Mixed Sessions: the Other Side of the Tape

Filipe Casal Andreia Mordido Vasco T. Vasconcelos
LASIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal
fmrcasal@fc.ul.pt afmordido@fc.ul.pt vmvasconcelos@fc.ul.pt

Vasconcelos et al. [11] introduced side A of the tape: there is an encoding of classical sessions into mixed sessions. Here we present side B: there is translation of (a subset of) mixed sessions into classical sessions. We prove that the translation is a minimal encoding, according to the criteria put forward by Kouzapas et al. [6].

1 Classical Sessions, Mixed Sessions

Mixed sessions were introduced by Vasconcelos et al. [11] as an extension of classical session types [3, 5, 10]. They form an interesting point in the design space of session-typed systems: an extremely concise process calculus (four constructors only) that allows the natural expression of algorithms quite cumbersome to write in classical sessions. The original paper on mixed sessions [11] shows that there is an encoding of classical sessions into mixed sessions. This abstract shows that the converse is also true for a fragment of mixed sessions.

A translation of mixed sessions into classical sessions would allow to leverage the tools available for the latter: one could program in mixed sessions, translate the source code into classical sessions, check the validity of the source code against the type system for the target language, and run the original program under an interpreter for classical sessions (SePi [2], for example). A mixed-to-classical encoding would further allow a better understanding of the relative expressiveness of the two languages.

Processes in classical binary sessions [3, 4, 5, 9] (here we follow the formulation in [10]) communicate by exchanging messages on bidirectional channels. We introduce classical sessions by means of a few examples. Each channel is denoted by its two ends and introduced in a process P as $(\mathbf{new}\ xy)P$. Writing a value v on channel end x and continuing as P is written as $x!v.P$. Reading a value from a channel end y, binding it to variable z and continuing as Q is written as $y?z.Q$. When the two processes get together under a new binder that ties together the two ends of the channel, such as in

$$(\mathbf{new}\ xy)\ x!v.P \mid y?z.Q$$

value v is communicated from the x channel end to the y end. The result is process $(\mathbf{new}\ xy)\,P \mid Q[v/z]$, where notation $Q[v/z]$ denotes the result of replacing v for z in Q.

Processes may also communicate by offering and selecting options in choices. The different choices are denoted by labels, ℓ and m for example. To select choice ℓ on channel end x and continue as P we write $x\ \mathbf{select}\ \ell.P$. To offer a collection of options at channel end y and continue with appropriate continuations Q and R, we write $\mathbf{case}\ y\ \mathbf{of}\ \{\ell \to Q, m \to R\}$. When **select** and **case** processes are put together under a **new** that binds together the two ends of a channel, such as in

$$(\mathbf{new}\ xy)\ x\ \mathbf{select}\ \ell.P \mid \mathbf{case}\ y\ \mathbf{of}\ \{\ell \to Q,\ m \to R\}$$

branch Q is selected in the **case** process. The result is the process $(\mathbf{new}\ xy)\,P \mid Q$. Selecting a choice is called an internal choice, offering a collection of choices is called an external choice. We thus see that

S. Balzer, L. Padovani (Eds.): Programming Language Approaches to Concurrency- & Communication-cEntric Software (PLACES 2020)
EPTCS 314, 2020, pp. 46–60, doi:10.4204/EPTCS.314.5

classical sessions comprise four atomic interaction primitives. Furthermore, choices are directional in the sense that one side offers a collection of possibilities, the other selects one of them.

To account for unbounded behavior classical sessions count with replication: an input process that yields a new copy of itself after reduction, written $y*?z.Q$. A process of the form

```
(new xy) x!v.P | y*?z.Q
```

reduces to (new xy) P | Q[v/z] | y*?z.Q. If we use the **lin** prefix to denote an ephemeral process and the **un** prefix to denote a persistent process, an alternative syntax for the above process is (new xy) **lin** x!v.P | **un** y?z.Q.

Mixed sessions blur the distinction between internal and external choice. Under a unified language construct—mixed choice—processes may non-deterministically select one choice from a multiset of output choices, or branch on one choice, again, from a multiset of possible input choices. Together with an output choice, a value is (atomically) sent; together with an input choice, a value is (again, atomically) received, following various proposals in the literature [1, 8, 12]. The net effect is that the four common operations on session types—output, input, selection, and branching—are effectively collapsed into one: mixed choice. Mixed choices can be labelled as ephemeral (linear, consumed by reduction) or persistent (unrestricted, surviving reduction), following conventional versus replicated inputs in some versions of the pi-calculus [7]. Hence, in order to obtain a core calculus, all we have to add is name restriction, parallel composition, and inaction (the terminated process), all standard in the pi-calculus.

We introduce mixed sessions by means of a few examples. Processes communicate by offering/selecting choices with the same label and opposite polarities.

```
(new xy) lin x (m!3.P + n?z.Q) | lin y (m?w.R + n!5.S + p!7.T)
```

The above processes communicate over the channel with ends named x and y and reduce in one step along label m to (new xy) P | R[3/w] or along label n to (new xy) Q[5/z] | S.

Non-determinism in mixed sessions can be further achieved by allowing duplicated labels in choices. An example in which a 3 or a 5 is non-deterministically sent over the channel is

```
(new xy) lin x (m!3.P + m!5.Q) | lin y (m?z.R)
```

This process reduces in one step to either (new xy) P | R[3/z] or (new xy) Q | R[5/z]. Unrestricted behavior in choices is achieved by the **un** qualifier in the choice syntax.

```
(new xy) un x (m!3.P + m!5.P) | un y (m?z.Q)
```

This process reduces to itself together with either of the choices taken,

```
(new xy)
un x (m!3.P + m!5.P) |
un y (m?z.Q) |
P | Q[3/z]
```

or

```
(new xy)
un x (m!3.P + m!5.P) |
un y (m?z.Q) |
P | Q[5/z]
```

The complete set of definitions for the syntax, operational semantics, and type system for mixed sessions are in appendix, Figures 6 to 8. For technical details and main results, we direct the reader to reference [11]. The complete set of definitions for the syntax, operational semantics, and type system for classical sessions are in appendix, Figure 9. For further details, we refer the reader to references [10, 11].

2 Mixed Sessions as Classical Sessions

This section shows that a subset of the language of mixed sessions can be embedded in that of classical sessions. We restrict our attention to choices that reduce against choices with the same qualifier, that is,

```
new x y: lin&{m: !int.end,
              n: ?bool.end}

// lin x (m!3.0 + n?w.0)
case x of
    m → new s1 t1: ∗+{ℓ}
          s1 select ℓ |
          case t1 of
              ℓ → x!3
    n → new s2 t2: ∗+{ℓ}
          s2 select ℓ |
          case t2 of
              ℓ → x?w
|
// lin y (m?z.0)
new s3 t3: ∗+{ℓ}
s3 select ℓ |
case t3 of
    ℓ → y select m.
          new s4 t4: ∗+{ℓ}
          s4 select ℓ |
          case t4 of
              ℓ → y?z
```

Figure 1: Translation of (**new** xy)(**lin** x (m!3.0 + n?w.0) | **lin** y (m?z.0))

we do not consider the case where an ephemeral (lin) process reduces against a persistent (un) one. For this reason, we assume that a process and its type always have the same lin/un qualifier.

