Introduction to Calculus of Communicating Systems (SSP Algebra)

Yulong Zou and Yu-Dong Yao
Department of Electrical and Computer Engineering

October 7, 2009
Outline

- Calculus of communicating systems (CCS) and process algebra
- Why CCS
- How to model a process in CCS
- CCS representations of a transmitter
- Future work
The Calculus of Communicating Systems (CCS), proposed by Milner [1], is now referred to as process algebra.

By using CCS, communicating processes can be modeled as algebraic functions, based on which we reason the behavior of processes with a deductive mechanism.

Why CCS?

- CCS enables concurrent, especially interleaved, processes to be equivalently converted into purely nondeterministic ones.
- CCS allows us to analyze the specified behaviour with the laws of algebra as well as reducing the process specifications, resulting in improving the implementation efficiency.
How to Model a Process in CCS?

- **Basic rule**: Each state transition in a process is modeled as one CCS equation, where the left-hand side denotes the current state and the right-hand represents the next state.

  CCS Equations: $S_0 = aS_2 + bS_1$; $S_1 = cS_2$, where “+” stands for an union operation.
CCS Representations of a Transmitter

Consider a simple transmitter as shown below

Next, we implement the simple transmitter with three concurrent, instead of the chronological, processes.
Using the three concurrent processes illustrated above, we now can describe the CCS representations.
Initial State

The initial state is $C_0()M_0()T_0()$, where "()" denotes the data queue of corresponding process. Following $C_0()M_0()T_0()$, the next state is described as

$$C_0()M_0()T_0() = \text{bits}\?C_0()\text{bits}M_0()T_0()$$
$$+ \text{coded_bits}\?C_0()\text{coded_bits}M_0()T_0()$$
$$+ \text{symbol}\?C_0()\text{symbol}M_0()T_0()$$

where the question mark ? means input of a signal. Here, we use a priority strategy in describing CCS equations proposed by Venemans [2].

Priority Scheme

Basic idea of priority scheme: A system shall finish its ‘internal work’ before responding to external signals. The priorities proposed in [2] are described as follows (in decreasing order):

1. $\tau$(tasks, decisions, etc)
2. $\tau^-$
3. $\tau^+$
4. external communication

where $\tau^-$ and $\tau^+$ denote ‘queue-emptying’ and ‘queue-filling’, respectively.
Priority Consideration: Queue-Emptying

As shown in Eq. (1), all possible actions have the same priority, thus we need to handle all the possibilities. First, let us consider the first term in the right-hand side of Eq. (1), i.e.,

\[ C_0(\text{bits})M_0()T_0() = \tau^{-}C_1()M_0()T_0() \]
\[ + \text{bits?}C_0(\text{bits,bits})M_0()T_0() \]
\[ + \text{coded_bits?}C_0(\text{bits})M_0(\text{coded_bits})T_0() \]
\[ + \text{symbol?}C_0(\text{bits})M_0()T_0(\text{symbol}) \]

The first line shows the consumption of signal ‘bits’, after which it enters the state C_1. The other lines represents further inputs from external environment.
Priority Consideration: Task (Channel Encoding)

Since the first term in the right-hand side of Eq. (2) is priori
to the others, we will omit these external inputs in following
equations. Hence, following Eq. (2), we have

\[ C_1 (\)M_0 (\)T_0 (\) = \tau (\text{channel coding})C_2 (\)M_0 (\)T_0 (\) + \ldots \]  

(3)

This is a task statement, i.e., channel encoding, which is not
directly related with communication. Such an action has the
highest priority among the four defined behaviors: task, queue-
emptying, queue-filling, and external communication.
Priority Consideration: Queue-Filling

Similarly, based on Eq. (3), we obtain

\[ C_2 (M_0 () T_0 ()) = \tau^+ C_0 (M_0 (\text{coded_bits}) T_0 ()) + \cdots \]  (4)

This is a ‘queue filling’ action: Channel encoder submits ‘coded_bits’ to the queue of constellation mapper. All the algebraic equations are described as follows:

\[ C_0 (M_0 (\text{coded_bits}) T_0 ()) = \tau^- C_0 (M_1 () T_0 ()) + \cdots \]

\[ C_0 (M_1 () T_0 ()) = \tau (\text{constellation mapping}) C_0 (M_2 () T_0 ()) + \cdots \]

\[ C_0 (M_2 () T_0 ()) = \tau^+ C_0 (M_0 () T_0 (\text{symbol})) + \cdots \]

\[ C_0 (M_0 () T_0 (\text{symbol})) = \tau^- C_0 (M_0 () T_1 ()) + \cdots \]

\[ C_0 (M_0 () T_1 ()) = \tau (\text{RF transmitting}) C_0 (M_0 () T_0 ()) + \cdots \]
Furthermore, by renaming state names and applying the standard reduction rules of CCS, the final description can be reduced to the following process definition:

\[ S_0 = \text{bits}S_1 + \text{coded_bits}S_2 + \text{symbol}S_3 \]

\[ S_1 = \tau^-\tau(\text{channel coding})\tau^+S_2 \]

\[ S_2 = \tau^-\tau(\text{constellation mapping})\tau^+S_3 \]

\[ S_3 = \tau^-\tau(\text{RF transmitting})S_0 \]

Now, this process definition can be the basis for the generation of executable codes.

To do: Use cognitive linguistics graphical representations
Future Work

➢ To consider the CCS description for a radio communication system including both the transmitter and the receiver (e.g., with ARQ protocol), where the concurrent processes of the transceivers interleave with each other.

➢ Cognitive linguistics representations

➢ Role of “algebra” (CCS?) in SSP/cognitive linguistics