Interactive verification of Lustre programs in Vélus

Timothy Bourke **Paul Jeanmaire** Marc Pouzet

ENS, Inria Parkas team

Synchron'21, November 26

Interactive verification of synchronous programs

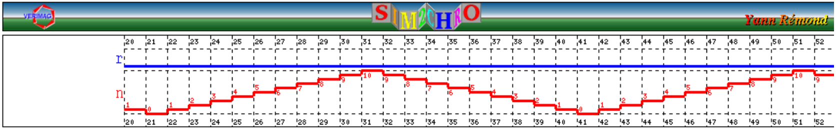
Goal: prove properties of streams in a program

```
node f (r : bool) returns (n : int)
var up : bool;
let
  up = true fby ((up and n < 9) or (not up and n <= 1));
  n = 0 fby (if up then n + 1 else n - 1);
tel
```

Interactive verification of synchronous programs

Goal: prove properties of streams in a program

```
node f (r : bool) returns (n : int)
var up : bool;
let
  up = true fby ((up and n < 9) or (not up and n <= 1));
  n = 0 fby (if up then n + 1 else n - 1);
tel
```

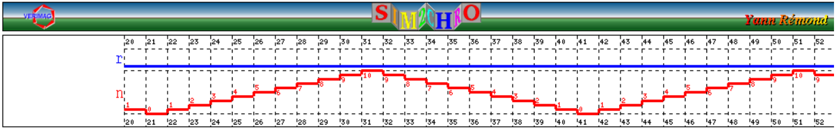


Interactive verification of synchronous programs

Goal: prove properties of streams in a program

```
node f (r : bool) returns (n : int)
var up : bool;
let
  up = true fby ((up and n < 9) or (not up and n <= 1));
  n = 0 fby (if up then n + 1 else n - 1);
tel
```

Show that values of stream `n` stay between 0 and 10



Interactive verification of synchronous programs

Goal: prove properties of streams in a program

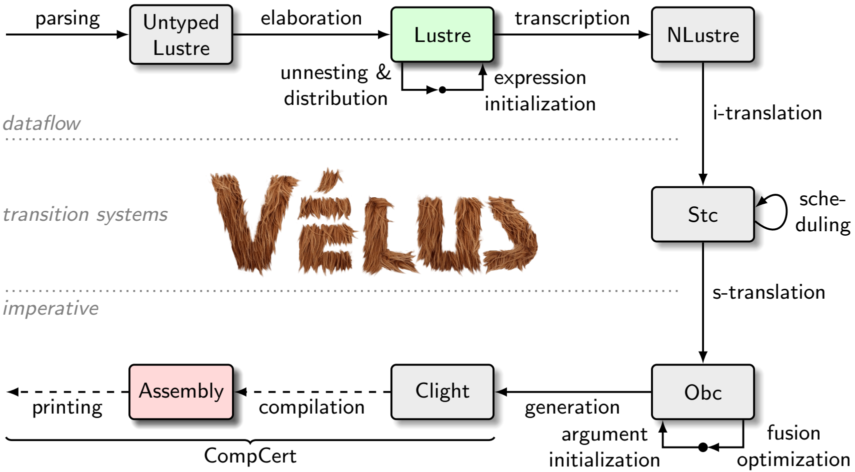
```
node f (r : bool) returns (n : int)
var up : bool;
let
  up = true fby ((up and n < 9) or (not up and n <= 1));
  n = 0 fby (if up then n + 1 else n - 1);
tel
```

Show that values of stream `n` stay between 0 and 10

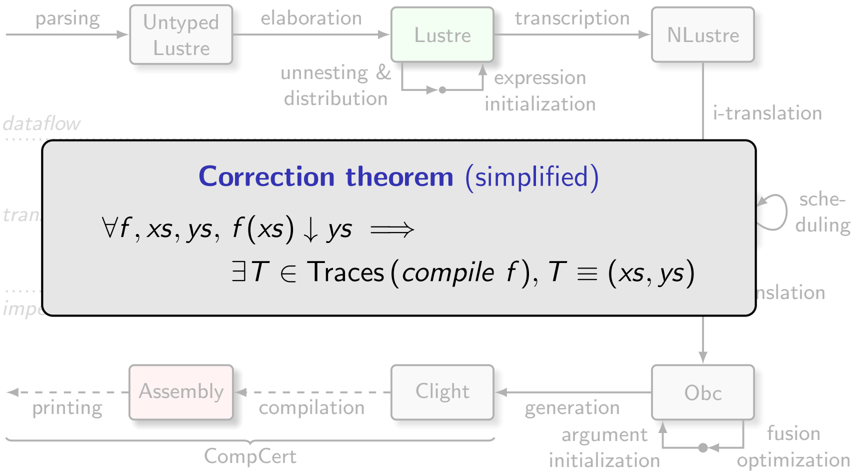
Interactive approach:

- load definitions in the proof assistant (ITP)
- use reasoning techniques to manipulate the goal/hypotheses
- obtain a mathematical proof of the result

Interactive verification in Vélus



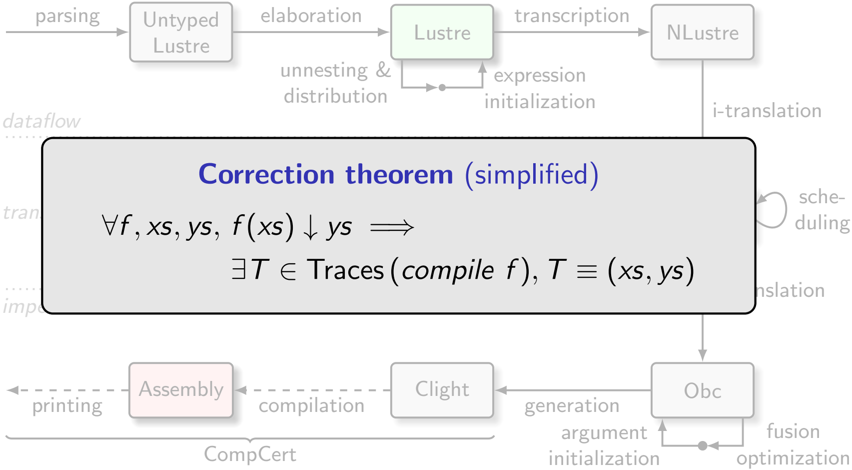
Interactive verification in Vélus



Correction theorem (simplified)

$$\forall f, xs, ys,\ f(xs) \downarrow ys \implies$$
$$\exists T \in \text{Traces}\,(\mathit{compile}\ f),\ T \equiv (xs, ys)$$

