Towards GCAB in Coq

Jean-Bernard Stefani

joint work with C. Di Giusto and A. Schmitt

INRIA

Outline

1. Motivations

2. CAB: A process calculus intepretation of BIP

3. GCAB: Towards reconciling BIP and Fractal

Outline

1. Motivations
2. CAB: A process calculus intepretation of BIP
3. GCAB: Towards reconciling BIP and Fractal

Motivations

1. Understanding various software engineering / programming structures: components, features, aspects
   - forms of modularity
   - structures with sharing
2. Programming model including dynamic modularity with preemption
   - isolation example

$$\texttt{SafePluginReceipt} = a(x).\texttt{new}\, i, s.\, \texttt{Watchdog}(s, i) \mid i : s[x] \mid \texttt{Alarm}(i)$$
$$\texttt{Watchdog}(s, i) = (i \mid s[z]).0$$

Motivations

Reconciling two influential component models:

- BIP
  - hierarchical components
  - gluing as parallel composition with superimposed synchronisation
  - multipoint synchronization under priority constraints
  - target: embedded, real-time systems
- Fractal
  - hierarchical components with sharing
  - reflective structure for dynamic reconfiguration
  - target: dynamic systems

Outline

1. Motivations
2. CAB: A process calculus intepretation of BIP
3. GCAB: Towards reconciling BIP and Fractal

BIP in a nutshell

- Primitive components: labelled transitions systems.
- Composite components $S = (B_1, \ldots, B_n)$ with behavioral rules obeying the format:

$$r : \frac{\{B_i \xrightarrow{a_i} B_i'\}_{i \in I} \quad \{B_j \overset{b_j^k}{\not\longrightarrow} \mid k \in [1..m_j]\}_{j \in J}}{(B_1, \ldots, B_n) \xrightarrow{a} (B_1', \ldots, B_n')}$$

CAB: syntax

- Given a family $\mathcal{P}$ of primitive components,
- CAB composites are built by superimposition of glue processes $P, Q, \ldots$ on components

$$
\begin{array}{llll}
S & ::= & & \textit{component ensembles} \\
 & \mid & \emptyset & \textit{null ensemble} \\
 & \mid & \{C_1, \ldots, C_n\} & \textit{finite ensemble} \\
 & & & \\
C & ::= & I[S \star P] & \textit{component} \\
 & \mid & I[A] & A \in \mathcal{P} \textit{ primitive component}
\end{array}
$$

$$I, I_i \in \mathcal{N}_I$$

CAB: syntax

$$
\begin{array}{rcll}
P, Q & ::= & & \textit{processes} \\
 & & 0 & \textit{null process} \\
 & & \mid \ X & \textit{process variable} \\
 & & \mid \ \alpha.P & \textit{prefix} \\
 & & \mid \ P \mid Q & \textit{parallel} \\
 & & \mid \ \mu X.\, P & \textit{recursion} \\
 & & & \\
\alpha & ::= & \langle pr :: a :: syn \rangle & \textit{actions}
\end{array}
$$

CAB: syntax

$$
\begin{array}{lll}
pr & ::= & \textit{priority constraints} \\
 & \emptyset & \textit{void} \\
 & \mid \{l_1 : a_1, \ldots, l_n : a_n\} & \textit{preemptive actions} \\
 & & \\
syn & ::= & \textit{synchronisation constraints} \\
 & \emptyset & \textit{void} \\
 & \mid \{l_1 : a_1, \ldots, l_n : a_n\} & \textit{synchronised actions}
\end{array}
$$

$$a, a_i \in \mathcal{N}_p \quad l, l_i \in \mathcal{N}_l$$

CAB: operational semantics

Behavioral rules for glue processes:

$$\text{Act } \alpha.P \xrightarrow{\alpha} P \qquad \text{Par1 } \frac{P \xrightarrow{\alpha} P'}{P \mid Q \xrightarrow{\alpha} P' \mid Q} \qquad \text{Par2 } \frac{Q \xrightarrow{\alpha} Q'}{P \mid Q \xrightarrow{\alpha} P \mid Q'}$$

$$\text{Rec } \frac{P\{^{\mu X.\,P}/_{X}\} \xrightarrow{\alpha} P'}{\mu X.\,P \xrightarrow{\alpha} P'}$$

