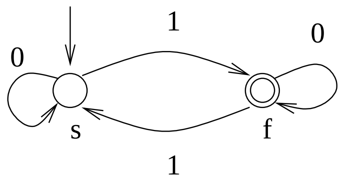
2.5 An Example: Finite Automata

One of the simplest example of computation with state is provided by finite automata. In this section we discuss possible ways to model non-deterministic finite automata in linear logic.

We represent each state of the automaton by a predicate on strings. If the automaton can go from state p to state q while reading string x then we have $p(x) \multimap q(\epsilon)$, where ϵ represents the empty string. More generally, we should have $p(x \cdot y) \multimap q(y)$ for any y, where $x \cdot y$ represent string concatenation. It is convenient to assume we have a single distinguished start state s and final state f. If the automaton has more than one accepting state, we can transform it by adding a new, sole accepting state f and add ϵ-transitions from all the previously accepting states to f.

Consider a simple automaton to accept binary strings with odd parity.



We can implement this with the following propositions, whose use is unrestricted.

$$\begin{array}{rcl} s^0 & : & \forall x.\ s(0 \cdot x) \multimap s(x) \\ s^1 & : & \forall x.\ s(1 \cdot x) \multimap f(x) \\ f^0 & : & \forall x.\ f(0 \cdot x) \multimap f(x) \\ f^1 & : & \forall x.\ f(1 \cdot x) \multimap s(x) \end{array}$$

Even though we do not have the tools to prove this at the moment, we should keep in mind what we would like to achieve. In this example, we can recognize strings with odd parity by adding the unrestricted assumption

$$\forall x.\ (s(x) \multimap f(\epsilon)) \multimap \mathrm{odd}(x).$$

Now we can prove $\mathrm{odd}(x)$ if and only if x is a binary string with odd parity. More generally, our encoding should satisfy the following *adequacy theorem.*

Adequacy for Sequential Encoding of Automata.

> Given a non-deterministic finite automaton M and its encoding Γ. Then for all states p and q and strings x, $p \xrightarrow{x} q$ if and only if $\Gamma; \cdot \vdash \forall y.\ p(x \cdot y) \multimap q(y)$. In particular, if M has initial state s and final state f, then M accepts x if and only if $\Gamma; \cdot \vdash s(x) \multimap f(\epsilon)$.

The direct representation given above maps every possible single-step transition $p \xrightarrow{x} q$ to the proposition $\forall y.\ p(x \cdot y) \multimap q(y)$. Typically, x would be a character c or the empty string ϵ, but we can also translate automata that can accept multiple characters in one step.

Non-deterministic finite automata accept exactly the regular languages as defined by regular expressions. In addition, regular languages are closed under some other operations such as intersection or complement. We now consider how to translate a regular expression into linear logic following a similar strategy as for automata above. In this case we give the construction of the linear logic propositions inductively, based on the shape of the regular expression. We write $\mathcal{L}(r)$ for the language of strings defined by a regular expression.

Adequacy for Sequential Encoding of Regular Expressions.

> Given a regular expression r and its encoding Γ with distinguished predicates s (*start*) and f (*final*). Then $x \in \mathcal{L}(r)$ if and only if $\Gamma; \cdot \vdash \forall y.\ s(x \cdot y) \multimap f(y)$.

For each form of regular expression we now go though the corresponding construction of Γ. We write $\Gamma(s, f)$ to identify the distinguished start and final predicate.

Case: $r = a$ for a character a where $\mathcal{L}(a) = \{a\}$. Then

$$\Gamma(s, f) = \forall y.\ s(a \cdot y) \multimap f(y)$$

Case: $r = r_1 \cdot r_2$ where $\mathcal{L}(r_1 \cdot r_2) = \{x_1 \cdot x_2 \mid x_1 \in \mathcal{L}(r_1) \text{ and } x_2 \in \mathcal{L}(r_2)\}$. Let $\Gamma_1(s_1, f_1)$ and $\Gamma_2(s_2, f_2)$ be the translations of r_1 and r_2 respectively. We construct

$$\begin{array}{rcl} \Gamma(s, f) & = & \forall x.\ s(x) \multimap s_1(x), \\ & & \Gamma_1(s_1, f_1), \\ & & \forall z.\ f_1(z) \multimap s_2(z), \\ & & \Gamma_2(s_2, f_2), \\ & & \forall y.\ f_2(y) \multimap f(y) \end{array}$$

