On the Expressiveness of Joining

Thomas Given-Wilson and Axel Legay

Inria

ICE, Grenoble, 5th June, 2015

Overview

- Introduction
- Process Calculi (by example)
- Encodings
- Joining vs Binary
- Joining and Synchronicity
- Joining and Arity
- Joining and Communication Medium
- Joining and Pattern Matching
- Conclusions

Background

Expressiveness in process calculi based on their communication primitives have been widely explored [14, 3, 4, 10, 7, 8]. In [10, 8] this is detailed by combinations of four features: *synchronicity*, *arity*, *communication medium*, and *pattern matching*.

Another kind of communication is *joining* such as in the join calculus [5], general rendezvous calculus [2], and m-calculus [17]. These shift away from *binary* interaction between processes, instead allowing any number of processes to coordinate in a single interaction.

Coordination

Here specific calculi are abstracted away in the style of [10, 8] and *coordination* is considered as a new feature.

The *binary* languages that have interaction in the style of π-calculus [12, 13]

$$\overline{a}\langle b\rangle.P \mid a(x).Q \quad \longmapsto \quad P \mid \{b/x\}Q$$

with a single output and input.

The *joining* languages that have interactions in the style of join calculus

$$\overline{a}\langle b\rangle \mid \overline{c}\langle d\rangle \mid (a(x) \mid c(y)) \rhd R \quad \longmapsto \quad \{b/x, d/y\}R$$

here for example with two outputs interacting with a single join.

Features

There are five features being considered which make up each language whose generic element is denoted as $\mathcal{L}_{\alpha,\beta,\gamma,\delta,\epsilon}$ where:

- $\alpha = A$ for asynchronous communication, and $\alpha = S$ for synchronous communication.
- $\beta = M$ for monadic data, and $\beta = P$ for polyadic data.
- $\gamma = D$ for dataspace-based communication, and $\gamma = C$ for channel-based communications.
- $\delta = NO$ for no matching capability, $\delta = NM$ for name-matching, and $\gamma = I$ for intensionality.
- $\epsilon = B$ for binary communication, and $\epsilon = J$ for joining communication.

$-$ is used when the instantiation of a feature is unimportant.

For example: $\mathcal{L}_{S,M,C,NO,B}$ is synchronous monadic π-calculus; $\mathcal{L}_{A,P,D,NM,B}$ is LINDA[6]; $\mathcal{L}_{S,M,C,I,B}$ represents Psi Calculi [1]; and $\mathcal{L}_{A,P,C,NO,J}$ the join calculus.

Generalising Names

The no-matching languages communicate only using *names*

$$\overline{a}\langle b, c\rangle.Q \mid a(x, y).P \longmapsto Q \mid \{b/x, c/y\}P$$

where $a, b, \ldots$ are used for channels and output, and $x, y, \ldots$ for binding/input.

The name-matching languages extend to allow the *name match* $\ulcorner a \urcorner$ that tests equality as part of a *name match pattern* $m, n, \ldots$

$$\langle a, b\rangle \mid (\ulcorner c \urcorner, x).P \longmapsto \{b/x\}P \qquad a = c\,.$$

Intensional languages support *compounds* $a \bullet b$ for example

$$P = \langle a \bullet b\rangle.P' \quad Q = (x \bullet y).Q' \quad R = (z).R' \quad S = (\ulcorner a \urcorner \bullet \ulcorner b \urcorner).S'\,.$$

these process can be combined to form three possible reductions:

$$P \mid Q \longmapsto P' \mid \{a/x, b/y\}Q' \qquad P \mid R \longmapsto P' \mid \{a \bullet b/z\}R'$$

$$P \mid S \longmapsto P' \mid S'\,.$$

Languages

The (parametric) syntax for the languages is:

$$P, Q, R \quad ::= \quad \mathbf{0} \mid \mathit{OutProc} \mid \mathit{InProc} \mid (\nu n)P \mid P|Q \\ \mid \textbf{if } s = t \textbf{ then } P \textbf{ else } Q \mid *P \mid \surd .$$