One of the novelties in mixed sessions is the possible presence of duplicated label-polarity pairs in choices. This introduces a form of non-determinism that can be easily captured in classical sessions. The NDchoice classical session process creates a race condition on a new channel with endpoints s,t featuring multiple selections on the s endpoint, for only one branch on the t endpoint. This guarantees that exactly one of the branches is non-deterministically selected. The remaining selections must eventually be garbage collected. We assume that $\prod_{1\leq i\leq n} Q_i$ denotes the process $Q_1 \mid \ldots \mid Q_n$ for $n > 0$, and that Π binds tighter than the parallel composition operator.

$$\text{NDChoice}\{P_i\}_{i\in I} = (\nu st)\left(\prod_{i\in I} s \lhd l_i.\mathbf{0} \mid t \rhd \{l_i\colon P_i\}_{i\in I}\right)$$

The type S of channel end s is of the form $\mathsf{un}\oplus\{l_i\colon S\}_{i\in I}$, an equation that can be solved by type $\mu a.\mathsf{un}\oplus\{l_i\colon a\}_{i\in I}$, and which SePi abbreviates to $*\oplus\{l_i\}_{i\in I}$. The qualifier must be un because s occurs in multiple threads in NDChoice; recursion arises because of the typing rules for processes reading or writing in unrestricted channels.

Equipped with NDChoice we describe the translation of mixed sessions to classical sessions via variants of the examples in Section 1. All examples fully type check and run in SePi [2]. To handle duplicated label-polarity pairs in choices, we organize choice processes by label-polarity fragments. Each such fragment represents a part of a choice operation where all possible outcomes have the same label and polarity. When a reduction occurs, one of the branches is taken, non-deterministically, using the NDChoice operator. After a non-deterministic choice of the branch, and depending on the polarity of the fragment, the process continues by either writing on or reading from the original channel.

The translation of choice processes is guided by their types. For each choice we need to know its qualifier (lin, un) and its view ($\oplus$, &), and this information is present in types alone.

Figure 1 shows the translation of the mixed process (**new** xy)(**lin** x (m!3.0 + n?w.0) | **lin** y (m?z.0)), where x is of type **lin**&{m!**int**.**end**, n?**bool**.**end**}. The corresponding type in classical sessions is **lin**&{m:!**int**.**end**, n:?**bool**.**end**}, which should not come as a surprise. Because channel end x is of an external choice type (&), the choice on x is encoded as a **case** process. The other end of the channel, y, is typed as an internal choice ($\oplus$) and is hence translated as a **select** process. Occurrences of the NDChoice process

```
new x y: lin&{m: !int.end}

// lin x (m!3.0 + m!5.0)
case x of
    m → new s1 t1: ∗+{ℓ1,ℓ2}
        s1 select ℓ1 |
        s1 select ℓ2 |
        case t1 of
            ℓ1 → x!3
            ℓ2 → x!5
|
// lin y (m?z.0)
new s2 t2: ∗+{ℓ}
s2 select ℓ |
case t2 of
    ℓ → y select m.
        new s3 t3: ∗+{ℓ}
        s3 select ℓ |
        case t3 of
            ℓ → y?z
```

Figure 2: Translation of (**new** xy)(**lin** x (m!3.0 + m!5.0) | **lin** y (m?z.0))

appear in a degenerate form, always applied to a single branch. We have four of them: three for each of the branches in **case** processes (s_1t_1, s_2t_2, and s_4t_4) and one for the external choice in the mixed session process (s_3t_3).

In general, an external choice is translated into a classical branching (**case**) over the unique labels of the fragments of the process, but where the polarity of each label is inverted. The internal choice, in turn, is translated as (possibly nondeterministic collection of) classical **select** process but keeps the label polarity. This preserves the behavior of the original process: in mixed choices, a reduction occurs when a branch $l^!v.P$ matches another branch $l^?z.Q$ with the same label but with dual polarity ($l^!$ against $l^?$), while in a classical session the labels alone must match (l against l). Needless to say, we could have followed the strategy of dualizing internal choices rather than external.

If we label reduction steps with the names of the channel ends on which they occur, we can see that, in this case a $\xrightarrow{xy}$ reduction step in mixed sessions is mimicked by a long series of classical reductions, namely $\xrightarrow{s_3t_3}\xrightarrow{xy}\xrightarrow{s_1t_1}\xrightarrow{s_4t_4}\xrightarrow{xy}$ or $\xrightarrow{s_3t_3}\xrightarrow{xy}\xrightarrow{s_4t_4}\xrightarrow{s_1t_1}\xrightarrow{xy}$. Notice the three reductions to resolve non-determinism (on s_it_i) and the two reductions on xy to encode branching followed by message passing, an atomic operation in mixed sessions.

Figure 2 shows an example of a mixed choice process with a duplicated label-polarity pair, m!. If we assign to x type **lin**&{m!**int**}, then we know that the choice on x is encoded as **case** and that on y as **select**. In this case, the NDChoice operator is applied in a non-degenerate manner to decide whether to send the values 3 or 5 on x channel end, by means of channel s_1t_1. Again we can see that the one step reduction on channel xy in the original mixed session process originates a sequence of five reduction steps in classical sessions, namely $\xrightarrow{s_2t_2}\xrightarrow{xy}\xrightarrow{s_1t_1}\xrightarrow{s_3t_3}\xrightarrow{xy}$ or $\xrightarrow{s_2t_2}\xrightarrow{xy}\xrightarrow{s_3t_3}\xrightarrow{s_1t_1}\xrightarrow{xy}$. In this case, however, the computation is non-deterministic: the last reduction step may carry integer 3 or 5.

Figure 3 shows the encoding of mixed choices on unrestricted channels. The mixed choice process is that of Figure 2 only that the two ephemeral choices (**lin**) have been replaced by their persistent counterparts (**un**). The novelty, in this case, is the loops that have been created around the **case** and the **select** process. Loops in classical sessions can be implemented with a replicated input: a process of the form v∗?x.P is a persistent process that, when invoked with a value v becomes the parallel composition P[v/x] | v∗?x.P. The general form of the loops we are interested in are (**new** uv : ∗!())(u!() | v∗?x.P), where *continue calls* in process P are of the form u!(). The contents of the messages that control the loop are not of interest and so we use the unit type (), so that u is of type ∗!(). We can easily see the calls

```
type Unr = lin&{m: !integer.end}
new x y: *?Unr

// un x (m!3.0 + m!5.0)
new u1 v1: *!()
u1!() |
v1*?(). x?a.
case a of
    m → new s1 t1: *+{ℓ1,ℓ2}
          s1 select ℓ1 |
          s1 select ℓ2 |
          case t1 of
              ℓ1 → a!3 . u1!()
              ℓ2 → a!5 . u1!()
```

```
// un y (m?z.0)
new u2 v2: *!()
u2!() |
v2*?().
new s t: *+{ℓ}
s select ℓ |
case t of
    ℓ → new a b: Unr
        y!a . b select m .
        new s2 t2: *+{ℓ}
        s2 select ℓ |
        case t2 of
            ℓ → b?z . u2!()
```

Figure 3: Translation of (**new** xy)(**un** x (m!3.0 + m!5.0)| **un** y (m?z.0))

$$
\begin{array}{lll}
(\!|\mathsf{lin}\oplus\{l_i^\star S_i.T_i\}_{i\in I}|\!) = \mathsf{lin}\oplus\{l_i{}^\star\colon \mathsf{lin}\star_i(\!|S_i|\!).(\!|T_i|\!)\}_{i\in I} & & \\
(\!|\mathsf{lin}\,\&\{l_i^\star S_i.T_i\}_{i\in I}|\!) = \mathsf{lin}\,\&\{l_i^\bullet\colon \mathsf{lin}\star_i(\!|S_i|\!).(\!|T_i|\!)\}_{i\in I} & & \text{where } \star_i \perp \bullet_i \\
(\!|\mathsf{un}\oplus\{l_i^\star S_i.T_i\}_{i\in I}|\!) = \mu b.\mathsf{un}!(\mathsf{lin}\oplus\{l_i{}^\star\colon \mathsf{lin}\star_i(\!|S_i|\!).\mathsf{end}\}_{i\in I}).b & & \text{where } T_i \approx \mathsf{un}\oplus\{l_i^\star S_i.T_i\}_{i\in I} \\
(\!|\mathsf{un}\,\&\{l_i^\star S_i.T_i\}_{i\in I}|\!) = \mu b.\mathsf{un}?(\mathsf{lin}\,\&\{l_i^\bullet\colon \mathsf{lin}\star_i(\!|S_i|\!).\mathsf{end}\}_{i\in I}).b & & \text{where } \star_i \perp \bullet_i \text{ and } T_i \approx \mathsf{un}\,\&\{l_i^\star S_i.T_i\}_{i\in I}
\end{array}
$$

(Homomorphic for end, unit, bool, $\mu a.T$, and a)

Figure 4: Translating mixed session types to traditional session types

$u_1!()$ and $u_2!()$ in the last lines in Figure 3, reinstating the unrestricted choice process. In this case, one step reduction in mixed sessions corresponds to a long sequence of transitions in their encodings.

