Nondeterministic Asynchronous Dataflow in Isabelle/HOL

Rafael Castro G. Silva, Laouen Fernet and Dmitriy Traytel

rasi@di.ku.dk

Department of Computer Science
University of Copenhagen

14/05/2025

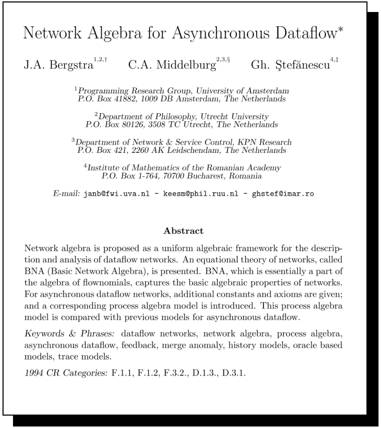
Introduction

Introduction

- Nondeterministic Asynchronous Dataflow: Directed graph of interconnected operators that perform event-wise transformations
  - Dataflow: Directed graph of interconnected operators that perform event-wise transformations
  - Asynchronous:
    - Operators execute independently (processes without an orchestrator)
    - Operators can communicate with the network (read/write); do silent computation steps
    - Networks are unbounded FIFO queues
  - Nondeterministic: Operators can make nondeterministic choices

The Process Algebra Model

- Nondeterministic Asynchronous Dataflow defined as an algebra by Bergstra et al.:



Network Algebra for Asynchronous Dataflow*

J.A. Bergstra[1,2,†] C.A. Middelburg[2,3,§] Gh. Ştefănescu[4,‡]

[1]Programming Research Group, University of Amsterdam
P.O. Box 41882, 1009 DB Amsterdam, The Netherlands

[2]Department of Philosophy, Utrecht University
P.O. Box 80126, 3508 TC Utrecht, The Netherlands

[3]Department of Network & Service Control, KPN Research
P.O. Box 421, 2260 AK Leidschendam, The Netherlands

[4]Institute of Mathematics of the Romanian Academy
P.O. Box 1-764, 70700 Bucharest, Romania

E-mail: `janb@fwi.uva.nl` - `keesm@phil.ruu.nl` - `ghstef@imar.ro`

Abstract

Network algebra is proposed as a uniform algebraic framework for the description and analysis of dataflow networks. An equational theory of networks, called BNA (Basic Network Algebra), is presented. BNA, which is essentially a part of the algebra of flownomials, captures the basic algebraic properties of networks. For asynchronous dataflow networks, additional constants and axioms are given; and a corresponding process algebra model is introduced. This process algebra model is compared with previous models for asynchronous dataflow.

Keywords & Phrases: dataflow networks, network algebra, process algebra, asynchronous dataflow, feedback, merge anomaly, history models, oracle based models, trace models.

1994 CR Categories: F.1.1, F.1.2, F.3.2., D.1.3., D.3.1.

- Provides a process algebra model (a process calculi) of Nondeterministic Asynchronous Dataflow
- Defined basic primitives: sequential and parallel composition; feedback loop
- Defined basic primitives: sequential and parallel composition; feedback loop

Motivation

- Stream Processing: programs that compute (possibly) unbounded sequences of data (streams)
- E.g.: Apache Flink, Apache Samza, Apache Spark, Google Cloud Dataflow, and Timely Dataflow

- Why?
    - Highly Parallel
    - Low latency (output as soon as possible)
    - Incremental computing (re-uses previous computations)

Isabelle/HOL Preliminaries

Isabelle/HOL

- Classical higher-order logic (HOL): Simple Typed Lambda Calculus + (Hilbert) axiom of choice + axiom of infinity + rank-1 polymorphism

Isabelle/HOL

- Classical higher-order logic (HOL): Simple Typed Lambda Calculus + (Hilbert) axiom of choice + axiom of infinity + rank-1 polymorphism
- Isabelle: A generic proof assistant

- Isabelle/HOL: Isabelle's flavor of HOL

Isabelle/HOL: Codatatypes

- Codatatypes

Isabelle/HOL: Corecursion

- Recursion: always eventually reduces an argument
- Corecursion: always eventually produces something

Isabelle/HOL: Corecursion

- Recursion: always eventually reduces an argument
- Corecursion: always eventually produces something
- Corec:

Isabelle/HOL: (Co)inductive Predicates

- Inductive predicate
  - Finite number of introduction rule applications

Isabelle/HOL: (Co)inductive Predicates

- Inductive predicate
  - Finite number of introduction rule applications
- Coinductive predicate
  - Infinite number of introduction rule applications

Isabelle/HOL: (Co)inductive Predicates

- Inductive predicate
  - Finite number of introduction rule applications
- Coinductive predicate
  - Infinite number of introduction rule applications
- Coinduction principle

Operators as a Codatatype

Operators

- Codatatypes

Examples

- Codatatypes

Operators Equivalences: Motivation

- foo

Operators Equivalences: Strong Bisimilarity

- foo

Operators Equivalences: Weak Bisimilarity

- foo

Operators Equivalences: Trace Equivalence

- foo

Numeral Types for Ports

- foo

Asynchronous Dataflow Operators

Buffer Infrastructure

- foo

Asynchronous Dataflow Properties

Conclusion

Conclusion

- Isabelle/HOL has a good tool set to formalize and reason about stream processing:
  - Codatatypes, coinductive predicates, corecursion with friends, reasoning up to friends (congruence),
    - Coinduction up to congruence principle is automatically derived for codatatypes (but not for coinductive principles)
- Next step: Feedback loop

Questions, comments and suggestions