Interaction is defined by the following axiom for binary

$$\overline{s}\langle\widetilde{t}\rangle.P \mid s(\widetilde{q}).Q \quad \longmapsto \quad P \mid \sigma Q \qquad\qquad \textsc{Match}(\widetilde{t};\widetilde{q}) = \sigma$$

and the following (generalised) for the joining languages:

$$\overline{s_1}\langle\widetilde{t_1}\rangle.P_1 \mid \overline{s_2}\langle\widetilde{t_2}\rangle.P_2 \mid (s_1(\widetilde{q_1}) \mid s_2(\widetilde{q_2})) \rhd Q \quad \longmapsto \quad P_1 \mid P_2 \mid \sigma Q$$

$$\text{where} \quad \sigma = \textsc{Match}(\widetilde{t_1};\widetilde{q_1}) \cup \textsc{Match}(\widetilde{t_2};\widetilde{q_2}) \ .$$

Encodings

Definition (Valid Encoding)

An encoding of $\mathcal{L}_1$ into $\mathcal{L}_2$ is *valid* if it satisfies:

1. *Compositionality:*
   $[\![\mathrm{op}(S_1, \ldots, S_k)]\!] = \mathcal{C}^N_{\mathrm{op}}([\![S_1]\!]; \ldots; [\![S_k]\!])$.
2. *Name invariance:*
   $[\![\sigma S]\!] = \sigma' [\![S]\!]$ if σ is injective, and $\simeq_2 \sigma' [\![S]\!]$ otherwise.
3. *Operational correspondence:*
   - for all $S \Longmapsto_1 S'$, it holds that $[\![S]\!] \Longmapsto_2 \simeq_2 [\![S']\!]$;
   - for all $[\![S]\!] \Longmapsto_2 T$, there exists S' such that $S \Longmapsto_1 S'$ and $T \Longmapsto_2 \simeq_2 [\![S']\!]$.
4. *Divergence reflection:* for every S such that $[\![S]\!] \longmapsto_2^\omega$, it holds that $S \longmapsto_1^\omega$.
5. *Success sensitiveness:* for every S, it holds that $S \Downarrow_1$ if and only if $[\![S]\!] \Downarrow_2$.

Joining vs Binary

Binary communication turns out to be strictly less expressive than joining communication. That is, every binary language can be encoded into a joining language, and no joining language can be encoded into a binary language.

To illustrate going from binary into joining, consider the encoding $[\![\cdot]\!]$ from $\mathcal{L}_{S,M,C,NO,B}$ into $\mathcal{L}_{S,M,C,NO,J}$ that is the identity on all process forms except for output and input which are encoded as follows:

$$\begin{aligned} [\![\overline{a}\langle b\rangle.P]\!] &\stackrel{\text{def}}{=} \overline{a}\langle b\rangle.[\![P]\!] \\ [\![a(x).Q]\!] &\stackrel{\text{def}}{=} (a(x)) \triangleright [\![Q]\!] . \end{aligned}$$

Theorem

The encoding $[\![\cdot]\!]$ from $\mathcal{L}_{S,M,C,NO,B}$ into $\mathcal{L}_{S,M,C,NO,J}$ is valid.

Binary into Joining

This simple approach holds for all the binary languages to allow an encoding into their corresponding joining languages.

Theorem

There exists a valid encoding $[\![\cdot]\!]$ *from any binary language* $\mathcal{L}_{\alpha,\beta,\gamma,\delta,B}$ *into the corresponding joining language* $\mathcal{L}_{\alpha,\beta,\gamma,\delta,J}$.

Joining into Binary

In the other direction it turns out to be impossible to encode any joining language into a binary language. The key to this proof is to consider the *coordination degree* $\mathrm{CD}(\mathcal{L})$ of a language $\mathcal{L}$ that is the least upper bound on the number of processes required to yield a reduction.

Observe that for the binary languages the coordination degree is always 2, while for the joining languages the coordination degree is ∞.

Theorem

If $\mathrm{CD}(\mathcal{L}_1) > \mathrm{CD}(\mathcal{L}_2)$ *then there exists no valid encoding* $[\![\cdot]\!]$ *from* $\mathcal{L}_1$ *into* $\mathcal{L}_2$.

No Joining into Binary

Proof Sketch.

