On Expressiveness and Behavioural Theory of Attribute-based Communication

Rocco De Nicola

Joint work with
Y. A. Alrahman and M. Loreti

Final Meeting CINA
Civitanova Marche – January 2016
Contents

1 Introduction

2 Programming Abstractions for CAS

3 AbC: A Process Calculus for CAS

4 Encoding other communication paradigms

5 A Behavioural Theory for AbC

6 Ongoing and Future work
CAS are software-intensive systems featuring:

- **massive numbers** of components
- **complex interactions** among components, and other systems
- operating in **open and non-deterministic environments**
- **dynamically adapting** to new requirements, technologies and environmental conditions

Challenges for software development for CAS:

- the **dimension** of the systems
- the **need to adapt** to changing environments and requirements
- the **emergent behaviour** resulting from complex interactions
- the **uncertainty** during design-time and run-time
Examples of CAS

Robot swarms

Cooperative e-vehicles

Clouds
Programming abstractions for CAS

The Service-Component Ensemble Language (SCEL) currently provides primitives and constructs for dealing with 4 programming abstractions.

1. **Knowledge**: to describe how data, information and (local and global) knowledge is managed
2. **Behaviours**: to describe how systems of components progress
3. **Aggregations**: to describe how different entities are brought together to form components, systems and, possibly, ensembles
4. **Policies**: to model and enforce the wanted evolutions of computations.
Components and Systems

Aggregations describe how different entities are brought together and controlled:

- **Components:**

- **Systems:**
Collective Adaptive Systems as Ensembles

Ensembles

- Systems are structured as sets of components dynamically forming ensembles, i.e. groups of components that collaborate to achieve specific tasks;
- Ensembles are not rigid networks but highly flexible structures where components linkages are dynamically established;
- Interaction is based on a novel communication paradigm different from one-to-one message passing and from broadcast.

Predicate based interaction

- Components expose their attributes in the interface;
- Interaction between components is based on predicates over attributes that permit dynamically specifying all targets of a given communication action.
Ensembles are determined by the predicates validated by each component.

There is no coordinator, hence no bottleneck or critical point of failure.

A component might be part of more than one ensemble.
Example Predicates

- \( id \in \{n, m, p\} \)
- \( active = yes \land \text{battery\_level} > 30\% \)
- \( \text{range}_{\text{max}} > \sqrt{(this.x - x)^2 + (this.y - y)^2} \)
- \( \text{true} \)
- \( \text{trust\_level} > \text{medium} \)
- \( \ldots \)
- \( \text{trousers} = \text{red} \)
- \( \text{shirt} = \text{green} \)
Towards a Theory of CAS

We aim at developing a theoretical foundation of CAS, starting from their distinctive features, summarized as follows:

- CAS consist of large numbers of interacting components which exhibit complex behaviours depending on their attributes, objectives and actions.
- CAS components may enter or leave the collective at anytime and might have different (possibly conflicting) objectives and need to dynamically adapt to new requirements and contextual conditions.

AbC: A calculus with Attribute based Communication

We have defined AbC, a calculus inspired by SCEL and focusing on a minimal set of primitives that rely on attribute-based communication for systems interaction.
AbC at a glance

- Systems are represented as sets of parallel components, each of them equipped with a set of attributes whose values can be modified by internal actions.
- Communication actions (send and receive) are decorated with predicates over attributes that partners have to satisfy to make the interaction possible.
- Communication takes place in an implicit multicast fashion, and communication partners are selected by relying on predicates over the attributes exposed in their interfaces.
- Components are unaware of the existence of each other and they receive messages only if they satisfy partners requirements.
- Components can offer different views of themselves and can interact with different partners according to different criteria.
- Semantics for output actions is non-blocking while input actions are blocking in that they can only take place through synchronisation with an available sent message.
AbC through a running example

- A swarm of robots is spread throughout a disaster area with the goal of locating victims to rescue.
- Robots have rôles modelled via functional behaviours that can be changed via appropriate adaptation mechanisms.
- Initially all robots are explorers; a robot that finds a victim becomes a rescuer and sends info about the victim to nearby explorers; to form ensembles.
- An explorer that receives information about a victim changes its rôle into helper and joins the rescuers ensemble.
- The rescuing procedure starts when the ensemble is complete.

