXIOMS
           true ≠ false,
           .not. true = false,
           .not. false = true,
           true .and. X = X,
           false .and. X = false,
           X .or. Y = .not.((.not. X) .and. (.not. Y))
endoftype.
```

1.2. The NAT and NAT* types

The NAT type uses the primitive type BOOL:

```
type NAT =
     SORT bool, nat,
     CONS
          zero: --> nat,
          succ: nat --> nat,
          pred: (nat x: x noteq zero) --> nat,
     OPNS
          *: nat, nat --> nat,
          eq: nat, nat --> bool,
          +: nat, nat --> nat,
          noteq: nat, nat --> bool,
          < : nat, nat --> bool,
          [_,_]: nat, (nat x: x noteq zero) --> nat,
          GCD(_,_): nat, nat --> nat,
     VARS
          R, P, M, N: nat,
     AXIOMS
     *    N eq M  =  M eq N,
```

```
*    zero eq zero  = true,
*    zero eq succ(N)  =  false,
*    succ(N) eq succ(M)  =  N eq M,
*    N noteq M  =  not (N eq M),
*    pred(succ(N))  =  N,
     [N,1]  =  N,
     (∃ R, P: nat : R < M, N = P * M + R) ==> [N , M]  = P,
     GCD(N, 0)  =  N,
     U < V ==> GCD(V, U)  =  GCD(U, V - U * [V , U]),
     GCD(U, V)  =  GCD(V, U),
*    N * 0  =  0  =  0 * N,
*    N * succ(M)  =  (N * M) + N  =  succ(M) * N,
*    N * pred(M)  =  (N * M) - N  =  pred(M) * N,
*    N + 0  =  0 + N  =  N,
*    N + succ(M)  =  succ(N + M)  =  succ(M) + N,
*    N + pred(M)  =  pred(N + M)  =  pred(M) + N,
     (0 < succ(0))  =  true,
     (pred(N) < succ(N))  =  true,
     (N < succ(N))  =  true,
     (pred(N) < N)  =  true,
     N < M  ==>  (succ(N) < succ(M)  =  true),
     N < M  ==>  (pred(N) < pred(M)  =  true),
*    N - 0  =  N,
*    IF  M < N  THEN  N - succ(M)  =  pred(N - M),
*    IF  M < N  THEN  N - pred(M)  =  succ(N - M),
endoftype.
```

We define NAT*, subtype of the NAT type:

```
type NAT* =
     SORT bool, nat*,
     CONS
          one: --> nat*,
          succ: nat* --> nat*,
          pred: (nat* x: x noteq one) --> nat*,
     OPNS
          +: nat* --> nat*,
          -: nat* --> nat*,
          *: nat*, nat* --> nat*,
          eq: nat*, nat* --> bool,
          noteq: nat*, nat* --> bool,
     VARS
          M, N: nat*,
     AXIOMS
          The axioms of NAT type marked with * are axioms of the
            subtype NAT*, with the following modifications:
          one eq one  =  true,
          one eq succ(N)  =  false,
          N * 1  =  1 * N  =  N,
          N + 1  =  1 + N  =  succ(N),
          IF  1 < N  THEN  N - 1  =  pred(N),
endoftype.
```

1.3. The INT type

The specification of INT type uses BOOL and NAT as primitive types:

```
type INT ≡
     SORT bool, int, nat,
     CONS
          zero: --> int,
          succ: int --> int,
          pred: int --> int,
     OPNS
          eq: int, int: --> bool,
          noteq: int, int --> bool,
          +: int, int --> int,
          -: int, int --> int,
          *: int, int --> int,
          <: int, int --> bool,
          |_|: int --> nat,
     VARS
          N, M: int,
     AXIOMS
          N eq M  =  M eq N,
          zero eq zero  =  true,
          succ(N) eq succ(M)  =  N eq M,
          N noteq M  =  not (N eq M),
          pred(succ(N))  =  N,
          succ(pred(N))  =  N,
          | 0 |  =  0,
          (pred(0) < 0)  =  true,
          The axioms of INT which refer to the operations +, -
             and < are inherited from NAT type;
endoftype.
```

Remark. The INT type contains the *zero* value and two unary operations, *succ* and *pred*. The integer number n (n>0) is obtained by n successive applications of the *succ* operation to 0. Analogously, the integer number -n (n>0) is constructed with *pred*, starting with 0.

1.4. The RAT type

The specification of RAT type uses INT and NAT* as primitive types:

```
type RAT ≡
     SORT int, nat*, rat,
     CONS
```

```
        _/_: int, nat* --> rat,
    OPNS
        _+_: rat, rat --> rat,
        _*_: rat, rat --> rat,
        _:_: rat, rat --> rat,
        _-_: rat, rat --> rat,
    VARS
        M/N, P/Q: rat,
        S, T: nat*,
    AXIOMS
        (M/N) + (P/Q)  = ((M * Q) + (P * N)) / (N * Q),
        (M/N) * (P/Q)  = (M * P) / (N * Q),
        (M/N) : (P/Q)  = (M * Q) / (N * P),
        (M/N) - (P/Q)  = ((M * Q) - (P * N)) / (N * Q),
        (∃ S: nat*, ∃ T: nat*, :
          M  =  S * GCD(|M|,|N|) ∧ N  =  T * GCD(|M|,|N|)) →
          → M/N = S/T
endoftype.
```

2. Trigonometric part specification. The definition of the trigonometric parts F_i from (4) follows the chain:

SET --> FLIN --> TTR

where:

SET - represents a symbol set;

FLIN - represents the abstract type of linear forms;

TTR - represents the abstract type of trigonometric parts.

2.1. The SET type

The SET type consists of a set of symbols. This type is used as primitive type for the specification of FLIN type.

```
type SET =
    SORT set,
    CONS
        X1 : --> set,
        X2 : --> set,
        - - - - - - -
        Xn : --> set,
        Y1 : --> set,
        Y2 : --> set,
        - - - - - - -
        Ym : --> set
endoftype.
```

2.2. The FLIN type

The FLIN type defines the linear forms over the types SET and INT.

```
type FLIN =
     SORT int, set, flin,
     CONS
          _·_ (concatenation) : int, set --> flin,
     OPNS
          _+_: flin, flin --> flin,
          _-_: flin, flin --> flin,
          _*_: int, flin --> flin,
     VARS
          X, Y, Z: set,
          M, N, K: int,
     AXIOMS
          X·1  =  1·X  =  X,
          X·0  =  0·X  =  0,
          N·X  =  X·N,
          X·(N + M)  =  (N + M)·X  =  N·X + M·X,
          N·X + M·Y  =  M·Y + N·X,
          (N·X + M·Y) + K·Z  =  N·X + (M·Y + K·Z),
          N·X + 0  =  0 + N·X  =  N·X,
          N * 0  =  0  =  0 * N,
          1 * X  =  X * 1  =  X,
          N * (M·X)  =  (N * M)·X,
          N * (M·X ± K·Y)  =  (N * M)·X ± (M * K)·Y
endoftype.
```

The FLIN type contains linear combinations of the form:

$$N_1 \cdot Y_1 + N_2 \cdot Y_2 + \ldots + N_k \cdot Y_k$$

which can be used as arguments for sine and cosine functions in trigonometric parts F_i. The three operations (+, * and concatenation), together with the axioms, are used for the specification of the trigonometric part.

2.3. The TTR type

```
type TTR =
     SORT flin, ttr,
     CONS
          cos: flin --> ttr,
          sin: flin --> ttr,
     AXIOMS
          cos 0  =  1,
          sin 0  =  0,
endoftype.
```

3. Polynomial part specification. The definition of the polynomial parts P_i from (3) follows the scheme:

MPOL --> PPOL

where:

MPOL - represents the set of monomials;

PPOL - represents the set of polynomials.

3.1. The MPOL type

The MPOL type defines the set of monomials of the form X^N, where X is in SET and N is of INT type.