Pick i processes S_1 to S_i where $i = \mathrm{CD}(\mathcal{L}_2) + 1$ such that $S_1 \mid \ldots \mid S_i \longmapsto \surd$ but not if any S_j (for $1 \leq j \leq i$) is replaced by the null process $\mathbf{0}$. By validity of the encoding it must be that $[\![S_1 \mid \ldots \mid S_i]\!] \longmapsto$ and $[\![S_1 \mid \ldots \mid S_i]\!] \Downarrow$.
Now consider the reduction $[\![S_1 \mid \ldots \mid S_i]\!] \longmapsto$ that can be at most between $i - 1$ processes by the coordination degree of $\mathcal{L}_2$. If the reduction does *not* involve some process $[\![S_j]\!]$ then it follows that $[\![S_1 \mid \ldots \mid S_{j-1} \mid \mathbf{0} \mid S_{j+1} \mid \ldots \mid S_i]\!] \longmapsto$. By construction of $S_1 \mid \ldots \mid S_i$ and $\mathrm{CD}(\mathcal{L}_2) < i$ there must exist some such S_j. However, this contradicts the validity of the encoding since $S_1 \mid \ldots \mid S_{j-1} \mid \mathbf{0} \mid S_{j+1} \mid \ldots \mid S_i \not\longmapsto$. □

Corollary

There exists no valid encoding from $\mathcal{L}_{-,-,-,-,J}$ into $\mathcal{L}_{-,-,-,-,B}$.

Aside: Prior Work

There exists an encoding from join calculus into π-calculus [5], however it is not a valid encoding. The weakness can best be illustrated by considering the encoding of a join:

$$[\![(a(x) \mid b(y)) \triangleright P]\!] \stackrel{\text{def}}{=} a(x).b(y).[\![P]\!] .$$

This can easily fail since

$$(a(x) \mid b(y)) \triangleright \Omega \mid (b(w) \mid a(z)) \triangleright \surd \mid \overline{a}\langle c \rangle \mid \overline{b}\langle d \rangle$$

can reduce to yield divergence or success, but it's encoding:

$$a(x).b(y).[\![\Omega]\!] \mid b(w).a(z).[\![\surd]\!] \mid \overline{a}\langle c \rangle \mid \overline{b}\langle d \rangle$$

can deadlock and then neither diverge or succeed:

$$\begin{aligned} & a(x).b(y).[\![\Omega]\!] \mid b(w).a(z).[\![\surd]\!] \mid \overline{a}\langle c \rangle \mid \overline{b}\langle d \rangle \\ \longmapsto \quad & b(y).[\![\Omega]\!] \mid b(w).a(z).[\![\surd]\!] \mid \overline{b}\langle d \rangle \\ \longmapsto \quad & b(y).[\![\Omega]\!] \mid a(z).[\![\surd]\!] . \end{aligned}$$

Joining and Synchronicity

For many synchronous languages there is a way to encode them into asynchronous languages. For example, the encoding technique [11] from $\mathcal{L}_{S,M,C,NO,B}$ into $\mathcal{L}_{A,M,C,NO,B}$ given by

$$\begin{aligned} [\![\, \overline{n}\langle a\rangle.P \,]\!] &\stackrel{\text{def}}{=} (\nu z)(\overline{n}\langle z\rangle \mid z(x).(\overline{x}\langle a\rangle \mid [\![\, P \,]\!])) \\ [\![\, n(a).Q \,]\!] &\stackrel{\text{def}}{=} (\nu x)n(z).(\overline{z}\langle x\rangle \mid x(a).[\![\, Q \,]\!]) \end{aligned}$$

can be adapted in the obvious manner for $\mathcal{L}_{S,M,C,NO,J}$ into $\mathcal{L}_{A,M,C,NO,J}$ as follows

$$\begin{aligned} [\![\, \overline{n}\langle a\rangle.P \,]\!] &\stackrel{\text{def}}{=} (\nu z)(\overline{n}\langle z\rangle \mid (z(x)) \rhd (\overline{x}\langle a\rangle \mid [\![\, P \,]\!])) \\ [\![\, (n_1(a_1) \mid \ldots \\ \quad \mid n_i(a_i)) \rhd Q \,]\!] &\stackrel{\text{def}}{=} (\nu x_1, \ldots, x_i)(n_1(z_1) \mid \ldots \mid n_i(z_i)) \rhd \\ &\qquad (\overline{z_1}\langle x_1\rangle \mid \ldots \mid \overline{z_i}\langle x_i\rangle \\ &\qquad \mid (x_1(a_1) \mid \ldots \mid x_i(a_i)) \rhd [\![\, Q \,]\!]) \,. \end{aligned}$$

Valid Synchronous Communication

Indeed this approach can be proven valid, the only non-trivial aspect is reasoning over the number of reductions required by the encoding, which is determined by the number of inputs in the encoded join.

