March 9, 2000

BRICS, University of Aarhus
Frank D. Valencia

(by Saraswat, Jagadeesan and Gupta)
Motivation

- Modularity in the spirit of process algebra.
- Declarative view.
- Partial information.
- Logic properties can be expressed within the calculus.

Why CCP?

Processing systems.

A CCP language for modeling timed reactive systems: Controllers and signal
The store evolves monotonically.

Operations \texttt{ReadWrite} replaced by \texttt{AskTell}.

\textbf{Basic Ideas of CCP:}

\textbf{Reminder: Motivation}

March 9, 2000
Reminder: CCP Scenario & Concurrency

\[ \text{tell}(X > 10) \quad \text{ask}(X < 50) \rightarrow P \]

\[ \text{tell}(X < 20) \quad \text{ask}(X = 15) \rightarrow Q \]

▷ **Concurrent Executions of Agents.**

▷ **Synchronization**: Via Blocking-Ask.
\( q \vdash \varnothing \) if \( q \vdash \varnothing \) for all \( a \in \mathcal{T} \) and \( \varnothing \vdash q \) \nolabel C_2

\( \varnothing \vdash a \) if \( a \in \varnothing \) \nolabel C_1

Definition 1. A constraint system \((cs)\) is a structure \(\langle D, \vdash \rangle\), where \( D \) is a set of primitive constraints and \( \vdash \) is a compact entailment relation satisfying:

Reminder: Constraint Systems à la Scott.
\[ \{ \; x > 7, x > 6, x > 5, \ldots \} \]

is represented as \[ \{ \; x > 5 \} \] provided a suitable set of natural numbers with \( \geq \) constraint.

Definition 2. Constraints are subsets of \( \mathbb{D} \) closed under \( \setminus \).

The set of all constraints is denoted by \( \mathcal{D} \).

System

Reminder: Constraints, the Elements of a Constraint
Reminder: Semantic domain; The Lattice

\(\Delta\) Structure \((|D|, \subseteq)\) is a complete (algebraic) lattice:

\[
\begin{array}{c}
\text{false} = D \\
\vdots \\
\vdots \\
\text{true} = \{a \in D | \emptyset \vdash a\}
\end{array}
\]
\(\{X \geq 10\}\) is a quiescent point for process \(\emptyset\).

A quiescent point for process \(p\) is a constraint on input of which \(p\) terminates without adding any information.

\(\Downarrow\)

It's set of quiescent points.

\(\Downarrow\)

Reminder: Quiescent Points.

March 9, 2000
Agenda
Intuitions: Reactive Systems
Absence of signals = Absence of Information = Negative Information.

Presence of signals = Presence of Information = Positive Information.

In TCC:

"Absence of signals can be detected immediately to input signals. At any instant the presence and the absence programs combinators are determined primitives that respond to the hypotheses of perfect synchrony:

Intuitions: Synchronous Languages
Program $P$ can be read as "The light is ON at time $t$ only if it is not ON at time $t$!"

\[ P \equiv \begin{cases} \text{IF LIGHT \neq ON THEN LIGHT = ON ELSE LIGHT \neq ON} \\ \end{cases} \]

To require a signal to be present at an instant only if it is not present at that instant.

Example: Consider CCP program $P$ to be run at time $t$.

Paradox

March 9, 2000
only in the next interval.

Absence of information detected at the end of the intervals triggers activity

Intervals must be timed bounded.

Happens in a very short bounded period of time: Programs in between the

Internal temporality (Bounded Asynchrony):

Timed Asynchrony Hypothesis:

\[
\begin{align*}
&\text{Sys} \\
&\text{E\&N} \\
&\text{TCP Scenario}
\end{align*}
\]
Program
recursive definition
definition call
Local X in A.
Parallel execution.
unless c do A next time
if c then now A
execute A next time
tell c
abort
do nothing

\[ A ::= (X)^d \]
\[ (X)d \mid A \parallel X \mid \text{now} \ c \ \text{else} \ A \mid \text{now} \ c \ \text{then} \ A \mid \text{next} \ A \mid c \mid \text{abort} \mid \text{skip} \]

Basic Agents A, B, ... = skip

Let X, Y, ... be variables and v be primitive constraints.

Syntax for the tcc-modal.
A LEGO robot example

\[
\text{robot} \overset{\text{def}}{=} \text{run.go-back.go.turn-left.} \ ( \text{run} \parallel \text{ENV} )
\]

\[
\begin{align*}
\text{run} &::= \text{now} (s_1 = \text{on}) \text{ then } \text{go-back} \\
& \quad \quad \parallel \text{now}(o_1 = o_2 = (\text{on, backwards})) \text{ else } \text{go-forward} \\
\text{go-back} &::= (o_1 = o_2 = (\text{on, backwards})) \parallel \text{next turn-left} \\
\text{turn-left} &::= (o_1 = (\text{on, backwards}) \land o_2 = (\text{off, -})) \parallel \text{next run} \\
\text{go-forward} &::= (o_1 = o_2 = (\text{on, forward})) \parallel \text{next run}
\end{align*}
\]
Absence of information detected at the end of intervals.

