HENRY DEYOUNG

SESSION-TYPED ORDERED LOGICAL SPECIFICATIONS

Contents

Part I

Preliminaries

Part II

Concurrency as proof construction

1
String rewriting for concurrent specifications

In this chapter, we consider abstract rewriting as a framework for specifying the dynamics of concurrent systems. This is not, of course, a new idea. Multiset rewriting,[1] as a state-transformation model of concurrency, has been used to describe security protocols,[2] for example. Unlike in multiset rewriting, we are particularly interested in concurrent systems whose components are arranged in a linear topology and have a monoidal structure. Given that finite strings over an alphabet Σ form a monoid, string rewriting, rather than multiset rewriting, is a good match for the structure we are interested in.

[1] Cervesato+Scedrov:IC09Meseguer:TCS92.

[2] Cervesato+:CSFW99Durgin+:JCS04.

For a broad sketch of string rewriting, consider the finite strings over the alphabet $\{a, b\}$, and let $\longrightarrow$ be the least compatible binary relation over those strings that satisfies the axioms

$$\frac{}{a\,b \longrightarrow b} \qquad \text{and} \qquad \frac{}{b \longrightarrow \epsilon}\,. \tag{1.1}$$

This relation can be seen as a rewriting relation on strings. For instance, because $a\,b\,b \longrightarrow b\,b$, we would say that $a\,b\,b$ may be rewritten to $b\,b$.

More generally, under the rewriting axioms of eq. (1.1), a string w can be rewritten to the empty string – that is, $w \longrightarrow \cdots \longrightarrow \epsilon$ – if, and only if, that string ends with b. For example, the string abb ends with b, and abb can indeed be rewritten to the empty string:

$$a\,b\,b \longrightarrow b\,b \longrightarrow b \longrightarrow \epsilon\,.$$

In this way, the rewriting axioms of eq. (1.1) constitute a specification of a system that identifies those strings over the alphabet $\{a, b\}$ that end with b.

The usual operation semantics for string rewriting employs committed-choice nondeterminism, which can lead to stuck, or otherwise undesirable, states. For example, although abb certainly ends with b, the string abb can be rewritten to a, a stuck state, if incorrect choices about which axioms to apply are made:

$$a\,b\,b \longrightarrow a\,b \longrightarrow a \not\longrightarrow\,.$$

No backtracking is performed to reconsider these choices. [3]

[3] Bring up concurrency here?

THE REMAINDER OF THIS CHAPTER describes a string rewriting framework, SR, in more detail (section 1.1) and examines its properties, most importantly concurrent rewritings. Then we present two extended examples of how string rewriting in SR may be used to specify concurrent systems: nondeterministic finite automata (section 1.2) and binary representations of natural numbers (section 1.3). These will serve as recurring examples throughout the remainder of this document.

1.1 *A string rewriting framework*

In this section, we present a string rewriting framework that we dub SR.

1.1.1 *Symbols and strings*

String rewriting in SR presupposes an alphabet, Σ, of symbols a from which finite strings are constructed. This alphabet is usually, but need not be, finite.

Strings, w, are then finite lists of symbols: $w = a_1\, a_2 \cdots a_n$. Algebraically, strings form a free (noncommutative) monoid over symbols $a \in \Sigma$ and may be described syntactically by the grammar

$$w ::= w_1\, w_2 \mid \epsilon \mid a\,,$$

where the monoid operation is string concatenation, denoted by $w_1\, w_2$, and the unit element is the empty string, denoted by ϵ. As a monoid, strings are equivalent up to associativity and unit laws (see adjacent figure). We choose to keep this equivalence implicit, however, treating equivalent strings as syntactically indistinguishable.[4]

$$(w_1\, w_2)\, w_3 = w_1\, (w_2\, w_3)$$
$$\epsilon\, w = w = w\, \epsilon$$

Figure 1.1: The monoid laws for strings

[4] As usual for a free monoid, the alternative grammar, $w ::= \epsilon \mid a\, w$, describes the same strings.

1.1.2 *A rewriting relation*

At the heart of string rewriting is a binary relation, $\longrightarrow$, over strings. When $w \longrightarrow w'$, we say that w can be rewritten to w'. This relation is defined as the least compatible relation satisfying a collection of rewriting axioms, chosen on a per-application basis, such as the axioms

$$\frac{}{a\, b \longrightarrow b} \qquad \text{and} \qquad \frac{}{b \longrightarrow \epsilon} \tag{1.2}$$

shown earlier. More generally, an axiom is any pair of concrete, finite strings, $w \longrightarrow w'$, although axioms of the form $\epsilon \longrightarrow w'$ are expressly forbidden.

To be more formal, these axioms are collected into a signature, Θ, that indexes the rewriting relation:

$$\Theta ::= \cdot \mid \Theta, w \longrightarrow w' \quad (w \neq \epsilon)$$

The axioms of this signature may then be used via a $\longrightarrow$AX rule,

$$\frac{w \longrightarrow w' \in \Theta}{w \longrightarrow_{\Theta} w'} \longrightarrow\text{AX}\,.$$

Aside from this rule, all of the other rules for the rewriting relation simply pass on the signature Θ untouched; for this reason, we nearly always elide the signature index on the rewriting relation, writing $\longrightarrow$ instead of $\longrightarrow_\Theta$.

In addition to the application-specific axioms contained within the signature, rewriting is always permitted within substrings, so we adopt the rule

$$\frac{w_0 \longrightarrow w_0'}{w_1\, w_0\, w_2 \longrightarrow w_1\, w_0'\, w_2}\ {\longrightarrow}\mathrm{C}$$

to ensure that the rewriting relation is compatible with the monoidal structure of strings.

The $\longrightarrow$ relation thus describes the rewritings that are possible in a single step: exactly one axiom, perhaps embellished by the compatibility rules. In addition to these single-step rewritings, it will frequently be useful to describe the rewritings that are possible in some finite number of steps. For this, we construct a multi-step rewriting relation, $\Longrightarrow$, from the reflexive, transitive closure of $\longrightarrow$.[5]

[5] Usually written as $\longrightarrow^*$, we instead chose $\Longrightarrow$ for the reflexive, transitive closure because of its similarity with standard process calculus notation for weak transitions, $\overset{\alpha}{\Longrightarrow}$. Our reasons for this choice of notation will become clearer in subsequent chapters.

Consistent with its monoidal structure, there are two equivalent formulations of this reflexive, transitive closure: each rewriting sequence $w \Longrightarrow w'$ can be viewed as either a list or tree of individual rewriting steps. We prefer the list-based formulation,

$$\frac{}{w \Longrightarrow w}\ {\Longrightarrow}\mathrm{R} \qquad \text{and} \qquad \frac{w \longrightarrow w' \quad w' \Longrightarrow w''}{w \Longrightarrow w''}\ {\Longrightarrow}\mathrm{T}\ ,$$

because it tends to streamline proofs by structural induction. However, on the basis of the following lemma, we allow ourselves to freely switch between the two formulations as needed.

LEMMA 1.1 (Transitivity of $\Longrightarrow$). *If $w \Longrightarrow w'$ and $w' \Longrightarrow w''$, then $w \Longrightarrow w''$.*

Proof. By structural induction over the first of the given rewriting sequences, $w \Longrightarrow w'$. □

A summary of string rewriting is shown in fig. 1.2.

1.1.3 *Properties of the SR string rewriting framework*

As an abstract rewriting system, SR can be evaluated for several properties: confluence, termination, and, of particular interest to us, concurrency.

CONCURRENCY As an example multi-step rewriting sequence, observe that $a\,b\,b \Longrightarrow \epsilon$, under the axioms of our running example (??). In fact, as shown in the adjacent figure, multiple sequences witness this rewriting. The initial $a\,b$ can first be rewritten to b and then the terminal b can be rewritten to ϵ (upper half of figure); or vice versa: the terminal b can first be rewritten to ϵ and then the initial $a\,b$ can be rewritten to b (lower half of figure). In either

$$\begin{tikzcd}
& b\,b \arrow[dr] & \\
a\,b\,b \arrow[ur] \arrow[rr, Rightarrow] \arrow[dr] & & b \longrightarrow \epsilon \\
& a\,b \arrow[ur] &
\end{tikzcd}$$

Figure 1.3: An example of concurrent string rewriting

$$\textsf{STRINGS} \quad w ::= w_1\, w_2 \mid \epsilon \mid a$$

$$\textsf{SIGNATURES} \quad \Theta ::= \cdot \mid \Theta, w \longrightarrow w' \qquad (w \neq \epsilon)$$

$$(w_1\, w_2)\, w_3 = w_1\, (w_2\, w_3)$$
$$\epsilon\, w = w = w\, \epsilon$$

$$\frac{w \longrightarrow w' \in \Theta}{w \longrightarrow_\Theta w'} \longrightarrow\text{AX} \qquad \frac{w_0 \longrightarrow w_0'}{w_1\, w_0\, w_2 \longrightarrow w_1\, w_0'\, w_2} \longrightarrow\text{C}$$

$$\frac{}{w \Longrightarrow w} \Longrightarrow\text{R} \qquad \frac{w \longrightarrow w' \quad w' \Longrightarrow w''}{w \Longrightarrow w''} \Longrightarrow\text{T}$$

Figure 1.2: The SR framework for string rewriting

case, the remaining b (which is the leftmost of the original bs) can finally be rewritten to ϵ.

Notice that these two sequences differ only in how non-overlapping, and therefore independent, rewritings of the string's two segments are interleaved. Consequently, the two sequences can be – and indeed should be – considered essentially equivalent. The details of how the individual, small steps are interleaved are irrelevant, so that – conceptually at least – only the big-step sequence from $a\,b\,b$ to b (and ultimately ϵ) remains (middle of figure).

In contrast, a third rewriting sequence does not admit this reordering: the leftmost b is rewritten first to ϵ and then the resulting $a\,b$ is rewritten to b (and ultimately ϵ). This sequence's two rewriting steps are not independent because the b that participates in the rewriting of $a\,b$ is not adjacent to the a until the first rewriting step occurs. [6]

[6] explain figure

More generally, this idea that the interleaving of independent actions is irrelevant is known as *concurrent equality*,[7] and it forms the basis of concurrency.[8] With the partial commutativity endowed by concurrent equality, the [free] monoid formed by rewriting sequences is, more specifically, a trace monoid. As such, we will frequently refer to rewriting sequences as *traces*.

$$\begin{tikzcd}
 & b_1\, b_2 \arrow[r] \arrow[dr] & b_2 \arrow[dr] & \\
a\, b_1\, b_2 \arrow[ur] \arrow[rr, Rightarrow] \arrow[dr] \arrow[ddr] & & b_1 \arrow[r] & \epsilon \\
 & a\, b_1 \arrow[ur] \arrow[r] & a \nrightarrow & \\
 & a\, b_2 \arrow[uuur] \arrow[ur] & &
\end{tikzcd}$$

Figure 1.4: When multiple occurences of b are properly distinguished, a complete trace diagram can be given.

[7] **Watkins+:CMU02**.

[8] **??**.

NON-CONFLUENCE We may also evaluate SR for confluence. Confluence requires that all strings with a common ancestor be joinable, *i.e.*, that $w_1' \Longleftarrow\Longrightarrow w_2'$ implies $w_1' \Longrightarrow\Longleftarrow w_2'$, for all strings w_1' and w_2'.

Because string rewriting is an asymmetric, committed-choice relation, some nondeterministic choices are irreversible. For example, under the axioms of our running example (eq. (1.2)), $a\,b$ can be nondeterministically rewritten into either a or ϵ, as shown in fig. 1.4. However, neither a nor ϵ can be rewritten, so confluence fails to hold for string rewriting in general.

NON-TERMINATION In our running example, rewriting always terminates: each possible rewriting step removes exactly one symbol, and each string contains only finitely many symbols.

In general, however, string rewriting does not terminate even though strings are finite. For a simple example, consider rewriting of strings over the alphabet $\{a, b\}$ with axioms

$$\overline{a \longrightarrow b} \qquad \text{and} \qquad \overline{b \longrightarrow a}\,.$$

Every finite trace from a nonempty string can always be extended by applying one of these axioms, so rewriting in SR never terminates.

1.2 *Extended example: Nondeterministic finite automata*

As an extended example of string rewriting in SR, we will specify how a nondeterministic finite automaton (NFA) processes its input. Beginning with this specification, NFAs will serve as a recurring example throughout the remainder of this document.

Given an NFA $\mathcal{A} = (Q, ?, F)$[9] over an input alphabet Σ, the idea is to introduce a string rewriting axiom for each transition that the NFA can make:

$$\overline{a\,q \longrightarrow q'_a} \;\text{ for each transition } q \xrightarrow{a} q'_a.$$

[9] fix

In addition, the NFA's acceptance criteria is captured by introducing a distinguished symbol[10] ϵ to act as an end-of-word marker, along with axioms[11]

[10] replace with $?

[11] Check these with choreography

$$\overline{\epsilon\,q \longrightarrow F(q)} \;\text{ for each state } q\text{, where } F(q) = \begin{cases} (\cdot) & \text{if } q \in F \\ n & \text{if } q \notin F\,. \end{cases}$$

These axioms imply that rewriting occurs over the strings $\{\epsilon\} \times \Sigma^* \times Q$.

For a concrete instance of this encoding, recall from **??** the NFA (repeated in the adjacent figure) that accepts exactly those words, over the alphabet $\Sigma = \{a, b\}$, that end with b; that NFA is specified by the following string rewriting axioms:

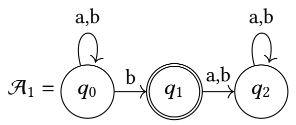


Figure 1.5: An NFA that accepts, from state q_0, exactly those words that end with b. (Repeated from **??**.)

$$\begin{array}{cccc}
\overline{a\,q_0 \longrightarrow q_0} & \overline{b\,q_0 \longrightarrow q_0} \text{ and } \overline{b\,q_0 \longrightarrow q_1} & \overline{\epsilon\,q_0 \longrightarrow n} \\
\overline{a\,q_1 \longrightarrow q_2} & \overline{b\,q_1 \longrightarrow q_2} & \overline{\epsilon\,q_1 \longrightarrow \cdot} \\
\overline{a\,q_2 \longrightarrow q_2} & \overline{b\,q_2 \longrightarrow q_2} & \overline{\epsilon\,q_2 \longrightarrow n}
\end{array}$$

Indeed, just as the NFA $\mathcal{A}_1$ accepts the input word abb, its rewriting specification admits a trace

$$\epsilon\,b\,b\,a\,q_0 \longrightarrow \epsilon\,b\,b\,q_0 \longrightarrow \epsilon\,b\,q_0 \longrightarrow \epsilon\,q_1 \longrightarrow (\cdot)\,.$$

More generally, this string rewriting specification of NFAs adequately describes their operational semantics, in the sense that it simulates all NFA transitions. Given the reversal (anti-)homomorphism for finite words defined in the adjacent figure, we can prove the following adequacy result.

$$(w_1\,w_2)^{\mathrm{R}} = w_2^{\mathrm{R}}\,w_1^{\mathrm{R}}$$
$$\epsilon^{\mathrm{R}} = \cdot$$
$$a^{\mathrm{R}} = a$$

Figure 1.6: An (anti-)homomorphism for reversal of finite words

Theorem 1.2 (Adequacy of NFA specification). *Let $\mathcal{A} = (Q, ?, F)$*[12] *be an NFA over the input alphabet Σ.*

[12] fix

- $q \xrightarrow{a} q'_a$ *if, and only if,* $a\, q \longrightarrow q'_a$*, for all input symbols* $a \in \Sigma$.
- $q \in F$ *if, and only if,* $\epsilon\, q \longrightarrow (\cdot)$.
- $q \xrightarrow{w} q'$ *if, and only if,* $w^{\mathrm{R}}\, q \Longrightarrow q'$*, for all finite words* $w \in \Sigma^*$.

Proof. The first two parts follow immediately from the NFA's specification; the third part follows by induction over the structure of the input word w. □

This adequacy theorem is relatively straightforward to state and prove because string rewriting is a good match for labeled transition systems, like the one that defines an NFA's operational semantics. On the other hand, when a system is not so clearly based on a labeled transition system, stating and proving the adequacy of its string rewriting specification becomes a bit more involved. This is the case for the next example, binary representations of natural numbers.

1.3 *Extended example: Binary representations of natural numbers*

For a second recurring example, we will use binary representations of natural numbers equipped with increment and decrement operations, or *binary counters*. Here we present a string rewriting specification of these binary counters.

1.3.1 *Binary representations*

In this setting, we represent a natural number in binary by a string that consists of a big-endian sequence of symbols b_0 and b_1, prefixed by the symbol e; leading b_0s are permitted. For example, both $\Omega = e\, b_1$ and $\Omega' = e\, b_0\, b_1$ are valid binary representations of the natural number 1.

To be more precise, we inductively define a relation, $\approx_{\mathrm{V}}$, that assigns to each binary representation a unique natural number denotation. If $\Omega \approx_{\mathrm{V}} n$, we say that Ω denotes, or represents, natural number n in binary.

$$\frac{}{e \approx_{\mathrm{V}} 0}\ e\text{-V} \qquad \frac{\Omega \approx_{\mathrm{V}} n}{\Omega\, b_0 \approx_{\mathrm{V}} 2n}\ b_0\text{-V} \qquad \frac{\Omega \approx_{\mathrm{V}} n}{\Omega\, b_1 \approx_{\mathrm{V}} 2n+1}\ b_1\text{-V}$$

Besides providing a denotational semantics of binary numbers, the $\approx_{\mathrm{V}}$ relation also serves to implicitly characterize the well-formed binary numbers as those strings Ω that form the relation's domain of definition.[13]

[13] Alternatively, the well-formed binary numbers could be described more explicitly by the grammar

$$\Omega ::= e \mid \Omega\, b_0 \mid \Omega\, b_1\,,$$

and then their denotations could be expressed in a more functional manner:

$$[\![e]\!]_{\mathrm{V}} = 0$$
$$[\![\Omega\, b_0]\!]_{\mathrm{V}} = 2[\![\Omega]\!]_{\mathrm{V}}$$
$$[\![\Omega\, b_1]\!]_{\mathrm{V}} = 2[\![\Omega]\!]_{\mathrm{V}} + 1\,.$$

The adequacy of the $\approx_{\mathrm{V}}$ relation is proved as the following theorem.

Theorem 1.3 (Adequacy of binary representations). *Binary representations and their $\approx_{\mathrm{V}}$ relation are:*

Functional *For each binary number Ω, there exists a unique natural number n such that $\Omega \approx_{\mathrm{V}} n$.*

Surjective *For each natural number n, there exists a binary number Ω such that $\Omega \approx_{\mathrm{V}} n$.*

Latent *If* $\Omega \approx_{\mathrm{V}} n$*, then* $\Omega \not\longrightarrow$.

Proof. The three claims may be proved by induction over the structure of Ω, and by induction on n, respectively. □

Notice that the above e-V and b_0-V rules overlap when the denotation is 0, giving rise to the leading b_0s that make the $\approx_{\mathrm{V}}$ relation non-injective: for example, both $e\, b_1 \approx_{\mathrm{V}} 1$ and $e\, b_0\, b_1 \approx_{\mathrm{V}} 1$ hold. However, if the rule for b_0 is restricted to *nonzero* even numbers, then each natural number has a unique, canonical representation that is free of leading b_0s.[14]

[14] A restriction of the b_0 rule to nonzero even numbers is:

$$\frac{\Omega \approx_{\mathrm{V}} n \quad (n > 0)}{\Omega\, b_0 \approx_{\mathrm{V}} 2n}\,.$$

The leading-b_0-free representations could alternatively be seen as the canonical representatives of the equivalence classes induced by the relation among binary numbers that have the same denotation: $\Omega \equiv \Omega'$ if $\Omega \approx_{\mathrm{V}} n$ and $\Omega' \approx_{\mathrm{V}} n$ for some n.

1.3.2 *An increment operation*

To use string rewriting to describe an increment operation on binary representations, we introduce a new symbol, i, that will serve as an increment instruction.

Given a binary number Ω that represents n, we may append i to form an active[15] , computational string, $\Omega\, i$. For i to adequately represent the increment operation, the string $\Omega\, i$ must meet two conditions, captured by the following global desiderata:

[15] The 'active', 'latent', and 'passive' terminology is borrowed from **Pfenning+Simmons:LICS09**. Active strings are immediately rewritable, but latent strings are rewritable only when combined with other, passive strings. The blurry line between latent and passive strings is exploited in ?? when we discuss choreographies.

- $\Omega\, i \Longrightarrow\approx_{\mathrm{V}} n+1$ – that is, *some* rewriting sequence results in a binary representation of $n+1$; and
- $\Omega\, i \Longrightarrow \Omega'$ implies $\Omega' \Longrightarrow\approx_{\mathrm{V}} n+1$ – that is, *any* rewriting sequence from $\Omega\, i$ can[16] result in a binary representation of $n+1$.

[16] ??

For example, because $e\, b_1$ denotes 1, a computation $e\, b_1\, i \Longrightarrow\approx_{\mathrm{V}} 2$ must exist; moreover, every computation $e\, b_1\, i \Longrightarrow\approx_{\mathrm{V}} n'$ must satisfy $n' = 2$.

To ACHIEVE THESE global desiderata, we introduce three string rewriting axioms that describe how the symbols e, b_0, and b_1 may be rewritten when they encounter i, the increment instruction:

$$\frac{}{e\, i \longrightarrow e\, b_1} \qquad \frac{}{b_0\, i \longrightarrow b_1} \qquad \text{and} \qquad \frac{}{b_1\, i \longrightarrow i\, b_0}\,.$$

These three axioms can be read as follows:

- To increment e, append b_1 as a new most[17] significant bit, resulting in $e\, b_1$.
- To increment a binary number ending in b_0, flip that bit to b_1.
- To increment a binary number ending in b_1, flip that bit to b_0 and carry the increment over to the more significant bits.

[17] or least?

Comfortingly, $1 + 1 = 2$: a trace $e\, b_1\, i \longrightarrow e\, i\, b_0 \longrightarrow e\, b_1\, b_0$ indeed exists.

Owing to the notion of concurrent equality that string rewriting admits, increments may even be performed concurrently. For example, there are two rewriting sequences that witness $e\,b_1\,i\,i \Longrightarrow e\,b_1\,b_1$:

$$\begin{tikzcd}
 & & e\,b_1\,b_0\,i \arrow[dr] & \\
e\,b_1\,i\,i \arrow[r] & e\,i\,b_0\,i \arrow[ur] \arrow[dr] \arrow[rr, Rightarrow, dashed] & & e\,b_1\,b_1 \\
 & & e\,i\,b_1 \arrow[ur] &
\end{tikzcd}$$

In other words, once the left most increment is carried past the least significant bit, the two increments can be interleaved, with no observable difference in the outcome.

THESE INCREMENT AXIOMS introduce strings that occur as intermediate computational states within traces, such as $e\,i\,b_0\,i$ and $e\,i\,b_1$ in the above diagram. To characterize the valid intermediate strings, we define a binary relation, $\approx_{\text{I}}$, that assigns a natural number denotation to each such intermediate string, not only to the terminal values, as $\approx_{\text{V}}$ did.[18]

[18] Like the $\approx_{\text{V}}$ relation does for values, the $\approx_{\text{I}}$ relation also serves to implicitly characterize the valid intermediate states as those contexts that form the relation's domain of definition. As with values, the valid intermediate states could also be enumerated more explicitly and syntactically with a grammar and denotation function:

$$\Omega ::= e \mid \Omega\,b_0 \mid \Omega\,b_1 \mid \Omega\,i$$

$$\begin{aligned}
[\![e]\!]_{\text{I}} &= 0\\
[\![\Omega\,b_0]\!]_{\text{I}} &= 2[\![\Omega]\!]_{\text{I}}\\
[\![\Omega\,b_1]\!]_{\text{I}} &= 2[\![\Omega]\!]_{\text{I}} + 1\\
[\![\Omega\,i]\!]_{\text{I}} &= [\![\Omega]\!]_{\text{I}} + 1
\end{aligned}$$

$$\frac{}{e \approx_{\text{I}} 0}\;e\text{-I} \qquad \frac{\Omega \approx_{\text{I}} n}{\Omega\,b_0 \approx_{\text{I}} 2n}\;b_0\text{-I} \qquad \frac{\Omega \approx_{\text{I}} n}{\Omega\,b_1 \approx_{\text{I}} 2n+1}\;b_1\text{-I} \qquad \frac{\Omega \approx_{\text{I}} n}{\Omega\,i \approx_{\text{I}} n+1}\;i\text{-I}$$

Binary values should themselves be valid, terminal computational states, so the first three rules are carried over from the $\approx_{\text{V}}$ relation. The i-I rule allows multiple increment instructions to be interspersed throughout the state.

With this $\approx_{\text{I}}$ relation in hand, we can now prove a stronger, small-step adequacy theorem. This small-step theorem then implies the big-step desiderata from above.

THEOREM 1.4 (Small-step adequacy of increments).
Value soundness *If* $\Omega \approx_{\text{V}} n$, *then* $\Omega \approx_{\text{I}} n$ *and* $\Omega \not\longrightarrow$.
Preservation *If* $\Omega \approx_{\text{I}} n$ *and* $\Omega \longrightarrow \Omega'$, *then* $\Omega' \approx_{\text{I}} n$.
Progress *If* $\Omega \approx_{\text{I}} n$, *then either:* $\Omega \longrightarrow \Omega'$ *for some* Ω'; *or;* $\Omega \approx_{\text{V}} n$.[19]
Termination *If* $\Omega \approx_{\text{I}} n$, *then every rewriting sequence from* Ω *is finite.*

[19] Compare with "If $\Omega \approx_{\text{I}} n$, then $\Omega \approx_{\text{V}} n$ if, and only if, $\Omega \not\longrightarrow$."

Proof. Each part is proved separately.

Value soundness can be proved by structural induction on the derivation of $\Omega \approx_{\text{V}} n$.

Preservation and progress can likewise be proved by structural induction on the derivation of $\Omega \approx_{\text{I}} n$.

Termination can be proved using an explicit termination measure, $|{-}|_{\text{I}}$, that is strictly decreasing across each rewriting, $\Omega \longrightarrow \Omega'$. Specifically, we use a measure (see the adjacent figure), adapted from the standard amortized constant work analysis of increment for binary counters.[20] The measure $|{-}|_{\text{I}}$ is such that $\Omega \longrightarrow \Omega'$ implies $|\Omega|_{\text{I}} > |\Omega'|_{\text{I}}$; because the measure is always nonnegative, only finitely many such rewritings can occur.

As an example case, consider the intermediate state $\Omega\,b_1\,i$ and its rewriting $\Omega\,b_1\,i \longrightarrow \Omega\,i\,b_0$. Indeed, $|\Omega\,b_1\,i|_{\text{I}} = |\Omega|_{\text{I}} + 3 > |\Omega|_{\text{I}} + 2 = |\Omega\,i\,b_0|_{\text{I}}$. □

$$\begin{aligned}
|e|_{\text{I}} &= 0\\
|\Omega\,b_0|_{\text{I}} &= |\Omega|_{\text{I}}\\
|\Omega\,b_1|_{\text{I}} &= |\Omega|_{\text{I}} + 1\\
|\Omega\,i|_{\text{I}} &= |\Omega|_{\text{I}} + 2
\end{aligned}$$

Figure 1.7: A termination measure, adapted from the standard amortized work analysis of increment for binary counters

[20] ??.

COROLLARY 1.5 (Big-step adequacy of increments).

Evaluation *If $\Omega \approx_{\mathrm{I}} n$, then $\Omega \Longrightarrow\approx_{\mathrm{V}} n$. In particular, if $\Omega \approx_{\mathrm{V}} n$, then $\Omega\, i \Longrightarrow\approx_{\mathrm{V}} n+1$.*

Preservation *If $\Omega \approx_{\mathrm{I}} n$ and $\Omega \Longrightarrow \Omega'$, then $\Omega' \approx_{\mathrm{I}} n$. In particular, if $\Omega \approx_{\mathrm{V}} n$ and $\Omega\, i \Longrightarrow \Omega'$, then $\Omega' \Longrightarrow\approx_{\mathrm{V}} n+1$.*

Proof. The two parts are proved separately.

Evaluation can be proved by repeatedly appealing to the progress and preservation results (theorem 1.4). By the accompanying termination result, a binary value must eventually be reached.

Preservation can be proved by structural induction on the given trace. □

1.3.3 *A decrement operation*

Binary counters may also be equipped with a decrement operation. Instead of examining decrements *per se*, we will describe a very closely related operation: the normalization of binary representations to what might be called *head-unary form*. (We will frequently abuse terminology, using 'head-unary normalization' and 'decrement operation' interchangably.) A string Ω will be said to be in head-unary form if it has one of two forms: $\Omega = z$; or $\Omega = \Omega'\, s$, for some binary number Ω'.

Just as appending the symbol i to a counter Ω initiates an increment, appending a symbol d will cause the counter to begin normalizing to head-unary form. For d to adequately represent this operation, the string $\Omega\, d$ must satisfy the following global desiderata when $\Omega \approx_{\mathrm{D}} n$:

- $\Omega\, d \Longrightarrow z$ if, and only if, $n = 0$;
- $\Omega\, d \Longrightarrow \Omega'\, s$ for some Ω' such that $\Omega' \approx_{\mathrm{V}} n-1$, if $n > 0$; and
- $\Omega\, d \Longrightarrow \Omega'\, s$ only if $n > 0$ and $\Omega' \approx_{\mathrm{V}} n-1$.

For example, because $e\, b_1$ denotes 1, a trace $e\, b_1\, d \Longrightarrow \Omega'\, s$ must exist, for some $\Omega' \approx_{\mathrm{V}} 0$.

To ACHIEVE THESE global desiderata, we introduce three additional axioms that describe how the symbols e, b_0, and b_1 may be rewritten when they encounter d, the decrement instruction; also, an intermediate symbol b_0' and two more axioms are introduced:

$$\frac{}{e\, d \longrightarrow z} \qquad \frac{}{b_1\, d \longrightarrow b_0\, s} \qquad \frac{}{b_0\, d \longrightarrow d\, b_0'}$$

$$\frac{}{z\, b_0' \longrightarrow z} \quad \text{and} \quad \frac{}{s\, b_0' \longrightarrow b_1\, s}\,.$$

These five axioms can be read as follows:

- Because e denotes 0, its head-unary form is simply z.

- Because $\Omega\, b_1$ denotes $2n{+}1$ if Ω denotes n, its head-unary form, $\Omega\, b_0\, s$, can be constructed by flipping the least significant bit to b_0 and appending s.
- Because $\Omega\, b_0$ denotes $2n$ if Ω denotes n, its head-unary form can be contructed by recursively putting the more significant bits, Ω, into head-unary form and appending b'_0 to process that result.
  - If Ω has head-unary form z and therefore denotes 0, then $\Omega\, b_0$ also denotes 0 and has head-unary form z.
  - Otherwise, if Ω has head-unary form $\Omega'\, s$ and thus denotes $n > 0$, then $\Omega\, b_0$ denotes $2n > 0$ and has head-unary form $\Omega'\, b_1\, s$, which can be constructed by replacing s with $b_1\, s$.

Comfortingly, $(1 + 1) - 1 = 1$: the head-unary form of $e\, b_1\, i$ is $e\, b_0\, b_1\, s$:

$$\begin{tikzcd}
 & & e\, b_1\, b_0\, d \arrow[dr] & & & \\
e\, b_1\, i\, d \arrow[r] & e\, i\, b_0\, d \arrow[ur] \arrow[dr] \arrow[rr, Rightarrow, dashed] & & e\, b_1\, d\, b'_0 \arrow[r] & e\, b_0\, s\, b'_0 \arrow[r] & e\, b_0\, b_1\, s\,. \\
 & & e\, i\, d\, b'_0 \arrow[ur] & & &
\end{tikzcd}$$

Note the concurrency that derives from the independence of the increment and decrement after the initial step of rewriting.

These decrement axioms introduce more strings that may occur as intermediate computational states. As before, we define a new binary relation, $\approx_{\mathrm{D}}$, that assigns a natural number denotation to each string that may appear as an intermediate state during a decrement.

$$\frac{\Omega \approx_{\mathrm{I}} n}{\Omega\, d \approx_{\mathrm{D}} n}\ d\text{-D} \qquad \frac{\Omega \approx_{\mathrm{D}} n}{\Omega\, b'_0 \approx_{\mathrm{D}} 2n}\ b'_0\text{-D} \qquad \frac{}{z \approx_{\mathrm{D}} 0}\ z\text{-D} \qquad \frac{\Omega \approx_{\mathrm{I}} n}{\Omega\, s \approx_{\mathrm{D}} n+1}\ s\text{-D}$$

At first glance, the d-D rule may look a bit odd: Why is the denotation unchanged by a decrement, $\Omega\, d$? Because the operation is more accurately characterized as head-unary normalization, it makes sense that the denotation remains unchanged. The operation described by d does not change the binary counter's value – it only expresses that same value in a different form.[21]

[21] Once again, the valid intermediate states could also be enumerated more explicitly and syntactically with a grammar and denotation function:

$$\Omega ::= e \mid \Omega\, b_0 \mid \Omega\, b_1 \mid \Omega\, i$$
$$\Delta ::= \Omega\, d \mid \Delta\, b'_0 \mid z \mid \Omega\, s$$

$$[\![\Omega\, d]\!]_{\mathrm{D}} = [\![\Omega]\!]_{\mathrm{I}}$$
$$[\![\Delta\, b'_0]\!]_{\mathrm{D}} = 2[\![\Delta]\!]_{\mathrm{D}}$$
$$[\![z]\!]_{\mathrm{D}} = 0$$
$$[\![\Omega\, s]\!]_{\mathrm{D}} = [\![\Omega]\!]_{\mathrm{I}} + 1$$

Also, notice that the premises of the d-D and s-D rules use the increment-only denotation relation, $\approx_{\mathrm{I}}$, not the decrement relation, $\approx_{\mathrm{D}}$. These choices ensure that each counter has at most one d and may not have any i or s symbols to the right of that d. But the premise of the b'_0-D does use the $\approx_{\mathrm{D}}$ relation, so d may have b'_0 symbols to its right.

With this $\approx_{\mathrm{D}}$ relation in hand, we can now prove a small-step adequacy theorem. This small-step theorem then implies the big-step desiderata from above.

Theorem 1.6 (Small-step adequacy of decrements).
Preservation *If $\Omega \approx_{\mathrm{D}} n$ and $\Omega \longrightarrow \Omega'$, then $\Omega' \approx_{\mathrm{D}} n$.*
Progress *If $\Omega \approx_{\mathrm{D}} n$, then [either]:*[22]

- *$\Omega \longrightarrow \Omega'$, for some Ω';*

[22] ?

- *$n = 0$ and $\Omega = z$; or*
- *$n > 0$ and $\Omega = \Omega'\, s$, for some Ω' such that $\Omega' \approx_{\mathrm{I}} n - 1$.*

Termination *If $\Omega \approx_{\mathrm{D}} n$, then every rewriting sequence from Ω is finite.*

Proof. Each part is proved separately.

Preservation and progress are proved, as before, by structural induction on the given derivation of $\Omega \approx_{\mathrm{D}} n$.

Termination is proved by exhibiting a measure, $|{-}|_{\mathrm{D}}$, given in the adjacent figure, that is strictly decreasing across each rewriting. Unlike the amortized constant work increments (see proof of theorem 1.4), this measure assigns a linear amount of potential to the decrement instruction.[23]

This measure is strictly decreasing across each rewriting: $\Omega \longrightarrow \Omega'$ only if $|\Omega|_{\mathrm{D}} > |\Omega'|_{\mathrm{D}}$. As an example case, consider the intermediate state $\Omega\, b_0\, d$ and its rewriting $\Omega\, b_0\, d \longrightarrow \Omega\, d\, b_0'$. Indeed,

$$|\Omega\, b_0\, d|_{\mathrm{D}} = |\Omega|_{\mathrm{I}} + 3|\Omega| + 3 > |\Omega|_{\mathrm{I}} + 3|\Omega| + 2 = |\Omega\, d\, b_0'|_{\mathrm{D}}\,. \qquad \square$$

$$\begin{aligned} |\Omega\, d|_{\mathrm{D}} &= |\Omega|_{\mathrm{I}} + 3|\Omega| \\ |\Omega\, b_0'|_{\mathrm{D}} &= |\Omega|_{\mathrm{D}} + 2 \\ |z|_{\mathrm{D}} &= 0 \\ |\Omega\, s|_{\mathrm{D}} &= |\Omega|_{\mathrm{I}} \end{aligned}$$

Figure 1.8: A termination measure for decrements, where $|\Omega|$ denotes the length of string Ω

[23] Actually, because the increment and decrement operations are defined only for binary representations, not head-unary forms, there can be at most one d. Therefore, it is actually possible to assign a constant amount of potential to each d. However, doing so would rely on a somewhat involved lexicographic measure that isn't particularly relevant to our aims in this document, so we use the simpler linear potential.

COROLLARY 1.7 (Big-step adequacy of decrements). *If $\Omega \approx_{\mathrm{D}} n$, then:*

- *$\Omega \Longrightarrow z$ if, and only if, $n = 0$;*
- *$\Omega \Longrightarrow \Omega'\, s$ for some Ω' such that $\Omega' \approx_{\mathrm{I}} n - 1$, if $n > 0$; and*
- *$\Omega \Longrightarrow \Omega'\, s$ only if $n > 0$ and $\Omega' \approx_{\mathrm{I}} n - 1$.*

Proof. From the small-step preservation result of theorem 1.6, it is possible to prove, using a structural induction on the given trace, a big-step preservation result: namely, that $\Omega \approx_{\mathrm{D}} n$ and $\Omega \Longrightarrow \Omega'$ only if $\Omega' \approx_{\mathrm{D}} n$. Each of the above claims then follows from either progress and termination (theorem 1.6) or big-step preservation together with inversion. $\square$

2

Ordered rewriting

In this chapter, we develop a rewriting interpretation of the ordered sequent calculus from the previous ??.

In **Lambek:AMM58**, **Lambek:AMM58** developed a syntactic calculus, now known as the Lambek calculus, for formally describing the structure of sentences.[1] Words are assigned syntactic types, which roughly correspond to grammatical parts of speech. From a logical perspective, the Lambek calculus can [also] be viewed as a precursor to (and generalization of) **Girard:TCS87**'s linear logic[2].[3] Implicit in **Lambek:AMM58**'s original article is a third perspective of the calculus: string rewriting.

[1] **Lambek:AMM58**.

[2] **Girard:TCS87**.

[3] **Polakow+Pfenning:MFPS99Polakow+Pfenning:TLCA99**.

In this chapter, we review the Lambek calculus from a [string] rewriting perspective.

The previous chapter showed how to use string rewriting to specify, on a global level, the [...] of concurrent systems that have a linear topology. Although useful for [...], these string rewriting specifications lack a clear notion of local, decentralized execution – for each step of rewriting, the entire string is rewritten as a monolithic whole by a central conductor.

Keeping in mind our ultimate goal of decentralized[4] implementations of concurrent systems, these string rewriting specifications are too abstract. Instead, we need to expose local interactions that are left implicit in the string rewriting specifications.

[4] distributed?

As an example, recall from chapter 1 the string rewriting specification of a system that may transform strings that end with b into the empty string:

$$\frac{}{a\,b \longrightarrow b} \qquad \frac{}{b \longrightarrow \cdot}\,. \tag{2.1}$$

This specification is non-local in two ways: the central conductor must identify those substrings that can be rewritten according to one of the axioms. In the [...] axiom, for example, there is no description of how the symbols a and b would identify each other and coordinate to effect a rewriting to b.

To [...], we introduce *choreographies*, which refine string rewriting specifications by consistently assigning each symbol one of two roles: message or process.

$$\frac{}{\underset{\rightarrow}{a}\,\hat{b} \longrightarrow \hat{b}} \qquad \text{and} \qquad \frac{}{\hat{b} \longrightarrow \cdot}$$

a recursively defined ordered proposition, such as

$$\hat{b} \triangleq (\underset{\rightarrow}{a} \setminus {\uparrow}{\downarrow}\hat{b}) \mathbin{\&} \mathbf{1}$$

for the process $\hat{b}$.

The remainder of this chapter presents a formulation of the Lambek calculus from the ordered sequent calculus of **??**.

Then, in

One valid choreography for this specification views each symbol b as a process that nondeterministically receives some number of messages a before terminating.

If we annotate messages with an underbar and processes with a circumflex, then $\underline{a}\,\hat{b} \longrightarrow \hat{b}$ and $\hat{b} \longrightarrow \cdot$.

2.1 *Ordered resource decomposition as rewriting*

2.1.1 *Most left rules decompose ordered resources*

Recall two of the ordered sequent calculus's left rules:

$$\frac{\Omega'_L\; A\; B\; \Omega'_R \vdash C}{\Omega'_L\; (A \bullet B)\; \Omega'_R \vdash C}\;{\bullet}\textsc{l} \qquad \text{and} \qquad \frac{\Omega'_L\; A\; \Omega'_R \vdash C}{\Omega'_L\; (A \mathbin{\&} B)\; \Omega'_R \vdash C}\;{\&}\textsc{l}_1\,.$$

Both rules decompose the principal resource: in the $\bullet$L rule, $A \bullet B$ into the separate resources $A\,B$; and, in the $\&$L$_1$ rule, $A \mathbin{\&} B$ into A. However, in both cases, the resource decomposition is somewhat obscured by boilerplate. The framed contexts Ω'_L and Ω'_R and goal C serve to enable the rules to be applied anywhere in the list of resources, without restriction; these concerns are not specific to the $\bullet$L and $\&$L$_1$ rules, but are general boilerplate that arguably should be factored out.

To decouple the resource decomposition from the surrounding boilerplate, we will introduce a new judgment, $\Omega \longrightarrow \Omega'$, meaning "Resources Ω may be decomposed into resources Ω'." The choice of notation for this judgment is not coincidental: resource decomposition is a generalization of the string rewriting shown in chapter 1.

With this new decomposition judgment comes a cut principle, CUT$^{\longrightarrow}$, into which all of the boilerplate is factored:

$$\frac{\Omega \longrightarrow \Omega' \qquad \Omega'_L\; \Omega'\; \Omega'_R \vdash C}{\Omega'_L\; \Omega\; \Omega'_R \vdash C}\;\textsc{cut}^{\longrightarrow}\,.$$

The standard left rules can then be recovered from resource decomposition rules using this cut principle. For example, the decomposition of $A \bullet B$ into $A\,B$ is captured by

$$\frac{}{A \bullet B \longrightarrow A\,B}\;{\bullet}\textsc{d}\,,$$

$$\frac{\Omega'_L\, A\, B\, \Omega'_R \vdash C}{\Omega'_L\, (A \bullet B)\, \Omega'_R \vdash C}\;{\bullet}\textsc{l}$$

$$\leftrightsquigarrow$$

$$\frac{\dfrac{}{A \bullet B \longrightarrow A\, B}\;{\bullet}\textsc{d} \qquad \Omega'_L\, A\, B\, \Omega'_R \vdash C}{\Omega'_L\, (A \bullet B)\, \Omega'_R \vdash C}\;\textsc{cut}^{\longrightarrow}$$

Figure 2.1: Refactoring the $\bullet$L rule in terms of resource decomposition

and the standard $\bullet$L rule can then be recovered as shown in the adjacent figure. The left rules for **1** and $A \mathbin{\&} B$ can be similarly refactored into the resource decomposition rules

$$\frac{}{\mathbf{1} \longrightarrow \cdot}\;\mathbf{1}\textsc{d} \qquad \frac{}{A \mathbin{\&} B \longrightarrow A}\;\&\textsc{d}_1 \qquad \text{and} \qquad \frac{}{A \mathbin{\&} B \longrightarrow B}\;\&\textsc{d}_2 \,.$$

Even the left rules for left- and right-handed implications can be refactored in this way, despite the additional, minor premises that those rules carry. To keep the correspondence between resource decomposition rules and left rules as close as possible, we could introduce the decomposition rules

$$\frac{\Omega \vdash A}{\Omega\, (A \setminus B) \longrightarrow B}\;{\setminus}\textsc{d}' \qquad \text{and} \qquad \frac{\Omega \vdash A}{(B \mathbin{/} A)\, \Omega \longrightarrow B}\;{/}\textsc{d}' \,. \tag{2.2}$$

Just as for ordered conjunction, the left rules for left- and right-handed implication would then be recoverable via the $\textsc{cut}^{\longrightarrow}$ rule (see adjacent figure).

$$\frac{\Omega \vdash A \qquad \Omega'_L\, B\, \Omega'_R \vdash C}{\Omega'_L\, \Omega\, (A \setminus B)\, \Omega'_R \vdash C}\;{\setminus}\textsc{l}$$

$$\leftrightsquigarrow$$

$$\frac{\dfrac{\Omega \vdash A}{\Omega\, (A \setminus B) \longrightarrow B}\;{\setminus}\textsc{d}' \qquad \Omega'_L\, B\, \Omega'_R \vdash C}{\Omega'_L\, \Omega\, (A \setminus B)\, \Omega'_R \vdash C}\;\textsc{cut}^{\longrightarrow}$$

Figure 2.2: A possible refactoring of the $\setminus$L rule in terms of resource decomposition

Although these rules keep the correspondence between resource decomposition rules and left rules close, they differ from the other decomposition rules in two significant ways. First, the above $\setminus\textsc{d}'$ and $/\textsc{d}'$ rules have premises, and those premises create a dependence of the decomposition judgment upon general provability. Second, the above $\setminus\textsc{d}'$ and $/\textsc{d}'$ rules do not decompose the principal proposition into *immediate* subformulas since Ω is involved. This contrasts with, for example, the $\bullet$D rule that decomposes $A \bullet B$ into the immediate subformulas $A\, B$.

For these reasons, the above $\setminus\textsc{d}'$ and $/\textsc{d}'$ rules are somewhat undesirable. Fortunately, there is an alternative. Filling in the $\Omega \vdash A$ premises with the $\textsc{id}^A$ rule, we arrive at the derivable rules

$$\frac{}{A\, (A \setminus B) \longrightarrow B}\;{\setminus}\textsc{d} \qquad \text{and} \qquad \frac{}{(B \mathbin{/} A)\, A \longrightarrow B}\;{/}\textsc{d} \,, \tag{2.3}$$

which we adopt as decomposition rules in place of those in eq. (2.2). The standard $\setminus$L and $/$L rules can still be recovered from these more specific decomposition rules, thanks to CUT (see adjacent figure). These revised, nullary decomposition rules correct the earlier drawbacks: like the other decomposition rules, they now have no premises and only refer to immediate subformulas. Moreover, these rules have the advantage of matching two of the axioms from **Lambek:AMM58**'s original article.[5]

$$\frac{\Omega \vdash A \qquad \Omega'_L\, B\, \Omega'_R \vdash C}{\Omega'_L\, \Omega\, (A \setminus B)\, \Omega'_R \vdash C}\;{\setminus}\textsc{l}$$

$$\leftrightsquigarrow$$

$$\frac{\Omega \vdash A \qquad \dfrac{\dfrac{}{A\, (A \setminus B) \longrightarrow B}\;{\setminus}\textsc{d} \qquad \Omega'_L\, B\, \Omega'_R \vdash C}{\Omega'_L\, A\, (A \setminus B)\, \Omega'_R \vdash C}\;\textsc{cut}^{\longrightarrow}}{\Omega'_L\, \Omega\, (A \setminus B)\, \Omega'_R \vdash C}\;\textsc{cut}^{A}$$

Figure 2.3: Refactoring the $\setminus$L rule in terms of resource decomposition, via $\setminus$D and $\textsc{cut}^{\longrightarrow}$

[5] **Lambek:AMM58**.

FOR MOST ORDERED LOGICAL CONNECTIVES, this approach works perfectly. Unfortunately, the left rules for additive disjunction, $A \oplus B$, and its unit, **0**, are resistant to this kind of refactoring. The difficulty with additive disjunction isn't that its left rule, $\oplus$L, doesn't decompose the resource $A \oplus B$. The $\oplus$L rule certainly does decompose $A \oplus B$, but it does so [...].[6] $A \oplus B \longrightarrow A \mid B$ [...] retain the standard $\oplus$L and $\mathbf{0}$L rules.

$$\frac{\Omega'_L\, A\, \Omega'_R \vdash C \qquad \Omega'_L\, B\, \Omega'_R \vdash C}{\Omega'_L\, (A \oplus B)\, \Omega'_R \vdash C}\;{\oplus}\textsc{l}$$

[6] fix

FIGURE 3.9 PRESENTS the refactored sequent calculus for ordered logic in its entirety. This calculus is sound and complete with respect to the ordered sequent calculus (??).

$$\frac{\Omega \vdash A \quad \Omega'_L\, A\, \Omega'_R \vdash C}{\Omega'_L\, \Omega\, \Omega'_R \vdash C}\ \textsc{cut}^A \qquad \frac{}{A \vdash A}\ \textsc{id}^A$$

$$\frac{\Omega \longrightarrow \Omega' \quad \Omega'_L\, \Omega'\, \Omega'_R \vdash C}{\Omega'_L\, \Omega\, \Omega'_R \vdash C}\ \textsc{cut}^{\longrightarrow}$$

$$\frac{\Omega_1 \vdash A \quad \Omega_2 \vdash B}{\Omega_1\, \Omega_2 \vdash A \bullet B}\ \bullet\textsc{r} \qquad \frac{}{A \bullet B \longrightarrow A\, B}\ \bullet\textsc{d}$$

$$\frac{}{\cdot \vdash \mathbf{1}}\ \mathbf{1}\textsc{r} \qquad \frac{}{\mathbf{1} \longrightarrow \cdot}\ \mathbf{1}\textsc{d}$$

$$\frac{\Omega \vdash A \quad \Omega \vdash B}{\Omega \vdash A \mathbin{\&} B}\ \&\textsc{r} \qquad \frac{}{A \mathbin{\&} B \longrightarrow A}\ \&\textsc{d}_1 \qquad \frac{}{A \mathbin{\&} B \longrightarrow B}\ \&\textsc{d}_2$$

$$\frac{}{\Omega \vdash \top}\ \top\textsc{r} \qquad \text{(no } \top\textsc{d} \text{ rule)}$$

$$\frac{A\, \Omega \vdash B}{\Omega \vdash A \setminus B}\ \setminus\textsc{r} \qquad \frac{}{A\, (A \setminus B) \longrightarrow B}\ \setminus\textsc{d}$$

$$\frac{\Omega\, A \vdash B}{\Omega \vdash B \,/\, A}\ /\textsc{r} \qquad \frac{}{(B \,/\, A)\, A \longrightarrow B}\ /\textsc{d}$$

$$\frac{\Omega \vdash A}{\Omega \vdash A \oplus B}\ \oplus\textsc{r}_1 \qquad \frac{\Omega \vdash B}{\Omega \vdash A \oplus B}\ \oplus\textsc{r}_2 \qquad \frac{\Omega'_L\, A\, \Omega'_R \vdash C \quad \Omega'_L\, B\, \Omega'_R \vdash C}{\Omega'_L\, (A \oplus B)\, \Omega'_R \vdash C}\ \oplus\textsc{l}$$

$$\text{(no } \mathbf{0}\textsc{r} \text{ rule)} \qquad \frac{}{\Omega'_L\, \mathbf{0}\, \Omega'_R \vdash C}\ \mathbf{0}\textsc{l}$$

Figure 2.4: A refactoring of the ordered sequent calculus to emphasize that most left rules amount to resource decomposition

$$\frac{}{A \bullet B \longrightarrow A\,B}\ {\bullet}\textsc{d} \qquad \frac{}{\mathbf{1} \longrightarrow \cdot}\ \mathbf{1}\textsc{d}$$

$$\frac{}{A \,\&\, B \longrightarrow A}\ \&\textsc{d}_1 \qquad \frac{}{A \,\&\, B \longrightarrow B}\ \&\textsc{d}_2 \qquad \text{(no } \top\textsc{d} \text{ rule)}$$

$$\frac{}{A\,(A \setminus B) \longrightarrow B}\ \setminus\textsc{d} \qquad \frac{}{(B \,/\, A)\,A \longrightarrow B}\ /\textsc{d} \qquad \text{(no } \oplus\textsc{d} \text{ and } \mathbf{0}\textsc{d} \text{ rules)}$$

$$\frac{\Omega \longrightarrow \Omega'}{\Omega_L\,\Omega\,\Omega_R \longrightarrow \Omega_L\,\Omega'\,\Omega_R}\ {\longrightarrow}\textsc{c}$$

$$\frac{}{\Omega \Longrightarrow \Omega}\ {\Longrightarrow}\textsc{r} \qquad \frac{\Omega \longrightarrow \Omega' \quad \Omega' \Longrightarrow \Omega''}{\Omega \Longrightarrow \Omega''}\ {\Longrightarrow}\textsc{t}$$

Figure 2.5: The OR rewriting fragment of ordered logic, based on resource decomposition

THEOREM 2.1 (Soundness and completeness). *$\Omega \vdash A$ is derivable in the refactored calculus of fig. 3.9 if, and only if $\Omega \vdash A$ is derivable in the usual ordered sequent calculus (??).*

Proof. Soundness, the right-to-left direction, can be proved by structural induction on the given derivation. The key lemma is the admissibility of $\textsc{cut}^{\longrightarrow}$ in the usual ordered sequent calculus:

If $\Omega \longrightarrow \Omega'$ and $\Omega_L'\,\Omega'\,\Omega_R' \vdash C$, then $\Omega_L'\,\Omega\,\Omega_R' \vdash C$.