Theorem

There is a valid encoding from $\mathcal{L}_{S,M,C,NO,J}$ into $\mathcal{L}_{A,M,C,NO,J}$.

This yields the following general result.

Corollary

If there exists a valid encoding from $\mathcal{L}_{S,\beta,\gamma,\delta,B}$ into $\mathcal{L}_{A,\beta,\gamma,\delta,B}$ then there exists a valid encoding from $\mathcal{L}_{S,\beta,\gamma,\delta,J}$ into $\mathcal{L}_{A,\beta,\gamma,\delta,J}$.

Joining Doesn't Yield New Encodings

However, joining does not yield new encodings of synchronous communication into asynchronous communication.

Theorem

There exists no valid encoding from $\mathcal{L}_{S,M,D,NM,J}$ *into* $\mathcal{L}_{A,M,D,NM,J}$.

Extremely Simplified Proof Sketch.

Consider $P = ((x)) \triangleright$ **if** $x = a$ **then** Ω **else** $\surd$ and $Q = \langle a \rangle.\mathbf{0}$. Since $P \mid Q \longmapsto^{\omega}$ by validity of the encoding $[\![P \mid Q]\!] \longmapsto^{\omega}$ and, given the substitution $\sigma = \{c/a\}$ then $P \mid \sigma Q \longmapsto \surd$ and so $[\![P \mid Q]\!] \Downarrow$.

Now it can be shown that $P \mid Q \mid \sigma(P \mid Q)$ can either diverge or succeed, however $[\![P \mid Q \mid \sigma(P \mid Q)]\!]$ can be shown to be able to both diverge and succeed together. $\square$

Coordination is Orthogonal to Synchronicity

This result can be generalised to show that if there exists no encoding from a language $\mathcal{L}_{S,\beta_1,\delta_1,\gamma_1,B}$ into $\mathcal{L}_{A,\beta_2,\delta_2,\gamma_2,B}$ then there is no encoding from $\mathcal{L}_{S,\beta_1,\delta_1,\gamma_1,J}$ into $\mathcal{L}_{A,\beta_2,\delta_2,\gamma_2,J}$.

Theorem

There exists no valid encoding from $\mathcal{L}_{S,\beta,D,NO,J}$ into $\mathcal{L}_{A,\beta,D,NO,J}$.

Thus it turns out that coordination and synchronicity are orthogonal features.

Joining and Arity

Similarly it turns out that coordination is unrelated to arity, despite their being some similarity in that both have a base case (binary or monadic) and an unbounded case (joining or polyadic, respectively).

This is illustrated in the following result that considers the simplest example.

Theorem

There exists no valid encoding from $\mathcal{L}_{A,P,D,NO,B}$ into $\mathcal{L}_{A,M,D,NO,J}$.

Proof Sketch.

Consider the processes $P = \langle a, b \rangle$ and $Q = (x, y).\surd$. It holds that $P \mid Q \longmapsto \surd$ and so $[\![P \mid Q]\!] \longmapsto$ and $[\![P \mid Q]\!] \Downarrow$ by validity of the encoding, now consider the reduction $[\![P \mid Q]\!] \longmapsto$.

The reduction must be of the form $\langle m_1 \rangle \mid \ldots \mid \langle m_i \rangle \mid ((z_1) \mid \ldots \mid (z_i)) \rhd T'$ for some $\tilde{m}$ and $\tilde{z}$ and i and T'. Now consider the process whose encoding produces $((z_1) \mid \ldots \mid (z_i)) \rhd T'$, assume Q although the results do not rely on this assumption. If any $\langle m_j \rangle$ are also from the encoding of Q then it follows that the encoding of i instances of Q in parallel will reduce, i.e. $[\![Q \mid \ldots \mid Q]\!] \longmapsto$, while $Q \mid \ldots \mid Q \not\longmapsto$.

