Relating Adaptable Processes and Compensations

Ongoing Work - Based on an STSM with Jovana Dedeić

Jorge A. Pérez

BETTY Meeting
London, April 17, 2015

Our Goal

To establish relationships of relative expressiveness between compensable and adaptable processes, including session types. This should enable sound transference of techniques.

During the STSM we understood how adaptable processes can encode compensable processes (untyped setting, this talk).

We still need to understand how types play role in encodability (a forthcoming STSM).

This Talk

An intuitive description of the work done so far.

Our Goal

To establish relationships of relative expressiveness between compensable and adaptable processes, including session types. This should enable sound transference of techniques.

During the STSM we understood how adaptable processes can encode compensable processes (untyped setting, this talk).

We still need to understand how types play role in encodability (a forthcoming STSM).

This Talk

An intuitive description of the work done so far.

Context: Compensable Processes

- Important in Web services and business process languages.
- Recover a consistent state in case a transaction aborts.
- Several calculi/constructs proposed. Lanese et al (ESOP'10) have studied their relative expressivity, in an untyped setting.
- On top of the π-calculus, they consider transaction scopes and protected blocks. Three approaches to recovery:
  (a) static (kill current behavior, add a protected compensation)
  (b) parallel (compensation reuses part of current behavior, in parallel)
  (c) general dynamic (as before, but with arbitrary contexts)
- Results: (a) - (b) equally expressive; (c) more expressive than (a).

Context: Compensable Processes

- Important in Web services and business process languages.
- Recover a consistent state in case a transaction aborts.
- Several calculi/constructs proposed. Lanese et al (ESOP'10) have studied their relative expressivity, in an untyped setting.
- On top of the π-calculus, they consider transaction scopes and protected blocks. Three approaches to recovery:
  (a) static (kill current behavior, add a protected compensation)
  (b) parallel (compensation reuses part of current behavior, in parallel)
  (c) general dynamic (as before, but with arbitrary contexts)
- Results: (a) - (b) equally expressive; (c) more expressive than (a).

Context: Adaptable Processes

- A process calculi approach to evolvability, in a broad sense. Proposed by Bravetti et al (FORTE'11, LMCS'12).
- Studied from several perspectives, including expressiveness, decidability/verification, and session types (SAC'13, WSFM'14).
- Runtime modifications to (located) process behaviors, upon exceptional circumstances – not necessarily negative.
- Very simple formulation: higher-order process passing.

Compensable and Adaptable Processes

Some similarities:

- In Compensable Processes (with static recovery):

$$\bar{t} \parallel t[P\,;\,Q] \xrightarrow{\tau} \langle Q \rangle$$

- In Adaptable Processes ($\mathcal{C}$ denotes a context - location nesting):

$$\overline{loc}\{(x)Q\}.R \parallel \mathcal{C}\big[loc[P]\big] \longrightarrow R \parallel \mathcal{C}\big[Q\{P/x\}\big]$$

Some differences:

- In CPs a default behavior comes with an exceptional behavior. In APs there is a separation between them.
- In CPs a name identifies both behaviors, but also the signal that triggers modifications. In APs location names are orthogonal.
- In CPs protected blocks influence the runtime semantics of compensations at runtime. There is no analogous in APs.

Compensable and Adaptable Processes

Some similarities:

- In Compensable Processes (with static recovery):

$$\bar{t} \parallel t[P\,;\,Q] \xrightarrow{\tau} \langle Q \rangle$$

- In Adaptable Processes ($\mathcal{C}$ denotes a context - location nesting):

$$\overline{loc}\{(x)Q\}.R \parallel \mathcal{C}\big[loc[P]\big] \longrightarrow R \parallel \mathcal{C}\big[Q\{P/x\}\big]$$

Some differences:

- In CPs a default behavior comes with an exceptional behavior. In APs there is a separation between them.
- In CPs a name identifies both behaviors, but also the signal that triggers modifications. In APs location names are orthogonal.
- In CPs protected blocks influence the runtime semantics of compensations at runtime. There is no analogous in APs.

The Encoding, By Example

We have obtained a correct encoding of Compensable Processes into Adaptable Processes. Our base language is CCS.

From the perspective of Adaptable Processes, the encoding is challenging due to two main reasons:

- Inner abortion: the default behavior may "commit suicide".
- Protected blocks: after compensation, some portions of the involved processes need be persistently protected.