This lemma can be proved by case analysis of the decomposition $\Omega \longrightarrow \Omega'$, reconstituting the corresponding left rule along the lines of the sketches from figs. 3.6 and 3.8.

Completeness, the left-to-right direction, can be proved by structural induction on the given derivation. The critical cases are the left rules; they are resolved along the lines of the sketches shown in figs. 3.6 and 3.8. □

2.1.2 *Ordered resource decomposition as rewriting*

Thus far, we have used the decomposition judgment, $\Omega \longrightarrow \Omega'$, and its rules as the basis for a reconfigured sequent-like calculus for ordered logic. Additionally, we can instead view decomposition as the foundation of a rewriting system grounded in ordered logic. For example, the decomposition of resource $A \bullet B$ into $A\,B$ by the $\bullet$D rule can also be seen as *rewriting* $A \bullet B$ into $A\,B$. More generally, the decomposition judgment $\Omega \longrightarrow \Omega'$ can be read as "Ω rewrites to Ω'."

Figure 3.10 summarizes the rewriting system that we obtain from the refactored sequent-like calculus of fig. 3.9; we dub this ordered rewriting system OR. Essentially, OR is obtained by discarding all rules except for the decomposition rules. However, if only the decomposition rules are used, rewritings cannot occur within a larger context. For example, the $\setminus$D rule derives $A\,(A \setminus B) \longrightarrow B$, but $\Omega_L'\,A\,(A \setminus B)\,\Omega_R' \longrightarrow \Omega_L'\,B\,\Omega_R'$ would not be derivable in general. In the refactored calculus of fig. 3.9, this kind of framing is taken care of by the cut principle for decomposition, $\textsc{cut}^{\longrightarrow}$. To express framing at the level of the $\Omega \longrightarrow \Omega'$ judgment itself, we ensure that rewriting is compatible with concatenation of ordered contexts:

$$\frac{\Omega \longrightarrow \Omega'}{\Omega_L\,\Omega\,\Omega_R \longrightarrow \Omega_L\,\Omega'\,\Omega_R}\ {\longrightarrow}\textsc{c}\,.$$

By forming the reflexive, transitive closure of $\longrightarrow$, we may construct a multi-step rewriting relation, which we choose to write as $\Longrightarrow$.[7] Consistent with its monoidal structure, there are two equivalent formulations of this reflexive, transitive closure: each rewriting sequence $\Omega \Longrightarrow \Omega'$ can be viewed as either a list or tree of individual rewriting steps.[8] We prefer the list-based formulation shown in fig. 3.10 because it tends to streamline proofs by structural induction, but, on the basis of the following ??, we allow ourselves to freely switch between the two formulations as needed.

[7] Usually written as $\longrightarrow^*$, we instead chose $\Longrightarrow$ for the reflexive, transitive closure because of its similarity with process calculus notation for weak transitions, $\overset{\alpha}{\Longrightarrow}$. Our reasons will become clearer in subsequent ??.

[8] rewrite with reference to string rewriting

Fact 2.2 (Transitivity of $\Longrightarrow$). *If $\Omega \Longrightarrow \Omega'$ and $\Omega' \Longrightarrow \Omega''$, then $\Omega \Longrightarrow \Omega''$.*

Proof. By induction on the structure of the first trace, $\Omega \Longrightarrow \Omega'$. □

A few remarks about these rewriting relations are in order. First, interpreting the resource decomposition rules as rewriting only confirms our preference for the nullary $\backslash\textsc{d}$ and $/\textsc{d}$ rules (eq. 2.3). The $\backslash\textsc{d}'$ and $/\textsc{d}'$ rules (eq. 2.2), with their $\Omega \vdash A$ premises, would be problematic as rewriting rules because they would introduce a dependence of rewriting upon general provability and the accompanying proof search would take OR too far afield from traditional, syntactic notions of string and multiset rewriting.

Second, multi-step rewriting, $\Longrightarrow$, is incomplete with respect to the usual ordered sequent calculus (??) because all right rules have been discarded.

False claim 2.3 (Completeness). *If $\Omega \vdash A$, then $\Omega \Longrightarrow A$.*

Counterexample. The sequent $A \backslash (C \,/\, B) \vdash (A \backslash C) \,/\, B$ is provable, and yet $A \backslash (C \,/\, B) \not\Longrightarrow (A \backslash C) \,/\, B$ (even though $A\,(A \backslash (C \,/\, B))\,B \Longrightarrow C$ does hold). □

As expected from the way in which it was developed, ordered rewriting in OR is, however, sound. To state and prove soundness, we must first define an operation $\bullet\Omega$ that reifies an ordered context as a single proposition (see adjacent figure).[9]

[9] fix

Lemma 2.4. *For all Ω and C, we have $\Omega \vdash C$ implies $\bullet\Omega \vdash C$, as well as $\Omega \vdash \bullet\Omega$.*

Proof. By induction on the structure of the given context, Ω. □

Theorem 2.5 (Soundness). *If $\Omega \longrightarrow \Omega'$, then $\Omega \vdash \bullet\Omega'$. Also, if $\Omega \Longrightarrow \Omega'$, then $\Omega \vdash \bullet\Omega'$.*

Proof. By induction on the structure of the given step or trace. □

$$\begin{aligned}(\Omega_1\,\Omega_2) &= (\Omega_1) \bullet (\Omega_2)\\ \bullet(\cdot) &= \mathbf{1}\\ A &= A\end{aligned}$$

$$\begin{aligned}\bullet(\Omega_1\,\Omega_2) &= (\bullet\Omega_1) \bullet (\bullet\Omega_2)\\ \bullet(\cdot) &= \mathbf{1}\\ \bullet A &= A\end{aligned}$$

Figure 2.6: From ordered contexts to propositions

Last, notice that every rewriting step, $\Omega \longrightarrow \Omega'$, strictly decreases the number of logical connectives that occur in the ordered context. More formally, let $|\Omega|_\star$ be a measure of the number of logical connectives that occur in Ω, as defined in the adjacent figure. We may then prove the following lemma.

$$
\begin{aligned}
|\Omega_1\,\Omega_2|_\star &= |\Omega_1|_\star + |\Omega_2|_\star \\
|\cdot|_\star &= 0 \\
|A \star B|_\star &= 1 + |A|_\star + |B|_\star \\
&\qquad \text{if } \star = \bullet, \mathbin{\&}, \backslash, /, \text{ or } \oplus \\
|A|_\star &= 1 \ \text{ if } A = a, \mathbf{1}, \top, \text{ or } \mathbf{0}
\end{aligned}
$$

Figure 2.7: A measure of the number of logical connectives within an ordered context

LEMMA 2.6. *If $\Omega \longrightarrow \Omega'$, then $|\Omega|_\star > |\Omega'|_\star$. If $\Omega \Longrightarrow \Omega'$, then $|\Omega|_\star \geq |\Omega'|_\star$.*

Proof. By induction on the structure of the rewriting step. □

On the basis of this lemma, we will frequently refer to the rewriting relation, $\longrightarrow$, as the *reduction relation*. We may use this lemma to prove that ordered rewriting is terminating.

THEOREM 2.7 (Termination). *For all ordered contexts Ω, every rewriting sequence from Ω is finite.*

Proof. Let Ω be an arbitrary ordered context. Beginning from state $\Omega_0 = \Omega$, some state Ω_i will eventually be reached such that either: $\Omega_i \not\longrightarrow$; or $|\Omega_i|_\star = 0$ and $\Omega_i \longrightarrow \Omega_{i+1}$. In the latter case, lemma 2.6 establishes $|\Omega_{i+1}|_\star < 0$, which is impossible because $|{-}|_\star$ is a measure. □

2.1.3 *Recursively defined propositions and unbounded ordered rewriting*

Although a seemingly pleasant property, termination (theorem 2.7) significantly limits the expressiveness of ordered rewriting. For example, without unbounded rewriting, we cannot even use ordered rewriting to describe producer-consumer systems or finite automata.

As the proof of termination shows, rewriting is bounded precisely because states consist of finitely many finite propositions. One way to admit unbounded rewriting is therefore to permit circular propositions in the form of mutually recursive definitions, $\hat{p} \triangleq A$, where the grammar of ordered propositions now includes these recursively defined propositions $\hat{p}$:

$$A, B ::= a \mid A \bullet B \mid \mathbf{1} \mid A \backslash B \mid B \,/\, A \mid A \mathbin{\&} B \mid \top \mid \hat{p}\,.$$

Sequent calculi with definitions of this kind have previously been studied by **Hallnas**:??**Erikkson**:??**Schroeder-Heister**:??**McDowell+Miller**:??**Tiu+Momigliano**:??, among others.

To rule out definitions like $\hat{p} \triangleq \hat{p}$ that do not correspond to sensible infinite propositions, we require that definitions be *contractive*[10] – *i.e.*, that the body of each recursive definition begin with a logical connective (or constant or atom) at the top level.

[10] **Gay+Hole:AI05**.

The recursive definitions are collected into a signature, Φ, which indexes the rewriting relations: $\longrightarrow_\Phi$ and $\Longrightarrow_\Phi$.[11] Syntactically, these signatures are given by

[11] We nearly always elide the index, as it is usually clear from context.

$$\Phi ::= \cdot \mid \Phi, (\hat{p} \triangleq A)\,.$$

By analogy with recursive types from functional programming,[12] we must now decide whether to treat definitions *iso*recursively or *equi*recursively. Under an equirecursive interpretation, definitions $\hat{p} \triangleq A$ may be silently unrolled or rolled at will; in other words, $\hat{p}$ is literally *equal* to its unrolling – $\hat{p} = A$. In contrast, under an isorecursive interpretation, unrolling a recursively defined proposition would count as an explicit step of rewriting – $\hat{p} \neq A$ but $\hat{p} \longrightarrow A$, for example.

[12] ??.

We choose to interpret definitions equirecursively because the equirecursive treatment, with its generous notion of equality, helps to minimize the overhead of recursively defined propositions. As a simple example, under the equirecursive definition $\hat{p} \triangleq a \setminus \hat{p}$, we have the trace

$$a\,a\,\hat{p} = a\,a\,(a \setminus \hat{p}) \longrightarrow a\,\hat{p} = a\,(a \setminus \hat{p}) \longrightarrow \hat{p}$$

or, more concisely, $a\,a\,\hat{p} \longrightarrow a\,\hat{p} \longrightarrow \hat{p}$. Had we chosen an isorecursive treatment of the same definition, we would have only the more laborious

$$a\,a\,\hat{p} \longrightarrow a\,a\,(a \setminus \hat{p}) \longrightarrow a\,\hat{p} \longrightarrow a\,(a \setminus \hat{p}) \longrightarrow \hat{p}\,.$$

This choice differs from the aforementioned works on definitions, which use an isorecursive treatment with explicit right and left rules for recursively defined propositions.

Replication In Milner's development of the π-calculus, there are two avenues to unbounded process behavior: recursive process definitions and replication.

[13]

[13] **Aranda+:FMCO06.**

2.1.4 *Properties of the* OR *ordered rewriting framework*

Concurrency As an example of multi-step rewriting, observe that

$$a\,(a \setminus b)\,(c \mathbin{/} a)\,a \Longrightarrow b\,c.$$

$$\begin{tikzcd}
 & a\,(a \setminus b)\,(c \mathbin{/} a)\,a \arrow[dl] \arrow[dr] \arrow[dd, Rightarrow] & \\
b\,(c \mathbin{/} a)\,a \arrow[dr] & & a\,(a \setminus b)\,c \arrow[dl] \\
 & b\,c &
\end{tikzcd}$$

Figure 2.8: An example of concurrency in ordered rewriting

In fact, as shown in the adjacent figure, two sequences witness this rewriting: either the initial state's left half, $a\,(a \setminus b)$, is first rewritten to b and then its right half, $(c/a)\,a$, is rewritten to c; or *vice versa*, the right half is first rewritten to c and then the left half is rewritten to b.

Notice that these two sequences differ only in how non-overlapping, and therefore independent, rewritings of the initial state's two halves are interleaved. Consequently, the two sequences can be – and indeed should be – considered essentially equivalent. The details of how the small-step rewrites are interleaved are irrelevant, so that conceptually, at least, only the big-step trace from $a\,(a \setminus b)\,(c \mathbin{/} a)\,a$ to $b\,c$ remains.

More generally, this idea that the interleaving of independent actions is irrelevant is known as *concurrent equality*,[14] and it forms the basis of concurrency.[15] Concurrent equality also endows traces $\Omega \Longrightarrow \Omega'$ with a free partially commutative monoid structure, *i.e.*, traces form a trace monoid.

[14] **Watkins+:CMU02.**

[15] ??.

NON-CONFLUENCE As the relation $\Longrightarrow$ forms a rewriting system, we may evaluate it along several standard dimensions: termination, confluence.

Although terminating, ordered rewriting is not confluent. Confluence requires that all states with a common ancestor, *i.e.*, states Ω'_1 and Ω'_2 such that $\Omega'_1 \Longleftarrow\!\!\Longrightarrow \Omega'_2$, be joinable, *i.e.*, $\Omega'_1 \Longrightarrow\!\Longleftarrow \Omega'_2$. Because ordered rewriting is directional[16] and the relation $\Longrightarrow$ is not symmetric, some nondeterministic choices are irreversible.

[16] Is this phrasing correct?

FALSE CLAIM 2.8 (Confluence). *If* $\Omega'_1 \Longleftarrow\!\!\Longrightarrow \Omega'_2$, *then* $\Omega'_1 \Longrightarrow\!\Longleftarrow \Omega'_2$.

Counterexamples. Consider the state $a \,\&\, b$. By the rewriting rules for additive conjunction, $a \longleftarrow a \,\&\, b \longrightarrow b$, and hence $a \Longleftarrow a \,\&\, b \Longrightarrow b$. However, being atoms, neither a nor b reduces. And $a \neq b$, so $a \Longrightarrow\!\Longleftarrow b$ does *not* hold.

Even in the $\&$-free fragment, ordered rewriting is not confluent: for example,

$$\not\longleftarrow c\,(a \setminus b) \Longleftarrow (c \,/\, a)\,a\,(a \setminus b) \Longrightarrow (c \,/\, a)\,b \not\longrightarrow . \qquad \square$$

2.2 *The* FOR *focused ordered rewriting framework*

The above ordered rewriting framework is based upon decomposition rules that are very fine-grained. Each step of rewriting decomposes a proposition into only its immediate subformulas, and no further, such as in the very fine-grained step $a\,\big((a \setminus c \bullet a) \,\&\, (b \setminus \mathbf{1})\big) \longrightarrow a\,(a \setminus c \bullet a)$. It is not possible to rewrite the context $a\,\big((a \setminus c \bullet a) \,\&\, (b \setminus \mathbf{1})\big)$ into $c\,a$ (or even $c \bullet a$) in a single step, although it is possible in several steps: $a\,\big((a \setminus c \bullet a) \,\&\, (b \setminus \mathbf{1})\big) \Longrightarrow c\,a$, because

$$a\,\big((a \setminus c \bullet a) \,\&\, (b \setminus \mathbf{1})\big) \longrightarrow a\,(a \setminus c \bullet a) \longrightarrow c \bullet a \longrightarrow c\,a\,.$$

The decomposition rules are so fine-grained that rewriting may even get stuck in undesirable and unintended ways. For instance, in the previous example, we might have instead nondeterministically committed to rewriting $a\,\big((a \setminus c \bullet a) \,\&\, (b \setminus \mathbf{1})\big)$ into $a\,(b \setminus \mathbf{1})$ as the first step, and then $a\,(b \setminus \mathbf{1})$ is stuck, with no further rewritings possible:

$$a\,\big((a \setminus c \bullet a) \,\&\, (b \setminus \mathbf{1})\big) \longrightarrow a\,(b \setminus \mathbf{1}) \not\longrightarrow .$$

Instead, we would rather have a coarser notion of decomposition so that $a\,\big((a \setminus c \bullet a) \,\&\, (b \setminus \mathbf{1})\big) \longrightarrow c\,a$ is a single step[17] and, conversely, so that $a\,\big((a \setminus c \bullet a) \,\&\, (b \setminus \mathbf{1})\big) \longrightarrow \Omega'$ only if $\Omega' = c\,a$.

[17] Or at least so that $a\,\big((a \setminus c \bullet a) \,\&\, (b \setminus \mathbf{1})\big) \longrightarrow c \bullet a$ is a single step.

FOCUSING, AS DEVELOPED BY **Andreoli:??**, provides just the right coarse-grained decomposition through its complementary inversion and chaining strategies for proof search. An inversion phase groups together successive invertible rules, and a chaining phase groups together successive noninvertible rules that are applied to a single *in-focus* proposition; together, a chaining phase followed by an inversion phase constitutes a *bipole*. Rather than having each of these rules give rise to a separate step, we can treat the entire bipole as an atomic step of rewriting.

This idea of using focusing to increase the granularity of rewriting steps dates back to, at least, the Concurrent Logical Framework (CLF)[18] and was later streamlined by **Cervesato+Scedrov:IC09**. **Simmons:CMU12** has studied a focused ordered rewriting framework, though in a somewhat different formulation than the one we present here.

[18] ????.

THE ORDERED PROPOSITIONS ARE POLARIZED into positive and negative classes, or *polarities*,[19] according to the invertibility of their sequent calculus rules; two 'shift' connectives, $\downarrow$ and $\uparrow$, mediate between the two classes.

[19] ??.

$$\begin{aligned} A^+ &::= a^+ \mid A^+ \bullet B^+ \mid \mathbf{1} \mid {\downarrow}A^- \\ A^- &::= A^+ \setminus B^- \mid B^- \,/\, A^+ \mid A^- \mathbin{\&} B^- \mid \top \mid {\uparrow}A^+ \end{aligned}$$

The positive propositions, A^+, are those propositions that have invertible left rules, such as ordered conjunction; the negative propositions, A^-, are those that have invertible right rules, such as the left- and right-handed implications. For reasons that will become clear in chapter 4, we choose to assign a positive polarity to all atomic propositions, a^+.

Ordered contexts are then formed as the free monoid over negative propositions and positive atoms:

$$\Omega ::= \Omega_1\,\Omega_2 \mid \cdot \mid A^- \mid a^+ \,.$$

As usual, we do not distinguish those ordered contexts that are equivalent up to the monoid laws. We may also reify an ordered context Ω as a positive proposition, $\bullet\Omega$, using the operation defined in the neighboring figure.

$$\begin{aligned} \bullet(\Omega_1\,\Omega_2) &= (\bullet\Omega_1) \bullet (\bullet\Omega_2) \\ \bullet(\cdot) &= \mathbf{1} \\ \bullet A^- &= {\downarrow}A^- \\ \bullet a^+ &= a^+ \end{aligned}$$

Figure 2.9: Reifying an ordered context as a positive proposition

EACH CLASS OF PROPOSITIONS is then equipped with its own focusing judgment: a *left-focus judgment*, $\Omega_L\;[A^-]\;\Omega_R \Vdash C^+$, that focuses on a negative proposition, A^-, that occurs to the left of the turnstile; and a *right-focus judgment*, $[A^+] \dashv\!\vert\; \Omega$, that focuses on a positive proposition, A^+, that occurs to the right of the turnstile.[20]

[20] We choose a left-facing turnstile for the right-focus judgment to emphasize its input/output mode; see the next paragraph.

Following **Zeilberger:??**, each of these judgments can be read as a function that provides a form of extended decomposition – the in-focus proposition is decomposed beyond its immediate subformulas, until subformulas of the opposite polarity are reached. The two focusing judgments are defined inductively on the structure of the in-focus proposition, with the left-focus judgment depending on the right-focus judgment (though not vice versa).

The right-focus judgment, $[A^+] \dashv\!\vert\; \Omega$, decomposes A^+ into the ordered context Ω of its nearest negative subformulas, treating A^+ as input and Ω as output.The judgment is given by the following rules.

$$\frac{[A^+] \dashv\!\dashv \Omega_1 \quad [B^+] \dashv\!\dashv \Omega_2}{[A^+ \bullet B^+] \dashv\!\dashv \Omega_1\, \Omega_2}\ \bullet\text{R} \qquad \frac{}{[\mathbf{1}] \dashv\!\dashv \cdot}\ \mathbf{1}\text{R}$$

$$\frac{}{[a^+] \dashv\!\dashv a^+}\ \text{ID}^{a^+} \qquad \frac{}{[{\downarrow}A^-] \dashv\!\dashv A^-}\ {\downarrow}\text{R}$$

Ordered conjunctions $A^+ \bullet B^+$ are decomposed into $\Omega_1\, \Omega_2$ by inductively decomposing A^+ and B^+ into Ω_1 and Ω_2, respectively, and $\mathbf{1}$ is decomposed into the empty context. Atoms a^+ are not decomposed further[21], and ${\downarrow}A^-$ is decomposed into its immediate subformula of negative polarity, A^-.

[21] Alternatively, following **Simmons:CMU12**, atoms a^+ could be decomposed to suspensions $\langle a^+ \rangle$, but we choose not to do that here.

This right-focus judgment is a left inverse of the $\bullet(-)$ operation:

LEMMA 2.9. *$[\bullet\Omega] \dashv\!\dashv \Omega'$ if, and only if, $\Omega = \Omega'$.*

Proof. Each direction is separately proved by structural induction on the context Ω_2. □

The left-focus judgment, $\Omega_L\, [A^-]\, \Omega_R \Vdash C^+$, decomposes A^- into the ordered contexts Ω_L and Ω_R and positive subformula C^+, treating A^- as input and Ω_L, Ω_R, and C^+ as outputs. The judgment is given by the following rules.

$$\frac{[A^+] \dashv\!\dashv \Omega_A \quad \Omega_L\, [B^-]\, \Omega_R \Vdash C^+}{\Omega_L\, \Omega_A\, [A^+ \setminus B^-]\, \Omega_R \Vdash C^+}\ \setminus\text{L} \qquad \frac{[A^+] \dashv\!\dashv \Omega_A \quad \Omega_L\, [B^-]\, \Omega_R \Vdash C^+}{\Omega_L\, [B^- \,/\, A^+]\, \Omega_A\, \Omega_R \Vdash C^+}\ /\text{L}$$

$$\frac{\Omega_L\, [A^-]\, \Omega_R \Vdash C^+}{\Omega_L\, [A^- \mathbin{\&} B^-]\, \Omega_R \Vdash C^+}\ \&\text{L}_1 \qquad \frac{\Omega_L\, [B^-]\, \Omega_R \Vdash C^+}{\Omega_L\, [A^- \mathbin{\&} B^-]\, \Omega_R \Vdash C^+}\ \&\text{L}_2 \qquad (\text{no } \top\text{L rule})$$

$$\frac{}{[{\uparrow}A^+] \Vdash A^+}\ {\uparrow}\text{L}$$

The left-focus judgment's rules parallel the usual sequent calculus rules, but maintaining focus on the immediate subformulas – left focus for subformulas of negative polarity and right focus for subformulas of positive polarity. The ${\uparrow}$L rule ends left focus by decomposing an ${\uparrow}A^+$ antecedent into an A^+ consequent.

Unlike the right-focus judgment, the left-focus judgment describes a relation (or nondeterministic function), owing to the two rules, $\&\text{L}_1$ and $\&\text{L}_2$, that may apply to alternative conjunctions. For example, both

$$a^+\, [(a^+ \setminus {\uparrow}(c^+ \bullet a^+)) \mathbin{\&} (b^+ \setminus {\uparrow}\mathbf{1})] \Vdash c^+ \bullet a^+$$

and

$$b^+\, [(a^+ \setminus {\uparrow}(c^+ \bullet a^+)) \mathbin{\&} (b^+ \setminus {\uparrow}\mathbf{1})] \Vdash \mathbf{1}$$

are both derivable when the negative proposition $(a^+ \setminus {\uparrow}(c^+ \bullet a^+)) \mathbin{\&} (a^+ \setminus {\uparrow}\mathbf{1})$ is in focus.

A FOCUSED REWRITING STEP ARISES when a negative proposition, A^-, is put into focus and the resulting consequent, C^+, is subsequently decomposed into the ordered context.[22][23] In addition, the compatibility rule $\longrightarrow$C is retained.

[22] name for this rule?

[23] Writing the right-focus judgment as $[B^+] \dashv\!\dashv \Omega'$ gives this rule the flavor of a cut principle.

$$\frac{\Omega_L\ [A^-]\ \Omega_R \Vdash C^+ \quad [C^+] \dashV\!\!\!\,\vdash \Omega'}{\Omega_L\ A^-\ \Omega_R \longrightarrow \Omega'} \longrightarrow\text{I} \qquad \frac{\Omega \longrightarrow \Omega'}{\Omega_L\ \Omega\ \Omega_R \longrightarrow \Omega_L\ \Omega'\ \Omega_R} \longrightarrow\text{C}$$

With this $\longrightarrow$I rule, it is indeed possible to rewrite[24]

[24] fix

$$a^+\ \big((a^+ \setminus {\uparrow}(c^+ \bullet a^+))\ \&\ (b^+ \setminus {\uparrow}\mathbf{1})\big) \longrightarrow c^+\ a^+$$

in a single, atomic step because $a^+\ [(a^+ \setminus {\uparrow}(c^+ \bullet a^+))\ \&\ (b^+ \setminus {\uparrow}\mathbf{1})] \Vdash c^+ \bullet a^+$ and $[c^+ \bullet a^+] \dashV\!\!\!\,\vdash c^+\ a^+$ hold. Moreover, the larger granularity afforded by the left- and right-focus judgments precludes the small steps that led to unintended stuck states. For example:

$$a^+\ \big((a^+ \setminus {\uparrow}(c^+ \bullet a^+))\ \&\ (b^+ \setminus {\uparrow}\mathbf{1})\big) \longrightarrow \Omega' \ \text{ only if } \ \Omega' = c^+\ a^+\,.$$

2.2.1 *Recursively defined propositions and focused ordered rewriting*

With the revisions to the granularity of rewriting steps that the focused rewriting framework brings, we should pause to consider how recursively defined propositions interact with focused rewriting.

Previously, in the unfocused rewriting framework, recursively defined propostions such as $\hat{p} \triangleq (a \setminus \hat{p} \bullet a)\ \&\ (b \setminus \mathbf{1})$ were permitted. With the fine granularity of rewriting imposed in that framework, it took three steps to rewrite $a\,\hat{p}$ into $\hat{p}\,a$:

$$a\,\hat{p} = a\,\big((a \setminus \hat{p} \bullet a)\ \&\ (b \setminus \mathbf{1})\big) \longrightarrow a\,(a \setminus \hat{p} \bullet a) \longrightarrow \hat{p} \bullet a \longrightarrow \hat{p}\,a\,.$$

In the polarized, focused rewriting framework, the analogous definition with only the minimally necessary shifts is $\hat{p}^- \triangleq (a^+ \setminus {\uparrow}({\downarrow}\hat{p}^- \bullet a^+))\ \&\ (b^+ \setminus {\uparrow}\mathbf{1})$. With the coarser granularity of rewriting now afforded by focusing, it takes only one focused step to rewrite $a^+\,\hat{p}^-$ into $\hat{p}^-\,a^+$:

$$a^+\,\hat{p}^- = a^+\ \big((a^+ \setminus {\uparrow}({\downarrow}\hat{p}^- \bullet a^+))\ \&\ (b^+ \setminus {\uparrow}\mathbf{1})\big) \longrightarrow \hat{p}^-\ a^+$$

because

$$a^+\ [\hat{p}^-] \Vdash {\downarrow}\hat{p}^- \bullet a^+ \qquad \text{and} \qquad [{\downarrow}\hat{p}^- \bullet a^+] \dashV\!\!\!\,\vdash \hat{p}^-\ a^+\,.$$

Because the left-focus judgment is defined inductively, not coinductively, there are some recursively defined negative propositions that cannot successfully be put into focus. Under the definition $\hat{p}^- \triangleq a^+ \setminus \hat{p}^-$, for example, there are no contexts Ω_L and Ω_R and positive consequent C^+ for which $\Omega_L\ [\hat{p}^-]\ \Omega_R \Vdash C^+$ is derivable. To derive a left-focus judgment on $\hat{p}^-$, the finite context Ω_L would need to hold an infinite stream of a^+ atoms, which is impossible in an inductively defined context.

However, by inserting a double shift, ${\uparrow}{\downarrow}$, which allows focus to be blurred at the ${\uparrow}$, the definition can be revised to one that admits left-focusing: when $\hat{p}^-$ is defined by $\hat{p}^- \triangleq a^+ \setminus {\uparrow}{\downarrow}\hat{p}^-$, the judgment $a^+\ [\hat{p}^-] \Vdash {\downarrow}\hat{p}^-$ is derivable. It follows that $a^+\ \hat{p}^- \longrightarrow \hat{p}^-$.

More generally, any recursively defined proposition that does not pass through an ${\uparrow}$ shift along a main branch cannot be successfully put into focus.

Figure 2.10: The FOR framework for focused ordered rewriting

POSITIVE PROPS.	$A^+ ::= A^+ \bullet B^+ \mid \mathbf{1} \mid a^+ \mid {\downarrow}A^-$
NEGATIVE PROPS.	$A^- ::= A^+ \setminus B^- \mid B^- / A^+ \mid A^- \mathbin{\&} B^- \mid \top \mid \hat{p}^- \mid {\uparrow}A^+$
CONTEXTS	$\Omega ::= \Omega_1\, \Omega_2 \mid \cdot \mid A^- \mid a^+$
SIGNATURES	$\Phi ::= \cdot \mid \Phi, \hat{p}^- \triangleq A^-$

$$\frac{[A^+] \dashV \Omega_1 \qquad [B^+] \dashV \Omega_2}{[A^+ \bullet B^+] \dashV \Omega_1\, \Omega_2}\ \bullet\textsc{r} \qquad\qquad \frac{}{[\mathbf{1}] \dashV \cdot}\ \mathbf{1}\textsc{r}$$

$$\frac{}{[a^+] \dashV a^+}\ \textsc{id}^{a^+} \qquad\qquad \frac{}{[{\downarrow}A^-] \dashV A^-}\ {\downarrow}\textsc{r}$$

$$\frac{[A^+] \dashV \Omega_A \qquad \Omega_L\, [B^-]\, \Omega_R \Vdash C^+}{\Omega_L\, \Omega_A\, [A^+ \setminus B^-]\, \Omega_R \Vdash C^+}\ \setminus\textsc{l} \qquad\qquad \frac{[A^+] \dashV \Omega_A \qquad \Omega_L\, [B^-]\, \Omega_R \Vdash C^+}{\Omega_L\, [B^- / A^+]\, \Omega_A\, \Omega_R \Vdash C^+}\ /\textsc{l}$$

$$\frac{\Omega_L\, [A^-]\, \Omega_R \Vdash C^+}{\Omega_L\, [A^- \mathbin{\&} B^-]\, \Omega_R \Vdash C^+}\ \&\textsc{l}_1 \qquad\qquad \frac{\Omega_L\, [B^-]\, \Omega_R \Vdash C^+}{\Omega_L\, [A^- \mathbin{\&} B^-]\, \Omega_R \Vdash C^+}\ \&\textsc{l}_2 \qquad\qquad \text{(no } \top\textsc{l} \text{ rule)}$$

$$\frac{}{[{\uparrow}A^+] \Vdash A^+}\ {\uparrow}\textsc{l}$$

$$\frac{\Omega_L\, [A^-]\, \Omega_R \Vdash C^+ \qquad [C^+] \dashV \Omega'}{\Omega_L\, A^-\, \Omega_R \longrightarrow \Omega'}\ {\longrightarrow}\textsc{i} \qquad\qquad \frac{\Omega \longrightarrow \Omega'}{\Omega_L\, \Omega\, \Omega_R \longrightarrow \Omega_L\, \Omega'\, \Omega_R}\ {\longrightarrow}\textsc{c}$$

$$\frac{}{\Omega \Longrightarrow \Omega}\ {\Longrightarrow}\textsc{r} \qquad\qquad \frac{\Omega \longrightarrow \Omega' \qquad \Omega' \Longrightarrow \Omega''}{\Omega \Longrightarrow \Omega''}\ {\Longrightarrow}\textsc{t}$$

2.3 *Using shifts to control focusing*

With careful placement of shifts, it is possible to control the behavior of focused ordered rewriting in FOR. It is even possible to embed unfocused ordered rewriting and weakly focused ordered rewriting within FOR in an operationally faithful way, as we show in ????. But first, we discuss a minimal polarization strategy for propositions.

2.3.1 *A minimal polarization strategy*

Because the unpolarized and polarized propositions share the same logical connectives and constants, there is an obvious polarization strategy: Given an unpolarized proposition, insert an $\uparrow$ in front of each positive proposition that occurs where a negative subformula is required; symmetrically, insert a $\downarrow$ in front of each negative proposition that occurs where a positive subformula is required. For example, the unpolarized proposition $a \bullet \big((a \setminus c \bullet a) \mathbin{\&} (b \setminus \mathbf{1})\big)$ becomes $a^+ \bullet {\downarrow}\big((a^+ \setminus {\uparrow}(c^+ \bullet a^+)) \mathbin{\&} (b^+ \setminus {\uparrow}\mathbf{1})\big)$ under the minimal polarization strategy.

In other words, the minimal polarization is one that adds $\uparrow$ and $\downarrow$ shifts only as required. We will frequently elide these shifts because they can be easily inferred.

2.3.2 *Embedding unfocused ordered rewriting*

With careful placement of additional, non-minimal shifts, it is possible to embed unfocused ordered rewriting within the focused ordered rewriting framework in a operationally faithful way. Specifically, we can define a mapping, $(-)^{\Box}$, from contexts of unpolarized propositions to contexts of negative propositions and positive atoms in a way that strongly respects the operational behavior of unfocused ordered rewriting:

- $\Omega \longrightarrow \Omega'$ implies $\Omega^{\Box} \longrightarrow \Omega'^{\Box}$; and
- $\Omega^{\Box} \longrightarrow \Delta'$ implies $\Omega \longrightarrow \Omega'$, for some Ω' such that $\Delta' = \Omega'^{\Box}$.

That is, $(-)^{\Box}$ will be a *strong reduction bisimulation.*[25]

[25] **Sangiorgi+Walker:CUP03**.

Essentially, this embedding inserts a double shift, $\downarrow\uparrow$, in front of each proper, nonatomic subformula. These double shifts cause chaining and inversion to be interrupted after each step, forcing the focused rewriting to mimic the small-step behavior of unfocused rewriting.

The mapping $(-)^{\Box}$ relies on two auxiliary mappings: $(-)^{\boxminus}$ and $(-)^{\boxplus}$, from unpolarized propositions to negative and positive propositions, respectively. $A^{\boxminus}$ and $A^{\boxplus}$ produce negative and positive polarizations of A that insert a $\downarrow\uparrow$ shift in front of every proper, nonatomic subformula of A. In addition, $A^{\boxminus}$ prepends an $\uparrow$ shift whenever the top-level connective of A has positive polarity, whereas $A^{\boxplus}$ prepends a $\downarrow$ shift whenever A is not atomic.

$$
\begin{aligned}
(A \bullet B)^{\boxminus} &= {\uparrow}(A^{\boxplus} \bullet B^{\boxplus}) \\
\mathbf{1}^{\boxminus} &= {\uparrow}\mathbf{1} \\
(A \setminus B)^{\boxminus} &= A^{\boxplus} \setminus {\uparrow}B^{\boxplus} \\
(B \,/\, A)^{\boxminus} &= {\uparrow}B^{\boxplus} \,/\, A^{\boxplus} \\
(A \mathbin{\&} B)^{\boxminus} &= {\uparrow}A^{\boxplus} \mathbin{\&} {\uparrow}B^{\boxplus} \\
\top^{\boxminus} &= \top \\
A^{\boxplus} &= \begin{cases} a^+ & \text{if } A = a \\ {\downarrow}A^{\boxminus} & \text{otherwise} \end{cases} \\
(\Omega_1\, \Omega_2)^{\Box} &= \Omega_1^{\Box}\, \Omega_2^{\Box} \\
(\cdot)^{\Box} &= \cdot \\
A^{\Box} &= \begin{cases} a^+ & \text{if } A = a \\ A^{\boxminus} & \text{otherwise} \end{cases}
\end{aligned}
$$

Figure 2.11: An embedding of unfocused ordered rewriting (*i.e.*, OR) within FOR

THEOREM 2.10. *The embedding $(-)^{\Box}$ satisfies the following properties.*

- *If* $\Omega \longrightarrow \Omega'$, *then* $\Omega^{\square} \longrightarrow \Omega'^{\square}$.
- *If* $\Omega^{\square} = \Delta \longrightarrow \Delta'$, *then* $\Omega \longrightarrow \Omega'$ *for some* Ω' *such that* $\Delta' = \Omega'^{\square}$.

Proof. The proofs of these properties require a straightforward lemma: for all unpolarized propositions A,

$$[A^{\boxplus}] \dashV \Delta \text{ if, and only if, } \Delta = A^{\square} .$$

The first property is then proved by induction over the structure of the given rewriting step, $\Omega \longrightarrow \Omega'$. As an example, consider the case in which $\Omega = A\,(A \setminus B) \longrightarrow B = \Omega'$. By definition, $\Omega^{\square} = A^{\square}\,(A^{\boxplus} \setminus \uparrow B^{\boxplus})$ and $\Omega'^{\square} = B^{\square}$, and we can indeed derive $A^{\square}\,[A^{\boxplus} \setminus \uparrow B^{\boxplus}] \Vdash B^{\boxplus}$ and $[B^{\boxplus}] \dashV B^{\square}$. So, as required, $\Omega^{\square} = A^{\square}\,(A^{\boxplus} \setminus \uparrow B^{\boxplus}) \longrightarrow B^{\square} = \Omega'^{\square}$.

The second property is also proved by induction over the structure of the given rewriting step, this time $\Omega^{\square} = \Delta \longrightarrow \Delta'$. As an example, consider the case in which $\Omega_L^{\square}\,[A^{\boxplus} \setminus \uparrow B^{\boxplus}]\,\Omega_R^{\square} \Vdash C^+$ and $[C^+] \dashV \Delta'$, for some Ω_L, A, B, Ω_R, and C^+ such that $\Omega = \Omega_L\,(A \setminus B)\,\Omega_R$. By inversion and the aforementioned lemma, we have $\Omega_L = A$, $\Omega_R = \cdot$, $C^+ = B^{\boxplus}$, and $\Delta' = B^{\square}$. Indeed, as required, $\Omega = A\,(A \setminus B) \longrightarrow B = \Omega'$ and $\Delta' = \Omega'^{\square}$. □

2.3.3 *Embedding weakly focused ordered rewriting*

It is similarly possible to embed weakly focused ordered rewriting, a rewriting discipline based on weak focusing[26] in which the granularity of steps lies between that of the unfocused and fully focused ordered rewriting frameworks. More specifically, weak focusing differs from full focusing in that it retains chaining but abandons eager inversion. For example, with weakly focused rewriting,

$$a^+ \downarrow\!\big((a^+ \setminus \uparrow(c^+ \bullet a^+)) \mathbin{\&} (b^+ \setminus \uparrow\mathbf{1})\big) \longrightarrow c^+ \bullet a^+ \longrightarrow c^+\,a^+ ,$$

where the inversion of $c^+ \bullet a^+$ is now an atomic step of its own.

[26] ??.

This weakly focused rewriting discipline could be achieved as an independent system with the rules shown in **??**. Notice that weakly focused rewriting restricts the left- and right-handed implications to have only atomic premises. Although weak focusing is well-defined for arbitrary implications,[27] it is not clear how to give a rewriting interpretation of weak focusing unless this restriction is made.

[27] ??.

In fact, there is a better approach than using weakly focused ordered rewriting as yet another independent rewriting system. Instead of using weakly focused rewriting directly, we can embed it within FOR by inserting shifts at specific locations and then use the embedding. From here on, we will exclusively use this embedding when weakly focused ordered rewriting is needed.

$$\begin{aligned}
(A^+)^{\boxplus} &= \begin{cases} a^+ & \text{if } A^+ = a^+ \\ \downarrow(A^+)^{\square} & \text{otherwise} \end{cases} \\
(a^+ \setminus B^-)^{\boxminus} &= a^+ \setminus (B^-)^{\boxminus} \\
(B^- \mathbin{/} a^+)^{\boxminus} &= (B^-)^{\boxminus} \mathbin{/} a^+ \\
(A^- \mathbin{\&} B^-)^{\boxminus} &= (A^-)^{\boxminus} \mathbin{\&} (B^-)^{\boxminus} \\
\top^{\boxminus} &= \top \\
(\uparrow A^+)^{\boxminus} &= \uparrow(A^+)^{\boxplus} \\
(\Omega_1^+\,\Omega_2^+)^{\square} &= (\Omega_1^+)^{\square}\,(\Omega_2^+)^{\square} \\
(\cdot)^{\square} &= \cdot \\
(a^+)^{\square} &= a^+ \\
(A^+ \bullet B^+)^{\square} &= \uparrow((A^+)^{\boxplus} \bullet (B^+)^{\boxplus}) \\
\mathbf{1}^{\square} &= \uparrow\mathbf{1} \\
(\downarrow A^-)^{\square} &= (A^-)^{\boxminus}
\end{aligned}$$

Figure 2.13: An embedding of weakly focused ordered rewriting (*i.e.*, OR) within FOR

THEOREM 2.11. *The embedding* $(-)^{\square}$ *satisfies the following properties.*

- *If* $\Omega^+ \longrightarrow \Omega'^+$, *then* $(\Omega^+)^{\square} \longrightarrow (\Omega'^+)^{\square}$.
- *If* $(\Omega^+)^{\square} \longrightarrow \Delta'$, *then* $\Omega^+ \longrightarrow \Omega'^+$ *for some* Ω'^+ *such that* $\Delta' = (\Omega'^+)^{\square}$.

$$\dfrac{\Omega_L^+ \, [A^-] \, \Omega_R^+ \Vdash C^+}{\Omega_L^+ \downarrow A^- \, \Omega_R^+ \longrightarrow C^+}\ {\downarrow}\textsc{d} \qquad \dfrac{}{A^+ \bullet B^+ \longrightarrow A^+ \, B^+}\ {\bullet}\textsc{d} \qquad \dfrac{}{\mathbf{1} \longrightarrow \cdot}\ \mathbf{1}\textsc{d}$$

$$\dfrac{\Omega^+ \longrightarrow \Omega'^+}{\Omega_L^+ \, \Omega^+ \, \Omega_R^+ \longrightarrow \Omega_L^+ \, \Omega'^+ \, \Omega_R^+}\ {\longrightarrow}\textsc{c}$$

$$\dfrac{\Omega_L^+ \, [B^-] \, \Omega_R^+ \Vdash C^+}{\Omega_L^+ \, a^+ \, [a^+ \setminus B^-] \, \Omega_R^+ \Vdash C^+}\ {\setminus}\textsc{l} \qquad \dfrac{\Omega_L^+ \, [B^-] \, \Omega_R^+ \Vdash C^+}{\Omega_L^+ \, [B^- \,/\, a^+] \, a^+ \, \Omega_R^+ \Vdash C^+}\ /\textsc{l}$$

$$\dfrac{\Omega_L^+ \, [A^-] \, \Omega_R^+ \Vdash C^+}{\Omega_L^+ \, [A^- \mathbin{\&} B^-] \, \Omega_R^+ \Vdash C^+}\ \&\textsc{l}_1 \qquad \dfrac{\Omega_L^+ \, [B^-] \, \Omega_R^+ \Vdash C^+}{\Omega_L^+ \, [A^- \mathbin{\&} B^-] \, \Omega_R^+ \Vdash C^+}\ \&\textsc{l}_2 \qquad \text{(no } \top\textsc{l} \text{ rule)}$$

$$\dfrac{}{[\uparrow A^+] \Vdash A^+}\ {\uparrow}\textsc{l}$$

Figure 2.12: A framework for *weakly* focused ordered rewriting

Proof. The proofs of these properties require two relatively straightforward lemmas: for all polarized propositions A^+ and A^-,

- $[(A^+)^{\boxplus}] \dashV \Delta$ if, and only if, $\Delta = (A^+)^{\square}$; and
- $\Delta_L \, [(A^-)^{\boxminus}] \, \Delta_R \Vdash B^+$ if, and only if, $\Omega_L^+ \, [A^-] \, \Omega_R^+ \Vdash C^+$ and $\Delta_L = (\Omega_L^+)^{\square}$, $\Delta_R = (\Omega_R^+)^{\square}$, and $B^+ = (C^+)^{\boxplus}$.

Both lemmas are proved by structural induction on the polarized proposition, A^+ and A^-, respectively.

The first of the above properties is then proved by induction over the structure of the given weakly focused rewriting step, $\Omega^+ \longrightarrow \Omega'^+$. As an example, consider the case in which $\Omega_L^+ \downarrow A^- \, \Omega_R^+ \longrightarrow C^+$ because $\Omega_L^+ \, [A^-] \, \Omega_R^+ \Vdash C^+$. By the above lemma, $(\Omega_L^+)^{\square} \, [(A^-)^{\boxminus}] \, (\Omega_R^+)^{\square} \Vdash (C^+)^{\boxplus}$ holds in the fully focused calculus. And so, as required, $(\Omega_L^+)^{\square} \, ({\downarrow} A^-)^{\square} \, (\Omega_R^+)^{\square} \longrightarrow (C^+)^{\square}$.

The second property is also proved by induction over the structure of the given rewriting step, this time the fully focused $(\Omega^+)^{\square} \longrightarrow \Delta'$. As an example, consider the case in which $\Delta_L \, [a_1^+ \setminus (A_2^-)^{\boxminus}] \, \Delta_R \Vdash B^+$ and $[B^+] \dashV \Delta'$. Inversion yields $\Delta'_L \, [(A_2^-)^{\boxminus}] \, \Delta_R \Vdash B^+$ for some Δ'_L such that $\Delta_L = \Delta'_L \, a_1^+$. Then, by the above lemma, $\Omega_L^+ \, [A_2^-] \, \Omega_R^+ \Vdash C^+$ holds in the weakly focused calculus, with $\Delta'_L = (\Omega_L^+)^{\square}$, $\Delta_R = (\Omega_R^+)^{\square}$, and $B^+ = (C^+)^{\boxplus}$. It follows that $\Omega_L^+ \, a_1^+ \, [a_1^+ \setminus A_2^-] \, \Omega_R^+ \Vdash C^+$, and so $\Omega_L^+ \, a_1^+ \downarrow (a_1^+ \setminus A_2^-) \, \Omega_R^+ \longrightarrow C^+$. Also notice that $\Delta_L = (\Omega_L^+ \, a_1^+)^{\square}$ and $\Delta' = (C^+)^{\square}$, as required. □

3
MOVE THESE

3.1 Choreographies

Recall the string rewriting specification

$$\frac{}{a\,b \longrightarrow b} \qquad \text{and} \qquad \frac{}{b \longrightarrow \epsilon}\,.$$

A choreography is a refinement of this specification in which each symbol a of the rewriting alphabet is mapped to an ordered proposition: either an atomic proposition, $\underleftarrow{a}$ or $\underrightarrow{a}$, or a recursively defined proposition, $\hat{a}$. In other words, a choreography is an injection from symbols to propositions.

$$\begin{tikzcd}
w \arrow[r] \arrow[d, dash] & w' \arrow[d, dash, dashed] \\
\theta(w) \arrow[r, Rightarrow, dashed] & \theta(w')
\end{tikzcd}
\qquad
\begin{tikzcd}
\theta(w) \arrow[r] \arrow[d, dash] & \Omega' \arrow[r, Rightarrow, dashed] & \theta(w') \arrow[d, dash, dashed] \\
w \arrow[rr, dashed] & & w'
\end{tikzcd}$$

$\underrightarrow{a}\,\hat{b}$

Suppose that θ is the mapping $a \mapsto \underrightarrow{a}$ and $b \mapsto \hat{b}$. and the choreography

$$\hat{b} \triangleq (\underrightarrow{a} \setminus \hat{b}) \mathbin{\&} \mathbf{1}\,.$$

Notice that

$$a\,b \longrightarrow b \qquad\qquad \text{and} \quad b \longrightarrow \epsilon$$

as well as

$$\underrightarrow{a}\,\hat{b} \longrightarrow \underrightarrow{a}\,(\underrightarrow{a} \setminus \hat{b}) \longrightarrow \hat{b} \quad \text{and} \quad \hat{b} \longrightarrow \mathbf{1} \longrightarrow \cdot\,.$$

$$\frac{}{a\,b \longrightarrow b} \qquad \text{and} \qquad \frac{}{c\,b \longrightarrow b}$$

$$\hat{b} \triangleq (\underrightarrow{a} \setminus \hat{b}) \mathbin{\&} (\underrightarrow{c} \setminus \hat{b})$$

$$a\,b \longrightarrow w' \text{ implies } w' = b \quad \text{but} \quad \underrightarrow{a}\,\hat{b} \longrightarrow \underrightarrow{a}\,(\underrightarrow{c} \setminus \hat{b}) \not\longrightarrow$$

[1]

[1] McDowell+:TCS03.

