Lecture Notes on Substructural Operational Semantics

15-816: Substructural Logics
Frank Pfenning

Lecture 19
November 3, 2016

Throughout this course we have already used substructural logic to specify the operational semantics of our (small) programming languages. For example, we have used ordered inference to represent computation based on proof reduction in subsingleton logic and we have used linear inference to represent computation in ordered logic. Focusing provides us with the needed connection between ordered and linear inference and *propositions* in ordered and linear logic, completing the picture. This use of the inference in linear logic goes back to CLF [WCPW02, CPWW02] which was in turn inspired by Forum [Chi95, Mil96]. A systematic, taxonomic approach was advocated [Pfe04] and then explored [SP08, SP09, PS09, SP11] culminating in Simmons's dissertation [Sim12].

Before we give further applications, we need to consider how focusing applies to structural logic, and the integration of structural and substructural logics.

1 Example: Increment

Let's recall the representation of binary numbers as string of $\mathsf{b0}$ and $\mathsf{b1}$ and a left endmarker $\$$ (previously written as eps). For example, the number 1011_2 would be represented by the ordered context $\$\ \mathsf{b1}\ \mathsf{b0}\ \mathsf{b1}\ \mathsf{b1}$. Now we specify incrementing such a number by adding a new ordered proposition inc with the following three rules.

$$\frac{\mathsf{b0}\quad \mathsf{inc}}{\mathsf{b1}} \qquad \frac{\mathsf{b1}\quad \mathsf{inc}}{\mathsf{inc}\quad \mathsf{b0}} \qquad \frac{\$\quad \mathsf{inc}}{\$\quad \mathsf{b1}}$$

We can represent them by the following propositions, where $\mathsf{b0}^+$, $\mathsf{b1}^+$, $\$^+$, and inc^+ are all positive atoms. We assume that $\bullet$ more binding strength than $\backslash$ and $/$, so that $A \bullet B \setminus C$ stands for $(A \bullet B) \setminus C$.

$$\begin{array}{l}
\mathsf{b0} \bullet \mathsf{inc} \setminus {\uparrow} \mathsf{b1} \\
\mathsf{b1} \bullet \mathsf{inc} \setminus {\uparrow} (\mathsf{inc} \bullet \mathsf{b0}) \\
\$ \bullet \mathsf{inc} \setminus {\uparrow} (\$ \bullet \mathsf{b1})
\end{array}$$

However, there is a fly in the ointment: the inference rules are persistent, while the propositions we may focus on are not. So the above propositions should be shifted from ordered O to structural U. We might try

$$\uparrow^{\mathsf{U}}_{\mathsf{O}} (\mathsf{b0} \bullet \mathsf{inc} \setminus \uparrow^{\mathsf{O}}_{\mathsf{O}} \mathsf{b1})$$

but this does not work, since the an up shift coerces a positive proposition to a negative one, but $(\mathsf{b0} \bullet \mathsf{inc} \setminus \uparrow^{\mathsf{O}}_{\mathsf{O}} \mathsf{b1})$ is already negative. So we could write it as

$$\uparrow^{\mathsf{U}}_{\mathsf{O}} \downarrow^{\mathsf{O}}_{\mathsf{O}} (\mathsf{b0} \bullet \mathsf{inc} \setminus \uparrow^{\mathsf{O}}_{\mathsf{O}} \mathsf{b1})$$

which is logically correct but, as we will see in Section 4, it may not have the expected focusing behavior. So we write

$${}^{\mathsf{U}}_{\mathsf{O}}{\uparrow\!\downarrow} (\mathsf{b0} \bullet \mathsf{inc} \setminus \uparrow^{\mathsf{O}}_{\mathsf{O}} \mathsf{b1})$$

to lift a negative ordered proposition to a negative unrestricted proposition. Before we investigate the properties of ${}^{\mathsf{U}}_{\mathsf{O}}{\uparrow\!\downarrow}$, an excursion to look at focusing for structural logic more generally.

