From LOTOS to LNT

Hubert Garavel

Frédéric Lang   Wendelin Serwe

INRIA Grenoble – LIG

Université Grenoble Alpes

http://convecs.inria.fr

Scope of this talk

Three lines of work in Ed Brinksma's publications:

- between 1984 and 1995
  specification of communication protocols and distributed systems — the LOTOS language
- starting from 1991
  conformance testing for protocols
- starting from 1995
  real time and performance evaluation

INTERNATIONAL STANDARD

ISO 8807

First edition 1989-02-15

Information processing systems — Open Systems Interconnection — LOTOS — A formal description technique based on the temporal ordering of observational behaviour

Systèmes de traitement de l'information — Interconnexion de systèmes ouverts — LOTOS — Technique de description formelle basée sur l'ordonnancement temporel du comportement observationnel

Reference number ISO 8807 : 1989 (E)

LOTOS (1984-1989)

ISO/IEC standard 8807:1989

LOTOS

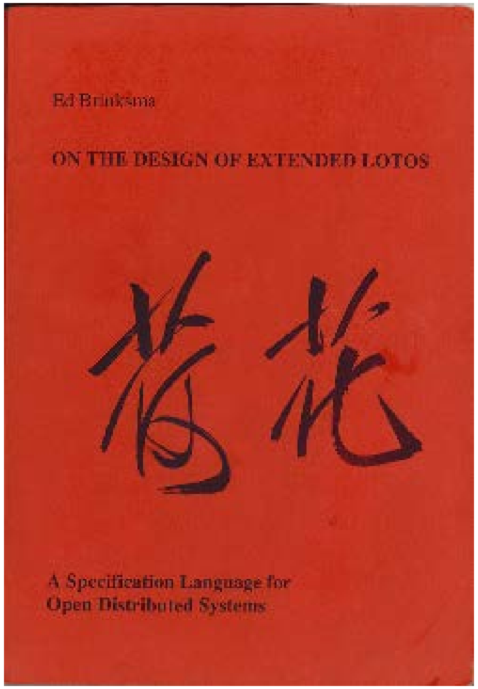
- LOTOS: a language for concurrent systems
  - data structures: abstract data types (ACT-ONE)
  - concurrent processes: process calculi (CCS, CSP, Circal)
  - original operators: ">>" (enable), "[>" (disable)

- Ed Brinskma's key contributions:
  - ISO/IEC standard 8807:1989, edited by Ed Brinksma
  - LOTOS tutorial [Bolognesi-Brinksma-88]
  - constraint-oriented style [Brinksma-89]

    "*parallel composition = conjunction of constraints*"

Assessment of LOTOS

- On the positive side:
  - working compromise between diverse concepts
  - high abstraction level and formal semantics
  - application to complex systems: OSI and ISDN protocols, hardware systems, etc.
  - many projects and tools: SEDOS, LOTOSphere, SPECS, EUCALYPTUS-1 and -2, etc
- On the negative side:
  - LOTOS did not unite the process-algebra community (existing calculi remained, and new calculi arose)
  - LOTOS did not gain wide industrial acceptance (mostly due to its "steep learning curve")
- Ed Brinksma also proposed enhancements to LOTOS



