Representing the Process Algebra CSP in Type Theory

Bashar Igried and Anton Setzer

Swansea University, Swansea,Wales, UK

bashar.igried@yahoo.com , a.g.setzer@swansea.ac.uk

TYPES 2016, Novi Sad, Serbia, 26 May 2016

Overview

1. Why Agda?
2. Process Algebra
3. CSP
4. CSP-Agda
5. Choice Sets
6. Future Work
7. Conclusion

Why Agda?

- Agda supports induction-recursion.
  Induction-Recursion allows to define universes.
- Agda supports definition of coalgebras by elimination rules and defining their elements by combined pattern and copattern matching.
- Using of copattern matching allows to define code which looks close to normal mathematical proofs.

Overview Of Process Algebras

- "Process algebra" was initiated in 1982 by Bergstra and Klop **?**, in order to provide a formal semantics to concurrent systems.

- Baeten et. al. Process algebra is the study of distributed or parallel systems by algebraic means.

- Three main process algebras theories were developed.
  - Calculus of Communicating Systems (CCS).
    Developed by Robin Milner in 1980.
  - Communicating Sequential Processes (CSP).
    Developed by Tony Hoare in 1978.
  - Algebra of Communicating Processes (ACP).
    Developed by Jan Bergstra and Jan Willem Klop, in 1982.

- Processes will be defined in Agda according to the operational behaviour of the corresponding CSP processes.

CSP

- CSP considered as a formal specification language, developed in order to describe concurrent systems.

  By identifying their behaviour through their communications.
- CSP is a notation for studying processes which interact with each other and their environment.
- In CSP we can describe a process by the way it can communicate with its environment.
- A system contains one or more processes, which interact with each other through their interfaces.

CSP Syntax

In the following table, we list the syntax of CSP processes:

Q ::=	STOP	$STOP$
|	SKIP	$SKIP$
|	prefix	$a \rightarrow Q$
|	external choice	$Q \,\square\, Q$
|	internal choice	$Q \sqcap Q$
|	hiding	$Q \setminus a$
|	renaming	$Q[R]$
|	parallel	$Q \;_X\|_Y\; Q$
|	interleaving	$Q \;\|\|\|\; Q$
|	interrupt	$Q \triangle Q$
|	composition	$Q \,;\, Q$

CSP-Agda

- We will represent the process algebra CSP in a coinductive form in dependent type theory.
- Implement it in the Agda.
- CSP processes can proceed at any time both with labelled transitions and with silent transitions.
- Therefore, processes in CSP-Agda have as well this possibility.

In Agda the corresponding code is as follows:

$$
\begin{array}{l}
\textsf{record Process : Set} \quad \textsf{where} \\
\quad \textsf{coinductive} \\
\quad \textsf{field} \\
\qquad \begin{array}{lcll}
\mathrm{E} & : & \textsf{Choice} & \\
\mathrm{Lab} & : & \textsf{ChoiceSet}\ \mathrm{E} & \rightarrow \quad \textsf{Label} \\
\mathrm{PE} & : & \textsf{ChoiceSet}\ \mathrm{E} & \rightarrow \quad \textsf{Process} \\
\mathrm{I} & : & \textsf{Choice} & \\
\mathrm{PI} & : & \textsf{ChoiceSet}\ \mathrm{I} & \rightarrow \quad \textsf{Process}
\end{array}
\end{array}
$$

CSP-Agda

So we have in case of a process progressing:

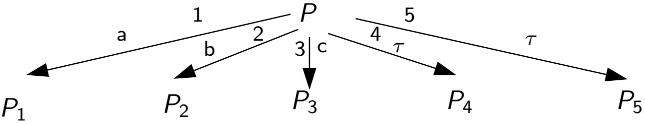
- an index set E of external choices and for each external choice e a label $(\mathrm{Lab}\ e)$ and a next process $(\mathrm{PE}\ e)$;
- an index set of internal choices $\imath$ and for each internal choice i a next process $(\mathrm{PI}\ i\ )$.