We now present translations for types and processes in general. The translation of mixed choice session types into classical session types is in Figure 4. In general, the (atomic) branch-communicate nature of mixed session types, $\{l_i^\star S_i\}$, is broken in its two parts, $\{l_i\colon \star S_i\}$, branch first, communicate after. In mixed sessions, choice types are labelled by label-polarity pairs ($l^!$ or $l^?$); in classical session choices are labelled by labels alone. Because we want the encoding of a label $l^!$ to match the encoding of $l^?$, we must dualize one of them. We arbitrarily chose do dualize the labels in the & type. The typing rules for classical unrestricted processes of type $S = \mathsf{un}\sharp\{l_i^\star S_i.T_i\}_{i\in I}$ require T_i to be equivalent ($\approx$) to S itself. We take advantage of this restriction when translating un types.

The translation of mixed choice processes is in Figure 5. Since the translation is guided by the type of the process to be translated, we also provide the typing context to the translation function, hence the notation $(\!|\Gamma \vdash P|\!)$. Because label-polarity pairs may be duplicated in choice processes, we organize such processes in label-polarity fragments, so that a process of the form $qx\sum_{i\in I} l_i^{\star_i} v_i.P_i$ (where $q ::= \mathsf{lin} \mid \mathsf{un}$ and $\star ::= ! \mid ?$) can be written as $qx\sum_{i\in I}(\sum_{j\in J} l_i^! v_{ij}.P_{ij} + \sum_{k\in K} l_i^? y_{ik}.P'_{ik})$. Each label-polarity fragment ($l_i^!$ or $l_i^?$) groups together branches with the same label and the same polarity. Such fragments may be empty for external choices, for not all label-polarity pairs in an external choice type need to be covered in the corresponding process (internal choice processes do not need to cover all choices offered by the external

$$(\!|\Gamma \vdash \mathsf{lin}x\sum_{i\in I}(\sum_{j\in J} l_i^!v_{ij}.P_{ij} + \sum_{k\in K} l_i^?y_{ik}.P'_{ik})|\!) = x \rhd \{l_i^?:\ \mathrm{NDChoice}\{x!v_{ij}.(\!|\Gamma_3, x\colon T_i \vdash P_{ij}|\!)\}_{j\in J},$$
$$l_i^!:\ \mathrm{NDChoice}\{x?y_{ik}.(\!|\Gamma_2 \circ \Gamma_3, x\colon T'_i, y_{ik}\colon S'_i \vdash P'_{ik}|\!)\}_{k\in K}\}_{i\in I}$$

where $\Gamma = \Gamma_1 \circ \Gamma_2 \circ \Gamma_3$ and $\Gamma_1 \vdash x\colon \mathsf{lin}\&\{l_i^!S_i.T_i, l_i^?S'_i.T'_i\}_{i\in I}$ and $\Gamma_2 \vdash v_{ij}\colon S_i$.

$$(\!|\Gamma \vdash \mathsf{lin}\,x\sum_{i\in I}(\sum_{j\in J} l_i^!v_{ij}.P_{ij} + \sum_{k\in K} l_i^?y_{ik}.P'_{ik})|\!) = \mathsf{NDChoice}\{x \lhd l_i^!.\mathrm{NDChoice}\{x!v_{ij}.(\!|\Gamma_3, x\colon T_i \vdash P_{ij}|\!)\}_{j\in J},$$
$$x \lhd l_i^?.\mathrm{NDChoice}\{x?y_{ik}.(\!|\Gamma_2 \circ \Gamma_3, x\colon T'_i, y_{ik}\colon S'_i \vdash P'_{ik}|\!)\}_{k\in K}\}_{i\in I}$$

where $\Gamma = \Gamma_1 \circ \Gamma_2 \circ \Gamma_3$ and $\Gamma_1 \vdash x\colon \mathsf{lin}\oplus\{l_i^!S_i.T_i, l_i^?S'_i.T'_i\}_{i\in I}$ and $\Gamma_2 \vdash v_{ij}\colon S_i$.

$$(\!|\Gamma \vdash \mathsf{un}\,x\sum_{i\in I}(\sum_{j\in J} l_i^!v_{ij}.P_{ij} + \sum_{k\in K} l_i^?y_{ik}.P'_{ik})|\!) = (\nu uv)(u!() \mid \mathsf{un}\,v?\_.x?a.$$
$$a \rhd \{l_i^?:\ \mathrm{NDChoice}\{a!v_{ij}.(u!() \mid (\!|\Gamma \vdash P_{ij}|\!))\}_{j\in J},$$
$$l_i^!:\ \mathrm{NDChoice}\{a?y_{ik}.(u!() \mid (\!|\Gamma, y_{ik}\colon S'_i \vdash P'_{ik}|\!))\}_{k\in K}\}_{i\in I})$$

where $\mathsf{un}(\Gamma)$ and $\Gamma \vdash x\colon \mathsf{un}\&\{l_i^!S_i.T_i, l_i^?S'_i.T'_i\}_{i\in I}$ and $\Gamma \vdash v_{ij}\colon S_i$ and $T_i \approx T'_i \approx \mathsf{un}\sharp\{l_i^!S_i.T_i, l_i^?S'_i.T'_i\}_{i\in I}$.

$$(\!|\Gamma \vdash \mathsf{un}\,x\sum_{i\in I}(\sum_{j\in J} l_i^!v_{ij}.P_{ij} + \sum_{k\in K} l_i^?y_{ik}.P'_{ik})|\!) = (\nu uv)(u!() \mid \mathsf{un}\,v?\_.$$
$$\mathsf{NDChoice}\{(\nu ab)x!a.b \lhd l_i^!.\mathrm{NDChoice}\{b!v_{ij}.(u!() \mid (\!|\Gamma \vdash P_{ij}|\!))\}_{j\in J},$$
$$(\nu ab)x!a.b \lhd l_i^?.\mathrm{NDChoice}\{b?y_{ik}.(u!() \mid (\!|\Gamma, y_{ik}\colon S'_i \vdash P'_{ik}|\!))\}_{k\in K}\}_{i\in I})$$

where $\mathsf{un}(\Gamma)$ and $\Gamma \vdash x\colon \mathsf{un}\oplus\{l_i^!S_i.T_i, l_i^?S'_i.T'_i\}_{i\in I}$ and $\Gamma \vdash v_{ij}\colon S_i$ and $T_i \approx T'_i \approx \mathsf{un}\sharp\{l_i^!S_i.T_i, l_i^?S'_i.T'_i\}_{i\in I}$.

$$(\!|\Gamma \vdash (\nu xy)P|\!) = (\nu xy)(\!|\Gamma, x\colon S, y\colon T \vdash P|\!) \qquad \text{where } S \perp T$$
$$(\!|\Gamma_1 \circ \Gamma_2 \vdash P_1 \mid P_2|\!) = (\!|\Gamma_1 \vdash P_1|\!) \mid (\!|\Gamma_2 \vdash P_2|\!)$$
$$(\!|\Gamma \vdash \mathbf{0}|\!) = \mathbf{0}$$
$$(\!|\Gamma_1 \circ \Gamma_2 \vdash \mathsf{if}\,v\,\mathsf{then}\,P_1\,\mathsf{else}\,P_2|\!) = \mathsf{if}\,v\,\mathsf{then}\,(\!|\Gamma_2 \vdash P_1|\!)\,\mathsf{else}\,(\!|\Gamma_2 \vdash P_2|\!) \qquad \text{where } \Gamma_1 \vdash v\colon \mathsf{bool}$$

Figure 5: Translating mixed session processes to classical session processes

counterpart). The essence of the translation is discussed in the three examples above.