Now consider two fresh processes S and T such that $S \mid T \longmapsto$ with some arity that is not 2 and $S \not\longmapsto$ and $T \not\longmapsto$. It follows that $[\![S \mid T]\!] \longmapsto$ (and $[\![S]\!] \not\longmapsto$ and $[\![T]\!] \not\longmapsto$) and $[\![S \mid T]\!]$ must include at least one $\langle n \rangle$ to do so. This $\langle n \rangle$ must arise from either $[\![S]\!]$ or $[\![T]\!]$, and conclude by showing that the encoding of i instances of either S or T in parallel with Q reduces, while the un-encoded processes do not. $\square$

Generalising Joining and Arity

This provides the basis for the following result, although small modifications are required to account for name-matching, and not being able to exploit channel-based communication.

Corollary

If there exists no valid encoding from $\mathcal{L}_{\alpha,P,\gamma,\delta,B}$ *into* $\mathcal{L}_{\alpha,M,\gamma,\delta,B}$*, then there exists no valid encoding from* $\mathcal{L}_{\alpha,P,\gamma,\delta,\_}$ *into* $\mathcal{L}_{\alpha,M,\gamma,\delta,J}$*.*

Thus conclude that coordination is orthogonal to arity with respect to expressiveness.

Joining and Communication Medium

Again joining turns out to be orthogonal to communication medium and neither can encode the other. The key to this is captured in the following result.

Theorem

There exists no valid encoding from $\mathcal{L}_{A,M,C,NO,B}$ *into* $\mathcal{L}_{A,M,D,NO,J}$.

The proof is very similar to the previous Theorem...

Proof Sketch.

Consider the processes $P = \overline{a}\langle b \rangle$ and $Q = a(x).\surd$. Clearly it holds that $P \mid Q \longmapsto \surd$ and so $[\![P \mid Q]\!] \longmapsto$ and $[\![P \mid Q]\!] \Downarrow$ by validity of the encoding. Now consider the reduction $[\![P \mid Q]\!] \longmapsto$.

The reduction must be of the form $\langle m_1 \rangle \mid \ldots \mid \langle m_i \rangle \mid ((z_1) \mid \ldots \mid (z_i)) \rhd T'$ for some $\widetilde{m}$ and $\widetilde{z}$ and i and T'. Now consider the process whose encoding produces $((z_1) \mid \ldots \mid (z_i)) \rhd T'$, assume Q although the results do not rely on this assumption. If any $\langle m_j \rangle$ are also from the encoding of Q then it follows that the encoding of i instances of Q in parallel will reduce, i.e. $[\![Q \mid \ldots \mid Q]\!] \longmapsto$, while $Q \mid \ldots \mid Q \not\longmapsto$.

Now consider two fresh processes $S = \overline{c}\langle d \rangle$ and $T = c(z).\mathbf{0}$. Since $S \mid T \longmapsto$ it follows that $[\![S \mid T]\!] \longmapsto$ and must include at least one $\langle n \rangle$ to do so. This $\langle n \rangle$ must arise from either $[\![S]\!]$ or $[\![T]\!]$, and conclude by showing that the encoding of i instances of either S or T in parallel with Q reduces, while the un-encoded processes do not. $\square$

Generalising Joining and Communication Medium

Again this provides the basis for the following general result, although small modifications are required to account for name-matching, and arity.

Corollary

If there exists no valid encoding from $\mathcal{L}_{\alpha,\beta,C,\delta,B}$ into $\mathcal{L}_{\alpha,\beta,D,\delta,B}$, then there exists no valid encoding from $\mathcal{L}_{\alpha,\beta,C,\delta,\_}$ into $\mathcal{L}_{\alpha,\beta,D,\delta,J}$.

Thus joining does not allow for encoding channels in a dataspace-based language unless it could already be encoded by some other means.

Joining and Pattern Matching

Pattern-matching is the strongest indicator of expressiveness in the binary languages, with intensional languages being the most expressive, and with name-matching and polyadicity being the next most expressive language group.

However, it turns out that coordination is still orthogonal to pattern-matching, with no expressiveness changes within the languages as a result of joining.