CSP-Agda

As an example the following Agda code describes the process pictured below:

$$
\begin{array}{llcllllcllllcl}
\mathrm{E} & P & = & \text{code for } \{1,2,3\} \\
\mathrm{Lab} & P\ 1 & = & a & & \mathrm{Lab} & P\ 2 & = & b & & \mathrm{Lab} & P\ 3 & = & c \\
\mathrm{PE} & P\ 1 & = & P_1 & & \mathrm{PE} & P\ 2 & = & P_2 & & \mathrm{PE} & P\ 3 & = & P_3
\end{array}
$$

$$
\begin{array}{llcllllcl}
\mathrm{I} & P & = & \text{code for } \{4,5\} \\
\mathrm{PI} & P\ 4 & = & P_4 & & \mathrm{PI} & P\ 5 & = & P_5
\end{array}
$$



Choices Set

- Choice sets are modelled by a universe.
- Universes go back to Martin-Löf in order to formulate the notion of a type consisting of types.
- Universes are defined in Agda by an inductive-recursive definition.

Choices Set

We give here the code expressing that Choice is closed under Bool, disjoint union $+$ and subset.

mutual

$$
\begin{array}{l}
\quad \mathsf{data}\ \mathrm{Choice} : \mathsf{Set}\quad \mathsf{where} \\
\qquad \begin{array}{lcl}
\widehat{\mathsf{Bool}} & : & \mathrm{Choice} \\
\_\,\widehat{+}\,\_ & : & \mathrm{Choice} \to \mathrm{Choice} \to \mathrm{Choice} \\
\mathrm{subset} & : & (E : \mathrm{Choice}) \to (\mathsf{ChoiceSet}\ E \to \mathrm{Bool}) \to \mathrm{Choice}
\end{array}
\end{array}
$$

$$
\begin{array}{l}
\mathsf{ChoiceSet} : \mathsf{Choice} \to \mathsf{Set} \\
\begin{array}{lcl}
\mathsf{ChoiceSet}\ \widehat{\mathsf{Bool}} & = & \mathrm{Bool} \\
\mathsf{ChoiceSet}\ (a\ \widehat{+}\ b) & = & \mathsf{ChoiceSet}\ a + \mathsf{ChoiceSet}\ b \\
\mathsf{ChoiceSet}\ (\mathrm{subset}\ E\ f) & = & \mathrm{Subset}\ (\mathsf{ChoiceSet}\ E)\ f
\end{array}
\end{array}
$$

Interleaving operator

- In this process, the components P and Q execute completely independently of each other.
- Each event is performed by exactly one process.
- The operational semantics rules are straightforward:

$$\frac{P \xrightarrow{l} \bar{P}}{P \mid\mid\mid Q \xrightarrow{l} \bar{P} \mid\mid\mid Q} \qquad \frac{Q \xrightarrow{l} \bar{Q}}{P \mid\mid\mid Q \xrightarrow{l} P \mid\mid\mid \bar{Q}}$$

Interleaving operator

We represent interleaving operator in CSP-Agda as follows

$$
\begin{array}{llll}
\_|||\_ & : \mathsf{Process} \;\rightarrow\; \mathsf{Process} \;\rightarrow\; \mathsf{Process} & & \\
\mathrm{E} & (P \,|||\, Q) & = & \mathrm{E}\; P \;+'\; \mathrm{E}\; Q \\
\mathrm{Lab} & (P \,|||\, Q) \quad (\mathsf{inl}\; x) & = & \mathrm{Lab}\; P\; x \\
\mathrm{Lab} & (P \,|||\, Q) \quad (\mathsf{inr}\; x) & = & \mathrm{Lab}\; Q\; x \\
\mathrm{PE} & (P \,|||\, Q) \quad (\mathsf{inl}\; x) & = & \mathrm{PE}\; P \;\; x \,|||\, Q \\
\mathrm{PE} & (P \,|||\, Q) \quad (\mathsf{inr}\; x) & = & P \,|||\, \mathrm{PE}\; Q\; x \\
\mathrm{I} & (P \,|||\, Q) & = & \mathrm{I}\; P \;+'\; \mathrm{I}\; Q \\
\mathrm{PI} & (P \,|||\, Q) \quad (\mathsf{inl}\; x) & = & \mathrm{PI}\; P \;\; x \,|||\, Q \\
\mathrm{PI} & (P \,|||\, Q) \quad (\mathsf{inr}\; x) & = & P \,|||\, \mathrm{PI}\; Q\; x
\end{array}
$$