2 Focusing for Structural Logic

We start with the polarized form of structural logic. As before we keep two forms of conjunction which are logically, but not computationally equivalent.

$$\begin{array}{lcl}
A^- & ::= & p^- \mid A^+ \to B^- \mid A^- \mathbin{\&} B^- \mid {\uparrow} A^+ \\
A^+ & ::= & p^+ \mid A^+ \times B^+ \mid \mathbf{1} \mid A^+ + B^+ \mid {\downarrow} A^-
\end{array}$$

The key insights are the following:

1. For left inversion rules, we do not need to keep a copy of the principal proposition of the inference. That's because its components are equivalent to the proposition itself. This also means that even structural antecedents are no longer persistent.

2. For left chaining rules, the proposition in focus is also not persistent. This also should be intuitive, since we cannot have the formula and its subformula both be in focus.

We say *antecedents* Γ are stable$^-$ if Γ consists only of negative propositions and positive atoms and *succedent* C is stable$^+$ if it is a positive proposition or a negative atom. We still have three judgment forms

$$\begin{array}{ll} \Gamma \Vdash A & \\ \Gamma \Vdash [A] & \text{with } \Gamma \text{ stable}^- \\ \Gamma, [A] \Vdash C & \text{with } \Gamma \text{ stable}^- \text{ and } C \text{ stable}^+ \end{array}$$

As in Lecture 18 we present a confluent focusing system with don't-care nondeterministic inversion rules. Simmons [Sim14] presents a system more suited for most implementations and also proofs of admissibility of cut and identity in which inversion takes place deterministically. We first show the structural rules.

$$\frac{}{\Gamma, p^+ \Vdash [p^+]}\ \mathsf{id}^+ \qquad \frac{}{\Gamma, [p^-] \Vdash p^-}\ \mathsf{id}^-$$

$$\frac{(\Gamma\ \mathsf{stable}^-) \quad \Gamma \Vdash [C^+]}{\Gamma \Vdash C^+}\ \mathsf{focus}^+$$

$$\frac{(\Gamma\ \mathsf{stable}^-, C\ \mathsf{stable}^+) \quad \Gamma, A^-, [A^-] \Vdash C}{\Gamma, A^- \Vdash C}\ \mathsf{focus}^-$$

Note that in the negative focusing rule we *copy* the proposition A^-, which will be the only instance of an explicit contraction-like behavior. In all other two-premise rules, we will propagate the antecedents to both premises. The remaining rules can be constructed straightforwardly from the general principles we have laid out.

$$\frac{\Gamma, A^+ \Vdash B^-}{\Gamma \Vdash A^+ \to B^-}\ {\to}R \qquad \frac{\Gamma \Vdash [A^+] \quad \Gamma, [B^-] \Vdash C}{\Gamma, [A^+ \to B^-] \Vdash C}\ {\to}L$$

$$\frac{\Gamma \Vdash A^- \quad \Gamma \Vdash B^-}{\Gamma \Vdash A^- \mathbin{\&} B^-}\ \&R \qquad \frac{\Gamma, [A^-] \Vdash C}{\Gamma, [A^- \mathbin{\&} B^-] \Vdash C}\ \&L_1 \qquad \frac{\Gamma, [B^-] \Vdash C}{\Gamma, [A^- \mathbin{\&} B^-] \Vdash C}\ \&L_2$$

$$\frac{\Gamma \Vdash A^+}{\Gamma \Vdash {\uparrow}A^+}\ {\uparrow}R \qquad \frac{\Gamma, A^+ \Vdash C}{\Gamma, [{\uparrow}A^+] \Vdash C}\ {\uparrow}L$$

$$\frac{\Gamma \Vdash [A^+] \quad \Gamma \Vdash [B^+]}{\Gamma \Vdash [A^+ \times B^+]}\ {\times}R \qquad \frac{\Gamma, A^+, B^+ \Vdash C}{\Gamma, A^+ \times B^+ \Vdash C}\ {\times}L$$