This is split across two main results, one for intensionality and one for name-matching.

Joining and Intensionality

Theorem

There exists no valid encoding from $\mathcal{L}_{A,M,D,I,B}$ *into* $\mathcal{L}_{-,-,-,\delta,J}$ *where* $\delta \neq I$.

Extremely Light Proof Sketch.

Consider the encoding of the processes $S_0 = ((x)) \triangleright \langle m \rangle$ and $S_1 = \langle a \rangle$, clearly $[\![S_0 \mid S_1]\!] \longmapsto$ since $S_0 \mid S_1 \longmapsto$. Now $[\![S_0 \mid S_1]\!] \longmapsto$ has some combined arity k and so define processes S_2 and S_3 that test $2k+1$ names for equality in $S_2 \mid S_3 \longmapsto$, and s.t. $S_2 \mid S_0 \longmapsto \langle m \rangle$.

1. If $[\![S_2 \mid S_0]\!] \longmapsto$ and $[\![S_2 \mid S_3]\!] \longmapsto$ both have arity k then one name is not being tested in the reduction $[\![S_2 \mid S_3]\!] \longmapsto$ and this can be exploited to yield contradiction.
2. Otherwise if $[\![S_2 \mid S_3]\!] \longmapsto$ has arity $j \neq k$ then it must be that $j > k$. However, by adding more names to S_2 and S_3 via a substitution yield $j > j + k$, or that $[\![S_2 \mid S_3]\!]$ diverges.
3. If $[\![S_2 \mid S_0]\!] \longmapsto$ has arity $j \neq k$ then considering $[\![S_2 \mid S_3]\!]$ can be shown to diverge or fail operational correspondence.

□

Joining and Name Matching

Theorem

There is no valid encoding from $\mathcal{L}_{A,M,D,NM,B}$ *into* $\mathcal{L}_{-,-,\gamma,NO,J}$.

Extremely Light Proof Sketch.

Consider the processes $P = \langle a \rangle$ and $Q = (\ulcorner a \urcorner).(\langle b \rangle \mid \surd)$, since $P \mid Q \longmapsto$ and $P \mid Q \Downarrow$ and so $[\![P \mid Q]\!] \longmapsto$ and $[\![P \mid Q]\!] \Downarrow$.

- If $\gamma = D$ then take the substitution $\sigma = \{a/b, b/a\}$ and considering $P \mid \sigma Q$ is sufficient to yield contradiction.
- If $\gamma = C$ the reduction $[\![P \mid Q]\!] \longmapsto$ must check at least one name c related to a. Consider the process $S = (x).S'$, clearly $P \mid S \longmapsto$ and so $[\![P \mid S]\!] \longmapsto$ must occur. Now if $[\![P \mid S]\!] \longmapsto$ checks the name c then name invariance is violated. Consider when $S' =$ **if** $x = a$ **then** Ω and the substitution $\sigma = \{a/b, b/a\}$, clearly $P \mid S \longmapsto^{\omega}$ and $\sigma P \mid S \longmapsto \equiv \mathbf{0}$ and so $[\![P \mid S]\!]$ diverges and $[\![\sigma P \mid S]\!] \Longmapsto \simeq \mathbf{0}$. However, $P \mid \sigma P \mid S \mid Q \longmapsto R$ s.t. either $R \Downarrow$ or $R \longmapsto^{\omega}$ while $[\![P \mid \sigma P \mid S \mid Q]\!] \longmapsto T$ s.t. $T \Downarrow$ and $T \longmapsto^{\omega}$ □

Coordination Unrelated to Pattern Matching

From these two theorems gain the following corollaries.

Corollary

If there exists no valid encoding from $\mathcal{L}_{\alpha,\beta,\gamma,I,B}$ into $\mathcal{L}_{\alpha,\beta,\gamma,\delta,B}$, then there exists no valid encoding from $\mathcal{L}_{\alpha,\beta,\gamma,I,-}$ into $\mathcal{L}_{\alpha,\beta,\gamma,\delta,J}$.

Corollary

If there exists no valid encoding from $\mathcal{L}_{\alpha,\beta,\gamma,NM,B}$ into $\mathcal{L}_{\alpha,\beta,\gamma,\delta,B}$, then there exists no valid encoding from $\mathcal{L}_{\alpha,\beta,\gamma,NM,-}$ into $\mathcal{L}_{\alpha,\beta,\gamma,\delta,J}$.

