Verified Time-Aware Stream Processing

Rafael Castro G. Silva

rasi@di.ku.dk

Department of Computer Science
University of Copenhagen

02/11/2023

What is this PhD/Status seminar about?

- Distributed Systems
  - Stream processing frameworks
    - Dataflow models
      - Time-Aware Computations
- Formal Methods
  - Verification using proof assistants
    - Isabelle proofs
      - Verified and executable code
- Formalization of Time-Aware Stream Processing

Contents

Introduction

Dataflow Models

- Stream Processing
- Dataflow Model
- Time-Aware Computations
- Bugs in Stream Processing

Preliminaries

Isabelle/HOL

- HOL
- Isabelle/HOL

Isabelle/HOL: (Co)datatypes

- Datatypes and Codatatypes

$$\textbf{codatatype}\ (\mathsf{lset}\colon {'a})\ \mathit{llist} = \mathsf{lnull}\colon \mathsf{LNil} \mid \mathsf{LCons}\ (\mathsf{lhd}\colon {'a})\ (\mathsf{ltl}\colon {'a}\ \mathit{llist})$$
$$\quad \textbf{for}\ \mathsf{map}\colon \mathsf{lmap}\ \textbf{where}\ \mathsf{ltl}\ \mathsf{LNil} = \mathsf{LNil}$$

- Induction principle for lset membership:

$$\begin{gathered} x \in \mathsf{lset}\ \mathit{lxs} \longrightarrow (\bigwedge x\ \mathit{lxs}.\ P\ x\ (\mathsf{LCons}\ x\ \mathit{lxs})) \longrightarrow \\ (\bigwedge x\ \mathit{lxs}\ y\ .\ y \in \mathsf{lset}\ \mathit{lxs} \longrightarrow P\ y\ \mathit{lxs} \longrightarrow P\ y\ (\mathsf{LCons}\ x\ \mathit{lxs})) \longrightarrow P\ x\ \mathit{lxs} \end{gathered} \tag{1}$$

- Coinductive principle for lazy list equality:
  - Show a *bisimulation* R that relates the lazy lists:

$$\begin{gathered} R\ \mathit{lxs}\ \mathit{lys} \longrightarrow \bigwedge \mathit{lxs}\ \mathit{lys}\ .\ R\ \mathit{lxs}\ \mathit{lys} \longrightarrow \mathsf{lnull}\ \mathit{lxs} \longleftrightarrow \mathsf{lnull}\ \mathit{lys} \land \\ \neg \mathsf{lnull}\ \mathit{lxs} \longrightarrow \neg \mathsf{lnull}\ \mathit{lys} \longrightarrow \mathsf{lhd}\ \mathit{lxs} = \mathsf{lhd}\ \mathit{lys} \land R\ (\mathsf{ltl}\ \mathit{lxs})\ (\mathsf{ltl}\ \mathit{lys})) \longrightarrow \mathit{lxs} = \mathit{lys} \end{gathered} \tag{2}$$

Isabelle/HOL: Recursion and While Combinator

- Recursion

$$\textbf{fun}\ \mathsf{lshift} :: {}'a\ list \Rightarrow {}'a\ llist \Rightarrow {}'a\ llist\ (infixr\ @@\ 65)\ \textbf{where}$$
$$\mathsf{lshift}\ []\ lxs = lxs$$
$$|\ \mathsf{lshift}\ (x \# xs)\ lxs = \mathsf{LCons}\ x\ (\mathsf{lshift}\ xs\ lxs)$$