Interactive verification in Vélus



Correction theorem (simplified)

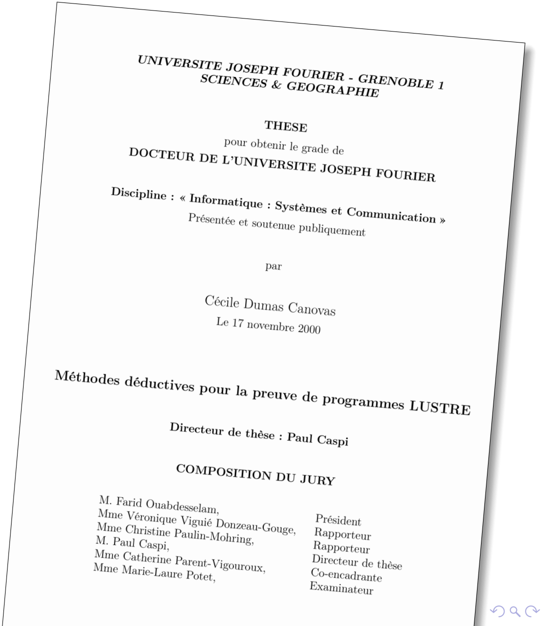
$$\forall f, xs, ys,\ f(xs) \downarrow ys \implies \exists T \in \mathrm{Traces}\,(compile\ f),\ T \equiv (xs, ys)$$

Using previous works

Cécile Dumas Canovas

Deductive methods for proof of Lustre programs

PhD thesis, 2000



UNIVERSITE JOSEPH FOURIER - GRENOBLE 1
SCIENCES & GEOGRAPHIE

THESE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITE JOSEPH FOURIER

Discipline : « Informatique : Systèmes et Communication »

Présentée et soutenue publiquement

par

Cécile Dumas Canovas

Le 17 novembre 2000

Méthodes déductives pour la preuve de programmes LUSTRE

Directeur de thèse : Paul Caspi

COMPOSITION DU JURY

M. Farid Ouabdesselam,	Président
Mme Véronique Viguié Donzeau-Gouge,	Rapporteur
Mme Christine Paulin-Mohring,	Rapporteur
M. Paul Caspi,	Directeur de thèse
Mme Catherine Parent-Vigouroux,	Co-encadrante
Mme Marie-Laure Potet,	Examinateur

Using previous works

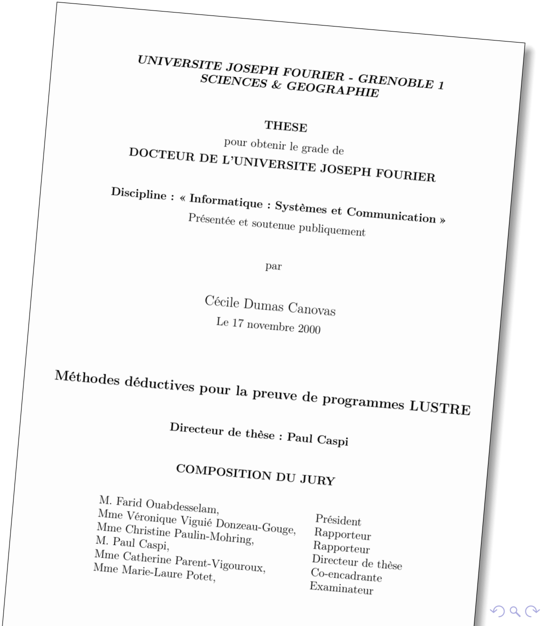
Cécile Dumas Canovas

Deductive methods for proof of Lustre programs

PhD thesis, 2000

Contents

- Example of program verification (train)
- Translation from Lustre to Atelier B
- Proof principles for Lustre
- From Lustre to PVS
- Thoughts on refinement



UNIVERSITE JOSEPH FOURIER - GRENOBLE 1
SCIENCES & GEOGRAPHIE

THESE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITE JOSEPH FOURIER

Discipline : « Informatique : Systèmes et Communication »

Présentée et soutenue publiquement

par

Cécile Dumas Canovas

Le 17 novembre 2000

Méthodes déductives pour la preuve de programmes LUSTRE

Directeur de thèse : Paul Caspi

COMPOSITION DU JURY

M. Farid Ouabdesselam,	Président
Mme Véronique Viguié Donzeau-Gouge,	Rapporteur
Mme Christine Paulin-Mohring,	Rapporteur
M. Paul Caspi,	Directeur de thèse
Mme Catherine Parent-Vigouroux,	Co-encadrante
Mme Marie-Laure Potet,	Examinateur

Using previous works

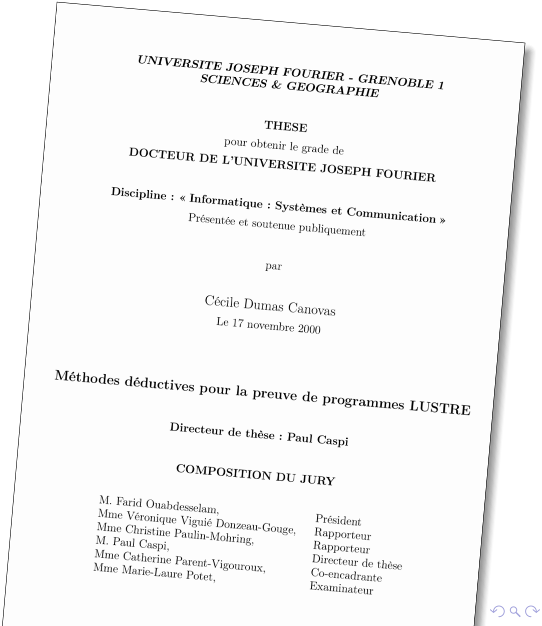
Cécile Dumas Canovas

Deductive methods for proof of Lustre programs

PhD thesis, 2000

Contents

- Example of program verification (train)
- Translation from Lustre to Atelier B
- **Proof principles for Lustre**
- From Lustre to PVS
- Thoughts on refinement