Some of the attributes (e.g. battery level) are the projection of the robot internal state controlled via sensors and actuators.
AbC Components

(Components) \( C ::= \Gamma : P \mid C_1 \parallel C_2 \mid \nu x C \)

- Single component \( \Gamma : P \) – \( \Gamma \) denotes sets of attributes and \( P \) processes
- Parallel composition \( \parallel \) – of components
- Name restriction \( \nu x \) (to delimit the scope of name \( x \)) – in \( C_1 \parallel (\nu x)C_2 \), name \( x \) is invisible from within \( C_1 \)

Running example (step 1/5)

- Each robot is modeled as an AbC component (\( Robot_i \)) of the following form (\( \Gamma_i : P_R \)).
- Robots execute in parallel and collaborate.

\[ Robot_1 \parallel \ldots \parallel Robot_n \]
AbC Processes

\[ P ::= 0 \mid \text{Act}.P \mid \text{new}(x)\ P \mid \langle \Pi \rangle P \mid P_1 + P_2 \mid P_1|P_2 \mid K \]

- \text{new}(x)\ P – Process name restriction
- \langle \Pi \rangle P – blocks \ P \text{ until the evaluation of } \Pi \text{ under the local environment becomes true (awareness operator).}
- \text{Act} – communication and attribute update actions

Running example (step 2/5)

\( P_R \) running on a robot has the following form:

\[ P_R \triangleq (\langle \Pi \rangle a_1.P_1 + a_2.P_2)|P_3 \]

- When \( \Pi \) evaluates to true (e.g., victim detection), the process performs action \( a_1 \) and continues as \( P_1 \);
- Otherwise \( P_R \) performs \( a_2 \) to continue as \( P_2 \) (help rescuing a victim).
AbC Actions

\[
\text{Act} ::= \Pi(\tilde{x}) \mid (\tilde{E})@\Pi \vdash \{s\} \mid [a := E]
\]

- \(\Pi(\tilde{x})\) – receive from any component satisfying \(\Pi\);
- \((\tilde{E})@\Pi \vdash \{s\}\) – send to components satisfying \(\Pi\) while exposing only the attributes in set \(s\);
- \([a := E]\) – updates the value of \(a\) with the result of evaluating \(E\).
AbC Actions

\[
\text{Act ::= } \Pi(\tilde{x}) \mid (\tilde{E})@\Pi \vdash \{s\} \mid [a := E]
\]

- \(\Pi(\tilde{x})\) – receive from any component satisfying \(\Pi\);
- \((\tilde{E})@\Pi \vdash \{s\}\) – send to components satisfying \(\Pi\) while exposing only the attributes in set \(s\);
- \([a := E]\) – updates the value of \(a\) with the result of evaluating \(E\).

Running example (step 3/5)

- By specifying \(\Pi, a_1,\) and \(a_2, P_R\) becomes:

\[
P_R \triangleq (\langle \text{this.victimPerceived} = \text{tt} \rangle [\text{this.state} := \text{stop}] . P_1 + (\text{this.id, qry}@ (\text{role} = \text{rescuer} \lor \text{role} = \text{helping}) \vdash \{\text{role}\} . P_2 ) | P_3
\]
AbC Actions

<table>
<thead>
<tr>
<th>Act ::= Π(˜(x))</th>
<th>((\tilde{E})@Π ⊢ {s})</th>
<th>([a := E])</th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ Π(˜(x)) – receive from any component satisfying (Π);</td>
<td></td>
<td></td>
</tr>
<tr>
<td>▶ ((\tilde{E})@Π ⊢ {s}) – send to components satisfying (Π) while exposing only the attributes in set (s);</td>
<td></td>
<td></td>
</tr>
<tr>
<td>▶ ([a := E]): – updates the value of (a) with the result of evaluating (E).</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Running example (step 3/5)

▶ By specifying \(\Pi\), \(a_1\), and \(a_2\), \(P_R\) becomes:

\[
P_R \triangleq (\langle \text{this.victimPerceived} = \text{tt} \rangle \ [\text{this.state} := \text{stop}] . P_1 + (\text{this.id, qry}@ (\text{role} = \text{rescuer} \lor \text{role} = \text{helping}) ⊢ \{\text{role}\} . P_2 ) ) \mid P_3
\]

We are dwelling whether to use \(\Pi(\tilde{x})(\sigma)\) with \(\sigma = [a_1 \mapsto E_1, \ldots, a_n \mapsto E_n]\) as input action to atomically update the local environment of the receiver.
(Components) \[ C ::= \Gamma : P \mid C_1 \parallel C_2 \mid \nu x C \]

(Processes) \[ P ::= \]

(Inaction) \[ 0 \]

(Input) \[ \Pi(\tilde{x}).P \]

(Output) \[ (\tilde{E})@\Pi \vdash \{s\} .P \]