Case: $r = \mathbf{1}$ where $\mathcal{L}(\mathbf{1}) = \{\epsilon\}$. Then

$$\Gamma(s, f) = \forall y.\ s(y) \multimap f(y)$$

Case: $r = r_1 + r_2$ where $\mathcal{L}(r_1 + r_2) = \mathcal{L}(r_1) \cup \mathcal{L}(r_2)$. Let $\Gamma_1(s_1, f_1)$ and $\Gamma_2(s_2, f_2)$ be the translations of r_1 and r_2 respectively. We construct

$$\begin{array}{rcl} \Gamma(s, f) & = & \forall x.\ s(x) \multimap (s_1(x) \& s_2(x)), \\ & & \Gamma_1(s_1, f_1), \Gamma_2(s_2, f_2), \\ & & \forall y.\ f_1(y) \multimap f(y), \\ & & \forall y.\ f_2(y) \multimap f(y) \end{array}$$

Alternatively, we could replace the first rule with the following two:

$$\begin{array}{l} \forall x.\ s(x) \multimap s_1(x), \\ \forall x.\ s(x) \multimap s_2(x) \end{array}$$

The first formulation may have the slight advantage that every state p except the final state has exactly one transition $\forall x.\ p(t) \multimap A$ for some string t and proposition A. We can also reaplce the last two propositions by

$$\forall y.\ (f_1(y) \oplus f_2(y)) \multimap f(y)$$

This may be preferable if we would like to maintain instead the invariant that every final state has one transition into it. These formulations are equivalent, since

$$\begin{array}{rcl} A \multimap (B \& C) & \dashv\vdash & (A \multimap B) \& (A \multimap C) \\ (A \oplus B) \multimap C & \dashv\vdash & (A \multimap C) \& (B \multimap C) \end{array}$$

and the fact that an *unrestricted* assumption $A \& B$ *valid* is equivalent to two *unrestricted* assumptions A *valid*, B *valid*.

Case: $r = \mathbf{0}$ where $\mathcal{L}(\mathbf{0}) = \{\,\}$.

$$\Gamma(s, f) \quad = \quad \forall x.\ s(x) \multimap \top$$

with no rule for f. In analogy with the previous case, we could also simply not generate any propositions for s and f, or generate the proposition

$$\forall y.\ \mathbf{0} \multimap f(y)$$

for f.

Case: $r = r_1^*$ where $\mathcal{L}(r_1^*) = \{x_1 \cdots x_n \mid x_i \in r_1 \text{ for } 1 \leq i \leq n \text{ and } n \geq 0\}$. Let $\Gamma_1(s_1, f_1)$ be the translation of r_1. Then we construct

$$\begin{array}{rcl} \Gamma(s, f) & = & \forall x.\ s(x) \multimap (f(x) \& s_1(x)), \\ & & \Gamma_1(s_1, f_1), \\ & & \forall y.\ f_1(y) \multimap s(y) \end{array}$$

Alternatively, the first proposition can be broken up into two as in the case for $r_1 + r_2$.

Regular languages are closed under intersection ($\cap$), the full language ($\mathbf{T}$) and complementation. At least the first two are relatively easy to implement.

Case: $r = r_1 \cap r_2$ where $\mathcal{L}(r_1 \cap r_2) = \mathcal{L}(r_1) \cap \mathcal{L}(r_2)$. Let $\Gamma_1(s_1, f_1)$ and $\Gamma_2(s_2, f_2)$ be the translation of r_1 and r_2. The difficult part of intersection is that r_1 and r_2 must consume *the same initial segment* of the input. This can be achieved using simultaneous conjunction.

$$\begin{array}{rcl} \Gamma(s, f) & = & \forall x.\ s(x) \multimap (s_1(x) \otimes s_2(x)), \\ & & \Gamma_1(s_1, f_1), \Gamma_2(s_2, f_2), \\ & & \forall y.\ (f_1(y) \otimes f_2(y)) \multimap f(y) \end{array}$$

Note how we exploit multiple linear hypotheses and force the synchronization of their accepting states on y.

Case: $r = \mathbf{T}$ where $\mathcal{L}(\mathbf{T}) = \Sigma^*$, the set of all strings over the alphabet Σ. $\mathbf{T}$ accepts *any* initial segment of the input string.

$$\Gamma(s, f) \quad = \quad \forall x.\, \forall y.\ s(x \cdot y) \multimap f(y)$$

Strictly speaking, this proposition could present some problems in that solving an equation such as $a \cdot b \cdot c = x \cdot y$ has multiple solutions if x and y both stand for arbitrary strings. If we would like to avoid the assumption that the logic understands the equational theory of strings, we could decompose this clause into

$$\begin{array}{rcl}\Gamma(s, f) & = & \forall x.\ s(x) \multimap f(x), \\ & & \forall a.\, \forall y.\ s(a \cdot y) \multimap s(y)\end{array}$$

where a ranges only over characters.

Complement appears to be more complicated, and we presently have no direct and elegant solution. Note that the encoding of $\mathbf{T}$ we take some algebraic properties of string concatenation for granted without axiomatizing them explicitly.