Judgments $\theta \vdash \Sigma \rightsquigarrow \Sigma'$ and $\theta \vdash w \longrightarrow w'\,[\hat{a}]\,A \rightsquigarrow$. In both judgments, all terms before the $\rightsquigarrow$ are inputs; all terms after the $\rightsquigarrow$ are outputs.

$$\frac{}{\theta \vdash \cdot \rightsquigarrow \cdot} \qquad \frac{\theta \vdash \Sigma \rightsquigarrow \Sigma' \quad \theta \vdash w \longrightarrow w'\,[\hat{a}]\,A_2 \rightsquigarrow \quad (\Sigma'(\hat{a}) = A_1)}{\theta \vdash \Sigma, w \longrightarrow w' \rightsquigarrow \Sigma', \hat{a} \triangleq A_1 \mathbin{\&} A_2}$$

$$\frac{\theta \vdash \Sigma \rightsquigarrow \Sigma' \quad \theta \vdash w \longrightarrow w'\,[\hat{a}]\,A \rightsquigarrow \quad (\hat{a} \notin \operatorname{dom} \Sigma')}{\theta \vdash \Sigma, w \longrightarrow w' \rightsquigarrow \Sigma', \hat{a} \triangleq A}$$

$$\frac{(\theta(a) = \hat{a}) \quad (\theta(w') = \Omega')}{\theta \vdash a \longrightarrow w'\,[\hat{a}]\,{\uparrow}(\bullet\Omega') \rightsquigarrow}$$

$$\frac{\theta \vdash w \longrightarrow w'\,[\hat{a}]\,A \rightsquigarrow \quad (\theta(b) = \underrightarrow{b})}{\theta \vdash b\,w \longrightarrow w'\,[\hat{a}]\,\underrightarrow{b} \setminus A \rightsquigarrow} \qquad \frac{\theta \vdash w \longrightarrow w'\,[\hat{a}]\,A \rightsquigarrow \quad (\theta(b) = \underleftarrow{b})}{\theta \vdash w\,b \longrightarrow w'\,[\hat{a}]\,A \mathbin{/} \underleftarrow{b} \rightsquigarrow}$$

THEOREM 3.1. • *If $\theta \vdash \Sigma \rightsquigarrow \Sigma'$ and $w \longrightarrow_{\Sigma} w'$, then $\theta(w) \longrightarrow_{\Sigma'} \theta(w')$. If $\theta \vdash \Sigma \rightsquigarrow \Sigma'$ and $\Omega \longrightarrow_{\Sigma'} \Omega'$, then $\theta^{-1}(\Omega) \longrightarrow_{\Sigma} \theta^{-1}(\Omega')$.*

- *If $\theta \vdash w \longrightarrow w'\,[\hat{a}]\,A \rightsquigarrow$, then $\theta(w) \longrightarrow_{\hat{a} \triangleq A} \theta(w')$. If $\theta \vdash w \longrightarrow w'\,[\hat{a}]\,A \rightsquigarrow$ and $\Omega \longrightarrow_{\hat{a} \triangleq A} \Omega'$, then $\theta^{-1}(\Omega) \longrightarrow \theta^{-1}(\Omega')$.*

Proof. $\hat{a} \longrightarrow \bullet\theta(w')$

$\underrightarrow{b}\,\theta(w) \longrightarrow_{\hat{a} \triangleq \underrightarrow{b} \setminus A} \theta(w')$ if $\theta(w) \longrightarrow_{\hat{a} \triangleq A} \theta(w')$

$\theta(w)\,\underleftarrow{b} \longrightarrow_{\hat{a} \triangleq A / \underleftarrow{b}} \theta(w')$ if $\theta(w) \longrightarrow_{\hat{a} \triangleq A} \theta(w')$

□

When $\theta = \{(a, \underrightarrow{a}), (b, \hat{b})\}$, the judgment $\theta \vdash \Sigma \rightsquigarrow \hat{b} \triangleq (\underrightarrow{a} \setminus \hat{b}) \mathbin{\&} \mathbf{1}$ holds. However, $b \longrightarrow \epsilon$ but $\hat{b} \not\longrightarrow \cdot$.

$$\begin{aligned} \hat{e} &\triangleq {\uparrow}({\downarrow}\hat{e} \bullet {\downarrow}\hat{b}_1) \mathbin{/} \underleftarrow{i} \\ \hat{b}_0 &\triangleq {\uparrow}{\downarrow}\hat{b}_1 \mathbin{/} \underleftarrow{i} \\ \hat{b}_1 &\triangleq {\uparrow}(\underleftarrow{i} \bullet {\downarrow}\hat{b}_0) \mathbin{/} \underleftarrow{i} \end{aligned}$$

$$\hat{e} \triangleq \big({\uparrow}({\downarrow}\hat{e} \bullet {\downarrow}\hat{b}_1) \mathbin{/} \underleftarrow{i}\big) \mathbin{\&} ({\uparrow}\underrightarrow{z} \mathbin{/} \underleftarrow{d})$$

$$\hat{\imath} \triangleq (\underrightarrow{e} \setminus {\uparrow}(\underrightarrow{e} \bullet \underrightarrow{b}_1)) \mathbin{\&} (\underrightarrow{b}_0 \setminus {\uparrow}\underrightarrow{b}_1) \mathbin{\&} (\underrightarrow{b}_1 \setminus {\uparrow}({\downarrow}\hat{\imath} \bullet \underrightarrow{b}_0))$$

3.1.1

3.2

Atomic ordered propositions are viewed as messages; compound ordered propositions, as processes; and ordered contexts, as configurations of processes.

The ordered contexts form a monoid over the positive propositions and are given by

$$\Omega ::= \Omega_1\,\Omega_2 \mid \cdot \mid A^+ .$$

In keeping with the monoid laws, we treat $(\Omega_1\,\Omega_2)\,\Omega_3$ and $\Omega_1\,(\Omega_2\,\Omega_3)$ as syntactically indistinguishable, as we also do for $\Omega\,(\cdot)$ and Ω and $(\cdot)\,\Omega$.

Each atom is consistently assigned a direction, either left-directed, $\underset{\leftarrow}{a}$, or right-directed, $\underset{\rightarrow}{a}$.

An atom's direction and position within the larger context together indicate whether, when viewed as a message, it is being sent or received. In the context $\Omega_1\,\underset{\rightarrow}{a}\,\Omega_2$, the atom $\underset{\rightarrow}{a}$ is a message being sent from Ω_1 to Ω_2. Symmetrically, in the context $\Omega_1\,\underset{\leftarrow}{a}\,\Omega_2$, the atom $\underset{\leftarrow}{a}$ is a message being sent from Ω_1 to Ω_2.

The context $\Omega = \Omega'\,\underset{\rightarrow}{a}$ is a process configuration that sends $\underset{\rightarrow}{a}$ to its right and continues as Ω'. Conversely, $\underset{\rightarrow}{a}\,\Omega$ is a process configuration in which Ω is the intended recipient of the message $\underset{\rightarrow}{a}$.

$$
\begin{aligned}
A^+ &::= \underset{\leftarrow}{a} \mid \underset{\rightarrow}{a} \mid \hat{p}^+ \mid A^+ \bullet B^+ \mid \mathbf{1} \mid {\downarrow}A^- \\
A^- &::= \hat{p}^- \mid \underset{\rightarrow}{a} \setminus B^- \mid B^- \mathbin{/} \underset{\leftarrow}{a} \mid A^- \mathbin{\&} B^- \mid \top \mid {\uparrow}A^+
\end{aligned}
$$

3.3 *Choreographing specifications*

$$
\frac{}{a\,b \longrightarrow b} \qquad \text{and} \qquad \frac{}{b \longrightarrow \cdot}
$$

As a specification, these string rewriting axioms are quite reasonable. However, as a [...], [...].

Toward our ultimate goal of relating the proof-construction and proof-reduction appraches to concurrency, we would like a description of this concurrent system that is slightly more concrete.

$$
\begin{aligned}
\underset{\rightarrow}{a}\,\hat{b} &\longrightarrow \hat{b} \\
\hat{b} &\longrightarrow \cdot
\end{aligned}
$$

$$
\hat{b} \triangleq (\underset{\rightarrow}{a} \setminus {\uparrow}{\downarrow}\hat{b}) \mathbin{\&} \mathbf{1}
$$

$$
\frac{}{e\,i \longrightarrow e\,b_1} \qquad \frac{}{b_0\,i \longrightarrow b_1} \qquad \text{and} \qquad \frac{}{b_1\,i \longrightarrow i\,b_0}
$$

$$
\begin{aligned}
\hat{e} &\triangleq \hat{e} \bullet \hat{b}_1 \mathbin{/} \underset{\leftarrow}{i} \\
\hat{b}_0 &\triangleq \hat{b}_1 \mathbin{/} \underset{\leftarrow}{i} \\
\hat{b}_1 &\triangleq \underset{\leftarrow}{i} \bullet \hat{b}_0 \mathbin{/} \underset{\leftarrow}{i}
\end{aligned}
$$

$$
\begin{aligned}
&e \mathcal{R} \hat{e}\\
&b_0 \mathcal{R} \hat{b}_0\\
&b_1 \mathcal{R} \hat{b}_1\\
&i \mathcal{R} \underleftarrow{i}\\
&e\, b_1 \mathcal{R} \hat{e} \bullet \hat{b}_1\\
&i\, b_0 \mathcal{R} \underleftarrow{i} \bullet \hat{b}_0
\end{aligned}
$$

$\mathcal{R}$ is a reduction bisimulation.

$\hat{e}\,\underleftarrow{i} \Longrightarrow \hat{e}\,\hat{b}_1$ and $e\,i \Longrightarrow e\,b_1$ and $e\,b_1 \Longrightarrow e\,b_1$

$$\hat{\imath} \triangleq (\underrightarrow{e} \setminus \underrightarrow{e} \bullet \underrightarrow{b}_1) \mathbin{\&} (\underrightarrow{b}_0 \setminus \underrightarrow{b}_1) \mathbin{\&} (\underrightarrow{b}_1 \setminus \hat{\imath} \bullet \underrightarrow{b}_0)$$

$\underrightarrow{e}\,\hat{\imath} \Longrightarrow \underrightarrow{e}\,\underrightarrow{b}_1$ and $e\,i \Longrightarrow e\,b_1$ and $e\,b_1 \Longrightarrow e\,b_1$

$$\hat{q} \triangleq \mathop{\&}_{a \in \Sigma} (\underrightarrow{a} \setminus \hat{q}'_a)$$

Compare:

- $q \xrightarrow{a} q'_a$ if, and only if, $\underrightarrow{a}\,\hat{q} \longrightarrow \hat{q}'_a$.
- $q \xrightarrow{a} q'_a$ if, and only if, $\underrightarrow{a}\,\hat{q} \longrightarrow \Omega'$ for some $\Omega' = \hat{q}'_a$.
- $q \xrightarrow{a} q'_a$ and $\hat{q}'_a = \Omega'$ for some q'_a if, and only if, $\underrightarrow{a}\,\hat{q} \longrightarrow \Omega'$.

These differ in the placement of the existential quantifier. The first pair are, in fact, false.

For the former: Assume that $q \xrightarrow{a} q''_a$ and $\hat{q}''_a = \Omega' = \hat{q}'_a$. It might be that the states q''_a and q'_a are only bisimilar, not equal. In that case, $q \xrightarrow{a}\sim q'_a$ but, in general, not $q \xrightarrow{a} q'_a$ directly.

For the latter: Assume that $q \xrightarrow{a} q''_a$ and $\hat{q}''_a = \Omega'$. Choosing $q'_a \coloneqq q''_a$, we indeed have $q \xrightarrow{a} q'_a$ and $\hat{q}'_a = \Omega'$.

$$\begin{tikzcd}
q \arrow[r, dashed, "a"] \arrow[d, dash, "\mathcal{R}"'] & q'_a \arrow[d, dash, "\mathcal{R}"] \\
\underrightarrow{a}\,\hat{q} \arrow[r] & \Omega' = \hat{q}'_a
\end{tikzcd}
\qquad
\begin{tikzcd}
q \arrow[r, dashed, "a"] \arrow[d, dash, "\mathcal{R}"'] & q'_a \arrow[d, dash, dashed, "\mathcal{R}"] \\
\underrightarrow{a}\,\hat{q} \arrow[r] & \Omega' = \hat{q}'_a
\end{tikzcd}$$

3.4 *Encoding deterministic finite automata*

Recall from ?? our string rewriting specification of how an NFA processes its input. Given a deterministic finite automaton (DFA) $\mathcal{A} = (Q, ?, F)$ over an input alphabet Σ, the NFA's operational semantics are adequately captured by the folllwing string rewriting axioms:

$$\frac{}{a\,q \longrightarrow q'_a} \text{ for each transition } q \xrightarrow{a} q'_a.$$

$$\frac{}{\epsilon\,q \longrightarrow F(q)} \text{ for each state } q\text{, where } F(q) = \begin{cases} (\cdot) & \text{if } q \in F \\ n & \text{if } q \notin F\,. \end{cases}$$

3.4.1 *A functional choreography*

One possible choreography for this specification treats the input symbols $a \in \Sigma$ as atomic propositions $\underrightarrow{a}$; states $q \in Q$ as recursively defined propostions $\hat{q}$;and the end-of-word marker ϵ as an atomic proposition $\underrightarrow{\epsilon}$. In other words, the NFA's input is treated as a sequence of messages, $\underrightarrow{\epsilon}\, \underrightarrow{a}_n \cdots \underrightarrow{a}_2\, \underrightarrow{a}_1$, and the NFA's states are treated as [recursive] processes.

$a \mapsto \underrightarrow{a}$ for all $a \in \Sigma$; $q \mapsto \hat{q}$ for all $q \in Q$; and $\epsilon \mapsto \underrightarrow{\epsilon}$.

Using this assignment, the choreography constructed from the specification consists of the following definition, one for each NFA state $q \in Q$:

$$\hat{q} \triangleq \mathop{\&}_{a \in \Sigma} \mathop{\&}_{q'_a} (\underrightarrow{a} \setminus \hat{q}'_a) \mathbin{\&} (\underrightarrow{\epsilon} \setminus \hat{F}(q))\,.$$

Corollary 3.2. *If $a\, q \longrightarrow q'_a$, then $\underrightarrow{a}\, \hat{q} \Longrightarrow \hat{q}'_a$. If $\underrightarrow{a}\, \hat{q} \longrightarrow \Omega'$, then $a\, q \longrightarrow w'$ and $\Omega' \Longrightarrow \theta(w')$.*

Corollary 3.3. *If $q \xrightarrow{a} q'_a$, then $\underrightarrow{a}\, \hat{q} \Longrightarrow \hat{q}'_a$. If $\underrightarrow{a}\, \hat{q} \longrightarrow \hat{q}'_a$, then $q \xrightarrow{a} q''_a$ for some q''_a such that $\hat{q}'_a = \hat{q}''_a$.*

As an extended example, we will use ordered rewriting to specify how a DFA processes its input. Given a DFA $\mathcal{A} = (Q, ?, F)$ over an input alphabet Σ, the idea is to encode each state, $q \in Q$, as an ordered proposition, $\hat{q}$, in such a way that the DFA's operational semantics are adequately captured by [ordered] rewriting. [2]

Ideally, DFA transitions $q \xrightarrow{a} q'_a$ would be in bijective correspondence with rewriting steps $a\, \hat{q} \longrightarrow \hat{q}'_a$, where each input symbol a is encoded by a matching [propositional] atom. We will return to the possibility of this kind of tight correspondence in **??**, but, for now, we will content ourselves with a correspondence with traces rather than individual steps, adopting the following desiderata:

- $q \xrightarrow{a} q'_a$ if, and only if, $a\, \hat{q} \Longrightarrow \hat{q}'_a$, for all input symbols $a \in \Sigma$.
- $q \in F$ if, and only if, $\epsilon\, \hat{q} \Longrightarrow \mathbf{1}$, where the atom ϵ functions as an end-of-word marker.

Given the reversal (anti-)homomorphism from finite words to ordered contexts defined in the adjacent figure, the first desideratum is subsumed by a third:

- $q \xrightarrow{w} q'$ if, and only if, $w^{\mathrm{R}}\, \hat{q} \Longrightarrow \hat{q}'$, for all finite words $w \in \Sigma^*$.

From these desiderata [and the observation that DFAs' graphs frequently[3] contain cycles], we arrive at the following encoding, in which each state is encoded by one of a collection of mutually recursive definitions:[4]

$$\hat{q} \triangleq \left(\&_{a \in \Sigma} (a \setminus \hat{q}'_a)\right) \mathbin{\&} \left(\epsilon \setminus \hat{F}(q)\right)$$

[2] [In general, the behavior of a DFA state is recursive, so the proposition $\hat{q}$ will be recursively defined.]

$$\begin{aligned} (w_1\, w_2)^{\mathrm{R}} &= w_2^{\mathrm{R}}\, w_1^{\mathrm{R}} \\ \epsilon^{\mathrm{R}} &= \cdot \\ a^{\mathrm{R}} &= a \end{aligned}$$

Figure 3.1: An (anti-)homomorphism for reversal of finite words to ordered contexts

[3] Actually, there is always at least one cycle in a well-formed DFA.

[4] q'_a, using function or relation?

where

$$q \xrightarrow{a} q'_a, \text{ for all input symbols } a \in \Sigma, \quad \text{and} \quad \hat{F}(q) = \begin{cases} \mathbf{1} & \text{if } q \in F \\ \top & \text{if } q \notin F\,. \end{cases}$$

Just as each state q has exactly one successor for each input symbol a, its encoding, $\hat{q}$, has exactly one input clause, $(a \setminus \cdots)$, for each symbol a.

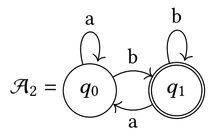


Figure 3.2: A DFA that accepts, from state q_0, exactly those words that end with b. (Repeated from ??.)

For a concrete instance of this encoding, recall from **??** the DFA (repeated in the adjacent figure) that accepts exactly those words, over the alphabet $\Sigma = \{a, b\}$, that end with b; that DFA is encoded by the following definitions:

$$\hat{q}_0 \triangleq (a \setminus \hat{q}_0) \mathbin{\&} (b \setminus \hat{q}_1) \mathbin{\&} (\epsilon \setminus \top)$$
$$\hat{q}_1 \triangleq (a \setminus \hat{q}_0) \mathbin{\&} (b \setminus \hat{q}_1) \mathbin{\&} (\epsilon \setminus \mathbf{1})$$

Indeed, just as the DFA has a transition $q_0 \xrightarrow{b} q_1$, its encoding admits a trace

$$b\,\hat{q}_0 = b\,\big((a \setminus \hat{q}_0) \mathbin{\&} (b \setminus \hat{q}_1) \mathbin{\&} (\epsilon \setminus \top)\big) \Longrightarrow b\,(b \setminus \hat{q}_1) \longrightarrow \hat{q}_1\,.$$

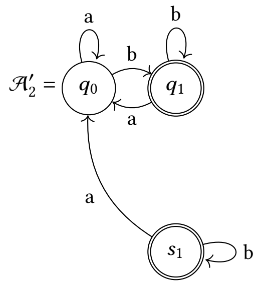
And, just as q_1 is an accepting state, its encoding also admits a trace

$$\epsilon\,\hat{q}_1 = \epsilon\,\big((a \setminus \hat{q}_0) \mathbin{\&} (b \setminus \hat{q}_1) \mathbin{\&} (\epsilon \setminus \mathbf{1})\big) \Longrightarrow \epsilon\,(\epsilon \setminus \mathbf{1}) \longrightarrow \mathbf{1}\,.$$

More generally, this encoding is complete, in the sense that it simulates all DFA transitions: $q \xrightarrow{a} q'$ implies $a\,\hat{q} \Longrightarrow \hat{q}'$, for all states q and q' and input symbols a.

However, the converse does not hold – the encoding is unsound because there are rewritings that cannot be simulated by a DFA transition.

False claim 3.4. *Let $\mathcal{A} = (Q, \longrightarrow, F)$ be a DFA over the input alphabet Σ. Then $a\,\hat{q} \Longrightarrow \hat{q}'$ implies $q \xrightarrow{a} q'$, for all input symbols $a \in \Sigma$.*



$$\hat{q}_0 \triangleq (a \setminus \hat{q}_0) \mathbin{\&} (b \setminus \hat{q}_1) \mathbin{\&} (\epsilon \setminus \top)$$
$$\hat{q}_1 \triangleq (a \setminus \hat{q}_0) \mathbin{\&} (b \setminus \hat{q}_1) \mathbin{\&} (\epsilon \setminus \mathbf{1})$$
$$\hat{s}_1 \triangleq (a \setminus \hat{q}_0) \mathbin{\&} (b \setminus \hat{s}_1) \mathbin{\&} (\epsilon \setminus \mathbf{1})$$

Figure 3.3: fig:ordered-rewriting:dfa-counterexample:dfa A slightly modified version of the DFA from fig. 3.15; and fig:ordered-rewriting:dfa-counterexample:encoding its encoding

Counterexample. Consider the DFA and encoding shown in the adjacent figure; it is the same DFA as shown in fig. 3.15, but with one added state, s_1, that is unreachable from q_0 and q_1. Notice that, as a coinductive consequence of the equirecursive treatment of definitions, $\hat{q}_1 = \hat{s}_1$. Previously, we saw that $b\,\hat{q}_0 \Longrightarrow \hat{q}_1$; hence $b\,\hat{q}_0 \Longrightarrow \hat{s}_1$. However, the DFA has no $q_0 \xrightarrow{b} s_1$ transition (because $q_1 \neq s_1$), and so this encoding is unsound with respect to the operational semantics of DFAs. □

As this counterexample shows, the lack of adequacy stems from attempting to use an encoding that is not injective – here, $q_1 \neq s_1$ even though $\hat{q}_1 = \hat{s}_1$. In other words, eqality of state encodings is a coarser eqvivalence than equality of the states themselves.

One possible remedy for this lack of adequacy might be to revise the encoding to have a stronger nominal character. By tagging each state's encoding with an atom that is unique to that state, we can make the encoding manifestly injective. For instance, given the pairwise distinct atoms $\{q \mid q \in F\}$

and $\{\bar{q} \mid q \in Q - F\}$ to tag final and non-final states, respectively, we could define an alternative encoding, $\check{q}$:

$$\check{q} \triangleq \left(\&_{a\in\Sigma}(a \setminus \check{q}'_a)\right) \,\&\, \left(\epsilon \setminus \check{F}(q)\right)$$

where

$$q \xrightarrow{a} q'_a, \text{ for all input symbols } a \in \Sigma, \quad \text{and} \quad \check{F}(q) = \begin{cases} q & \text{if } q \in F \\ \bar{q} & \text{if } q \notin F\,. \end{cases}$$

Under this alternative encoding, the states q_1 and s_1 of fig. 3.16 are no longer a counterexample to injectivity: Because q_1 and s_1 are distinct states, they correspond to distinct tags, and so $\check{q}_1 \neq \check{s}_1$.

Although such a solution is certainly possible, it seems unsatisfyingly ad hoc. A closer examination of the preceding counterexample reveals that the states q_1 and s_1, while not equal, are in fact bisimilar (??). In other words, although the encoding is not, strictly speaking, injective, it is injective *up to bisimilarity*: $\hat{q} = \hat{s}$ implies $q \sim s$. This suggests a more elegant solution to the apparent lack of adequacy: the encoding's adequacy should be judged up to DFA bisimilarity.

Theorem 3.19 (DFA adequacy up to bisimilarity). *Let $\mathcal{A} = (Q, ?, F)$ be a DFA over the input alphabet Σ. Then, for all states q, q', and s:*

1. *$q \sim s$ if, and only if, $\hat{q} = \hat{s}$.*
2. *$q \sim\!\xrightarrow{a}\!\sim q'$ if, and only if, $a\,\hat{q} \Longrightarrow \hat{q}'$, for all input symbols $a \in \Sigma$. More generally, $q \sim\!\xrightarrow{w}\!\sim q'$ if, and only if, $w^{\mathrm{R}}\,\hat{q} \Longrightarrow \hat{q}'$, for all finite words $w \in \Sigma^*$.*
3. *$q \in F$ if, and only if, $\epsilon\,\hat{q} \Longrightarrow \mathbf{1}$.*

Before proving this theorem, we must first prove a lemma: the only traces from one state's encoding to another's are the trivial traces.

Lemma 3.5. *Let $\mathcal{A} = (Q, ?, F)$ be a DFA over the input alphabet Σ. For all states q and s, if $\hat{q} \Longrightarrow \hat{s}$, then $\hat{q} = \hat{s}$.*

Proof. Assume that a trace $\hat{q} \Longrightarrow \hat{s}$ exists. If the trace is trivial, then $\hat{q} = \hat{s}$ is immediate. Otherwise, the trace is nontrivial and consists of a strictly positive number of rewriting steps. By inversion, those rewriting steps drop one or more conjuncts from $\hat{q}$ to form $\hat{s}$. Every DFA state's encoding contains exactly $|\Sigma| + 1$ conjuncts – one for each input symbol a and one for the end-of-word marker, ϵ. If even one conjunct is dropped from $\hat{q}$, not enough conjuncts will remain to form $\hat{s}$. Thus, a nontrivial trace $\hat{q} \Longrightarrow \hat{s}$ cannot exist. □

It is important to differentiate this lemma from the false claim that a state's encoding can take no rewriting steps. There certainly exist nontrivial traces from $\hat{q}$, but they do not arrive at the encoding of any state.

With this lemma now in hand, we can proceed to proving adequacy up to bisimilarity.

THEOREM 3.6 (DFA adequacy up to bisimilarity). *Let $\mathcal{A} = (Q, ?, F)$ be a DFA over the input alphabet Σ. Then, for all states q, q', and s:*

1. *$q \sim s$ if, and only if, $\hat{q} = \hat{s}$.*
2. *$q \sim\!\xrightarrow{a}\!\sim q'$ if, and only if, $a\,\hat{q} \Longrightarrow \hat{q}'$, for all input symbols $a \in \Sigma$. More generally, $q \sim\!\xrightarrow{w}\!\sim q'$ if, and only if, $w^{\mathrm{R}}\,\hat{q} \Longrightarrow \hat{q}'$, for all finite words $w \in \Sigma^*$.*
3. *$q \in F$ if, and only if, $\epsilon\,\hat{q} \Longrightarrow \mathbf{1}$.*

Proof. Each part is proved in turn. The proof of part 2 depends on the proof of part 1.

1. We shall show that bisimilarity coincides with equality of encodings, proving each direction separately.
   - To prove that bisimilar DFA states have equal encodings – *i.e.*, that $q \sim s$ implies $\hat{q} = \hat{s}$ – a fairly straightforward proof by coinduction suffices.

     Let q and s be bisimilar states. By the definition of bisimilarity (??), two properties hold:
     - For all input symbols a, the unique a-successors of q and s are also bisimilar.
     - States q and s have matching finalities – *i.e.*, $q \in F$ if and only if $s \in F$.

     Applying the coinductive hypothesis to the former property, we may deduce that, for all symbols a, the a-successors of q and s also have equal encodings. From the latter property, it follows that $\hat{F}(q) = \hat{F}(s)$. Because definitions are interpreted equirecursively, these equalities together imply that q and s themselves have equal encodings.
   - To prove the converse – that states with equal encodings are bisimilar – we will show that the relation $\mathcal{R} = \{(q, s) \mid \hat{q} = \hat{s}\}$, which relates states if they have equal encodings, is a bisimulation and is therefore included in bisimilarity.
     - The relation $\mathcal{R}$ is symmetric.
     - We must show that $\mathcal{R}$-related states have $\mathcal{R}$-related a-successors, for all input symbols a.
       Let q and s be $\mathcal{R}$-related states. Being $\mathcal{R}$-related, q and s have equal encodings; because definitions are interpreted equirecursively, the unrollings of those encodings are also equal. By definition of the encoding, it follows that, for each input symbol a, the unique a-successors of q and s have equal encodings. Therefore, for each a, the a-successors of q and s are themselves $\mathcal{R}$-related.
     - We must show that $\mathcal{R}$-related states have matching finalities.
       Let q and s be $\mathcal{R}$-related states, with q a final state. Being $\mathcal{R}$-related, q and s have equal encodings; because definitions are interpreted equirecursively, the unrollings of those encodings are also equal. It follows that $\hat{F}(q) = \hat{F}(s)$, and so s is also a final state.

2. We would like to prove that $q \sim\!\xrightarrow{a}\!\sim q'$ if, and only if, $a\,\hat{q} \Longrightarrow \hat{q}'$, or, more generally, that $q \sim\!\xrightarrow{w}\!\sim q'$ if, and only if, $w^{\mathrm{R}}\,\hat{q} \Longrightarrow \hat{q}'$. Because bisimilar states have equal encodings (part 1) and bisimilarity is reflexive (??), it suffices to show two stronger statements: (a) that $q \xrightarrow{w} q'$ implies $w^{\mathrm{R}}\,\hat{q} \Longrightarrow \hat{q}'$; and (b) that $w^{\mathrm{R}}\,\hat{q} \Longrightarrow \hat{q}'$ implies $q \xrightarrow{w}\!\sim q'$. We prove these in turn.
   (a) We shall prove that $q \xrightarrow{w} q'$ implies $w^{\mathrm{R}}\,\hat{q} \Longrightarrow \hat{q}'$ by induction over the structure of word w.
      - Consider the case of the empty word, ϵ; we must show that $q = q'$ implies $\hat{q} \Longrightarrow \hat{q}'$. Because the encoding is a function, this is immediate.
      - Consider the case of a nonempty word, $a\,w$; we must show that $q \xrightarrow{a}\xrightarrow{w} q'$ implies $w^{\mathrm{R}}\,a\,\hat{q} \Longrightarrow \hat{q}'$. Let q'_a be an a-successor of state q that is itself w-succeeded by state q'. There exists, by definition of the encoding, a trace

        $$w^{\mathrm{R}}\,a\,\hat{q} \Longrightarrow w^{\mathrm{R}}\,a\,(a \setminus \hat{q}'_a) \longrightarrow w^{\mathrm{R}}\,\hat{q}'_a \Longrightarrow \hat{q}'\,,$$

        with the trace's tail justified by an appeal to the inductive hypothesis.
   (b) We shall prove that $w^{\mathrm{R}}\,\hat{q} \Longrightarrow \hat{q}'$ implies $q \xrightarrow{w}\!\sim q'$ by induction over the structure of word w.
      - Consider the case of the empty word, ϵ; we must show that $\hat{q} \Longrightarrow \hat{q}'$ implies $q \sim q'$. By lemma 3.18, $\hat{q} \Longrightarrow \hat{q}'$ implies that q and q' have equal encodings. Part 1 can then be used to establish that q and q' are bisimilar.
      - Consider the case of a nonempty word, $a\,w$; we must show that $w^{\mathrm{R}}\,a\,\hat{q} \Longrightarrow \hat{q}'$ implies $q \xrightarrow{a}\xrightarrow{w}\!\sim q'$. By inversion[5], the given trace can only begin by inputting a:

        $$w^{\mathrm{R}}\,a\,\hat{q} \Longrightarrow w^{\mathrm{R}}\,a\,(a \setminus \hat{q}'_a) \longrightarrow w^{\mathrm{R}}\,\hat{q}'_a \Longrightarrow \hat{q}'\,,$$

        where q'_a is an a-successor of state q. An appeal to the inductive hypothesis on the trace's tail yields $q'_a \xrightarrow{w}\!\sim q'$, and so the DFA admits $q \xrightarrow{a}\xrightarrow{w}\!\sim q'$, as required.

[5] Is this enough justification?

3. We shall prove that the final states are exactly those states q such that $\epsilon\,\hat{q} \Longrightarrow \mathbf{1}$.
   - Let q be a final state; accordingly, $\hat{F}(q) = \mathbf{1}$. There exists, by definition of the encoding, a trace

     $$\epsilon\,\hat{q} \Longrightarrow \epsilon\,(\epsilon \setminus \hat{F}(q)) \longrightarrow \hat{F}(q) = \mathbf{1}\,.$$

   - Assume that a trace $\epsilon\,\hat{q} \Longrightarrow \mathbf{1}$ exists. By inversion[6], this trace can only begin by inputting ϵ:

     $$\epsilon\,\hat{q} \Longrightarrow \epsilon\,(\epsilon \setminus \hat{F}(q)) \longrightarrow \hat{F}(q) \Longrightarrow \mathbf{1}\,.$$

   The tail of this trace, $\hat{F}(q) \Longrightarrow \mathbf{1}$, can exist only if q is a final state. □

[6] Is this enough justification?

3.4.2 *Encoding nondeterministic finite automata?*

We would certainly be remiss if we did not attempt to generalize the rewriting specification of DFAs to one for their nondeterministic cousins.

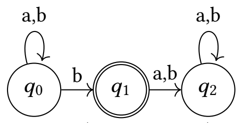
Differently from DFA states, an NFA state q may have several nondeterministic successors for each input symbol a. To encode the NFA state q, all of its a-successors are collected in an alternative conjunction underneath the left-handed input of a. Thus, the encoding of an NFA state q becomes

$$\hat{q} \triangleq \Bigl(\mathop{\&}_{a \in \Sigma} \bigl(a \setminus (\&_{q'_a} \hat{q}'_a)\bigr)\Bigr) \mathbin{\&} \bigl(\epsilon \setminus \hat{F}(q)\bigr) ,$$

where $\hat{F}(q)$ is defined as for DFAs.

The adjacent figure recalls from **??** an NFA that accepts exactly those words, over the alphabet $\Sigma = \{a, b\}$, that end with b. Using the above encoding of NFAs, ordered rewriting does indeed simulate this NFA. For example, just as there are transitions $q_0 \xrightarrow{b} q_0$ and $q_0 \xrightarrow{b} q_1$, there are traces

$$b\,\hat{q}_0 \Longrightarrow b\,\bigl(b \setminus (\hat{q}_0 \mathbin{\&} \hat{q}_1)\bigr) \longrightarrow \hat{q}_0 \mathbin{\&} \hat{q}_1 \begin{cases} \nearrow \hat{q}_0 \\ \searrow \hat{q}_1 \end{cases}$$



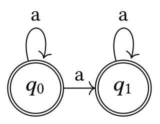
$$\hat{q}_0 \triangleq (a \setminus \hat{q}_0) \mathbin{\&} \bigl(b \setminus (\hat{q}_0 \mathbin{\&} \hat{q}_1)\bigr) \mathbin{\&} (\epsilon \setminus \top)$$
$$\hat{q}_1 \triangleq (a \setminus \hat{q}_2) \mathbin{\&} (b \setminus \hat{q}_2) \mathbin{\&} (\epsilon \setminus \mathbf{1})$$
$$\hat{q}_2 \triangleq (a \setminus \hat{q}_2) \mathbin{\&} (b \setminus \hat{q}_2) \mathbin{\&} (\epsilon \setminus \top)$$

Figure 3.4: fig:ordered-rewriting:nfa-example:nfa An NFA that accepts exactly those words, over the alphabet $\Sigma = \{a, b\}$, that end with b; and fig:ordered-rewriting:nfa-example:encoding its encoding

Unfortunately, while it does simulate NFA behavior, this encoding is not adequate. Like DFA states, NFA states that have equal encodings are bisimilar. However, for NFAs, the converse does not hold: bisimilar states do not necessarily have equal encodings.

FALSE CLAIM 3.7. *Let $\mathcal{A} = (Q, ?, F)$ be an NFA over input alphabet Σ. Then $q \sim s$ implies $\hat{q} = \hat{s}$, for all states q and s.*

Counterexample. Consider the NFA and encoding depicted in the adjacent figure. It is easy to verify that the relation $\{q_1\} \times \{q_0, q_1\}$ is a bisimulation; in particular, q_1 simulates the $q_0 \xrightarrow{a} q_1$ transition by its self-loop, $q_1 \xrightarrow{a} q_1$. Hence, q_0 and s_0 are bisimilar. It is equally easy to verify, by unrolling the definitions used in the encoding, that $\hat{q}_0 \neq \hat{s}_0$. □



$$\hat{q}_0 \triangleq \bigl(a \setminus (\hat{q}_0 \mathbin{\&} \hat{q}_1)\bigr) \mathbin{\&} (\epsilon \setminus \mathbf{1})$$
$$\hat{q}_1 \triangleq (a \setminus \hat{q}_1) \mathbin{\&} (\epsilon \setminus \mathbf{1})$$

Figure 3.5: An NFA that accepts all finite words over the alphabet $\Sigma = \{a\}$

For DFAs, bisimilar states do have equal encodings because the inherent determinism DFA bisimilarity is a rather fine-grained equivalence. Because each DFA state has exactly one successor for each input symbol The additional flexibility entailed by nondeterminism

Once again, it would be possible to construct an adequate encoding, by tagging each state with a unique atom.

For the moment, we will put aside the question of an adequate encoding of NFAs.

3.5 *Introduction*

In the previous **??**, we saw that the ordered sequent calculus can be given a resource interpretation in which sequents $\Omega \vdash A$ may be read as “From resources Ω, resource goal A is achievable." For instance, the left rule for ordered conjunction ($\bullet$L, see adjacent display) was read "Goal C is achievable from resource $A \bullet B$ if it is achievable from the separate resources $A\,B$."

$$\frac{\Omega'_L\; A\, B\; \Omega'_R \vdash C}{\Omega'_L\; (A \bullet B)\; \Omega'_R \vdash C}\;\bullet\text{L}$$

As alluded in the previous ??'s discussion of ordered conjunction[7], this $\bullet$L rule is essentially a rule of resource decomposition: it decomposes [the resource] $A \bullet B$ into the separate resources $A\,B$ and relegates the unchanged goal C to a secondary role.

[7] See ??.

THIS CHAPTER begins by exploring a refactoring of the ordered sequent calculus's left rules around this idea of resource decomposition (??). Most of the left rules can be easily refactored in this way, although a few will prove resistant to the change.

Emphasizing resource decomposition naturally leads us to a rewriting interpretation of (a fragment of) ordered logic (??). This rewriting system is closely related to traditional notions of string rewriting,[8] but simultaneously restricts and generalizes [...] along distinct axes.

[8] ??.

The connection of ordered logic and the Lambek calculus to rewriting is certainly not new. **Lambek:AMM58**'s original article[9]

[9] **Lambek:AMM58**.

This development borrows from **Cervesato+Scedrov:IC09**'s work on intuitionistic linear logic as an expressive rewriting framework that generalizes traditional notions of multiset rewriting.[10]

[10] **Cervesato+Scedrov:IC09**.

MOST of the left rules could be seen as decomposing resources. The left rules were seen as decomposing resources, such as the $\bullet$L rule decomposing $A \bullet B$ into the resources $A\,B$. The right rules, on the other hand, were seen as ...

$$\frac{\Omega'_L\; A\, B\; \Omega'_R \vdash C}{\Omega'_L\; (A \bullet B)\; \Omega'_R \vdash C}\;\bullet\text{L}$$

Replacing the left rules with a single, common rule ... and a new judgment, $\Omega \longrightarrow \Omega'$, that exposes [makes [more] explicit] the decomposition of resources/state transformation aspect.

3.6 *Most left rules decompose ordered resources*

Recall two of the ordered sequent calculus's left rules: $\bullet$L and $\&$L$_1$.

$$\frac{\Omega'_L\; A\, B\; \Omega'_R \vdash C}{\Omega'_L\; (A \bullet B)\; \Omega'_R \vdash C}\;\bullet\text{L} \qquad \frac{\Omega'_L\; A\; \Omega'_R \vdash C}{\Omega'_L\; (A \,\&\, B)\; \Omega'_R \vdash C}\;\&\text{L}_1$$

Both rules decompose the principal resource: in the $\bullet$L rule, $A \bullet B$ into the separate resources $A\,B$; and, in the $\&$L$_1$ rule, $A \,\&\, B$ into A. However, in both cases, the resource decomposition is somewhat obscured by boilerplate. The framed contexts Ω'_L and Ω'_R and goal C serve to enable the rules to be applied anywhere [in the string of resources], without restriction; these concerns are not specific to the $\bullet$L and $\&$L$_1$ rules, but are general boilerplate that arguably should be factored out.

To decouple the resource decomposition from the surrounding boilerplate, we will introduce a new judgment, $\Omega \longrightarrow \Omega'$, meaning "Resources Ω may be decomposed into [resources] Ω'." With this new judgment comes a cut principle, $\textsc{cut}^{\longrightarrow}$, into which all of the boilerplate is factored:

$$\frac{\Omega \longrightarrow \Omega' \quad \Omega'_L\,\Omega'\,\Omega'_R \vdash C}{\Omega'_L\,\Omega\,\Omega'_R \vdash C}\;\textsc{cut}^{\longrightarrow}\,.$$

The standard left rules can be recovered from resource decomposition rules using this cut principle. For example, the decomposition of $A \bullet B$ into $A\,B$ is captured by

$$\frac{}{A \bullet B \longrightarrow A\,B}\;{\bullet}\textsc{d}\,,$$

and the standard $\bullet\textsc{l}$ rule can then be recovered as shown in the neighboring figure. The left rules for $\mathbf{1}$ and $A \mathbin{\&} B$ can be similarly refactored into resource decomposition rules.

$$\frac{\Omega'_L\,A\,B\,\Omega'_R \vdash C}{\Omega'_L\,(A \bullet B)\,\Omega'_R \vdash C}\;{\bullet}\textsc{l}$$

$$\leftrightsquigarrow$$

$$\frac{\dfrac{}{A \bullet B \longrightarrow A\,B}\;{\bullet}\textsc{d} \quad \Omega'_L\,A\,B\,\Omega'_R \vdash C}{\Omega'_L\,(A \bullet B)\,\Omega'_R \vdash C}\;\textsc{cut}^{\longrightarrow}\,.$$

Figure 3.6: A refactoring of the $\bullet\textsc{l}$ rule as resource decomposition

Even the left rules for left- and right-handed implications can be refactored in this way, despite the additional, minor premises that those rules carry. To keep the correspondence between resource decomposition rules and left rules close, we could introduce the decomposition rules

$$\frac{\Omega \vdash A}{\Omega\,(A \mathbin{\backslash} B) \longrightarrow B}\;{\backslash}\textsc{d}' \qquad \text{and} \qquad \frac{\Omega \vdash A}{(B \mathbin{/} A)\,\Omega \longrightarrow B}\;{/}\textsc{d}'\,.$$

Just as for ordered conjunction, the left rules for left- and right-handed implication are then recovered by combining a decomposition rule with the $\textsc{cut}^{\longrightarrow}$ rule (see adjacent figure).

$$\frac{\Omega \vdash A \quad \Omega'_L\,B\,\Omega'_R \vdash C}{\Omega'_L\,\Omega\,(A \mathbin{\backslash} B)\,\Omega'_R \vdash C}\;{\backslash}\textsc{l}$$

$$\leftrightsquigarrow$$

$$\frac{\dfrac{\Omega \vdash A}{\Omega\,(A \mathbin{\backslash} B) \longrightarrow B}\;{\backslash}\textsc{d}' \quad \Omega'_L\,B\,\Omega'_R \vdash C}{\Omega'_L\,\Omega\,(A \mathbin{\backslash} B)\,\Omega'_R \vdash C}\;\textsc{cut}^{\longrightarrow}$$

Figure 3.7: A refactoring of the $\backslash\textsc{l}$ rule using a resource decomposition rule

Although these $\backslash\textsc{d}'$ and $/\textsc{d}'$ rules keep the correspondence between resource decomposition rules and left rules close, they differ from the other decomposition rules in two significant ways. First, the above $\backslash\textsc{d}'$ and $/\textsc{d}'$ rules have premises, and those premises create a dependence of the decomposition judgment upon general provability. Second, the above $\backslash\textsc{d}'$ and $/\textsc{d}'$ rules do not decompose the principal proposition into immediate subformulas. This contrasts with, for example, the $\bullet\textsc{d}$ rule that decomposes $A \bullet B$ into the immediate subformulas $A\,B$.

For these reasons, the above $\backslash\textsc{d}'$ and $/\textsc{d}'$ rules are somewhat undesirable. Fortunately, there is an alternative. Filling in the $\Omega \vdash A$ premises with the $\textsc{id}^A$ rule, we arrive at the derivable rules

$$\frac{}{A\,(A \mathbin{\backslash} B) \longrightarrow B}\;{\backslash}\textsc{d} \qquad \text{and} \qquad \frac{}{(B \mathbin{/} A)\,A \longrightarrow B}\;{/}\textsc{d}\,.$$

The standard $\backslash\textsc{l}$ and $/\textsc{l}$ rules can still be recovered from these more specific decomposition rules, thanks to $\textsc{cut}$ (see adjacent figure). These revised, nullary decomposition rules correct the earlier drawbacks: like the other decomposition rules, they now have no premises and only refer to immediate subformulas. Moreover, these rules have the advantage of matching two of the axioms from **Lambek:AMM58**'s original article.[11]

$$\frac{\Omega \vdash A \quad \Omega'_L\,B\,\Omega'_R \vdash C}{\Omega'_L\,\Omega\,(A \mathbin{\backslash} B)\,\Omega'_R \vdash C}\;{\backslash}\textsc{l}$$

$$\leftrightsquigarrow$$

$$\frac{\Omega \vdash A \quad \dfrac{\dfrac{}{A\,(A \mathbin{\backslash} B) \longrightarrow B}\;{\backslash}\textsc{d} \quad \Omega'_L\,B\,\Omega'_R \vdash C}{\Omega'_L\,A\,(A \mathbin{\backslash} B)\,\Omega'_R \vdash C}\;\textsc{cut}^{\longrightarrow}}{\Omega'_L\,\Omega\,(A \mathbin{\backslash} B)\,\Omega'_R \vdash C}\;\textsc{cut}^{A}$$

Figure 3.8: A refactoring of the $\backslash\textsc{l}$ rule using an alternative resource decomposition rule

[11] **Lambek:AMM58**.

So, for most ordered logical connectives, this approach works perfectly. Unfortunately, the left rules for additive disjunction, $A \oplus B$, and its unit, $\mathbf{0}$, are

Figure 3.9: A refactoring of the ordered sequent calculus to emphasize that most left rules amount to resource decomposition

$$\frac{\Omega \vdash A \quad \Omega'_L\, A\, \Omega'_R \vdash C}{\Omega'_L\, \Omega\, \Omega'_R \vdash C}\ \textsc{cut}^A \qquad \frac{}{A \vdash A}\ \textsc{id}^A$$

$$\frac{\Omega \longrightarrow \Omega' \quad \Omega'_L\, \Omega'\, \Omega'_R \vdash C}{\Omega'_L\, \Omega\, \Omega'_R \vdash C}\ \textsc{cut}^{\longrightarrow}$$

$$\frac{\Omega_1 \vdash A \quad \Omega_2 \vdash B}{\Omega_1\, \Omega_2 \vdash A \bullet B}\ {\bullet}\textsc{r} \qquad \frac{}{A \bullet B \longrightarrow A\, B}\ {\bullet}\textsc{d}$$

$$\frac{}{\cdot \vdash \mathbf{1}}\ \mathbf{1}\textsc{r} \qquad \frac{}{\mathbf{1} \longrightarrow \cdot}\ \mathbf{1}\textsc{d}$$

$$\frac{\Omega \vdash A \quad \Omega \vdash B}{\Omega \vdash A \mathbin{\&} B}\ \&\textsc{r} \qquad \frac{}{A \mathbin{\&} B \longrightarrow A}\ \&\textsc{d}_1 \qquad \frac{}{A \mathbin{\&} B \longrightarrow B}\ \&\textsc{d}_2$$

$$\frac{}{\Omega \vdash \top}\ \top\textsc{r} \qquad \text{(no } \top\textsc{d} \text{ rule)}$$

$$\frac{A\, \Omega \vdash B}{\Omega \vdash A \setminus B}\ \setminus\textsc{r} \qquad \frac{}{A\, (A \setminus B) \longrightarrow B}\ \setminus\textsc{d}$$

$$\frac{\Omega\, A \vdash B}{\Omega \vdash B \mathbin{/} A}\ /\textsc{r} \qquad \frac{}{(B \mathbin{/} A)\, A \longrightarrow B}\ /\textsc{d}$$

$$\frac{\Omega \vdash A}{\Omega \vdash A \oplus B}\ \oplus\textsc{r}_1 \qquad \frac{\Omega \vdash B}{\Omega \vdash A \oplus B}\ \oplus\textsc{r}_2 \qquad \frac{\Omega'_L\, A\, \Omega'_R \vdash C \quad \Omega'_L\, B\, \Omega'_R \vdash C}{\Omega'_L\, (A \oplus B)\, \Omega'_R \vdash C}\ \oplus\textsc{l}$$

$$\text{(no } \mathbf{0}\textsc{r} \text{ rule)} \qquad \frac{}{\Omega'_L\, \mathbf{0}\, \Omega'_R \vdash C}\ \mathbf{0}\textsc{l}$$

resistant to this kind of refactoring. The difficulty with additive disjunction isn't that its left rule, $\oplus$L, doesn't decompose the resource $A \oplus B$. The $\oplus$L rule certainly does decompose $A \oplus B$, but it does so [...]. $A \oplus B \longrightarrow A \mid B$ [...] retain the standard $\oplus$L and $\mathbf{0}$L rules.

$$\frac{\Omega'_L\, A\, \Omega'_R \vdash C \quad \Omega'_L\, B\, \Omega'_R \vdash C}{\Omega'_L\, (A \oplus B)\, \Omega'_R \vdash C}\ \oplus\textsc{l}$$

Figure 3.9 presents the fully refactored sequent calculus for ordered logic. This refactored calculus is sound and complete with respect to the ordered sequent calculus (??).

Theorem 3.8 (Soundness). *If $\Omega \vdash A$ is derivable in the refactored calculus of fig. 3.9, then $\Omega \vdash A$ is derivable in the ordered sequent calculus (??).*

Proof. By structural induction on the given derivation. The key lemma is the admissibility of $\textsc{cut}^{\longrightarrow}$ in the ordered sequent calculus:

If $\Omega \longrightarrow \Omega'$ and $\Omega'_L\, \Omega'\, \Omega'_R \vdash C$, then $\Omega'_L\, \Omega\, \Omega'_R \vdash C$.

This lemma can be proved by case analysis of the decomposition $\Omega \longrightarrow \Omega'$, reconstituting the corresponding left rule along the lines of the sketches from figs. 3.6 and 3.8. □

$$\frac{}{A \bullet B \longrightarrow A\, B}\;{\bullet}\textsc{d} \qquad \frac{}{\mathbf{1} \longrightarrow \cdot}\;\mathbf{1}\textsc{d}$$

$$\frac{}{A \mathbin{\&} B \longrightarrow A}\;{\&}\textsc{d}_1 \qquad \frac{}{A \mathbin{\&} B \longrightarrow B}\;{\&}\textsc{d}_2 \qquad \text{(no } \top\textsc{d} \text{ rule)}$$

$$\frac{}{A\,(A \setminus B) \longrightarrow B}\;\setminus\textsc{d} \qquad \frac{}{(B \mathbin{/} A)\,A \longrightarrow B}\;/\textsc{d}$$

$$\text{(no } \oplus\textsc{d} \text{ and } \mathbf{0}\textsc{d} \text{ rules)}$$

$$\frac{\Omega_1 \longrightarrow \Omega_1'}{\Omega_1\,\Omega_2 \longrightarrow \Omega_1'\,\Omega_2}\;{\longrightarrow}\textsc{c}_\textsc{l} \qquad \frac{\Omega_2 \longrightarrow \Omega_2'}{\Omega_1\,\Omega_2 \longrightarrow \Omega_1\,\Omega_2'}\;{\longrightarrow}\textsc{c}_\textsc{r}$$

$$\frac{}{\Omega \Longrightarrow \Omega}\;{\Longrightarrow}\textsc{r} \qquad \frac{\Omega \longrightarrow \Omega' \quad \Omega' \Longrightarrow \Omega''}{\Omega \Longrightarrow \Omega''}\;{\Longrightarrow}\textsc{t}$$

Figure 3.10: A rewriting fragment of ordered logic, based on resource decomposition

THEOREM 3.9 (Completeness). *If $\Omega \vdash A$ is derivable in the ordered sequent calculus (??), then $\Omega \vdash A$ is derivable in the refactored calculus of fig. 3.9.*

Proof. By structural induction on the given derivation. The critical cases are the left rules; they are resolved along the lines of the sketches shown in figs. 3.6 and 3.8. □

3.7 *Decomposition as rewriting*

Thus far, we have used the decomposition judgment, $\Omega \longrightarrow \Omega'$, and its rules as the basis for a reconfigured sequent-like calculus for ordered logic. Instead, we can also view decomposition as the foundation of a rewriting system grounded in ordered logic. For example, the decomposition of resource $A \bullet B$ into $A\,B$ by the $\bullet$D rule can also be seen as *rewriting* $A \bullet B$ into $A\,B$. More generally, the decomposition judgment $\Omega \longrightarrow \Omega'$ can be read as "Ω rewrites to Ω'."