UNIVERSITE JOSEPH FOURIER - GRENOBLE 1
SCIENCES & GEOGRAPHIE

THESE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITE JOSEPH FOURIER

Discipline : « Informatique : Systèmes et Communication »

Présentée et soutenue publiquement

par

Cécile Dumas Canovas

Le 17 novembre 2000

Méthodes déductives pour la preuve de programmes LUSTRE

Directeur de thèse : Paul Caspi

COMPOSITION DU JURY

M. Farid Ouabdesselam,	Président
Mme Véronique Viguié Donzeau-Gouge,	Rapporteur
Mme Christine Paulin-Mohring,	Rapporteur
M. Paul Caspi,	Directeur de thèse
Mme Catherine Parent-Vigouroux,	Co-encadrante
Mme Marie-Laure Potet,	Examinateur

Three different encodings of streams

1. Temporal & consecutive representations: $\mathbb{N} \to \mathsf{Val}$

Three different encodings of streams

1. Temporal & consecutive representations: $\mathbb{N} \to \mathsf{Val}$
2. Co-induction with infinite proof principle ($\approx$ Vélus)

Three different encodings of streams

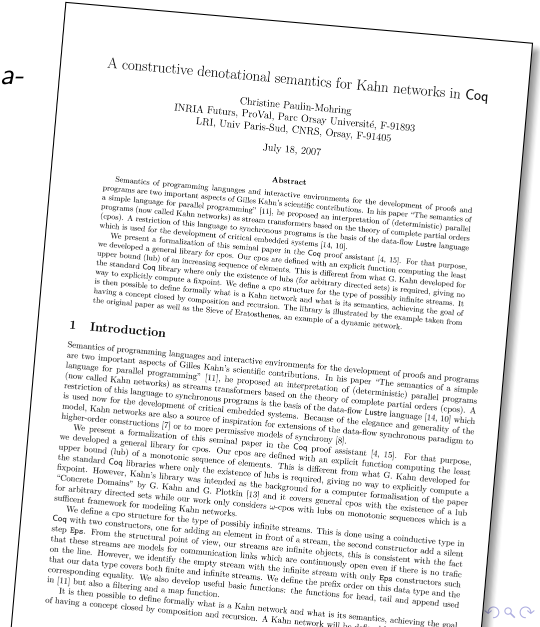
1. Temporal & consecutive representations: $\mathbb{N} \to \mathsf{Val}$
2. Co-induction with infinite proof principle ($\approx$ Vélus)
3. **Kahn streams** with a principle of *continuous induction*
   - the set of streams is a CPO
   - with a smallest element ϵ
   - stream operations are defined as least fix-points

Using previous works

Christine Paulin-Mohring

A constructive denotational semantics for Kahn networks in Coq

From semantics to CS, 2007



A constructive denotational semantics for Kahn networks in Coq

Christine Paulin-Mohring
INRIA Futurs, ProVal, Parc Orsay Université, F-91893
LRI, Univ Paris-Sud, CNRS, Orsay, F-91405

July 18, 2007

Abstract

Semantics of programming languages and interactive environments for the development of proofs and programs are two important aspects of Gilles Kahn's scientific contributions. In his paper "The semantics of a simple language for parallel programming" [11], he proposed an interpretation of (deterministic) parallel programs (now called Kahn networks) as stream transformers based on the theory of complete partial orders (cpos). A restriction of this language to synchronous programs is the basis of the data-flow Lustre language which is used for the development of critical embedded systems [14, 10].

We present a formalization of this seminal paper in the Coq proof assistant [4, 15]. For that purpose, we developed a general library for cpos. Our cpos are defined with an explicit function computing the least upper bound (lub) of an increasing sequence of elements. This is different from what G. Kahn developed for the standard Coq library where only the existence of lubs (for arbitrary directed sets) is required, giving no way to explicitly compute a fixpoint. We define a cpo structure for the type of possibly infinite streams. It is then possible to define formally what is a Kahn network and what is its semantics, achieving the goal of having a concept closed by composition and recursion. The library is illustrated by the example taken from the original paper as well as the Sieve of Eratosthenes, an example of a dynamic network.

1 Introduction

Semantics of programming languages and interactive environments for the development of proofs and programs are two important aspects of Gilles Kahn's scientific contributions. In his paper "The semantics of a simple language for parallel programming" [11], he proposed an interpretation of (deterministic) parallel programs (now called Kahn networks) as streams transformers based on the theory of complete partial orders (cpos). A restriction of this language to synchronous programs is the basis of the data-flow Lustre language [14, 10] which is used now for the development of critical embedded systems. Because of the elegance and generality of the model, Kahn networks are also a source of inspiration for extensions of the data-flow synchronous paradigm to higher-order constructions [7] or to more permissive models of synchrony [8].

We present a formalization of this seminal paper in the Coq proof assistant [4, 15]. For that purpose, we developed a general library for cpos. Our cpos are defined with an explicit function computing the least upper bound (lub) of a monotonic sequence of elements. This is different from what G. Kahn developed for the standard Coq libraries where only the existence of lubs is required, giving no way to explicitly compute a fixpoint. However, Kahn's library was intended as the background for a computer formalisation of the paper "Concrete Domains" by G. Kahn and G. Plotkin [13] and it covers general cpos with the existence of a lub for arbitrary directed sets while our work only considers ω-cpos with lubs on monotonic sequences which is a sufficent framework for modeling Kahn networks.