(Update) \[ [a := E].P \]

(New) \[ \text{new}(x)P \]

(Match) \[ \langle \Pi \rangle P \]

(Choice) \[ P_1 + P_2 \]

(Par) \[ P_1 | P_2 \]

(Call) \[ K \]

(Predicates) \[ \Pi ::= \text{tt} \mid \text{ff} \mid E_1 \triangleright E_2 \mid \Pi_1 \land \Pi_2 \mid \ldots \]

(Data) \[ E ::= v \mid x \mid a \mid \text{this}.a \mid \ldots \]
Transitions Labels

- we use the $\lambda$-label to range over broadcast, input, update and internal labels respectively

\[
\lambda \in \{\nu \tilde{x} \Gamma:(\tilde{v})@\Pi, \quad \Gamma:(\tilde{v})@\Pi, \quad [a := v], \quad \tau\}
\]

- we use the $\alpha$-label to range over all $\lambda$-labels plus the input-discarding label as follows:

\[
\alpha \in \lambda \cup \{\Gamma:(\tilde{v})@\Pi\}
\]
Operational Semantics

Processes and Systems Semantics

AbC is equipped with a two levels labelled semantics.

1. the behaviour of processes is modelled by the transition relation

\[ \rightarrow \subseteq \text{Proc} \times \text{PLAB} \times \text{Proc} \]

2. the behaviour of component is modelled by the transition relation:

\[ \rightarrow \subseteq \text{Comp} \times \text{CLAB} \times \text{Comp} \]

where

- \( \text{Proc} \) stands for Processes and \( \text{Comp} \) stands for Components,
- \( \text{PLAB} \) stands for

\[ \{ \nu \tilde{x} \Gamma : (\tilde{\nu}) @ \Pi, \quad \Gamma : (\tilde{\nu}) @ \Pi, \quad [a := \nu], \quad \tau, \quad \Gamma : (\tilde{\nu}) @ \Pi \} \]
- \( \text{CLAB} \) stands for

\[ \{ \nu \tilde{x} \Gamma : (\tilde{\nu}) @ \Pi, \quad \Gamma : (\tilde{\nu}) @ \Pi, \quad \tau \} \]
Semantics of Processes (excerpt)

\[(\text{Brd})\]

\[
\begin{align*}
\Gamma \models [E] & = \tilde{v} \\
\Gamma \models [\Pi_1] & = \Pi \\
(\tilde{E})@\Pi_1 \models s . P & \xrightarrow{\Gamma | s : (\tilde{v})@\Pi} \Gamma P
\end{align*}
\]

\[
\Gamma | s = \begin{cases} 
\Gamma(a) & \text{if } a \in s \\
\bot & \text{otherwise}
\end{cases}
\]

\[(\text{Rcv})\]

\[
\begin{align*}
\Pi_1[\tilde{v}/\tilde{x}] \models & = \Pi_1' \\
(P' : (\tilde{v})@\Pi_2) & \xrightarrow{\Gamma} P[\tilde{v}/\tilde{x}]
\end{align*}
\]

Running example (step 4/5)

- \(P_R\) resides within a robot with \(\Gamma(id) = 1\)
- Some possible evolutions where \(\Gamma' = \Gamma_1|\{role\}\) are:

\[
\begin{align*}
P_R & \xrightarrow{\text{this.state:=stop}} \Gamma_1 \quad P_1|P_3 \\
P_R & \xrightarrow{\Gamma' : (1, \text{qry}):(role=\text{rescuer} \lor role=\text{helping})} \Gamma_1 \quad P_2|P_3
\end{align*}
\]
Semantics of Processes (excerpt)

Discarding Label

(FBrd) \((\tilde{E})@\Pi_1 \vdash_s P \xrightarrow{\Gamma' : (\tilde{v})@\Pi_2} \Gamma (\tilde{E})@\Pi_1 \vdash_s P\)

(FSum) \(P_1 \xrightarrow{\Gamma' : (\tilde{v})@\Pi} \Gamma P_1 \quad P_2 \xrightarrow{\Gamma' : (\tilde{v})@\Pi} \Gamma P_2\)

\[ P_1 + P_2 \xrightarrow{\Gamma' : (\tilde{v})@\Pi} \Gamma P_1 + P_2 \]

- Rules like (FBrd) models the non-blocking nature of the broadcast;
- Rules like (FSum)), are instead used to control internal non-determinism as side-effect and avoid unwanted evolutions.

Running example (step 4/6)

- \(P_R\) resides within a robot with explorer role.
- \(P_R\) can discard unwanted broadcasts.