$$\frac{}{\Gamma \Vdash [\mathbf{1}]}\ \mathbf{1}R \qquad \frac{\Gamma \Vdash C}{\Gamma, \mathbf{1} \Vdash C}\ \mathbf{1}L$$

$$\frac{\Gamma \Vdash [A^+]}{\Gamma \Vdash [A^+ + B^+]}\ {+}R_1 \qquad \frac{\Gamma \Vdash [B^+]}{\Gamma \Vdash [A^+ + B^+]}\ {+}R_2 \qquad \frac{\Gamma, A^+ \Vdash C \quad \Gamma, B^+ \Vdash C}{\Gamma, A^+ + B^+ \Vdash C}\ {+}L$$

$$\frac{\Gamma \Vdash A^-}{\Gamma \Vdash [{\downarrow}A^-]}\ {\downarrow}R \qquad \frac{\Gamma, A^- \Vdash C}{\Gamma, {\downarrow}A^- \Vdash C}\ {\downarrow}L$$

3 Quantifiers

The quantifiers follow the familiar patterns; we saw their basic structure in Lecture 14. We can deduce their polarity by seeing which rules are invertible, or which rule is applied first in the identity expansion. Clearly, they are $\forall R$ and $\exists L$.

$$\begin{array}{lcl} A^- & ::= & \ldots \mid \forall x{:}\tau.\ A^- \\ A^+ & ::= & \ldots \mid \exists x{:}\tau.\ A^+ \end{array}$$

We generalize the judgment to allow term variables τ to be declared in a signature Σ which is propagated to all premises in all rules: type declarations for term variables are persistent.

$$\frac{\Sigma, a{:}\tau \;;\; \Gamma \Vdash A(a)^-}{\Sigma \;;\; \Gamma \Vdash \forall x{:}\tau.\, A(x)^-}\;\forall R^a \qquad \frac{\Sigma \vdash t : \tau \quad \Sigma \;;\; \Gamma, [A(t)^-] \Vdash C}{\Sigma \;;\; \Gamma, [\forall x{:}\tau.\, A(x)^-] \Vdash C}\;\forall L$$

$$\frac{\Sigma \vdash t : \tau \quad \Sigma \;;\; \Gamma \Vdash [A(t)^+]}{\Sigma \;;\; \Gamma \Vdash [\exists x{:}\tau.\, A(x)^+]}\;\exists R \qquad \frac{\Sigma, a{:}\tau \;;\; \Gamma, A(a)^+ \Vdash C}{\Sigma \;;\; \Gamma, \exists x{:}\tau.\, A(x)^+ \Vdash C}\;\exists L^a$$

4 Shifting Focus Between Logics

For the moment, we are interested in a logic that combines ordered with unrestricted propositions, with a very thin layer of unrestricted propositions.[1]

$$\begin{array}{lcl}
A^-_{\mathsf{U}} & ::= & p^-_{\mathsf{U}} \mid {}^{\mathsf{U}}_{\mathsf{O}}\!\upharpoonleft\!\downharpoonright A^-_{\mathsf{O}} \\
A^+_{\mathsf{U}} & ::= & p^+_{\mathsf{U}} \\
A^-_{\mathsf{O}} & ::= & p^-_{\mathsf{O}} \mid \ldots \mid \uparrow^{\mathsf{O}}_{\mathsf{O}} A^+ \\
A^+_{\mathsf{O}} & ::= & p^+_{\mathsf{O}} \mid \ldots \mid \downarrow^{\mathsf{O}}_{\mathsf{O}} A^- \mid {}^{\mathsf{U}}_{\mathsf{O}}\!\downharpoonleft\!\upharpoonright A^+_{\mathsf{U}}
\end{array}$$