- While Combinator

$$\textbf{definition}\ \mathsf{while\_option} :: ({}'a \Rightarrow bool) \Rightarrow ({}'a \Rightarrow {}'a) \Rightarrow {}'a \Rightarrow {}'a\ option\ \textbf{where}$$
$$\mathsf{while\_option}\ b\ c\ s = (\mathtt{if}\ (\exists k.\ \neg\ b\ ((c\ \hat{}\hat{}\ k)\ s))$$
$$\mathtt{then}\ \mathsf{Some}\ ((c\ \hat{}\hat{}\ (\mathsf{LEAST}\ k.\ \neg\ b\ ((c\ \hat{}\hat{}\ k)\ s)))\ s)$$
$$\mathtt{else}\ \mathsf{None})$$

- While rule

$$\boxed{Q\ s \longrightarrow \mathsf{while\_option}\ t\ b\ s = \mathsf{Some}\ s' \longrightarrow (\bigwedge s.\ Q\ s \longrightarrow t\ s \longrightarrow Q\ (b\ s)) \longrightarrow Q\ s' \qquad (3)}$$

Isabelle/HOL: Corecursion and Friends

- Corec
- Friend

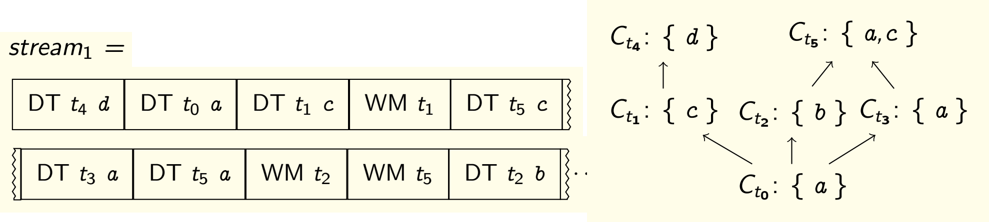
Isabelle/HOL: (Co)inductive Predicates

- Inductive
- Coinductive
- Coinduction

Lazy Lists Processors

Operators

- Operator
- Produce produce
- Example



(a) Prefix of $stream_1$ (b) Corresponding set of collections

Figure: An example stream and its collections (ordered by their time-stamps)

Sequential Composition

- Composition
- Skip n

Time-Aware Operators

Monotone and Productive Time-Aware Streams

- Monotone
- Productive

Building Blocks: Batch Operator

Batch Operator: Soundness

Batch Operator: Completeness

- Uses soundness of batch_op
- Proof by induction over n

$$
\begin{aligned}
&\text{mono\_prod } \mathit{lxs}\ W \longrightarrow (\exists i\ d.\ \text{enat } i < \text{llength } \mathit{lxs} \land \text{lnth } \mathit{lxs}\ i = \text{DT } t\ d \land n = \text{Suc } i) \lor \\
&n = 0 \land t \in \text{set\_t } \mathit{buf} \longrightarrow (\forall t' \in \text{set\_t } \mathit{buf}.\ \text{lfinite } \mathit{lxs} \lor \exists wm \geq t'.\ \text{WM } wm \in \text{lset } \mathit{lxs}) \longrightarrow \\
&\exists wm\ batch.\ \text{DT } wm\ batch \in \text{lset } (\text{produce } (\text{batch\_op } \mathit{buf})\ \mathit{lxs}) \land t \in \text{set\_t } batch \lor \\
&(\forall k \in \{n\ ..< \text{the\_enat } (\text{llength } \mathit{lxs})\}.\ \neg (\exists t' \geq t.\ \text{lnth } \mathit{lxs}\ k = \text{WM } t')) \land \text{lfinite } \mathit{lxs}
\end{aligned} \tag{4}
$$

Batch Operator: Monotone

Batch Operator: Productive

Building Blocks: Incremental Operator

Batch Operator: Soundness

Batch Operator: Completeness

Batch Operator: Monotone

Batch Operator: Productive

Compositional Reasoning

Case Study

Histogram

Histogram: Soundness

Histogram: Completeness

Histogram: Monotone

Histogram: Productive

Efficient Histogram

- Foo

Join

Join: Soundness

Join: Completeness

Join: Monotone

Next Steps

Next Steps

Questions, comments and suggestions