The Encoding, By Example

We have obtained a correct encoding of Compensable Processes into Adaptable Processes. Our base language is CCS.

From the perspective of Adaptable Processes, the encoding is challenging due to two main reasons:

- Inner abortion: the default behavior may "commit suicide".
- Protected blocks: after compensation, some portions of the involved processes need be persistently protected.

The Encoding, By Example (1)

- A representative transition in CPs with static recovery:

$$\bar{t}.\mathbf{0} \parallel t\big[P\,;\,Q\big] \parallel R \xrightarrow{\tau} \langle Q\rangle \parallel R$$

- The current default behavior is killed; the compensable behavior is triggered inside a protected block.
- The encoding of $t[P\,;\,Q]$ as APs is in two processes in parallel. One for encoding for default behavior P, another for encoding for the compensable behavior Q.

The Encoding, By Example (1)

Given a process in CPs:

$$S = t\big[P \,;\, Q\big]$$

We have the following encoding in APs:

$$\begin{aligned} [\![S]\!] &= t\big[[\![P]\!]\big] \parallel \{\!\{Q\}\!\} \parallel t.\overline{l_t}.k_t.\mathbf{0} \\ &= t\big[[\![P]\!]\big] \parallel l_t.\overline{m_t}.p_t\big[[\![Q]\!]\big] \parallel m_t.\overline{k_t}.t\{\mathbf{0}\} \parallel t.\overline{l_t}.k_t.\mathbf{0} \end{aligned}$$

We then have that

$$[\![\overline{t}.\mathbf{0} \parallel t\big[P \,;\, Q\big] \parallel R]\!] \longrightarrow^{*} p_t\big[[\![Q]\!]\big] \parallel [\![R]\!]$$

The Encoding, By Example (1)

Given a process in CPs:

$$S = t\big[P \,;\, Q\big]$$

We have the following encoding in APs:

$$\begin{aligned}\llbracket S \rrbracket &= t\big[\llbracket P \rrbracket\big] \parallel \{\!\{Q\}\!\} \parallel t.\overline{l_t}.k_t.\mathbf{0} \\ &= t\big[\llbracket P \rrbracket\big] \parallel l_t.\overline{m_t}.p_t\big[\llbracket Q \rrbracket\big] \parallel m_t.\overline{k_t}.t\{\mathbf{0}\} \parallel t.\overline{l_t}.k_t.\mathbf{0}\end{aligned}$$

We then have that

$$\llbracket \overline{t}.\mathbf{0} \parallel t\big[P \,;\, Q\big] \parallel R \rrbracket \longrightarrow^{*} p_t\big[\llbracket Q \rrbracket\big] \parallel \llbracket R \rrbracket$$

The Encoding, By Example (1)

Given a process in CPs:

$$S = t\big[P\,;\,Q\big]$$

We have the following encoding in APs:

$$\begin{aligned}[\![S]\!] &= t\big[[\![P]\!]\big] \parallel \{\!\{Q\}\!\} \parallel t.\overline{l_t}.k_t.\mathbf{0} \\ &= t\big[[\![P]\!]\big] \parallel l_t.\overline{m_t}.p_t\big[[\![Q]\!]\big] \parallel m_t.\overline{k_t}.t\{\mathbf{0}\} \parallel t.\overline{l_t}.k_t.\mathbf{0}\end{aligned}$$

We then have that

$$[\![\overline{t}.\mathbf{0} \parallel t\big[P\,;\,Q\big] \parallel R]\!] \longrightarrow^* p_t\big[[\![Q]\!]\big] \parallel [\![R]\!]$$

The Encoding, By Example (2)

- Abortion entails running the prescribed compensable behavior, and keeping protected blocks (if any) from the default behavior.
- A representative transition in CPs with parallel recovery:

$$t\Big[\bar{t}.\mathbf{0} \parallel P_1 \parallel \langle P_2 \rangle \,;\, Q\Big] \parallel R \xrightarrow{\tau} \langle P_2 \rangle \parallel \langle Q \rangle \parallel R$$

The encoding of $t[P\,;\,Q]$ as APs is in three processes in parallel:

(a) Encoding for default behavior P

(b) Encoding for compensable behavior Q

(c) A "collector" of protected blocks in P that joins them to Q

The Encoding, By Example (2)