Figure 3.10 summarizes the rewriting system that we obtain from the refactored sequent-like calculus of fig. 3.9. Essentially, the ordered rewriting system is obtained by discarding all rules except for the decomposition rules. However, if only the decomposition rules are used, rewritings cannot occur within a larger context. For example, the $\setminus$D rule derives $A\,(A \setminus B) \longrightarrow B$, but $\Omega_L'\,A\,(A \setminus B)\,\Omega_R' \longrightarrow \Omega_L'\,B\,\Omega_R'$ would not be derivable in general. In the refactored calculus of fig. 3.9, this kind of framing is taken care of by the cut principle for decomposition, $\textsc{cut}^{\longrightarrow}$. To express framing at the level of the $\Omega \longrightarrow \Omega'$ judgment, we introduce two compatibility rules: together,

$$\frac{\Omega_1 \longrightarrow \Omega_1'}{\Omega_1\,\Omega_2 \longrightarrow \Omega_1'\,\Omega_2}\;{\longrightarrow}\textsc{c}_\textsc{l} \quad \text{and} \quad \frac{\Omega_2 \longrightarrow \Omega_2'}{\Omega_1\,\Omega_2 \longrightarrow \Omega_1\,\Omega_2'}\;{\longrightarrow}\textsc{c}_\textsc{r}$$

ensure that rewriting is compatible with concatenation of ordered contexts.[12]

[12] Because ordered contexts form a monoid, these compatibility rules are equivalent to the unified rule

$$\frac{\Omega \longrightarrow \Omega'}{\Omega_L\,\Omega\,\Omega_R \longrightarrow \Omega_L\,\Omega'\,\Omega_R}\;{\longrightarrow}\textsc{c}.$$

However, we prefer the two-rule formulation of compatibility because it better aligns with the syntactic structure of contexts.

By forming the reflexive, transitive closure of $\longrightarrow$, we may construct a multi-step rewriting relation, which we choose to write as $\Longrightarrow$.[13]

[13] Usually written as $\longrightarrow^*$, we instead chose $\Longrightarrow$ for the reflexive, transitive closure because of its similarity with process calculus notation for weak transitions, $\overset{\alpha}{\Longrightarrow}$. Our reasons will become clearer in subsequent ??.

Consistent with its [free] monoidal structure, there are two equivalent formulations of this reflexive, transitive closure: each rewriting sequence $\Omega \Longrightarrow \Omega'$ can be viewed as either a list or tree of individual rewriting steps. We prefer the list-based formulation shown in fig. 3.10 because it tends to [...] proofs by structural induction, but, on the basis of the following **??**, we allow ourselves to freely switch between the two formulations as needed.

FACT 3.10 (Transitivity of $\Longrightarrow$). *If $\Omega \Longrightarrow \Omega'$ and $\Omega' \Longrightarrow \Omega''$, then $\Omega \Longrightarrow \Omega''$.*

Proof. By induction on the structure of the first trace, $\Omega \Longrightarrow \Omega'$. □

A FEW REMARKS about these rewriting relations are in order. First, interpreting the resource decomposition rules as rewriting only confirms our preference for the nullary $\backslash$D and /D rules. The $\backslash$D′ and /D′ rules, with their $\Omega \vdash A$ premises, would be problematic as rewriting rules because they would introduce a dependence of ordered rewriting upon general provability, and the concomitant[/attendant] proof search would take ordered rewriting too far afield from traditional, syntactic[14] notions of string and multiset rewriting. [mechanical, computational]

[14] Is this the right word?

Second, multi-step rewriting is incomplete with respect to the ordered sequent calculus (**??**) because all right rules have been discarded.

FALSE CLAIM 3.11 (Completeness). *If $\Omega \vdash A$, then $\Omega \Longrightarrow A$.*

Counterexample. The sequent $A \backslash (C \,/\, B) \vdash (A \backslash C) \,/\, B$ is provable, but $A \backslash (C \,/\, B) \not\Longrightarrow (A \backslash C) \,/\, B$ even though $A\,(A \backslash (C \,/\, B))\,B \Longrightarrow C$ does hold. □

As expected from the way in which it was developed, ordered rewriting is, however, sound. Before stating and proving soundness, we must define an operation $\bullet\Omega$ that reifies an ordered context as a single proposition (see adjacent figure).

$$(\Omega_1\,\Omega_2) = (\Omega_1) \bullet (\Omega_2)$$
$$\bullet(\cdot) = \mathbf{1}$$
$$A = A$$

$$\bullet(\Omega_1\,\Omega_2) = (\bullet\Omega_1) \bullet (\bullet\Omega_2)$$
$$\bullet(\cdot) = \mathbf{1}$$
$$\bullet A = A$$

THEOREM 3.12. *If $\Omega \longrightarrow \Omega'$, then $\Omega \vdash \bullet\Omega'$. Also, if $\Omega \Longrightarrow \Omega'$, then $\Omega \vdash \bullet\Omega'$.*

Figure 3.11: From ordered contexts to propositions

THEOREM 3.13 (Soundness). *If $\Omega \longrightarrow \Omega'$, then $\Omega \vdash \bullet\Omega'$. Also, if $\Omega \Longrightarrow \Omega'$, then $\Omega \vdash \bullet\Omega'$.*

Proof. By induction on the structure of the given step or trace. □

Last, notice that every rewriting step, $\Omega \longrightarrow \Omega'$, strictly decreases the number of logical connectives that occur in the ordered context. More formally, let $|\Omega|$ be a measure of the number of logical connectives that occur in Ω, as defined in the adjacent figure. We may then prove the following ??.

$$|\Omega_1\,\Omega_2| = |\Omega_1| + |\Omega_2|$$
$$|\cdot| = 0$$
$$|A \star B| = 1 + |A| + |B| \quad \text{if } \star = \bullet, \&, \backslash, /, \text{ or } \oplus$$
$$|A| = 1 \quad \text{if } A = \alpha, \mathbf{1}, \top, \text{ or } \mathbf{0}$$

Figure 3.12: A measure of the number of logical connectives within an ordered context

FACT 3.14. *If $\Omega \longrightarrow \Omega'$, then $|\Omega| > |\Omega'|$. Also, if $\Omega \Longrightarrow \Omega'$, then $|\Omega| \geq |\Omega'|$.*

Proof. By induction on the structure of the rewriting step. □

On the basis of this ??, we will frequently refer to the rewriting relation, $\longrightarrow$, as reduction.

3.8

3.8.1 *Binary counters*

$$
\begin{aligned}
\hat{e}\,\underleftarrow{i} &\Longrightarrow \hat{e}\,\hat{b}_1 \\
\hat{b}_0\,\underleftarrow{i} &\Longrightarrow \hat{b}_1 \\
\hat{b}_1\,\underleftarrow{i} &\Longrightarrow \underleftarrow{i}\,\hat{b}_0
\end{aligned}
$$

$$
\begin{aligned}
\hat{e}\,\underleftarrow{d} &\Longrightarrow \underrightarrow{z} \\
\hat{b}_0\,\underleftarrow{d} &\Longrightarrow \underleftarrow{d}\,\hat{b}'_0 \\
\hat{b}_1\,\underleftarrow{d} &\Longrightarrow \hat{b}_0\,\underrightarrow{s} \\
\underrightarrow{z}\,\hat{b}'_0 &\Longrightarrow \underrightarrow{z} \\
\underrightarrow{s}\,\hat{b}'_0 &\Longrightarrow \hat{b}_1\,\underrightarrow{s}
\end{aligned}
$$

$$
\frac{}{e \mathrel{\mathcal{R}} \hat{e}}
\qquad
\frac{\Omega \mathrel{\mathcal{R}} \Omega'}{\Omega\, b_0 \mathrel{\mathcal{R}} \Omega'\, \hat{b}_0}
\qquad
\frac{\Omega \mathrel{\mathcal{R}} \Omega'}{\Omega\, b_1 \mathrel{\mathcal{R}} \Omega'\, \hat{b}_1}
\qquad
\frac{\Omega \mathrel{\mathcal{R}} \Omega'}{\Omega\, i \mathrel{\mathcal{R}} \Omega'\, \underleftarrow{i}}
$$

$$
\frac{\Omega \mathrel{\mathcal{R}} \Omega'}{\Omega\, i\, b_0 \mathrel{\mathcal{R}} \Omega'\, (\underleftarrow{i} \bullet b_0)}
$$

3.8.2 *Automata*

$$
\frac{\Omega \mathrel{\mathcal{R}} \Omega'}{a\,\Omega \mathrel{\mathcal{R}} \underrightarrow{a}\,\Omega'}
\qquad
\frac{}{q \mathrel{\mathcal{R}} \hat{q}}
$$

$$
\frac{q \sim q'}{q \mathrel{\mathcal{R}} \hat{q}'}
$$

3.9 *Ordered rewriting for specifications*

3.9.1 *Deterministic finite automata*

$$
\frac{}{a\,q \longrightarrow q'_a}
$$

for each DFA transition $q \xrightarrow{a} q'_a$, and

$$
\frac{}{\epsilon\,q \longrightarrow F(q)}
$$

for each state q, where $F(q) = \mathbf{1}$ if q is a final state and $F(q) = \top$ otherwise.

- $q \xrightarrow{a} q'_a$ if, and only if, $a\,q \longrightarrow q'_a$; and
- $q \in F$ if, and only if, $\epsilon\,q \longrightarrow \mathbf{1}$.

3.9.2 *Nondeterministic finite automata*

Equally straightforward

3.9.3 *Binary counters*

Values

AN INCREMENT OPERATION To use ordered rewriting to specify [...]

$$\overline{e\;i \longrightarrow e\;b_1} \qquad \overline{b_0\;i \longrightarrow b_1} \qquad \overline{b_1\;i \longrightarrow i\;b_0}$$

Small- and big-step adequacy theorems for increments

- Slightly simplified because there is no •

A DECREMENT OPERATION

$$\overline{e\;d \longrightarrow z} \qquad \overline{b_0\;d \longrightarrow d\;b_0'} \qquad \overline{b_1\;d \longrightarrow b_0\;s} \qquad \overline{z\;b_0' \longrightarrow z} \qquad \overline{s\;b_0' \longrightarrow b_1\;s}$$

- Significantly simpler because there is no &, so we don't need (weak) focusing

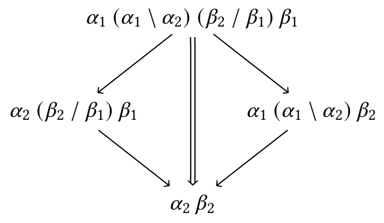
3.10

3.10.1 *Concurrency in ordered rewriting*

As an example of multi-step rewriting, observe that

$$\alpha_1\;(\alpha_1 \setminus \alpha_2)\;(\beta_2 \mathbin{/} \beta_1)\;\beta_1 \Longrightarrow \alpha_2\;\beta_2.$$

In fact, as shown in the adjacent figure, two sequences witness this rewriting: either the initial state's left half, $\alpha_1\;(\alpha_1 \setminus \alpha_2)$, is first rewritten to α_2 and then its right half, $(\beta_2 \mathbin{/} \beta_1)\;\beta_1$, is rewritten to β_2; or *vice versa*, the right half is first rewritten to β_2 and then the left half is rewritten to α_2.



$$\begin{tikzcd}
 & \alpha_1\;(\alpha_1 \setminus \alpha_2)\;(\beta_2 \mathbin{/} \beta_1)\;\beta_1 \arrow[dl] \arrow[dr] \arrow[dd, Rightarrow] & \\
\alpha_2\;(\beta_2 \mathbin{/} \beta_1)\;\beta_1 \arrow[dr] & & \alpha_1\;(\alpha_1 \setminus \alpha_2)\;\beta_2 \arrow[dl] \\
 & \alpha_2\;\beta_2 &
\end{tikzcd}$$

Figure 3.13: An example of concurrent ordered rewriting

Notice that these two sequences differ only in how non-overlapping, and therefore independent, rewritings of the initial state's two halves are interleaved. Consequently, the two sequences can be – and indeed should be – considered essentially equivalent. The details of how the small-step rewrites are interleaved are irrelevant, so that conceptually, at least, only the big-step trace from $\alpha_1\;(\alpha_1 \setminus \alpha_2)\;(\beta_2 \mathbin{/} \beta_1)\;\beta_1$ to $\alpha_2\;\beta_2$ remains.

More generally, this idea that the interleaving of independent actions is irrelevant is known as *concurrent equality*,[15] and it forms the basis of concurrency.[16] Concurrent equality also endows traces $\Omega \Longrightarrow \Omega'$ with a free partially commutative monoid structure, *i.e.*, traces form a trace monoid.

[15] **Watkins+:CMU02**.

[16] ??.

Because the two indivisual rewriting steps are independent, Nothing about the final result, $\alpha_2\;\beta_2$, suggests which rewriting sequence

The rewritings of the left and right halves are not overlapping and therefore independent. Their independence means that we may view the two rewriting sequences as equivalent – the two rewriting steps

More generally, any non-overlapping rewritings are independent and may occur in any order. Rewriting sequences that differ only by the order in which independent rewritings occur may be seen as equivalent sequences. This equivalence relation, *concurrent equality*[17]

because the left half of Ω may be rewritten by the $\backslash\textsc{d}$ rule to α_2, and then the right half may be rewritten to β_2:

[17] **Watkins+:CMU02**.

3.10.2 *Other properties of ordered rewriting*

As the relation $\Longrightarrow$ forms a rewriting system, we may evaluate it along several standard dimensions: termination, confluence.

Because each rewriting step reduces the number of logical connectives present in the state (?? 3.14), ordered rewriting is terminating.

THEOREM 3.15 (Termination). *No infinite rewriting sequence* $\Omega_0 \longrightarrow \Omega_1 \longrightarrow \Omega_2 \longrightarrow \cdots$ *exists.*

Proof. Beginning from state Ω_0, some state Ω_i will eventually be reached such that either: $\Omega_i \not\longrightarrow$; or $|\Omega_i| = 0$ and $\Omega_i \longrightarrow \Omega_{i+1}$. In the latter case, ?? 3.14 establishes $|\Omega_{i+1}| < 0$, which is impossible. □

Although terminating, ordered rewriting is not confluent. Confluence requires that all states with a common ancestor, *i.e.*, states Ω'_1 and Ω'_2 such that $\Omega'_1 \Longleftarrow\Longrightarrow \Omega'_2$, be joinable, *i.e.*, $\Omega'_1 \Longrightarrow\Longleftarrow \Omega'_2$. Because ordered rewriting is directional[18] and the relation $\Longrightarrow$ is not symmetric, some nondeterministic choices are irreversible.

[18] Is this phrasing correct?

FALSE CLAIM 3.16 (Confluence). *If* $\Omega'_1 \Longleftarrow\Longrightarrow \Omega'_2$*, then* $\Omega'_1 \Longrightarrow\Longleftarrow \Omega'_2$.

Counterexamples. Consider the state $\alpha \& \beta$. By the rewriting rules for additive conjunction, $\alpha \longleftarrow \alpha \& \beta \longrightarrow \beta$, and hence $\alpha \Longleftarrow \alpha \& \beta \Longrightarrow \beta$. However, being atoms, neither α nor β reduces. And $\alpha \neq \beta$, so $\alpha \Longrightarrow\Longleftarrow \beta$ does *not* hold.

Even in the $\&$-free fragment, ordered rewriting is not confluent. For example,

$$\not\longleftarrow \beta_1\,(\alpha \setminus \beta_2) \Longleftarrow (\beta_1 \,/\, \alpha)\,\alpha\,(\alpha \setminus \beta_2) \Longrightarrow (\beta_1 \,/\, \alpha)\,\beta_2 \not\longrightarrow .$$ □

3.11 *Unbounded ordered rewriting*

[19]

[19] **Aranda+:FMCO06**.

Although a seemingly pleasant property, termination (theorem 2.7) significantly limits the expressiveness of ordered rewriting. For example, without unbounded rewriting, we cannot even give ordered rewriting specifications of producer-consumer systems or finite automata.

As the proof of termination shows, rewriting is bounded precisely because states consist of finitely many finite propositions. To admit unbounded rewriting, we therefore choose to permit infinite propositions in the form of mutually recursive definitions, $\alpha \triangleq A$. These definitions are collected into a signature, $\Sigma = (\alpha_i \triangleq A_i)_i$, which indexes the rewriting relations: $\longrightarrow_\Sigma$ and $\Longrightarrow_\Sigma$.[20]

[20] We frequently elide the indexing signature, as it is usually clear from context.

To rule out definitions like $\alpha \triangleq \alpha$ that do not correspond to sensible infinite propositions, we also require that definitions be *contractive*[21] – *i.e.*, that the body of each recursive definition begin with a logical connective at the top level.

[21] **Gay+Hole:AI05**.

By analogy with recursive types from functional programming,[22] we must now decide whether to treat definitions *iso*recursively or *equi*recursively. Under an equirecursive interpretation, definitions $\alpha \triangleq A$ may be silently unrolled or rolled at will; in other words, α is literally *equal* to its unrolling, A. In contrast, under an isorecursive interpretation, unrolling a recursively defined proposition would count as an explicit step of rewriting – $\alpha \longrightarrow A$, for example.

[22] **??**.

We choose to interpret definitions equirecursively because the equirecursive treatment, with its generous notion of equality, helps to minimize the overhead of recursively defined propositions. As a simple example, under the equirecursive definition $\beta \triangleq a \setminus \beta$, we have the trace

$$a\,a\,\beta = a\,a\,(a \setminus \beta) \longrightarrow a\,\beta = a\,(a \setminus \beta) \longrightarrow \beta$$

or, more concisely, $a\,a\,\beta \longrightarrow a\,\beta \longrightarrow \beta$. Had we chosen an isorecursive treatment of the same definition, we would have only the more laborious

$$a\,a\,\beta \longrightarrow a\,a\,(a \setminus \beta) \longrightarrow a\,\beta \longrightarrow a\,(a \setminus \beta) \longrightarrow \beta.$$

3.11.1 *Replication*

In Milner's development of the π-calculus, there are two avenues to unbounded process behavior: recursive process definitions and replication.

3.12 *Extended examples of ordered rewriting*

3.12.1 *Encoding deterministic finite automata*

As an extended example, we will use ordered rewriting to specify how a DFA processes its input. Given a DFA $\mathcal{A} = (Q, ?, F)$ over an input alphabet Σ, the idea is to encode each state, $q \in Q$, as an ordered proposition, $\hat{q}$, in such a way that the DFA's operational semantics are adequately captured by [ordered] rewriting. [23]

[23] [In general, the behavior of a DFA state is recursive, so the proposition $\hat{q}$ will be recursively defined.]

Ideally, DFA transitions $q \xrightarrow{a} q'_a$ would be in bijective correspondence with rewriting steps $a\,\hat{q} \longrightarrow \hat{q}'_a$, where each input symbol a is encoded by a matching [propositional] atom. We will return to the possibility of this kind of tight correspondence in **??**, but, for now, we will content ourselves with a correspondence with traces rather than individual steps, adopting the following desiderata:

- $q \xrightarrow{a} q'_a$ if, and only if, $a\,\hat{q} \Longrightarrow \hat{q}'_a$, for all input symbols $a \in \Sigma$.
- $q \in F$ if, and only if, $\epsilon\,\hat{q} \Longrightarrow \mathbf{1}$, where the atom ϵ functions as an end-of-word marker.

Given the reversal (anti-)homomorphism from finite words to ordered contexts defined in the adjacent figure, the first desideratum is subsumed by a third:

- $q \xrightarrow{w} q'$ if, and only if, $w^{\mathsf{R}}\,\hat{q} \Longrightarrow \hat{q}'$, for all finite words $w \in \Sigma^*$.

$$(w_1\,w_2)^{\mathsf{R}} = w_2^{\mathsf{R}}\,w_1^{\mathsf{R}}$$
$$\epsilon^{\mathsf{R}} = \cdot$$
$$a^{\mathsf{R}} = a$$

Figure 3.14: An (anti-)homomorphism for reversal of finite words to ordered contexts

From these desiderata [and the observation that DFAs' graphs frequently[24] contain cycles], we arrive at the following encoding, in which each state is encoded by one of a collection of mutually recursive definitions:[25]

[24] Actually, there is always at least one cycle in a well-formed DFA.

[25] q'_a, using function or relation?

$$\hat{q} \triangleq \bigl(\&_{a\in\Sigma}(a \setminus \hat{q}'_a)\bigr) \mathbin{\&} \bigl(\epsilon \setminus \hat{F}(q)\bigr)$$

where

$$q \xrightarrow{a} q'_a,\ \text{for all input symbols } a \in \Sigma, \quad \text{and} \quad \hat{F}(q) = \begin{cases} \mathbf{1} & \text{if } q \in F \\ \top & \text{if } q \notin F\,. \end{cases}$$

Just as each state q has exactly one successor for each input symbol a, its encoding, $\hat{q}$, has exactly one input clause, $(a \setminus \cdots)$, for each symbol a.

For a concrete instance of this encoding, recall from **??** the DFA (repeated in the adjacent figure) that accepts exactly those words, over the alphabet $\Sigma = \{a, b\}$, that end with b; that DFA is encoded by the following definitions:

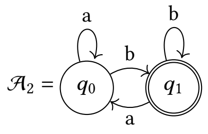


Figure 3.15: A DFA that accepts, from state q_0, exactly those words that end with b. (Repeated from **??**.)

$$\hat{q}_0 \triangleq (a \setminus \hat{q}_0) \mathbin{\&} (b \setminus \hat{q}_1) \mathbin{\&} (\epsilon \setminus \top)$$
$$\hat{q}_1 \triangleq (a \setminus \hat{q}_0) \mathbin{\&} (b \setminus \hat{q}_1) \mathbin{\&} (\epsilon \setminus \mathbf{1})$$

Indeed, just as the DFA has a transition $q_0 \xrightarrow{b} q_1$, its encoding admits a trace

$$b\,\hat{q}_0 = b\,\bigl((a \setminus \hat{q}_0) \mathbin{\&} (b \setminus \hat{q}_1) \mathbin{\&} (\epsilon \setminus \top)\bigr) \Longrightarrow b\,(b \setminus \hat{q}_1) \longrightarrow \hat{q}_1\,.$$

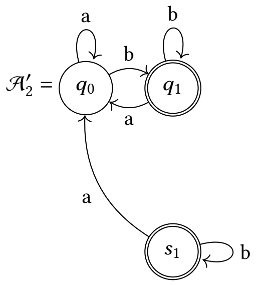
And, just as q_1 is an accepting state, its encoding also admits a trace

$$\epsilon\,\hat{q}_1 = \epsilon\,\bigl((a \setminus \hat{q}_0) \mathbin{\&} (b \setminus \hat{q}_1) \mathbin{\&} (\epsilon \setminus \mathbf{1})\bigr) \Longrightarrow \epsilon\,(\epsilon \setminus \mathbf{1}) \longrightarrow \mathbf{1}\,.$$

More generally, this encoding is complete, in the sense that it simulates all DFA transitions: $q \xrightarrow{a} q'$ implies $a\,\hat{q} \Longrightarrow \hat{q}'$, for all states q and q' and input symbols a.

However, the converse does not hold – the encoding is unsound because there are rewritings that cannot be simulated by a DFA transition.

False claim 3.17. *Let $\mathcal{A} = (Q, \longrightarrow, F)$ be a DFA over the input alphabet Σ. Then $a\,\hat{q} \Longrightarrow \hat{q}'$ implies $q \xrightarrow{a} q'$, for all input symbols $a \in \Sigma$.*



$$\hat{q}_0 \triangleq (a \setminus \hat{q}_0) \mathbin{\&} (b \setminus \hat{q}_1) \mathbin{\&} (\epsilon \setminus \top)$$
$$\hat{q}_1 \triangleq (a \setminus \hat{q}_0) \mathbin{\&} (b \setminus \hat{q}_1) \mathbin{\&} (\epsilon \setminus \mathbf{1})$$
$$\hat{s}_1 \triangleq (a \setminus \hat{q}_0) \mathbin{\&} (b \setminus \hat{s}_1) \mathbin{\&} (\epsilon \setminus \mathbf{1})$$

Figure 3.16: fig:ordered-rewriting:dfa-counterexample:dfa A slightly modified version of the DFA from fig. 3.15; and fig:ordered-rewriting:dfa-counterexample:encoding its encoding

Counterexample. Consider the DFA and encoding shown in the adjacent figure; it is the same DFA as shown in fig. 3.15, but with one added state, s_1, that is unreachable from q_0 and q_1. Notice that, as a coinductive consequence of the equirecursive treatment of definitions, $\hat{q}_1 = \hat{s}_1$. Previously, we saw that $b\,\hat{q}_0 \Longrightarrow \hat{q}_1$; hence $b\,\hat{q}_0 \Longrightarrow \hat{s}_1$. However, the DFA has no $q_0 \xrightarrow{b} s_1$ transition (because $q_1 \neq s_1$), and so this encoding is unsound with respect to the operational semantics of DFAs. □

As this counterexample shows, the lack of adequacy stems from attempting to use an encoding that is not injective – here, $q_1 \neq s_1$ even though $\hat{q}_1 = \hat{s}_1$. In other words, eqality of state encodings is a coarser eqvivalence than equality of the states themselves.

One possible remedy for this lack of adequacy might be to revise the encoding to have a stronger nominal character. By tagging each state's encoding with an atom that is unique to that state, we can make the encoding manifestly injective. For instance, given the pairwise distinct atoms $\{q \mid q \in F\}$ and $\{\bar{q} \mid q \in Q - F\}$ to tag final and non-final states, respectively, we could define an alternative encoding, $\check{q}$:

$$\check{q} \triangleq \big(\&_{a\in\Sigma}(a \setminus \check{q}'_a)\big) \mathbin{\&} \big(\epsilon \setminus \check{F}(q)\big)$$

where

$$q \xrightarrow{a} q'_a, \text{ for all input symbols } a \in \Sigma, \quad \text{and} \quad \check{F}(q) = \begin{cases} q & \text{if } q \in F \\ \bar{q} & \text{if } q \notin F\,. \end{cases}$$

Under this alternative encoding, the states q_1 and s_1 of fig. 3.16 are no longer a counterexample to injectivity: Because q_1 and s_1 are distinct states, they correspond to distinct tags, and so $\check{q}_1 \neq \check{s}_1$.

Although such a solution is certainly possible, it seems unsatisfyingly ad hoc. A closer examination of the preceding counterexample reveals that the states q_1 and s_1, while not equal, are in fact bisimilar (**??**). In other words, although the encoding is not, strictly speaking, injective, it is injective *up to bisimilarity*: $\hat{q} = \hat{s}$ implies $q \sim s$. This suggests a more elegant solution to the apparent lack of adequacy: the encoding's adequacy should be judged up to DFA bisimilarity.

THEOREM 3.19 (DFA adequacy up to bisimilarity). *Let $\mathcal{A} = (Q, ?, F)$ be a DFA over the input alphabet Σ. Then, for all states q, q', and s:*

1. *$q \sim s$ if, and only if, $\hat{q} = \hat{s}$.*
2. *$q \mathrel{\sim\!\xrightarrow{a}\!\sim} q'$ if, and only if, $a\,\hat{q} \Longrightarrow \hat{q}'$, for all input symbols $a \in \Sigma$. More generally, $q \mathrel{\sim\!\xrightarrow{w}\!\sim} q'$ if, and only if, $w^{\mathrm{R}}\,\hat{q} \Longrightarrow \hat{q}'$, for all finite words $w \in \Sigma^*$.*
3. *$q \in F$ if, and only if, $\epsilon\,\hat{q} \Longrightarrow \mathbf{1}$.*

Before proving this theorem, we must first prove a lemma: the only traces from one state's encoding to another's are the trivial traces.

LEMMA 3.18. *Let $\mathcal{A} = (Q, ?, F)$ be a DFA over the input alphabet Σ. For all states q and s, if $\hat{q} \Longrightarrow \hat{s}$, then $\hat{q} = \hat{s}$.*

Proof. Assume that a trace $\hat{q} \Longrightarrow \hat{s}$ exists. If the trace is trivial, then $\hat{q} = \hat{s}$ is immediate. Otherwise, the trace is nontrivial and consists of a strictly positive number of rewriting steps. By inversion, those rewriting steps drop one or more conjuncts from $\hat{q}$ to form $\hat{s}$. Every DFA state's encoding contains exactly $|\Sigma| + 1$ conjuncts – one for each input symbol a and one for the end-of-word marker, ϵ. If even one conjunct is dropped from $\hat{q}$, not enough conjuncts will remain to form $\hat{s}$. Thus, a nontrivial trace $\hat{q} \Longrightarrow \hat{s}$ cannot exist. □

It is important to differentiate this lemma from the false claim that a state's encoding can take no rewriting steps. There certainly exist nontrivial traces from $\hat{q}$, but they do not arrive at the encoding of any state.

With this lemma now in hand, we can proceed to proving adequacy up to bisimilarity.

THEOREM 3.19 (DFA adequacy up to bisimilarity). *Let $\mathcal{A} = (Q, ?, F)$ be a DFA over the input alphabet Σ. Then, for all states q, q', and s:*

1. *$q \sim s$ if, and only if, $\hat{q} = \hat{s}$.*
2. *$q \sim\!\xrightarrow{a}\!\sim q'$ if, and only if, $a\,\hat{q} \Longrightarrow \hat{q}'$, for all input symbols $a \in \Sigma$. More generally, $q \sim\!\xrightarrow{w}\!\sim q'$ if, and only if, $w^{\mathrm{R}}\,\hat{q} \Longrightarrow \hat{q}'$, for all finite words $w \in \Sigma^*$.*
3. *$q \in F$ if, and only if, $\epsilon\,\hat{q} \Longrightarrow \mathbf{1}$.*

Proof. Each part is proved in turn. The proof of part 2 depends on the proof of part 1.

1. We shall show that bisimilarity coincides with equality of encodings, proving each direction separately.
   - To prove that bisimilar DFA states have equal encodings – *i.e.*, that $q \sim s$ implies $\hat{q} = \hat{s}$ – a fairly straightforward proof by coinduction suffices.

     Let q and s be bisimilar states. By the definition of bisimilarity (**??**), two properties hold:
     – For all input symbols a, the unique a-successors of q and s are also bisimilar.
     – States q and s have matching finalities – *i.e.*, $q \in F$ if and only if $s \in F$.

     Applying the coinductive hypothesis to the former property, we may deduce that, for all symbols a, the a-successors of q and s also have equal encodings. From the latter property, it follows that $\hat{F}(q) = \hat{F}(s)$. Because definitions are interpreted equirecursively, these equalities together imply that q and s themselves have equal encodings.
   - To prove the converse – that states with equal encodings are bisimilar – we will show that the relation $\mathcal{R} = \{(q, s) \mid \hat{q} = \hat{s}\}$, which relates states if they have equal encodings, is a bisimulation and is therefore included in bisimilarity.
     – The relation $\mathcal{R}$ is symmetric.
     – We must show that $\mathcal{R}$-related states have $\mathcal{R}$-related a-successors, for all input symbols a.

       Let q and s be $\mathcal{R}$-related states. Being $\mathcal{R}$-related, q and s have equal encodings; because definitions are interpreted equirecursively, the unrollings of those encodings are also equal. By definition of the encoding, it follows that, for each input symbol a, the unique a-successors of q and s have equal encodings. Therefore, for each a, the a-successors of q and s are themselves $\mathcal{R}$-related.

– We must show that $\mathcal{R}$-related states have matching finalities. Let q and s be $\mathcal{R}$-related states, with q a final state. Being $\mathcal{R}$-related, q and s have equal encodings; because definitions are interpreted equirecursively, the unrollings of those encodings are also equal. It follows that $\hat{F}(q) = \hat{F}(s)$, and so s is also a final state.

2. We would like to prove that $q \sim\!\xrightarrow{a}\!\sim q'$ if, and only if, $a\,\hat{q} \Longrightarrow \hat{q}'$, or, more generally, that $q \sim\!\xrightarrow{w}\!\sim q'$ if, and only if, $w^{\mathrm{R}}\,\hat{q} \Longrightarrow \hat{q}'$. Because bisimilar states have equal encodings (part 1) and bisimilarity is reflexive (??), it suffices to show two stronger statements: (a) that $q \xrightarrow{w} q'$ implies $w^{\mathrm{R}}\,\hat{q} \Longrightarrow \hat{q}'$; and (b) that $w^{\mathrm{R}}\,\hat{q} \Longrightarrow \hat{q}'$ implies $q \xrightarrow{w}\!\sim q'$. We prove these in turn.

   (a) We shall prove that $q \xrightarrow{w} q'$ implies $w^{\mathrm{R}}\,\hat{q} \Longrightarrow \hat{q}'$ by induction over the structure of word w.

   - Consider the case of the empty word, ϵ; we must show that $q = q'$ implies $\hat{q} \Longrightarrow \hat{q}'$. Because the encoding is a function, this is immediate.
   - Consider the case of a nonempty word, $a\,w$; we must show that $q \xrightarrow{a}\xrightarrow{w} q'$ implies $w^{\mathrm{R}}\,a\,\hat{q} \Longrightarrow \hat{q}'$. Let q'_a be an a-successor of state q that is itself w-succeeded by state q'. There exists, by definition of the encoding, a trace

$$w^{\mathrm{R}}\,a\,\hat{q} \Longrightarrow w^{\mathrm{R}}\,a\,(a \setminus \hat{q}'_a) \longrightarrow w^{\mathrm{R}}\,\hat{q}'_a \Longrightarrow \hat{q}'\,,$$

   with the trace's tail justified by an appeal to the inductive hypothesis.

   (b) We shall prove that $w^{\mathrm{R}}\,\hat{q} \Longrightarrow \hat{q}'$ implies $q \xrightarrow{w}\!\sim q'$ by induction over the structure of word w.

   - Consider the case of the empty word, ϵ; we must show that $\hat{q} \Longrightarrow \hat{q}'$ implies $q \sim q'$. By lemma 3.18, $\hat{q} \Longrightarrow \hat{q}'$ implies that q and q' have equal encodings. Part 1 can then be used to establish that q and q' are bisimilar.
   - Consider the case of a nonempty word, $a\,w$; we must show that $w^{\mathrm{R}}\,a\,\hat{q} \Longrightarrow \hat{q}'$ implies $q \xrightarrow{a}\xrightarrow{w}\!\sim q'$. By inversion[26], the given trace can only begin by inputting a:

$$w^{\mathrm{R}}\,a\,\hat{q} \Longrightarrow w^{\mathrm{R}}\,a\,(a \setminus \hat{q}'_a) \longrightarrow w^{\mathrm{R}}\,\hat{q}'_a \Longrightarrow \hat{q}'\,,$$

   where q'_a is an a-successor of state q. An appeal to the inductive hypothesis on the trace's tail yields $q'_a \xrightarrow{w}\!\sim q'$, and so the DFA admits $q \xrightarrow{a}\xrightarrow{w}\!\sim q'$, as required.

[26] Is this enough justification?

3. We shall prove that the final states are exactly those states q such that $\epsilon\,\hat{q} \Longrightarrow \mathbf{1}$.

- Let q be a final state; accordingly, $\hat{F}(q) = \mathbf{1}$. There exists, by definition of the encoding, a trace

$$\epsilon\,\hat{q} \Longrightarrow \epsilon\,(\epsilon \setminus \hat{F}(q)) \longrightarrow \hat{F}(q) = \mathbf{1}\,.$$

- Assume that a trace $\epsilon\,\hat{q} \Longrightarrow \mathbf{1}$ exists. By inversion[27], this trace can only begin by inputting ϵ:

$$\epsilon\,\hat{q} \Longrightarrow \epsilon\,(\epsilon \setminus \hat{F}(q)) \longrightarrow \hat{F}(q) \Longrightarrow \mathbf{1}\,.$$

The tail of this trace, $\hat{F}(q) \Longrightarrow \mathbf{1}$, can exist only if q is a final state. □

[27] Is this enough justification?

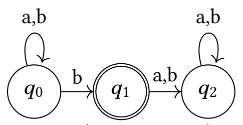
3.12.2 *Encoding nondeterministic finite automata?*

We would certainly be remiss if we did not attempt to generalize the rewriting specification of DFAs to one for their nondeterministic cousins.

Differently from DFA states, an NFA state q may have several nondeterministic successors for each input symbol a. To encode the NFA state q, all of its a-successors are collected in an alternative conjunction underneath the left-handed input of a. Thus, the encoding of an NFA state q becomes

$$\hat{q} \triangleq \Bigl(\mathop{\&}_{a\in\Sigma} \bigl(a \setminus (\&_{q'_a}\, \hat{q}'_a)\bigr)\Bigr) \mathbin{\&} \bigl(\epsilon \setminus \hat{F}(q)\bigr)\,,$$

where $\hat{F}(q)$ is defined as for DFAs.



$\hat{q}_0 \triangleq (a \setminus \hat{q}_0) \mathbin{\&} \bigl(b \setminus (\hat{q}_0 \mathbin{\&} \hat{q}_1)\bigr) \mathbin{\&} (\epsilon \setminus \top)$
$\hat{q}_1 \triangleq (a \setminus \hat{q}_2) \mathbin{\&} (b \setminus \hat{q}_2) \mathbin{\&} (\epsilon \setminus \mathbf{1})$
$\hat{q}_2 \triangleq (a \setminus \hat{q}_2) \mathbin{\&} (b \setminus \hat{q}_2) \mathbin{\&} (\epsilon \setminus \top)$

Figure 3.17: fig:ordered-rewriting:nfa-example:nfa An NFA that accepts exactly those words, over the alphabet $\Sigma = \{a, b\}$, that end with b; and fig:ordered-rewriting:nfa-example:encoding its encoding

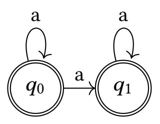
The adjacent figure recalls from **??** an NFA that accepts exactly those words, over the alphabet $\Sigma = \{a, b\}$, that end with b. Using the above encoding of NFAs, ordered rewriting does indeed simulate this NFA. For example, just as there are transitions $q_0 \xrightarrow{b} q_0$ and $q_0 \xrightarrow{b} q_1$, there are traces

$$\begin{tikzcd} & & & \hat{q}_0 \\ b\,\hat{q}_0 \Longrightarrow b\,\bigl(b \setminus (\hat{q}_0 \mathbin{\&} \hat{q}_1)\bigr) \arrow[r] & \hat{q}_0 \mathbin{\&} \hat{q}_1 \arrow[urr] \arrow[drr] & & \\ & & & \hat{q}_1 \end{tikzcd}$$

Unfortunately, while it does simulate NFA behavior, this encoding is not adequate. Like DFA states, NFA states that have equal encodings are bisimilar. However, for NFAs, the converse does not hold: bisimilar states do not necessarily have equal encodings.

False claim 3.20. *Let $\mathcal{A} = (Q, ?, F)$ be an NFA over input alphabet Σ. Then $q \sim s$ implies $\hat{q} = \hat{s}$, for all states q and s.*

Counterexample. Consider the NFA and encoding depicted in the adjacent figure. It is easy to verify that the relation $\{q_1\} \times \{q_0, q_1\}$ is a bisimulation; in particular, q_1 simulates the $q_0 \xrightarrow{a} q_1$ transition by its self-loop, $q_1 \xrightarrow{a} q_1$. Hence, q_0 and s_0 are bisimilar. It is equally easy to verify, by unrolling the definitions used in the encoding, that $\hat{q}_0 \neq \hat{s}_0$. □



$\hat{q}_0 \triangleq \bigl(a \setminus (\hat{q}_0 \mathbin{\&} \hat{q}_1)\bigr) \mathbin{\&} (\epsilon \setminus \mathbf{1})$
$\hat{q}_1 \triangleq (a \setminus \hat{q}_1) \mathbin{\&} (\epsilon \setminus \mathbf{1})$

Figure 3.18: An NFA that accepts all finite words over the alphabet $\Sigma = \{a\}$

For DFAs, bisimilar states do have equal encodings because the inherent determinism DFA bisimilarity is a rather fine-grained equivalence. Because each DFA state has exactly one successor for each input symbol The additional flexibility entailed by nondeterminism

Once again, it would be possible to construct an adequate encoding, by tagging each state with a unique atom.

For the moment, we will put aside the question of an adequate encoding of NFAs.

3.12.3 *Binary representation of natural numbers*

As a further example of ordered rewriting, consider a rewriting specification of binary counters: binary representations of natural numbers equipped with increment and decrement operations.

BINARY REPRESENTATIONS In this setting, we represent a natural number in binary by an ordered context that consists of a big-endian sequence of atoms b_0 and b_1, prefixed by the atom e; leading b_0s are permitted. For example, both $\Omega = e\,b_1$ and $\Omega' = e\,b_0\,b_1$ are valid binary representations of the natural number 1.

To be more precise, we inductively define a relation, $\approx_\text{v}$, that assigns to each binary representation a unique natural number denotation. When $\Omega \approx_\text{v} n$, we say that Ω denotes, or represents, natural number n in binary.

$$\frac{}{e \approx_\text{v} 0}\;e\text{-v} \qquad \frac{\Omega \approx_\text{v} n}{\Omega\,b_0 \approx_\text{v} 2n}\;b_0\text{-v} \qquad \frac{\Omega \approx_\text{v} n}{\Omega\,b_1 \approx_\text{v} 2n+1}\;b_1\text{-v}$$

Besides providing a denotational semantics of binary numbers, the $\approx_\text{v}$ relation also serves to implicitly characterize the well-formed binary numbers as those ordered contexts Ω that form the relation's domain of definition.[28]

[28] Alternatively, the well-formed binary numbers could be described more explicitly by the grammar

$$\Omega ::= e \mid \Omega\,b_0 \mid \Omega\,b_1\,.$$

These properties[29] of the $\approx_\text{v}$ relation are proved as the following adequacy theorem.

[29] which properties?

THEOREM 3.21 (Adequacy of binary representations).

Functional *For every binary number Ω, there exists a unique natural number n such that $\Omega \approx_\text{v} n$.*

Surjectivity *For every natural number n, there exists a binary number Ω such that $\Omega \approx_\text{v} n$.*

Value *If $\Omega \approx_\text{v} n$, then $\Omega \not\longrightarrow$.*

Proof. The three claims may be proved by induction over the structure of Ω, and by induction on n, respectively. □

Notice that the above e-v and b_0-v rules overlap when the denotation[30] is 0, giving rise to the leading b_0s that make the $\approx_\text{v}$ relation surjective: for example, both $e\,b_1 \approx_\text{v} 1$ and $e\,b_0\,b_1 \approx_\text{v} 1$ hold. However, if the rule for b_0 is restricted to *nonzero* even numbers, then each natural number has a unique, canonical representation that is free of leading b_0s.[31]

[30] represented natural number?

[31] A restriction of the b_0 rule to nonzero even numbers is:

$$\frac{\Omega \approx_\text{v} n \quad (n > 0)}{\Omega\,b_0 \approx_\text{v} 2n}\,.$$

The leading-b_0-free representations could alternatively be seen as the canonical representatives of the equivalence classes induced by the equivalence relation among binary numbers that have the same denotation: $\Omega \equiv \Omega'$ if $\Omega \approx_\text{v} n$ and $\Omega' \approx_\text{v} n$ for some n.

AN INCREMENT OPERATION To use ordered rewriting to describe an increment operation on binary representations, we introduce a new, uninterpreted atom i that will serve as an increment instruction.

Given a binary number Ω that represents n, we may append i to form a computational state, $\Omega\, i$. For i to adequately represent the increment operation, the state $\Omega\, i$ must meet two conditions, captured by the following global desiderata:

THEOREM 3.22. *Let Ω be a binary representation of n. Then:*

- some *computation from $\Omega\, i$ results in a binary representation of $n + 1$ – that is, $\Omega\, i \Longrightarrow\approx_V n + 1$; and*
- any *computation from $\Omega\, i$ results in a binary representation of $n + 1$ – that is, $\Omega\, i \Longrightarrow\approx_V n'$ only if $n' = n + 1$.*[32]

[32] Compare "If $\Omega\, i \Longrightarrow \Omega'$, then $\Omega' \Longrightarrow\approx_V n + 1$."

For example, because $e\, b_1$ denotes 1, a computation $e\, b_1\, i \Longrightarrow\approx_V 2$ must exist; moreover, every computation $e\, b_1\, i \Longrightarrow\approx_V n'$ must satisfy $n' = 2$.

TO IMPLEMENT THESE global desiderata locally, the previously uninterpreted atoms e, b_0, and b_1 are now given mutually recursive definitions that describe how they may be rewritten when the increment instruction, i, is encountered.

$e \triangleq e \bullet b_1 \;/\; i$ To increment e, append b_1 as a new most[33] significant bit, resulting in $e\, b_1$; the rewriting sequence $e\, i \longrightarrow e \bullet b_1 \longrightarrow e\, b_1$ is entailed by this definition.

[33] or least?

$b_0 \triangleq b_1 \;/\; i$ To increment a binary number ending in b_0, flip that bit to b_1; the entailed rewriting step is $\Omega\, b_0\, i \longrightarrow \Omega\, b_1$.

$b_1 \triangleq i \bullet b_0 \;/\; i$ To increment a binary number ending in b_1, flip that bit to b_0 and carry the increment over to the more significant bits; the entailed rewriting sequence is $\Omega\, b_1\, i \longrightarrow \Omega\, (i \bullet b_0) \longrightarrow \Omega\, i\, b_0$.

Comfortingly, $1 + 1 = 2$: that is, a computation $e\, b_1\, i \Longrightarrow e\, i\, b_0 \Longrightarrow e\, b_1\, b_0$ indeed exists.

IT SHOULD ALSO be possible to permit several increments at once, such as in $e\, b_1\, i\, i$. We could, of course, handle the increments sequentially from left to right, fully computing a binary value before moving on to the subsequent increment:

$$e\, b_1\, i\, i \Longrightarrow e\, b_1\, b_0\, i \longrightarrow e\, b_1\, b_1\,.$$

However, a strictly sequential treatment of increments would be rather disappointing. Because the ordered rewriting framework[34] is inherently concurrent, a truly concurrent treatment of multiple increments would be far more satisfying.

[34] wc?

For example, consider the several computations of $(1 + 1) + 1 = 3$ from $e\,b_1\,i\,i$:

$$\begin{tikzcd}
 & & e\,b_1\,b_0\,i \arrow[dr] & \\
e\,b_1\,i\,i \arrow[r, Rightarrow] & e\,i\,b_0\,i \arrow[ur, Rightarrow] \arrow[dr] \arrow[rr, Rightarrow, dashed] & & e\,b_1\,b_1 \\
 & & e\,i\,b_1 \arrow[ur, Rightarrow] &
\end{tikzcd}$$

In other words, once the leftmost increment is carried past the least significant bit, the two increments can be processed concurrently – the increments' rewriting steps can be interleaved, with no observable difference between the various interleavings. We can even abstract from the interleavings by writing simply $e\,i\,b_0\,i \Longrightarrow e\,b_1\,b_1$.

Unfortunately, a concurrent treatment of increments falls outside the domain of ??. Intermediate computational states, such as ...,

$$e\,i\,b_0\,i \longrightarrow e\,i\,b_1$$

because $e\,i\,b_0$ is simply not a binary value. An adequacy theorem stronger than ?? is needed.

The situation here is roughly analogous to the desire, in a functional language, for stronger metatheorems than a big-step, natural sematics admits, and we adopt a similar solution.

To THIS END, we define a binary relation, $\approx_{\mathrm{I}}$, that assigns a natural number denotation to each intermediate computational state, not only to the terminal values as $\approx_{\mathrm{V}}$ did..[35]

[35] Like the $\approx_{\mathrm{V}}$ relation does for values, the $\approx_{\mathrm{I}}$ relation also serves to implicitly characterize the valid intermediate states as those contexts that form the relation's domain of definition. As with values, the valid intermediate states could also be enumerated more explicitly and syntactically with a grammar:

$$\Omega ::= e \mid \Omega\,b_0 \mid \Omega\,b_1 \mid \Omega\,i \mid e{\bullet}b_1 \mid \Omega\,(i{\bullet}b_0)$$

$$\frac{}{e \approx_{\mathrm{I}} 0}\ e\text{-I} \qquad \frac{\Omega \approx_{\mathrm{I}} n}{\Omega\,b_0 \approx_{\mathrm{I}} 2n}\ b_0\text{-I} \qquad \frac{\Omega \approx_{\mathrm{I}} n}{\Omega\,b_1 \approx_{\mathrm{I}} 2n+1}\ b_1\text{-I} \qquad \frac{\Omega \approx_{\mathrm{I}} n}{\Omega\,i \approx_{\mathrm{I}} n+1}\ i\text{-I}$$

$$\frac{}{e \bullet b_1 \approx_{\mathrm{I}} 1}\ \bullet_1\text{-I} \qquad \frac{\Omega \approx_{\mathrm{I}} n}{\Omega\,(i \bullet b_0) \approx_{\mathrm{I}} 2(n+1)}\ \bullet_2\text{-I}$$

Binary values should themselves be valid, terminal computational states, so the first three rules are carried over from the $\approx_{\mathrm{V}}$ relation. The i-I rule allows multiple increment instructions to be interspersed throughout the state. Lastly, because the atomicity of ordered rewriting steps is very fine-grained, the $\bullet_1$-I and $\bullet_2$-I rules are needed to completely describe the valid intermediate states and their denotations. For instance, the state $e\,i$ first rewrites to the intermediate $e \bullet b_1$ before eventually rewriting to $e\,b_1$; the state $\Omega\,(i \bullet b_0)$ has a similar status.

With this $\approx_{\mathrm{I}}$ relation in hand, we can now prove a stronger, small-step adequacy theorem.

THEOREM 3.23 (Small-step adequacy of increments).
Value soundness *If $\Omega \approx_{\mathrm{V}} n$, then $\Omega \approx_{\mathrm{I}} n$ and $\Omega \not\longrightarrow$.*
Preservation *If $\Omega \approx_{\mathrm{I}} n$ and $\Omega \longrightarrow \Omega'$, then $\Omega' \approx_{\mathrm{I}} n$.*
Progress *If $\Omega \approx_{\mathrm{I}} n$, then either: $\Omega \longrightarrow \Omega'$ for some Ω'; or; $\Omega \approx_{\mathrm{V}} n$.*[36]
Termination *If $\Omega \approx_{\mathrm{I}} n$, then every rewriting sequence from Ω is finite.*

[36] Compare with "If $\Omega \approx_{\mathrm{I}} n$, then $\Omega \approx_{\mathrm{V}} n$ if, and only if, $\Omega \not\longrightarrow$."

Proof. Each part is proved separately.

Value soundness can be proved by structural induction on the derivation of $\Omega \approx_{\mathrm{V}} n$.

Preservation and progress can likewise be proved by structural induction on the derivation of $\Omega \approx_{\mathrm{I}} n$. In particular, the $e \bullet b_1$ and $\Omega\ (i \bullet b_0)$ rules

Termination can be proved using an explicit termination measure, $|\Omega|$, that is strictly decreasing across each rewriting, $\Omega \longrightarrow \Omega'$. Specifically, we use a measure (see the adjacent figure), adapted from the standard amortized work analysis of increment for binary counters,[37] for which $\Omega \longrightarrow \Omega'$ implies $|\Omega| > |\Omega'|$. Because the measure is always nonnegative, only finitely many such rewritings can occur.

As an example case, consider the intermediate state $\Omega\, b_0\, i$ and its rewriting $\Omega\, b_0\, i \longrightarrow \Omega\, b_1$. It follows that $|\Omega\, b_0\, i| = |\Omega| + 4 > |\Omega| + 2 = |\Omega\, b_1|$. □

$$|e| = 0 \qquad |e \bullet b_1| = 3$$
$$|\Omega\, b_0| = |\Omega| \qquad |\Omega\ (i \bullet b_0)| = |\Omega| + 5$$
$$|\Omega\, b_1| = |\Omega| + 2$$
$$|\Omega\, i| = |\Omega| + 4$$

Figure 3.19: A termination measure, adapted from the standard amortized work analysis of increment for binary counters

[37] ??.

COROLLARY 3.24 (Big-step adequacy of increments).
Evaluation *If $\Omega \approx_{\mathrm{I}} n$, then $\Omega \Longrightarrow\approx_{\mathrm{V}} n$. In particular, if $\Omega \approx_{\mathrm{V}} n$, then $\Omega\, i \Longrightarrow\approx_{\mathrm{V}} n + 1$.*
Preservation *If $\Omega \approx_{\mathrm{I}} n$ and $\Omega \Longrightarrow \Omega'$, then $\Omega' \approx_{\mathrm{I}} n$. In particular, if $\Omega \approx_{\mathrm{V}} n$ and $\Omega\, i \Longrightarrow\approx_{\mathrm{V}} n'$, then $n' = n + 1$.*

Proof. The two parts are proved separately.