We get the following additional rules

$$\frac{}{\Gamma, p^+_{\mathsf{U}} \Vdash [p^+_{\mathsf{U}}]}\;\mathsf{id}^+_{\mathsf{U}} \qquad \frac{}{\Gamma, [p^-_{\mathsf{U}}] \Vdash p^-_{\mathsf{U}}}\;\mathsf{id}^-_{\mathsf{U}}$$

$$\frac{\begin{array}{c}(\Gamma\ \mathsf{stable}^-, \Omega\ \mathsf{stable}^-, C_{\mathsf{O}}\ \mathsf{stable}^+)\\ \Gamma, A^-_{\mathsf{U}}, [A^-_{\mathsf{U}}] \;;\; \Omega \Vdash C_{\mathsf{O}}\end{array}}{\Gamma, A^-_{\mathsf{U}} \;;\; \Omega \Vdash C_{\mathsf{O}}}\;\mathsf{focus}^-_{\mathsf{UO}} \qquad \frac{\begin{array}{c}(\Gamma\ \mathsf{stable}^-, C_{\mathsf{U}}\ \mathsf{stable}^+)\\ \Gamma, A^-_{\mathsf{U}}, [A^-_{\mathsf{U}}] \Vdash C_{\mathsf{U}}\end{array}}{\Gamma, A^-_{\mathsf{U}} \Vdash C_{\mathsf{U}}}\;\mathsf{focus}^-_{\mathsf{UU}}$$

$$\frac{\begin{array}{c}(\Gamma\ \mathsf{stable}^-)\\ \Gamma \Vdash [A^+_{\mathsf{U}}]\end{array}}{\Gamma \Vdash A^+_{\mathsf{U}}}\;\mathsf{focus}^+_{\mathsf{U}}$$

$$\frac{\Gamma \;;\; \cdot \Vdash A^-_{\mathsf{O}}}{\Gamma \Vdash {}^{\mathsf{U}}_{\mathsf{O}}\!\upharpoonleft\!\downharpoonright A^-_{\mathsf{O}}}\;{}^{\mathsf{U}}_{\mathsf{O}}\!\upharpoonleft\!\downharpoonright R \qquad \frac{\Gamma \;;\; \Omega_1\ [A^-_{\mathsf{O}}]\ \Omega_2 \Vdash C}{\Gamma, [{}^{\mathsf{U}}_{\mathsf{O}}\!\upharpoonleft\!\downharpoonright A^-_{\mathsf{O}}] \;;\; \Omega_1\ \Omega_2 \Vdash C}\;{}^{\mathsf{U}}_{\mathsf{O}}\!\upharpoonleft\!\downharpoonright L$$

$$\frac{\Gamma \Vdash [A^+_{\mathsf{O}}]}{\Gamma \;;\; \cdot \Vdash [{}^{\mathsf{U}}_{\mathsf{O}}\!\downharpoonleft\!\upharpoonright A^+_{\mathsf{O}}]}\;{}^{\mathsf{U}}_{\mathsf{O}}\!\downharpoonleft\!\upharpoonright R \qquad \frac{\Gamma, A^+_{\mathsf{U}} \;;\; \Omega_1\ \Omega_2 \Vdash C_{\mathsf{O}}}{\Gamma \;;\; \Omega_1\ ({}^{\mathsf{U}}_{\mathsf{O}}\!\downharpoonleft\!\upharpoonright A^+_{\mathsf{U}})\ \Omega_2 \Vdash C_{\mathsf{O}}}\;{}^{\mathsf{U}}_{\mathsf{O}}\!\downharpoonleft\!\upharpoonright L$$

[1]As I am writing this, I am not at all sure that the polarity-preserving, mode-shifting modalities really work.