\(P_R \xrightarrow{\Gamma'_2 : (\text{info})@(_\text{role=explorer})} \Gamma_1 P_R\)
From Processes to Components (excerpt)

\[
\begin{align*}
\text{(C-Brd)} & \quad P \xrightarrow{\Gamma : P \vdash \Gamma : P'} P' \\
\text{(C-Rcv)} & \quad P \xrightarrow{\Gamma : P \vdash \Gamma : P'} P' \quad (\Gamma \vdash \Pi)
\end{align*}
\]

\[
\begin{align*}
\text{(Com)} & \quad C_1 \xrightarrow{\nu \tilde{x} \Gamma : (\tilde{\nu}) \@ \Pi} C_1' \\
& \quad C_2 \xrightarrow{\Gamma : (\tilde{\nu}) \@ \Pi} C_2' \\
& \quad C_1 \parallel C_2 \xrightarrow{\nu \tilde{x} \Gamma : (\tilde{\nu}) \@ \Pi} C_1' \parallel C_2'
\end{align*}
\]

Running example (step 5/5): Further specifying \( P_2 \) in \( P_R \)

\[
\text{Query} \triangleq (\text{this}.id, \text{qry}) \@ (\text{role} = \text{rescuer} \lor \text{role} = \text{helper}) \vdash \{\text{role}\} \cdot
\]

\[
( ((\text{role} = \text{rescuer} \lor \text{role} = \text{helper}) \land x = \text{ack})
\]

\[
(\text{victim}_\text{pos}, x).P_2'
\]

[\text{Query}]

\[
\text{AbC: A Process Calculus for CAS} \quad \text{R. De Nicola} \quad 20/31
\]
Running example (step 5/5): Cont.

- Assume \textit{Robot}_2 is “rescuer”, \textit{Robot}_3 is “helper”, and all others are explorers.

- \textit{Robot}_3 received victim information from \textit{Robot}_2 and now is in charge.

- \textit{Robot}_1 sent a msg containing its identity “\textit{this.id}” and “\textit{qry}” request and \textit{Robot}_3 caught it. Now by using rule (\textbf{C-Brd}), \textit{Robot}_3 sends the victim position “\textless 3, 4 \textgreater” and “ack” back to \textit{Robot}_1 as follows:

\[
\Gamma_3 : P_{R_3} \quad \xrightarrow{\Gamma: (<3,4>, \text{ack}@id=1)} \quad \Gamma_3 : P'_{R_3} \quad \text{where } \Gamma = \Gamma_3 \mid \{\text{role}\}.
\]

- \textit{Robot}_1 applies rule (\textbf{C-Rcv}) to receive victim information and generates this transition.

\[
\Gamma_1 : P_{R_1} \quad \xrightarrow{\Gamma: (<3,4>, \text{ack}@id=1)} \quad \Gamma_1 : P'_{R_2}[< 3, 4 >/\textit{victim}_\text{pos}, \text{ack}/x]
\]
Running example (step 5/5): Cont.

- Robots can perform the above transitions since

\[ \Gamma_1 \models (id = 1) \text{ and } \Gamma \models ((role = rescuer \lor role = helper) \land x = ack). \]

Other robots discard the broadcast.

- Now the overall system evolves by applying rule (Com) as follows:

\[
S \quad \underbrace{\Gamma : (\langle 3, 4 \rangle, ack)@id = 1}_{\text{S}} \implies \Gamma_1 : P'_2[\langle 3, 4 \rangle/victim_{pos}, ack/x] \parallel \Gamma_2 : P_{R_2} \parallel \Gamma_3 : P'_{R_3} \parallel \ldots \parallel \Gamma_n : P_{R_n}
\]
Encoding other communication paradigms

A number of alternative communication paradigms such as:

▶ Explicit Message Passing
▶ Group based Communications
▶ Publish-Subscribe

can be easily modelled by relying on *AbC primitives*
A \( b\pi \)-calculus process \( P \) is rendered as an \( AbC \) component \( \Gamma : P \) where \( \Gamma = \emptyset \).

Possible problem

Impossibility of specifying the channel along which the exchange has to happen instantaneously.