Evaluation can be proved by repeatedly appealing to the progress and preservation results(**??**). By the accompanying termination result, a binary value must eventually be reached.

Preservation can be proved by structural induction on the given rewriting sequence. □

BUT, OF COURSE, a few isolated examples do not make a proof.

By analogy with functional programming, the above adequacy conditions can be seen as stating evaluation and termination results for a big-step, evaluation semantics of increments, with $\Omega \approx_{\mathrm{V}} n$ acting as a kind of typing judgment – admittedly, a very precise one.

In functional programming, big-step results like these are usually proved by first providing a small-step operational semantics, then characterizing the valid intermediate states that arise with small steps, and finally establishing type preservation, progress, and termination results for the small-step semantics. We will adopt the same proof strategy here.

In this case, the small-step operational semantics already exists – it is simply the individual rewriting steps entailed by the definitions of e, b_0, and b_1. So our first task is to characterize the valid intermediate states that arise during a computation. To this end, we define a binary relation, $\approx_{\mathrm{I}}$, that, like the $\approx_{\mathrm{V}}$ relation, serves the dual purposes of enumerating the valid intermediate states and assigning to each state a natural number denotation.[38]

[38] As with values, we could also choose to enumerate the valid immediate states more explicitly and syntactically with a grammar:

$$\Omega ::= e \mid \Omega\, b_0 \mid \Omega\, b_1 \mid \Omega\, i \mid e \bullet b_1 \mid \Omega\ (i \bullet b_0)$$

$$\frac{}{e \approx_{\text{I}} 0} \qquad \frac{\Omega \approx_{\text{I}} n}{\Omega\, b_0 \approx_{\text{I}} 2n} \qquad \frac{\Omega \approx_{\text{I}} n}{\Omega\, b_1 \approx_{\text{I}} 2n+1} \qquad \frac{\Omega \approx_{\text{I}} n}{\Omega\, i \approx_{\text{I}} n+1}$$

$$\frac{}{e \bullet b_1 \approx_{\text{I}} 1} \qquad \frac{\Omega \approx_{\text{I}} n}{\Omega\, (i \bullet b_0) \approx_{\text{I}} 2(n+1)}$$

Binary values should themselves be valid, terminal computational states, so the first three rules are carried over from the $\approx_{\text{V}}$ relation. The fourth rule, involving i, allows multiple increments to be interspersed throughout the counter.

Because ordered rewriting steps are quite fine-grained, two final rules are needed to completely describe the valid intermediate states and their denotations. For instance, the state $e\, i$ first rewrites to $e \bullet b_1$ before eventually rewriting to $e\, b_1$.

Having characterized the valid intermediate states, we may state and prove the small-step adequacy of increments: preservation, progress, and termination.

Theorem 3.25 (Small-step adequacy of increments).
Value inclusion *If $\Omega \approx_{\text{V}} n$, then $\Omega \approx_{\text{I}} n$.*
Preservation *If $\Omega \approx_{\text{I}} n$ and $\Omega \longrightarrow \Omega'$, then $\Omega' \approx_{\text{I}} n$.*
Progress *If $\Omega \approx_{\text{I}} n$, then either: $\Omega \longrightarrow \Omega'$ for some Ω'; or; $\Omega \not\longrightarrow$ and $\Omega \approx_{\text{V}} n$.*
Termination *If $\Omega \approx_{\text{I}} n$, then every rewriting sequence from Ω is finite.*

Proof. Each part is proved separately.

Value inclusion can be proved by structural induction on the derivation of $\Omega \approx_{\text{V}} n$.

Preservation and progress can likewise be proved by structural induction on the derivation of $\Omega \approx_{\text{I}} n$. In particular, the $e \bullet b_1$ and $\Omega\, (i \bullet b_0)$ rules

Termination can be proved using an explicit termination measure, $|\Omega|$, that is strictly decreasing across each rewriting, $\Omega \longrightarrow \Omega'$. Specifically, we use a measure (see the adjacent figure), adapted from the standard amortized work analysis of increment for binary counters,[39] for which $\Omega \longrightarrow \Omega'$ implies $|\Omega| > |\Omega'|$. Because the measure is always nonnegative, only finitely many such rewritings can occur.

As an example case, consider the intermediate state $\Omega\, b_0\, i$ and its rewriting $\Omega\, b_0\, i \longrightarrow \Omega\, b_1$. It follows that $|\Omega\, b_0\, i| = |\Omega| + 4 > |\Omega| + 2 = |\Omega\, b_1|$. □

$$\begin{aligned} |e| &= 0 & |e \bullet b_1| &= 3 \\ |\Omega\, b_0| &= |\Omega| & |\Omega\, (i \bullet b_0)| &= |\Omega| + 5 \\ |\Omega\, b_1| &= |\Omega| + 2 \\ |\Omega\, i| &= |\Omega| + 4 \end{aligned}$$

Figure 3.20: A termination measure, adapted from the standard amortized work analysis of increment for binary counters

[39] ??.

Theorem 3.26 (Big-step adequacy of increments).
Preservation *If $\Omega \approx_{\text{I}} n$ and $\Omega \Longrightarrow\approx_{\text{V}} n'$, then $n' = n$.*
Termination? *If $\Omega \approx_{\text{I}} n$, then $\Omega \Longrightarrow\approx_{\text{V}} n$.*

Proof. Both parts are consequences of the small-step adequacy of increments (??).

Preservation is proved by structural induction on the given rewriting sequence. The base case follows [...] by an inner structural induction on the derivation of $\Omega \approx_{\mathrm{V}} n'$. The inductive case can be proved by first appealing to small-step preservation (??) and then to the inductive hypothesis.

Termination? is proved by repeatedly appealing to small-step progress (??). The small-step termination [...] (??) ensures that a value will be reached after finitely many such appeals. □

Corollary 3.27 (Structural adequacy of increments). *If $\Omega \approx_{\mathrm{V}} n$, then $\Omega\, i \Longrightarrow\approx_{\mathrm{V}} n'$ if, and only if, $n' = n + 1$.*

As an example computation, consider incrementing $e\, b_1$ twice, as captured by the state $e\, b_1\, i\, i$.

$$\begin{tikzcd}
 & & e\, b_1\, b_0\, i \arrow[dr] & \\
e\, b_1\, i\, i \arrow[r, Rightarrow] & e\, i\, b_0\, i \arrow[ur, Rightarrow] \arrow[dr] & & e\, b_1\, b_1 \\
 & & e\, i\, b_1 \arrow[ur, Rightarrow] &
\end{tikzcd}$$

First, processing of the leftmost increment begins: the least significant bit is flipped, and the increment is carried over to the more significant bits. This corresponds to the reduction $e\, b_1\, i\, i \Longrightarrow e\, i\, b_0\, i$. Next, either of the two remaining increments may be processed – that is, either $e\, i\, b_0\, i \Longrightarrow e\, b_1\, b_0\, i$ or $e\, i\, b_0\, i \Longrightarrow e\, i\, b_1$.

We should like to prove the correctness of this specification of increments by establishing a computational adequacy result:

Theorem 3.28 (Adequacy of increments). *If $\Omega \approx_{\mathrm{V}} n$ and $\Omega\, i \Longrightarrow\approx_{\mathrm{V}} n'$, then $n' = n + 1$. Moreover, if $\Omega \approx_{\mathrm{V}} n$, then $\Omega\, i \Longrightarrow\approx_{\mathrm{V}} n + 1$.*

By analogy with functional programming, this theorem can be seen as stating evaluation and termination results for a big-step evaluation semantics of increments – the judgment $\Omega \approx_{\mathrm{V}} n$ is acting as a kind of typing judgment, with n being the "type" [abstract interpretation?] of the counter Ω.

In functional programming, these sorts of big-step results are proved by first providing a small-step operational semantics, then characterizing the valid intermediate states that arise with small steps, and finally establishing type preservation, progress, and termination results for the small-step semantics. We will adopt the same strategy here.

First, we define a relation, $\approx_{\mathrm{I}}$, that characterizes the valid intermediate states that arise during increments.

To prove this **??**, we will first introduce an auxiliary relation, $\approx_{\mathrm{I}}$, that characterizes the valid states that arise during increments. This relation is defined inductively by the following rules.

$$\frac{}{e \approx_{\mathrm{I}} 0} \qquad \frac{\Omega \approx_{\mathrm{I}} n}{\Omega\, b_0 \approx_{\mathrm{I}} 2n} \qquad \frac{\Omega \approx_{\mathrm{I}} n}{\Omega\, b_1 \approx_{\mathrm{I}} 2n + 1} \qquad \frac{\Omega \approx_{\mathrm{I}} n}{\Omega\, i \approx_{\mathrm{I}} n + 1}$$

$$\frac{}{e \bullet b_1 \approx_{\mathrm{I}} 1} \qquad \frac{\Omega \approx_{\mathrm{I}} n}{\Omega\, (i \bullet b_0) \approx_{\mathrm{I}} 2(n + 1)}$$

The latter two

A DECREMENT OPERATION Binary counters may also be equipped with a decrement operation. Instead of examining decrements *per se*, we will implement a closely related operation: the normalization of binary representations to what might be called *head-unary form.*[40] An ordered context Ω will be said to be in head-unary form if it has one of two forms: $\Omega = z$; or $\Omega = \Omega'\, s$, for some binary number Ω'.

[40] We will frequently abuse terminology, using 'head-unary normalization' and 'decrement' interchangably.

Just as appending the atom i to a counter initiates an increment, appending an uninterpreted atom d will cause the counter to begin normalizing to head-unary form. The following ?? serves as a specification of head-unary normalization, relating a value's head-unary form to its denotation.

THEOREM 3.29 (Structural adequacy of decrements). *If $\Omega \approx_{\mathrm{V}} n$, then:*

- $\Omega\, d \Longrightarrow z$ *if, and only if,* $n = 0$*;*
- $\Omega\, d \Longrightarrow \Omega'\, s$ *for some* Ω' *such that* $\Omega' \approx_{\mathrm{V}} n - 1$*, if* $n > 0$*; and*
- $\Omega\, d \Longrightarrow \Omega'\, s$ *only if* $n > 0$ *and* $\Omega' \approx_{\mathrm{V}} n - 1$*.*

For example, because $e\, b_1$ denotes 1, a computation $e\, b_1\, d \Longrightarrow \Omega'\, s$ must exist, for some $\Omega' \approx_{\mathrm{V}} 0$.

ONCE AGAIN, to implement these desiderata locally, the recursive definitions of e, b_0, and b_1 will be revised with an additional clause that handles decrements; also, a recursively defined proposition b_0' is introduced:

$e \triangleq (\cdots / i) \mathbin{\&} (z / d)$ Because e denotes 0, its head-unary form is simply z.

$b_0 \triangleq (\cdots / i) \mathbin{\&} (d \bullet b_0' / d)$ Because $\Omega\, b_0$ denotes $2n$ if Ω denotes n, its head-unary form can be contructed by recursively putting the more significant bits into head-unary form and appending b_0' to process that result.

$b_0' \triangleq (z \setminus z) \mathbin{\&} (s \setminus b_1 \bullet s)$ If the more significant bits have head-unary form z and therefore denote 0, then $\Omega\, b_0$ also denotes 0 and has head-unary form z. Otherwise, if they have head-unary form $\Omega'\, s$ and therefore denote $n > 0$, then $\Omega\, b_0$ denotes $2n$ and has head-unary form $\Omega'\, b_1\, s$, which can be constructed by replacing s with $b_1\, s$.

$b_1 \triangleq (\cdots / i) \mathbin{\&} (b_0 \bullet s / d)$ Because $\Omega\, b_1$ denotes $2n + 1$ if Ω denotes n, its head-unary form, $\Omega\, b_0\, s$, can be constructed by flipping the least significant bit to b_0 and appending s.

Comfortingly, $2 - 1 = 1$: the head-unary form of $e\, b_1$ is $e\, b_0\, b_1\, s$:

$$e\, b_1\, b_0\, d \Longrightarrow e\, b_1\, d\, b_0' \Longrightarrow e\, b_0\, s\, b_0' \Longrightarrow e\, b_0\, b_1\, s\,.$$

AT THIS POINT, we would like to prove the adequacy of decrements. However, having just revised the definitions of e, b_0, and b_1, we must first recheck the adequacy of binary representation(see ??). Unfortunately, the newly introduced alternative conjunctions, together with the fine-grained atomicity of ordered rewriting, cause [...].

FALSE CLAIM 3.30 (Adequacy of binary representations).

Functional *For every binary number Ω, there exists a unique natural number n such that $\Omega \approx_{\mathrm{V}} n$.*

Surjectivity *For every natural number n, there exists a binary number Ω such that $\Omega \approx_{\mathrm{V}} n$.*

Values *If $\Omega \approx_{\mathrm{V}} n$, then $\Omega \not\longrightarrow$.*

Counterexample. Although the $\approx_{\mathrm{V}}$ relation remains functional and surjective, it does not satisfy [...]. Because $e \approx_{\mathrm{V}} 0$, the counter e is a value (with denotation 0). However, because the atomicity of ordered rewriting is extremely fine-grained, e can be rewritten:

$$\begin{tikzcd} & e \bullet b_1 \mathbin{/} i \\ e = (e \bullet b_1 \mathbin{/} i) \mathbin{\&} (z \mathbin{/} d) \arrow[ur] \arrow[dr] & \\ & z \mathbin{/} d \end{tikzcd}$$

That e is an active proposition violates our conception of values as inactive. □

AT THIS POINT, we would like to prove the adequacy of decrements. However, having just revised the definitions of e, b_0, and b_1, we must first recheck the adequacy of increments. Unfortunately, the newly introduced alternative conjunctions, together with the fine-grained atomicity of ordered rewriting, cause the preservation and progress properties to fail.

FALSE CLAIM 3.31 (Small-step adequacy of increments).

Value inclusion *If $\Omega \approx_{\mathrm{V}} n$, then $\Omega \approx_{\mathrm{I}} n$.*

Preservation *If $\Omega \approx_{\mathrm{I}} n$ and $\Omega \longrightarrow \Omega'$, then $\Omega' \approx_{\mathrm{I}} n$.*

Progress *If $\Omega \approx_{\mathrm{I}} n$, then either: $\Omega \longrightarrow \Omega'$ for some Ω'; $\Omega \not\longrightarrow$ and $\Omega \approx_{\mathrm{V}} n$*

Termination *If $\Omega \approx_{\mathrm{I}} n$, then every rewriting sequence from Ω is finite.*

Counterexample. As a counterexample to preservation, notice that $e\, i$ denotes 1 and that

$$e\, i = (e \bullet b_1 \mathbin{/} i) \mathbin{\&} (z \mathbin{/} d) \longrightarrow (e \bullet b_1 \mathbin{/} i)\, i\,,$$

but that $(e \bullet b_1 \mathbin{/} i)\, i$ does not have a denotation under the $\approx_{\mathrm{I}}$ relation.

Even worse, computations can now enter stuck states – $e\, i \longrightarrow (z / d)\, i \not\longrightarrow$, for example. It's difficult to imagine assigning denotations to these stuck states, making them counterexamples to preservation. Even if denotations were somehow assigned to them, such states would anyway violate the desired progress theorem. □

In both cases, these counterexamples arise from the very fine-grained atomicity of ordered rewriting. Now that the definitions of e, b_0, and b_1 include alternative conjunctions, [...].

THEOREM 3.32. *Evaluation* *If $\Omega \approx_{\mathrm{I}} n$, then $\Omega \Longrightarrow \approx_{\mathrm{V}} n$. In particular, if $\Omega \approx_{\mathrm{V}} n$, then $\Omega \Longrightarrow n + 1 \approx_{\mathrm{V}}$.*

Preservation If $\Omega \approx_{\text{I}} n$ and $\Omega \Longrightarrow\approx_{\text{V}} n'$, then $n' = n$.

Proof. By structural induction on the given derivation of $\Omega \approx_{\text{I}} n$. □

The solution is to chain several small rewriting steps together into a single, larger atomic step.

3.13 *Weakly focused rewriting*

Andreoli:??'s observation was that propositions can be partitioned into two classes, or *polarities*[41], according to the invertibility of their sequent calculus rules, and that [...].

[41] reference?

The ordered propositions are polarized into two classes, the positive and negative propositions, according to the invertibility of their sequent calculus rules.

POSITIVE PROPS. $A^+ ::= \alpha^+ \mid A^+ \bullet B^+ \mid \mathbf{1} \mid {\downarrow}A^-$

NEGATIVE PROPS. $A^- ::= \alpha^- \mid A^+ \setminus B^- \mid B^- \mathbin{/} A^+ \mid A^- \mathbin{\&} B^- \mid \top \mid {\uparrow}A^+$

The positive propositions are those propositions that have invertible left rules, such as ordered conjunction; the negative propositions are those that have invertible right rules, such as the ordered implications.

ORDERED CONTEXTS $\Omega ::= \Omega_1\, \Omega_2 \mid \cdot \mid A^+$

Left rules for negative connectives may be chained together into a single *left-focusing phase*, reflected by the pattern judgment $\Omega_L\,[A^-]\,\Omega_R \Vdash C^+$. Following **Zeilberger**:??, this judgment can be read as a function of an in-focus negative proposition, A^-, that produces the ordered contexts Ω_L and Ω_R and the positive consequent C^+ as outputs.

The left-focus judgment is defined inductively on the structure of the in-focus proposition by the following rules.

$$\frac{\Omega_L\,[B^-]\,\Omega_R \Vdash C^+}{\Omega_L\, A^+\,[A^+ \setminus B^-]\,\Omega_R \Vdash C^+}\ {\setminus}\textsc{l}' \qquad \frac{\Omega_L\,[B^-]\,\Omega_R \Vdash C^+}{\Omega_L\,[B^- \mathbin{/} A^+]\,A^+\,\Omega_R \Vdash C^+}\ {/}\textsc{l}'$$

$$\frac{\Omega_L\,[A^-]\,\Omega_R \Vdash C^+}{\Omega_L\,[A^- \mathbin{\&} B^-]\,\Omega_R \Vdash C^+}\ \&\textsc{l}_1 \qquad \frac{\Omega_L\,[B^-]\,\Omega_R \Vdash C^+}{\Omega_L\,[A^- \mathbin{\&} B^-]\,\Omega_R \Vdash C^+}\ \&\textsc{l}_2 \qquad (\text{no } \top\textsc{l} \text{ rule})$$

$$\frac{}{[{\uparrow}A^+] \Vdash A^+}\ {\uparrow}\textsc{l}$$

These rules parallel the usual sequent calculus rules, maintaining focus on the subformulas of negative polarity. First, the ${\uparrow}$L rule finishes a left-focusing phase by producing the consequent A^+ from ${\uparrow}A^+$.

Second, the $\setminus$L′ and /L′ rules diverge slightly from the usual left rules for left- and right-handed implication in that they have no premises decomposing [the antecedent[42]] A^+. This would mean that a weakly focused sequent

[42] wc?

calculus based on $\backslash\text{L}'$ and $/\text{L}'$ would be incomplete for provability. It is possible to [...].[43] However, because our goal here is a rewriting framework and such a framework is inherently incomplete[44], [...].

[43] Simmons:CMU??.

[44] Is this right?

$$\frac{\Omega_L\ [A^-]\ \Omega_R \Vdash C^+}{\Omega_L \downarrow A^-\ \Omega_R \longrightarrow C^+}\ {\downarrow}\text{D}$$

Consider the recursively defined proposition $\alpha \triangleq (\beta \backslash \alpha) \,\&\, (\gamma \backslash \mathbf{1})$. Previously, in the unfocused rewriting framework, it took two steps to rewrite $\beta\,\alpha$ into α:

$$\beta\,\alpha = \beta\,\big((\beta \backslash \alpha) \,\&\, (\gamma \backslash \mathbf{1})\big) \longrightarrow \beta\,(\beta \backslash \alpha) \longrightarrow \alpha$$

Now, in the polarized, weakly focused rewriting framework, the analogous recursive definition is $\alpha^- \triangleq (\beta^+ \backslash {\uparrow}{\downarrow}\alpha^-) \,\&\, (\gamma^+ \backslash {\uparrow}\mathbf{1})$, and it takes only one step to rewrite $\beta^+ \downarrow\alpha^-$ into $\downarrow\alpha^-$:

$$\beta^+ \downarrow\alpha^- = \beta^+ \downarrow\big((\beta^+ \backslash {\uparrow}{\downarrow}\alpha^-) \,\&\, (\gamma^+ \backslash {\uparrow}\mathbf{1})\big) \longrightarrow {\downarrow}\alpha^-$$

because $\beta^+\ [\alpha^-] \Vdash {\downarrow}\alpha^-$.

Notice that, because the left-focus judgment is defined inductively, there are some recursively defined negative propositions that cannot successfully be put in focus. For example, under the definition $\alpha^- \triangleq \beta^+ \backslash \alpha^-$, there are no contexts Ω_L and Ω_R and conseqeunt C^+ for which $\Omega_L\ [\alpha^-]\ \Omega_R \Vdash C^+$ is derivable.

In addition to the ${\downarrow}\text{D}$ rule for decomposing ${\downarrow}A^-$, weakly focused ordered rewriting retains the $\bullet\text{D}$ and $\mathbf{1}\text{D}$ rules for decomposing $A^+ \bullet B^+$ and $\mathbf{1}$ and the compatability rules, $\longrightarrow\text{C}_\text{L}$ and $\longrightarrow\text{C}_\text{R}$. Together, these five rules and the left focus[45] rules comprise the weakly focused ordered rewriting framework; they are summarized in ??.

[45] focal?

$$\begin{aligned}(\Omega_1\,\Omega_2)^\circ &= \Omega_1^\circ\,\Omega_2^\circ & ({\downarrow}A^-)^\circ &= (A^-)^\circ\\ (\cdot)^\circ &= \cdot & ({\uparrow}A^+)^\circ &= (A^+)^\circ\\ (A^+)^\circ &= (A^+)^\circ & (A^+ \backslash B^-)^\circ &\\ && = (A^+)^\circ \backslash (B^-)^\circ &\end{aligned}$$

etc.

Figure 3.22: Depolarization of propositions and contexts

Weakly focused ordered rewriting is sound with respect to the unfocused rewriting framework of ??. Given a depolarization function $(-)^\circ$ that maps polarized propositions and contexts to their unpolarized counterparts,we may state and prove the following soundness theorem for weakly focused rewriting.

THEOREM 3.33 (Soundness of weakly focused rewriting). *If* $\Omega \Longrightarrow \Omega'$, *then* $\Omega^\circ \Longrightarrow (\Omega')^\circ$.

Proof. By structural induction on the given rewriting step, after generalizing the inductive hypothesis to include:

- If $\Omega \longrightarrow \Omega'$, then $\Omega^\circ \Longrightarrow (\Omega')^\circ$.
- If $\Omega_L\ [A^-]\ \Omega_R \Vdash C^+$, then $(\Omega_L \downarrow A^-\ \Omega_R)^\circ \Longrightarrow (C^+)^\circ$. □

A completeness theorem also holds, but we forgo its development because it is not essential to the remainder of this work.

Second, with the lone exception negative propositions are latent[46] –

[46] ??.

Figure 3.21: A weakly focused ordered rewriting framework

POSITIVE PROPS. $A^+ ::= \alpha^+ \mid A^+ \bullet B^+ \mid \mathbf{1} \mid {\downarrow}A^-$

NEGATIVE PROPS. $A^- ::= \alpha^- \mid A^+ \setminus B^- \mid B^- / A^+ \mid A^- \mathbin{\&} B^- \mid \top \mid {\uparrow}A^+$

ORDERED CONTEXTS $\Omega ::= \Omega_1\,\Omega_2 \mid \cdot \mid A^+$

REWRITING: $\Omega \longrightarrow \Omega'$ AND $\Omega \Longrightarrow \Omega'$

$$\frac{\Omega_L\ [A^-]\ \Omega_R \Vdash C^+}{\Omega_L\ {\downarrow}A^-\ \Omega_R \longrightarrow C^+}\ {\downarrow}\textsc{d} \qquad \frac{}{A^+ \bullet B^+ \longrightarrow A^+\ B^+}\ {\bullet}\textsc{d} \qquad \frac{}{\mathbf{1} \longrightarrow \cdot}\ \mathbf{1}\textsc{d}$$

(no $\oplus$D and $\mathbf{0}$D rules)

$$\frac{\Omega_1 \longrightarrow \Omega_1'}{\Omega_1\,\Omega_2 \longrightarrow \Omega_1'\,\Omega_2}\ {\longrightarrow}\textsc{c}_\textsc{l} \qquad \frac{\Omega_1 \longrightarrow \Omega_1'}{\Omega_1\,\Omega_2 \longrightarrow \Omega_1'\,\Omega_2}\ {\longrightarrow}\textsc{c}_\textsc{r}$$

$$\frac{}{\Omega \Longrightarrow \Omega}\ {\Longrightarrow}\textsc{r} \qquad \frac{\Omega \longrightarrow \Omega' \quad \Omega' \Longrightarrow \Omega''}{\Omega \Longrightarrow \Omega''}\ {\Longrightarrow}\textsc{t}$$

LEFT FOCUS: $\Omega_L\ [A^-]\ \Omega_R \Vdash C^+$

$$\frac{\Omega_L\ A^+\ [B^-]\ \Omega_R \Vdash C^+}{\Omega_L\ [A^+ \setminus B^-]\ \Omega_R \Vdash C^+}\ \setminus\textsc{l}' \qquad \frac{\Omega_L\ [B^-]\ A^+\ \Omega_R \Vdash C^+}{\Omega_L\ [B^- / A^+]\ \Omega_R \Vdash C^+}\ /\textsc{l}'$$

$$\frac{\Omega_L\ [A^-]\ \Omega_R \Vdash C^+}{\Omega_L\ [A^- \mathbin{\&} B^-]\ \Omega_R \Vdash C^+}\ \&\textsc{l}_1 \qquad \frac{\Omega_L\ [B^-]\ \Omega_R \Vdash C^+}{\Omega_L\ [A^- \mathbin{\&} B^-]\ \Omega_R \Vdash C^+}\ \&\textsc{l}_2 \qquad \text{(no } \top\textsc{l} \text{ rule)}$$

$$\frac{}{[{\uparrow}A^+] \Vdash A^+}\ {\uparrow}\textsc{l}$$

3.14 *Revisiting automata*

$$\hat{q} \triangleq (\&_{a\in\Sigma}(a \setminus {\uparrow}{\downarrow}\hat{q}'_a)) \mathbin{\&} (\epsilon \setminus {\uparrow}\hat{F}(q))$$

where

$$q \xrightarrow{a} q'_a,\text{ for all } a \in \Sigma \quad\text{and}\quad \hat{F}(q) = \begin{cases} \mathbf{1} & \text{if } q \in F \\ {\downarrow}\top & \text{if } q \notin F \end{cases}$$

THEOREM 3.34 (DFA adequacy up to bisimilarity). *Let $\mathcal{A} = (Q, ?, F)$ be a DFA over the input alphabet Σ. Then, for all states q, q', and s:*

1. *$q \sim s$ if, and only if, $\hat{q} = \hat{s}$.*
2. *$q \sim\xrightarrow{a}\sim q'$ if, and only if, $a\,\hat{q} \Longrightarrow \hat{q}'$, for all input symbols $a \in \Sigma$. More generally, $q \sim\xrightarrow{w}\sim q'$ if, and only if, $w^{\mathrm{R}}\,\hat{q} \Longrightarrow \hat{q}'$, for all finite words $w \in \Sigma^*$.*
3. *$q \in F$ if, and only if, $\epsilon\,\hat{q} \Longrightarrow \mathbf{1}$.*

Lemma**??** is still needed, but now has a much different proof. Previously, the proof of **??** relied on a very specific and delicate property of DFAs, namely that each DFA state has one and only one a-successor for each input symbol a. Now, with weakly focused ordered rewriting, the **??**'s proof is much less fragile. With the larger granularity of individual rewriting steps that the weakly focused framework affords, a state's encoding is a latent proposition

3.15 *Revisiting binary counters*

With ordered rewriting now based on a weakly focused sequent calculus, we can revisit our previous attempt to extend binary counters with support for decrements or head-unary normalization.

The propositions e, b_0, b'_0, and b_1 are recursively defined in nearly the same way as before. With one exception discussed below, only the necessary shifts are inserted to consistently assign a negative polarity to the defined atoms e, b_0, b'_0, and b_1 and a positive polarity to the uninterpreted atoms i, d, z, and s.

$$\begin{aligned} e &\triangleq (e \bullet b_1 \mathbin{/} i) \mathbin{\&} (z \mathbin{/} d) \\ b_0 &\triangleq ({\uparrow}{\downarrow}b_1 \mathbin{/} i) \mathbin{\&} (d \bullet b'_0 \mathbin{/} d) \\ b'_0 &\triangleq (z \setminus z) \mathbin{\&} (s \setminus b_1 \bullet s) \\ b_1 &\triangleq (i \bullet b_0 \mathbin{/} i) \mathbin{\&} (b_0 \bullet s \mathbin{/} d) \end{aligned}$$

VALUES Once again, we use the same $\approx_{\mathrm{v}}$ relation to assign a unique natural number denotation to each binary representation.

$$\frac{}{e \approx_{\mathrm{v}} 0}\ e\text{-v} \qquad \frac{\Omega \approx_{\mathrm{v}} n}{\Omega\, b_0 \approx_{\mathrm{v}} 2n}\ b_0\text{-v} \qquad \frac{\Omega \approx_{\mathrm{v}} n}{\Omega\, b_1 \approx_{\mathrm{v}} 2n+1}\ b_1\text{-v}$$

Because the underlying ordered rewriting framework has changed, we must verify that $\approx_{\mathrm{v}}$ is adequate – inparticular, the […] property that values cannot be independently rewritten.

THEOREM 3.35 (Adequacy of binary representations).

Functional *For every binary number Ω, there exists a unique natural number n such that $\Omega \approx_{\mathrm{V}} n$.*

Surjectivity *For every natural number n, there exists a binary number Ω such that $\Omega \approx_{\mathrm{V}} n$.*

Value *If $\Omega \approx_{\mathrm{V}} n$, then $\Omega \not\longrightarrow$.*

Proof. By induction over the structure of Ω. As an example, consider the case in which $e \approx_{\mathrm{V}} 0$. Indeed, $e \not\longrightarrow$ because $e = (e \bullet b_1 \mathbin{/} i) \mathbin{\&} (z \mathbin{/} d)$ and

$$\Omega_L \, [(e \bullet b_1 \mathbin{/} i) \mathbin{\&} (z \mathbin{/} d)] \, \Omega_R \Vdash C^{+} \text{ only if } \Omega_L = \cdot \text{ and either } \Omega_R = i \text{ or } \Omega_R = d.$$

The other cases are similar. □

INCREMENT Previously, under the unfocused rewriting framework[47], rewriting $e\,i$ into $e \bullet b_1$ took two small steps:

[47] wc?

$$e\,i = \big((e \bullet b_1 \mathbin{/} i) \mathbin{\&} (z \mathbin{/} d)\big)\, i \longrightarrow (e \bullet b_1 \mathbin{/} i)\, i \longrightarrow e \bullet b_1$$

But now, with weakly focused rewriting, those two steps are combined into one atomic whole: $e\,i \longrightarrow e \bullet b_1$.

As for the unfocused rewriting implementation of binary increments, we use a $\approx_{\mathrm{I}}$ relation to assign a natural number denotation to each computational state. In fact, the specific definition of the $\approx_{\mathrm{I}}$ remains unchanged from ??:

$$\frac{}{e \approx_{\mathrm{I}} 0}\;e\text{-I} \qquad \frac{\Omega \approx_{\mathrm{I}} n}{\Omega\, b_0 \approx_{\mathrm{I}} 2n}\;b_0\text{-I} \qquad \frac{\Omega \approx_{\mathrm{I}} n}{\Omega\, b_1 \approx_{\mathrm{I}} 2n+1}\;b_1\text{-I} \qquad \frac{\Omega \approx_{\mathrm{I}} n}{\Omega\, i \approx_{\mathrm{I}} n+1}\;i\text{-I}$$

$$\frac{}{e \bullet b_1 \approx_{\mathrm{I}} 1}\;\bullet_1\text{-I} \qquad \frac{\Omega \approx_{\mathrm{I}} n}{\Omega\,(i \bullet b_0) \approx_{\mathrm{I}} 2(n+1)}\;\bullet_2\text{-I}$$

The only exception to [...] is the appearance of $\uparrow\downarrow b_1$ in the definition of b_0. Without this double shift, $e\, b_0\, i$ would be latent, unable to rewrite to a value until a second increment is appended, because the necessary $[(b_1 \mathbin{/} i) \mathbin{\&} (d \bullet b_0' \mathbin{/} d)]\; i \Vdash b_1$ is not derivable. However, with the double shift, $e\, b_0\, i \longrightarrow e\, b_1$ because $[(\uparrow\downarrow b_1 \mathbin{/} i) \mathbin{\&} (d \bullet b_0' \mathbin{/} d)]\; i \Vdash \downarrow b_1$ is derivable.

With weakly focused rewriting, it is no longer possible to reach the stuck state ....

THEOREM 3.36 (Small-step adequacy of increments).

Value soundness *If $\Omega \approx_{\mathrm{V}} n$, then $\Omega \approx_{\mathrm{I}} n$ and $\Omega \not\longrightarrow$.*

Preservation *If $\Omega \approx_{\mathrm{I}} n$ and $\Omega \longrightarrow \Omega'$, then $\Omega' \approx_{\mathrm{I}} n$.*

Progress *If $\Omega \approx_{\mathrm{I}} n$, then either: $\Omega \longrightarrow \Omega'$ for some Ω'; or; $\Omega \approx_{\mathrm{V}} n$.*[48]

Termination *If $\Omega \approx_{\mathrm{I}} n$, then every rewriting sequence from Ω is finite.*

[48] Compare with "If $\Omega \approx_{\mathrm{I}} n$, then $\Omega \approx_{\mathrm{V}} n$ if, and only if, $\Omega \not\longrightarrow$."

Proof. As before, each part is proved separately.

Value soundness, preservation, and progress can be proved by structural induction on the derivation of $\Omega \approx_{\mathrm{I}} n$.

Termination can be proved using the same explicit termination measure, $|\Omega|$, as in ??. □

DECREMENTS

$$\frac{\Omega \approx_{\text{I}} n}{\Omega\, d \approx_{\text{D}} n}\ d\text{-D} \qquad \frac{\Omega \approx_{\text{D}} n}{\Omega\, b_0' \approx_{\text{D}} 2n}\ b_0'\text{-D} \qquad \frac{}{z \approx_{\text{D}} 0}\ z\text{-D} \qquad \frac{\Omega \approx_{\text{I}} n}{\Omega\, s \approx_{\text{D}} n+1}\ s\text{-D}$$

$$\frac{\Omega \approx_{\text{I}} n}{\Omega\, (d \bullet b_0') \approx_{\text{D}} 2n}\ \bullet_1\text{-D}$$

$$\frac{\Omega \approx_{\text{I}} n}{\Omega\, (b_0 \bullet s) \approx_{\text{D}} 2n+1}\ \bullet_2\text{-D} \qquad \frac{\Omega \approx_{\text{I}} n}{\Omega\, (b_1 \bullet s) \approx_{\text{D}} 2n+2}\ \bullet_3\text{-D}$$

THEOREM 3.37 (Small-step adequacy of decrements).
Preservation *If $\Omega \approx_{\text{D}} n$ and $\Omega \longrightarrow \Omega'$, then $\Omega' \approx_{\text{D}} n$.*
Progress *If $\Omega \approx_{\text{D}} n$, then [either]:*

- *$\Omega \longrightarrow \Omega'$, for some Ω';*
- *$n = 0$ and $\Omega = z$; or*
- *$n > 0$ and $\Omega = \Omega'\, s$, for some Ω' such that $\Omega' \approx_{\text{I}} n - 1$.*

Termination *If $\Omega \approx_{\text{D}} n$, then every rewriting sequence from Ω is finite.*

Proof. *Preservation and progress* are proved, as before, by structural induction on the given derivation of $\Omega \approx_{\text{D}} n$.

Termination is proved by exhibiting a measure, $|{-}|_{\text{D}}$, that is strictly decreasing across each rewriting. Following the example of termination for increment-only binary counters(??), we could try to assign a constant amount of potential to each of the counter's constituents. Leaving these potentials as unknowns, we can generate a set of constraints from the allowed rewritings and then attempt to solve them.

For instance, here are several rewritings and their corresponding potential constraints.

Some selected rewritings	*Potential constraints*
$\Omega\, b_1\, i \longrightarrow \Omega\, (i \bullet b_0) \longrightarrow \Omega\, i\, b_0$	$b_1 + i > i + b_0 + 1$
$\Omega\, b_0\, d \longrightarrow \Omega\, (d \bullet b_0') \longrightarrow \Omega\, d\, b_0'$	$b_0 + d > d + b_0' + 1$
$\Omega\, s\, b_0' \longrightarrow \Omega\, (b_1 \bullet s) \longrightarrow \Omega\, b_1\, s$	$s + b_0' > b_1 + s + 1$

These constraints are satisfiable only if $b_1 > b_0 > b_0' > b_1$, which is, of course, impossible.

However, notice that each b_1 that arises from an interaction between s and b_0' will never participate in further rewritings because any increments remaining to the left of b_1 will only involve more significant bits, not this less significant b_1. A similar argument can be made for all bits that occur between the rightmost i and the terminal s, suggesting that those bits be assigned no potential at all.

This leads to the termination measure, $|{-}|_{\text{D}}$, and its auxiliary measures, $|{-}|_{\text{I}}$ and $|{-}|_{\text{S}}$, shown in the adjacent ??. (Note that the measure $|{-}|_{\text{I}}$ is not the same as the measure used for increment-only binary counters(??).)x is proved by exhibiting a pair of measures, $|\Omega|_d$ and $|\Omega|_s$, ordered lexicographically:

- If $\Omega \approx_{\mathrm{D}} n$ and $\Omega \longrightarrow \Omega'$, then either: $|\Omega|_d > |\Omega'|_d$; or $|\Omega|_d = 0$ and $|\Omega|_s > |\Omega'|_s$.

These measures are shown in the adjacent ??, rely on an auxiliary measure, $|\Omega|$, for increment states. Unfortunately, it is not possible to simply reuse the measure from ??. In that measure, each b_0 bit was assigned no potential. With decrements, however, b_0 needs to carry enough potential to transfer to b_0' in case a decrement instruction is encountered.

$\lvert\Omega\, s\rvert_s = \lvert\Omega\rvert$	$\lvert\Omega\, d\rvert_d = \lvert\Omega\rvert + 1$
$\lvert e\rvert = 0$	$\lvert\Omega\, b_0'\rvert_d = \lvert\Omega\rvert_d + 2$
$\lvert\Omega\, b_0\rvert = \lvert\Omega\rvert + 4$	$\lvert z\rvert_d = 0$
$\lvert\Omega\, b_1\rvert = \lvert\Omega\rvert + 6$	$\lvert\Omega\, s\rvert_d = 0$
$\lvert\Omega\, i\rvert = \lvert\Omega\rvert + 8$	$\lvert\Omega\,(d \bullet b_0')\rvert_d = \lvert\Omega\rvert + 4$
$\lvert e \bullet b_1\rvert = 7$	$\lvert\Omega\,(b_0 \bullet s)\rvert_d = 1$
$\lvert\Omega\,(i \bullet b_0)\rvert = \lvert\Omega\rvert + 13$	$\lvert\Omega\,(b_1 \bullet s)\rvert_d = 1$

For the rewritings $\Omega\, b_1\, i \longrightarrow \Omega\,(i \bullet b_0) \longrightarrow \Omega\, i\, b_0$, the assigned potentials must satisfy $b_1 + i > i + b_0 + 1$

No

As an example case, consider the intermediate state $\Omega\,(b_1 \bullet s)$ and its rewriting $\Omega\,(b_1 \bullet s) \longrightarrow \Omega\, b_1\, s$. It follows $|\Omega\,(b_1 \bullet s)|_d = 1 > 0 = |\Omega\, b_1\, s|_d$. Any subsequent rewritings of Ω are justified by a decrease in $|\Omega\, b_1\, s|_s > |\Omega|$.

□

Corollary 3.38 (Big-step adequacy of decrements). *If $\Omega \approx_{\mathrm{D}} n$, then:*
- *$\Omega \Longrightarrow z$ if, and only if, $n = 0$;*
- *$\Omega \Longrightarrow \Omega'\, s$ for some Ω' such that $\Omega' \approx_{\mathrm{I}} n - 1$, if $n > 0$; and*
- *$\Omega \Longrightarrow \Omega'\, s$ only if $n > 0$ and $\Omega' \approx_{\mathrm{I}} n - 1$.*

Proof. From the small-step preservation result of ??, it is possible to prove, using a structural induction on the given trace, a big-step preservation result: namely, that $\Omega \approx_{\mathrm{D}} n$ and $\Omega \Longrightarrow \Omega'$ only if $\Omega' \approx_{\mathrm{D}} n$. Each of the above claims then follows from either progress and termination(??) or big-step preservation together with inversion. □

$\lvert\Omega\, d\rvert_{\mathrm{D}} = \lvert\Omega\rvert_{\mathrm{I}} + 1$	$\lvert e\rvert_{\mathrm{I}} = 0$	$\lvert e\rvert_{\mathrm{S}} = \lvert e\rvert_{\mathrm{I}} = 0$
$\lvert\Omega\, b_0'\rvert_{\mathrm{D}} = \lvert\Omega\rvert_{\mathrm{D}} + 2$	$\lvert\Omega\, b_0\rvert_{\mathrm{I}} = \lvert\Omega\rvert_{\mathrm{I}} + 4$	$\lvert\Omega\, b_0\rvert_{\mathrm{S}} = \lvert\Omega\rvert_{\mathrm{S}}$
$\lvert z\rvert_{\mathrm{D}} = 0$	$\lvert\Omega\, b_1\rvert_{\mathrm{I}} = \lvert\Omega\rvert_{\mathrm{I}} + 6$	$\lvert\Omega\, b_1\rvert_{\mathrm{S}} = \lvert\Omega\rvert_{\mathrm{S}}$
$\lvert\Omega\, s\rvert_{\mathrm{D}} = \lvert\Omega\rvert_{\mathrm{S}}$	$\lvert\Omega\, i\rvert_{\mathrm{I}} = \lvert\Omega\rvert_{\mathrm{I}} + 8$	$\lvert\Omega\, i\rvert_{\mathrm{S}} = \lvert\Omega\, i\rvert_{\mathrm{I}} = \lvert\Omega\rvert_{\mathrm{I}} + 8$
$\lvert\Omega\,(d \bullet b_0')\rvert_{\mathrm{D}} = \lvert\Omega\, d\, b_0'\rvert_{\mathrm{D}} + 1$	$\lvert e \bullet b_1\rvert_{\mathrm{I}} = \lvert e\, b_1\rvert_{\mathrm{I}} + 1$	$\lvert e \bullet b_1\rvert_{\mathrm{S}} = \lvert e\, b_1\rvert_{\mathrm{S}} + 1$
$\lvert\Omega\,(b_0 \bullet s)\rvert_{\mathrm{D}} = \lvert\Omega\, b_0\, s\rvert_{\mathrm{D}} + 1$	$\lvert\Omega\,(i \bullet b_0)\rvert_{\mathrm{I}} = \lvert\Omega\, i\, b_0\rvert_{\mathrm{I}} + 1$	$\lvert\Omega\,(i \bullet b_0)\rvert_{\mathrm{S}} = \lvert\Omega\, i\, b_0\rvert_{\mathrm{S}} + 1$
$\lvert\Omega\,(b_1 \bullet s)\rvert_{\mathrm{D}} = \lvert\Omega\, b_1\, s\rvert_{\mathrm{D}} + 1$		

- If $\Omega \approx_{\mathrm{D}} n$ and $\Omega \longrightarrow \Omega'$, then $|\Omega|_{\mathrm{D}} > |\Omega'|_{\mathrm{D}}$.

- If $\Omega \approx_{\mathrm{I}} n$ and $\Omega \longrightarrow \Omega'$, then $|\Omega|_{\mathrm{I}} > |\Omega'|_{\mathrm{I}}$ and $|\Omega|_{\mathrm{S}} > |\Omega'|_{\mathrm{S}}$.

3.16 *Temporary*

$$\frac{\Omega \approx_{\mathrm{I}} n}{\Omega\, d \approx_{\mathrm{D}} n} \qquad \frac{\Omega \approx_{\mathrm{D}} n}{\Omega\, b_0' \approx_{\mathrm{D}} 2n} \qquad \frac{}{z \approx_{\mathrm{D}} 0} \qquad \frac{\Omega \approx_{\mathrm{I}} n}{\Omega\, s \approx_{\mathrm{D}} n+1}$$

$$\frac{\Omega \approx_{\mathrm{D}} n}{\Omega\, (d \bullet b_0') \approx_{\mathrm{D}} 2n} \qquad \frac{\Omega \approx_{\mathrm{I}} n}{\Omega\, (b_1 \bullet s) \approx_{\mathrm{D}} 2n+2} \qquad \frac{\Omega \approx_{\mathrm{I}} n}{\Omega\, (b_0 \bullet s) \approx_{\mathrm{D}} 2n+1}$$

As the first rule exhibits, a binary number and its head-unary form denote the same value. The last three rules are included by analogy with the $e \bullet b_1$ and $i \bullet b_0$ rules of the $\approx_{\mathrm{I}}$ relation.

False claim 3.39 (Small-step adequacy of decrements).
Preservation *If* $\Omega \approx_{\mathrm{D}} n$ *and* $\Omega \longrightarrow \Omega'$*, then* $\Omega' \approx_{\mathrm{D}} n$.
Progress *If* $\Omega \approx_{\mathrm{D}} n$*, then either:*

- $\Omega \longrightarrow \Omega'$;
- $n = 0$ *and* $\Omega = z$*; or*
- $n > 0$ *and* $\Omega = \Omega'\, s$ *for some* Ω' *such that* $\Omega' \approx_{\mathrm{I}} n - 1$.

Productivity *If* $\Omega \approx_{\mathrm{D}} n$*, then every rewriting sequence from* Ω *has a finite prefix* $\Omega \Longrightarrow \Omega'$ *such that either:*

- $n = 0$ *and* $\Omega' = z$*; or*
- $n > 0$ *and* $\Omega' = \Omega_0'\, s$*, for some* Ω_0' *such that* $\Omega_0' \approx_{\mathrm{I}} n - 1$.

Proof. The fine-grained atomicity of ordered rewriting, together with the use of alternative conjunction in the recursively defined propositions e, b_0, b_0', and b_1, causes both preservation and progress properties to fail.

As a counterexample to preservation, $e\, d \approx_{\mathrm{D}} 0$ and $e\, d \longrightarrow (z \mathbin{/} d)\, d$, but $(z \mathbin{/} d)\, d \approx_{\mathrm{D}} 0$ does *not* hold.

Even worse, the fine-grained atomicity of ordered rewriting means that computations can enter stuck states, which shouldn't have denotations and which would violate progress if they were somehow assigned denotations. For example, $e\, d \approx_{\mathrm{D}} 0$ and $e\, d \longrightarrow (e \bullet b_1 \mathbin{/} i)\, d \not\longrightarrow$.

$e\, i \approx_{\mathrm{I}} 0$ and $e\, i \longrightarrow (z \mathbin{/} d)\, i \not\longrightarrow$ □

These binary counters may also be equipped with a decrement operation. Although "decrement" is a convenient name for this operation, it is more accurate to implement decrements by converting the binary representation to what might be called *head-unary form*: an ordered context Ω is said to be in head-unary form if either: $\Omega = z$; or $\Omega = \Omega_0\, s$ for some binary representation Ω_0.

Similar to how the atom i is used to describe increments, a decrement is initiated by appending an atom d to the counter; d is then processed from right to left by the counter's bits. To support this, the definitions of e, b_0, and

b_1 are revised

$$\begin{aligned}
e &\triangleq (e \bullet b_1 \mathbin{/} i) \mathbin{\&} (\cdots \mathbin{/} d) \\
b_0 &\triangleq (b_1 \mathbin{/} i) \mathbin{\&} (\cdots \mathbin{/} d) \\
b_1 &\triangleq (i \bullet b_0 \mathbin{/} i) \mathbin{\&} (\cdots \mathbin{/} d)
\end{aligned}$$

To initiate a decrement of a counter Ω, we append the uninterpreted atom d to the counter, forming $\Omega\, d$.

To implement the decrement operation, we instead

Although "decrement" is a convenient name for this operation, it is perhaps more accurate to think of this operation as putting the binary representation into a head-unary form: either z or $\Omega'\, s$ for some $\Omega' \approx_{\mathrm{I}} n - 1$.

- If $\Omega \approx_{\mathrm{I}} n$, then:
  - $n = 0$ if, and only if, $\Omega\, d \Longrightarrow z$; and
  - $n > 0$ implies $\Omega\, d \Longrightarrow \Omega'\, s$ for some Ω' such that $\Omega' \approx_{\mathrm{I}} n - 1$; and
  - $\Omega\, d \Longrightarrow \Omega'\, s$ implies $n > 0$ and $\Omega' \approx_{\mathrm{I}} n - 1$.

$$\begin{aligned}
e &\triangleq (e \bullet b_1 \mathbin{/} i) \mathbin{\&} (z \mathbin{/} d) \\
b_0 &\triangleq (b_1 \mathbin{/} i) \mathbin{\&} (d \bullet b_0' \mathbin{/} d) \\
b_0' &\triangleq (z \backslash z) \mathbin{\&} (s \backslash b_1 \bullet s) \\
b_1 &\triangleq (i \bullet b_0 \mathbin{/} i) \mathbin{\&} (b_0 \bullet s \mathbin{/} d)
\end{aligned}$$

$e \triangleq \cdots \mathbin{\&} (z \mathbin{/} d)$ Because the counter e represents 0, its head-unary form is simply z.

$b_1 \triangleq \cdots \mathbin{\&} (b_0 \bullet s \mathbin{/} d)$ Because the counter $\Omega\, b_1$ represents $2n{+}1 > 0$ when Ω represents n, its head-unary form must then be the successor of a counter representing $2n$ – that is, $\Omega\, b_0\, s$.

$b_0 \triangleq \cdots \mathbin{\&} (d \bullet b_0' \mathbin{/} d)$ The natural number that the counter $\Omega\, b_0$ represents could be either zero or positive, depending on whether Ω represents zero or a positive natural number. Thus, to put $\Omega\, b_0$ into head-unary form, we first put Ω into head-unary form and then use b_0' to branch on the result.

$b_0' \triangleq (z \backslash z) \mathbin{\&} (s \backslash b_1 \bullet s)$ If the head-unary form of Ω is z, then $\Omega\, b_0$ also represents 0 and has head-unary form z. Otherwise, if the head-unary form of Ω is $\Omega'\, s$ for some $\Omega' \approx_{\mathrm{I}} n'$, then $\Omega\, b_0$ represents $2n' + 2$ and has head-unary form $\Omega'\, b_1\, s$.

Decrements actually do not literally decrement the counter, but instead put it into a "head unary" form in which the couter is either z or s with a binary counter beneath.