We distinguish four cases for choices, according to qualifiers (lin or un) and views ($\oplus$ or &) in types. In all of them an NDChoice process takes care of duplicated label-polarity pairs in branches. Internal choice processes feature an extra occurrence of NDChoice to non-deterministically select between output and input *on the same label*. Notice that external choice must still accept both choices, so that it is not equipped with an NDChoice. Finally, unrestricted mixed choices require the encoding of a loop, accomplished by creating a new channel for the effect (uv), installing a replicated input $\mathsf{un}\,v?\_.P$ at one end of the channel, and invoking the input once to "start" the loop and again at the end of the interaction on channel end x. The calls are all accomplished with processes of the form $u!()$. The contents of the messages are of no interest and so we use the unit value $()$.

Following the encoding for types, the encoding for external choice processes exchanges the polarities of choice labels: a label $l_i^!$ in mixed sessions is translated into $l_i^?$, and vice-versa, in the cases for lin& and un& choices. This allows reduction to happen in classical sessions, where we require an exact match between the label of the **select** process and that of the **case** process.

3 A Minimal Encoding

This section covers typing and operational correspondences; we follow Kouzapas et al. [6] criteria for typed encodings, and aim at a minimal encoding.

Let $\mathscr{C}$ range over classical processes, and $\mathscr{M}_0$ range over the fragment of mixed choice processes where lin processes only reduce against lin processes, and un processes only reduce against un processes, i.e., the reduction rules for $\mathscr{M}_0$ are those for mixed processes, except for [R-LinUn] and [R-UnLin] (Figure 6). The function $(\!|\cdot|\!) : \mathscr{M}_0 \longrightarrow \mathscr{C}$ in Figure 5 denotes a translation from mixed choice processes in $\mathscr{M}_0$ to classical processes in $\mathscr{C}$. We overload the notation and denote by $(\!|\cdot|\!)$ the encoding of both types (Figure 4) and processes (Figure 5).

We start by addressing typing criteria. The *type preservation* criterion requires that $(\!|\mathrm{op}(T_1,\ldots,T_n)|\!) = \mathrm{op}((\!|T_1|\!),\ldots,(\!|T_n|\!))$. Our encoding, in Figure 4, can be called weakly type preserving in the sense that we preserve the direction of type operations, but not the exact type operator. For example, a un$\oplus$ type is translated in a un! type (and un& type is translated in un?). Both $\oplus$ and ! can be seen as output types (and & and ? as input), so that direction is preserved.

We now move to *type soundness*, but before we need to be able to type the NDChoice operator.

Lemma 1. *The following is an admissible typing rule for typing* NDChoice.

$$\frac{\Gamma \vdash P_i \quad i \in I}{\Gamma \vdash \mathrm{NDChoice}\{P_i\}_{i\in I}}$$

Proof. The typing derivation of the expansion of NDChoice leaves open the derivations for $\Gamma \vdash P_i$. □

The type soundness theorem for our translation is item 5 below; the remaining items help in building the main result.

Theorem 2 (Type Soundness)**.**

1. *If* un T, *then* un $(\!|T|\!)$.
2. *If* un Γ, *then* un $(\!|\Gamma|\!)$.
3. *If* $S <: T$, *then* $(\!|S|\!) <: (\!|T|\!)$
4. *If* $\Gamma \vdash v\colon T$, *then* $(\!|\Gamma|\!) \vdash v\colon (\!|T|\!)$.
5. *If* $\Gamma \vdash P$, *then* $(\!|\Gamma|\!) \vdash (\!|\Gamma \vdash P|\!)$.

Proof. 1: By case analysis on T and the fact that types are contractive. 2: By induction on Γ using case 1. 3: By coinduction on the hypothesis. 4: By rule induction on the hypothesis using items 2 and 3. 5: By coinduction on the hypothesis, using items 2 and 4, and lemma 1. □

The *syntax preservation* criterion consists of ensuring that parallel composition is translated into parallel composition and that name restriction is translated into name restriction, which is certainly the case with our translation. It further requires the translation to be name invariant. Our encoding transforms

each channel end in itself and hence is trivially name invariant. We conclude that our translation is syntax preserving.

We now address the criteria related to the operational semantics. We denote by $\Rightarrow$ the reflexive and transitive closure of the reduction relations, $\rightarrow$, in both the source and target languages. Sometimes we use subscript $\mathscr{M}_0$ to denote the reduction of mixed choice processes and the subscript $\mathscr{C}$ for the reduction of classical processes, even though it should be clear from context. The behavioral equivalence $\asymp$ for classical sessions we are interested in extends structural congruence $\equiv$ with the following rule

$$(\nu ab)\prod_{i\in I} a \lhd l_i.\mathbf{0} \asymp \mathbf{0}.$$

The new rule allows collecting processes that are left by the encoding of non-deterministic choice. We call it *extended structural congruence*. The following lemma characterizes the reductions of NDChoice processes: they reduce to one of the processes that are to be chosen and leave an inert term G.

Lemma 3. $\mathsf{NDChoice}\{P_i\}_{i\in I} \rightarrow P_k \mid G \asymp P_k$, *for any* $k \in I$.

Proof. $\mathsf{NDChoice}\{P_i\}_{i\in I} \rightarrow P_k \mid G$, where $G = (\nu st)\prod_{i\in I}^{i\neq k} s \lhd l_i.\mathbf{0}$ and $G \asymp \mathbf{0}$. □

We now turn our attention to *barbs* and *barb preservation*. We say that a typed classical session process P *has a barb in* x, notation $\Gamma \vdash P \downarrow_x$, if $\Gamma \vdash P$ and

- either $P \equiv (\nu x_n y_n)\ldots(\nu x_1 y_1)(x!v.Q \mid R)$ where $x \notin \{x_i, y_i\}_{i=1}^n$
- or $P \equiv (\nu x_n y_n)\ldots(\nu x_1 y_1)(x \lhd l.Q \mid R)$ where $x \notin \{x_i, y_i\}_{i=1}^n$.

On the other hand, we say that a typed mixed session process P *has a barb in* x, notation $\Gamma \vdash P \downarrow_x$, if $\Gamma \vdash P$ and $P \equiv (\nu x_n y_n)\ldots(\nu x_1 y_1)(qx\sum_{i\in I} M_i \mid R)$ where $x \notin \{x_i, y_i\}_{i=1}^n$ and $\Gamma \vdash x\colon q \oplus \{U_i\}_{i\in I}$. Notice that only types can discover barbs in processes since internal choice is indistinguishable from external choice at the process level in $\mathscr{M}_0$.

The processes with *weak barbs* are those which reduce to a barbed process: we say that a process P *has a weak barb in* x, notation $\Gamma \vdash P \Downarrow_x$, if $P \Rightarrow P'$ and $\Gamma' \vdash P' \downarrow_x$.

The following theorem fulfills the *barb preservation criterion*: if a mixed process has a barb, its translation has a weak barb on the same channel.

Theorem 4 (Barb Preservation)**.** *The translation* $(\!|\cdot|\!) : \mathscr{M}_0 \longrightarrow \mathscr{C}$ *preserves barbs, that is, if* $\Gamma \vdash P \downarrow_x$, *then* $(\!|\Gamma|\!) \vdash (\!|\Gamma \vdash P|\!) \Downarrow_x$.