Way out

Send the communication channel as a part of the transmitted values and the receiver checks its compatibility.

\[
\langle \\bar{a}x.P \rangle \triangleq (a, x)@ (a = a) \vdash \{ .(P) \}
\]

\[
\langle a(x).P \rangle \triangleq \Pi(y, x).(P) \quad \text{with} \quad \Pi = (y = a) \quad \text{and} \quad y \notin n((P))
\]
Group-based interaction

- A group name is encoded as an attribute in AbC.
- The constructs for joining or leaving a given group can be encoded as attribute updates.
- ...
Publish-Subscribe interaction is a simple special case of attribute-based communication:

- A Publisher sends tagged messages for all subscribers by exposing from his environment only the current topic.
- Subscribers check compatibility of messages according to their subscriptions.

\[
\Gamma_1 : (msg)@ (tt) \vdash \{\text{topic}\} \parallel \\
\Gamma_2 : (\text{topic} = \text{this.subscription})(x) \parallel \\
\vdots \\
\Gamma_n : (\text{topic} = \text{this.subscription})(x) \parallel
\]

Observation

Dynamic updates of attributes and the possibility of controlling their visibility give AbC great flexibility and expressive power.
Some Notations

- \( \Rightarrow \) denotes \( \tau \rightarrow^* \)
- \( \xRightarrow{\gamma} \) denotes \( \Rightarrow \xRightarrow{\gamma} \Rightarrow \) if \( \gamma \neq \tau \)
- \( \xRightarrow{\gamma} \) denotes \( \Rightarrow \xRightarrow{\gamma} \) otherwise.
- \( \rightarrow \) denotes \{ \( \xRightarrow{\gamma} \mid \gamma \) is an output or \( \gamma = \tau \) \}
- \( \rightarrow^* \) denotes \( (\rightarrow)^* \)

AbC Contexts

A context \( C[\bullet] \) is a component term with a hole, denoted by \( [\bullet] \) and AbC contexts are generated by the following grammar:

\[
C[\bullet] ::= [\bullet] \mid [\bullet] \parallel C \mid C \parallel [\bullet] \mid \nu X[\bullet]
\]
Observable Barbs

Let $C \Downarrow_{\Pi}$ mean that component $C$ can broadcast a message with a predicate $\Pi$ (i.e., $C \xrightarrow{\nu\bar{\Gamma}:(\bar{\nu}\bar{\Pi})} \text{ where } [\Pi] \neq \text{ff}$). We write $C \Downarrow_{\Pi}$ if $C \rightarrow^* C' \Downarrow_{\Pi}$.

Weak Reduction Barbed Congruence Relations

A Weak Reduction Barbed Relation is a symmetric relation $R$ over the set of $AbC$-components which is barb-preserving, reduction-closed, and context-closed.

Barbed Bisimilarity

Two components are weakly reduction barbed congruent, written $C_1 \simeq C_2$, if $(C_1, C_2) \in R$ for some weak reduction barbed congruent relation $R$. The strong reduction congruence “$\simeq$” is obtained in a similar way by replacing $\Downarrow$ with $\Downarrow$ and $\rightarrow^*$ with $\rightarrow^\ast$. 
Bisimulation for \textit{AbC} Components

**Weak Labelled Bisimulation**

A symmetric binary relation \( \mathcal{R} \) over the set of \textit{AbC}-components is a weak bisimulation if for every action \( \gamma \), whenever \((C_1, C_2) \in \mathcal{R}\) and

- \( \gamma \) is of the form \( \tau \), \( \Gamma : (\tilde{v}) \circ \Pi \), or \( (\nu \tilde{x} \Gamma : (\tilde{v}) \circ \Pi \) with \( [\Pi] \neq \text{ff} \), it holds that \( C_1 \xrightarrow{\gamma} C'_1 \) implies \( C_2 \xrightarrow{\hat{\gamma}} C'_2 \) and \((C'_1, C'_2) \in \mathcal{R}\)

**Bisimilarity**

Two components \( C_1 \) and \( C_2 \) are weak bisimilar, written \( C_1 \approx C_2 \) if there exists a weak bisimulation \( \mathcal{R} \) relating them. Strong bisimilarity, \( \sim \), is defined in a similar way by replacing \( \Rightarrow \) with \( \rightarrow \).

**Bisimilarity and Barbed Congruence do coincide**

\( C_1 \cong C_2 \) if and only if \( C_1 \approx C_2 \).
We are currently continuing our investigations by:

- Looking for simpler operators and semantics, aiming at minimality;
- Evaluating the relative expressiveness of the calculus
- Developing quantitative variants to support components in taking decisions (e.g. via probabilistic model checking).
- Considering alternative semantics and behavioural equivalences for AbC
- Studying the impact of bisimulation (algebraic laws, proof techniques, …)
- Implementing the calculus and devising a full fledged language based on it, as alternative to SCEL.
Many thanks for your time.

Questions?