We will use the same strategy for proving the adequacy of decrements as we did for increments: Characterize the valid states

$$\frac{\Omega \approx_{\text{I}} n}{\Omega\, d \approx_{\text{D}} n} \qquad \frac{\Omega \approx_{\text{D}} n}{\Omega\, b_0' \approx_{\text{D}} 2n} \qquad \frac{}{z \approx_{\text{D}} 0} \qquad \frac{\Omega \approx_{\text{I}} n}{\Omega\, s \approx_{\text{D}} n+1}$$

$$\frac{}{e \bullet b_1 \,/\, i \approx_{\text{I}} 0} \qquad \frac{}{z \,/\, d \approx_{\text{I}} 0}$$

$$\frac{\Omega \approx_{\text{I}} n}{\Omega\, (b_1 \,/\, i) \approx_{\text{I}} 2n} \qquad \frac{\Omega \approx_{\text{I}} n}{\Omega\, (d \bullet b_0' \,/\, d) \approx_{\text{I}} 2n} \qquad \frac{\Omega \approx_{\text{I}} n}{\Omega\, (d \bullet b_0') \approx_{\text{D}} 2n}$$

$$\frac{\Omega \approx_{\text{I}} n}{\Omega\, (i \bullet b_0 \,/\, i) \approx_{\text{I}} 2n+1} \qquad \frac{\Omega \approx_{\text{I}} n}{\Omega\, (b_0 \bullet s \,/\, d) \approx_{\text{I}} 2n+1} \qquad \frac{\Omega \approx_{\text{I}} n}{\Omega\, (i \bullet b_0) \approx_{\text{I}} 2n+2} \qquad \frac{\Omega \approx_{\text{I}} n}{\Omega\, (b_0 \bullet s) \approx_{\text{D}} 2n+1}$$

$$\frac{\Omega \approx_{\text{D}} n}{\Omega\, (z \setminus z) \approx_{\text{D}} 2n} \qquad \frac{\Omega \approx_{\text{D}} n}{\Omega\, (s \setminus b_1 \bullet s) \approx_{\text{D}} 2n} \qquad \frac{\Omega \approx_{\text{I}} n}{\Omega\, (b_1 \bullet s) \approx_{\text{D}} 2n+2}$$

Notice that $e\,s\,b_0' \approx_{\text{D}} 0$ but $e\,s\,b_0' \Longrightarrow e\,b_1\,s \approx_{\text{D}} 1$. If we revise the s rule to use $n+1$, then a different problem arises: $e\,b_1\,d \approx_{\text{D}} 0$ but $e\,b_1\,d \Longrightarrow e\,b_0\,s \approx_{\text{D}} 1$.

THEOREM 3.40 (Adequacy). *If $\Omega \approx_{\text{I}} n$, then:*

- *$n = 0$ if and only if $\Omega\, d \Longrightarrow z$; and*
- *$n > 0$ implies $\Omega\, d \Longrightarrow \Omega'\, s$ and $\Omega' \approx_{\text{I}} n-1$;*
- *$\Omega\, d \Longrightarrow \Omega'\, s$ implies $n > 0$ and $\Omega' \approx_{\text{I}} n-1$.*

Proof. □

THEOREM 3.41 (Small-step adequacy).
Preservation *If $\Omega \approx_{\text{D}} n$ and $\Omega \longrightarrow \Omega'$, then $\Omega' \approx_{\text{D}} n$.*
Progress *If $\Omega \approx_{\text{D}} n$, then either:*

- *$\Omega \longrightarrow \Omega'$;*
- *$n = 0$ and $\Omega = z$; or*
- *$n = n' + 1$ and $\Omega = \Omega'\, s$ for some n' and Ω' such that $\Omega' \approx_{\text{I}} n'$.*

3.17

COROLLARY 3.42 (Big-step adequacy of decrements). *If $\Omega \approx_{\text{D}} n$, then:*

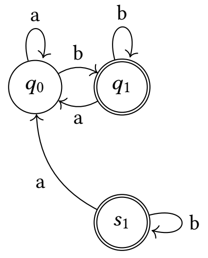
- *$\Omega \Longrightarrow \underset{\rightarrow}{z}$ if, and only if, $n = 0$;*
- *$\Omega \Longrightarrow \Omega'\, \underset{\rightarrow}{s}$ for some Ω' such that $\Omega' \approx_{\text{I}} n-1$, if $n > 0$; and*
- *$\Omega \Longrightarrow \Omega'\, \underset{\rightarrow}{s}$ only if $n > 0$ and $\Omega' \approx_{\text{I}} n-1$.*

3.18

3.18.1 *Automata*

1. Traces do not imply DFA transitions:

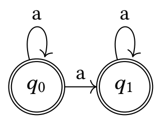
$$\hat{q}_0 \triangleq (a \backslash \hat{q}_0) \,\&\, (b \backslash \hat{q}_1) \,\&\, (\epsilon \backslash \top)$$
$$\hat{q}_1 \triangleq (a \backslash \hat{q}_0) \,\&\, (b \backslash \hat{q}_1) \,\&\, (\epsilon \backslash \mathbf{1})$$
$$\hat{s}_1 \triangleq (a \backslash \hat{q}_0) \,\&\, (b \backslash \hat{s}_1) \,\&\, (\epsilon \backslash \mathbf{1})$$



Notice that $b\,\hat{q}_0 \Longrightarrow \hat{q}_1 = \hat{s}_1$ but s_1 is not reachable from q_0. ($\hat{q}_1 = \hat{s}_1$ is proved coinductively.)

2. NFA bisimilarity does not imply equality of encodings:

$$\hat{q}_0 \triangleq (a \backslash (\hat{q}_0 \,\&\, \hat{q}_1)) \,\&\, (\epsilon \backslash \mathbf{1})$$
$$\hat{q}_1 \triangleq (a \backslash \hat{q}_1) \,\&\, (\epsilon \backslash \mathbf{1})$$



Notice that q_0 and q_1 are bisimilar, as witnessed by the reflexive closure of $\{(q_0, q_1)\}$. However, $\hat{q}_0 \neq \hat{q}_1$.

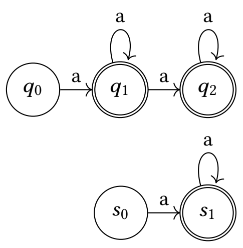
3. NFA similarity does not imply reduction. In the above example, NFA states q_0 and q_1 are bisimilar, andhence q_1 simulates q_0 (and vice versa). However, neither $\hat{q}_0 \Longrightarrow \hat{q}_1$ nor $\hat{q}_1 \Longrightarrow \hat{q}_0$ hold.

4. Even if an alternative, flatter encoding is used, NFA similarity does not imply reduction. Consider the following NFAs:

$$\hat{q}_0 \triangleq (a \backslash \hat{q}_1) \,\&\, (\epsilon \backslash \top)$$
$$\hat{q}_1 \triangleq (a \backslash \hat{q}_1) \,\&\, (a \backslash \hat{q}_2) \,\&\, (\epsilon \backslash \mathbf{1})$$
$$\hat{q}_2 \triangleq (a \backslash \hat{q}_2) \,\&\, (\epsilon \backslash \mathbf{1})$$

and

$$\hat{s}_0 \triangleq (a \backslash \hat{s}_1) \,\&\, (\epsilon \backslash \top)$$
$$\hat{s}_1 \triangleq (a \backslash \hat{s}_1) \,\&\, (\epsilon \backslash \mathbf{1})$$



As witnessed by the relation $\{(q_0, s_0), (q_1, s_1), (q_2, s_1)\}$, state s_0 simulates q_0. However, $\hat{q}_0 \not\Longrightarrow \hat{s}_0$. Essentially, similarity and reduction do not coincide because similarity is successor-congruent, whereas reduction is not $\backslash$-congruent.

5. Focusing with eager inversion does not solve this problem. For DFAs, we would be able to prove:

   - q and s are bisimular if, and only if, $\hat{q} = \hat{s}$.
   - $q \sim^{-1}\xrightarrow{a}\sim q'$ if, and only if, $a\,\hat{q} \longrightarrow \hat{q}'$.

6. For NFAs, we will be able to prove:

   - q and s are bisimular if, and only if, $\hat{q} \cong \hat{s}$.
   - $q \sim^{-1}\xrightarrow{a}\sim q'$ if, and only if, $a\,\hat{q} \cong^{-1}\longrightarrow\cong \hat{q}'$.

3.18.2 *Extended example: NFAs*

As an example of ordered rewriting, consider a specification of NFAs. Recall from **??** the NFA (repeated in the adjacent figure) that accepts exactly those words, over the alphabet $\Sigma = \{a, b\}$, that end with b. We may represent that NFA as a rewriting specification using a collection of recursive definitions, one for each of the NFA's states:[49]

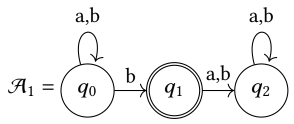


Figure 3.23: An NFA that accepts, from state q_0, exactly those words that end with b. (Repeated from ??.)

[49] Should I include $\mathbin{\&} (\epsilon \setminus \top)$?

$$\begin{aligned}\hat{q}_0 &\triangleq (a \setminus \hat{q}_0) \mathbin{\&} (b \setminus (\hat{q}_0 \mathbin{\&} \hat{q}_1)) \mathbin{\&} (\epsilon \setminus \top)\\ \hat{q}_1 &\triangleq (a \setminus \hat{q}_2) \mathbin{\&} (b \setminus \hat{q}_2) \mathbin{\&} (\epsilon \setminus \mathbf{1})\\ \hat{q}_2 &\triangleq (a \setminus \hat{q}_2) \mathbin{\&} (b \setminus \hat{q}_2) \mathbin{\&} (\epsilon \setminus \top)\end{aligned}$$

The NFA's acceptance of words is represented by the existence of traces. For example, because the word *ab* ends with b, a trace $\epsilon\; b\; a\; \hat{q}_0 \Longrightarrow \cdot$ exists. On the other hand, $\epsilon\; a\; b\; \hat{q}_0 \not\Longrightarrow \cdot$ because the word *ba* does not end with b.

More generally, an NFA $\mathcal{A} = (Q, \longrightarrow, F)$ over an input alphabet Σ can be represented as the ordered rewriting specification in which each state $q \in Q$ corresponds to a recursively defined proposition $\hat{q}$:

$$\hat{q} \triangleq \Big(\mathop{\&}_{a \in \Sigma} \big(a \setminus \&_{q'_a \in \Delta(q,a)}\, \hat{q}'_a\big)\Big) \mathbin{\&} \big(\epsilon \setminus \hat{F}(q)\big) \quad \text{where} \quad \hat{F}(q) = \begin{cases} \mathbf{1} & \text{if } q \in F \\ \top & \text{if } q \notin F\,. \end{cases}$$

After defining a representation, $\underline{w}$, of words w (see adjacent figure), we may state and prove that ordered rewriting under these definitions is sound and complete with respect to the NFA semantics given in ??.

$$\underline{\epsilon} = \cdot$$
$$\underline{a\,w} = \underline{w}\; a$$

Figure 3.24: Words as ordered contexts

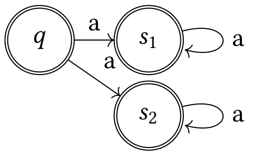
THEOREM 3.43. • *$q \xrightarrow{a} q'$ if, and only if, $a\,\hat{q} \Longrightarrow \hat{q}'$.*

- *$q \in F$ if, and only if, $\epsilon\,\hat{q} \Longrightarrow \cdot$.*

False claim 3.44. *Let $\mathcal{A} = (Q, \longrightarrow, F)$ be an NFA over the input alphabet Σ. Then:*

- $q \xrightarrow{a}\sim s'$ *if, and only if,* $a\,\hat{q} \Longrightarrow \hat{s}'$.
- $q \sim s$ *if, and only if,* $\hat{q} = \hat{s}$.

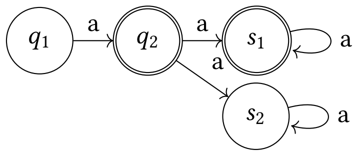
Counterexample. First, $q \sim s$ does not imply $\hat{q} = \hat{s}$. Consider the following NFA and its corresponding definitions:



$$\hat{q} \triangleq (a \setminus \hat{s}_1) \mathbin{\&} (a \setminus \hat{s}_2) \mathbin{\&} (\epsilon \setminus \mathbf{1})$$
$$\hat{s}_1 \triangleq (a \setminus \hat{s}_1) \mathbin{\&} (\epsilon \setminus \mathbf{1})$$
$$\hat{s}_2 \triangleq (a \setminus \hat{s}_2) \mathbin{\&} (\epsilon \setminus \mathbf{1})$$

Observe that the universal binary relation on states is a bisimulation: every state has an a-successor and every state is an accepting state. Therefore, all pairs of states are bisimilar; in particular, $q \sim s_1$. However, $\hat{q} \neq \hat{s}_1$.

Second, $a\,\hat{q} \Longrightarrow \hat{s}'$ does not imply $q \xrightarrow{a}\sim s'$. Consider the following NFA and its corresponding definitions:



$$\hat{q}_1 \triangleq (a \setminus \hat{q}_2) \mathbin{\&} (\epsilon \setminus \top)$$
$$\hat{q}_2 \triangleq (a \setminus \hat{s}_1) \mathbin{\&} (a \setminus \hat{s}_2) \mathbin{\&} (\epsilon \setminus \mathbf{1})$$
$$\hat{s}_1 \triangleq (a \setminus \hat{s}_1) \mathbin{\&} (\epsilon \setminus \mathbf{1})$$
$$\hat{s}_2 \triangleq (a \setminus \hat{s}_2) \mathbin{\&} (\epsilon \setminus \top)$$

Observe that $a\,\hat{q}_1 \Longrightarrow \hat{q}_2 \Longrightarrow \hat{s}_1$. However, $q_2 \nsim s_1$, and so $q_1 \xrightarrow{a}\sim s_1$ does *not* hold. To see why $q_2 \nsim s_1$, notice that $q_2 \xrightarrow{a} s_2 \notin F$ is not matched from s_1, which has only $s_1 \xrightarrow{a} s_1 \in F$. □

Definition 3.1. A binary relation $\mathcal{R}$ on states is a simulation if:

- $s \mathrel{\mathcal{R}^{-1}} \xrightarrow{a} q'$ implies $s \xrightarrow{a} \mathcal{R}^{-1}\, q'$; and
- $s \mathrel{\mathcal{R}^{-1}} q \in F$ implies $s \in F$.

Similarity, $\lesssim$, is the largest simulation.

Lemma 3.45. *If* $\hat{q} \Longleftarrow \hat{s}$, *then* $q \lesssim s$.

Proof. We must check two properties:

- Suppose that $\hat{s} \Longrightarrow \hat{q}$ and $q \xrightarrow{a} q'_a$ for some state q'_a; we must show that $s \xrightarrow{a} s'_a$ and $\hat{s}'_a \Longleftarrow \hat{q}'_a$, for some state s'_a. According to the definition, the definiens of $\hat{q}$ contains a clause $(a \setminus \hat{q}'_a)$. Because $\hat{s} \Longrightarrow \hat{q}$, the definiens of $\hat{s}$ also contains the clause $(a \setminus \hat{q}'_a)$. It follows that $s \xrightarrow{a} q'_a$ and $\hat{q}'_a \Longleftarrow \hat{q}'_a$.
- Suppose that $\hat{s} \Longrightarrow \hat{q}$ and $q \in F$; we must show that $s \in F$. According to the definition, the definiens of $\hat{q}$ contains a clause $(\epsilon \setminus \mathbf{1})$. Because $\hat{s} \Longrightarrow \hat{q}$, the definiens of $\hat{s}$ also contains the clause $(\epsilon \setminus \mathbf{1})$. It follows that $s \in F$. □

Theorem 3.46 (Adequacy). *Let $\mathcal{A} = (Q, \longrightarrow, F)$ be an NFA over the input alphabet Σ. If $q \xrightarrow{a} q'$, then $a\,\hat{q} \Longrightarrow \hat{q}'$. Moreover, if $a\,\hat{q} \Longrightarrow \hat{s}'$, then $q \xrightarrow{a}\gtrsim s'$.*

Proof. The first part follows by construction.

To prove the second part, suppose $a\,\hat{q} \Longrightarrow \hat{s}'$. By the lemma, $\hat{q} \Longrightarrow (a \setminus B)\,\Omega'_a$ and $B\,\Omega'_a \Longrightarrow \hat{s}'$ for some B and Ω'_a. By inversion, $\Omega'_a = \cdot$ and $B = \hat{q}'_a$ for some state q'_a such that $q \xrightarrow{a} q'_a$. Therefore, $\hat{q}'_a \Longrightarrow \hat{s}'$. By the lemma, $s' \lesssim q'_a$ and so $q \xrightarrow{a}\gtrsim s'$. □

Theorem 3.47 (Adequacy). *Let $\mathcal{A} = (Q, \longrightarrow, F)$ be an NFA over the input alphabet Σ. Then:*

1. *If $q \xrightarrow{a}\sim s'$, then $a\,\hat{q} \Longrightarrow \hat{s}'$.*

2. *If $q \sim s$, then $\hat{q} = \hat{s}$.*

3. *If $\hat{q} = \hat{s}$, then $q \sim s$.*

4. *If $a\,\hat{q} \Longrightarrow \hat{s}'$, then $q \xrightarrow{a}\sim s'$.*

Proof. 1. Suppose that $q \xrightarrow{a} q' \sim s'$; we must show that $a\,\hat{q} \Longrightarrow \hat{s}'$. By construction, $a\,\hat{q} \Longrightarrow \hat{q}'$. It follows from part [...] that $\hat{q}' = \hat{s}'$, and so $a\,\hat{q} \Longrightarrow \hat{s}'$.

2. Suppose $q \sim s$; we must show that $\hat{q} = \hat{s}$.

   - Choose an arbitrary symbol $a \in \Sigma$. If $q \xrightarrow{a} q'_a$, then there exists an NFA state s'_a such that $s \xrightarrow{a} s'_a \sim^{-1} q'_a$, and, by the coinductive hypothesis, $\hat{q}'_a = \hat{s}'_a$. Conversely, if $s \xrightarrow{a} s'_a$, then there exists an NFA state q'_a such that $q \xrightarrow{a} q'_a$ and $\hat{q}'_a = \hat{s}'_a$.
   - Also, q is an accepting state if and only if s is an accepting state.

   Therefore, the definientia of $\hat{q}$ and $\hat{s}$ are equal, and, by the equirecursive interpretation of definitions, so are the definienda $\hat{q}$ and $\hat{s}$.

3. Suppose that $\hat{s} = \hat{q}$ and $q \xrightarrow{a} q'$; we must show that $s \xrightarrow{a} s'$ and $\hat{s}' = \hat{q}'$, for some NFA state s'. By its definition, the definiens of $\hat{q}$ therefore contains the clause $(a \setminus \hat{q}')$. Because $\hat{s} = \hat{q}$, the definiens of $\hat{s}$ must also contain a clause $(a \setminus \hat{s}')$ for some state s' such that $s \xrightarrow{a} s'$ and $\hat{s}' = \hat{q}'$.

   Symmetrically, if $\hat{q} = \hat{s}$ and $s \xrightarrow{a} s'$, then $q \xrightarrow{a} q'$ and $\hat{q}' = \hat{s}'$, for some state q'.

4. Suppose $a\,\hat{q} \Longrightarrow \hat{q}'$. By the lemma, $a\,\hat{q} \Longrightarrow (a \setminus B)\,\Omega'_a$ and $B\,\Omega'_a \Longrightarrow \hat{q}'$ for some B and Ω'_a. By inversion, $B = \hat{q}'_a$ and $\Omega'_a = \cdot$. Therefore, $\hat{q}'_a \Longrightarrow \hat{q}'$. How to show that $q'_a \sim q'$?

□

Proof. By coinduction on $q \sim s$.

- Suppose $\hat{s} = \hat{q}$ and $q \xrightarrow{a} q'$; we must show that $s \xrightarrow{a} s'$ and $\hat{s}' = \hat{q}'$ for some NFA state s'. It follows from the coinductive hypothesis that $a\,\hat{s} = a\,\hat{q} \Longrightarrow \hat{q}'$.

□

Proof. In the left-to-right directions, by unrolling the definition of $\hat{q}$ (and a structural induction on the word w).

In the right-to-left directions, by structural induction on the given trace, using the following lemma:

If $a\,\Omega \Longrightarrow \Omega''$ and there is no Ω''_0 for which $\Omega'' = a\,\Omega''_0$, then $\Omega \Longrightarrow (a \setminus B)\,\Omega'$ for some B and Ω' such that $B\,\Omega' \Longrightarrow \Omega''$.

Assume that $a\,\hat{q} \Longrightarrow \hat{q}'$. Using the above lemma, $\hat{q} \Longrightarrow (a \setminus B)\,\Omega'$ for some B and Ω' such that $B\,\Omega' \Longrightarrow \hat{q}'$. By inversion on the trace from $\hat{q}$, it must be that $B = \&_{q'_a \in \Delta(q,a)} \hat{q}'_a$ and $\Omega' = \cdot$. Further inversion on the trace from $B\,\Omega'$ establishes that $q' \in \Delta(q,a)$ and hence $q \xrightarrow{a} q'$. □

$$\hat{q} \triangleq \left(\underset{a\in\Sigma}{\&}(a \setminus \hat{q}'_a \bullet \hat{v}_a)\right) \& (\epsilon \setminus \hat{\rho}(q)) \quad \text{where } q'_a = \delta(q,a) \text{ and } v_a = \sigma(q,a) \text{ and } v = \rho(q)$$

3.18.3 *Extended example: Binary representation of natural numbers*

As a second example, consider a rewriting specification of the binary representation of natural numbers with increment and decrement operations.

For this specification, a natural number is represented in binary by an ordered context consisting of a big-endian sequence of atoms b_0 and b_1, prefixed by the atom e; leading b_0s are permitted. For example, both $\Omega = e\,b_1$ and $\Omega = e\,b_0\,b_1$ are valid binary representations of the natural number 1.

More generally, let $\mathbb{V}(-)$ be the partial function from ordered contexts to natural numbers defined as follows; we say that the ordered context Ω *represents* natural number n if $\mathbb{V}(\Omega) = n$.

$$\begin{aligned} \mathbb{V}(e) &= 0 \\ \mathbb{V}(\Omega\,b_0) &= 2\mathbb{V}(\Omega) \\ \mathbb{V}(\Omega\,b_1) &= 2\mathbb{V}(\Omega) + 1 \end{aligned}$$

The partial function $\mathbb{V}(-)$ defines an adequate representation because, up to leading b_0s, the natural numbers and valid binary representations (*i.e.*, the domain of definition of $\mathbb{V}(-)$) are in bijective correspondence.

THEOREM 3.48 (Representational adequacy). *For all natural numbers $n \in \mathbb{N}$, there exists a context Ω such that $\mathbb{V}(\Omega) = n$. Moreover, if $\mathbb{V}(\Omega_1) = n$ and $\mathbb{V}(\Omega_2) = n$, then Ω_1 and Ω_2 are identical up to leading b_0s.*

Proof. The first part follows by induction on the natural number n; the second part follows by induction on the structure of the contexts Ω_1 and Ω_2. □

Next, we may describe an increment operation on these binary representations as an ordered rewriting specification; because of these increments, [...]. To indicate that an increment should be performed, a new, uninterpreted atom i is introduced. The previously uninterpreted atoms e, b_0, and b_1 are now given mutually recursive definitions that describe their interactions with i.

$e \triangleq e \bullet b_1 \;/\; i$ To increment the counter e, introduce b_1 as a new most significant bit, resulting in the counter $e\,b_1$. That is, $e\,i \Longrightarrow e\,b_1$. Having started at value 0 (*i.e.*, $\mathbb{V}(e) = 0$), an increment results in value 1 (*i.e.*, $\mathbb{V}(e\,b_1) = 1$).

$b_0 \triangleq b_1 \;/\; i$ To increment a counter that ends with least significant bit b_0, simply flip that bit to b_1. That is, $\Omega\,b_0\,i \Longrightarrow \Omega\,b_1$. Having started at value $2n$ (*i.e.*, $\mathbb{V}(\Omega\,b_0) = 2\mathbb{V}(\Omega)$), an increment results in value $2n + 1$ (*i.e.*, $\mathbb{V}(\Omega\,b_1) = 2\mathbb{V}(\Omega) + 1$).

$b_1 \triangleq i \bullet b_0 \;/\; i$ To increment a counter that ends with least significant bit b_1, flip that bit to b_0 and propagate the increment on to the more significant bits as a carry. That is, $\Omega\,b_1\,i \Longrightarrow \Omega\,i\,b_0$. Having started at value $2n+1$ (*i.e.*, $\mathbb{V}(\Omega\,b_1) = 2\mathbb{V}(\Omega) + 1$), an increment results in value $2n + 2 = 2(n + 1)$ (*i.e.*, $\mathbb{V}(\Omega\,i\,b_0) = 2\mathbb{V}(\Omega) + 1$).

As an example, consider incrementing $e\,b_1$ twice, as captured by the state $e\,b_1\,i\,i$. First, processing of the leftmost increment begins: the least significant bit is flipped, and the increment is carried over to the more significant bits. This corresponds to the reduction $e\,b_1\,i\,i \Longrightarrow e\,i\,b_0\,i$. Next, either of the two remaining increments may be processed – that is, either $e\,i\,b_0\,i \Longrightarrow e\,b_1\,b_0\,i$ or $e\,i\,b_0\,i \Longrightarrow e\,i\,b_1$.

$$\begin{tikzcd}
 & & e\,b_1\,b_0\,i \arrow[dr, Rightarrow] & \\
e\,b_1\,i\,i \arrow[r, Rightarrow] & e\,i\,b_0\,i \arrow[ur, Rightarrow] \arrow[dr, Rightarrow] \arrow[rr, Rightarrow, dashed] & & e\,b_1\,b_1 \\
 & & e\,i\,b_1 \arrow[ur, Rightarrow] &
\end{tikzcd}$$

$$\begin{aligned}
&e\,b_1\,i\,i && e\,b_1\,i\,i \\
&\Longrightarrow e\,i\,b_0\,i && \longrightarrow e\,(i \bullet b_0 \;/\; i)\,i\,i \longrightarrow e\,(i \bullet b_0)\,i \longrightarrow e\,i\,b_0\,i \\
&\Longrightarrow e\,b_1\,b_0\,i && \longrightarrow (e \bullet b_1 \;/\; i)\,i\,b_0\,i \longrightarrow (e \bullet b_1)\,b_0\,i \longrightarrow e\,b_1\,b_0\,i \\
&\Longrightarrow e\,b_1\,b_1 && \longrightarrow e\,b_1\,(b_1 \;/\; i)\,i \longrightarrow e\,b_1\,b_1
\end{aligned}$$

$$\frac{}{e \approx_{\mathrm{V}} 0} \qquad \frac{\Omega \approx_{\mathrm{V}} n}{\Omega\, b_0 \approx_{\mathrm{V}} 2n} \qquad \frac{\Omega \approx_{\mathrm{V}} n}{\Omega\, b_1 \approx_{\mathrm{V}} 2n+1}$$

THEOREM 3.49 (Adequacy). *If $\Omega \approx_{\mathrm{V}} n$ and $\Omega\, i \Longrightarrow \Omega'$, then $\Omega' \Longrightarrow\!\approx_{\mathrm{V}} n+1$.*

Proof. • Suppose that $e\, i \Longrightarrow \Omega'$; we must show that $\Omega' \Longrightarrow\!\approx_{\mathrm{V}} 1$.

• Suppose that $\Omega\, b_0\, i \Longrightarrow \Omega'$ and $\Omega \approx_{\mathrm{V}} n$; we must show that $\Omega' \Longrightarrow\!\approx_{\mathrm{V}} 2n$.

□

$$\frac{\Omega \approx_{\mathrm{V}} n}{\Omega \approx_{\mathrm{I}} n} \qquad \frac{\Omega \approx_{\mathrm{I}} n}{\Omega\, i \approx_{\mathrm{I}} n+1} \qquad \frac{\Omega \approx_{\mathrm{I}} n}{\Omega\, b_0 \approx_{\mathrm{I}} 2n} \qquad \frac{\Omega \approx_{\mathrm{I}} n}{\Omega\, b_1 \approx_{\mathrm{I}} 2n+1}$$

$$\frac{\Omega_L\, \alpha\, \Omega_R \approx_{\mathrm{I}} n \quad (\alpha \triangleq A) \in \Sigma}{\Omega_L\, A\, \Omega_R \approx_{\mathrm{I}} n}$$

THEOREM 3.50 (Preservation). *If $\Omega \approx_{\mathrm{I}} n$ and $\Omega \longrightarrow \Omega'$, then $\Omega' \Longrightarrow\!\approx_{\mathrm{I}} n$.*

Proof. • Suppose that $\Omega_0 \approx_{\mathrm{I}} n$ and $\Omega = \Omega_0\, i \longrightarrow \Omega'$; we must show that $\Omega' \Longrightarrow\!\approx_{\mathrm{I}} n+1$.

– Consider the case in which $\Omega_0 \longrightarrow \Omega_0'$ and $\Omega' = \Omega_0'\, i$. By the inductive hypothesis, $\Omega_0' \Longrightarrow\!\approx_{\mathrm{I}} n$. From the increment rule, it follows that $\Omega' = \Omega_0'\, i \Longrightarrow\!\approx_{\mathrm{I}} n+1$.

– Consider the case in which $\Omega_0 = \Omega_L\, (A_0 \mathbin{/} i)$ and $\Omega_L\, \alpha \approx_{\mathrm{I}} n$ and $\Omega' = \Omega_L\, A_0$ such that $(\alpha \triangleq A_0 \mathbin{/} i) \in \Sigma$. There are three subcases:

  * Consider the subcase in which $\alpha = b_0$ and $n = 2n_0$ and $\Omega_L \approx_{\mathrm{I}} n_0$. By inversion on the signature, $A_0 = b_1$. It follows that $\Omega' = \Omega_L\, b_1 \approx_{\mathrm{I}} 2n_0 + 1 = n+1$.
  * Consider the subcase in which $\alpha = b_1$ and $n = 2n_0 + 1$ and $\Omega_L \approx_{\mathrm{I}} n_0$. By inversion on the signature, $A_0 = i \bullet b_0$. It follows that $\Omega' = \Omega_L\, (i \bullet b_0) \longrightarrow \Omega_L\, i\, b_0 \approx_{\mathrm{I}} 2(n_0+1) = n+1$.
  * Consider the subcase in which $\alpha = e$ and $n = 0$ and $\Omega_L = \cdot$. By inversion on the signature, $A_0 = e \bullet b_1$. It follows that $\Omega' = e \bullet b_1 \longrightarrow e\, b_1 \approx_{\mathrm{I}} 1 = n+1$.

□

THEOREM 3.51 (Progress). *If $\Omega \approx_{\mathrm{I}} n$, then either: $\Omega \longrightarrow \Omega'$ for some Ω'; or $\Omega \approx_{\mathrm{V}} n$.*

$$\frac{}{z \approx_{\mathrm{D}} 0} \qquad \frac{\Omega \approx_{\mathrm{I}} n}{\Omega\, s \approx_{\mathrm{D}} n+1} \qquad \frac{\Omega \approx_{\mathrm{I}} n}{\Omega\, d \approx_{\mathrm{D}} n} \qquad \frac{\Omega \approx_{\mathrm{D}} n}{\Omega\, b_0' \approx_{\mathrm{D}} 2n}$$

$$\frac{\Omega_L\, \alpha\, \Omega_R \approx_{\mathrm{D}} n \quad (\alpha \triangleq A) \in \Sigma}{\Omega_L\, A\, \Omega_R \approx_{\mathrm{D}} n}$$

$\Omega' \approx_{\mathrm{D}} n$ if, and only if, $\Omega\, d \Longrightarrow \Omega'$ for some Ω such that $\Omega \approx_{\mathrm{I}} n$.

THEOREM 3.52 (Preservation). *If $\Omega \approx_{\mathrm{D}} n$ and $\Omega \Longrightarrow \Omega'$, then $\Omega' \approx_{\mathrm{D}} n$.*

THEOREM 3.53 (Progress). *If $\Omega \approx_{\mathrm{D}} n$, then either:*

- *$\Omega \longrightarrow \Omega'$ for some Ω';*
- *$\Omega = \Omega'\, s$ and $n = n' + 1$ and $\Omega' \approx_{\mathrm{I}} n'$ for some Ω' and n'; or*
- *$\Omega = z$ and $n = 0$.*

3.18.4 *Examples*

- Alternative choreography – how are these related?

$$
\begin{aligned}
&p \triangleq (i \bullet p \mathbin{/} \underleftarrow{i}) \mathbin{\&} (d \bullet p' \mathbin{/} \underleftarrow{d})\\
&p' \triangleq (\underrightarrow{z} \setminus \underrightarrow{z}) \mathbin{\&} (\underrightarrow{s} \setminus p \bullet \underrightarrow{s})\\
&i \triangleq (\underrightarrow{e} \setminus \underrightarrow{e} \bullet \underrightarrow{b}_1) \mathbin{\&} (\underrightarrow{b}_0 \setminus \underrightarrow{b}_1) \mathbin{\&} (\underrightarrow{b}_1 \setminus i \bullet \underrightarrow{b}_0)\\
&d \triangleq (\underrightarrow{e} \setminus \underrightarrow{z}) \mathbin{\&} (\underrightarrow{b}_0 \setminus d \bullet b_0') \mathbin{\&} (\underrightarrow{b}_1 \setminus \underrightarrow{b}_0 \bullet \underrightarrow{s})\\
&b_0' \triangleq (\underrightarrow{z} \setminus \underrightarrow{z}) \mathbin{\&} (\underrightarrow{s} \setminus \underrightarrow{b}_1 \bullet \underrightarrow{s})
\end{aligned}
$$

$$\frac{\Omega \approx_{\mathrm{I}} n}{\Omega\, d \approx_{\mathrm{D}} n}$$

If $\Omega\, \underleftarrow{i} \longrightarrow \Omega'$, then $\underrightarrow{\Omega}\, i \longrightarrow \underrightarrow{\Omega}'$.

3.19

4 A formula-as-process interpretation of ordered rewriting

In chapter 1, we saw that string rewriting can be used to specify the dynamics of concurrent systems, but that those specifications are quite abstract.[1] Even the operational semantics is left completely abstract: String rewriting axioms $w \longrightarrow w'$ state merely that a substring of the form w may be replaced, en masse, with w'. Nothing is said about this replacement is achieved – permitted rewritings just *happen*, as if a central, meta-level actor schedules an otherwise coordinates rewriting, with substrings and their constituent symbols as mere passive accessories[2].

[1] something about state-transformation model?

[2] word choice?

In the previous chapter, we presented a different rewriting framework, this one derived from the (focused) ordered sequent calculus and closely related to the **Lambek:AMM58** calculus.[3] Ordered rewriting, in both its unfocused and focused variants, still leaves the operational semantics abstract, as if a central, meta-level actor governs rewriting.

[3] **Lambek:AMM58**.

The string rewriting and (focused) ordered rewriting frameworks are both expressive enough to describe concurrency[4], but without concrete operational semantics neither framework is yet suitable for our ultimate goal – a method for [cleanly and mechanically] extracting local, message-passing implementations from concurrent specifications.

[4] See ????.

This chapter takes three significant steps toward this end.

- In **??**, we refine the focused ordered rewriting framework of the previous chapter into one that can be given a *formula-as-process* interpretation[5] in which [ordered] rewriting faithfully represents message-passing communication among concurrent processes that are arranged in a linear topology. In this way, the formula-as-process interpretation assigns a concrete operational semantics to ordered rewriting, nudging it away from a state-transformation model of concurrency and toward a process-based model.

  Specifically, we show that atomic propositions may be interpreted as messages; (compound)[6] propositions, as processes; contexts, as configurations comprised of those messages and processes; and rewriting, as message-passing communication among a configuration's constituent processes. Perhaps surprisingly, only three small tweaks to the structure of proposi-

[5] This interpretation is *very* closely related to the process-as-formula view of concurrency put forth by **Miller:??Cervesato+Scedrov:IC09**. For us, however, the logical aspects, and propositions in particular, are conceptually prior to any notion of process, hence our use of the reversed *formula-as-process* terminology.

[6] $\top$ and $\mathbf{1}$ aren't exactly compound.

tions are needed to make this formula-as-process interpretation viable.

- With this new formula-as-process perspective and its accompanying message-passing semantics, (focused) ordered rewriting can be understood in terms of local interactions alone.

  By analogy with the π-calculus's operational semantics, the existing rewriting relation, $\longrightarrow$, serves as a reduction semantics, but now an alternative, labeled transition semantics can also be given (**??**). These two equivalent semantics [...][7]

  [7] ??

- Having established a formula-as-process refinement of the focused ordered rewriting framework that permits only local, message-passing interactions, we then revisit string rewriting specifications.

  In **????**, we describe, first informally and then formally, a method for operationalizing, or *choreographing*, string rewriting specifications within the formula-as-process ordered rewriting framework. Symbols in the specification's alphabet are uniquely mapped to propositions, thereby casting each symbol in one of two roles – either a message role or a process role.

  Not all such role assignments give rise to well-formed choreographies, however. But, for those that do, the resulting choreography adequately embeds[8] the string rewriting specification: the specification's axioms are in one-to-one correspondence with the choreography's derivable ordered rewritings, as we prove in **??**. Stated differently, the string rewriting specification and choreography will be (strongly) bisimilar, with the role assignment being a bisimulation between the two.

  [8] terminology?

Even though this chapter introduces a notion of process, it should be noted that computation is still driven by derivability and proof construction, not by proof reduction. Only in part will we begin to examine a proof-reduction account of concurrency.

In process calculi like the π-calculus, a reduction semantics often plays a lesser role than a labeled transition semantics does. But here the roles are reversed, with the labeled transition semantics taking a back seat, because of the reduction semantics's clearer connection to rewriting.

The other contribution of this **??** is the idea of choreographing string rewriting specifications into this formula-as-process refinement of ordered rewriting.

This formula-as-process interpretation nudges ordered rewriting away from a state-transformation model of concurrency toward a process-based model. Computation is still driven by derivability

As shown in chapter 1, string rewriting is a suitable framework for describing the dynamics of concurrent systems whose components have a monoidal structure. But these string rewriting descriptions are only abstract specifications from a state-transformation perspective – they lack a clear notion of local, decentralized execution and therefore give only a global characterization of the interactions between components.

String rewriting axioms $w \longrightarrow w'$ are strictly global in their phrasing, stating merely that any substring of the form w may be replaced, en masse, with w' – nothing is said about how this replacement is achieved. The string rewriting framework therefore implicitly suggests that a meta-level actor is responsible for coordinating and conducting the rewriting, with substrings and their constituent symbols as mere passive accessories[9].

[9] word choice?

In this ??, our goal is to move toward a lower level of abstraction.

As an example, recall from chapter 1 the string rewriting specification of a system that may transform strings that end with b into the empty string:

$$\frac{}{a\,b \longrightarrow b} \qquad \frac{}{b \longrightarrow \epsilon} \tag{4.1}$$

Accordingly, we refine the focused ordered rewriting framework of the previous chapter into one that can be given a *formula-as-process* interpretation in which [ordered] rewriting faithfully represents message-passing communication among processes that are arranged in a linear topology. Then, in ??, we describe how a string rewriting specification may be transformed into an ordered rewriting *choreography*.

In this ??, we refine the focused ordered rewriting framework of the previous chapter into one that can be given a *formula-as-process* interpretation in which rewriting faithfully represents message-passing communication among

As argued

Then we show how, given a mapping of symbols to either [...], choreographies can be generated from a string rewriting specification.

4.1 *Refining ordered rewriting: A formula-as-process interpretation*

In this section, we present the formula-as-process interpretation of focused ordered rewriting sketched above.

More specifically, (positive) atoms, a^+, may be viewed as messages, and negative propositions, A^-, as processes that receive and react to those messages. Ordered contexts, Ω, which consist of negative propositions and (positive) atoms, are then linear-topology run-time configurations of processes and messages. And positive propositions, A^+, which reify ordered contexts as propositions, are process expressions that reify configurations. Lastly, but most importantly, the rewriting relation, $\longrightarrow$, is viewed as a reduction semantics for message-passing communication among the processes in a configuration.

$\underleftarrow{a}^+$	*left-directed message*
$\underrightarrow{a}^+$	*right-directed message*
A^-	*message-passing process*
Ω	*run-time process configuration*
A^+	*configuration reified as an expression*

Table 4.1: A formula-as-process interpretation of polarized ordered propositions and contexts

Perhaps surprisingly, only three small tweaks to the structure of propositions are needed to make this formula-as-process reading viable.

- The (positive) atoms are now partitioned into two classes, left- and right-directed atoms, to allow us to identify the direction in which a message is flowing.

- The left- and right-handed implications are now restricted to have atomic premises with a complementary direction , so that they may then be cleanly interpreted as input processes that receive single incoming messages.
- The negative propositions are extended with recursively defined (negative) propositions, $\hat{p}^- \triangleq A^-$, that will correspond to recursively defined processes.

Together, the first two of these tweaks serve to provide a modicum of static typing for the otherwise untyped processes, as we will discuss in further detail in ??.

Positive atoms, as mentioned previously, are now partitioned into two classes, left- and right-directed atoms, to allow us to identify the direction in which a message is flowing. These directions are denoted by an arrow placed below the atom: Left-directed atoms, $\underset{\leftarrow}{a}^+$, are messages that are being sent to the left; right-directed atoms, $\underset{\rightarrow}{a}^+$, are messages that are being sent to the right.

Negative propositions, A^-, are processes that receive and react to those messages.

$$A^-, B^- ::= \underset{\rightarrow}{a}^+ \setminus B^- \mid B^- \mathbin{/} \underset{\leftarrow}{a}^+ \mid A^- \mathbin{\&} B^- \mid \top \mid {\uparrow}A^+ \mid \hat{p}^-$$

$\underset{\rightarrow}{a}^+ \setminus B^-$	*receive message $\underset{\rightarrow}{a}^+$ from the right*
$B^- \mathbin{/} \underset{\leftarrow}{a}^+$	*receive message $\underset{\leftarrow}{a}^+$ from the left*
$A^- \mathbin{\&} B^-$	*nondeterministic branching*
$\top$	*stuck process*
${\uparrow}A^+$	*quoted configuration*
$\hat{p}^-$	*call a recursively defined process*

Table 4.2: A formula-as-process interpretation of negative propositions

- Instead of the more general $A^+ \setminus B^-$ and $B^- \mathbin{/} A^+$, left- and right-handed implications are now restricted to be only $\underset{\rightarrow}{a}^+ \setminus B^-$ and $B^- \mathbin{/} \underset{\leftarrow}{a}^+$. These propositions are then interpreted as input processes: $\underset{\rightarrow}{a}^+ \setminus B^-$ is a process that waits to receive a message, $\underset{\rightarrow}{a}^+$, from its left-hand neighbor and then continues as B^-; symmetrically, $B^- \mathbin{/} \underset{\leftarrow}{a}^+$ is a process that awaits message $\underset{\leftarrow}{a}^+$ from its right-hand neighbor.

  Because implications are restricted to atoms with *complementary*
- The proposition $A^- \mathbin{\&} B^-$ is interpreted as a process that branches nondeterministically, continuing as either A^- or B^-. And $\top$, as the nullary form of $\&$, is a stuck process that cannot continue.
- The proposition ${\uparrow}A^+$ is interpreted as a process that holds a suspended, or quoted, configuration. When the process ${\uparrow}A^+$ is executed, it unfolds to that configuration.
- Lastly, $\hat{p}^-$ is a recursively defined negative proposition that is interpreted as a recursive process. We will discuss these recursive definitions in more detail in ??.

Ordered contexts, Ω, are interpreted as linear-topology run-time configurations of processes and the messages that pass between them.

$$\Omega, \Delta ::= \Omega_1\,\Omega_2 \mid \cdot \mid A^- \mid \underset{\leftarrow}{a}^+ \mid \underset{\rightarrow}{a}^+$$

Just as ordered contexts form a monoid over negative propositions and positive atoms, their formula-as-process interpretation forms a monoid over processes and messages. The monoid operation is now parallel, end-to-end composition of process configurations: $\Omega_1\,\Omega_2$ composes the configurations Ω_1 and Ω_2 so that they may interact along their mutual interface. The empty context, $(\cdot)$, is now the empty configuration.

$\Omega_1\,\Omega_2$	*parallel composition of configurations*
$(\cdot)$	*empty configuration*
A^-	*single-process configuration*
$\underset{\leftarrow}{a}^+$	*left-directed message*
$\underset{\rightarrow}{a}^+$	*right-directed message*

Table 4.3: A formula-as-process interpretation of contexts

As usual, we do not distinguish configurations that are equivalent up to the monoid's associativity and unit laws. This equivalence acts as an implicit structural congruence, of the sort found explicitly in the π-calculus's definition.

With the introduction of atom directions, it will often be useful to describe contexts that contain only atoms of one direction or the other. We will use the metavariables $\underset{\leftarrow}{\Omega}$ and $\underset{\rightarrow}{\Omega}$ for those contexts that contain only left- and right-directed atoms, respectively. More explicitly:

$$\underset{\leftarrow}{\Omega}, \underset{\leftarrow}{\Delta} ::= \underset{\leftarrow}{\Omega_1}\,\underset{\leftarrow}{\Omega_2} \mid \cdot \mid \underset{\leftarrow}{a}^+ \qquad \text{and} \qquad \underset{\rightarrow}{\Omega}, \underset{\rightarrow}{\Delta} ::= \underset{\rightarrow}{\Omega_1}\,\underset{\rightarrow}{\Omega_2} \mid \cdot \mid \underset{\rightarrow}{a}^+ \,.$$

Positive propositions, A^+, are processes that reify run-time configurations Ω as static expressions.

$$A^+, B^+ ::= \underset{\leftarrow}{a}^+ \mid \underset{\rightarrow}{a}^+ \mid A^+ \bullet B^+ \mid \mathbf{1} \mid {\downarrow}A^-$$

This reification is expressed as $\bullet\Omega = A^+$, as defined in the adjacent figure.

$$\begin{aligned} \bullet\underset{\leftarrow}{a}^+ &= \underset{\leftarrow}{a}^+ \\ \bullet\underset{\rightarrow}{a}^+ &= \underset{\rightarrow}{a}^+ \\ \bullet(\Omega_1\,\Omega_2) &= (\bullet\Omega_1) \bullet (\bullet\Omega_2) \\ \bullet(\cdot) &= \mathbf{1} \\ \bullet A^- &= {\downarrow}A^- \end{aligned}$$

Figure 4.1: Reifying a configuration as a process

The propositions $\underset{\leftarrow}{a}^+$ and $\underset{\rightarrow}{a}^+$ are the expressions for left- and right-directed messages.

The proposition $A^+ \bullet B^+$ reifies parallel, end-to-end composition of configurations: $A^+ \bullet B^+$ is now interpreted as the expression for a process that spawns a new process, A^+, and then continues as B^+. And $\mathbf{1}$ is interpreted as the expression for a process that immediately terminates, thereby reifying the empty configuration, $(\cdot)$.

The proposition ${\downarrow}A^-$ is interpreted as the expression for a quoted process: executing that expression will result in the running process A^-.

$\underset{\leftarrow}{a}^+$	*left-directed message*
$\underset{\rightarrow}{a}^+$	*right-directed message*
$A^+ \bullet B^+$	*parallel composition of A^+ and B^+*
$\mathbf{1}$	*terminating process*
${\downarrow}A^-$	*quoted process*

Table 4.4: A formula-as-process interpretation of positive propositions

4.1.1 *Focused ordered rewriting as message-passing communication*

The three tweaks introduced by the formula-as-process interpretation to the structure of propositions – especially atom directions and atomic premises for implications – trickle down through the right- and left-focus judgments used to define rewriting:

- First, because each positive atom is now marked with a direction, the ID^{a^+} rule that was previously part of the right-focus judgment's definition is

$$\frac{}{[a^+] \dashv\vdash a^+}\;\text{ID}^{a^+}$$

replaced by two similar rules – one for each direction:

$$\frac{}{[\underset{\leftarrow}{a}^+] \dashv\!\Vdash \underset{\leftarrow}{a}^+}\ \text{ID}^{\underset{\leftarrow}{a}^+} \qquad \text{and} \qquad \frac{}{[\underset{\rightarrow}{a}^+] \dashv\!\Vdash \underset{\rightarrow}{a}^+}\ \text{ID}^{\underset{\rightarrow}{a}^+}\ .$$

The other right-focusing rules remain unchanged.

- Second, because implications $\underset{\rightarrow}{a}^+ \setminus B^+$ and $B^- / \underset{\leftarrow}{a}^+$ are the only valid forms of implications, the left-focus judgement and its rules may be refined. Instead of $\Omega_L\ [A^-]\ \Omega_R \Vdash C^+$, which has arbitrary contexts to the left and right of A^-, the judgment is now $\underset{\rightarrow}{\Omega}_L\ [A^-]\ \underset{\leftarrow}{\Omega}_R \Vdash C^+$ – the left-hand context consists only of right-directed atoms, hence $\underset{\rightarrow}{\Omega}_L$; symmetrically, the right-hand context consists only of left-directed atoms, hence $\underset{\leftarrow}{\Omega}_R$. The left-focus rules for the left- and right-handed implications are also revised to

  $$\frac{\underset{\rightarrow}{\Omega}_L\ [B^-]\ \underset{\leftarrow}{\Omega}_R \Vdash C^+}{\underset{\rightarrow}{\Omega}_L\ \underset{\rightarrow}{a}^+\ [\underset{\rightarrow}{a}^+ \setminus B^-]\ \underset{\leftarrow}{\Omega}_R \Vdash C^+}\ \backslash\mathrm{L}' \qquad \text{and} \qquad \frac{\underset{\rightarrow}{\Omega}_L\ [B^-]\ \underset{\leftarrow}{\Omega}_R \Vdash C^+}{\underset{\rightarrow}{\Omega}_L\ [B^- / \underset{\leftarrow}{a}^+]\ \underset{\leftarrow}{a}^+\ \underset{\leftarrow}{\Omega}_R \Vdash C^+}\ /\mathrm{L}'\ ,$$

  which are the derivable from the earlier $\backslash$L and /L rules (??).

  $$\dfrac{\dfrac{}{[\underset{\rightarrow}{a}^+] \dashv\!\Vdash \underset{\rightarrow}{a}^+}\ \text{ID}^{\underset{\rightarrow}{a}^+} \qquad \underset{\rightarrow}{\Omega}_L\ [B^-]\ \underset{\leftarrow}{\Omega}_R \Vdash C^+}{\underset{\rightarrow}{\Omega}_L\ \underset{\rightarrow}{a}^+\ [\underset{\rightarrow}{a}^+ \setminus B^-]\ \underset{\leftarrow}{\Omega}_R \Vdash C^+}\ \backslash\mathrm{L}$$

  $$\begin{tikzcd}{} \arrow[d, squiggly] \\ {}\end{tikzcd}$$

  $$\frac{\underset{\rightarrow}{\Omega}_L\ [B^-]\ \underset{\leftarrow}{\Omega}_R \Vdash C^+}{\underset{\rightarrow}{\Omega}_L\ \underset{\rightarrow}{a}^+\ [\underset{\rightarrow}{a}^+ \setminus B^-]\ \underset{\leftarrow}{\Omega}_R \Vdash C^+}\ \backslash\mathrm{L}'$$

  The other rules for the left-focus judgment remain fundamentally unchanged, save for the fact that the left- and right-hand contexts now contain only atoms of the complementary direction.

Having refined the left-focus judgment to use input message contexts, we may similarly refine the principal reduction rule, $\longrightarrow$I:

$$\frac{\underset{\rightarrow}{\Omega}_L\ [A^-]\ \underset{\leftarrow}{\Omega}_R \Vdash B^+ \quad [B^+] \dashv\!\Vdash \Omega'}{\underset{\rightarrow}{\Omega}_L\ A^-\ \underset{\leftarrow}{\Omega}_R \longrightarrow \Omega'}\ \longrightarrow\mathrm{I}\ .$$

The compatibility rule, $\longrightarrow$C, remains unchanged. **??** summarizes the revised rules for the formula-as-process ordered rewriting framework.

4.1.2 *Comments*

Now we are in a position to understand how the two principal syntactic changes – atom directions and atomic premises for implications – combine to endow the otherwise untyped processes with a modicum of static typing.

In the expression $\downarrow A^- \bullet \underset{\rightarrow}{a}^+$, the message $\underset{\rightarrow}{a}^+$ is an outgoing message, owing to its direction away from the (quoted) process A^-. If the premises of left- and right-handed implications were *not* restricted to atoms of complementary direction, then A^- might possibly be the input process $\uparrow\downarrow B^- / \underset{\rightarrow}{a}^+$, which could incorrectly (re-)capture the outgoing message, $\underset{\rightarrow}{a}^+$, that it just sent:

$$\uparrow\!\big(\downarrow(\uparrow\downarrow B^- / \underset{\rightarrow}{a}^+) \bullet \underset{\rightarrow}{a}^+\big) \longrightarrow (\uparrow\downarrow B^- / \underset{\rightarrow}{a}^+)\ \underset{\rightarrow}{a}^+ \longrightarrow B^-\ .$$

However, because the premises of left- and right-handed implications are indeed restricted to atoms of complementary direction, this scenario is impossible – $\uparrow\downarrow B^- / \underset{\rightarrow}{a}^+$ is not even a well-formed proposition!

