

Lightweight Formal Methods for the Development of High-Assurance Networking Systems

(Extended Abstract for an Invited Talk at VECoS 2009)

Assaf Kfoury
Computer Science Department
Boston University

Overview. In recent years, concepts and techniques rooted in formal methods of computer science and, more generally, mathematical logic have acquired an increasingly important role in systems research. One particular area that has witnessed a blossoming of formal approaches to specification, analysis, and verification, is that of networking systems. The **iBench** [2] and **snBench** [6] projects, both based in the Computer Science Department of Boston University, pursue several research activities in this direction. The following is a recent sample, listing the topic and the main collaborators on each:

1. *Compositional Analysis/Specification and its Benefits*
(mostly A. Bestavros, A. Kfoury, and M. Ocean).
2. *An Application of Model Checking: Safe Composition of Arbitrary Network Protocols*
(mostly A. Bradley, A. Bestavros, and A. Kfoury).
3. *Resource Allocation in Sensor Networks using a Strongly-Typed Domain-Specific Language*
(mostly A. Bestavros, A. Kfoury, and M. Ocean).
4. *The Stable-Paths Problem and the Promise of an Automatic Lightweight Proof-Assistant*
(K. Donnelly, A. Kfoury, and A. Lapets).

Small as it is, the preceding sample is illustrative of a far wider area, with many approaches pursued by many researchers around the world, to promote the use of formal methods in networking systems research.

In this talk I will focus on **topic 1**, explaining the formal methodology we are currently developing, in the context of the **iBench** project, for what we call *compositional analysis* of networking systems. Some preliminary reports can be downloaded from the **iBench** publications page [3].

For the three last topics in my sample, I refer to our published reports and articles. For **topic 2**, search the **iBench** publications page [3] for articles containing the keyword “CHAIN” (Canonical Homomorphic Abstraction of Infinite Network) with Adam Bradley as co-author. For **topic 3**, search the **snBench** homepage [6] for reports which include Michael Ocean among its co-authors. For **topic 4**, preliminary results are reported in [1, 5, 4].

What Is Compositional Analysis? Many networking problems naturally lend themselves to graph-theoretic formulations that are far more complex than classical problems of graph theory. The modeling graphs are typically large, with hundreds or thousands or more nodes, often placed on a fluctuating grid, e.g., paths between nodes may fail or degrade or reappear, slowly or abruptly, and parameters regulating flow along paths may be conflicting requirements or defined differently at different nodes, locally or globally.

We propose a methodology for the specification and analysis of networking systems which attempts to support such uneven features in the large graphs modeling them. The proposed methodology would allow for a *compositional* (as opposed to *whole-system*) analysis which is additionally incremental (distributed in time) and modular

(distributed in space). Schematically, we can contrast *whole-system* and *compositional* analyses as follows:

the modules of a network	whole-system analysis	=	compositional analysis
A	$\llbracket A \rrbracket$	=	$\llbracket A \rrbracket$
$A \otimes B$	$\llbracket A \otimes B \rrbracket$	=	$\llbracket A \rrbracket \star \llbracket B \rrbracket$
$A \otimes B \otimes C$	$\llbracket A \otimes B \otimes C \rrbracket$	=	$\llbracket A \rrbracket \star \llbracket B \rrbracket \star \llbracket C \rrbracket$
$A \otimes \langle \rangle \otimes C$	$\llbracket A \otimes \langle \rangle \otimes C \rrbracket ??$	$\stackrel{?}{=}$	$\llbracket A \rrbracket \star \llbracket \langle \rangle \rrbracket \star \llbracket C \rrbracket$
$A \otimes B' \otimes C$	$\llbracket A \otimes B' \otimes C \rrbracket$	=	$\llbracket A \rrbracket \star \llbracket B' \rrbracket \star \llbracket C \rrbracket$
...

where “ $\llbracket x \rrbracket$ ” denotes “the analysis of x ”, “ \otimes ” an associative operation for connecting two modules, and “ \star ” an associative operation for combining two analyses.

By its nature, a whole-system analysis cannot be undertaken if a module (such as B above) is missing or if it breaks down (indicated by the double question marks “ $??$ ”). Moreover, if the missing module is scheduled to be replaced by a new module (B' above), a whole-system analysis must be delayed until the new module becomes available for examination and then the entire network must be re-analyzed from scratch. If we are interested in certifying that a particular invariant is preserved throughout the network without running into the limitations of a whole-system analysis – inability to deal with incomplete or fluctuating topologies, or incurring the cost of having to re-examine the entire network – and if we can formalize this invariant using type-theoretic notions at module interfaces (denoted by a “ $\langle \rangle$ ” above), then we can adopt the alternative approach of a compositional analysis: Our envisioned compositional analysis will not be invalidated by the presence of holes (the empty envelopes or interfaces, “ $\langle \rangle$ ” above) where we can place different modules satisfying the same interface types, interchangeably and at different times.

Our schema above, contrasting compositional and whole-system analyses, calls for several additional provisos, if we are to reap all the benefits of the former. We mention only one such proviso, namely, that the cost of combining two analyses via “ \star ” (this is compositional analysis) should be far smaller – specifically, below a computational complexity that is acceptable to the user – than the cost of combining two networks via “ \otimes ” and then analyzing the combination (this is whole-system analysis).