Ed Brinksma

ON THE DESIGN OF EXTENDED LOTOS

A Specification Language for
Open Distributed Systems

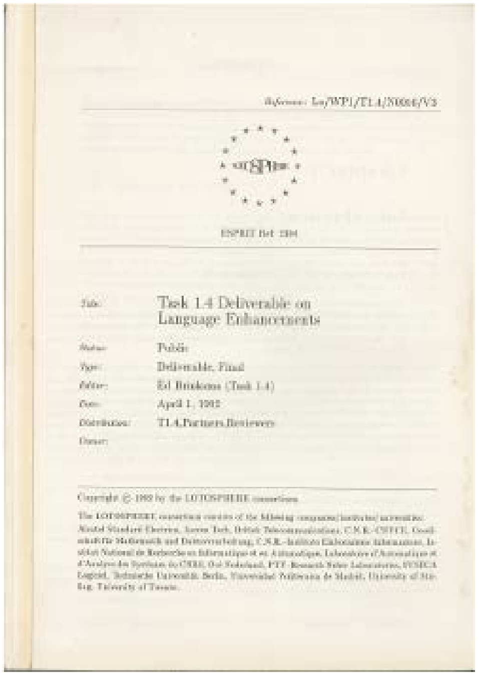
Extended LOTOS (1988)

Extended LOTOS

- Extended LOTOS was the subject of Ed Brinksma's PhD thesis (1988)
- Proposed enhancements to LOTOS, with a focus on the behavioural part:
  - introduction of SCCS-like action product
  - attempt to unify both LOTOS operators for sequential composition (";" and ">>")
  - OCCAM-like n-ary operators with a fully bracketed syntax

    sel B_1 **[ ]** B_2 **[ ]** ... **[ ]** B_n **endsel**

    par B_1 **||** B_2 **||** ... **||** B_n **endpar**
  - **par** operator ranging over a finite domain of values
  - better support for modules



Reference: Lo/WP1/T1.4/N0006/V3

ESPRIT Ref. 2304

Title: Task 1.4 Deliverable on Language Enhancements

Status: Public

Type: Deliverable, Final

Editor: Ed Brinksma (Task 1.4)

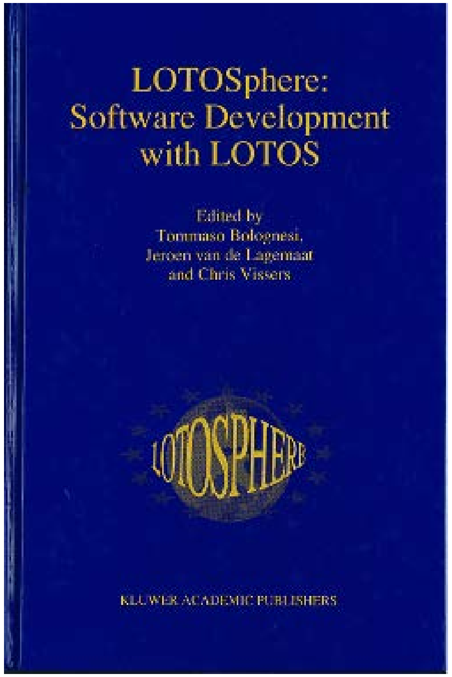
Date: April 1, 1992

Distribution: T1.4,Partners,Reviewers

Owner:

Copyright © 1992 by the LOTOSPHERE consortium

The LOTOSPHERE consortium consists of the following companies/institutes/universities: Alcatel Standard Electrica, Ascom Tech, British Telecommunications, C.N.R.-CNUCE, Gesellschaft für Mathematik und Datenverarbeitung, C.N.R.-Istituto Elaborazione Informazione, Institut National de Recherche en Informatique et en Automatique, Laboratoire d'Automatique et d'Analyse des Systèmes du CNRS, Océ-Nederland, PTT Research Neher Laboratories, SYSECA Logiciel, Technische Universität Berlin, Universidad Politécnica de Madrid, University of Stirling, University of Twente.