Figure 4.2: A formula-as-process ordered rewriting framework

POSITIVE PROPS. $A^+, B^+ ::= \underleftarrow{a}^+ \mid \underrightarrow{a}^+ \mid A^+ \bullet B^+ \mid \mathbf{1} \mid {\downarrow}A^-$

NEGATIVE PROPS. $A^-, B^- ::= \underrightarrow{a}^+ \setminus B^- \mid B^- \,/\, \underleftarrow{a}^+ \mid A^- \mathbin{\&} B^- \mid \top \mid {\uparrow}A^+ \mid \hat{p}^-$

CONTEXTS $\Omega, \Delta ::= \Omega_1\, \Omega_2 \mid \cdot \mid A^- \mid \underleftarrow{a}^+ \mid \underrightarrow{a}^+$

SIGNATURES $\Phi ::= \cdot \mid \Phi, \hat{p}^- \triangleq A^-$

$$\frac{[A^+] \dashV \Omega_1 \quad [B^+] \dashV \Omega_2}{[A^+ \bullet B^+] \dashV \Omega_1\, \Omega_2}\ \bullet\text{R} \qquad \frac{}{[\mathbf{1}] \dashV \cdot}\ \mathbf{1}\text{R}$$

$$\frac{}{[\underleftarrow{a}^+] \dashV \underleftarrow{a}^+}\ \text{ID}^{\underleftarrow{a}^+} \qquad \frac{}{[\underrightarrow{a}^+] \dashV \underrightarrow{a}^+}\ \text{ID}^{\underrightarrow{a}^+} \qquad \frac{}{[{\downarrow}A^-] \dashV A^-}\ {\downarrow}\text{R}$$

$$\frac{\underrightarrow{\Omega}_L\ [B^-]\ \underleftarrow{\Omega}_R \Vdash C^+}{\underrightarrow{\Omega}_L\ \underrightarrow{a}^+\ [\underrightarrow{a}^+ \setminus B^-]\ \underleftarrow{\Omega}_R \Vdash C^+}\ \setminus\text{L}' \qquad \frac{\underrightarrow{\Omega}_L\ [B^-]\ \underleftarrow{\Omega}_R \Vdash C^+}{\underrightarrow{\Omega}_L\ [B^- \,/\, \underleftarrow{a}^+]\ \underleftarrow{\Omega}_R \Vdash C^+}\ /\text{L}'$$

$$\frac{\underrightarrow{\Omega}_L\ [A^-]\ \underleftarrow{\Omega}_R \Vdash C^+}{\underrightarrow{\Omega}_L\ [A^- \mathbin{\&} B^-]\ \underleftarrow{\Omega}_R \Vdash C^+}\ \&\text{L}_1 \qquad \frac{\underrightarrow{\Omega}_L\ [B^-]\ \underleftarrow{\Omega}_R \Vdash C^+}{\underrightarrow{\Omega}_L\ [A^- \mathbin{\&} B^-]\ \underleftarrow{\Omega}_R \Vdash C^+}\ \&\text{L}_2 \qquad \text{(no } \top\text{L rule)}$$

$$\frac{}{[{\uparrow}A^+] \Vdash A^+}\ {\uparrow}\text{L}$$

$$\frac{\underrightarrow{\Omega}_L\ [A^-]\ \underleftarrow{\Omega}_R \Vdash C^+ \quad [C^+] \dashV \Omega'}{\underrightarrow{\Omega}_L\ A^-\ \underleftarrow{\Omega}_R \longrightarrow \Omega'}\ {\longrightarrow}\text{I} \qquad \frac{\Omega \longrightarrow \Omega'}{\Omega_L\, \Omega\, \Omega_R \longrightarrow \Omega_L\, \Omega'\, \Omega_R}\ {\longrightarrow}\text{C}$$

$$\frac{}{\Omega \Longrightarrow \Omega}\ {\Longrightarrow}\text{R} \qquad \frac{\Omega \longrightarrow \Omega' \quad \Omega' \Longrightarrow \Omega''}{\Omega \Longrightarrow \Omega''}\ {\Longrightarrow}\text{T}$$

(Even with the restriction of left- and right-handed implication premises to atoms of complementary direction, it is nevertheless possible for a process to send *itself* a message, as in

$$\uparrow\!\big(\downarrow\!(\uparrow\downarrow B^- \mathbin{/} \underset{\leftarrow}{a}^{+}) \bullet \underset{\leftarrow}{a}^{+}\big) \longrightarrow (\uparrow\downarrow B^- \mathbin{/} \underset{\leftarrow}{a}^{+})\, \underset{\leftarrow}{a}^{+} \longrightarrow B^-\,,$$

but this is not troubling because the intended recipient – the process itself – does indeed receive the message.)

As a related consequence of these syntactic restrictions, there is no contention for messages. Without these restrictions, the above trace could be adapted to one in which a race could arise between two processes contending for the same message:

$$\begin{tikzcd}
 & B^- \\
(\uparrow\downarrow B^- \mathbin{/} \underset{\rightarrow}{a}^{+})\, \underset{\rightarrow}{a}^{+}\, (\underset{\rightarrow}{a}^{+} \mathbin{\backslash} \uparrow\downarrow C^-) \arrow[ur] \arrow[dr] & \\
 & C^-
\end{tikzcd}$$

However, with these restrictions in place, there is no message – neither $\underset{\leftarrow}{a}^{+}$ nor $\underset{\rightarrow}{a}^{+}$ – that can cause contention between $\uparrow B^- \mathbin{/} \underset{\leftarrow}{a}^{+}$ and $\underset{\rightarrow}{a}^{+} \mathbin{\backslash} \uparrow C^-$ because $\underset{\leftarrow}{a}^{+} \neq \underset{\rightarrow}{a}^{+}$.[10] Other forms of nondeterminism still exist in the form of $A^- \mathbin{\&} B^-$.

[10] That is:

- $(\uparrow B^- \mathbin{/} \underset{\leftarrow}{a}^{+})\, \underset{\leftarrow}{a}^{+}\, (\underset{\rightarrow}{a}^{+} \mathbin{\backslash} \uparrow C^-) \longrightarrow \Omega'$ only if $\Omega' = B^-\, (\underset{\rightarrow}{a}^{+} \mathbin{\backslash} \uparrow C^-)$; and
- $(\uparrow B^- \mathbin{/} \underset{\leftarrow}{a}^{+})\, \underset{\rightarrow}{a}^{+}\, (\underset{\rightarrow}{a}^{+} \mathbin{\backslash} \uparrow C^-) \longrightarrow \Omega'$ only if $\Omega' = (\uparrow B^- \mathbin{/} \underset{\leftarrow}{a}^{+})\, C^-$.

One might think that ordered conjunctions ought to be restricted to those of the form $\underset{\leftarrow}{a}^{+} \bullet B^+$ and $B^+ \bullet \underset{\rightarrow}{a}^{+}$, as a kind of dual restriction to those placed on left- and right-handed implications. Although certainly possible, such restrictions would limit the expressiveness of formula-as-process ordered rewriting by precluding a process from sending itself a message – $\downarrow\!(\uparrow\downarrow B^- \mathbin{/} \underset{\leftarrow}{a}^{+}) \bullet \underset{\leftarrow}{a}^{+}$ would not be well-formed, for example.

$\underset{\leftarrow}{A}^{+}$ and $\underset{\rightarrow}{A}^{+}$

4.1.3 *Recursively defined negative propositions*

Recall from chapter 2 that focused ordered rewriting is terminating: for all ordered contexts Ω, every rewriting sequence from Ω is finite (??). Although a seemingly pleasant property, termination significantly limits the expressiveness of focused ordered rewriting. For example, without unbounded rewriting, we cannot even describe producer–consumer systems or finite automata.

As the proof of termination shows, rewriting is bounded precisely because contexts consist of finitely many *finite* propositions. In multiset and ordered rewriting, unbounded behavior is traditionally introduced by way of persistent propositions that may be replicated as much as needed.[11] This is related to **Milner**:??'s use of replication, $!P$, in the π-calculus.[12]

[11] **??Polakow:CMU??Simmons:CMU12**.

[12] **Milner**:??.

However, another option – and the one that we pursue here – is to permit circular negative propositions in the form of mutually recursive definitions, $\hat{p}^- \triangleq A^-$, where the grammar of negative propositions includes these recursively defined propositions:

$$A^-, B^- ::= \underrightarrow{a}^+ \setminus B^- \mid B^- \,/\, \underleftarrow{a}^+ \mid A^- \mathbin{\&} B^- \mid \top \mid \hat{p}^- \,.$$

Sequent calculi with definitions of this kind have previously been studied,[13] but, to the best of our knowledge, the use of recursive definitions in the context of logically motivated rewriting systems is new.

[13] **Hallnas:??Erikkson:??Schroeder-Heister:??McDowell+Miller:**

To rule out definitions like $\hat{p}^- \triangleq \hat{p}^-$ that do not correspond to sensible infinite propositions and also to rule out sensible but inessential definitions like $\hat{p}^- \triangleq \hat{q}^-$, we require that all definitions be *contractive*[14] – *i.e.*, that the body of each recursive definition begin with a logical connective, logical constant, or atomic proposition at the top level.

[14] ??.

Recursive definitions are collected into a signature, Φ, that indexes the rewriting relations: $\longrightarrow_\Phi$ and $\Longrightarrow_\Phi$.[15] Syntactically, these signatures are given by

$$\Phi ::= \cdot \mid \Phi, (\hat{p}^- \triangleq A^-) \,.$$

[15] We often elide the index, as it is usually clear from context.

By analogy with recursive types from functional programming,[16] we must now decide whether to treat definitions *iso*recursively or *equi*recursively. Under an equirecursive treatment, definitions may be silently unrolled or rolled at will; in other words, $\hat{p}^-$ is literally *equal* to its unrolling: $\hat{p}^- = A^-$. In contrast, under an isorecursive treatment, unrolling a recursively defined proposition would count only as an explicit step of rewriting: $\hat{p}^- \neq A^-$ but $\hat{p}^- \longrightarrow A^-$.[17]

[16] ??.

[17] Is this right for isorecursive in the presence of focusing?

We choose an equirecursive treatment of definitions because the accompanying generous notion of equality helps to minimize the overhead of recursively defined propositions. As a simple example, under the equirecursive definition $\hat{p}^- \triangleq \underrightarrow{a}^+ \setminus {\uparrow}{\downarrow}\hat{p}^-$, we have the trace

$$\underrightarrow{a}^+\, \underrightarrow{a}^+\, \hat{p}^- = \underrightarrow{a}^+\, \underrightarrow{a}^+\, (\underrightarrow{a}^+ \setminus {\uparrow}{\downarrow}\hat{p}^-) \longrightarrow \underrightarrow{a}^+\, \hat{p}^-$$

However, because the left-focus judgment is defined inductively, not coinductively, there are some recursively defined negative propositions that cannot successfully be put into focus. Under the definition $\hat{p}^- \triangleq \underrightarrow{a}^+ \setminus \hat{p}^-$, for example, there are no contexts $\underrightarrow{\Omega}_L$ and $\underleftarrow{\Omega}_R$ and positive consequent C^+ for which $\underrightarrow{\Omega}_L\ [\hat{p}^-]\ \underleftarrow{\Omega}_R \Vdash C^+$ is derivable. To derive a left-focus judgment on $\hat{p}^-$, the finite context $\underrightarrow{\Omega}_L$ would need to hold an infinite stream of $\underrightarrow{a}^+$ atoms, which is impossible in an inductively defined, and hence finite, context.

However, by inserting ${\uparrow}{\downarrow}$ as a double shift to blur focus – in a way similar to how double shifts were used in the embedding of unfocused rewriting (??) – the definition can be revised to one that admits a left-focus judgment. Specifically, if $\hat{p}^- \triangleq \underrightarrow{a}^+ \setminus {\uparrow}{\downarrow}\hat{p}^-$, then $\underrightarrow{a}^+\ [\hat{p}^-] \Vdash {\downarrow}\hat{p}^-$ is derivable, and so $\underrightarrow{a}^+\, \hat{p}^- \longrightarrow \hat{p}^-$.

4.1.4

In unfocused and focused ordered rewriting of the OR (??) and FOR (??) frameworks, the rewriting relation, $\longrightarrow$, described a purely internal operation: $\Omega \longrightarrow$

Ω' held independently of any environment that might surround Ω. Ordered rewriting's isolationism was affirmed by its compatibility rule, $\longrightarrow$C, which shows that the environment remains unaffected by the rewriting of its [...].

$$\frac{\Omega \longrightarrow \Omega'}{\Omega_L\ \Omega\ \Omega_R \longrightarrow \Omega_L\ \Omega'\ \Omega_R}\ {\longrightarrow}\mathrm{C}$$

Now that we have a formula-as-process interpretation to ordered rewriting, we should reconsider this strict isolationism. Under our formula-as-process reading, this isolationist rewriting judgment corresponds to a reduction semantics for processes. But now, messages, as represented by the directed $\underleftarrow{a}^+$ and $\underrightarrow{a}^+$ atoms, make it possible to describe the interactions that a configuration Ω offers to its surroundings.

4.1.5 *Input transitions*

For the formula-as-process interpretation, we have thus far examined the rewriting judgement, $\Omega \longrightarrow \Omega'$, and suggested that it represents a kind of reduction semantics for the underlying processes.

A reduction semantics is not the only way to describe the operational semantics. In process calculi like the π-calculus, labeled transition systems are frequently used as an alternative to a reduction semantics, particularly when [...]. Now that we are ascribing a formula-as-process interpretation to ordered rewriting, we can similarly conceive of a labeled transition system for ordered contexts.

Interactions [...] Rather than adopting an explicit judgement for output interactions, we make use of context equality. The context Ω outputs messages $\underleftarrow{\Omega}_L$ to the left and messages $\underrightarrow{\Omega}_R$ to the right exactly when $\Omega = \underleftarrow{\Omega}_L\,\Omega'\,\underrightarrow{\Omega}_R$ for some context Ω'. We will sometimes refer to the context Ω' here as the *continuation context* because it represents[18] the state after the output of $\underleftarrow{\Omega}_L$ and $\underrightarrow{\Omega}_R$ occurs.

[18] wc?

Input interactions do require an explicit judgment, $\underrightarrow{\Omega}_L\,[\Omega]\,\underleftarrow{\Omega}_R \longrightarrow \Omega'$, indicating that upon receiving messages $\underrightarrow{\Omega}_L$ from the left and $\underleftarrow{\Omega}_R$ from the right, the context Ω evolves to Ω' in a single step.

We think of Ω as the sole innput to the input transition judgement, and $\underrightarrow{\Omega}_L$, $\underleftarrow{\Omega}_R$, and Ω' as outputs of the judgement. The input transition judgement asks "What input messages suffice for Ω to make a transition?"

At its heart, each transition derives from focusing on a single negative proposition, A^-, as captured by the [....] rule:

$$\frac{\underrightarrow{\Omega}_L\,[A^-]\,\underleftarrow{\Omega}_R \Vdash C^+ \qquad [C^+] \dashv\Vdash \Omega'}{\underrightarrow{\Omega}_L\,[A^-]\,\underleftarrow{\Omega}_R \longrightarrow \Omega'}\,?\,.$$

Aside from the change of judgment in the rule's conclusion, this [...] rule is identical to the core [...] rule for reduction. How can we claim that input transition is distinct from reduction?

The difference is one of input/output modes. In a reduction $\underrightarrow{\Omega}_L\,A^-\,\underleftarrow{\Omega}_R \longrightarrow \Omega'$, the entire $\underrightarrow{\Omega}_L\,A^-\,\underleftarrow{\Omega}_R$ context is treated as an input to the reduction judgment. In the input transition $\underrightarrow{\Omega}_L\,[A^-]\,\underleftarrow{\Omega}_R \longrightarrow \Omega'$, on the other hand, the proposition A^- is treated as an input to the input transition judgment, but the contexts $\underrightarrow{\Omega}_L$, $\underleftarrow{\Omega}_R$, and Ω' are all treated as outputs.

In addition to this core input transition rule,

$$\frac{\underrightarrow{\Omega}_L\,[A^-]\,\underleftarrow{\Omega}_R \Vdash C^+ \qquad [C^+] \dashv\Vdash \Omega'}{\underrightarrow{\Omega}_L\,[A^-]\,\underleftarrow{\Omega}_R \longrightarrow \Omega'}\,?$$

Also discuss here?

$$\frac{\underrightarrow{\Omega}_L\ [A^-]\ \underleftarrow{\Omega}_R \Vdash C^+ \quad [C^+] \dashV \Omega'}{\underrightarrow{\Omega}_L\ [A^-]\ \underleftarrow{\Omega}_R \longrightarrow \Omega'}$$

$$\frac{\underrightarrow{\Omega}_L\ \underrightarrow{a}\ [\Omega]\ \underleftarrow{\Omega}_R \longrightarrow \Omega'}{\underrightarrow{\Omega}_L\ [\underrightarrow{a}\ \Omega]\ \underleftarrow{\Omega}_R \longrightarrow \Omega'} \qquad \frac{\underrightarrow{\Omega}_L\ [\Omega]\ \underleftarrow{a}\ \underleftarrow{\Omega}_R \longrightarrow \Omega'}{\underrightarrow{\Omega}_L\ [\Omega\ \underleftarrow{a}]\ \underleftarrow{\Omega}_R \longrightarrow \Omega'}$$

$$\frac{[\Omega]\ \underleftarrow{\Omega}_R \longrightarrow \Omega'}{[\underleftarrow{a}\ \Omega]\ \underleftarrow{\Omega}_R \longrightarrow \underleftarrow{a}\ \Omega'} \qquad \frac{[\Omega]\ \underleftarrow{\Omega}_R \longrightarrow \Omega'}{[\underrightarrow{a}\ \Omega]\ \underleftarrow{\Omega}_R \longrightarrow \underrightarrow{a}\ \Omega'} \qquad \frac{[\Omega]\ \underleftarrow{\Omega}_R \longrightarrow \Omega'}{[A^-\ \Omega]\ \underleftarrow{\Omega}_R \longrightarrow A^-\ \Omega'}$$

$$\frac{\underrightarrow{\Omega}_L\ [\Omega] \longrightarrow \Omega'}{\underrightarrow{\Omega}_L\ [\Omega\ \underleftarrow{a}] \longrightarrow \Omega'\ \underleftarrow{a}} \qquad \frac{\underrightarrow{\Omega}_L\ [\Omega] \longrightarrow \Omega'}{\underrightarrow{\Omega}_L\ [\Omega\ \underrightarrow{a}] \longrightarrow \Omega'\ \underrightarrow{a}} \qquad \frac{\underrightarrow{\Omega}_L\ [\Omega] \longrightarrow \Omega'}{\underrightarrow{\Omega}_L\ [\Omega\ A^-] \longrightarrow \Omega'\ A^-}$$

THEOREM 4.1. *If $\underrightarrow{\Omega}_L\ [\Omega]\ \underleftarrow{\Omega}_R \longrightarrow \Omega'$, then $\underrightarrow{\Omega}_L\ \Omega\ \underleftarrow{\Omega}_R \longrightarrow \Omega'$. Conversely, if $\Omega \longrightarrow \Omega'$, then $[\Omega] \longrightarrow \Omega'$.*

Proof. The two claims are proved by induction on the structure of the given input transition and reduction, respectively, after first proving an easy lemma:

- If $\underrightarrow{\Omega}_L\ [\Omega]\ \underleftarrow{\Omega}_R \longrightarrow \Omega'$, then $[\underrightarrow{\Omega}_L\ \Omega\ \underleftarrow{\Omega}_R] \longrightarrow \Omega'$. □

$$\frac{\Omega_L = \Omega'_L\ \underrightarrow{\Omega}_L \quad \underrightarrow{\Omega}_L\ [\Omega]\ \underleftarrow{\Omega}_R \longrightarrow \Omega' \quad \underleftarrow{\Omega}_R\ \Omega'_R = \Omega_R}{\Omega_L\ \Omega\ \Omega_R \longrightarrow \Omega'_L\ \Omega'\ \Omega'_R}$$

4.2

In chapter 1, we saw that string rewriting can be used to specify the dynamics of concurrent systems, but that those specifications are quite abstract. Even the [...] is left completely abstract: permitted rewritings just *happen*, as if a central, meta-level actor schedules and otherwise coordinates rewriting.

In the previous chapter, we presented a different rewriting framework, this one derived from the ordered sequent calculus and closely related to the **Lambek:AMM58** calculus.[19] Ordered rewriting, too, leaves the [...] completely abstract

[19] **Lambek:AMM58**.

At this high level of abstraction, string rewriting specifications are not amenable to

In this and the previous **??**, we have also seen that the formula-as-process ordered rewriting framework permits only those rewritings that have a sensible interpretation under local, message-passing communication. Thus far, we have seen that the formula-as-process ordered rewriting framework precludes rewritings, such as $\underleftarrow{a}$, that are not sensible in [...].

Given a string rewriting alphabet Σ, a mapping $\theta\colon \Sigma^* \to ?$ is a *role assignment for* Σ if it is a monoid homomorphism between finite strings and ordered contexts that uniquely casts each symbol $a \in \Sigma$ in the role of either: an atom, $\underleftarrow{a}$ or $\underrightarrow{a}$; or of a recursively defined proposition, $\hat{a}$.

A pair (θ, Φ) is a *choreography* of the string rewriting specification (Σ, Θ) if:

- $\theta \colon \Sigma^* \longrightarrow ?$ is a role assignment for Σ;
- Φ is a formula-as-process signature that provides definitions for each of the recursively defined propositions that appear in the image of θ; and
- θ is a (strong) bisimulation between $\longrightarrow_\Theta$ and $\longrightarrow_\Phi$, the string rewriting and (formula-as-process) rewriting relations:

$$\begin{tikzcd}
w \arrow[r, "\Theta"'] \arrow[d, no head, "\theta"'] & w' \arrow[d, no head, dashed, "\theta"] & \text{and} & w \arrow[r, dashed, "\Theta"'] \arrow[d, no head, "\theta"'] & w' \arrow[d, no head, dashed, "\theta"] \\
\Omega \arrow[r, dashed, "\Phi"'] & \Omega' & & \Omega \arrow[r, "\Phi"'] & \Omega' .
\end{tikzcd}$$

Using the formula-as-process ordered rewriting as a substrate, we would like to choreograph string rewriting specifications (Σ, Θ) by mapping them to formula-as-process ordered rewriting. Specifically, we would like to find a binary relation $\mathcal{R}$ between strings and ordered contexts and an ordered rewriting signature Φ such that $\mathcal{R}$ is a (strong) bisimulation between $\longrightarrow_\Theta$ and $\longrightarrow_\Phi$, the string rewriting and (formula-as-process) ordered rewriting relations.

$$\begin{tikzcd}
w \arrow[r, "\Theta"'] \arrow[d, no head, "\mathcal{R}"'] & w' \arrow[d, no head, dashed, "\mathcal{R}"] & \text{and} & w \arrow[r, dashed, "\Theta"'] \arrow[d, no head, "\mathcal{R}"'] & w' \arrow[d, no head, dashed, "\mathcal{R}"] \\
\Omega \arrow[r, dashed, "\Phi"'] & \Omega' & & \Omega \arrow[r, "\Phi"'] & \Omega' .
\end{tikzcd}$$

[20]

[20] A relation $\mathcal{R}$ such that: $\Omega \ \mathcal{R}^{-1} \ w \longrightarrow_\Theta w'$ implies $\Omega \longrightarrow_\Phi \Omega' \ \mathcal{R}^{-1} \ w'$ for some Ω'; and $w \ \mathcal{R} \ \Omega \longrightarrow_\Phi \Omega'$ implies $w \longrightarrow_\Theta w' \ \mathcal{R} \ \Omega'$ for some w'.

Because the formula-as-process ordered rewriting framework precludes rewritings that [...], this choreographing operationalizes string rewriting.

Given a string rewriting specification Θ, we would like to find an ordered rewriting signature Φ that mimics

In other words, More specifically, the relation $\mathcal{R}$ will be a monoid homomorphism so that

Not all string rewriting specifications have a valid choreography. For instance, the string rewriting specification (Σ, Θ) where

$$\begin{aligned}
\Sigma &= \{a, b\} \\
\Theta &= (a\,b \longrightarrow b)\,, (b \longrightarrow \epsilon)\,, (a \longrightarrow \epsilon)
\end{aligned}$$

has no valid choreography. Suppose that θ were a role assignment that led to a valid choreography, (θ, Φ). For the constraints $\theta(b) \longrightarrow_\Phi (\cdot)$ and $\theta(a) \longrightarrow_\Phi (\cdot)$ to be satisfiable, θ would have to map $b \mapsto \hat{b}$ and $a \mapsto \hat{a}$. However, the first axiom would then induce the constraint $\hat{a}\,\hat{b} \longrightarrow_\Phi \hat{b}$, which is not satisfiable – there are no definitions for [21]

[21] Except, there are: $\hat{a} \triangleq \uparrow 1$.

Our goal is not to synthesize a choreography from scratch for a given string rewriting specification, (Σ, Θ). Instead, our goal is to synthesize a (formula-as-process) ordered rewriting signature from *a role assignment* θ for a given string rewriting specification.

Given a string rewriting specification (Σ, Θ) and a role assignment $\theta \colon \Sigma^* \longrightarrow \underleftarrow{\Sigma} \cup \underrightarrow{\Sigma} \cup \hat{\Sigma}$[22], we would like to determine whether θ gives rise to a meaningful

[22] fix

choreography of (Σ, Θ). That is, we would to construct, if possible, an ordered rewriting signature Φ that makes θ a (strong) bisimulation between the string rewriting and formula-as-process ordered rewriting relations, $\longrightarrow_\Theta$ and $\longrightarrow_\Phi$, respectively.

$$\begin{tikzcd}
w \arrow[r] \arrow[d, "\theta"'] & {}_\Theta\, w' \arrow[d, dashed, "\theta"] \\
\Omega \arrow[r, dashed] & {}_\Phi\, \Omega'
\end{tikzcd}
\quad \text{and} \quad
\begin{tikzcd}
w \arrow[r, dashed] \arrow[d, "\theta"'] & {}_\Theta\, w' \arrow[d, dashed, "\theta"] \\
\Omega \arrow[r] & {}_\Phi\, \Omega'
\end{tikzcd} .$$

We will first explain by example how such a signature Φ is constructed, reserving a formal description of the choreographing procedure to ??.

Recall from chapter 1 the string rewriting specification of a system that can rewrite strings over $\Sigma = \{a, b\}$ into the empty string if the initial string ends in b; that specification consists of the axioms

$$\Theta = (a\,b \longrightarrow b), (b \longrightarrow \epsilon) .$$

The monoid homomorphism θ such that $\theta(a) = \underset{\rightarrow}{a}$ and $\theta(b) = \hat{b}$ is a role assignment for this specification.

We can apply the role assignment θ to the axioms Θ to see which ordered rewritings must hold of the relation $\longrightarrow_\Phi$ if (θ, Φ) is to be a meaningful choreography of the specification (Σ, Θ). In this example, the axioms Θ together with θ induce the rewritings

$$\theta(a\,b) = \underset{\rightarrow}{a}\,\hat{b} \longrightarrow_\Phi \hat{b} = \theta(b) \quad \text{and} \quad \theta(b) = \hat{b} \longrightarrow_\Phi (\cdot) = \theta(\epsilon) .$$

$$\begin{tikzcd}
a\,b \arrow[r] \arrow[d, "\theta"'] & {}_\Theta\, b \arrow[d, "\theta"] \\
\underset{\rightarrow}{a}\,\hat{b} \arrow[r] & {}_\Phi\, \hat{b}
\end{tikzcd}
\quad \text{and} \quad
\begin{tikzcd}
b \arrow[r] \arrow[d, "\theta"'] & {}_\Theta\, \epsilon \arrow[d, "\theta"] \\
\hat{b} \arrow[r] & {}_\Phi\, (\cdot)
\end{tikzcd}$$

So, to find a meaningful choreography (θ, Φ) for the string rewriting specification (Σ, Θ), it suffices to find a signature Φ for which the rewritings $\underset{\rightarrow}{a}\,\hat{b} \longrightarrow_\Phi \hat{b}$ and $\hat{b} \longrightarrow_\Phi (\cdot)$ – and only those rewritings – are derivable. In other words, $\underset{\rightarrow}{a}\,\hat{b} \longrightarrow_\Phi \hat{b}$ and $\hat{b} \longrightarrow_\Phi (\cdot)$ serve as constraints on Φ that we must solve.

To solve these constraints, we must find a definition for $\hat{b}$ that makes those – and only those – rewritings derivable. In this instance, such a solution is the definition $\hat{b} \triangleq (\underset{\rightarrow}{a} \setminus \uparrow\downarrow\hat{b}) \mathbin{\&} \uparrow\mathbf{1}$ and the corresponding signature, $\Phi \triangleq \big(\hat{b} \triangleq (\underset{\rightarrow}{a} \setminus \uparrow\downarrow\hat{b}) \mathbin{\&} \uparrow\mathbf{1}\big)$. Here is how we arrive at that solution:

- Let's temporarily restrict our attention to the constraint $\underset{\rightarrow}{a}\,\hat{b} \longrightarrow_\Phi \hat{b}$. Notice that $\underset{\rightarrow}{a}\;(\underset{\rightarrow}{a} \setminus \uparrow\downarrow\hat{b}) \longrightarrow \hat{b}$. By the universal property of left-handed implication, there must exist an open derivation of $\underset{\rightarrow}{a}\;[\hat{b}] \Vdash C_1^+$ from [...].

- Turning our attention to the constraint $\hat{b} \longrightarrow_\Phi (\cdot)$, notice that $\uparrow\mathbf{1} \longrightarrow (\cdot)$. By the universal properties of $\uparrow\mathbf{1}$, there must exist an open derivation of $\underset{\rightarrow}{\Omega}_L\;[\hat{b}]\;\underset{\leftarrow}{\Omega}_R \Vdash C^+$ from $\underset{\rightarrow}{\Omega}_L\;[\uparrow\mathbf{1}]\;\underset{\leftarrow}{\Omega}_R \Vdash C^+$.

The least proposition $\hat{b}$ that has both of these open derivations is $\hat{b} \triangleq (\underrightarrow{a} \setminus {\uparrow}{\downarrow}\hat{b}) \mathbin{\&} {\uparrow}\mathbf{1}$.

More generally, suppose that we have a constraint $\underrightarrow{\Omega}_L\, \hat{a}\, \underleftarrow{\Omega}_R \longrightarrow_{\Phi} \Omega'$. Then notice that (morally) $\underrightarrow{\Omega}_L\, (\underrightarrow{\Omega}_L \setminus ({\uparrow}{\bullet}\Omega') \mathbin{/} \underleftarrow{\Omega}_R)\, \underleftarrow{\Omega}_R \longrightarrow \Omega'$. By the universal propeties of $\underrightarrow{\Omega}_L \setminus ({\uparrow}{\bullet}\Omega') \mathbin{/} \underleftarrow{\Omega}_R$, there must exist an open derivation of $\underrightarrow{\Omega}_L\, [\hat{a}]\, \underleftarrow{\Omega}_R \Vdash C^+$ from $\underrightarrow{\Omega}_L\, [\underrightarrow{\Omega}_L \setminus ({\uparrow}{\bullet}\Omega') \mathbin{/} \underleftarrow{\Omega}_R]\, \underleftarrow{\Omega}_R \Vdash C^+$.

Returning to our running example, we need to find a definition for $\hat{b}$ such that both $\underrightarrow{a}\, \hat{b} \longrightarrow \hat{b}$ and $\hat{b} \longrightarrow (\cdot)$ will be derivable. By inversion, these induced rewritings will be derivable exactly when both

$$\underrightarrow{a}\, [\hat{b}] \Vdash C_1^+ \text{ for some } C_1^+ \text{ such that } [C_1^+] \dashv\!\!\vdash \hat{b}$$

and

$$[\hat{b}] \Vdash C_2^+ \text{ for some } C_2^+ \text{ such that } [C_2^+] \dashv\!\!\vdash (\cdot)\,.$$

It is easy to check *(i)* that the first condition would be satisfied if $\hat{b}$ were $\underrightarrow{a} \setminus {\uparrow}{\downarrow}\hat{b}$; and *(ii)* that the second condition would be satisfied if $\hat{b}$ were ${\uparrow}\mathbf{1}$. If $\hat{b}$ were somehow simultaneously both $\underrightarrow{a} \setminus {\uparrow}{\downarrow}\hat{b}$ and ${\uparrow}\mathbf{1}$, then both conditions would be satisfied. Fortunately, additive conjunction allows us to do just that: when $\hat{b} \triangleq (\underrightarrow{a} \setminus {\uparrow}{\downarrow}\hat{b}) \mathbin{\&} {\uparrow}\mathbf{1}$, the induced rewritings, $\underrightarrow{a}\, \hat{b} \longrightarrow \hat{b}$ and $\hat{b} \longrightarrow (\cdot)$ – and only those rewritings – are derivable. $\Omega_L\, \hat{b}\, \Omega_R \longrightarrow \Omega'$ only if either

- $\Omega_L \longrightarrow \Omega'_L$ and $\Omega' = \Omega'_L\, \hat{b}\, \Omega_R$ for some Ω'_L;
- $\Omega' = \Omega_L\, \Omega_R$;
- $\Omega_L = \Omega'_L\, \underrightarrow{a}$ and $\Omega' = \Omega'_L\, \hat{b}\, \Omega_R$; or
- $\Omega_R \longrightarrow \Omega'_R$ and $\Omega' = \Omega_L\, \hat{b}\, \Omega'_R$ for some Ω'_R.

To choreograph a string rewriting specification, we would like to assign one, and only one, role to each symbol $a \in \Sigma$: in the choreography, each symbol a becomes either a message, $\underleftarrow{a}$ or $\underrightarrow{a}$, or a recursively defined process, $\hat{a}$. A monoid homomorphism[23] from strings to ordered contexts that satisfies this condition is called a *[...] assignment*.

[23] isomorphism?

When applied to the specification's axioms, the [...] assignment θ induces the rewriting steps

$$\underrightarrow{a}\, \hat{b} \longrightarrow \hat{b} \quad \text{and} \quad \hat{b} \longrightarrow (\cdot)\,,$$

which we denote by $\theta(\Theta)$.

For the [...] assignment θ to yield an actual choreography of the axioms Θ, we must be able to solve these induced rewritings for $\hat{b}$, determining a definition for $\hat{b}$ that makes these – and only these – rewriting steps derivable.

More generally, a [...] assignment θ yields a well-specified choreography for a specification with axioms Θ if the induced ordered rewriting steps $\theta(\Theta)$ are solvable with definitions for all recursively defined processes that make the induced rewritings $\theta(\Theta)$ – and only those rewritings – derivable. In other words, θ

Returning to our running example, we need to find a definition for $\hat{b}$ such that both $\underset{\rightarrow}{a}\,\hat{b} \longrightarrow \hat{b}$ and $\hat{b} \longrightarrow (\cdot)$ will be derivable. By inversion, these induced rewritings will be derivable exactly when both

$$\underset{\rightarrow}{a}\,[\hat{b}] \Vdash C_1^+ \text{ for some } C_1^+ \text{ such that } [C_1^+] \dashv\!\vert\, \hat{b}$$

and

$$[\hat{b}] \Vdash C_2^+ \text{ for some } C_2^+ \text{ such that } [C_2^+] \dashv\!\vert\, (\cdot)\,.$$

It is easy to check *(i)* that the first condition would be satisfied if $\hat{b}$ were $\underset{\rightarrow}{a} \backslash {\uparrow}{\downarrow}\hat{b}$; and *(ii)* that the second condition would be satisfied if $\hat{b}$ were ${\uparrow}\mathbf{1}$. If $\hat{b}$ were somehow simultaneously both $\underset{\rightarrow}{a} \backslash {\uparrow}{\downarrow}\hat{b}$ and ${\uparrow}\mathbf{1}$, then both conditions would be satisfied. Fortunately, additive conjunction allows us to do just that: when $\hat{b} \triangleq (\underset{\rightarrow}{a} \backslash {\uparrow}{\downarrow}\hat{b}) \mathbin{\&} {\uparrow}\mathbf{1}$, the induced rewritings, $\underset{\rightarrow}{a}\,\hat{b} \longrightarrow \hat{b}$ and $\hat{b} \longrightarrow (\cdot)$ – and only those rewritings – are derivable. $\Omega_L\,\hat{b}\,\Omega_R \longrightarrow \Omega'$ only if either

- $\Omega_L \longrightarrow \Omega_L'$ and $\Omega' = \Omega_L'\,\hat{b}\,\Omega_R$ for some Ω_L';
- $\Omega' = \Omega_L\,\Omega_R$;
- $\Omega_L = \Omega_L'\,\underset{\rightarrow}{a}$ and $\Omega' = \Omega_L'\,\hat{b}\,\Omega_R$; or
- $\Omega_R \longrightarrow \Omega_R'$ and $\Omega' = \Omega_L\,\hat{b}\,\Omega_R'$ for some Ω_R'.

Not all [...] assignments yield well-specified choreographies. This happens when there is no solution for the recursively defined propositionsthat makes all of the induced rewritings derivable.

- *Each induced rewriting must have at least one process in its premise*[24]. For example, the [...] assignments θ' such that either $\theta'(b) = \underset{\leftarrow}{b}$ or $\theta'(b) = \underset{\rightarrow}{b}$ holds do *not* yield well-specified choreographies. From the string rewriting axiom $b \longrightarrow \epsilon$, the [...] assignment θ' induces either $\underset{\leftarrow}{b} \longrightarrow (\cdot)$ or $\underset{\rightarrow}{b} \longrightarrow (\cdot)$, and there is no solution that makes either of these induced ordered rewritings derivable.

[24] wc

- *Each induced rewriting must have at most one process in its premise*[25]. For example, the [...] assignment θ' such that $\theta'(a) = \hat{a}$ and $\theta'(b) = \hat{b}$ hold does not yield a well-specified choreography because there is no solution that makes $\hat{a}\,\hat{b} \longrightarrow \hat{b}$ derivable.

[25] wc

- *Each message in the premises of induced rewritings must be flowing toward that premise's process*[26]. For example, the [...] assignment θ' such that $\theta'(a) = \underset{\leftarrow}{a}$ and $\theta'(b) = \hat{b}$ hold does not yield a well-specified choreography because there is no solution that makes $\underset{\leftarrow}{a}\,\hat{b} \longrightarrow \hat{b}$ derivable. In formula-as-processfocused ordered rewriting (PFOR) there is no process $\hat{b}$ that can receive a message, like $\underset{\leftarrow}{a}$, that is flowing away.

[26] wc

For the choreography to be well-specified, this [...] assignment must induce from the string rewriting specification's axioms a collection of locally achievable ordered rewriting steps[27]. If the ordered rewriting steps induced by the [...] assignment cannot be achieved by local communication, then the choreography is not well-specified.

[27] reductions

For example, recall from chapter 1 the string rewriting specification of a system that can rewrite strings over $\Sigma = \{a, b\}$ into the empty string if the initial string ends in b; that specification used axioms

$$\Theta = (a\,b \longrightarrow b), (b \longrightarrow \epsilon)\,.$$

So, to choreograph this specification, we must choose an assignment of roles – either message or process – to symbols a and b – let's choose $a \mapsto \underrightarrow{a}$ and $b \mapsto \hat{b}$. From the axioms Θ, this assignment induces the rewritings

$$\underrightarrow{a}\,\hat{b} \longrightarrow \hat{b} \quad \text{and} \quad \hat{b} \longrightarrow (\cdot)\,.$$

Are these reductions achievable by purely local communication? Because our formula-as-process interpretation of ordered rewriting ensures that all communication is local, we need only verify that there is a solution for $\hat{b}$ [...].

Any solution for $\hat{b}$ must be consistent with $\underrightarrow{a} \setminus {\uparrow}{\downarrow}\hat{b}$ so that $\underrightarrow{a}\,\hat{b} \longrightarrow \hat{b}$ is derivable. Furthermore, any solution for $\hat{b}$ must be consistent with ${\uparrow}\mathbf{1}$ so that $\hat{b} \longrightarrow \cdot$ is derivable. The least such solution is

$$\hat{b} \triangleq (\underrightarrow{a} \setminus {\uparrow}{\downarrow}\hat{b}) \mathbin{\&} {\uparrow}\mathbf{1}\,,$$

It indeed validates the required reductions,

$$\begin{aligned} \underrightarrow{a}\,\hat{b} &= \underrightarrow{a}\,\big((\underrightarrow{a} \setminus {\uparrow}{\downarrow}\hat{b}) \mathbin{\&} {\uparrow}\mathbf{1}\big) \longrightarrow \hat{b} \\ \hat{b} &= \underrightarrow{a}\,\big((\underrightarrow{a} \setminus {\uparrow}{\downarrow}\hat{b}) \mathbin{\&} {\uparrow}\mathbf{1}\big) \longrightarrow (\cdot)\,, \end{aligned}$$

and only the required reductions:

If $\Omega_L\,\hat{b}\,\Omega_R \longrightarrow \Omega'$, then either:

- $\Omega_L = \Omega'_L\,\underrightarrow{a}$ and $\Omega' = \Omega'_L\,\hat{b}\,\Omega_R$, for some Ω'_L;
- $\Omega' = \Omega_L\,\Omega_R$;
- $\Omega_L \longrightarrow \Omega'_L$ and $\Omega' = \Omega'_L\,\hat{b}\,\Omega_R$, for some Ω'_L; or
- $\Omega_R \longrightarrow \Omega'_R$ and $\Omega' = \Omega_L\,\hat{b}\,\Omega'_R$, for some Ω'_R.

$$\begin{aligned} \underleftarrow{a}\,\hat{b} \longrightarrow \hat{b} &\quad \text{and} \quad \hat{b} \longrightarrow (\cdot) \\ \hat{a}\,\hat{b} \longrightarrow \hat{b} &\quad \text{and} \quad \hat{b} \longrightarrow (\cdot) \\ \hat{a}\,\underleftarrow{b} \longrightarrow \underleftarrow{b} &\quad \text{and} \quad \underleftarrow{b} \longrightarrow (\cdot) \end{aligned}$$

To be well-specified,

An *[...] assignment* θ is a monoid homomorphism from strings to ordered contexts that injectively maps each symbol $a \in \Sigma$ to either a message, $\underleftarrow{a}$ or $\underrightarrow{a}$, or a recursively defined process, $\hat{a}$.[28]

[28] Injectivity keeps θ from identifying distinct symbols.

Given an [...] assignment θ, a string rewriting specification's axioms induce rewriting steps that must hold if the specification is to have a choreography. For each axiom $w \longrightarrow w' \in \Theta$, the [...] assignment θ induces a requirement that a faithful choreography must satisfy the rewriting step $\theta(w) \longrightarrow \theta(w')$.

4.3 *Constructing a choreography from a specification*

For an example of this procedure, let's construct a choreography for the string rewriting specification of the system from chapter 1 that can rewrite strings over $\Sigma = \{a, b\}$ into the empty string. Recall that that specification consisted of the axioms

$$\Theta = (a\,b \longrightarrow b), (b \longrightarrow \epsilon)\,.$$

The first step in constructing a choreography is to choose a *[...] assignment* that maps each symbol to either an atom or recursively defined proposition, [which represent a message or recursively defined process,respectively.] For example, $\theta = \{a \mapsto \underset{\rightarrow}{a}, b \mapsto \hat{b}\}$ is an [...] assignment that maps a to a right-directed message and b to a process. Like θ, all [...] assignments must be injective, to keep distinct symbols from becoming identified in the choreography.

Next, we apply the [...] assignment to each of the string rewriting specification's axioms and simultaneously replace the empty string with the empty ordered context.[29] This results in a collection of ordered rewriting steps that the choreography must satisfy if it is to be a faithful reflection of the string rewriting specification. Applying θ to the axioms of **??** yields

[29] Strictly speaking, the monoid operations are also exchanged, but because both are indicated by juxtaposition, this happens silently.

$$\underset{\rightarrow}{a}\,\hat{b} \longrightarrow \hat{b} \quad \text{and} \quad \hat{b} \longrightarrow \cdot$$

as rewritings required of the choreography.

Finally, we solve for the recursively defined propositions that appear in the required rewritings. In this example, $\hat{b}$ must be consistent with $\underset{\rightarrow}{a} \backslash {\uparrow}{\downarrow}\hat{b}$ if $\underset{\rightarrow}{a}\,\hat{b} \longrightarrow \hat{b}$ is to be derivable; $\hat{b}$ must also be consistent with ${\uparrow}\mathbf{1}$ if $\hat{b} \longrightarrow \cdot$ is to be derivable. The least such solution is $\hat{b} \triangleq (\underset{\rightarrow}{a} \backslash {\uparrow}{\downarrow}\hat{b}) \mathbin{\&} {\uparrow}\mathbf{1}$. Indeed, under this definition,

$$\underset{\rightarrow}{a}\,\hat{b} \longrightarrow \hat{b} \quad \text{and} \quad \hat{b} \longrightarrow \cdot$$

are both derivable.

4.3.1

Not all [...] assignments yield choreographies. For instance, suppose we had chosen $\theta' = \{a \mapsto \hat{a}, b \mapsto \underrightarrow{b}\}$ or any other assignment θ' that maps b to an atom. Applying θ' to the second axiom would yield either $\underrightarrow{b} \longrightarrow \cdot$ or $\underleftarrow{b} \longrightarrow \cdot$ as required rewriting steps. Neither of these make for a valid choreography both of which require a message to be recognized and acted upon by the ether.

4.3.2

To construct a choreography, we need to find a *choreographing assignment* that consistently localizes each axiom.

An assignment that maps both a and b to messages, such as $\theta = \{a \mapsto \underrightarrow{a}, b \mapsto \underrightarrow{b}\}$ which results in $\underrightarrow{a}\,\underrightarrow{b} \longrightarrow \underrightarrow{b}$ and $\underrightarrow{b} \longrightarrow \cdot$,

As a string rewriting specification, the axioms are interpreted from a global perspective. For instance, the first axiom states that when the symbols $a\,b$ occur in that order, they may be rewritten to b. But the axiom does not describe how that rewriting occurs.

With choreographies, we would like to work at a (slightly) lower level of abstraction to describe

Suppose that we are given a string rewriting specification that consists of axioms ? over the rewriting alphabet Σ. A *choreographing assignment* is an injection in which each symbol $a \in \Sigma$ is mapped to an ordered proposition: either an atomic proposition, $\underleftarrow{a}$ or $\underrightarrow{a}$, or a recursively defined proposition, $\hat{a}$.

Given a choregraphing assignment θ, we may construct a choregraphy from the string rewriting specification. Intuitively, each axiom is annotated according to θ, and then the resulting [...] are used to construct a family of recursive definitions, one for each $\hat{a}$ in the image of θ.

A choreography is an ordered rewriting specification that simulates the string rewriting specification [...].

Consider the recurring string rewriting specification with axioms

$$\overline{a\,b \longrightarrow b} \qquad \text{and} \qquad \overline{b \longrightarrow \epsilon}\,.$$

We must consistently annotate each symbol as either a left-directed atom, right-directed atom, or recursively defined proposition in sucha way that each axiom's premise w has the form $w_1\,a\,w_2$ with

$$\underrightarrow{a}\,\hat{b} \longrightarrow \hat{b} \qquad \text{and} \qquad \hat{b} \longrightarrow \cdot$$

Now we must solve for $\hat{b}$, determining a definition $\hat{b} \triangleq B^-$ such that these two rewriting steps are derivable. For the first step to be derivable, $\hat{b}$ should have a definition that is consistent with $\underrightarrow{a} \setminus {\uparrow}{\downarrow}\hat{b}$, for

$$\underrightarrow{a}\,(\underrightarrow{a} \setminus {\uparrow}{\downarrow}\hat{b}) \longrightarrow \hat{b}$$

Consider the choreographing assignment θ that maps a to the atom $\underrightarrow{a}$ and b to the recursively defined proposition $\hat{b}$. Upon annotating the above string rewriting axioms according to θ, we arrive at the [...]

$$\underset{\rightarrow}{a}\,\hat{b} \longrightarrow \hat{b} \qquad \text{and} \qquad \hat{b} \longrightarrow \mathbf{1}\,.$$

$$\hat{b} \triangleq \underset{\rightarrow}{a} \setminus {\uparrow}{\downarrow}\hat{b} \qquad \text{and} \qquad \hat{b} \triangleq {\uparrow}\mathbf{1}\,,$$

respectively. Combining these into a single definition that allows a nondeterministic choice between the two, we have

$$\hat{b} \triangleq (\underset{\rightarrow}{a} \setminus {\uparrow}{\downarrow}\hat{b}) \mathbin{\&} {\uparrow}\mathbf{1}\,,$$

or $\hat{b} \triangleq (\underset{\rightarrow}{a} \setminus {\uparrow}\hat{b}) \mathbin{\&} \mathbf{1}$ if the minimally necessary shifts are elided.

By construction, this choreography is adequate with respect to the specification, in the sense that it can simulate each of the specification's possible steps and vice versa.

- $w \longrightarrow w'$ only if $\theta(w) \longrightarrow \theta(w')$ For example, just as the string rewriting specification admits $a\,b \longrightarrow b$, the ordered rewriting choreography admits

$$\theta(a\,b) = \underset{\rightarrow}{a}\,\hat{b} = \underset{\rightarrow}{a}\,\big((\underset{\rightarrow}{a} \setminus {\uparrow}{\downarrow}\hat{b}) \mathbin{\&} \mathbf{1}\big) \longrightarrow \hat{b} = \theta(b)\,.$$

- $\theta(w) \longrightarrow \Omega'$ only if $w \longrightarrow \theta^{-1}(\Omega')$ For example, just as the ordered rewriting choreography admits $\theta(b) = \hat{b} \longrightarrow \cdot$, the string rewriting specification admits $b \longrightarrow \epsilon = \theta^{-1}(\cdot)$.

4.3.3 *Formal description*

In this ??, we present a more formal description of the above procedure for choreographing string rewriting specifications. We define a judgment $\theta \vdash \Theta \rightsquigarrow \Phi$ that, when given a string rewriting specification (Σ, Θ) and [...] assignment θ, yields a formula-as-process ordered rewriting signature Φ that makes θ a bisimulation [between $\longrightarrow_\Theta$ and $\longrightarrow_\Phi$] if such a signature exists:

$$\theta \vdash \Theta \rightsquigarrow \Phi \quad \text{implies} \quad \begin{tikzcd} w \arrow[r, "\Theta"'] \arrow[d, "\theta"'] & w' \arrow[d, dashed, "\theta"] \\ \Omega \arrow[r, dashed, "\Phi"'] & \Omega' \end{tikzcd} \quad \text{and} \quad \begin{tikzcd} w \arrow[r, dashed, "\Theta"'] \arrow[d, "\theta"'] & w' \arrow[d, dashed, "\theta"] \\ \Omega \arrow[r, "\Phi"'] & \Omega' \end{tikzcd}\,.$$

This principal judgment also relies on an auxiliary judgment, $\underset{\rightarrow}{\Omega}_L \setminus {\uparrow}A^+ / \underset{\leftarrow}{\Omega}_R \rightsquigarrow B^-$.