In the representation, non-determinism arising from $r_1 + r_2$ is represented by an internal choice in

$$\forall x.\ s(x) \multimap (s_1(x) \& s_2(x)).$$

That is, in the course of the proof itself we have to guess (internal choice) whether $s_1(x)$ or $s_2(x)$ will lead to success.

An alternative model of computation would try all successor states essentially concurrently. The corresponds to the idea for transforming non-deterministic automata into deterministic ones: we keep track of all the possible states we might be in instead of guessing which transition to make. While in the encoding above, we have essentially one linear hypothesis (the current state, applied to the remaining input string), here we will have multiple ones. The adequacy for this kind of representation is more difficult to formulate precisely, because the additional threads of computation.

Adequacy of Concurrent Encoding of Automata.

Given a non-deterministic finite automaton M and its concurrent encoding Γ. Then for all states p and q and strings x, $p \xrightarrow{x} q$ if and only if $\Gamma; \cdot \vdash \forall y.\ p(x \cdot y) \multimap (q(y) \otimes \top)$. In particular, if M has initial state s and final state f, then M accepts x if and only if $\Gamma; \cdot \vdash s(x) \multimap (f(\epsilon) \otimes \top)$.

While this expresses correctness, it does not explicitly address the concurrency aspects. For example, even our prior encoding would satisfy this requirement even though it does not encode any concurrency. We omit the hypothesis labels in this encoding.

Cases: $r = a$ or $r = r_1 \cdot r_2$ or $r = \mathbf{1}$. As before.

Case: $r = r_1 + r_2$. Let $\Gamma_1(s_1, f_1)$ and $\Gamma_2(s_2, f_2)$ be the translations of r_1 and r_2 respectively. We construct

$$\begin{array}{rcl} \Gamma(s,f) & = & \forall x.\, s(x) \multimap (s_1(x) \otimes s_2(x)), \\ & & \Gamma_1(s_1, f_1), \Gamma_2(s_2, f_2), \\ & & \forall y.\, f_1(y) \multimap f(y), \\ & & \forall y.\, f_2(y) \multimap f(y) \end{array}$$

Now there is no alternative formulation of the first rule.

Case: $r = \mathbf{0}$ where $\mathcal{L}(\mathbf{0}) = \{\,\}$.

$$\Gamma(s,f) \quad = \quad \forall x.\, s(x) \multimap \mathbf{1}$$

with no rule for f.

Case: $r = r_1^*$ where $\mathcal{L}(r_1^*) = \{x_1 \cdots x_n \mid x_i \in \mathcal{L}(r_1) \text{ for } 1 \leq i \leq n \text{ and } n \geq 0\}$. Let $\Gamma_1(s_1, f_1)$ be the translation of r_1. Then we construct

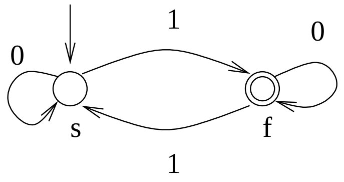
$$\begin{array}{rcl} \Gamma(s,f) & = & \forall x.\, s(x) \multimap (f(x) \otimes s_1(x)), \\ & & \Gamma_1(s_1, f_1), \\ & & \forall y.\, f_1(y) \multimap s(y) \end{array}$$

Cases: $r = r_1 \cap r_2$ and $r_1 = \mathbf{T}$. As before.

This form of concurrent encoding requires some consideration of scheduling the processes represented by linear hypotheses. For example, if we have a regular expression $\epsilon^* \cdot a$ we have to be careful not to schedule the process corresponding to ϵ^* indefinitely without scheduling the process for a, or we may never accept the string a.

These issues foreshadow similar considerations for more complex concurrent systems in linear logic later on. Note that computation in this setting corresponds to a kind of forward-chaining proof search. Other models of computation are also possible and often appropriate. In particular, we may describe computation via backward-chaining search, or by proof reduction. We close this section by showing a backward-chaining implementation of finite automata and regular expressions.

Recall the simple automaton to accept binary strings with odd parity.



In the coding we now simply *reverse* all the arrows.

$$\begin{array}{rcl} s^0 & : & \forall x.\ s(0 \cdot x) \multimapinv s(x) \\ s^1 & : & \forall x.\ s(1 \cdot x) \multimapinv f(x) \\ f^0 & : & \forall x.\ f(0 \cdot x) \multimapinv f(x) \\ f^1 & : & \forall x.\ f(1 \cdot x) \multimapinv s(x) \end{array}$$

Then, the automaton accepts x if and only if we can prove

$$f(\epsilon) \multimap s(x),$$

again reversing the arrow from the previous statement where we prove $s(x) \multimap f(\epsilon)$ instead.