In addition to concepts borrowed from formal methods of computer science and, more generally, mathematical logic, our methodology calls for the definition of a strongly-typed *domain-specific language* (DSL) to specify network configurations and the invariants we wish to enforce. We pay special attention to keeping these concepts and techniques lightweight. i.e., making the parts available to users “friendly” (relatively simple and easy to use) while the more complicated parts remain hidden “under the hood”.

Traffic networks are ideally suited as a test bed for such a methodology. There is a wide range of traffic networks (Internet traffic, air traffic, railroad traffic, vehicular traffic, etc.), each with its own specific characteristics. For definiteness, we choose vehicular traffic networks.

Applying Compositional Analysis to Networks of Vehicular Traffic. If \mathcal{M} and \mathcal{N} are vehicular traffic networks, and $\mathcal{M} : (I_1, O_1)$ and $\mathcal{N} : (I_2, O_2)$ are typings of \mathcal{M} and \mathcal{N} assigning appropriately defined types to their input (entering) and output (exiting) links, then the formal syntax of our strongly-typed DSL is defined by rules of the form:

$$\begin{array}{l}
 \text{CONNECT} \quad \frac{\Gamma \vdash \mathcal{M} : (I_1, O_1) \quad \Gamma \vdash \mathcal{N} : (I_2, O_2)}{\Gamma \vdash \mathbf{conn}_{\theta, m, n}(\mathcal{M}, \mathcal{N}) : (I, O)} \\
 \text{parameters}(\mathcal{M}) \cap \text{parameters}(\mathcal{N}) = \emptyset \\
 m = \text{length}(O_1), \quad n = \text{length}(I_2), \quad \theta \subseteq_{1-1} \{1, \dots, m\} \times \{1, \dots, n\} \\
 I = I_1 \cdot (I_2 / \text{range}(\theta)), \quad O = (O_1 / \text{domain}(\theta)) \cdot O_2, \quad [\text{Typ}(O_1)]_p <: [\text{Typ}(I_2)]_q \text{ for every } (p, q) \in \theta
 \end{array}$$

Without delving into the details of the rule `CONNECT`, it formalizes the idea that, if we want to safely connect the p -th exiting link of \mathcal{M} to the q -th entering link of \mathcal{N} , then the type of the exiting link must be a subtype of the type of the entering link. Another rule of our DSL is of the form:

$$\text{LET} \quad \frac{\Gamma \vdash \mathcal{M}_1 : (I_1, O_1) \cdots \Gamma \vdash \mathcal{M}_n : (I_n, O_n) \quad \Gamma, X : (I, O) \vdash \mathcal{N} : (I', O')}{\Gamma \vdash \mathbf{let} X \in \{\mathcal{M}_1, \dots, \mathcal{M}_n\} \mathbf{in} \mathcal{N} : (I', O')}$$

for every $1 \leq m \leq n$, $\text{parameters}(\mathcal{M}_m) \cap \text{parameters}(\mathcal{N}) = \emptyset$
for every $1 \leq m \leq n$, $\text{length}(I_m) = \text{length}(I)$ and $\text{length}(O_m) = \text{length}(O)$
 $\text{Typ}(I_1) = \dots = \text{Typ}(I_n) = \text{Typ}(I)$ and $\text{Typ}(O_1) = \dots = \text{Typ}(O_n) = \text{Typ}(O)$

The rule `LET` formalizes the idea that, in a hole X of a network \mathcal{N} , we can place at will any of n different networks $\{\mathcal{M}_1, \dots, \mathcal{M}_n\}$ that satisfy the same interface types.

There are various refinements of the preceding two rules, as well as several other rules (omitted here), that define the syntax of typed specifications for networks of vehicular traffic. The types for this application are *velocity types* and *density types*, which are both formalized as non-empty intervals over the natural numbers. They are inferred for a network by starting from the constraints regulating traffic at each of the nodes in the network. Constraints are equalities and inequalities between polynomial expressions over velocity and density parameters.

With such a DSL, we can enforce various desirable invariants across a traffic network, such as *fairness* (there is no link along which traffic is permanently prevented from moving), *conservation of flow* (traffic flow entering the network is equal to exiting traffic flow), *gridlock-free* (mutually conflicting traffics along some of the links ultimately result in blocking traffics along all links), etc.

Presentation. Slides for the presentation can be downloaded from the **iBench** publications webpage [3] or the **snBench** homepage [6]. The slides are in three parts and include details considerably expanding the lecture at VeCOS 2009: Part 1 (slides 1-49), Part 2 (slides 50-86), and Part 3 (slides 87-111).

References

- [1] Kevin Donnelly and Assaf Kfoury. On the stable paths problem and a restricted variant. Technical Report BUCS-TR-2008-001, CS Department, Boston University, January 10 2008.
- [2] iBench Initiative Homepage. <http://www.cs.bu.edu/groups/ibench/>.
- [3] iBench Publications Page. <http://www.cs.bu.edu/groups/ibench/bib/>.
- [4] Usability in Automated Formal Verification. http://safre.net/wiki/index.php?title=restricted_stable_paths_problem.
- [5] Andrei Lapets. Improving the accessibility of lightweight formal verification systems. Technical Report BUCS-TR-2009-015, CS Department, Boston University, April 30 2009.
- [6] snBench: The Sensor Network Workbench. <http://csr.bu.edu/snbench/>.