LOTOSphere: Software Development with LOTOS

Edited by Tommaso Bolognesi, Jeroen van de Lagemaat and Chris Vissers

KLUWER ACADEMIC PUBLISHERS

Modular LOTOS (1992-1995)

Modular LOTOS

- Modular LOTOS was defined in a LOTOSphere deliverable edited by Ed Brinksma
- Proposed enhancements to the data part of LOTOS:
  - distinction between constructors and functions
  - introduction of partial functions
  - built-in types: natural numbers, integer numbers, strings
  - generic data structures: lists, sets, arrays, etc.
  - module interfaces (called descriptions) for hiding details
  - renaming to avoid name clashes between modules
  - generic modules parameterized by descriptions

E[nhanced]-LOTOS (1993-2001)

ISO/IEC standard 15437:2001

E-LOTOS (Enhanced LOTOS)

- An impressive effort to address LOTOS shortcomings:
  - abstract data types replaced with functional data types
  - imperative style: variable assignment, output parameters
  - language unification: functions being a subset of processes
  - a single sequential composition operator
  - "graphical" parallel composition operator
  - typed communication gates
  - exception handling (e.g., partial functions)
  - quantitative time (delays, timeouts, urgency)
  - new operators: gate renaming, suspend-resume
  - modules, interfaces, combinators, genericity

Assessment of E-LOTOS

- On the positive side:
  - an ambitious evolution of LOTOS and process calculi
  - many inspiring ideas and new language features
- On the negative side:
  - a complex language, with many semantic rules
    - LOTOS standard: 70 pages (+ 70 pages of annexes)
    - E-LOTOS standard: 120 pages (+ 80 pages of annexes)
  - the "steep learning curve" problem remains
  - few case studies done with E-LOTOS
  - no software implementation available

LOTOS NT (1997-now)

LOTOS NT ("New Technology")

- A fallback approach designed at INRIA Grenoble to avoid the ever-growing complexity of E-LOTOS
- LOTOS NT: a simplified version of E-LOTOS
  - no type synonyms
  - no ML-like anonymous tuples
  - no extensible records
  - no structure equivalence for types (name equivalence instead)
  - no subtyping relation based on record subtyping
  - no support for quantitative time
  - no suspend-resume operator
- LOTOS NT influenced the latest evolutions of E-LOTOS

Implementation of LOTOS NT

- TRAIAN: a LOTOS NT → C compiler
  - developed at INRIA Grenoble (10 releases since 1998)
  - 55,000 lines of code (using the SYNTAX/FNC2 compiler generation system based on attribute grammars)
  - translates LOTOS NT types and functions to C ones
  - incomplete: does not handle LOTOS NT processes (since the maintenance of FNC2 stopped in 1999)
- Useful applications for compiler construction
  - idea: SYNTAX + LOTOS NT + very little C code
  - 12 compilers (including CADP tools) written this way

LNT (2005-now)

A brief history of LNT (1/2)

- 2005: request from Bull to replace LOTOS data types
  - mix LOTOS processes with LOTOS NT types/functions
  - design of a translator: LOTOS NT data types → LOTOS (+C)
- The translator was progressively extended to handle LOTOS NT processes as well
  - no need to write processes in LOTOS any more
- At present, a suite of three tools:
  - LPP (LOTOS Pre-Processor): 2000 lines of code (C + Lex)
  - LNT2LOTOS: 42,200 lines (SYNTAX + LOTOS NT + C)
  - LNT.OPEN: 400 lines (Bourne shell)

see Section 7.2 of the paper for details about the translation

A brief history of LNT (2/2)

- 2009: the translator being complete and robust enough, INRIA Grenoble shifted from LOTOS to LOTOS NT
  - no more LOTOS code manually written since then
  - more than 15,000 LOTOS NT specifications so far
- 2010: the translator became part of the CADP toolbox
- 2014: "LOTOS NT" was renamed to "LNT" to avoid ambiguities with the language supported by TRAIAN
- 2015: LNT used for teaching concurrency at University Grenoble Alpes and ENSIMAG engineering school

Two main design challenges

- Combine two programming paradigms in one
  - sequential programming: functional/imperative traits
  - concurrent programming: process calculi

  Most formal languages have stumbled on this difficulty LOTOS, Estelle, SDL, etc.: no unification — just two heterogeneous languages put together

- Design a language for engineers, not for theoreticians
  - reuse existing concepts as much as possible
  - standard notions should be handled in the usual way
  - cf. the idea of "disappearing formal methods"