We define a cpo structure for the type of possibly infinite streams. This is done using a coinductive type in Coq with two constructors, one for adding an element in front of a stream, the second constructor add a silent step Eps. From the structural point of view, our streams are infinite objects, this is consistent with the fact that these streams are models for communication links which are continuously open even if there is no trafic on the line. However, we identify the empty stream with the infinite stream with only Eps constructors such that our data type covers both finite and infinite streams. We define the prefix order on this data type and the corresponding equality. We also develop useful basic functions: the functions for head, tail and append used in [11] but also a filtering and a map function.

It is then possible to define formally what is a Kahn network and what is its semantics, achieving the goal of having a concept closed by composition and recursion. A Kahn network will be defined ...

Using previous works

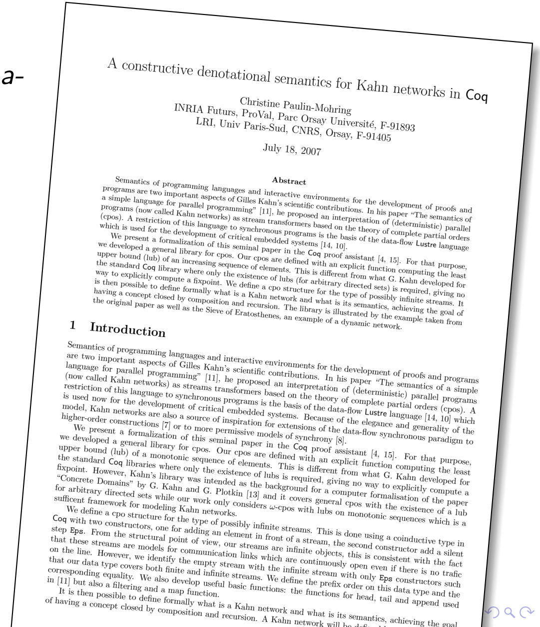
Christine Paulin-Mohring

A constructive denotational semantics for Kahn networks in Coq

From semantics to CS, 2007

Contents

- General library for CPOs



A constructive denotational semantics for Kahn networks in Coq

Christine Paulin-Mohring
INRIA Futurs, ProVal, Parc Orsay Université, F-91893
LRI, Univ Paris-Sud, CNRS, Orsay, F-91405

July 18, 2007

Abstract

Semantics of programming languages and interactive environments for the development of proofs and programs are two important aspects of Gilles Kahn's scientific contributions. In his paper "The semantics of a simple language for parallel programming" [11], he proposed an interpretation of (deterministic) parallel programs (now called Kahn networks) as stream transformers based on the theory of complete partial orders (cpos). A restriction of this language to synchronous programs is the basis of the data-flow Lustre language which is used for the development of critical embedded systems [14, 10].

We present a formalization of this seminal paper in the Coq proof assistant [4, 15]. For that purpose, we developed a general library for cpos. Our cpos are defined with an explicit function computing the least upper bound (lub) of an increasing sequence of elements. This is different from what G. Kahn developed for the standard Coq library where only the existence of lubs (for arbitrary directed sets) is required, giving no way to explicitly compute a fixpoint. We define a cpo structure for the type of possibly infinite streams. It is then possible to define formally what is a Kahn network and what is its semantics, achieving the goal of having a concept closed by composition and recursion. The library is illustrated by the example taken from the original paper as well as the Sieve of Eratosthenes, an example of a dynamic network.

1 Introduction

Semantics of programming languages and interactive environments for the development of proofs and programs are two important aspects of Gilles Kahn's scientific contributions. In his paper "The semantics of a simple language for parallel programming" [11], he proposed an interpretation of (deterministic) parallel programs (now called Kahn networks) as streams transformers based on the theory of complete partial orders (cpos). A restriction of this language to synchronous programs is the basis of the data-flow Lustre language [14, 10] which is used now for the development of critical embedded systems. Because of the elegance and generality of the model, Kahn networks are also a source of inspiration for extensions of the data-flow synchronous paradigm to higher-order constructions [7] or to more permissive models of synchrony [8].

We present a formalization of this seminal paper in the Coq proof assistant [4, 15]. For that purpose, we developed a general library for cpos. Our cpos are defined with an explicit function computing the least upper bound (lub) of a monotonic sequence of elements. This is different from what G. Kahn developed for the standard Coq libraries where only the existence of lubs is required, giving no way to explicitly compute a fixpoint. However, Kahn's library was intended as the background for a computer formalisation of the paper "Concrete Domains" by G. Kahn and G. Plotkin [13] and it covers general cpos with the existence of a lub for arbitrary directed sets while our work only considers ω-cpos with lubs on monotonic sequences which is a sufficent framework for modeling Kahn networks.

We define a cpo structure for the type of possibly infinite streams. This is done using a coinductive type in Coq with two constructors, one for adding an element in front of a stream, the second constructor add a silent step Eps. From the structural point of view, our streams are infinite objects, this is consistent with the fact that these streams are models for communication links which are continuously open even if there is no trafic on the line. However, we identify the empty stream with the infinite stream with only Eps constructors such that our data type covers both finite and infinite streams. We define the prefix order on this data type and the corresponding equality. We also develop useful basic functions: the functions for head, tail and append used in [11] but also a filtering and a map function.

It is then possible to define formally what is a Kahn network and what is its semantics, achieving the goal of having a concept closed by composition and recursion. A Kahn network

Using previous works

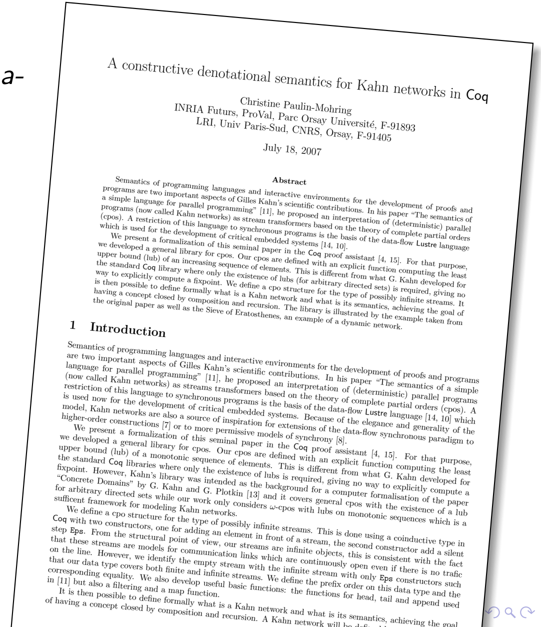
Christine Paulin-Mohring

A constructive denotational semantics for Kahn networks in Coq

From semantics to CS, 2007

Contents

- General library for CPOs
- Encoding of streams
  $s := \text{Eps } s \mid \text{Cons } a\ s$



