Note: the electronic PDF-version of this document contains hyperlinks for easy reference.

Proposal for a Formal Methods 2008 Tutorial

Formal Methods and Signal Processing

Raymond Boute, INTEC—Ghent University, boute@intec.UGent.be

Motivation The benefits of combining FM and SP are mutual: (i) bringing mathematical techniques and styles fostered by Formal Methods into Signal Processing; (ii) enhancing the potential of Signal Processing as a vehicle for broadening Formal Methods into a wider engineering discipline. Why Signal Processing is especially well-suited as a target area follows from the following considerations.

(a) Signal processing is traditionally a mathematics-intensive area and mathematical modeling is regular engineering practice. Still, literature abounds with defective formulations as noted and partially corrected by various Signal Processing professionals [7, 8]. Fully defect-free conventions [1, 3, 4] can add a new dimension to mathematical modeling and reasoning by "letting the symbols do the work", which is one of the major benefits of the new mathematical style fostered by Computer Science [6].

(b) Signal processing has broadened from analog models and circuits to including multidimensional information and algorithms implemented in software. Unfortunately, whereas discrete models for signals and systems are well-known (since they originated in EE), those for algorithms and software (which originated in CS, e.g., formal semantics, data type theory) remain largely unexploited in the SP community. A unifying formalism that captures the commonalities (while respecting the differences) can bridge this gap.

(c) As the importance of SP continues growing, computing professionals and software engineers will be increasingly involved in SP-related projects, which requires a suitable background about signals and systems and makes Parnas's observations from nearly two decades ago [9] even more relevant today. Establishing this background in a Formal Methods setting provides significant synergy.

Objectives and contents A formalism (*Functional Mathematics*) is presented that avoids all defects of the traditional conventions while still maintaining the familiar "look and feel" of the formulas [1]. It supports the same calculational style of reasoning in differential/integral calculus, formal logic [4], theories of signals and systems [3], and programs [5]. The two main components are:

(i) an algebra of generic functionals [2], originally designed for structuring signals and systems, but which has turned out equally useful in formal logic and modeling software;

(ii) a functional predicate calculus [4] that allows engineers to calculate as fluently with quantifiers ($\forall$, $\exists$) as they have learned to do with derivatives and integrals, and hence opens the door to modeling programs mathematically. The latter is done via program equations with the same "flavor" as circuit equations, and with functionals relating them to specifications in a way that has an appealing formal analogy with transform methods.

Outline of the lectures What follows is just a list; a first taste of the approach, style and content can be obtained by downloading [3] via the web.

0. Motivation: need for defect-free conventions, benefits in calculation.
1. Background review: equality (Leibniz's principle), basic proposition calculus, sets, conditional expressions.

2. Functions: specifications, styles of expression (pointwise, point-free), tuples, function abstraction.
3. Functionals for structuring signals and systems. Generic design of functionals. State and signal flow models. Functional characterization of systems characteristics (memory, causality, time invariance, linearity etc.)
4. Functional predicate calculus, quantifiers ($\forall$, $\exists$) as functionals, calculation rules.
5. Using predicate calculus in specifying function(als) for and reasoning about signals and systems.
6. From recursive sequence definitions to signal flow expressions, application to signal flow languages.
7. Using predicate logic in classical calculus; casting transforms (Fourier, Laplace) in functional style.
8. Brief intermezzo about syntactic functions salvaging ill-founded yet potentially very convenient notational conventions.
9. Modeling programs by equations. Specification by ante-and postconditions.
10. Specifying temporal behavior by temporal operators.
11. Concluding remarks: ramifications; remarks on automated support.

Organizational aspects

Intended audience Students, practitioners, researchers in Formal Methods with interest in Signal Processing and, conversely, SP-oriented people interested in FM.

Prerequisites Interest in Formal Methods in the wider sense, namely, as a unifying approach to mathematical modeling in classical and computer engineering. Basic notions of signal processing are an asset but not required.

Duration half day (3 * 50 min); full day (6 * 50 min) possible if preferred by the workshop organizers.

Equipment Video projector, PC with USB port. If no PC is available, the lecturer uses a laptop.

Earlier tutorial experience Related tutorials (for a CS audience) presented at:
SEFM2003 - Intl. Conf. on Software Engineering and Formal Methods (Brisbane, Aus)
ICTAC2004 - Intl. Colloq. on Theoretical Aspects of Computing (Guiyang, China)
WCC2004 - World Computer Congress (Toulouse, France)
FM2005 - Formal Methods Conference (Newcastle upon Tyne, UK)
First presentations to a SP audience: CHESS-seminar (2004, Berkeley) SPS-DARTS 2005.

Biography of the lecturer Raymond Boute holds MSc degrees in engineering from Ghent University, Belgium (1966, 1968) and a MSc and a PhD degree in EE/CS from Stanford University, USA (1969, 1973). From 1974 to 1981 he did research at the Bell Telephone Mfg. Cy. (Belgium) on systems architectures. In 1981 he became a full professor of Computer Science at Nijmegen University (The Netherlands) teaching digital design, operating systems, networks and functional languages. His research on functional languages gradually expanded to unifying mathematical methods for classical and computer engineering. In 1994 Boute became a full professor at Ghent University (Belgium), continuing this work in teaching and research (http://www.funmath.be). He is a member of the IEEE, ACM, Sigma Xi, and various other professional and cultural associations.

References

[1] Raymond Boute, *Formal Systems Modeling*. Course notes, Ghent University (1994)

[2] Raymond Boute, "Concrete Generic Functionals: Principles, Design and Applications", in: Jeremy Gibbons, Johan Jeuring, eds., *Generic Programming*, pp. 89–119, Kluwer (2003)

[3] Raymond Boute, "Signal processing functions, algorithms and smurfs: the need for declarativity", in: Wilfried Philips, ed., *Proc. SPS-DARTS 2005 Signal Processing Symposium*, pp. 109–114 (Apr. 2005)

[4] Raymond Boute, "Functional declarative language design and predicate calculus: a practical approach", *ACM Trans. Progr. Lang. and Syst. 27*, 5, pp. 988–1047 (Sep. 2005)

[5] Raymond Boute, "Calculational semantics: deriving programming theories from equations by functional predicate calculus", *ACM Trans. Prog. Lang. Syst. 28*, 4, pp. 747–793 (Jul. 2006)

[6] Edsger W. Dijkstra, "How Computing Science created a new mathematical style", *EWD 1073* (1990) Web: `http://www.cs.utexas.edu/users/EWD/ewd10xx/EWD1073.PDF`

[7] Edward A. Lee and Pravin Varaiya, "Introducing Signals and Systems — the Berkeley Approach". *First Signal Processing Education Workshop*, (Oct. 2000). Web: `http://ptolemy.eecs.berkeley.edu/publications/papers/00/spe1/`

[8] Edward A. Lee and Pravin Varaiya, *Structure and Interpretation of Signals and Systems*. Addison-Wesley (2003)

[9] David L. Parnas, "Education for Computing Professionals", *IEEE Computer 23*, 1, pp. 17–20 (January 1990)