How do we now specify the ordered operational semantics of subsingleton logic? We consider a couple of rules.

$$\frac{\mathsf{proc}(P \mid Q)}{\mathsf{proc}(\mathsf{P}) \quad \mathsf{proc}(Q)}\ \mathsf{cmp}$$

This rule may be used many times, so we get

$${}^{\mathsf{U}}_{\mathsf{O}}{\upharpoonleft\!\downharpoonright}(\forall P.\, \forall Q.\, \mathsf{proc}(P \mid Q) \setminus {\uparrow}(\mathsf{proc}(P) \bullet \mathsf{proc}(Q)))$$

Focusing on this unrestricted proposition will create the computation rule

$$\frac{\Omega_1\ \mathsf{proc}(P)\ \mathsf{proc}(Q)\ \Omega_2 \Vdash C}{\Omega_1\ \mathsf{proc}(P \mid Q)\ \Omega_2 \Vdash C}\ \mathsf{cmp}$$

which corresponds exactly the original ordered inference rule cmp just above. In the following, we omit the explicit ${}^{\mathsf{U}}_{\mathsf{O}}{\upharpoonleft\!\downharpoonright}$ and the outermost quantifiers, using the conventions that rules are persistent and that upper case variables are implicitly universally quantified.

Next, the rule for disjunction.

$$\frac{\mathsf{proc}(\mathsf{R}.l_k\ ;\ P) \quad \mathsf{proc}(\mathsf{caseL}\ (l_i \Rightarrow Q_i)_{i \in I})}{\mathsf{proc}(P) \quad \mathsf{proc}(Q_k)}\ \oplus C$$

Propositionally:

$$\begin{array}{l} {}^{\mathsf{U}}_{\mathsf{O}}{\upharpoonleft\!\downharpoonright}\, \forall P{:}\mathsf{pexp}.\ \forall I{:}\mathsf{idx}.\ \forall Q{:}\Pi i{\in}I.\ \mathsf{pexp}.\ \forall k{\in}I. \\ \quad \mathsf{proc}(\mathsf{R}.l_k\ ;\ P) \bullet \mathsf{proc}(\mathsf{caseL}\ (l_i \Rightarrow Q_i)_{i \in I}) \setminus {\uparrow}\,(\mathsf{proc}(P) \bullet \mathsf{proc}(Q_k)) \end{array}$$

We add the remaining rules, omitting the leading shifts and quantifiers which can easily be inferred. We also label each rule with a name, which we will eventually see as a dependent type declaration.

$$\begin{array}{lcl} \mathsf{cmp} & : & \mathsf{proc}(P \mid Q) \setminus {\uparrow}(\mathsf{proc}(P) \bullet \mathsf{proc}(Q)) \\ \mathsf{fwd} & : & \mathsf{proc}(\leftrightarrow) \setminus {\uparrow}\, \mathbf{1} \\ \oplus C & : & \mathsf{proc}(\mathsf{R}.l_k\ ;\ P) \bullet \mathsf{proc}(\mathsf{caseL}\ (l_i \Rightarrow Q_i)_{i \in I}) \setminus {\uparrow}\,(\mathsf{proc}(P) \bullet \mathsf{proc}(Q_k)) \\ \& C & : & \mathsf{proc}(\mathsf{caseR}\ (l_i \Rightarrow P_i)_{i \in I}) \bullet \mathsf{proc}(\mathsf{L}.l_k\ ;\ Q) \setminus {\uparrow}\,(\mathsf{proc}(P_k) \bullet \mathsf{proc}(Q)) \\ \mathbf{1}C & : & \mathsf{proc}(\mathsf{closeR}) \bullet \mathsf{proc}(\mathsf{waitL}\ ;\ Q) \setminus {\uparrow}\, \mathsf{proc}(Q) \end{array}$$

5 Example: Ordered Processes

When we generalize away from the subsingleton fragment, we needed to introduce channels because a process might communicate along any of the channels it uses. Consequently, we used *linear inference* rather than ordered inference to describe the operational semantics. The substructural operational semantics then uses the linear fragment rather than the ordered one. The principles of chaining, inversion, and focusing are completely analogous, so we will just use it without further formalities.