Proof. An analysis of the translations of processes with barbs. In the case that x is linear, rearranging the choice in P in fragments, we obtain that $P \equiv (\nu x_n y_n)\ldots(\nu x_1 y_1)(\mathsf{lin}\, x\sum_{i\in I}(\sum_{j\in J} l_i^! v_{ij}.P_{ij} + \sum_{k\in K} l_i^? y_{ik}.P'_{ik}) \mid R)$ and so its translation is

$$\begin{aligned}(\!|\Gamma \vdash P|\!) \equiv (\nu x_n y_n)\ldots(\nu x_1 y_1)(\mathsf{NDChoice}\{\ & x \lhd l_i^!.\mathsf{NDChoice}\, x!v_{ij}.(\!|\Gamma_3, x\colon T_i \vdash P_{ij}|\!)\}_{j\in J},\\ & x \lhd l_i^?.\mathsf{NDChoice}\, x?y_{ik}.(\!|\Gamma_2 \circ \Gamma_3, x\colon T'_i, y_{ik}\colon S'_i \vdash P'_{ik}|\!)\}_{k\in K}\}_{i\in I} \mid\\ & (\!|\Gamma' \vdash R|\!)).\end{aligned}$$

This process makes internal reduction steps in the resolution of the outermost NDChoice, non-deterministically choosing one of the possible fragments, via Lemma 3. However, independently of which branch is chosen, they are all of the form $x \lhd \ell.C$, which has a barb in x. That is: $(\!|\Gamma \vdash P|\!) \Rightarrow (\nu x_n y_n)\ldots(\nu x_1 y_1)(x \lhd \ell.C \mid (\!|\Gamma' \vdash R|\!) \mid G)$, which has a barb in x. The G term is the inert remainder of the

NDChoice reduction. In the unrestricted case, we have $P \equiv (\nu x_n y_n)\ldots(\nu x_1 y_1)(\mathsf{un}\, x \sum_{i\in I}(\sum_{j\in J} l_i^! v_{ij}.P_{ij} + \sum_{k\in K} l_i^? y_{ik}.P'_{ik}) \mid R)$. The translation is

$$\begin{aligned}(\!|\Gamma \vdash P|\!) \equiv (\nu x_n y_n)\ldots(\nu x_1 y_1)((\nu uv)(u!() \mid \mathsf{un}\, v?\_.\mathsf{NDChoice}\{\\ (\nu ab)x!a.b \lhd l_i^!.\mathsf{NDChoice}\{b!v_{ij}.(u!() \mid (\!|\Gamma_1 \vdash P_{ij}|\!))\}_{j\in J},\\ (\nu ab)x!a.b \lhd l_i^?.\mathsf{NDChoice}\{b?y_{ik}.(u!() \mid (\!|\Gamma_1, y_{ik} : S'_i \vdash P'_{ik}|\!))\}_{k\in K}\}_{i\in I}) \mid (\!|\Gamma_2 \vdash R|\!)).\end{aligned}$$

The process starts by reducing via [R-UNCOM] on the u,v channels to the process

$$\begin{aligned}(\!|\Gamma \vdash P|\!) \Rightarrow (\nu x_n y_n)\ldots(\nu x_1 y_1)(\nu uv)(\mathsf{NDChoice}\{\\ (\nu ab)x!a.b \lhd l_i^!.\mathsf{NDChoice}\{b!v_{ij}.(u!() \mid (\!|\Gamma_1 \vdash P_{ij}|\!))\}_{j\in J},\\ (\nu ab)x!a.b \lhd l_i^?.\mathsf{NDChoice}\{b?y_{ik}.(u!() \mid (\!|\Gamma_1, y_{ik} : S'_i \vdash P'_{ik}|\!))\}_{k\in K}\}_{i\in I} \mid (\!|\Gamma_2 \vdash R|\!) \mid U)\end{aligned}$$

where U is the persistent part of the unrestricted process. This process, in turn, reduces via the NDChoice (Lemma 3) to one of the possible branches which are all of the form $(\nu ab)x!a.C$,

$$(\!|\Gamma \vdash P|\!) \Rightarrow (\nu x_n y_n)\ldots(\nu x_1 y_1)(\nu uv)(\nu ab)(x!a.C) \mid (\!|\Gamma' \vdash R|\!) \mid U \mid G).$$

Since P has a barb in x, $x \notin \{x_i, y_i\}_{i=1}^n$ and so this process also has a barb in x, concluding that $(\!|\Gamma \vdash P|\!)$ has indeed a weak barb in x. □

Finally, we look at operational completeness. Operational completeness relates the behavior of mixed sessions against their classical sessions images: any reduction step in mixed sessions can be mimicked by a sequence of reductions steps in classical sessions, modulo extended structural congruence. The ghost reductions result from the new channels and communication inserted by the translation, namely those due to the NDChoice and to the encoding of "loops" for un mixed choices.

Theorem 5 (Reduction Completeness)**.** *The translation* $(\!|\cdot|\!) : \mathscr{M}_0 \longrightarrow \mathscr{C}$ *is* operationally complete*, that is, if* $P \to_{\mathscr{M}_0} P'$*, then* $(\!|\Gamma \vdash P|\!) \Rightarrow_{\mathscr{C}} \asymp_{\mathscr{C}} (\!|\Gamma \vdash P'|\!)$*,*

Proof. By rule induction on the derivation of $P \to_{\mathscr{M}_0} P'$. We detail two cases.

Case [R-PAR]. We can show that if $Q_1 \Rightarrow_{\mathscr{C}} Q'_1$, then $Q_1 \mid Q_2 \Rightarrow_{\mathscr{C}} Q'_1 \mid Q_2$, by induction on the length of the reduction. Then we have $(\!|\Gamma \vdash P_1 \mid P_2|\!) = (\!|\Gamma_1 \vdash P_1|\!) \mid (\!|\Gamma_2 \vdash P_2|\!)$ with $\Gamma = \Gamma_1 \circ \Gamma_2$. By induction we have $(\!|\Gamma_1 \vdash P_1|\!) \Rightarrow_{\mathscr{C}} Q \asymp_{\mathscr{C}} (\!|\Gamma_1 \vdash P'_1|\!)$. Using the above result and the fact that $\asymp_{\mathscr{C}}$ is a congruence, we get $(\!|\Gamma_1 \vdash P_1|\!) \mid (\!|\Gamma_2 \vdash P_2|\!) \Rightarrow_{\mathscr{C}} Q \mid (\!|\Gamma_2 \vdash P_2|\!) \asymp_{\mathscr{C}} (\!|\Gamma_1 \vdash P'_1|\!) \mid (\!|\Gamma_2 \vdash P_2|\!) = (\!|\Gamma \vdash P'_1 \mid P_2|\!)$. The cases for [R-RES] and [R-STRUCT] are similar.