- Abortion entails running the prescribed compensable behavior, and keeping protected blocks (if any) from the default behavior.
- A representative transition in CPs with parallel recovery:

$$t\Big[\bar{t}.\mathbf{0} \parallel P_1 \parallel \langle P_2 \rangle \,;\, Q\Big] \parallel R \xrightarrow{\tau} \langle P_2 \rangle \parallel \langle Q \rangle \parallel R$$

The encoding of $t[P\,;\,Q]$ as APs is in three processes in parallel:

(a) Encoding for default behavior P

(b) Encoding for compensable behavior Q

(c) A “collector” of protected blocks in P that joins them to Q

The Encoding, By Example (2)

Given a process in CPs:

$$S = t\Big[\bar{t}.\mathbf{0} \parallel P_1 \parallel \langle P_2 \rangle \,;\, Q\Big]$$

We have the following encoding in APs (note: y_1, y_2 dummy vars)

$$\begin{aligned}
\llbracket S \rrbracket = \;& t\big[\llbracket \bar{t}.\mathbf{0} \parallel P_1 \parallel \langle P_2 \rangle \rrbracket\big] \parallel \{\!\{ Q \}\!\} \parallel f_t(P) \\
= \;& t\Big[\llbracket \bar{t}.\mathbf{0} \rrbracket \parallel \llbracket P_1 \rrbracket \parallel p_t\big[\llbracket P_2 \rrbracket\big]\Big] \parallel \{\!\{ Q \}\!\} \parallel f_t(P) \\
= \;& t\Big[\bar{t}.\mathbf{0} \parallel \llbracket P_1 \rrbracket \parallel p_t\big[\llbracket P_2 \rrbracket\big]\Big] \\
\parallel \;& l_t.p_t\Big\{(x)\, z\big\{(y_1)\, p_t[x] \parallel \overline{m_t}.p_t\big[\llbracket Q \rrbracket\big]\big\}\Big\}.\big(z[\mathbf{0}] \parallel m_t.\overline{k_t}.t\{(y_2)\mathbf{0}\}\big) \\
\parallel \;& t.\overline{l_t}.k_t.\mathbf{0}
\end{aligned}$$

The Encoding, By Example (2)

Given a process in CPs:

$$S = t\Big[\overline{t}.\mathbf{0} \parallel P_1 \parallel \langle P_2 \rangle \; ; \; Q\Big]$$

We have the following encoding in APs (note: y_1, y_2 dummy vars)

$$\begin{aligned}
[\![S]\!] &= t\big[[\![\,\overline{t}.\mathbf{0} \parallel P_1 \parallel \langle P_2\rangle]\!]\,\big] \parallel \{\!\{Q\}\!\} \parallel f_t(P) \\
&= t\Big[[\![\,\overline{t}.\mathbf{0}]\!] \parallel [\![P_1]\!] \parallel p_t\big[[\![P_2]\!]\big]\Big] \parallel \{\!\{Q\}\!\} \parallel f_t(P) \\
&= t\Big[\overline{t}.\mathbf{0} \parallel [\![P_1]\!] \parallel p_t\big[[\![P_2]\!]\big]\Big] \\
&\parallel \; l_t.p_t\Big\{(x)\, z\big\{(y_1)\, p_t[x] \parallel \overline{m_t}.p_t\big[[\![Q]\!]\big]\big\}\Big\}.(z[\mathbf{0}] \parallel m_t.\overline{k_t}.t\{(y_2)\mathbf{0}\}) \\
&\parallel \; t.\overline{l_t}.k_t.\mathbf{0}
\end{aligned}$$

The Encoding, By Example

Some observations:

- The encoding depends on the number of protected blocks to be collected.
- We need to play with the fact that an adaptation “enters” into the context of the location to be modified.
- Lanese et al consider a sort of priority: the default behavior proceeds as long as there is no pending compensation. We do not yet capture that possibility.
- The encoding is environmentally friendly - it leaves no garbage. Pleasant operational correspondences.

Future Plans

- Compensations with general recovery (possible, still not fully formalized).
- We would like to move our encoding to the session typed setting. Type preserving encodings: usually an extra layer of difficulty.
- APs have session types (SAC'13, WS-FM'14) which ensure safety and consistency (no session disruption due to updates).
- Lanese et al study the untyped setting.
  We plan to explore the encodability of (binary) session types with exceptions (Carbone et al, ESOP'07).

Relating Adaptable Processes and Compensations

Ongoing Work - Based on an STSM with Jovana Dedeić

Jorge A. Pérez

university of groningen 2014 | 400 years

BETTY Meeting
London, April 17, 2015