Processes are now captured with the predicate $\mathsf{proc}(x, P)$ which is process P offering a service along channel x. We begin with the rule of composition for spawning a new process, providing along a new channel z.

$$\frac{\mathsf{proc}(x, y \leftarrow P(y) \; ; \; Q(y))}{\mathsf{proc}(z, P(z)) \quad \mathsf{proc}(x, Q(z))}\ \mathsf{cmp}^z$$

Omitting quantifiers as in the previous example, we start with something like

$$\mathsf{cmp} \quad : \quad \mathsf{proc}(X, y \leftarrow P(y) \; ; \; Q(y)) \multimap {\uparrow}(\mathsf{proc}(z, P(z)) \otimes \mathsf{proc}(X, Q(z))) \quad ??$$

The problem here is the status of z. It it were a free variable in the rule and therefore implicitly universally quantified, we could choose any channel for z, including, say, X, which is obviously incorrect: z must be chosen fresh. The answer here is to *existentially quantify* over z.

$$\mathsf{cmp} \quad : \quad \mathsf{proc}(X, y \leftarrow P(y) \; ; \; Q(y)) \multimap {\uparrow}(\exists z{:}\mathsf{ch}.\, \mathsf{proc}(z, P(z)) \otimes \mathsf{proc}(X, Q(z)))$$

To see why this is correct, let us consider focusing on

$$a^+ \multimap {\uparrow}(\exists z.\, b^+(z) \otimes c^+(z))$$

which is a slightly abstracted version of the rule above. First, the chaining phase.

$$\frac{(\Delta = (\Delta_1, \Delta_2)) \quad \dfrac{(\Delta_1 = a^+)}{\Sigma \; ; \; \Delta_1 \Vdash [a^+]}\ \mathsf{id}^+ \quad \dfrac{\begin{array}{c}\vdots \\ \Sigma \; ; \; \Delta_2, \exists z.\, b^+(z) \otimes c^+(z) \Vdash G\end{array}}{\Sigma \; ; \; \Delta_2, [{\uparrow}(\exists z.\, b^+(z) \otimes c^+(z))] \Vdash G}\ {\uparrow}L}{\Sigma \; ; \; \Delta, [a^+ \multimap {\uparrow}(\exists z.\, b^+(z) \otimes c^+(z))] \Vdash G}\ {\multimap}L$$

In the only remaining subgoal, we need to complete the inversion phase.

$$\dfrac{(\Delta = (\Delta_1, \Delta_2)) \quad \dfrac{(\Delta_1 = a^+)}{\Sigma \,;\, \Delta_1 \Vdash [a^+]}\,\mathsf{id}^+ \quad \dfrac{\dfrac{\dfrac{\Sigma, z{:}\tau \,;\, \Delta_2, b^+(z), c^+(z) \Vdash G}{\Sigma, z{:}\tau \,;\, \Delta_2, b^+(z) \otimes c^+(z) \Vdash G}\,\otimes L}{\Sigma \,;\, \Delta_2, \exists z.\, b^+(z) \otimes c^+(z) \Vdash G}\,\exists L^z}{\Sigma \,;\, \Delta_2, [\uparrow(\exists z.\, b^+(z) \otimes c^+(z))] \Vdash G}\,\uparrow L}{\Sigma \,;\, \Delta, [a^+ \multimap \uparrow(\exists z.\, b^+(z) \otimes c^+(z))] \Vdash G}\,\multimap L$$

This gives us the following synthetic rule of inference:

$$\dfrac{\Sigma, z{:}\tau \,;\, \Delta_2, b^+(z), c^+(z) \Vdash G}{\Sigma \,;\, \Delta_2, a^+ \Vdash G}\,R^z$$

In our particular example, we get

$$\dfrac{\Sigma, z{:}\mathsf{ch} \,;\, \Delta, \mathsf{proc}(z, P(z)), \mathsf{proc}(X, Q(z)) \Vdash G}{\Sigma \,;\, \Delta, \mathsf{proc}(X, y \leftarrow P(y) \,;\, Q(y)) \Vdash G}\,\mathsf{cmp}^z$$

which is exactly what we were hoping for, since it is the correct sequent calculus rendering of our original linear inference rule

$$\dfrac{\mathsf{proc}(x, y \leftarrow P(y) \,;\, Q(y))}{\mathsf{proc}(z, P(z)) \quad \mathsf{proc}(x, Q(z))}\,\mathsf{cmp}^z$$