Case [R-LINLIN]. Let $\Gamma, x\colon R, y\colon S = \Gamma' \circ \Gamma'' \circ \Gamma'''$ and $\Gamma' \vdash x\colon \mathsf{lin}\&\{l^! T_0.R_0, \ldots\}$ and $\Gamma'' \vdash y\colon \mathsf{lin}\oplus\{l^? U_0.S_0, \ldots\}$, with $T_0 \approx U_0$ and $R_0 \perp S_0$. Let $\Gamma' = \Gamma'_1 \circ \Gamma'_2 \circ \Gamma'_3$ and $\Gamma'' = \Gamma''_1 \circ \Gamma''_2 \circ \Gamma''_3$. We have:

$$\begin{aligned}&(\!|\Gamma \vdash (\nu xy)(\mathsf{lin}x(l^! v.P + M) \mid \mathsf{lin}y(l^? z.Q + N) \mid O)|\!)\\ =&(\nu xy)(x \rhd \{l^? : \mathsf{NDChoice}\{x!v.(\!|\Gamma'_3, x : R_0 \vdash P|\!), \ldots\}, \ldots\} \mid\\ &\quad \mathsf{NDChoice}\{y \lhd l^?.\mathsf{NDChoice}\{y?z.(\!|(\Gamma''_2 \circ \Gamma''_3, y : S_0, z : U_0) \vdash Q|\!), \ldots\}, \ldots\} \mid (\!|\Gamma''' \vdash O|\!))\\ \to\asymp&(\nu xy)(x \rhd \{l^? : \mathsf{NDChoice}\{x!v.(\!|\Gamma'_3, x : R_0 \vdash P|\!), \ldots\}, \ldots\} \mid\\ &\quad y \lhd l^?.\mathsf{NDChoice}\{y?z.(\!|(\Gamma''_2 \circ \Gamma''_3, y : S_0, z : U_0) \vdash Q|\!), \ldots\} \mid (\!|\Gamma''' \vdash O|\!))\\ \to&(\nu xy)(\mathsf{NDChoice}\{x!v.(\!|\Gamma'_3, x : R_0 \vdash P|\!), \ldots\} \mid\\ &\quad \mathsf{NDChoice}\{y?z.(\!|(\Gamma''_2 \circ \Gamma''_3, y : S_0, z : U_0) \vdash Q|\!), \ldots\} \mid (\!|\Gamma''' \vdash O|\!))\\ \to\to\asymp&(\nu xy)(x!v.(\!|\Gamma'_3, x : R_0 \vdash P|\!) \mid y?z.(\!|(\Gamma''_2 \circ \Gamma''_3, y : S_0, z : U_0) \vdash Q|\!) \mid (\!|\Gamma''' \vdash O|\!))\end{aligned}$$

$$\begin{aligned}
&\to(\nu xy)(\llparenthesis \Gamma'_3, x : R_0 \vdash P \rrparenthesis \mid \llparenthesis \Gamma''_2 \circ \Gamma''_3, y : S_0, z : U_0 \vdash Q \rrparenthesis [v/z] \mid \llparenthesis \Gamma''' \vdash O \rrparenthesis) \\
&=(\nu xy)(\llparenthesis \Gamma'_3, x : R_0 \vdash P \rrparenthesis \mid \llparenthesis \Gamma'_2 \circ \Gamma''_2 \circ \Gamma''_3, y : S_0 \vdash Q[v/z] \rrparenthesis \mid \llparenthesis \Gamma''' \vdash O \rrparenthesis) \\
&=\llparenthesis \Gamma'_2 \circ \Gamma'_3 \circ \Gamma''_2 \circ \Gamma''_3 \circ \Gamma''' \vdash (\nu xy)(P \mid Q[v/z] \mid O) \rrparenthesis \\
&=\llparenthesis \Gamma \vdash (\nu xy)(P \mid Q[v/z] \mid O) \rrparenthesis
\end{aligned}$$

Notice that $\Gamma'_1 = \Delta_1, x\colon R$ where Δ_1 is un, hence Δ_1 is in Γ'_2 and in Γ'_3. The same reasoning applies to Γ''_1. Since context Γ'_2 is used to type v, the substitution lemma [10] reintroduces it in the context for $Q[v/z]$.

The case for [R-UnUn] is similar, albeit more verbose. The cases for [R-IfT] and [R-IfF] are direct. □

We can show that the translation does *not* enjoy reduction soundness. Consider the classical process Q to be the encoding of process P of the form $\mathsf{un}y(m^?z.\mathbf{0})$, described in the right part of Figure 3. Soundness requires that if $Q \to_{\mathscr{C}} Q'$, then $P \Rightarrow_{\mathscr{M}_0} P'$ and $Q \Rightarrow_{\mathscr{C}} \asymp_{\mathscr{C}} \llparenthesis \Gamma \vdash P' \rrparenthesis$. Clearly, Q has an initial reduction step (on channel u_2v_2), which cannot be mimicked by P. But this reduction is a transition internal to process Q, a τ transition. Equipped with a suitable notion of labelled transition systems on both languages that include τ transitions, and by using a weak bisimulation that ignores such transitions, we expect soundness to hold.

4 Further Work

There are two avenues that may be followed. One extends the encoding to the full language of mixed sessions, by taking into consideration the axioms in the reduction relation that match lin choices against un choices. The other pursues semantic preservation [6] by establishing a full abstraction result, requiring the development of typed equivalences for the two languages.

Acknowledgements This work was supported by FCT through the LASIGE Research Unit, ref. UIDB/00408/2020, and by Cost Action CA15123 EUTypes.

References

[1] Romain Demangeon & Kohei Honda (2011): *Full Abstraction in a Subtyped pi-Calculus with Linear Types*. In: *CONCUR 2011 - Concurrency Theory*, *Lecture Notes in Computer Science* 6901, Springer, pp. 280–296, doi:10.1007/978-3-642-23217-6_19.

[2] Juliana Franco & Vasco Thudichum Vasconcelos (2013): *A Concurrent Programming Language with Refined Session Types*. In: *Software Engineering and Formal Methods*, *Lecture Notes in Computer Science* 8368, Springer, pp. 15–28, doi:10.1007/978-3-319-05032-4_2.

[3] Simon J. Gay & Malcolm Hole (2005): *Subtyping for session types in the pi calculus*. *Acta Inf.* 42(2-3), pp. 191–225, doi:10.1007/s00236-005-0177-z.

[4] Kohei Honda (1993): *Types for Dyadic Interaction*. In: *CONCUR '93, 4th International Conference on Concurrency Theory*, *Lecture Notes in Computer Science* 715, Springer, pp. 509–523, doi:10.1007/3-540-57208-2_35.

[5] Kohei Honda, Vasco Thudichum Vasconcelos & Makoto Kubo (1998): *Language Primitives and Type Discipline for Structured Communication-Based Programming*. In: *Programming Languages and Systems*, *Lecture Notes in Computer Science* 1381, Springer, pp. 122–138, doi:10.1007/BFb0053567.

[6] Dimitrios Kouzapas, Jorge A. Pérez & Nobuko Yoshida (2019): *On the relative expressiveness of higher-order session processes*. *Inf. Comput.* 268, doi:10.1016/j.ic.2019.06.002.

[7] Robin Milner (1992): *Functions as Processes*. *Mathematical Structures in Computer Science* 2(2), pp. 119–141, doi:10.1017/S0960129500001407.

[8] Davide Sangiorgi (1998): *An Interpretation of Typed Objects into Typed pi-Calculus*. *Inf. Comput.* 143(1), pp. 34–73, doi:10.1006/inco.1998.2711.

[9] Kaku Takeuchi, Kohei Honda & Makoto Kubo (1994): *An Interaction-based Language and its Typing System*. In: *PARLE '94: Parallel Architectures and Languages Europe*, *Lecture Notes in Computer Science* 817, Springer, pp. 398–413, doi:10.1007/3-540-58184-7_118.

[10] Vasco T. Vasconcelos (2012): *Fundamentals of session types*. *Inf. Comput.* 217, pp. 52–70, doi:10.1016/j.ic.2012.05.002.

[11] Vasco T. Vasconcelos, Filipe Casal, Bernardo Almeida & Andreia Mordido (2020): *Mixed Sessions*. In: *Programming Languages and Systems, 29th European Symposium on Programming, ESOP 2020*, *Lecture Notes in Computer Science* 12075, Springer.

[12] Vasco Thudichum Vasconcelos (1994): *Typed Concurrent Objects*. In: *Object-Oriented Programming*, *Lecture Notes in Computer Science* 821, Springer, pp. 100–117, doi:10.1007/BFb0052178.