Behavioral rules for composites:

$$\text{Comp}\ \frac{\{C_i \xrightarrow{l_i:a_i} C_i' \mid i \in I\} \qquad P \xrightarrow{\langle pr::a::\{l_i:a_i \mid i \in I\}\rangle} P' \qquad \{C_i \mid i \in I\} \subseteq S \qquad S \models pr}{l[S \star P] \xrightarrow{l:a} l[(S \setminus \{C_i \mid i \in I\}) \cup \{C_i' \mid i \in I\} \star P']}$$

$$\text{Tau}\ \frac{C \xrightarrow{h:\tau} C'}{l[\{C\} \cup S \star P] \xrightarrow{l:\tau} l[\{C'\} \cup S \star P]}$$

BIP in CAB

BIP glues can be encoded in CAB as glue processes of the form $\prod_{i=1}^{n} [\![r_i]\!]$, where:

$$r : \frac{\{C_i \xrightarrow{a_i} C'_i\}_{i \in I} \quad \{C_j \overset{c_j^k}{\not\longrightarrow} \mid k \in [1..m_j]\}_{j \in J}}{(C_1, \ldots, C_n) \xrightarrow{tag} (C'_1, \ldots, C'_n)}$$

$$[\![r]\!] = !\langle \{h_j : c_j^k \mid k \in [1, m_j]\}_{j \in J}, tag, \{h_i : a_i\}_{i \in I} \rangle$$

$$!\alpha.P = rec\, X.\ \alpha.(P \parallel X)$$

Theorem

BIP systems defined over a set $\mathcal{P}$ of components can be faithfully encoded in CAB($\mathcal{P}$): any BIP system S is strongly bisimilar to its encoding $[\![S]\!]$.

Some results on CAB

Theorem

CAB($\emptyset$) is Turing complete.

Proved by encoding of Minsky machines in CAB($\emptyset$).

Theorem

CAB($\emptyset$) without priorities is not Turing complete.

Proved by encoding of CAB($\emptyset$) without priorities in Petri nets.

Outline

1. Motivations

2. CAB: A process calculus intepretation of BIP

3. GCAB: Towards reconciling BIP and Fractal

GCAB: intuitions

GCAB generalizes CAB in four ways:

- Pure port synchronization $\longrightarrow$ value passing on ports.
- Tree structure for composites $\longrightarrow$ directed graph.
- Static composite structure $\longrightarrow$ dynamic structure.
- CCS-based glue language $\longrightarrow$ π-calculus-based glue language.

GCAB (static) core: syntax

$\kappa ::= \Gamma \odot S$	*configurations*
$\Gamma, \Delta ::=$	*control graphs*
$\emptyset$	*empty graph*
$\mid \{h_1 \triangleright l_1, \ldots, h_n \triangleright l_n\}$	*graph*
$S ::=$	*component ensembles*
$\emptyset$	*null ensemble*
$\{C_1, \ldots, C_n\}$	*finite ensemble*
$C ::= l[P]$	*component*

$h, l \in \mathcal{N}_l$

GCAB core: LTS for processes

$$\text{Act } \alpha.P \xrightarrow{\alpha} P$$

$$\text{Par1 } \frac{P \xrightarrow{\alpha} P'}{P \mid Q \xrightarrow{\alpha} P' \mid Q} \qquad \text{Par2 } \frac{Q \xrightarrow{\alpha} Q'}{P \mid Q \xrightarrow{\alpha} P \mid Q'}$$

$$\text{Rec } \frac{P\{^{\mu X.\, P}/_{X}\} \xrightarrow{\alpha} P'}{\mu X.\, P \xrightarrow{\alpha} P'}$$