No information carried through the intervals.

Queries answers bounded in the size of the input.

Recursion calls only allowed in `next` and `else` bodies.

Note that:
A quiescent sequence is a sequence \( q_1, q_2, \ldots \) such that:

Observations to make? Quiescent sequences of constraints.

TCC Observations
Let $\mathcal{O}$ be the set of all processes, \( \mathcal{P} \), and \( \mathcal{D} \) be the set of observable actions. Let the set of finite sequences of elements of \( \mathcal{D} \) be denoted by \( \mathcal{D}^{*} \) and let \( \mathcal{D}^{\omega} \) be the set of infinite sequences.

**Definition 3.** The semantic domain \( \mathcal{D} \) is defined as the set of all functions from \( \mathcal{D}^{*} \) to \( \mathcal{D}^{\omega} \).

The set of all processes, \( \mathcal{P} \), is a complete lattice:

\[
\{ \epsilon \} = \top = \bot \triangleleft
\]

\( 1 \) if \( s \in D \) then \( s \in P \) and

\( 2 \) if \( s \in P \) then \( s \in P' \)

\( 3 \) if \( P \subseteq P' \) is a process iff constraints. Set \( P \subseteq \mathcal{O} \) is a process iff

\( \{e\} = \top = \bot \triangleleft \)
<table>
<thead>
<tr>
<th>$\forall \ p \land \ q$</th>
<th>$[B] \cup [A]$</th>
<th>$I, p \leftarrow I, q$</th>
<th>$B \parallel A$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\forall \ o \ [A] \supseteq s : o \in s \cdot p \cap {\varepsilon}$</td>
<td>$I, p \leftarrow I, q$</td>
<td>next $A$</td>
<td></td>
</tr>
<tr>
<td>$\forall \ o \land \ c \ [A] \supseteq s$</td>
<td>${\varepsilon}$</td>
<td>$I, p \leftarrow I, q$</td>
<td>now else $A$</td>
</tr>
<tr>
<td>$\forall \ o \land \ c \ [A] \supseteq s \cdot p$</td>
<td>${\varepsilon}$</td>
<td>$I, p \leftarrow I, q$</td>
<td>now then $A$</td>
</tr>
<tr>
<td>$c$</td>
<td>${\varepsilon}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>False</td>
<td></td>
<td></td>
<td>abort</td>
</tr>
<tr>
<td>True</td>
<td></td>
<td></td>
<td>skip</td>
</tr>
<tr>
<td>Logically</td>
<td></td>
<td></td>
<td>Operationally</td>
</tr>
<tr>
<td>Denotationally</td>
<td></td>
<td></td>
<td>Process $p$</td>
</tr>
</tbody>
</table>

**Semantics**

March 9, 2000
\[ \{ \exists B \} \{ \forall \} \leftarrow \{ m > \} \ \text{next} \ \{ \forall \} \ \text{now} \ \text{else} \ \{ \forall \} \ \text{now} \ \text{else} \ \{ \forall \} \]

Interval to next one
Corresponding Results

Soundness

Theorem. Two programs are denotationally distinct if there is a context in which can operationally distinguish between them.

Partial Completeness

Theorem. \( A \rightarrow B \) implies \( A \subseteq [B] \).

Existential quantifiers.

Theorem. \( \forall [B] A \rightarrow B \) for \( A \), \( B \) without procedures calls and

\[ [A] A \subseteq [B] \]
always A. "Execute A always"

whenever c do A. "Execute A the first time c is entailed"

S.\text{\textup{\textsuperscript{e}}}A\textsuperscript{e} (A). (Suspension-Activation) "Suspend A if c, re-activate A if e"

do A watching c. (Watchdog): "Do A till c is entailed, then Kill A"

do A on c. (Multi-form time): "Do A at the instance the store entails c"

Some derived combinator actions.
clock skip do \( A = A \).  

clock abort do \( A = \text{skip} \).  

do \( A \) whenever \( c \) = clock (whenever \( c \) do next abort) do \( A \).  

do \( A \) when (watching \( c \) = clock) do \( A \).  

time \( A \) on \( c \) = clock (always \( c \) do \( A \).  

do \( A \) whenever \( c \) do \( A \) = clock \( c \) do \( A \).  

\[
\{ [A] \in B \triangleright t : O \in f \} = \left[\begin{array}{c}
\\text{clock } B \text{ do } A
\end{array}\right]
\]

The clock combinator
Deterministic finite state automaton.

Compositional translation of tcc programs into finite state automata.

**States:** Labeled with recursion-free determinate ccp. A distinguished start state.

**Transitions:** Directed edges between states, labeled with constraints the set of outgoing nodes is closed under lubs.

For example:
Future work: Verification techniques, theory of preemption and languages.

Conclusions