A The Syntax, Operational Semantics, and Type System of Mixed and Classical Sessions

Mixed Sessions The syntax of process and the operational semantics are in Figure 6. The syntax of types, and the notions of subtyping and type duality are in Figure 7. The un and lin predicates, the context split and update operations, and the typing rules are in Figure 8.

Classical Sessions The syntax, operational semantics, and type system are in Figure 9.

Mixed syntactic forms

$$
\begin{array}{rll}
v & ::= & \text{Values:} \\
 & x & \text{variable} \\
 & \mathsf{true} \mid \mathsf{false} & \text{boolean values} \\
 & () & \text{unit value} \\
P & ::= & \text{Processes:} \\
 & qx\sum_{i\in I} M_i & \text{choice} \\
 & P \mid P & \text{parallel composition} \\
 & (\nu xx)P & \text{scope restriction} \\
 & \mathsf{if}\, v\, \mathsf{then}\, P\, \mathsf{else}\, P & \text{conditional} \\
 & \mathbf{0} & \text{inaction} \\
M & ::= & \text{Branches:} \\
 & l^{\star} v.P & \text{branch} \\
\star & ::= & \text{Polarities:} \\
 & ! \mid ? & \text{out and in} \\
q & ::= & \text{Qualifiers:} \\
 & \mathsf{lin} & \text{linear} \\
 & \mathsf{un} & \text{unrestricted}
\end{array}
$$

Structural congruence, $P \equiv P$

$$
P \mid Q \equiv Q \mid P \qquad (P \mid Q) \mid R \equiv P \mid (Q \mid R) \qquad P \mid \mathbf{0} \equiv P
$$
$$
(\nu xy)P \mid Q \equiv (\nu xy)(P \mid Q) \qquad (\nu xy)\mathbf{0} \equiv \mathbf{0} \qquad (\nu wx)(\nu yz)P \equiv (\nu yz)(\nu wx)P
$$

Mixed reduction rules, $P \to P$

$$
\mathsf{if}\,\mathsf{true}\,\mathsf{then}\,P\,\mathsf{else}\,Q \to P \qquad \mathsf{if}\,\mathsf{false}\,\mathsf{then}\,P\,\mathsf{else}\,Q \to Q \qquad \text{[R-IfT] [R-IfF]}
$$
$$
(\nu xy)(\mathsf{lin}x(l^{!}v.P+M) \mid \mathsf{lin}y(l^{?}z.Q+N) \mid R) \to (\nu xy)(P \mid Q[v/z] \mid R) \qquad \text{[R-LinLin]}
$$
$$
(\nu xy)(\mathsf{lin}x(l^{!}v.P+M) \mid \mathsf{un}y(l^{?}z.Q+N) \mid R) \to (\nu xy)(P \mid Q[v/z] \mid \mathsf{un}y(l^{?}z.Q+N) \mid R) \qquad \text{[R-LinUn]}
$$
$$
(\nu xy)(\mathsf{un}x(l^{!}v.P+M) \mid \mathsf{lin}y(l^{?}z.Q+N) \mid R) \to (\nu xy)(P \mid Q[v/z] \mid \mathsf{un}x(l^{!}v.P+M) \mid R) \qquad \text{[R-UnLin]}
$$
$$
(\nu xy)(\mathsf{un}x(l^{!}v.P+M) \mid \mathsf{un}y(l^{?}z.Q+N) \mid R) \to \qquad \text{[R-UnUn]}
$$
$$
(\nu xy)(P \mid Q[v/z] \mid \mathsf{un}x(l^{!}v.P+M) \mid \mathsf{un}y(l^{?}z.Q+N) \mid R)
$$
$$
\frac{P \to Q}{(\nu xy)P \to (\nu xy)Q} \qquad \frac{P \to Q}{P \mid R \to Q \mid R} \qquad \frac{P \equiv P' \qquad P' \to Q' \qquad Q' \equiv Q}{P \to Q}
$$
[R-Res] [R-Par] [R-Struct]

Figure 6: Mixed session types: process syntax and reduction

$$
\begin{array}{rll}
T ::= & & \text{Types:} \\
 & q\sharp\{U_i\}_{i\in I} & \text{choice} \\
 & \mathsf{end} & \text{termination} \\
 & \mathsf{unit} \mid \mathsf{bool} & \text{unit and boolean} \\
 & \mu a.T & \text{recursive type} \\
 & a & \text{type variable} \\
U ::= & & \text{Branches:} \\
 & l^\star T.T & \text{branch} \\
\sharp ::= & & \text{Views:} \\
 & \oplus \mid \& & \text{internal and external} \\
\Gamma ::= & & \text{Contexts:} \\
 & \cdot & \text{empty} \\
 & \Gamma, x\colon T & \text{entry}
\end{array}
$$

The un *predicate*, $\mathsf{un}\,T$, $\mathsf{un}\,\Gamma$

$$\mathsf{un}\,(\mathsf{un}\sharp\{U_i\}_{i\in I}) \quad \mathsf{un}\,(\mu a.T) \text{ if } \mathsf{un}\,T \quad \mathsf{un}\,(\mathsf{end},\mathsf{unit},\mathsf{bool}) \qquad \mathsf{un}\,\cdot \quad \mathsf{un}\,(\Gamma, x\colon T) \text{ if } \mathsf{un}\,\Gamma \wedge \mathsf{un}\,T$$

Branch subtyping, $U <: U$

$$\frac{S_2 <: S_1 \quad T_1 <: T_2}{l^! S_1.T_1 <: l^! S_2.T_2} \qquad \frac{S_1 <: S_2 \quad T_1 <: T_2}{l^? S_1.T_1 <: l^? S_2.T_2}$$

Coinductive subtyping rules, $T <: T$

$$\frac{}{\mathsf{end} <: \mathsf{end}} \qquad \frac{}{\mathsf{unit} <: \mathsf{unit}} \qquad \frac{}{\mathsf{bool} <: \mathsf{bool}} \qquad \frac{S[\mu a.S/a] <: T}{\mu a.S <: T} \qquad \frac{S <: T[\mu a.T/a]}{S <: \mu a.T}$$

$$\frac{J \subseteq I \qquad U_j <: V_j}{q\oplus\{U_i\}_{i\in I} <: q\oplus\{V_j\}_{j\in J}} \qquad \frac{I \subseteq J \qquad U_i <: V_i}{q\&\{U_i\}_{i\in I} <: q\&\{V_j\}_{j\in J}}$$

Polarity duality and view duality, $\sharp \perp \sharp$ and $\star \perp \star$

$$!\perp ? \qquad ? \perp ! \qquad\qquad \oplus \perp \& \qquad \& \perp \oplus$$

Coinductive type duality rules, $T \perp T$

$$\frac{}{\mathsf{end} \perp \mathsf{end}} \qquad \frac{\sharp \perp \flat \qquad \star_i \perp \bullet_i \qquad S_i \approx S_i' \qquad T_i \perp T_i'}{q\sharp\{l_i^\star S_i.T_i\}_{i\in I} \perp q^\flat\{l_i^\bullet S_i'.T_i'\}_{i\in I}}$$

$$\frac{S[\mu a.S/a] \perp T}{\mu a.S \perp T} \qquad \frac{S \perp T[\mu a.T/a]}{S \perp \mu a.T}$$

Figure 7: Mixed session types: types syntax, subtyping, and duality

un *and* lin *predicates*, $\mathsf{un}(T)$, $\mathsf{lin}(T)$

$$\mathsf{un}(\mathsf{end}) \quad \mathsf{un}(\mathsf{unit}) \quad \mathsf{un}(\mathsf{bool}) \quad \mathsf{un}(\mathsf{un}\sharp\{U_i\}) \quad \frac{\mathsf{un}(T)}{\mathsf{un}(\mu a.T)} \qquad \overline{\mathsf{lin}(T)}$$