```
type MPOL =
      SORT int, set, mpol,
      CONS
           _^_ (raise to power) : set, int --> mpol
                        ( X^N  =  X^N )
      VARS
           X: set, N: int,
      AXIOMS
           (X ^ 0)  =  1,
           (X ^ 1)  =  X,
endoftype.
```

3.2. The PPOL type

The PPOL type defines the set of polynomials of the form (3):

```
type PPOL =
      SORT mpol, ppol
      CONS
           _*_: mpol, mpol --> mpol,
      VARS
           X^M , X^N , Y^M , Z^K : ppol,
      AXIOMS
           X^M  * X^N   =  X^N  * X^M   =  X^(M+N) ,
           X^N  * Y^M   =  Y^M  * X^N ,
           X^N  * 1  =  X^N  =  1 * X^N ,
           X^N  * (Y^M  * Z^K )  =  (X^N  * Y^M ) * Z^K
endoftype.
```

4. Term specification. The terms T_i from (2) are defined as follows:

```
type TERM ≡
     SORT rat, ttr, ppol, term,
     CONS
          · (concatenation) : rat, ttr, ppol --> term,
     VARS
          N/M  : rat, cosY, sinY: ttr, X: ppol,
* * * We denote with Y the linear form and with X the polynomial
* * * form
     AXIOMS
                    sinY                     sinY
          (N/M · {      }) · X = N / M · ({      }) · X  ,
                    cosY                     cosY
                   sinY         sinY                sinY
          N/M · {      } · X = {      } · N/M · X = {      } · X · N/M
                   cosY         cosY                cosY
                                                sinY
                                = X · N/M · {      } ,
                                                cosY
          0 · 0 · X = 0 ,
          1/1 · 1 · X = X ,
               sinY
          0 · {      } · X = N/M · 0 · X = 0
               cosY
endoftype.
```

5. Poisson expression specification. Taking into account the above definitions, the Poisson expressions (5) will be defined in the following form:

```
type EXP ≡
     SORT term, exp,
     CONS
          _+_: term, term --> exp,
          _-_: term, term --> exp,
          _*_: term, term --> exp,
     OPNS
          ∂ (differentiation) : exp, set --> exp,
          ∫ (integration) : exp, set --> exp,
     VARS
          N/M , P/Q : rat,
          Y, Y1 , Y2 : flin,
          X, X1 , X2  : mpol,
     AXIOMS
                   sinY                 sinY
          N/M · {      } · X ± P/Q · {      } · X =
                   cosY                 cosY
```

$$= (N/M \pm P/Q) \cdot X \cdot \left\{ \begin{matrix} \sin Y \\ \cos Y \end{matrix} \right\},$$

$$(N/M \cdot \cos Y_1 \cdot X_1) * (P/Q \cdot \sin Y_2 \cdot X_2) =$$
$$= (1/2 * N/M * P/Q) \cdot X_1 \cdot X_2 \cdot \sin(Y_1+Y_2) +$$
$$+ (1/2 * N/M * P/Q) \cdot X_1 \cdot X_2 \cdot \sin(Y_2-Y_1),$$

$$(N/M \cdot \sin Y_1 \cdot X_1) * (P/Q \cdot \sin Y_2 \cdot X_2) =$$
$$= (1/2 * N/M * P/Q) \cdot X_1 \cdot X_2 \cdot \cos(Y_1-Y_2) -$$
$$- (1/2 * N/M * P/Q) \cdot X_1 \cdot X_2 \cdot \cos(Y_1+Y_2),$$

$$(N/M \cdot \cos Y_1 \cdot X_1) * (P/Q \cdot \cos Y_2 \cdot X_2) =$$
$$= (1/2 * N/M * P/Q) \cdot X_1 \cdot X_2 \cdot \cos(Y_1+Y_2) +$$
$$+ (1/2 * N/M * P/Q) \cdot X_1 \cdot X_2 \cdot \cos(Y_1-Y_2),$$

$$\frac{\partial}{\partial Y_k} (N/M \cdot \left\{ \begin{matrix} \cos(N_1 \cdot Y_1+N_2 \cdot Y_2+\ldots+N_k \cdot Y_k+\ldots+N_l \cdot Y_l) \\ \sin(N_1 \cdot Y_1+N_2 \cdot Y_2+\ldots+N_k \cdot Y_k+\ldots+N_l \cdot Y_l) \end{matrix} \right\} \cdot$$
$$\cdot X_1^{M1} \cdot \ldots \cdot Y_k^{Mk} \cdot \ldots \cdot X_h^{Mh}) =$$

$$= N*M_k/M \cdot \left\{ \begin{matrix} \cos(N_1 \cdot Y_1+N_2 \cdot Y_2+\ldots+N_k \cdot Y_k+\ldots+N_l \cdot Y_l) \\ \sin(N_1 \cdot Y_1+N_2 \cdot Y_2+\ldots+N_k \cdot Y_k+\ldots+N_l \cdot Y_l) \end{matrix} \right\} \cdot$$
$$\cdot X_1^{M1} \cdot \ldots \cdot Y_k^{Mk-1} \cdot \ldots \cdot X_h^{Mh}) \mp$$

$$\mp N*N_k/M \cdot \left\{ \begin{matrix} \sin(N_1 \cdot Y_1+N_2 \cdot Y_2+\ldots+N_k \cdot Y_k+\ldots+N_l \cdot Y_l) \\ \cos(N_1 \cdot Y_1+N_2 \cdot Y_2+\ldots+N_k \cdot Y_k+\ldots+N_l \cdot Y_l) \end{matrix} \right\} \cdot$$
$$\cdot X_1^{M1} \cdot \ldots \cdot Y_k^{Mk} \cdot \ldots \cdot X_h^{Mh}),$$

$$I_0^{N_k} = \int N/M \cdot \sin(N_1 \cdot Y_1+N_2 \cdot Y_2+\ldots+N_k \cdot Y_k+\ldots+N_l \cdot Y_l) \cdot$$
$$\cdot X_1^{M_1} \cdot \ldots \cdot X_{k-1}^{M_{k-1}} \cdot Y_k^{0} \cdot X_{k+1}^{M_{k+1}} \cdot \ldots \cdot X_h^{M_h} \, dY_k =$$
$$= (-1/N_k*N/M) \cdot \cos(N_1 \cdot Y_1+N_2 \cdot Y_2+\ldots+N_k \cdot Y_k+\ldots+N_l \cdot Y_l) \cdot$$
$$\cdot X_1^{M_1} \cdot \ldots \cdot X_{k-1}^{M_{k-1}} \cdot Y_k^{0} \cdot X_{k+1}^{M_{k+1}} \cdot \ldots \cdot Y_h^{M_h}$$

$$I_1^{N_k} = \int N/M \cdot \sin(N_1 \cdot Y_1+N_2 \cdot Y_2+\ldots+N_k \cdot Y_k+\ldots+N_l \cdot Y_l) \cdot$$
$$\cdot X_1^{M_1} \cdot \ldots \cdot X_{k-1}^{M_{k-1}} \cdot Y_k^{1} \cdot X_{k+1}^{M_{k+1}} \cdot \ldots \cdot X_h^{M_h} \, dY_k =$$

$$= (-1/N_k{*}N/M) \cdot \cos(N_1 \cdot Y_1+N_2 \cdot Y_2+\ldots+N_k \cdot Y_k+\ldots+N_l \cdot Y_l) \cdot$$

$$\cdot\ X_1^{M_1} \cdot \ldots . X_{k-1}^{M_{k-1}} \cdot Y_k^{1} \cdot X_{k+1}^{M_{k+1}} \cdot \ \ldots \ \cdot X_h^{M_h} +$$