From this example, we can now write some of the other rules. For the sake of brevity, we specify the synchronous versions.

$$
\begin{array}{lcl}
\mathsf{cmp} & : & \mathsf{proc}(X, y \leftarrow P(y)\ ;\ Q(y)) \\
& & \quad \multimap \uparrow (\exists z{:}\mathsf{ch}.\ \mathsf{proc}(z, P(z)) \otimes \mathsf{proc}(X, Q(z))) \\
\oplus C & : & \mathsf{proc}(X, X.l_k\ ;\ P) \otimes \mathsf{proc}(Z, \mathsf{case}\ X\ (l_i \Rightarrow Q_i)_{i \in I}) \\
& & \quad \multimap \uparrow (\mathsf{proc}(X, P) \otimes \mathsf{proc}(Z, Q_k)) \\
\& C & : & \mathsf{proc}(X, \mathsf{case}\ X\ (l_i \Rightarrow P_i)_{i \in I}) \otimes \mathsf{proc}(Z, X.l_k\ ;\ Q) \\
& & \quad \multimap \uparrow (\mathsf{proc}(X, P_k) \otimes \mathsf{proc}(Z, Q)) \\
\otimes C & : & \mathsf{proc}(X, \mathsf{send}\ X\ W\ ;\ P) \otimes \mathsf{proc}(Z, y \leftarrow \mathsf{recv}\ X\ ;\ Q(y)) \\
& & \quad \multimap \uparrow (\mathsf{proc}(X, P) \otimes \mathsf{proc}(Z, Q(W))) \\
\multimap C & : & \mathsf{proc}(X, y \leftarrow \mathsf{recv}\ X\ ;\ P(y)) \otimes \mathsf{proc}(Z, \mathsf{send}\ X\ W\ ;\ Q) \\
& & \quad \multimap \uparrow (\mathsf{proc}(X, P(W)) \otimes \mathsf{proc}(Z, Q)) \\
\mathbf{1} C & : & \mathsf{proc}(X, \mathsf{close}\ X) \otimes \mathsf{proc}(Z, \mathsf{wait}\ X\ ;\ Q) \\
& & \quad \multimap \uparrow \mathsf{proc}(Q) \\
\mathsf{fwd} & : & \mathsf{proc}(X, X \leftarrow Y) \multimap \uparrow X \doteq Y
\end{array}
$$

Only the last rule requires a new form of linear proposition, namely equality. We use it here only for parameters at type ι without any constant constructors to avoid a more extended development. Its right rule is just reflexivity; its left rules performs substitution.

$$
\frac{}{\Sigma, x{:}\iota\ ;\ \cdot \vdash x \doteq x}\ \doteq R
\qquad
\frac{\Sigma, z{:}\iota\ ;\ \Delta(z,z) \vdash C(z,z)}{\Sigma, x{:}\iota, y{:}\iota\ ;\ \Delta(x,y), x \doteq y \vdash C(x,y)}\ \doteq L^z
$$

This form of equality is *positive*: the left rule is invertible. This means that the focusing versions are

$$
\frac{}{\Sigma, x{:}\iota\ ;\ \cdot \Vdash [x \doteq x]}\ \doteq R
\qquad
\frac{\Sigma, z{:}\iota\ ;\ \Delta(z,z) \Vdash C(z,z)}{\Sigma, x{:}\iota, y{:}\iota\ ;\ \Delta(x,y), x \doteq y \Vdash C(x,y)}\ \doteq L^z
$$

Note that we could and perhaps should retain x and y in the signature in the premise, but they can no longer occur in $\Delta(z,z)$ or $G(z,z)$ so we have removed them.