Overview of LNT constructs

- LNT specification = set of modules
- Each module may contain:
  - types:
    - predefined: **bool**, **nat**, **int**, **real**, **char**, **string**
    - free constructors, including enumerations, records, unions
    - combinators: ranges, arrays, lists, sorted lists, sets, sorted sets, predicate subtypes
  - functions: either mathematical or procedural
    - predefined: arithmetical, logical, relational operators
    - generated automatically for user-defined types
    - handwritten by the user
  - channels: gate types, including **none** and **any**
  - processes: concurrent agents communicating using gates

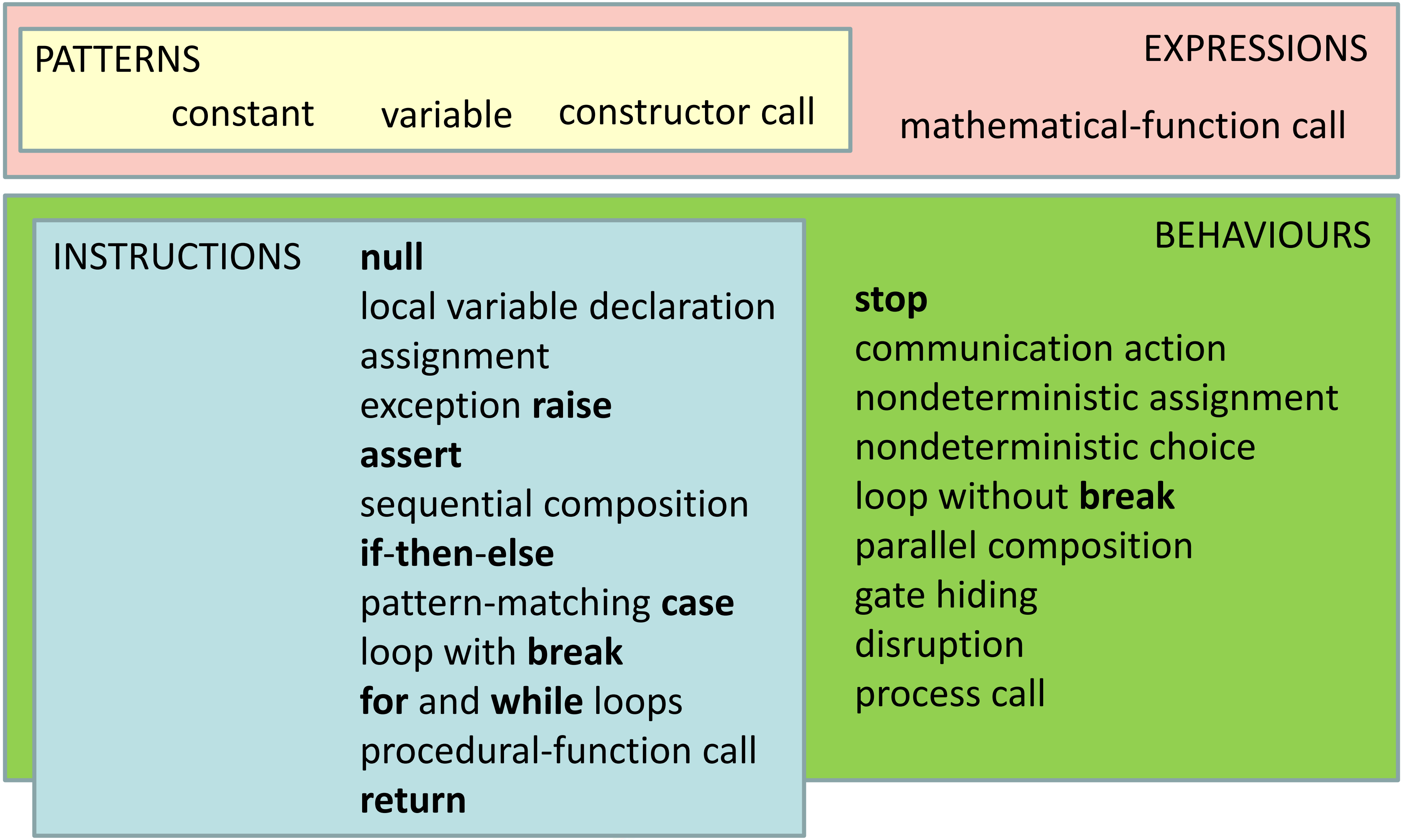
Expressions, instructions, behaviours

Semantics	expressions	instructions	behaviours
Can assign variables?	no	yes	yes
Can send/receive messages?	no	no	yes
Can execute nondeterministically?	no	no	yes
Can execute non-atomically?	no	no	yes
Can never terminate?	no	no	yes

Constructors, functions, processes

Semantics	constructor	mathematical function	procedural function	process
Can have "in" parameters? (i.e., call by value)	yes	yes	yes	yes
Can raise exceptions? (i.e., partial definition)	no	yes	yes	yes
Can have "out" parameters? (i.e., call by result)	no	no	yes	yes
Can have "in out" parameters? (i.e., call by value-result)	no	no	yes	yes