A constructive denotational semantics for Kahn networks in Coq

Christine Paulin-Mohring
INRIA Futurs, ProVal, Parc Orsay Université, F-91893
LRI, Univ Paris-Sud, CNRS, Orsay, F-91405

July 18, 2007

Abstract

Semantics of programming languages and interactive environments for the development of proofs and programs are two important aspects of Gilles Kahn's scientific contributions. In his paper "The semantics of a simple language for parallel programming" [11], he proposed an interpretation of (deterministic) parallel programs (now called Kahn networks) as stream transformers based on the theory of complete partial orders (cpos). A restriction of this language to synchronous programs is the basis of the data-flow Lustre language which is used for the development of critical embedded systems [14, 10].

We present a formalization of this seminal paper in the Coq proof assistant [4, 15]. For that purpose, we developed a general library for cpos. Our cpos are defined with an explicit function computing the least upper bound (lub) of an increasing sequence of elements. This is different from what G. Kahn developed for the standard Coq library where only the existence of lubs (for arbitrary directed sets) is required, giving no way to explicitly compute a fixpoint. We define a cpo structure for the type of possibly infinite streams. It is then possible to define formally what is a Kahn network and what is its semantics, achieving the goal of having a concept closed by composition and recursion. The library is illustrated by the example taken from the original paper as well as the Sieve of Eratosthenes, an example of a dynamic network.

1 Introduction

Semantics of programming languages and interactive environments for the development of proofs and programs are two important aspects of Gilles Kahn's scientific contributions. In his paper "The semantics of a simple language for parallel programming" [11], he proposed an interpretation of (deterministic) parallel programs (now called Kahn networks) as streams transformers based on the theory of complete partial orders (cpos). A restriction of this language to synchronous programs is the basis of the data-flow Lustre language [14, 10] which is used now for the development of critical embedded systems. Because of the elegance and generality of the model, Kahn networks are also a source of inspiration for extensions of the data-flow synchronous paradigm to higher-order constructions [7] or to more permissive models of synchrony [8].

We present a formalization of this seminal paper in the Coq proof assistant [4, 15]. For that purpose, we developed a general library for cpos. Our cpos are defined with an explicit function computing the least upper bound (lub) of a monotonic sequence of elements. This is different from what G. Kahn developed for the standard Coq libraries where only the existence of lubs is required, giving no way to explicitly compute a fixpoint. However, Kahn's library was intended as the background for a computer formalisation of the paper "Concrete Domains" by G. Kahn and G. Plotkin [13] and it covers general cpos with the existence of a lub for arbitrary directed sets while our work only considers ω-cpos with lubs on monotonic sequences which is a sufficent framework for modeling Kahn networks.

We define a cpo structure for the type of possibly infinite streams. This is done using a coinductive type in Coq with two constructors, one for adding an element in front of a stream, the second constructor add a silent step Eps. From the structural point of view, our streams are infinite objects, this is consistent with the fact that these streams are models for communication links which are continuously open even if there is no trafic on the line. However, we identify the empty stream with the infinite stream with only Eps constructors such that our data type covers both finite and infinite streams. We define the prefix order on this data type and the corresponding equality. We also develop useful basic functions: the functions for head, tail and append used in [11] but also a filtering and a map function.

It is then possible to define formally what is a Kahn network and what is its semantics, achieving the goal of having a concept closed by composition and recursion. A Kahn network will be defi…

Using previous works

Christine Paulin-Mohring

A constructive denotational semantics for Kahn networks in Coq

From semantics to CS, 2007

Contents

- General library for CPOs
- Encoding of streams
  $s := \text{Eps } s \mid \text{Cons } a\ s$
  $\bot := \text{Eps } \bot$

A constructive denotational semantics for Kahn networks in Coq

Christine Paulin-Mohring
INRIA Futurs, ProVal, Parc Orsay Université, F-91893
LRI, Univ Paris-Sud, CNRS, Orsay, F-91405

July 18, 2007

Abstract

Semantics of programming languages and interactive environments for the development of proofs and programs are two important aspects of Gilles Kahn's scientific contributions. In his paper "The semantics of a simple language for parallel programming" [11], he proposed an interpretation of (deterministic) parallel programs (now called Kahn networks) as stream transformers based on the theory of complete partial orders (cpos). A restriction of this language to synchronous programs is the basis of the data-flow Lustre language which is used for the development of critical embedded systems [14, 10].

We present a formalization of this seminal paper in the Coq proof assistant [4, 15]. For that purpose, we developed a general library for cpos. Our cpos are defined with an explicit function computing the least upper bound (lub) of an increasing sequence of elements. This is different from what G. Kahn developed for the standard Coq library where only the existence of lubs (for arbitrary directed sets) is required, giving no way to explicitly compute a fixpoint. We define a cpo structure for the type of possibly infinite streams. It is then possible to define formally what is a Kahn network and what is its semantics, achieving the goal of having a concept closed by composition and recursion. The library is illustrated by the example taken from the original paper as well as the Sieve of Eratosthenes, an example of a dynamic network.

1 Introduction

Semantics of programming languages and interactive environments for the development of proofs and programs are two important aspects of Gilles Kahn's scientific contributions. In his paper "The semantics of a simple language for parallel programming" [11], he proposed an interpretation of (deterministic) parallel programs (now called Kahn networks) as streams transformers based on the theory of complete partial orders (cpos). A restriction of this language to synchronous programs is the basis of the data-flow Lustre language [14, 10] which is used now for the development of critical embedded systems. Because of the elegance and generality of the model, Kahn networks are also a source of inspiration for extensions of the data-flow synchronous paradigm to higher-order constructions [7] or to more permissive models of synchrony [8].