$$+ (1/(N_k{*}N_k)) \cdot \sin(N_1 \cdot Y_1+N_2 \cdot Y_2+\ldots+N_k \cdot Y_k+\ldots+N_l \cdot Y_l) \cdot$$

$$\cdot\ X_1^{M_1} \cdot \ldots . X_{k-1}^{M_{k-1}} \cdot Y_k^{0} \cdot X_{k+1}^{M_{k+1}} \cdot \ \ldots \ \cdot X_h^{M_h}$$

$$I_p^{N_k} = \int N/M \cdot \sin(N_1 \cdot Y_1+N_2 \cdot Y_2+\ldots+N_k \cdot Y_k+\ldots+N_l \cdot Y_l) \cdot$$

$$\cdot\ X_1^{M_1} \cdot \ldots . X_{k-1}^{M_{k-1}} \cdot Y_k^{p} \cdot X_{k+1}^{M_{k+1}} \cdot \ \ldots \ \cdot X_h^{M_h}\ dY_k =$$

$$= (-1/N_k{*}N/M) \cdot \cos(N_1 \cdot Y_1+N_2 \cdot Y_2+\ldots+N_k \cdot Y_k+\ldots+N_l \cdot Y_l) \cdot$$

$$\cdot\ X_1^{M_1} \cdot \ldots . X_{k-1}^{M_{k-1}} \cdot Y_k^{M_k} \cdot X_{k+1}^{M_{k+1}} \cdot \ \ldots \ \cdot X_h^{M_h} +$$

$$+(N/M{*}p/(N_k{*}N_k)) \cdot \sin(N_1 \cdot Y_1+N_2 \cdot Y_2+\ldots+N_k \cdot Y_k+\ldots+N_l \cdot Y_l) \cdot$$

$$\cdot\ X_1^{M_1} \cdot \ldots . X_{k-1}^{M_{k-1}} \cdot Y_k^{p-1} \cdot X_{k+1}^{M_{k+1}} \cdot \ \ldots \ \cdot X_h^{M_h} -$$

$$- (N/M{*}p/N_k{*}(p-1)/N_k) \cdot I_{p-2}^{N_k}$$

endoftype.

Remark. The EXP type must also contain the axioms referring to the recurrent computation of the integral:

$$J_p^{M_k} = \int N/M \cdot \cos(N_1 \cdot Y_1+N_2 \cdot Y_2+\ldots+N_k \cdot Y_k+\ldots+N_l \cdot Y_l) \cdot$$

$$\cdot\ X_1^{M_1} \cdot \ldots . X_{k-1}^{M_{k-1}} \cdot Y_k^{p} \cdot X_{k+1}^{M_{k+1}} \cdot \ \ldots \ \cdot X_h^{M_h}\ dY_k =$$

$$= (-1/N_k{*}N/M) \cdot \sin(N_1 \cdot Y_1+N_2 \cdot Y_2+\ldots+N_k \cdot Y_k+\ldots+N_l \cdot Y_l) \cdot$$

$$\cdot\ X_1^{M_1} \cdot \ldots . X_{k-1}^{M_{k-1}} \cdot Y_k^{p} \cdot X_{k+1}^{M_{k+1}} \cdot \ \ldots \ \cdot X_h^{M_h} +$$

$$+(N/M{*}p/(N_k{*}N_k)) \cdot \cos(N_1 \cdot Y_1+N_2 \cdot Y_2+\ldots+N_k \cdot Y_k+\ldots+N_l \cdot Y_l) \cdot$$

$$\cdot\ X_1^{M_1} \cdot \ldots . X_{k-1}^{M_{k-1}} \cdot Y_k^{p-1} \cdot X_{k+1}^{M_{k+1}} \cdot \ \ldots \ \cdot X_h^{M_h} -$$

$$- (N/M{*}p/N_k{*}(p-1)/N_k) \cdot J_{p-2}^{p}$$

CONCLUDING REMARKS. This paper specifies the data types defined in PSP (implemented in Pascal) in a hierarchical way. Some of the advantages of algebraic specifications are used, especially those involving the correctness of the defined operations. In such a way the singular cases (as non-determinations or exceptions) are avoided. The authors intention is to simulate the operational semantics of PSP by using terms rewrite rules. PSP can also be specified in the specification and programming language OBJ3 (see [Gog88] and [Kir87]).

REFERENCES

[Wir82] Wirsing,M., Broy,M.: *An analysis of semantic models for algebraic specification,* in Broy,M., Schmidt,G.(eds.), Theoretical foundations of programming methodology, Reidel, Dordrecht, 1982, 351-412.

[Wir83] Wirsing,M.,Pepper,P.,Partsch,H.,Dosch,W.,Broy,M.: *On hierarchies of abstract data types,* Acta Informatica, 20, 1983, 1-33.

[Kir87] Kirchner,C.,Kirchner,H.,Meseguer,J.: *Operational semantics of OBJ3,* Technical Report, SRI International, 1987.

[Gog88] Goguen,J.,Winkler,T.: *Introducing OBJ3,* Technical Report, SRI International, 1988.

[Pâr89] Pârv,B.: *Poisson Symbolic Processor,* Studia, Mathematica, XXXIV, 1989, No.3, 17-29.