Can return no result? (i.e., have a "void" result)	no	no	yes	yes

A unifying view of LNT



EXPRESSIONS

PATTERNS

constant variable constructor call

mathematical-function call

BEHAVIOURS

INSTRUCTIONS

null
local variable declaration
assignment
exception **raise**
assert
sequential composition
if-then-else
pattern-matching **case**
loop with **break**
for and **while** loops
procedural-function call
return

stop
communication action
nondeterministic assignment
nondeterministic choice
loop without **break**
parallel composition
gate hiding
disruption
process call

Impact of LNT so far

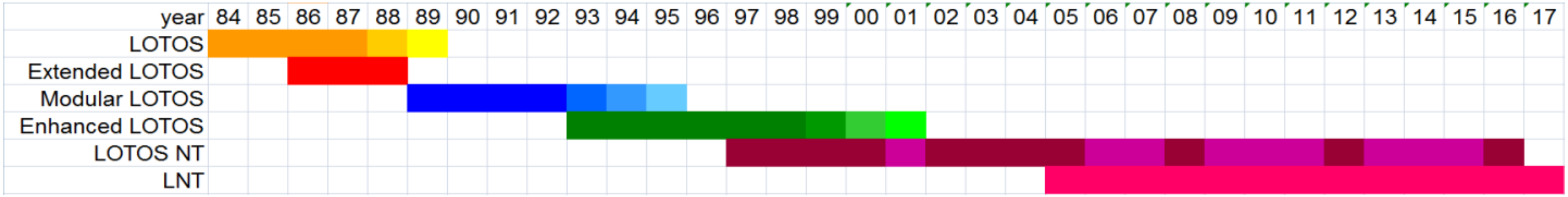
- 17 case studies done with LNT [21 publications]
  - avionics: 2
  - cloud computing: 3
  - distributed algorithms: 4
  - hardware design: 4
  - human/computer interfaces: 2
  - other industrial systems: 2

- 9 translators to LNT [11 publications]

AADL: 1	Toulouse-Sfax
applied π-calculus: 1	Grenoble
BPEL-WSDL: 2	MIT-Tsinghua, Bucharest-Grenoble
BPMN: 2	Nantes, Paris
DFT: 1	Twente
EB3: 1	Paris-Grenoble
GRL: 1	Grenoble

Conclusion

A long-term story…



- Ed Brinksma has set a promising research agenda that has been pursued by others
- After many attempts, there is now a proper replacement language for LOTOS: LNT
- On-going research directions:
  - Extend the LNT language
  - Design a native LNT→C compiler