NAME

pic2lnt − translator of the applied pi-calculus to LOTOS NT

SYNOPSIS

pic2lnt [**−pidlist**] [**−root** *instantiation*] [**−silent** | **−verbose**] [**−version**] *filename*[**.pic**]

DESCRIPTION

The pi-calculus [MPW92] is a specification formalism for concurrent processes involving mobile communication. LOTOS NT is an imperatively styled specification language for concurrent processes. The applied pi-calculus (hereafter noted PIC) is an extension of the polyadic pi-calculus with the data types, functions, and expressions of LOTOS NT. The **pic2lnt** program translates a PIC specification into a LOTOS NT specification. The translation implemented by **pic2lnt** generalizes the approach proposed in [MS10] for translating polyadic pi-calculus into LOTOS NT.

The input to **pic2lnt** is a PIC file, which must have the extension **`.pic`**. If the user does not specify the extension **`.pic`** on the command line, it will be appended automatically and **pic2lnt** will read *filename***`.pic`** as input.

The data types and functions used in a PIC specification must be defined in external LOTOS NT files **`.lib`**, which must be included in the PIC file (see SPECIFICATION below).

The output of **pic2lnt** is a LOTOS NT specification. The name of the input file is used to construct the name of the output file. For an input file **`example.pic`**, **pic2lnt** creates the LOTOS NT specification **`example.lnt`**.

OPTIONS

−pidlist List the names of all processes without parameters that occur in the input file and exit. In other words, list all processes that can be used as main process (by instantiation with the **−root** option).

−root *instantiation*

Use the process *instantiation* as the main process, meaning that the generated LOTOS NT specification has the behaviour of the main process. The *instantiation* is the identifier of a PIC process having no parameters. If this option is absent, the PIC specification must contain a process without parameters, named MAIN (letter case is not relevant), which is the default main process.

−version

Display the tool version and exit.

APPLIED PI-CALCULUS

The PIC language is an extension of the polyadic pi-calculus with the data types, functions, and expressions of LOTOS NT.

An overview of PIC is given below. The abstract syntax of PIC is defined by an extended BNF (EBNF) grammar and the semantics of each PIC construct is defined informally. The EBNF grammar uses the following conventions: PIC keywords are enclosed between double quotes; a list of *n* elements is noted *ELEM1 ... ELEMn*; a list indexed from 0 (e.g., *X0, ..., Xn*) has at least one element; a list indexed from 1 (e.g., *X1, ..., Xn*) may have 0 elements; optional constructs are enclosed between square brackets.

The symbols of the PIC grammar are listed in the table below. The axiom of the grammar is the *SPEC* symbol.

	Symbol	Description
Non-terminal	*SPEC*	specification
	PROC	process definition
	DECL	channel or variable declaration

	B	**behaviour**
Terminal	*M*	**LOTOS NT module**
	C	**channel**
	X	**variable**
	F	**function**
	OP	**operator**
	T	**type**
	K	**constant**

Identifiers are built from letters, digits, and underscores (beginning with a letter or an underscore). Keywords must be written in lowercase. Comments are enclosed between "**(**∗" and "∗**)**" and may not be nested. **pic2lnt** is case-sensitive.

VALUE EXPRESSIONS

The syntax of PIC value expressions is defined by the following grammar:

E **::=** *C*

| *X*

| *K*

| *F* **["(**" *E1* "**,**" ... "**,**" *En* "**)**" **]**

| *OP E*

| *E1 OP E2*

| *E* **"of"** *T*

| *E* **"."** *X*

| *E* **"."** **"{"** *X0* **"=>"** *E0*, ..., *Xn* **"=>"** *En* **"}"**

| *E1* **"["** *E2* **"]"**

| **"("** *E* **")"**

The semantics of PIC value expressions is defined informally below.

C

is an occurrence of a channel name *C*. As in standard pi-calculus, channel occurrences can be either *bound* (i.e., corresponding to channel names *C* defined by an enclosing reception or a "new" operator, see BEHAVIOURS below) or *free*.

Channel names *C* are equipped with the standard comparison operators "**=**" and "**<>**".

X

is a use occurrence of a data variable previously defined in the PIC specification.

K

is a constant. The syntax of numerical, character, and string constants in PIC is identical to that of LOTOS NT (e.g., **13**, **-1**, **1.618**, **'a'**, **'\007'**, **'\n'**, **"hello world\n"**).

F ["(" *E1* "," ... "," *En* ")"]

is the call of function *F* having as arguments the values of expressions *E1*, ..., *En*. The arguments must correspond in number and types to the formal parameters given in the definition of *F* (which must be present in some external LOTOS NT module, see SPECIFICATION below). Function calls with no arguments can be written either without parentheses (e.g., *F*), or with parentheses (e.g., *F* ()).