Traces

$$
\begin{array}{l}
\mathsf{data}\,\mathrm{Tr} : \mathsf{List\ Label} \to \mathsf{Process} \to \mathsf{Set} \quad \mathsf{where} \\
\quad \begin{array}{lcl}
\mathrm{empty} & : & \{P : \mathsf{Process}\ \} \\
 & \to & \mathrm{Tr}\ []\ P \\
\mathrm{trE} & : & \{P : \mathsf{Process}\ \} \\
 & \to & (x : \mathsf{ChoiceSet}\ (\mathrm{E}\ P)) \\
 & \to & (l\ : \mathsf{List\ Label}) \\
 & \to & \mathrm{Tr}\ l\ (\mathrm{PE}\ P\ x) \\
 & \to & \mathrm{Tr}\ (\mathrm{Lab}\ P\ x\ ::\ l)\ P \\
\mathrm{trI} & : & \{P : \mathsf{Process}\} \\
 & \to & (x : \mathsf{ChoiceSet}\ (\mathrm{I}\ P)) \\
 & \to & (l\ : \mathsf{List\ Label}) \\
 & \to & \mathrm{Tr}\ l\ (\mathrm{PI}\ P\ x) \\
 & \to & \mathrm{Tr}\ l\ P
\end{array}
\end{array}
$$

Traces Refinement

The refinement relation $\sqsubseteq_T$ on process is defined by

$$P \sqsubseteq_T Q$$
$$\text{if and only if}$$
$$\mathrm{traces}(Q) \subseteq \mathrm{traces}(P)$$

- The subscript T indicates that we are working with traces. The Agda definition is as follows:

$$\_\sqsubseteq_T\_ : (P : \text{Process}) \to (P' : \text{Process}) \to \text{Set}$$
$$P \sqsubseteq_T P' = (l : \text{List Label}) \to \mathrm{Tr}\ l\ P' \to \mathrm{Tr}\ l\ P$$

Proving Symmetry of Interleaving operator

$$\begin{array}{ll}
\mathrm{Sym}||| : (P\ Q\ :\ \textsf{Process}) \ \rightarrow\ (P\ |||\ Q)\ \sqsubseteq\ (Q\ |||\ P) & \\
\mathrm{Sym}|||\ P\ Q\ \ \mathrm{empty} & = \mathrm{empty} \\
\mathrm{Sym}|||\ P\ Q\ (\mathrm{trE}\ (\textsf{inl}\ x)\ /\ tr) & = \mathrm{trE}\ (\textsf{inr}\ x)\ /\ (\mathrm{Sym}|||\ P\ (\mathrm{PE}\ Q\ x)\ tr) \\
\mathrm{Sym}|||\ P\ Q\ (\mathrm{trE}\ (\textsf{inr}\ x)\ /\ tr) & = \mathrm{trE}\ (\textsf{inl}\ x)\ /\ (\mathrm{Sym}|||\ (\mathrm{PE}\ P\ x)\ Q\ tr) \\
\mathrm{Sym}|||\ P\ Q\ (\mathrm{trI}\ \ (\textsf{inl}\ x)\ /\ tr) & = \mathrm{trI}\ \ (\textsf{inr}\ x)\ /\ (\mathrm{Sym}|||\ P\ (\mathrm{PI}\ Q\ x)\ \ tr) \\
\mathrm{Sym}|||\ P\ Q\ (\mathrm{trI}\ \ (\textsf{inr}\ x)\ /\ tr) & = \mathrm{trI}\ \ (\textsf{inl}\ x)\ /\ (\mathrm{Sym}|||\ (\mathrm{PI}\ P\ x)\ Q\ \ tr)
\end{array}$$

Future Work

- Looking to the future, we would like to model complex systems in Agda.
- Model examples of processes occurring in the European Train Management System (ERTMS) in Agda.
- Show correctness.

Further Work

- The other operations (external choice, internal choice, parallel operations, hiding, renaming, etc.) are defined in a similar way.
- Several laws of CSP have been shown with respect to traces semantics and bisimulation.
- A simulator of CSP processes in Agda has been developed.
- Define approach using Sized types.
- For complex examples (e.g recursion) sized types are used to allow application of functions to the co-IH.

Conclusion

- A formalisation of CSP in Agda has been developed using coalgebra types and copattern matching.
- We have shown CSP-Agda supports refinement proofs over CSP traces model.

The End