Concluding that coordination is unrelated to pattern matching.

Conclusions & Future/Ongoing Work

A couple of general results:

- Joining is strictly more expressive than binary communication.
- Coordination as a feature is unrelated to any other feature.

Some areas for ongoing and future work:

- Coordination can also be extended to *splitting* or *full-coordination*.
- Process calculi such as fusion calculus [15] and Concurrent Pattern Calculus [9] have symmetric communication primitives.
- Process calculi such as Concurrent Constraint Programming (CCP) [16] or Psi Calculi [1] include logic, perhaps another feature.
- Discuss the various expressiveness results in a less formal manner, building on various results.

References I

J. Bengtson, M. Johansson, J. Parrow, and B. Victor.

Psi-calculi: a framework for mobile processes with nominal data and logic.
Logical Methods in Computer Science, 7(1), 2011.

L. Bocchi and L. Wischik.

A process calculus of atomic commit.
Electronic Notes in Theoretical Computer Science, 105(0):119 – 132, 2004.
Proceedings of the First International Workshop on Web Services and Formal Methods (WSFM 2004) M. Bravetti and G. Zavattaro.

N. Busi, R. Gorrieri, and G. Zavattaro.

On the expressiveness of linda coordination primitives.
Information and Computation, 156(1-2):90–121, 2000.

R. De Nicola, D. Gorla, and R. Pugliese.

On the expressive power of klaim-based calculi.
Theoretical Computer Science, 356(3):387–421, May 2006.

C. Fournet and G. Gonthier.

The reflexive cham and the join-calculus.
In *IN PROCEEDINGS OF THE 23RD ACM SYMPOSIUM ON PRINCIPLES OF PROGRAMMING LANGUAGES*, pages 372–385. ACM Press.

D. Gelernter.

Generative communication in LINDA.
ACM Transactions on Programming Languages and Systems, 7(1):80–112, 1985.

References II

T. Given-Wilson.

Concurrent Pattern Unification.
PhD thesis, University of Technology, Sydney, Australia, 2012.

T. Given-Wilson.

On the Expressiveness of Intensional Communication.
In *Combined 21th International Workshop on Expressiveness in Concurrency and 11th Workshop on Structural Operational Semantics*, Rome, Italie, Sept. 2014.

T. Given-Wilson, D. Gorla, and B. Jay.

A Concurrent Pattern Calculus.
To appear in: Logical Methods in Computer Science, 2014.

D. Gorla.

Comparing communication primitives via their relative expressive power.
Information and Computation, 206(8):931–952, 2008.

K. Honda and M. Tokoro.

An object calculus for asynchronous communication.
In *ECOOP'91 European Conference on Object-Oriented Programming*, pages 133–147. Springer, 1991.

R. Milner, J. Parrow, and D. Walker.

A calculus of mobile processes, I.
Information and Computation, 100(1):1–40, Sept. 1992.

R. Milner, J. Parrow, and D. Walker.

A calculus of mobile processes, II.
Information and Computation, 100(1):41–77, Sept. 1992.

References III

C. Palamidessi.

Comparing the expressive power of the synchronous and asynchronous pi-calculi.
Mathematical. Structures in Comp. Sci., 13(5):685–719, Oct. 2003.

J. Parrow and B. Victor.

The fusion calculus: expressiveness and symmetry in mobile processes.
In *Proceedings of Thirteenth Annual IEEE Symposium on Logic in Computer Science*, pages 176–185, Jun 1998.

V. A. Saraswat, M. Rinard, and P. Panangaden.

The semantic foundations of concurrent constraint programming.
In *Proceedings of the 18th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '91, pages 333–352, New York, NY, USA, 1991. ACM.

A. Schmitt and J. Stefani.

The m-calculus: a higher-order distributed process calculus.
In A. Aiken and G. Morrisett, editors, *Conference Record of POPL 2003: The 30th SIGPLAN-SIGACT Symposium on Principles of Programming Languages, New Orleans, Louisisana, USA, January 15-17, 2003*, pages 50–61. ACM, 2003.