We present a formalization of this seminal paper in the Coq proof assistant [4, 15]. For that purpose, we developed a general library for cpos. Our cpos are defined with an explicit function computing the least upper bound (lub) of a monotonic sequence of elements. This is different from what G. Kahn developed for the standard Coq libraries where only the existence of lubs is required, giving no way to explicitly compute a fixpoint. However, Kahn's library was intended as the background for a computer formalisation of the paper "Concrete Domains" by G. Kahn and G. Plotkin [13] and it covers general cpos with the existence of a lub for arbitrary directed sets while our work only considers ω-cpos with lubs on monotonic sequences which is a sufficent framework for modeling Kahn networks.

We define a cpo structure for the type of possibly infinite streams. This is done using a coinductive type in Coq with two constructors, one for adding an element in front of a stream, the second constructor add a silent step Eps. From the structural point of view, our streams are infinite objects, this is consistent with the fact that these streams are models for communication links which are continuously open even if there is no trafic on the line. However, we identify the empty stream with the infinite stream with only Eps constructors such that our data type covers both finite and infinite streams. We define the prefix order on this data type and the corresponding equality. We also develop useful basic functions: the functions for head, tail and append used in [11] but also a filtering and a map function.

It is then possible to define formally what is a Kahn network and what is its semantics, achieving the goal of having a concept closed by composition and recursion. A Kahn network will be def…

Using previous works

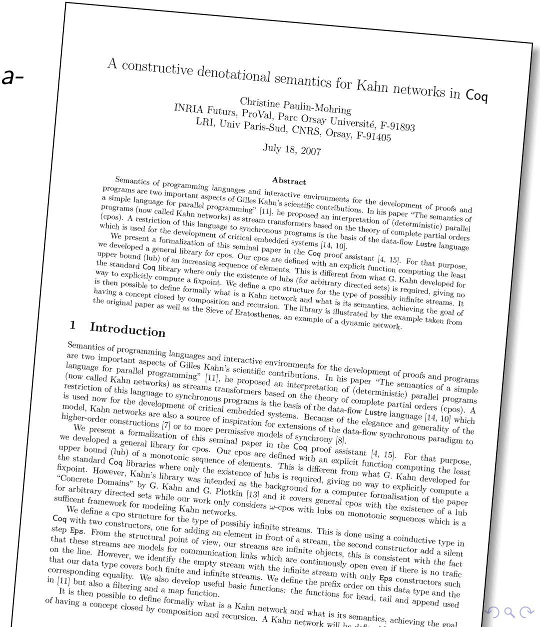
Christine Paulin-Mohring

A constructive denotational semantics for Kahn networks in Coq

From semantics to CS, 2007

Contents

- General library for CPOs
- Encoding of streams
  $s := \text{Eps } s \mid \text{Cons } a\ s$
  $\bot := \text{Eps } \bot$
- Primitive stream functions



A constructive denotational semantics for Kahn networks in Coq

Christine Paulin-Mohring
INRIA Futurs, ProVal, Parc Orsay Université, F-91893
LRI, Univ Paris-Sud, CNRS, Orsay, F-91405

July 18, 2007

Abstract

Semantics of programming languages and interactive environments for the development of proofs and programs are two important aspects of Gilles Kahn's scientific contributions. In his paper "The semantics of a simple language for parallel programming" [11], he proposed an interpretation of (deterministic) parallel programs (now called Kahn networks) as stream transformers based on the theory of complete partial orders (cpos). A restriction of this language to synchronous programs is the basis of the data-flow Lustre language which is used for the development of critical embedded systems [14, 10].

We present a formalization of this seminal paper in the Coq proof assistant [4, 15]. For that purpose, we developed a general library for cpos. Our cpos are defined with an explicit function computing the least upper bound (lub) of an increasing sequence of elements. This is different from what G. Kahn developed for the standard Coq library where only the existence of lubs (for arbitrary directed sets) is required, giving no way to explicitly compute a fixpoint. We define a cpo structure for the type of possibly infinite streams. It is then possible to define formally what is a Kahn network and what is its semantics, achieving the goal of having a concept closed by composition and recursion. The library is illustrated by the example taken from the original paper as well as the Sieve of Eratosthenes, an example of a dynamic network.

1 Introduction

Semantics of programming languages and interactive environments for the development of proofs and programs are two important aspects of Gilles Kahn's scientific contributions. In his paper "The semantics of a simple language for parallel programming" [11], he proposed an interpretation of (deterministic) parallel programs (now called Kahn networks) as streams transformers based on the theory of complete partial orders (cpos). A restriction of this language to synchronous programs is the basis of the data-flow Lustre language [14, 10] which is used now for the development of critical embedded systems. Because of the elegance and generality of the model, Kahn networks are also a source of inspiration for extensions of the data-flow synchronous paradigm to higher-order constructions [7] or to more permissive models of synchrony [8].

We present a formalization of this seminal paper in the Coq proof assistant [4, 15]. For that purpose, we developed a general library for cpos. Our cpos are defined with an explicit function computing the least upper bound (lub) of a monotonic sequence of elements. This is different from what G. Kahn developed for the standard Coq libraries where only the existence of lubs is required, giving no way to explicitly compute a fixpoint. However, Kahn's library was intended as the background for a computer formalisation of the paper "Concrete Domains" by G. Kahn and G. Plotkin [13] and it covers general cpos with the existence of a lub for arbitrary directed sets while our work only considers ω-cpos with lubs on monotonic sequences which is a sufficent framework for modeling Kahn networks.

We define a cpo structure for the type of possibly infinite streams. This is done using a coinductive type in Coq with two constructors, one for adding an element in front of a stream, the second constructor add a silent step Eps. From the structural point of view, our streams are infinite objects, this is consistent with the fact that these streams are models for communication links which are continuously open even if there is no trafic on the line. However, we identify the empty stream with the infinite stream with only Eps constructors such that our data type covers both finite and infinite streams. We define the prefix order on this data type and the corresponding equality. We also develop useful basic functions: the functions for head, tail and append used in [11] but also a filtering and a map function.