GCAB core: LTS for configurations

$$\text{GComp}\ \frac{\{\Gamma \odot S_i \xrightarrow{l_i:a_i} \Gamma \odot S_i' \mid i \in I\} \qquad P \xrightarrow{\langle pr::a::\{l_i:a_i \mid i \in I\}\rangle} P' \quad \{l \triangleright l_i \mid i \in I\} \subseteq \Gamma \qquad \bigcup_{i \in I} S_i \subseteq S \qquad \Gamma \odot S \models_l pr}{\Gamma \odot \{l[P]\} \cup S \xrightarrow{l:a} \Gamma \odot \{l[P']\} \cup (S \setminus \bigcup_{i \in I} S_i) \cup \bigcup_{i \in I} S_i'}$$

$$\text{GTau}\ \frac{\Gamma \odot S \xrightarrow{h:\tau} \Gamma \odot S' \qquad l \triangleright h \in \Gamma}{\Gamma \odot \{l[P]\} \cup S \xrightarrow{l:\tau} \Gamma \odot \{l[P]\} \cup S'}$$

GCAB core: priority constraints

$$
\begin{aligned}
\Gamma \odot S \models_I \{l_i : a_i \mid i \in I\} \iff\ & \exists \{C_i \mid i \in I\}, \\
& \{C_i \mid i \in I\} \subseteq S \\
& \wedge\ \forall i \in I, \\
& \quad l_i = \mathtt{nm}(C_i) \\
& \quad\quad \wedge \neg(\Gamma \odot S \xrightarrow{l_i : a_i}) \\
& \quad\quad \wedge I \triangleright l_i \in \Gamma
\end{aligned}
$$

GCAB core: Defining the LTS as a least fixpoint

$$
\begin{aligned}
&\mathcal{F}(R_1, R_2) = \langle R_1', R_2' \rangle \\
&R_1' = R_1 \cup \mathtt{r}(R_1, R_2) \\
&R_2' = R_2 \cap \mathtt{r}(R_2, R_1) \\
&\mathtt{r}(R_1, R_2) = \{\langle \kappa, l : a, \kappa' \rangle \mid \mathtt{gcomp}(R_1, R_2, \kappa, l, a, \kappa')\} \\
&\qquad\qquad\qquad \cup \{\langle \kappa, l, \kappa' \rangle \mid \mathtt{gtau}(R_1, \kappa, l : \tau, \kappa')\}
\end{aligned}
$$

$$
\begin{aligned}
\mathtt{gtau}(R_1, \kappa, l, \kappa', n) \iff & \\
& \exists \Gamma, P, S, S', h, \\
& \wedge\, \kappa = \Gamma \odot \{l[P]\} \cup S \\
& \wedge\, \kappa' = \Gamma \odot \{l[P]\} \cup S' \\
& \wedge\, l \triangleright h \in \Gamma \\
& \wedge\, \langle \Gamma \odot S, h : \tau, \Gamma \odot S' \rangle \in R_1
\end{aligned}
$$

GCAB core: Defining the LTS as a least fixpoint

$$\langle R_1, R_2 \rangle \sqsubseteq \langle T_1, T_2 \rangle \iff R_1 \subseteq T_1 \wedge T_2 \subseteq R_2$$

$\mathcal{F}$ is an order-preserving function on a complete lattice

$$\rightarrow \;\triangleq\; \pi_1(\mu\mathcal{F})$$

Theorem

$\rightarrow$ is the least well supported model of the GCAB core rules and it is complete, i.e. $\mu\mathcal{F} = (\rightarrow, \rightarrow)$.

Dynamic extensions

- Creating new components: $\texttt{new}(x, P)$
- Adding an edge: $\triangleright a$
- Removing an edge: $\triangleleft a$
- Updating processes: $\texttt{swap}(a, X, P)$

Formalizing GCAB in Coq ?

- Start with GCAB core
- Essential use of negative premises in rule GComp $\implies$ no inductive Coq definition possible for $\rightarrow$.
- Strategy: formalize $\mathcal{F}$ fixpoint construction.
  - Define `gcomp` and `gtau` as inductive predicates with relation parameters
  - Define `r` and $\mathcal{F}$ as functions on relations
  - Prove $\mathcal{F}$ monotonous wrt $\sqsubseteq$
  - Obtain $\mu\mathcal{F}$ as intersection of pre-fixed points
  - Prove $\mu\mathcal{F}$ of the form (R, R) and define $\rightarrow \triangleq R$
  - Define inductive principle on $\mathcal{F}$.

Question: can we envisage Coq tools to support this way of dealing with negative premises ?