Note: If a name *F* without parentheses occurs in an expression, two cases are possible:

- If the name *F* does not correspond to a data variable previously declared, then *F* is interpreted as the free occurrence of a channel. If the name *F* denotes a function without arguments defined in some LOTOS NT library, then it must be written using parentheses (i.e., *F* ()) in order to avoid its interpretation as a free channel occurrence.

- If the name *F* corresponds both to a data variable *and* to a bound channel previously declared, then *F* is interpreted as a use occurrence of the data variable. If the name *F* must be interpreted as a channel name, then the name of the channel must be changed in order to avoid the clash with the name of the data variable.

OP E

E1 OP E2

are calls to unary (prefixed) and binary (infixed) operators, respectively. Standard arithmetic and relational operators are allowed (e.g., "**-**", "**+**", "∗", "**<=**", etc.).

Syntactically, all binary operators are left-associative and have the same precedence (hence, one should explicitly add parentheses around subexpressions to ensure a desired evaluation order, e.g., write "**x + (y ∗ z)**" instead of "**x + y ∗ z**", which would be evaluated by default as "**(x + y) ∗ z**"). All unary operators have the same precedence, which is higher than the precedence of binary operators.

E "of" *T*

is the explicit typing operator, which indicates that expression *E* is of type *T*. This operator is useful for disambiguating certain value expressions containing overloaded LOTOS NT functions.

E "." *X*

is the field selection operator. It denotes the value of field *X* of the record value produced by evaluating expression *E* (which must be of record type containing a field *X*).

E "." "{" *X0* "=>" *E0* "," ... "," *Xn* "=>" *En* "}"

is the field updating operator. It denotes the value of expression *E* (which must be of record type containing the fields *X0*, ..., *Xn*) in which the fields *X0*, ..., *Xn* have been replaced with the values of expressions *E0*, ..., *En*, respectively.

E1 "[" *E2* "]"

is the array element operator, which denotes the element stored at the index obtained by evaluating expression *E2* of the array obtained by evaluating expression *E1*. Expression *E1* must be of array type and expression *E2* must be of type **Nat**, which is predefined in LOTOS NT.

"(" *E* ")"

has the same meaning as expression *E*. Parentheses are useful for imposing an evaluation order of subexpressions different from the order given by the associativity and precedence of operators.

DECLARATIONS

Similarly to classical programming languages, PIC provides mechanisms for declaring channels and data variables. The syntax of PIC declarations (without initialization) is defined by the following grammar:

DECL **::=** *C*

| *X* **":"** *T*

Variable declarations have also a generalized form:

X01 **","** **...** **","** *X0m0* **":"** *T0*
"," **...** **","**
Xn1 **","** **...** **","** *Xnmn* **":"** *Tn*

which declares the data variables *Xi1*, ..., *Ximi* of type *Ti* for each 0 <= *i* <= *n*. This general form of declaration is equivalent to the simplified form below, which will be used in the remainder of this manual page:

X01 **":"** *T0* **","** **...** **","** *X0m0* **":"** *T0*
"," **...** **","**
Xn1 **":"** *Tn* **","** **...** **","** *Xnmn* **":"** *Tn*

In the same way, variable declarations with initialization have the following general form:

X01 **","** **...** **","** *X0m0* **":"** *T0* **":="** *E0*
"," **...** **","**
Xn1 **","** **...** **","** *Xnmn* **":"** *Tn* **":="** *En*

which declares the data variables *Xi1*, ..., *Ximi* of type *Ti* and initializes them with the value of the expression *Ei* for each 0 <= *i* <= *n*. This general form of declaration with initialization is equivalent to the simplified form below, which will be used in the remainder of this manual page:

X01 **":"** *T0* **":="** *E0* **","** **...** **","** *X0m0* **":"** *T0* **":="** *E0*
"," **...** **","**
Xn1 **":"** *Tn* **":="** *En* **","** **...** **","** *Xnmn* **":"** *Tn* **":="** *En*

BEHAVIOURS

Concurrent processes in PIC can manipulate data values in addition to communication channels. Their behaviour is described using (generalizations of) the polyadic pi-calculus operators and some data-handling constructs inherited from LOTOS NT. The syntax of PIC behaviours is defined by the following grammar:

B **::=** **"nil"**

| *P* **[** **"("** *E0* **","** **...** **","** *En* **")"** **]**

| **"tau"** **"."** *B*

| **"'"** *C* **[** **"<"** *E1* **","** **...** **","** *En* **">"** **]** **"."** *B*

| *C* **[** **"("** *DECL1* **","** **...** **","** *DECLn* **")"** **]** **"."** *B*

| **"["** *E* **"]"** *B*

| **"!"** **[** *K* **]** *B*

| **"("** **"new"** *C0* **","** **...** **","** *Cn* **")"** *B*

| **"var"** *X0* **":"** *T0* **":="** *E0* **","** **...** **","** *Xn* **":"** *Tn* **":="** *En* **"in"**
B
"end" **"var"**

| *B1* **"+"** *B2*

| *B1* "|" *B2*

| **"("** *B* **")"**

Syntactically, all binary operators are left-associative. Unary operators ("**.**", "**[**...**]**", "**!**", "new") have the highest precedence, followed by "**|**", followed by "**+**".

The semantics of a PIC behaviour *B* is given by a Labeled Transition System (LTS) $M = < S, A, T, s0 >$, where: *S* is the set of states; *A* is the set of actions; *T* included in $S * A * S$ is the transition relation; and *s0* in *S* is the initial state. An action *a* of *A* is either the silent action *i* (this is the LOTOS NT notation, since the LTS denoting the semantics of a PIC behaviour is produced from the corresponding LOTOS NT code generated by **pic2lnt**), or it has the form *c v1 ... vn*, where *c* is a channel name and *v1, ..., vn* is a (possibly empty) list of channel names or data values exchanged during the rendezvous communication on channel *c*. This form of action corresponds to the *early* operational semantics of the pi-calculus (see, e.g., [MP95,Par01]). A transition (*s1*, *a*, *s2*) of *T* (also noted *s1-a->s2*) means that the system can move from state *s1* to state *s2* by performing action *a*.

The semantics of PIC behaviours is described informally below.

"nil"

is the empty behaviour (deadlock), which does not perform any action.

P ["(" *E0* "," ... "," *En* ")"]

is a call to process *P* having as arguments the values of expressions *E0, ..., En*. The arguments must correspond in number and types to the formal parameters given in the definition of process *P* (see PROCESS DEFINITIONS below). This behaviour corresponds to the body of process *P*, in which the formal parameters have been replaced by the arguments *E0, ..., En*, respectively.

"tau" "." *B*

is the silent prefix operator. It performs a transition labeled by the silent action (noted **i** in LOTOS NT) and continues with the behaviour *B*.

"'" *C* ["<" *E1* "," ... "," *En* ">"] "." *B*

is the emission on channel *C* of the channel names or values *v1, ..., vn* obtained by evaluating the expressions *E1, ..., En*. It performs a transition labeled by an action of the form *C v1 ... vn* and continues with the behaviour *B*.

C ["(" *DECL1* "," ... "," *DECLn* ")"] "." *B*

is the reception on channel *C* of *n* channel names or values *v1, ..., vn*, the types of which must correspond to the declarations *DECL1, ..., DECLn*, respectively. If a declaration *DECLi* denotes a channel name *Ci*, then a channel name *Ci'* is expected to be received at position *i* and is stored in *Ci*. If a declaration *DECLj* denotes a variable *Xi* of type *Ti*, then a value *vi* of type *Ti* is expected to be received at position *j* and is stored in *Xi*. It performs a transition labeled by an action of the form *C v1 ... vn* and continues with the behaviour *B*.

All channel names and variables declared by *DECL1, ..., DECLn* are visible in *B*.

"[" *E* "]" *B*

is the guard operator. It has the behaviour *B* if the boolean condition *E* evaluates to true and is equivalent to "nil" otherwise. The guard operator of PIC accepts any boolean expression *E*,

whereas the guard operator of standard pi-calculus allows only the equality test between channel names. The "if-then-else" conditional behaviour can be simulated by using the selection operator and mutually exclusive guard operators as follows [Par01]:

if E then B1 else B2 = [E] B1 + [not (E)] B2

"!" [*K*] *B*

is the bounded replication operator. It is a shorthand notation for the behaviour "*B* | ... | *B*", where *B* occurs *K* times, where *K*>=1 is a constant value of type **`Nat`**. If *K* is absent, it has the default value 1, meaning that the bounded replication operator is a shorthand for *B*.

"(" "new" *C0* "," ... "," *Cn* ")" *B*

is the channel creation operator. It defines the "fresh" channel names *C0*, ..., *Cn*, which are visible in *B*. These channel names are *private*, meaning that they cannot be used for communication between *B* and another behaviour *B'* unless they are communicated by *B* to *B'* on some other channel name known by both *B* and *B'* (in this case, a channel *Ck* sent by *B* to *B'* becomes visible in *B'* by scope extrusion).