It is then possible to define formally what is a Kahn network and what is its semantics, achieving the goal of having a concept closed by composition and recursion. A Kahn network will be defi…

Using previous works

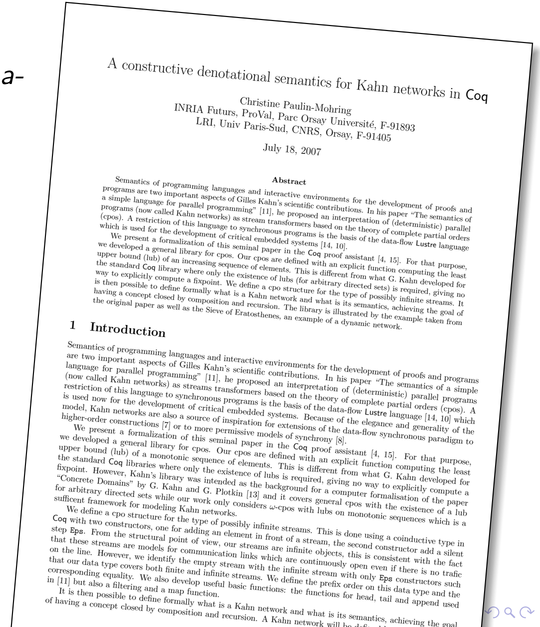
Christine Paulin-Mohring

A constructive denotational semantics for Kahn networks in Coq

From semantics to CS, 2007

Contents

- General library for CPOs
- Encoding of streams
  $s := \text{Eps } s \mid \text{Cons } a \; s$
  $\bot := \text{Eps } \bot$
- Primitive stream functions
- Example: modeling a (recursive) Kahn network



A constructive denotational semantics for Kahn networks in Coq

Christine Paulin-Mohring
INRIA Futurs, ProVal, Parc Orsay Université, F-91893
LRI, Univ Paris-Sud, CNRS, Orsay, F-91405

July 18, 2007

Abstract

Semantics of programming languages and interactive environments for the development of proofs and programs are two important aspects of Gilles Kahn's scientific contributions. In his paper "The semantics of a simple language for parallel programming" [11], he proposed an interpretation of (deterministic) parallel programs (now called Kahn networks) as stream transformers based on the theory of complete partial orders (cpos). A restriction of this language to synchronous programs is the basis of the data-flow Lustre language which is used for the development of critical embedded systems [14, 10].

We present a formalization of this seminal paper in the Coq proof assistant [4, 15]. For that purpose, we developed a general library for cpos. Our cpos are defined with an explicit function computing the least upper bound (lub) of an increasing sequence of elements. This is different from what G. Kahn developed for the standard Coq library where only the existence of lubs (for arbitrary directed sets) is required, giving no way to explicitly compute a fixpoint. We define a cpo structure for the type of possibly infinite streams. It is then possible to define formally what is a Kahn network and what is its semantics, achieving the goal of having a concept closed by composition and recursion. The library is illustrated by the example taken from the original paper as well as the Sieve of Eratosthenes, an example of a dynamic network.

1 Introduction

Semantics of programming languages and interactive environments for the development of proofs and programs are two important aspects of Gilles Kahn's scientific contributions. In his paper "The semantics of a simple language for parallel programming" [11], he proposed an interpretation of (deterministic) parallel programs (now called Kahn networks) as streams transformers based on the theory of complete partial orders (cpos). A restriction of this language to synchronous programs is the basis of the data-flow Lustre language [14, 10] which is used now for the development of critical embedded systems. Because of the elegance and generality of the model, Kahn networks are also a source of inspiration for extensions of the data-flow synchronous paradigm to higher-order constructions [7] or to more permissive models of synchrony [8].

We present a formalization of this seminal paper in the Coq proof assistant [4, 15]. For that purpose, we developed a general library for cpos. Our cpos are defined with an explicit function computing the least upper bound (lub) of a monotonic sequence of elements. This is different from what G. Kahn developed for the standard Coq libraries where only the existence of lubs is required, giving no way to explicitly compute a fixpoint. However, Kahn's library was intended as the background for a computer formalisation of the paper "Concrete Domains" by G. Kahn and G. Plotkin [13] and it covers general cpos with the existence of a lub for arbitrary directed sets while our work only considers ω-cpos with lubs on monotonic sequences which is a sufficent framework for modeling Kahn networks.

We define a cpo structure for the type of possibly infinite streams. This is done using a coinductive type in Coq with two constructors, one for adding an element in front of a stream, the second constructor add a silent step Eps. From the structural point of view, our streams are infinite objects, this is consistent with the fact that these streams are models for communication links which are continuously open even if there is no trafic on the line. However, we identify the empty stream with the infinite stream with only Eps constructors such that our data type covers both finite and infinite streams. We define the prefix order on this data type and the corresponding equality. We also develop useful basic functions: the functions for head, tail and append used in [11] but also a filtering and a map function.

It is then possible to define formally what is a Kahn network and what is its semantics, achieving the goal of having a concept closed by composition and recursion. A Kahn network will be

Using previous works

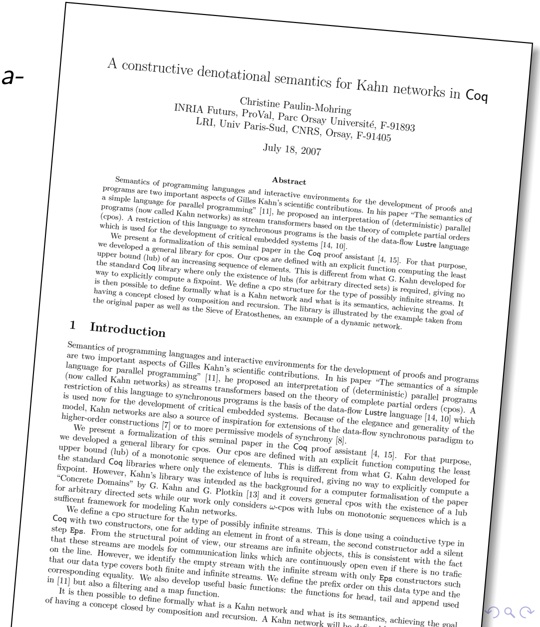
Christine Paulin-Mohring

A constructive denotational semantics for Kahn networks in Coq

From semantics to CS, 2007

Contents

- General library for CPOs
- Encoding of streams
  $s := \text{Eps } s \mid \text{Cons } a\ s$
  $\bot := \text{Eps } \bot$
- Primitive stream functions
- Example: modeling a (recursive) Kahn network