Context split, $\Gamma = \Gamma \circ \Gamma$

$$\cdot = \cdot \circ \cdot \qquad \frac{\Gamma_1 \circ \Gamma_2 = \Gamma \qquad \mathsf{un}(T)}{\Gamma, x\colon T = (\Gamma_1, x\colon T) \circ (\Gamma_2, x\colon T)}$$

$$\frac{\Gamma = \Gamma_1 \circ \Gamma_2}{\Gamma, x\colon \mathsf{lin}\, p = (\Gamma_1, x\colon \mathsf{lin}\, p) \circ \Gamma_2} \qquad \frac{\Gamma = \Gamma_1 \circ \Gamma_2}{\Gamma, x\colon \mathsf{lin}\, p = \Gamma_1 \circ (\Gamma_2, x\colon \mathsf{lin}\, p)}$$

Context update, $\Gamma + x\colon T = \Gamma$

$$\frac{x\colon U \notin \Gamma}{\Gamma + x\colon T = \Gamma, x\colon T} \qquad \frac{\mathsf{un}(T) \qquad T \approx U}{(\Gamma, x\colon T) + x\colon U = (\Gamma, x\colon T)}$$

Typing rules for values, $\Gamma \vdash v\colon T$

$$\frac{\mathsf{un}(\Gamma)}{\Gamma \vdash ()\colon \mathsf{unit}} \qquad \frac{\mathsf{un}(\Gamma)}{\Gamma \vdash \mathsf{true}, \mathsf{false}\colon \mathsf{bool}} \qquad \frac{\mathsf{un}(\Gamma_1, \Gamma_2)}{\Gamma_1, x\colon T, \Gamma_2 \vdash x\colon T} \qquad \frac{\Gamma \vdash v\colon S \quad S <\colon T}{\Gamma \vdash v\colon T}$$

[T-Unit] [T-True] [T-False] [T-Var] [T-Subt]

Typing rules for branches, $\Gamma \vdash M\colon U$

$$\frac{\Gamma_1 \vdash v\colon S \qquad \Gamma_2 \vdash P}{\Gamma_1 \circ \Gamma_2 \vdash l^{!}v.P\colon l^{!}S.T} \qquad \frac{\Gamma, x\colon S \vdash P}{\Gamma \vdash l^{?}x.P\colon l^{?}S.T}$$

[T-Out] [T-In]

Typing rules for processes, $\Gamma \vdash P$

$$\frac{q_1(\Gamma_1 \circ \Gamma_2) \quad \Gamma_1 \vdash x\colon q_2\sharp\{l_i^{\star}S_i.T_i\}_{i\in I} \quad \Gamma_2 + x\colon T_j \vdash l_j^{\star}v_j.P_j\colon l_j^{\star}S_j.T_j \quad \{l_j^{\star}\}_{j\in J} = \{l_i^{\star}\}_{i\in I}}{\Gamma_1 \circ \Gamma_2 \vdash q_1 x \sum_{j\in J} l_j^{\star}v_j.P_j} \quad \text{[T-Choice]}$$

$$\frac{\mathsf{un}(\Gamma)}{\Gamma \vdash \mathbf{0}} \qquad \frac{\Gamma_1 \vdash P \quad \Gamma_2 \vdash Q}{\Gamma_1 \circ \Gamma_2 \vdash P \mid Q} \qquad \frac{\Gamma_1 \vdash v\colon \mathsf{bool} \quad \Gamma_2 \vdash P \quad \Gamma_2 \vdash Q}{\Gamma_1 \circ \Gamma_2 \vdash \mathsf{if}\, v\, \mathsf{then}\, P\, \mathsf{else}\, Q} \qquad \frac{S \perp T \quad \Gamma, x\colon S, y\colon T \vdash P}{\Gamma \vdash (\nu xy)P}$$

[T-Inact] [T-Par] [T-If] [T-Res]

Figure 8: Mixed session types: un and lin predicates, context split and update, and typing

Syntactic forms

$$
\begin{array}{lll}
P ::= \ldots & & \text{Processes:} \\
\quad & x!v.P & \text{output} \\
\quad & qx?x.P & \text{input} \\
\quad & x \lhd l.P & \text{selection} \\
\quad & x \rhd \{l_i\colon P_i\}_{i\in I} & \text{branching} \\
T ::= \ldots & & \text{Types:} \\
\quad & q \star T.T & \text{communication} \\
\quad & q\sharp\{l_i\colon T_i\}_{i\in I} & \text{choice}
\end{array}
$$

Reduction rules, $P \to P$, (plus [R-RES] [R-PAR] [R-STRUCT] from Figure 6)

$$(\nu xy)(x!v.P \mid \mathsf{lin}\, y?z.Q \mid R) \to (\nu xy)(P \mid Q[v/z] \mid R) \qquad \text{[R-LINCOM]}$$

$$(\nu xy)(x!v.P \mid \mathsf{un}\, y?z.Q \mid R) \to (\nu xy)(P \mid Q[v/z] \mid \mathsf{un}\, y?z.Q \mid R) \qquad \text{[R-UNCOM]}$$

$$\frac{j \in I}{(\nu xy)(x \lhd l_j.P \mid y \rhd \{l_i\colon Q_i\}_{i\in I} \mid R) \to (\nu xy)(P \mid Q_j \mid R)} \qquad \text{[R-CASE]}$$

Subtyping rules, $T <: T$

$$\frac{T <: S \qquad S' <: T'}{q!S.S' <: q!T.T'} \qquad \frac{S <: T \qquad S' <: T'}{q?S.S' <: q?T.T'}$$

$$\frac{J \subseteq I \qquad S_j <: T_j}{q \oplus \{l_i\colon S_i\}_{i\in I} <: q \oplus \{l_j\colon T_j\}_{j\in J}} \qquad \frac{I \subseteq J \qquad S_i <: T_i}{q\&\{l_i\colon S_i\}_{i\in I} <: q\&\{l_j\colon T_j\}_{j\in J}}$$

Type duality rules, $T \perp T$

$$\frac{S <: T \qquad T <: S \qquad S' \perp T'}{q?S.S' \perp q!T.T'} \qquad \frac{S_i \perp T_i}{q \oplus \{l_i\colon S_i\}_{i\in I} \perp q\&\{l_i\colon T_i\}_{i\in I}}$$

Typing rules, $\Gamma \vdash P$, (plus [T-INACT] [T-PAR] [T-RES] from Figure 7)

$$\frac{\Gamma_1 \vdash x\colon q!T.U \qquad \Gamma_2 \vdash v\colon T \qquad \Gamma_3 + x\colon U \vdash P}{\Gamma_1 \circ \Gamma_2 \circ \Gamma_3 \vdash x!v.P} \qquad \text{[T-TOUT]}$$

$$\frac{q_1(\Gamma_1 \circ \Gamma_2) \qquad \Gamma_1 \vdash x\colon q_2?T.U \qquad (\Gamma_2 + x\colon U), y\colon T \vdash P}{\Gamma_1 \circ \Gamma_2 \vdash q_1 x?y.P} \qquad \text{[T-TIN]}$$

$$\frac{\Gamma_1 \vdash x\colon q\&\{l_i\colon T_i\}_{i\in I} \qquad \Gamma_2 + x\colon T_i \vdash P_i \qquad \forall i \in I}{\Gamma_1 \circ \Gamma_2 \vdash x \rhd \{l_i\colon P_i\}_{i\in I}} \qquad \text{[T-BRANCH]}$$

$$\frac{\Gamma_1 \vdash x\colon q\oplus\{l_i\colon T_i\}_{i\in I} \qquad \Gamma_2 + x\colon T_j \vdash P \qquad j \in I}{\Gamma_1 \circ \Gamma_2 \vdash x \lhd l_j.P} \qquad \text{[T-SEL]}$$

Figure 9: Classical session types