"var" *X0* ":" *T0* ":=" *E0* "," ... "," *Xn* ":" *Tn* ":=" *En* "in"
B
"end" "var"

is the variable definition operator. It declares the variables *X0*, ..., *Xn* and initializes them with the result of evaluating the expressions *E0*, ..., *En*, which must be of type *T0*, ..., *Tn*, respectively.

The variables *X0*, ..., *Xn* are visible in *B*.

B1 "+" *B2*

is the choice operator. It performs either an action of *B1* and continues with the behaviour of *B1*, or an action of *B2* and continues with the behaviour of *B2*.

B1 "|" *B2*

is the parallel composition operator. It produces the interleaved execution of the behaviours *B1* and *B2* according to the early operational semantics of pi-calculus [MP95,Par01].

"(" *B* ")"

has the same meaning as behaviour *B*. Parentheses are useful for imposing an evaluation order of behaviours different from the order given by the associativity and precedence of operators.

PROCESS DEFINITIONS

Concurrent behaviours can be encapsulated into processes parameterized by channel names and data values, which can be instantiated multiple times in a PIC specification. The syntax of PIC process definitions is defined by the following grammar:

PROC **::=** *P* **[** **"("** *DECL 1* **","** **...** **","** *DECL n* **")"** **]** **"="** *B*

This defines the process *P*, equipped with the optional (channel or variable) parameters declared by *DECL1*, ..., *DECLn*. The parameters are visible in the behaviour *B*, which is the body of process *P*. A process *P* without parameters can be called using the syntax *P* or *P*(). Processes can be mutually recursive.

Note: The current version of **pic2lnt** handles only PIC specifications satisfying the *finite control* property [Dam94], which forbids recursive process calls through the parallel composition operator. This hypothesis was used in the translation from polyadic pi-calculus to LOTOS NT [MS10] and was maintained for the translation from PIC to LOTOS NT.

For example, the process definition below does not satisfy the finite control property:

P = P | (a . nil)

Note: Recursive process calls through the channel creation operator are permitted. For instance, the following process definition is handled by **pic2lnt**:

Q = (new b) (b . Q)

A call to process Q yields an infinite LTS, because each recursive call will generate a new transition labeled by a fresh channel name "**b (n)**" for n >= 0. Therefore, trying to generate the LTS of a call to process Q will exhaust the memory available on the host computer.

SPECIFICATION

A PIC specification consists of a list of process definitions possibly preceded by the declaration of external LOTOS NT modules. The syntax of a PIC specification is defined by the grammar below:

SPEC **::= ["library"** *M0* **"," ..."," ***Mm* **"end" "library"]**
PROC0 **...** *PROCn*

If present, the module identifiers *M0*, ..., *Mm* must correspond to LOTOS NT files named *M0***.lib**, ..., *Mm***.lib** present in the current directory (letter case is not significant). The LOTOS NT modules *M0*, ..., *Mm* must define all the data types and functions used in the process definitions *PROC0*, ..., *PROCn*. If the "library" clause is absent, the PIC specification is dataless, i.e., it contains only polyadic pi-calculus agent definitions.

LABELED TRANSITION SYSTEM

For technical reasons, the translation from PIC to LOTOS NT adds extra information on the actions (transition labels) of the LTS w.r.t. the original early operational semantics of the pi-calculus [MP95,Par01]. This extra information regards two aspects:

- *Channel names.* All channel names occurring in the PIC specification are converted to uppercase. To bring the PIC behaviours to normal form (i.e., to avoid that the same channel name has both free and bound occurrences in the same PIC behaviour), each channel name is postfixed by "_*k*", where *k* is a natural number unique for each channel name. For example, a channel name "a" present in the PIC specification is translated into "A_*k*" for some *k*. Public (resp. private) channel names are encapsulated into a "PUB" (resp. "PRIV") constructor. Moreover, private channel names have a natural number parameter allowing to generate fresh private channel names during the execution of the PIC specification. For example, a public channel name "b" present in the PIC specification will be translated into "PUB (B_*k*)" for some *k*, and a private channel name "c" will be translated into "PRIV (C_*k* (*j*))" for some *k* and *j*.
- *Communication.* All visible actions in the LTS take place on a special channel named "PUBLIC". The first value exchanged on each action is the name of the PIC channel on which the communication takes place in the PIC specification. The other values are those communicated by the PIC action. In addition, there is a last boolean value added on each LTS action to indicate whether the corresponding PIC action is an emission (value TRUE) or a reception (value FALSE). According to the LOTOS NT and LOTOS semantics conventions, all values exchanged on channel "PUBLIC" are preceded by '!'. For example, an emission of the form "'a < b, c, 3 of Nat >", where **a**, **b** are public channels and **c** is a private channel (i.e., defined by a "new" operator), will yield an LTS transition labeled by the following action:

  PUBLIC !PUB (A_0) !PUB (B_0) !PRIV (C_1 (1)) !3 !TRUE

Starting from a PIC specification *filename***.pic**, the corresponding LTS can be produced by applying **pic2lnt** and then generating the LTS *filename***.bcg**, encoded in the BCG file format, of the resulting LOTOS NT specification. This LTS can be made compatible with the original pi-calculus semantics by renaming its actions using **bcg_labels**(LOCAL) and the renaming files present in $PIC/incl as follows:

bcg_labels -rename -multiple $PIC/incl/pic_renaming_1.ren
-rename -multiple $PIC/incl/pic_renaming_2.ren
-rename $PIC/incl/pic_renaming_3.ren
filename`.bcg`

Note: this renaming does not delete the suffixes "*_k*" of the channel names, in order to facilitate the understanding of the PIC behaviours that are not in normal form (i.e., contain both free and bound occurrences of the same channel name(s)).

SYNTACTIC COMPATIBILITY

For compatibility with the Mobility Workbench (MWB) [VM94], the alternative syntax below is also accepted by **pic2lnt** for the polyadic pi-calculus constructs:

pi-calculus construct	Mobility Workbench (MWB) syntax
empty	"0"
silent prefix	"t" "." *B*
emission	"'" *C* "<" *C1 ... Cn* ">" "." *B*
reception	*C* "(" *C1 ... Cn* ")" "." *B*
channel creation	"(" "new" *C0 ... Cn* ")" *B*
	"(" "^" *C0 ... Cn* ")" *B*
	"(" "^" *C0* "," ... "," *Cn* ")" *B*
process (agent) definition	"agent" *P* ["(" *C0* "," ... "," *Cn* ")"] "=" *B*

OPERANDS

filename`.pic`	PIC specification (input)
module`.lib`	LOTOS NT code for data types and functions (input)
filename`.lnt`	LOTOS NT code (output)

ENVIRONMENT VARIABLES

$PIC	Name of the directory where **pic2lnt** is installed.

FILES

$PIC/incl/pic_renaming_{1,2,3}.ren	Renaming files for converting the LTS labels according to the original semantics of the polyadic pi-calculus.
$PIC/incl/pic2lnt_dyn.tnt	Auxiliary file necessary for compiling the LOTOS NT specification *filename*`.lnt` produced as output by **pic2lnt**. To compile *filename*`.lnt` using **lnt2lotos**(LOCAL) or **lnt.open**(LOCAL), the file $PIC/incl/pic2lnt_dyn.tnt should be copied in the current directory and renamed into *filename*`.tnt`.

EXIT STATUS

If the translation was successful the exit status is 0, even if warnings were issued during the execution. If any error occurred during translation, the exit status is 1.

BIBLIOGRAPHY

[Dam94]
M. Dam. "Model Checking Mobile Processes." Research Report RR 94:1, Swedish Institute of Computer Science, Kista, Sweden, 1994.

[MP95] U. Montanari and M. Pistore. "Checking Bisimilarity for Finitary Pi-Calculus." Proceedings of CONCUR'95, LNCS v. 962, p. 42-56, 1995.

[MS10] R. Mateescu and G. Salaun. "Translating Pi-Calculus to LOTOS NT." Proceedings of IFM'10, LNCS v. 6396, p. 229-244, 2010.

[Par01] J. Parrow. "An Introduction to the Pi-Calculus." Handbook of Process Algebra, chap. 8, p. 479-544, North-Holland, 2001.

[VM94] B. Victor and F. Moller. "The Mobility Workbench: A Tool for the Pi-Calculus." Proceedings of CAV'94, LNCS v. 818, p. 428-440, 1994.

[MPW92]
R. Milner, J. Parrow, and D. Walker. "A Calculus of Mobile Processes." Information and Computation 100(1):1-77, 1992.

AUTHORS

Gwen Salaun (Grenoble INP) and Radu Mateescu (INRIA Grenoble - Rhone−Alpes).

SEE ALSO

lnt.open(LOCAL), **bcg**(LOCAL), **bcg_labels**(LOCAL). For a complete description of LOTOS NT, see the **lnt2lotos** reference manual.

BUGS

Please report any mistranslations or other problems with **pic2lnt** to cadp@inria.fr