THE AUXILIARY judgment $\underset{\rightarrow}{\Omega}_L \setminus {\uparrow}A^+ / \underset{\leftarrow}{\Omega}_R \rightsquigarrow B^-$ elaborates[30] the quasi-proposition $\underset{\rightarrow}{\Omega}_L \setminus {\uparrow}A^+ / \underset{\leftarrow}{\Omega}_R$ into a well-formed proposition B^- by nondeterministically abstracting one-by-one from either the left or right contexts.[31] This proposition B^- is semantically equivalent to the quasi-proposition $\underset{\rightarrow}{\Omega}_L \setminus {\uparrow}A^+ / \underset{\leftarrow}{\Omega}_R$ in the sense that the two intuitively satisfy the same "left-focus judgments": $\underset{\rightarrow}{\Delta}_L\ [B^-]\ \underset{\leftarrow}{\Delta}_R \Vdash C^+$ if, and only if, $\underset{\rightarrow}{\Delta}_L = \underset{\rightarrow}{\Omega}_L$ and $\underset{\leftarrow}{\Delta}_R = \underset{\leftarrow}{\Omega}_R$ and $C^+ = A^+$. (This is proved below as lemma 4.2.)

[30] word choice?

[31] This procedure could be made deterministic by preferring one side over the other, but we refrain from doing so because the choice of side to prefer is completely arbitrary.

The auxiliary elaboration judgment is defined inductively by the following rules.

$$\frac{}{(\cdot) \setminus {\uparrow}A^+ \mathrel{/} (\cdot) \rightsquigarrow {\uparrow}A^+}\ {\uparrow}\textsc{q}$$

$$\frac{\underrightarrow{\Omega}_L \setminus {\uparrow}A^+ \mathrel{/} \underleftarrow{\Omega}_R \rightsquigarrow B^-}{(\underrightarrow{\Omega}_L\, \underrightarrow{a}^+) \setminus {\uparrow}A^+ \mathrel{/} \underleftarrow{\Omega}_R \rightsquigarrow \underrightarrow{a}^+ \setminus B^-}\ \setminus\textsc{q} \qquad \frac{\underrightarrow{\Omega}_L \setminus {\uparrow}A^+ \mathrel{/} \underleftarrow{\Omega}_R \rightsquigarrow B^-}{\underrightarrow{\Omega}_L \setminus {\uparrow}A^+ \mathrel{/} (\underleftarrow{a}^+\, \underleftarrow{\Omega}_R) \rightsquigarrow B^- \mathrel{/} \underleftarrow{a}^+}\ /\textsc{q}$$

The $\setminus\textsc{q}$ rule states that the quasi-proposition $(\underrightarrow{\Omega}_L\, \underrightarrow{a}^+) \setminus {\uparrow}A^+ \mathrel{/} \underleftarrow{\Omega}_R$ is equivalent to $\underrightarrow{a}^+ \setminus B^-$ if $\underrightarrow{\Omega}_L \setminus {\uparrow}A^+ \mathrel{/} \underleftarrow{\Omega}_R$ is equivalent to B^-. Notice that the $\setminus\textsc{q}$ rule moves $\underrightarrow{a}^+$from the right of $\underrightarrow{\Omega}_L$ to the left of B^-; this is admittedly counterintuitive, but it is closely related to the equally counterintuitive currying law for left-handed implication in ordered logic: $(A \bullet B) \setminus C \dashv\Vdash B \setminus (A \setminus C)$. Symmetrically, the $/\textsc{q}$ rule is closely related to the currying law for right-handed implication: $C \mathrel{/} (A \bullet B) \dashv\Vdash (C \mathrel{/} B) \mathrel{/} A$.

This intuition is captured in the proof of the following lemma.

Lemma 4.2. *If $\underrightarrow{\Omega}_L \setminus {\uparrow}A^+ \mathrel{/} \underleftarrow{\Omega}_R \rightsquigarrow B^-$, then $\underrightarrow{\Delta}_L\, [B^-]\, \underleftarrow{\Delta}_R \Vdash C^+$ if, and only if, $\underrightarrow{\Delta}_L = \underrightarrow{\Omega}_L$ and $\underleftarrow{\Delta}_R = \underleftarrow{\Omega}_R$ and $C^+ = A^+$.*

Proof. By induction over the structure of the given elaboration.

As an example case, consider

$$\frac{\underrightarrow{\Omega}_L \setminus {\uparrow}A^+ \mathrel{/} \underleftarrow{\Omega}_R \rightsquigarrow B^-}{(\underrightarrow{\Omega}_L\, \underrightarrow{a}^+) \setminus {\uparrow}A^+ \mathrel{/} \underleftarrow{\Omega}_R \rightsquigarrow \underrightarrow{a}^+ \setminus B^-}\,.$$

We must show that $\underrightarrow{\Delta}_L\, [\underrightarrow{a}^+ \setminus B^-]\, \underleftarrow{\Delta}_R \Vdash C^+$ if, and only if, $\underrightarrow{\Delta}_L = \underrightarrow{\Omega}_L\, \underrightarrow{a}^+$ and $\underleftarrow{\Delta}_R = \underleftarrow{\Omega}_R$ and $C^+ = A^+$. Indeed, the $\setminus\textsc{l}$ rule is the unique rule for left-focusing on the proposition $\underrightarrow{a}^+ \setminus B^-$, so $\underrightarrow{\Delta}_L\, [\underrightarrow{a}^+ \setminus B^-]\, \underleftarrow{\Delta}_R \Vdash C^+$ if, and only if, $\underrightarrow{\Delta}_L = \underrightarrow{\Delta}'_L\, \underrightarrow{a}^+$ and $\underrightarrow{\Delta}'_L\, [B^-]\, \underleftarrow{\Delta}_R \Vdash C^+$ for some $\underrightarrow{\Delta}'_L$. By the inductive hypothesis, we have $\underrightarrow{\Delta}'_L\, [B^-]\, \underleftarrow{\Delta}_R \Vdash C^+$ if, and only if, $\underrightarrow{\Delta}'_L = \underrightarrow{\Omega}_L$ and $\underleftarrow{\Delta}_R = \underleftarrow{\Omega}_R$ and $C^+ = A^+$. Putting everything together, $\underrightarrow{\Delta}_L\, [\underrightarrow{a}^+ \setminus B^-]\, \underleftarrow{\Delta}_R \Vdash C^+$ if, and only if, $\underrightarrow{\Delta}_L = \underrightarrow{\Omega}_L\, \underrightarrow{a}^+$ and $\underleftarrow{\Delta}_R = \underleftarrow{\Omega}_R$ and $C^+ = A^+$, as required. □

The principal judgment is $\theta \vdash \Theta \rightsquigarrow \Phi$. Given a [...] assignment θ and a string rewriting signature Θ, this judgment produces a formula-as-process ordered rewriting signature Φ that, together with θ, constitutes a [well-specified][32] choreography of the string rewriting specification (Σ, Θ).

[32] ?

In other words, Φ is a solution to $\theta(\Theta)$, the rewritings induced by θ from the string rewriting axioms Θ. That is, if $\theta \vdash \Theta \rightsquigarrow \Phi$, then θ is a (strong) bisimulation between $\longrightarrow_\Theta$ and $\longrightarrow_\Phi$. [33] If $\theta \vdash \Theta \rightsquigarrow \Phi$ is not derivable for any Φ, then the [...] assignment θ does not yield a [well-specified][34] choreography of Θ.

$$\begin{tikzcd} w \arrow[r, "\Theta"'] \arrow[d, "\theta"'] & w' \arrow[d, dashed, "\theta"] \\ \theta(w) \arrow[r, dashed, "\Phi"'] & \theta(w') \end{tikzcd} \quad \text{and} \quad \begin{tikzcd} w \arrow[r, dashed, "\Theta"'] \arrow[d, "\theta"'] & w' \arrow[d, dashed, "\theta"] \\ \theta(w) \arrow[r, "\Phi"'] & \Omega' = \theta(w') \end{tikzcd}$$

[33] Actually, we end up proving a stronger soundness result in ??.

[34] ?

This choreographing judgment is defined by just two rules:

$$\frac{}{\theta \vdash \cdot \rightsquigarrow \cdot}$$

$$\frac{\begin{array}{c}(\theta(a) = \hat{a}) \quad \theta \vdash \Theta_0 \rightsquigarrow \Phi_0 \quad (\hat{a} \notin \operatorname{dom} \Phi_0) \\ \forall i \in \mathcal{I}:\ \left(\theta(w_i^L) = \underset{\rightarrow}{\Omega}{}_i^L\right) \quad \left(\theta(w_i^R) = \underset{\leftarrow}{\Omega}{}_i^R\right) \quad \underset{\rightarrow}{\Omega}{}_i^L \setminus {\uparrow}{\bullet}\theta(w_i') \,/\, \underset{\leftarrow}{\Omega}{}_i^R \rightsquigarrow A_i^-\end{array}}{\theta \vdash \Theta_0, \left(w_i^L\, a\, w_i^R \longrightarrow w_i'\right)_{i \in \mathcal{I}} \rightsquigarrow \Phi_0, \left(\hat{a} \triangleq \&_{i \in \mathcal{I}} A_i^-\right)}$$

The first of these rules is straightforward: an empty string rewriting signature choreographs as an empty ordered rewriting signature. The second rule is quite a lot to parse and needs to be broken down step by step:

1. Choose a symbol a that is mapped by θ to a recursively defined proposition, $\hat{a}$. Then reorganize the SR signature Θ, collecting together all axioms in Θ that have an a in their premises. Let $\left(w_i^L\, a\, w_i^R \longrightarrow w_i'\right)_{i \in \mathcal{I}}$ be those axioms, so that $\Theta = \Theta_0, \left(w_i^L\, a\, w_i^R \longrightarrow w_i'\right)_{i \in \mathcal{I}}$ for some Θ_0.

2. Construct a well-specified choreography Φ_0 from Θ_0 and θ, using the judgment $\theta \vdash \Theta_0 \rightsquigarrow \Phi_0$. Check that Φ_0 gives no definition for $\hat{a}$, otherwise there is some axiom in Θ_0 that contains a in its premise and $\left(w_i^L\, a\, w_i^R \longrightarrow w_i'\right)_{i \in \mathcal{I}}$ does not correctly constitute all such axioms.

3. Check that each w_i^L contains only those symbols that map to right-directed atoms, using the side condition $\theta(w_i^L) = \underset{\rightarrow}{\Omega}{}_i^L$. Symmetrically, check that each w_i^R contains only symbols that map to left-directed atoms, using the side condition $\theta(w_i^R) = \underset{\leftarrow}{\Omega}{}_i^R$.

4. Elaborate each quasi-proposition $\underset{\rightarrow}{\Omega}{}_i^L \setminus {\uparrow}{\bullet}\theta(w_i') \,/\, \underset{\leftarrow}{\Omega}{}_i^R$ into a semantically equivalent proposition A_i^-. Based on ??, $\theta(w_i^L)\, [A_i^-]\, \theta(w_i^R) \Vdash \bullet\theta(w_i')$, and so this proposition acts as the image of the axiom $w_i^L\, a\, w_i^R \longrightarrow w_i'$ under θ – that is, $\theta(w_i^L)\, A_i^-\, \theta(w_i^R) \longrightarrow \theta(w_i')$.

5. Collect the A_i^-s into a single definition, $\hat{a} \triangleq \&_{i \in \mathcal{I}} A_i^-$, which, based on steps 2 and 4, describes all of the axioms from Θ that contain a in their premises.

If $\theta \vdash \Theta \rightsquigarrow \Phi$, then θ is a bisimulation. That is, $\theta \vdash \Theta \rightsquigarrow \Phi$ implies

$$\begin{tikzcd} w \arrow[r, "\Theta"'] \arrow[d, "\theta"'] & w' \arrow[d, dashed, "\theta"] \\ \theta(w) \arrow[r, dashed, "\Phi"'] & \theta(w') \end{tikzcd} \quad \text{and} \quad \begin{tikzcd} w \arrow[r, dashed, "\Theta"'] \arrow[d, "\theta"'] & w' \arrow[d, dashed, "\theta"] \\ \theta(w) \arrow[r, "\Phi"'] & \Omega' = \theta(w') \end{tikzcd}.$$

As stated earlier, when $\theta \vdash \Theta \rightsquigarrow \Phi$, the string rewriting step $w \longrightarrow_\Theta w'$ holds if, and only if, the ordered rewriting step $\theta(w) \longrightarrow_\Phi \theta(w')$ holds. We prove the left-to-right direction as the following completeness ?? and then prove a stronger soundness ?? that implies the right-to-left direction.

LEMMA 4.3 (Weakening). *If* $\Omega \longrightarrow_\Phi \Omega'$ *and* $\operatorname{dom} \Phi \cap \operatorname{dom} \Phi' = \emptyset$*, then* $\Omega \longrightarrow_{\Phi,\Phi'} \Omega'$.

Proof. By induction over the structure of the given rewriting step. □

THEOREM 4.4 (Completeness). *If* $\theta \vdash \Theta \rightsquigarrow \Phi$, *then* $w \longrightarrow_\Theta w'$ *implies* $\theta(w) \longrightarrow_\Phi \theta(w')$.

If $\theta \vdash \Theta \rightsquigarrow \Phi$, then
$$\begin{tikzcd} w \arrow[r, "\Theta"'] \arrow[d, "\theta"'] & w' \arrow[d, dashed, "\theta"] \\ \theta(w) \arrow[r, dashed, "\Phi"'] & \theta(w') \end{tikzcd}$$

Proof. By simultaneous structural induction on the given choreographing derivation, $\theta \vdash \Theta \rightsquigarrow \Phi$, and ordered rewriting step, $w \longrightarrow_\Theta w'$.

- Consider the case in which

$$\theta \vdash \Theta \rightsquigarrow \Phi \qquad \text{and} \qquad w = \dfrac{w_0 \longrightarrow_\Theta w_0'}{w_1\, w_0\, w_2 \longrightarrow_\Theta w_1\, w_0'\, w_2} \longrightarrow\mathrm{C} = w'\,.$$

  By the inductive hypothesis, $\theta(w_0) \longrightarrow_\Phi \theta(w_0')$. It follows from ordered rewriting's $\longrightarrow$C rule that

$$\theta(w) = \theta(w_1)\,\theta(w_0)\,\theta(w_2) \longrightarrow_\Phi \theta(w_1)\,\theta(w_0')\,\theta(w_2) = \theta(w')\,.$$

- Consider the case in which

$$\dfrac{\begin{array}{c}(\theta(a) = \hat{a}) \quad \theta \vdash \Theta_0 \rightsquigarrow \Phi_0 \quad (\hat{a} \notin \operatorname{dom}\Phi_0) \\ \forall i \in \mathcal{I}:\ \ (\theta(w_i^L) = \underset{\rightarrow}{\Omega}{}_i^L) \quad (\theta(w_i^R) = \underset{\leftarrow}{\Omega}{}_i^R) \quad \underset{\rightarrow}{\Omega}{}_i^L \setminus {\uparrow}{\bullet}\theta(w_i') \,/\, \underset{\leftarrow}{\Omega}{}_i^R \rightsquigarrow A_i^-\end{array}}{\theta \vdash \Theta_0, \left(w_i^L\, a\, w_i^R \longrightarrow w_i'\right)_{i\in\mathcal{I}} \rightsquigarrow \Phi_0, \left(\hat{a} \triangleq \mathbin{\&}_{i\in\mathcal{I}} A_i^-\right)}$$

  and

$$w = \dfrac{(w_k^L\, a\, w_k^R \longrightarrow w_k') \in \Theta}{w_k^L\, a\, w_k^R \longrightarrow_\Theta w_k'} \longrightarrow\mathrm{AX} = w'$$

  for some $k \in \mathcal{I}$, where $\Theta = \Theta_0, (w_i^L\, a\, w_i^R \longrightarrow w_i')_{i\in\mathcal{I}}$ and $\Phi = \Phi_0, (\mathbin{\&}_{i\in\mathcal{I}} A_i^-)$.

  By ??, $\theta(w_k^L)\,[A_k^-]\,\theta(w_k^R) \Vdash \bullet\theta(w_k')$. $\theta(w_k^L)\,[\mathbin{\&}_{i\in\mathcal{I}} A_i^-]\,\theta(w_k^R) \Vdash \bullet\theta(w_k')$. Because $[\bullet\theta(w_k')] \dashv\!\!\!\mid\; \theta(w_k')$ (??), it follows by the $\longrightarrow$I rule that $\theta(w_k^L)\,\left(\mathbin{\&}_{i\in\mathcal{I}} A_i^-\right)\theta(w_k^R) \longrightarrow_\Phi \theta(w_k')$, and so $\theta(w) = \theta(w_k^L)\,\hat{a}\,\theta(w_k^R) \longrightarrow_\Phi \theta(w_k') = \theta(w')$.

- Consider the case in which

$$\dfrac{\begin{array}{c}(\theta(a) = \hat{a}) \quad \theta \vdash \Theta_0 \rightsquigarrow \Phi_0 \quad (\hat{a} \notin \operatorname{dom}\Phi_0) \\ \forall i \in \mathcal{I}:\ \ (\theta(v_i^L) = \underset{\rightarrow}{\Omega}{}_i^L) \quad (\theta(v_i^R) = \underset{\leftarrow}{\Omega}{}_i^R) \quad \underset{\rightarrow}{\Omega}{}_i^L \setminus {\uparrow}{\bullet}\theta(v_i') \,/\, \underset{\leftarrow}{\Omega}{}_i^R \rightsquigarrow A_i^-\end{array}}{\theta \vdash \Theta_0, \left(v_i^L\, a\, v_i^R \longrightarrow v_i'\right)_{i\in\mathcal{I}} \rightsquigarrow \Phi_0, \left(\hat{a} \triangleq \mathbin{\&}_{i\in\mathcal{I}} A_i^-\right)}$$

  and

$$\dfrac{(w \longrightarrow w') \in \Theta_0}{w \longrightarrow_\Theta w'} \longrightarrow\mathrm{AX}$$

  where $(w \longrightarrow w') \in \Theta_0$ and $\Theta = \Theta_0, (v_i^L\, a\, v_i^R \longrightarrow v_i')_{i\in\mathcal{I}}$ and $\Phi = \Phi_0, (\mathbin{\&}_{i\in\mathcal{I}} A_i^-)$.

  By the inductive hypothesis, $\theta(w) \longrightarrow_{\Phi_0} \theta(w')$. It follows from weakening (??) that $\theta(w) \longrightarrow_\Phi \theta(w')$.

- The case in which

$$\dfrac{}{\theta \vdash \cdot \rightsquigarrow \cdot} \qquad \text{and} \qquad \dfrac{(w \longrightarrow w') \in \Theta}{w \longrightarrow_\Theta w'} \longrightarrow\mathrm{AX}$$

  where $\Theta = \cdot$ and $\Phi = \cdot$ is vacuous.

□

Lemma 4.5. *If $\theta \vdash \Theta \rightsquigarrow \Phi$ and $\underrightarrow{\Omega}_L\ [\hat{a}]\ \underleftarrow{\Omega}_R \Vdash_\Phi C^+$, then there exists an axiom $(w_1\, a\, w_2 \longrightarrow w') \in \Theta$ such that $\underrightarrow{\Omega}_L = \theta(w_1)$, $\underleftarrow{\Omega}_R = \theta(w_2)$, and $C^+ = \bullet\theta(w')$.*

Proof. By induction over the structure of the given choreographing derivation, $\theta \vdash \Theta \rightsquigarrow \Phi$.

- Consider the case in which

$$\frac{\begin{array}{c}\theta \vdash \Theta_0 \rightsquigarrow \Phi_0 \quad (\theta(a) = \hat{a}) \quad (\hat{a} \notin \operatorname{dom} \Phi_0) \\ \forall i \in \mathcal{I}: \quad \big(\theta(w_i^L) = \underrightarrow{\Delta}_i^L\big) \quad \big(\theta(w_i^R) = \underleftarrow{\Delta}_i^R\big) \quad \underrightarrow{\Delta}_i^L \setminus {\uparrow}\bullet\theta(w_i') \,/\, \underleftarrow{\Delta}_i^R \rightsquigarrow A_i^-\end{array}}{\theta \vdash \Theta_0, \big(w_i^L\, a\, w_i^R \longrightarrow w_i'\big)_{i\in\mathcal{I}} \rightsquigarrow \Phi_0, \big(\hat{a} \triangleq \&_{i\in\mathcal{I}}\, A_i^-\big)}$$

  and

$$\underrightarrow{\Omega}_L\ [\hat{a} = \&_{i\in\mathcal{I}}\, A_i^-]\ \underleftarrow{\Omega}_R \Vdash_\Phi C^+$$

  where $\Theta = \Theta_0, \big(w_i^L\, a\, w_i^R \longrightarrow w_i'\big)_{i\in\mathcal{I}}$ and $\Phi = \Phi_0, \big(\hat{a} \triangleq \&_{i\in\mathcal{I}}\, A_i^-\big)$.

  By inversion on the left-focus derivation, either: $\underrightarrow{\Omega}_L\ [A_k^-]\ \underleftarrow{\Omega}_R \Vdash C^+$ for some $k \in \mathcal{I}$; or $\mathcal{I}$ is empty.

  – If $\underrightarrow{\Omega}_L\ [A_k^-]\ \underleftarrow{\Omega}_R \Vdash C^+$ for some $k \in \mathcal{I}$, then ?? allows us to conclude that $\underrightarrow{\Omega}_L = \underrightarrow{\Delta}_k^L = \theta(w_k^L)$ and $\underleftarrow{\Omega}_R = \underleftarrow{\Delta}_k^R = \theta(w_k^R)$ and $C^+ = \bullet\theta(w_k')$. Also, the axiom $w_k^L\, a\, w_k^R \longrightarrow w_k'$ is contained in Θ.

  – Otherwise, if $\mathcal{I}$ is empty, then $\&_{i\in\mathcal{I}}\, A_i^- = \top$. There is no $\top$L rule to derive $\underrightarrow{\Omega}_L\ [\hat{a} = \top]\ \underleftarrow{\Omega}_R \Vdash_\Phi C^+$, so this case is vacuous.

- Consider the case in which

$$\frac{\begin{array}{c}\theta \vdash \Theta_0 \rightsquigarrow \Phi_0 \quad (\theta(b) = \hat{b}) \quad (\hat{b} \notin \operatorname{dom} \Phi_0) \\ \forall i \in \mathcal{I}: \quad \big(\theta(v_i^L) = \underrightarrow{\Delta}_i^L\big) \quad \big(\theta(v_i^R) = \underleftarrow{\Delta}_i^R\big) \quad \underrightarrow{\Delta}_i^L \setminus {\uparrow}\bullet\theta(v_i') \,/\, \underleftarrow{\Delta}_i^R \rightsquigarrow B_i^-\end{array}}{\theta \vdash \Theta_0, \big(v_i^L\, b\, v_i^R \longrightarrow v_i'\big)_{i\in\mathcal{I}} \rightsquigarrow \Phi_0, \big(\hat{b} \triangleq \&_{i\in\mathcal{I}}\, B_i^-\big)}$$

  and

$$\underrightarrow{\Omega}_L\ [\hat{a}]\ \underleftarrow{\Omega}_R \Vdash_\Phi C^+$$

  where $a \neq b$ and $\Theta = \Theta_0, \big(v_i^L\, b\, v_i^R \longrightarrow v_i'\big)_{i\in\mathcal{I}}$ and $\Phi = \Phi_0, \big(\hat{b} \triangleq \&_{i\in\mathcal{I}}\, B_i^-\big)$.

  By the inductive hypthesis, there exists a string rewriting axiom $(w_1\, a\, w_2 \longrightarrow w') \in \Theta_0$ such that $\underrightarrow{\Omega}_L = \theta(w_1)$ and $\underleftarrow{\Omega}_R = \theta(w_2)$ and $C^+ = \bullet\theta(w')$. The same axiom is contained in the signature Θ.

- The case in which

$$\frac{}{\theta \vdash \cdot \rightsquigarrow \cdot} \qquad \text{and} \qquad \underrightarrow{\Omega}_L\ [\hat{a}]\ \underleftarrow{\Omega}_R \Vdash_\Phi C^+$$

  where $\Theta = \cdot$ and $\Phi = \cdot$ is vacuous because there is no definition for $\hat{a}$ in the signature Φ.

□

Theorem 4.6 (Soundness). *If $\theta \vdash \Theta \rightsquigarrow \Phi$ and $\theta(a) = \hat{a}$ and $\Omega_L\, \hat{a}\, \Omega_R \longrightarrow_\Phi \Omega'$, then either:*

- $\Omega_L = \Omega'_L\, \theta(w_1)$ *and* $\Omega_R = \theta(w_2)\, \Omega'_R$ *and* $\Omega' = \Omega'_L\, \theta(w')\, \Omega'_R$ *for some contexts* Ω'_L *and* Ω'_R *and some strings* w_1, w_2, *and* w' *such that* $(w_1\, a\, w_2 \longrightarrow w') \in \Theta$ *and* $\theta(w_1)\ [\hat{a}]\ \theta(w_2) \Vdash \bullet\theta(w')$;
- $\Omega_L \longrightarrow_\Phi \Omega'_L$ *for some context* Ω'_L *such that* $\Omega' = \Omega'_L\, \hat{a}\, \Omega_R$; *or*
- $\Omega_R \longrightarrow_\Phi \Omega'_R$ *for some context* Ω'_R *such that* $\Omega' = \Omega_L\, \hat{a}\, \Omega'_R$.

Proof. As a negative proposition, $\hat{a}$ serves as a barrier for interactions between Ω_L and Ω_R – in PFOR, implications cannot consume negative propositions. Thus, any reduction on $\Omega_L\, \hat{a}\, \Omega_R$ must occur within either Ω_L or Ω_R alone or must arise from $\hat{a}$.

If the reduction on $\Omega_L\, \hat{a}\, \Omega_R$ arises from $\hat{a}$, then it arises from a bipole that begins by focusing on $\hat{a}$. In other words, $\Omega_L = \Omega'_L\, \underrightarrow{\Delta}_L$ and $\Omega_R = \underleftarrow{\Delta}_R\, \Omega'_R$ and $\Omega' = \Omega'_L\, \Delta'\, \Omega'_R$ for some contexts $\underrightarrow{\Delta}_L$, $\underleftarrow{\Delta}_R$, and Δ' and positive proposition C^+ such that $\underrightarrow{\Delta}_L\ [\hat{a}]\ \underleftarrow{\Delta}_R \Vdash C^+$ and $[C^+] \dashv\!\!\!\!\;\Vert\ \Delta'$. By ??, there exists an axiom $(w_1\, a\, w_2 \longrightarrow w') \in \Theta$ such that $\underrightarrow{\Delta}_L = \theta(w_1)$ and $\underleftarrow{\Delta}_R = \theta(w_2)$ and $C^+ = \bullet\theta(w')$. It follows that $\Delta' = \theta(w')$. □

COROLLARY 4.7 (Soundness). *If* $\theta \vdash \Theta \rightsquigarrow \Phi$ *and* $\theta(w) \longrightarrow_\Phi \Omega'$, *then* $\Omega' = \theta(w')$ *for some* w' *such that* $w \longrightarrow_\Theta w'$.

4.3.4 *No choreography*

Not all string rewriting specifications admit a choreography. For example, the specification

$$\frac{}{a\,b \longrightarrow b} \qquad \frac{}{a \longrightarrow \epsilon} \qquad \text{and} \qquad \frac{}{b \longrightarrow \epsilon}$$

cannot be given a choreography. More precisely, there is no choreographing assignment θ such that $\theta \vdash \Sigma \rightsquigarrow \Sigma'$ is derivable for some signature Σ'. For the sake of contradiction, suppose that θ were such a choreographing assignment. Then, for the specification's latter two axioms to be choreographable, both $\theta(a) = \hat{a}$ and $\theta(b) = \hat{b}$ must hold. In that case, however, the specification's first axiom cannot be choreographed properly because θ maps more than one of the axiom's symbols to recursively defined propositions.

4.4 *Encoding nondeterministic finite automata*

Recall from **??** our string rewriting specification of how an NFA processes its input. Given an NFA $\mathcal{A} = (Q, ?, F)$[35] over an input alphabet Σ, the NFA's operational semantics is adequately captured by the following string rewriting axioms: Θ is the least SR signature such that

[35] fix

$$(a\,q \longrightarrow q'_a) \in \Theta \text{ for each transition } q \xrightarrow{a} q'_a.$$

$$(\epsilon\,q \longrightarrow F(q)) \in \Theta \text{ for each state } q\text{, where } F(q) = \begin{cases} (\cdot) & \text{if } q \in F \\ n & \text{if } q \notin F\,. \end{cases}$$

We would like to choreograph this string rewriting specification. There are, in fact, two distinct choreographies for this specification.

4.4.1 *A functional choreography*

One possible choreography for this specification interprets each input symbol $a \in \Sigma$ as a right-directed atom, $\underset{\rightarrow}{a}$; each state $q \in Q$ as a recursively defined proposition, $\hat{q}$; and the end-of-word marker, ϵ, as a right-directed atom, $\underset{\rightarrow}{\epsilon}$. In other words, the input string is transmitted as a sequence of messages to a process $\hat{q}$ that tracks the NFA's current state.

Under this choreographing assignment, the string rewriting axioms become rewriting steps that must be derivable:

$$\underset{\rightarrow}{a}\,\hat{q} \longrightarrow \hat{q}'_a \underset{\rightarrow}{\epsilon}\,\hat{q} \longrightarrow \begin{cases} & \text{if } q \in F \\ & \text{if } q \notin F \end{cases}$$

These required rewritings are indeed local: each one contains exactly one recursively defined process in its premise, with all remaining propositions in its premise being input messages for that process. For instance, each $\underset{\rightarrow}{a}\,\hat{q} \longrightarrow$ $\hat{q}'_a$

Solving for each recursively defined proposition, we have one definition,

$$\hat{q} \triangleq \mathop{\&}_{a \in \Sigma} \mathop{\&}_{q'_a} (\underset{\rightarrow}{a} \setminus \hat{q}'_a) \mathbin{\&} (\underset{\rightarrow}{\epsilon} \setminus \hat{F}(q))\,,$$

for each NFA state $q \in Q$. This choreography might be called 'functional' because the data, an input string, are represented by messages that are acted on in a function-like way by the current state's process, $\hat{q}$.

THEOREM 4.8. *$\theta \vdash \Theta \rightsquigarrow \Phi$ is derivable.*

The proof of this theorem is a completely straightforward, if tedious, formalization of the preceding intuition, along the line of ??.

COROLLARY 4.9. *If $q \xrightarrow{a} q'_a$, then $\underset{\rightarrow}{a}\,\hat{q} \Longrightarrow \hat{q}'_a$. If $\underset{\rightarrow}{a}\,\hat{q} \longrightarrow \Omega'$, then $\Omega' \Longrightarrow \hat{q}'_a$ for some q'_a that a-succeeds q. If $q \xrightarrow{a} q'_a$, then $\underset{\rightarrow}{a}\,\hat{q} \Longrightarrow \hat{q}'_a$. If $\underset{\rightarrow}{a}\,\hat{q} \longrightarrow \Omega'$, then $\Omega' \Longrightarrow \hat{q}'_a$ for some q'_a that a-succeeds q.*

COROLLARY 4.10. *If $q \xrightarrow{a} q'_a$, then $\underset{\rightarrow}{a}\,\hat{q} \Longrightarrow \hat{q}'_a$. If $\underset{\rightarrow}{a}\,\hat{q} \longrightarrow \Omega'$, then $\Omega' = \hat{q}'_a$ for some q'_a that a-succeeds q.*

THEOREM 4.11. *Let $\mathcal{A} = (\ldots)$ be a DFA over the input alphabet Σ. $q \sim s$ if, and only if, $\hat{q} = \hat{s}$, for all states q and s.*

Proof. ... □

But the same does not hold for NFAs.

Counterexample. ... □

4.4.2 *An object-oriented choreography*

The functional choreography for our string rewriting specification of NFAs is not the only possible choreography. Instead of treating the input symbols as messages and the states as processes, we may uae a dual assignment:

$$a\,q \longrightarrow q'_a becomes \hat{a}\,\underset{\leftarrow}{q} \longrightarrow \underset{\leftarrow}{q}'_a \epsilon\,q \longrightarrow F(q) becomes \hat{\epsilon}\,\underset{\leftarrow}{q} \longrightarrow ?$$

Solving for the recursively defined propositions $\hat{a}$ for each $a \in \Sigma$ and $\hat{\epsilon}$, we have the following definitions:

$$\Phi = \left(\hat{a} \triangleq \mathop{\&}_{q \in Q} \mathop{\&}_{q'_a} (\underset{\leftarrow}{q}'_a \mathbin{/} \underset{\leftarrow}{q})\right)_{a \in \Sigma}\,,$$
$$\hat{\epsilon} \triangleq \mathop{\&}_{q \in Q} (\hat{F}(q) \mathbin{/} \underset{\leftarrow}{q})\,.$$

COROLLARY 4.12. *If $q \xrightarrow{a} q'_a$, then $\hat{a}\,\underset{\leftarrow}{q} \Longrightarrow \underset{\leftarrow}{q}'_a$. If $\hat{a}\,\underset{\leftarrow}{q} \longrightarrow \Omega'$, then $\Omega' \Longrightarrow \underset{\leftarrow}{q}'_a$ for some q'_a that a-succeeds q.*

Define a relation on input symbols a and b such that $\hat{a} = \hat{b}$. Two symbols are then related exactly when they lead to the same successor states.

Axioms, Θ	*Rewriting constraints on* Φ	*Solution,* Φ
$e\,i \longrightarrow e\,b_1$ and $e\,d \longrightarrow z$	$\hat{e}\,\underleftarrow{i} \longrightarrow_\Phi \hat{e}\,\hat{b}_1$ and $\hat{e}\,\underleftarrow{d} \longrightarrow_\Phi \underrightarrow{z}$	$\hat{e} \triangleq (\hat{e} \bullet \hat{b}_1 \mathbin{/} \underleftarrow{i}) \mathbin{\&} (\underrightarrow{z} \mathbin{/} \underleftarrow{d})$
$b_0\,i \longrightarrow b_1$ and $b_0\,d \longrightarrow d\,b_0'$	$\hat{b}_0\,\underleftarrow{i} \longrightarrow_\Phi \hat{b}_1$ and $\hat{b}_0\,\underleftarrow{d} \longrightarrow_\Phi \underleftarrow{d}\,\hat{b}_0'$	$\hat{b}_0 \triangleq ({\uparrow}{\downarrow}\hat{b}_1 \mathbin{/} \underleftarrow{i}) \mathbin{\&} (\underleftarrow{d} \bullet \hat{b}_0' \mathbin{/} \underleftarrow{d})$
$b_1\,i \longrightarrow i\,b_0$ and $b_1\,d \longrightarrow b_0\,s$	$\hat{b}_1\,\underleftarrow{i} \longrightarrow_\Phi \underleftarrow{i}\,\hat{b}_0$ and $\hat{b}_1\,\underleftarrow{d} \longrightarrow_\Phi \hat{b}_0\,\underrightarrow{s}$	$\hat{b}_1 \triangleq (\underleftarrow{i} \bullet \hat{b}_0 \mathbin{/} \underleftarrow{i}) \mathbin{\&} (\hat{b}_0 \bullet \underrightarrow{s} \mathbin{/} \underleftarrow{d})$
$z\,b_0' \longrightarrow z$ and $s\,b_0' \longrightarrow b_1\,s$	$\underrightarrow{z}\,\hat{b}_0' \longrightarrow_\Phi \underrightarrow{z}$ and $\underrightarrow{s}\,\hat{b}_0' \longrightarrow_\Phi \hat{b}_1\,\underrightarrow{s}$	$\hat{b}_0' \triangleq (\underrightarrow{z} \mathbin{\backslash} \underrightarrow{z}) \mathbin{\&} (\underrightarrow{s} \mathbin{\backslash} \hat{b}_1 \bullet \underrightarrow{s})$

Table 4.5: Deriving an object-oriented choreography of binary counters

4.5 *Binary counters*

Recall from ?? a string rewriting specification (Σ, Θ) of binary counters, *i.e.*, binary representations of natural numbers equipped with increment and decrement operations, where the alphabet Σ and the axioms Θ are:

$$\begin{aligned}
\Sigma &= \{e, b_0, b_1, i, b_0', d, z, s\} \\
\Theta &= (e\,i \longrightarrow e\,b_1)\,, (b_0\,i \longrightarrow b_1)\,,\ (b_1\,i \longrightarrow i\,b_0)\,, \\
&\quad\ (e\,d \longrightarrow z)\,,\quad (b_0\,d \longrightarrow b_0')\,, (b_1\,d \longrightarrow b_0\,s)\,, \\
&\quad\ (z\,b_0' \longrightarrow z)\,,\ (s\,b_0' \longrightarrow b_1\,s)
\end{aligned}$$

In this section, we present two distinct meaningful choreograghies of binary counters: an object-oriented choreography that treats the increment and decrement operations as messages, and a functional choreography that instead treats those operations as processes.

4.5.1 *An object-oriented choreography*

Let θ be the [...] assignment that maps the bits e, b_0, and b_1 to recursively defined processes $\hat{e}$, $\hat{b}_0$, and $\hat{b}_1$; increments i and decrements d to left-directed messages $\underleftarrow{i}$ and $\underleftarrow{d}$; unary constructors z and s to right-directed messages $\underrightarrow{z}$ and $\underrightarrow{s}$; and b_0' to recursively defined process $\hat{b}_0'$.

$e \mapsto \hat{e}$	$i \mapsto \underleftarrow{i}$
$b_0 \mapsto \hat{b}_0$	$d \mapsto \underleftarrow{d}$
$b_1 \mapsto \hat{b}_1$	$z \mapsto \underrightarrow{z}$
$b_0' \mapsto \hat{b}_0'$	$s \mapsto \underrightarrow{s}$

Two axioms mention e in their premises: $e\,i \longrightarrow e\,b_1$ and $e\,d \longrightarrow z$. Under the [...] assignment θ, these axioms induce the rewritings

$$\hat{e}\,\underleftarrow{i} \longrightarrow_\Phi \hat{e}\,\hat{b}_1 \qquad \text{and} \qquad \hat{e}\,\underleftarrow{d} \longrightarrow_\Phi \underrightarrow{z}$$

as constraints on Φ that must be satisfied if (θ, Φ) is to be a meaningful choreography. Solving these for $\hat{e}$, we obtain the definition

$$\hat{e} \triangleq (\hat{e} \bullet \hat{b}_1 \mathbin{/} \underleftarrow{i}) \mathbin{\&} (\underrightarrow{z} \mathbin{/} \underleftarrow{d})\,.$$

Similar reasoning allows us to construct definitions for $\hat{b}_0$, $\hat{b}_1$, and $\hat{b}_0'$ as the solutions of the other constraints induced from the axioms Θ by θ. (See ?? for a sketch.) In full, the signature Φ is

$$\begin{aligned}
\Phi = {} & \big(\hat{e} \triangleq (\hat{e} \bullet \hat{b}_1 \mathbin{/} \underleftarrow{i}) \mathbin{\&} (\underrightarrow{z} \mathbin{/} \underleftarrow{d})\big)\,, \\
& \big(\hat{b}_0 \triangleq ({\uparrow}{\downarrow}\hat{b}_1 \mathbin{/} \underleftarrow{i}) \mathbin{\&} (\underleftarrow{d} \bullet \hat{b}_0' \mathbin{/} \underleftarrow{d})\big)\,, \\
& \big(\hat{b}_1 \triangleq (\underleftarrow{i} \bullet \hat{b}_0 \mathbin{/} \underleftarrow{i}) \mathbin{\&} (\hat{b}_0 \bullet \underrightarrow{s} \mathbin{/} \underleftarrow{d})\big)\,, \\
& \big(\hat{b}_0' \triangleq (\underrightarrow{z} \mathbin{\backslash} \underrightarrow{z}) \mathbin{\&} (\underrightarrow{s} \mathbin{\backslash} \hat{b}_1 \bullet \underrightarrow{s})\big)\,.
\end{aligned}$$

In other words, under the [...] assignment θ, the string rewriting axioms for the binary counter are choreographed to the recursive propositions defined in Φ. It is easy, if tedious, to verify that the formal construction generates the same ordered rewriting signature, Φ: For the above string rewriting specification (Σ, Θ) and [...] assignment θ, the judgment $\theta \vdash \Theta \rightsquigarrow \Phi$ holds.

This choreography might be called *object-oriented* for its similarity to the eponymous[36] programming paradigm. In that paradigm, computation centers around message exchange between stateful objects – data is stored by objects, and that data is manipulated by exchanging messages with the relevant objects. This choreography of the binary counter specification behaves similarly: its data – the bits e, b_0, and b_1 – are represented as processes, and its operations – the increments i and decrements d – are represented as messages.

[36] ?

For example, $\hat{e}$ is the recursively defined process that waits to receive either the increment message $\underleftarrow{i}$ or the decrement message $\underleftarrow{d}$ from its right-hand neighbor. If $\underleftarrow{i}$ is received, then $\hat{e}$ spawns a new process, $\hat{b}_1$, to its right and then continues recursively as $\hat{e}$. Otherwise, if $\underleftarrow{d}$ is received, then $\hat{e}$ sends the message $\underrightarrow{z}$ as a response.

Recall from ?? that [37]

[37] ?

Combining this with ??, we have the immediate ??

Corollary 4.13. *If $w \approx_{\mathrm{I}} n$, then $\theta(w) \Longrightarrow_{\Phi} \theta(w')$ and $w' \approx_{\mathrm{V}} n$, for some w'.*

Is the following result what we want?

Corollary 4.14. *If $w \approx_{\mathrm{I}} n$, then $\theta(w) \Longrightarrow \theta(w')$ and $w' \approx_{\mathrm{V}} n$, for some w'. If $w \approx_{\mathrm{I}} n$ and $\theta(w) \Longrightarrow \Omega'$, then $\Omega' = \theta(w')$ and $w' \approx_{\mathrm{I}} n$, for some w'.*

$$\begin{aligned}
\hat{e} &\triangleq (\hat{e} \bullet \hat{b}_1 \mathbin{/} \underleftarrow{i}) \mathbin{\&} (\hat{z} \mathbin{/} \underleftarrow{d}) \\
\hat{b}_0 &\triangleq ({\uparrow}{\downarrow}\hat{b}_1 \mathbin{/} \underleftarrow{i}) \mathbin{\&} (\underleftarrow{d} \bullet \underleftarrow{b}'_0 \mathbin{/} \underleftarrow{d}) \\
\hat{b}_1 &\triangleq (\underleftarrow{i} \bullet \hat{b}_0 \mathbin{/} \underleftarrow{i}) \mathbin{\&} (\hat{b}_0 \bullet \hat{s} \mathbin{/} \underleftarrow{d}) \\
\hat{z} &\triangleq {\uparrow}{\downarrow}\hat{z} \mathbin{/} \underleftarrow{b}'_0 \\
\hat{s} &\triangleq \hat{b}_1 \bullet \hat{s} \mathbin{/} \underleftarrow{b}'_0
\end{aligned}$$

$$\begin{aligned}
\hat{z} &\triangleq ({\uparrow}{\downarrow}\hat{z} \mathbin{/} \underleftarrow{b}'_0) \mathbin{\&} (\underrightarrow{z} \mathbin{/} \underleftarrow{x}) \\
\hat{s} &\triangleq (\hat{b}_1 \bullet \hat{s} \mathbin{/} \underleftarrow{b}'_0) \mathbin{\&} (\underrightarrow{s} \mathbin{/} \underleftarrow{x})
\end{aligned}$$

Axioms, Θ	*Rewriting constraints on* Φ'	*Solution,* Φ'
$e\,i \longrightarrow e\,b_1$ and $b_0\,i \longrightarrow b_1$ and $b_1\,i \longrightarrow i\,b_0$	$\underset{\rightarrow}{e}\,\hat{\imath} \longrightarrow_{\Phi'} \underset{\rightarrow}{e}\,\underset{\rightarrow}{b}_1$ and $\underset{\rightarrow}{b}_0\,\hat{\imath} \longrightarrow_{\Phi'} \underset{\rightarrow}{b}_1$ and $\underset{\rightarrow}{b}_1\,\hat{\imath} \longrightarrow_{\Phi'} \hat{\imath}\,\underset{\rightarrow}{b}_0$	$\hat{\imath} \triangleq (\underset{\rightarrow}{e} \setminus \underset{\rightarrow}{e} \bullet \underset{\rightarrow}{b}_1) \mathbin{\&} (\underset{\rightarrow}{b}_0 \setminus \underset{\rightarrow}{b}_1) \mathbin{\&} (\underset{\rightarrow}{b}$
$e\,d \longrightarrow z$ and $b_0\,d \longrightarrow d\,b_0'$ and $b_1\,d \longrightarrow b_0\,s$	$\underset{\rightarrow}{e}\,\hat{d} \longrightarrow_{\Phi'} \underset{\rightarrow}{z}$ and $\underset{\rightarrow}{b}_0\,\hat{d} \longrightarrow_{\Phi'} \hat{d}\,\hat{b}_0'$ and $\underset{\rightarrow}{b}_1\,\hat{d} \longrightarrow_{\Phi'} \underset{\rightarrow}{b}_0\,\underset{\rightarrow}{s}$	$\hat{d} \triangleq (\underset{\rightarrow}{e} \setminus \underset{\rightarrow}{z}) \mathbin{\&} (\underset{\rightarrow}{b}_0 \setminus \hat{d} \bullet \hat{b}_0') \mathbin{\&} (\underset{\rightarrow}{b}$
$z\,b_0' \longrightarrow z$ and $s\,b_0' \longrightarrow b_1\,s$	$\underset{\rightarrow}{z}\,\hat{b}_0' \longrightarrow_{\Phi'} \underset{\rightarrow}{z}$ and $\underset{\rightarrow}{s}\,\hat{b}_0' \longrightarrow_{\Phi'} \hat{b}_1\,\underset{\rightarrow}{s}$	$\hat{b}_0' \triangleq (\underset{\rightarrow}{z} \setminus \underset{\rightarrow}{z}) \mathbin{\&} (\underset{\rightarrow}{s} \setminus \underset{\rightarrow}{b}_1 \bullet \underset{\rightarrow}{s})$

Table 4.6: Deriving a functional choreography of binary counters

4.5.2 *A functional choreography*

The object-oriented choreography is not the only choreography possible for the binary counter, however.

Let θ' be the [...] assignment that is (roughly) dual to θ – that is, θ' maps the bits e, b_0, and b_1 to right-directed messages $\underset{\rightarrow}{e}$, $\underset{\rightarrow}{b}_0$, and $\underset{\rightarrow}{b}_1$; increments i and decrements d to recursively defined processes $\hat{\imath}$ and $\hat{d}$; unary constructors z and s to right-directed messages $\underset{\rightarrow}{z}$ and $\underset{\rightarrow}{s}$; and b_0' to recursively defined process $\hat{b}_0'$.

$e \mapsto \underset{\rightarrow}{e}$	$i \mapsto \hat{\imath}$
$b_0 \mapsto \underset{\rightarrow}{b}_0$	$d \mapsto \hat{d}$
$b_1 \mapsto \underset{\rightarrow}{b}_1$	$z \mapsto \underset{\rightarrow}{z}$
$b_0' \mapsto \hat{b}_0'$	$s \mapsto \underset{\rightarrow}{s}$

Once again, we can construct a choreography from the string rewriting axioms by solving constraints in the form of rewritings. Three axioms mention i in their premises: $e\,i \longrightarrow e\,b_1$, $b_0\,i \longrightarrow b_1$, and $b_1\,i \longrightarrow i\,b_0$. Under the [...] assignment θ', these axioms induce the rewritings

$$\underset{\rightarrow}{e}\,\hat{\imath} \longrightarrow_{\Phi'} \underset{\rightarrow}{e}\,\underset{\rightarrow}{b}_1 \qquad \text{and} \qquad \underset{\rightarrow}{b}_0\,\hat{\imath} \longrightarrow_{\Phi'} \underset{\rightarrow}{b}_1 \qquad \text{and} \qquad \underset{\rightarrow}{b}_1\,\hat{\imath} \longrightarrow_{\Phi'} \hat{\imath}\,\underset{\rightarrow}{b}_0$$

as constraints on Φ' [that must be satisfied if (θ', Φ') is to be a meaningful choreography]. Solving these contraints for $\hat{\imath}$, we obtain the definition

$$\hat{\imath} \triangleq (\underset{\rightarrow}{e} \setminus \underset{\rightarrow}{e} \bullet \underset{\rightarrow}{b}_1) \mathbin{\&} (\underset{\rightarrow}{b}_0 \setminus \underset{\rightarrow}{b}_1) \mathbin{\&} (\underset{\rightarrow}{b}_1 \setminus \hat{\imath} \bullet \underset{\rightarrow}{b}_0)\,.$$

Upon solving the remaining constraints for the other recursively defined propositions, $\hat{d}$ and $\hat{b}_0'$,[38] we arrive at the complete signature

[38] See ?? for a sketch.

$$\begin{aligned}\Phi' = {} & \big(\hat{\imath} \triangleq (\underset{\rightarrow}{e} \setminus \underset{\rightarrow}{e} \bullet \underset{\rightarrow}{b}_1) \mathbin{\&} (\underset{\rightarrow}{b}_0 \setminus \underset{\rightarrow}{b}_1) \mathbin{\&} (\underset{\rightarrow}{b}_1 \setminus \hat{\imath} \bullet \underset{\rightarrow}{b}_0)\big)\,,\\ & \big(\hat{d} \triangleq (\underset{\rightarrow}{e} \setminus \underset{\rightarrow}{z}) \mathbin{\&} (\underset{\rightarrow}{b}_0 \setminus \hat{d} \bullet \hat{b}_0') \mathbin{\&} (\underset{\rightarrow}{b}_1 \setminus \underset{\rightarrow}{b}_0 \bullet \underset{\rightarrow}{s})\big)\,,\\ & \big(\hat{b}_0' \triangleq (\underset{\rightarrow}{z} \setminus \underset{\rightarrow}{z}) \mathbin{\&} (\underset{\rightarrow}{s} \setminus \hat{b}_1 \bullet \underset{\rightarrow}{s})\big)\,.\end{aligned}$$

Again, it is easy to verify that this signature is exactly what is contructed by the formal description of the choreographing algorithm: For the above string rewriting specification (Σ, Θ) and [...] assignment θ', the judgment $\theta' \vdash \Theta \rightsquigarrow \Phi'$ holds.

In contrast with the object-oriented choreography, this choreography treats its data – the bits e, b_0, and b_1 – as messages that are manipulated by processes that represent the operations – increments i and decrements d. In this way, this choreography, (θ', Φ'), might be called *functional* for its similarity to functional programming.

These two (roughly) dual choreographies hint at a fundamental duality between the object-oriented and functional programming paradigms. A general duality theorem does not exist: even if the constraint $\underrightarrow{a}\,\hat{b}\,\underleftarrow{c} \longrightarrow_{\Phi} \theta(w')$ is satisfiable, the dual[39] $\hat{a}\,\underleftarrow{b}\,\hat{c} \longrightarrow \theta^{\perp}(w')$ nor $\hat{a}\,\underrightarrow{b}\,\hat{c} \longrightarrow \theta^{\perp}(w')$ are [...].

[39] ?

$$\begin{aligned}\Phi' = {} & \big(\hat{\imath} \triangleq (\underrightarrow{e} \setminus \underrightarrow{e} \bullet \underrightarrow{b}_1) \mathbin{\&} (\underrightarrow{b}_0 \setminus \underrightarrow{b}_1) \mathbin{\&} (\underrightarrow{b}_1 \setminus \hat{\imath} \bullet \underrightarrow{b}_0)\big)\,, \\ & \big(\hat{d} \triangleq (\underrightarrow{e} \setminus {\uparrow}{\downarrow}\hat{z}) \mathbin{\&} (\underrightarrow{b}_0 \setminus \hat{d} \bullet \underleftarrow{b}'_0) \mathbin{\&} (\underrightarrow{b}_1 \setminus \underrightarrow{b}_0 \bullet \hat{s})\big)\,, \\ & \big(\hat{z} \triangleq {\uparrow}{\downarrow}\hat{z} \,/\, \underleftarrow{b}'_0\big)\,, \\ & \big(\hat{s} \triangleq \underrightarrow{b}_1 \bullet \hat{s} \,/\, \underleftarrow{b}'_0\big)\end{aligned}$$

Part III

Concurrency as proof reduction

Part IV

Comparing the two approaches