A constructive denotational semantics for Kahn networks in Coq

Christine Paulin-Mohring
INRIA Futurs, ProVal, Parc Orsay Université, F-91893
LRI, Univ Paris-Sud, CNRS, Orsay, F-91405

July 18, 2007

Abstract

Semantics of programming languages and interactive environments for the development of proofs and programs are two important aspects of Gilles Kahn's scientific contributions. In his paper "The semantics of a simple language for parallel programming" [11], he proposed an interpretation of (deterministic) parallel programs (now called Kahn networks) as stream transformers based on the theory of complete partial orders (cpos). A restriction of this language to synchronous programs is the basis of the data-flow Lustre language which is used for the development of critical embedded systems [14, 10].

We present a formalization of this seminal paper in the Coq proof assistant [4, 15]. For that purpose, we developed a general library for cpos. Our cpos are defined with an explicit function computing the least upper bound (lub) of an increasing sequence of elements. This is different from what G. Kahn developed for the standard Coq library where only the existence of lubs (for arbitrary directed sets) is required, giving no way to explicitly compute a fixpoint. We define a cpo structure for the type of possibly infinite streams. It is then possible to define formally what is a Kahn network and what is its semantics, achieving the goal of having a concept closed by composition and recursion. The library is illustrated by the example taken from the original paper as well as the Sieve of Eratosthenes, an example of a dynamic network.

1 Introduction

Semantics of programming languages and interactive environments for the development of proofs and programs are two important aspects of Gilles Kahn's scientific contributions. In his paper "The semantics of a simple language for parallel programming" [11], he proposed an interpretation of (deterministic) parallel programs (now called Kahn networks) as streams transformers based on the theory of complete partial orders (cpos). A restriction of this language to synchronous programs is the basis of the data-flow Lustre language [14, 10] which is used now for the development of critical embedded systems. Because of the elegance and generality of the model, Kahn networks are also a source of inspiration for extensions of the data-flow synchronous paradigm to higher-order constructions [7] or to more permissive models of synchrony [8].

We present a formalization of this seminal paper in the Coq proof assistant [4, 15]. For that purpose, we developed a general library for cpos. Our cpos are defined with an explicit function computing the least upper bound (lub) of a monotonic sequence of elements. This is different from what G. Kahn developed for the standard Coq libraries where only the existence of lubs is required, giving no way to explicitly compute a fixpoint. However, Kahn's library was intended as the background for a computer formalisation of the paper "Concrete Domains" by G. Kahn and G. Plotkin [13] and it covers general cpos with the existence of a lub for arbitrary directed sets while our work only considers ω-cpos with lubs on monotonic sequences which is a sufficent framework for modeling Kahn networks.

We define a cpo structure for the type of possibly infinite streams. This is done using a coinductive type in Coq with two constructors, one for adding an element in front of a stream, the second constructor add a silent step Eps. From the structural point of view, our streams are infinite objects, this is consistent with the fact that these streams are models for communication links which are continuously open even if there is no trafic on the line. However, we identify the empty stream with the infinite stream with only Eps constructors such that our data type covers both finite and infinite streams. We define the prefix order on this data type and the corresponding equality. We also develop useful basic functions: the functions for head, tail and append used in [11] but also a filtering and a map function.

It is then possible to define formally what is a Kahn network and what is its semantics, achieving the goal of having a concept closed by composition and recursion. A Kahn network will be defi…

Conclusion, discussion

Done

Conclusion, discussion

Done

- Reproducing old results in the context of Vélus/Coq ITP

Conclusion, discussion

Done

- Reproducing old results in the context of Vélus/Coq ITP
- Identify best methods and use it to give a natural semantics to the full language

Conclusion, discussion

Done

- Reproducing old results in the context of Vélus/Coq ITP
- Identify best methods and use it to give a natural semantics to the full language
- Verify some small programs with clocks

Conclusion, discussion

Done

- Reproducing old results in the context of Vélus/Coq ITP
- Identify best methods and use it to give a natural semantics to the full language
- Verify some small programs with clocks

To do

- Verify more (parameterized) programs, new proof techniques

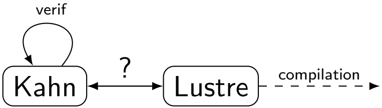
Conclusion, discussion

Done

- Reproducing old results in the context of Vélus/Coq ITP
- Identify best methods and use it to give a natural semantics to the full language
- Verify some small programs with clocks

To do

- Verify more (parameterized) programs, new proof techniques
- Link with the semantic model of Vélus



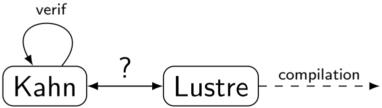
Conclusion, discussion

Done

- Reproducing old results in the context of Vélus/Coq ITP
- Identify best methods and use it to give a natural semantics to the full language
- Verify some small programs with clocks

To do

- Verify more (parameterized) programs, new proof techniques
- Link with the semantic model of Vélus



- Is the Kahn semantics suitable for some compilation steps?

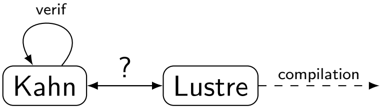
Conclusion, discussion

Done

- Reproducing old results in the context of Vélus/Coq ITP
- Identify best methods and use it to give a natural semantics to the full language
- Verify some small programs with clocks

To do

- Verify more (parameterized) programs, new proof techniques
- Link with the semantic model of Vélus



- Is the Kahn semantics suitable for some compilation steps?
- What about the *existence* of a semantics?