Playing through focusing on the forwarding rule

$$
\mathsf{fwd} : \mathsf{proc}(X, X \leftarrow Y) \multimap \uparrow X \doteq Y
$$

we get the following synthetic rule

$$\frac{\Sigma, z{:}\mathsf{ch} \; ; \; \Delta(z,z) \Vdash C(z,z)}{\Sigma, x{:}\mathsf{ch}, y{:}\mathsf{ch} \; ; \; \Delta(x,y), \mathsf{proc}(x, x \leftarrow y) \Vdash C(x,y)}\ \mathsf{fwd}^z$$

which is the correct rendering of our (sketched) rule of linear inference: the channels x and y are identified. Since variables can be consistently renamed in a judgment, we could write equivalently $\Sigma, x{:}\mathsf{ch} \; ; \; \Delta(x,x) \Vdash C(x,x)$ or $\Sigma, y{:}\mathsf{ch} \; ; \; \Delta(y,y) \Vdash C(y,y)$. Parameters here are *not* names in the sense of nominal logic, since we cannot compare them for *disequality*. In fact, doing so would be wrong: the identity rule would then be unsound since it unifies two previously distinct parameters.

Exercises

Exercise 1 Write out the ordered SSOS rules for the asynchrononous semantics for subsingleton logic, using a msg predicate.

Exercise 2 Write out the linear SSOS rules for the asynchronous semantics for ordered logic, using a msg predicate.

Exercise 3 Integrate a structural layer into ordered logic using $\uparrow_{\mathsf{O}}^{\mathsf{U}} A_{\mathsf{O}}$ and $\downarrow_{\mathsf{O}}^{\mathsf{U}} A_{\mathsf{U}}$. Then use the modalities of Section 4 to extend the synchronous semantics of Section 5 to the new shift constructs.

References

[Chi95] Jawahar Lal Chirimar. *Proof Theoretic Approach to Specification Languages*. PhD thesis, University of Pennsylvania, May 1995.

[CPWW02] Iliano Cervesato, Frank Pfenning, David Walker, and Kevin Watkins. A concurrent logical framework II: Examples and applications. Technical Report CMU-CS-02-102, Department of Computer Science, Carnegie Mellon University, 2002. Revised May 2003.

[Mil96] Dale Miller. Forum: A multiple-conclusion specification logic. *Theoretical Computer Science*, 165(1):201–232, 1996.

[Pfe04] Frank Pfenning. Substructural operational semantics and linear destination-passing style. In W.-N. Chin, editor, *Proceedings of the 2nd Asian Symposium on Programming Languages and Systems (APLAS'04)*, page 196, Taipei, Taiwan, November 2004. Springer-Verlag LNCS 3302. Abstract of invited talk.

[PS09] Frank Pfenning and Robert J. Simmons. Substructural operational semantics as ordered logic programming. In *Proceedings of the 24th Annual Symposium on Logic in Computer Science (LICS 2009)*, pages 101–110, Los Angeles, California, August 2009. IEEE Computer Society Press.

[Sim12] Robert J. Simmons. *Substructural Logical Specifications*. PhD thesis, Carnegie Mellon University, November 2012. Available as Technical Report CMU-CS-12-142.

[Sim14] Robert J. Simmons. Structural focalization. *ACM Transactions on Computational Logic*, 15(3):21:1–21:33, 2014.

[SP08] Robert J. Simmons and Frank Pfenning. Linear logical algorithms. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming (ICALP'08)*, pages 336–345, Reykjavik, Iceland, July 2008. Springer LNCS 5126.

[SP09] Robert J. Simmons and Frank Pfenning. Linear logical approximations. In G. Puebla and G. Vidal, editors, *Proceedings of the Workshop on Partial Evaluation and Program Manipulation*, pages 9–20, Savannah, Georgia, January 2009. ACM SIGPLAN.

[SP11] Robert J. Simmons and Frank Pfenning. Logical approximation for program analysis. *Higher-Order and Symbolic Computation*, 24(1–2):41–80, 2011.

[WCPW02] Kevin Watkins, Iliano Cervesato, Frank Pfenning, and David Walker. A concurrent logical framework I: Judgments and properties. Technical Report CMU-CS-02-101, Department of Computer Science, Carnegie Mellon University, 2002. Revised May 2003.