Adequacy for Backward-Chaining Encoding of Automata.

> Given a non-deterministic finite automaton M and its encoding Γ. Then for all states p and q and strings x, $p \xrightarrow{x} q$ if and only if $\Gamma; \cdot \vdash \forall y.\ q(y) \multimap p(x \cdot y)$. In particular, if M has initial state s and final state f, then M accepts x if and only if $\Gamma; \cdot \vdash f(\epsilon) \multimap s(x)$.

For completness, we now give the backward-chaining encoding of regular expressions.

Case: $r = c$ for a character c. Then

$$\Gamma(s, f) = \forall y.\ s(c \cdot y) \multimapinv f(y)$$

Case: $r = r_1 \cdot r_2$. Let $\Gamma_1(s_1, f_1)$ and $\Gamma_2(s_2, f_2)$ be the translations of r_1 and r_2 respectively. We construct

$$\begin{array}{rcl} \Gamma(s, f) & = & \forall x.\ s(x) \multimapinv s_1(x), \\ & & \Gamma_1(s_1, f_1), \\ & & \forall z.\ f_1(z) \multimapinv s_2(z), \\ & & \Gamma_2(s_2, f_2), \\ & & \forall y.\ f_2(y) \multimapinv f(y) \end{array}$$

Case: $r = \mathbf{1}$ where $\mathcal{L}(\mathbf{1}) = \{\epsilon\}$. Then

$$\Gamma(s, f) = \forall y.\ s(y) \multimapinv f(y)$$

Case: $r = r_1 + r_2$. Let $\Gamma_1(s_1, f_1)$ and $\Gamma_2(s_2, f_2)$ be the translations of r_1 and r_2 respectively. We construct

$$\begin{array}{rcl} \Gamma(s, f) & = & \forall x.\ s(x) \multimapinv (s_1(x) \oplus s_2(x)), \\ & & \Gamma_1(s_1, f_1), \Gamma_2(s_2, f_2), \\ & & \forall y.\ f_1(y) \multimapinv f(y), \\ & & \forall y.\ f_2(y) \multimapinv f(y) \end{array}$$

Interestingly, the internal choice $s_1(x) \& s_2(x)$ is turned into an external choice $s_1(x) \oplus s_2(x)$ when we turn a forward chaining to a backward chaining implementation. Again, there are some alternatives. For example, the last two propositions can be combined into

$$\forall y.\ (f_1(y) \& f_2(y)) \circ\!- f(y)$$

Case: $r = \mathbf{0}$ where $\mathcal{L}(\mathbf{0}) = \{\,\}$.

$$\Gamma(s, f) \quad = \quad \forall x.\ s(x) \circ\!- \mathbf{0}$$

with no rule for f. Again, alternatives are possible.

Case: $r = r_1^*$. Let $\Gamma_1(s_1, f_1)$ be the translation of r_1. Then we construct

$$\begin{array}{rcl} \Gamma(s, f) & = & \forall x.\ s(x) \circ\!- (f(x) \oplus s_1(x)), \\ & & \Gamma_1(s_1, f_1), \\ & & \forall y.\ f_1(y) \circ\!- s(y) \end{array}$$

Case: $r = r_1 \cap r_2$. Let $\Gamma_1(s_1, f_1)$ and $\Gamma_2(s_2, f_2)$ be the translation of r_1 and r_2. This case appears to be quite tricky, because of the lack of any natural concurrency in the backward-chaining model.[1]

Case: $r = \mathbf{T}$. Then

$$\Gamma(s, f) \quad = \quad \forall x.\ \forall y.\ s(x \cdot y) \circ\!- f(y)$$

We now return to our first encoding of regular expressions. How can we prove the adequacy of the representation? Recall the statement of the adequacy theorem.

Adequacy for Encoding of Regular Expressions.

Given a regular expression r and its encoding Γ with distinguished predicates s (*start*) and f (*final*). Then $x \in \mathcal{L}(r)$ if and only if $\Gamma; \cdot \vdash \forall y.\ s(x \cdot y) \multimap f(y)$.

We first prove that if $x \in \mathcal{L}(r)$, then $\Gamma(s, f); \cdot \vdash \forall y.\ s(x \cdot y) \multimap f(y)$. In all cases we reduce this to proving

$$\Gamma(s, f); s(x \cdot y) \vdash f(y)$$

for a new parameter y. The adequacy follows from this in two steps by $\multimap$I and $\forall$I. The proof is now by induction on the structure of r. We restate the translations, this time giving explicit labels to assumptions so we can refer to them in the proof.

[1]Suggestions welcome!

Case: $r = a$ for a character a where $\mathcal{L}(a) = \{a\}$. Then

$$\Gamma(s, f) = v_s : \forall y.\ s(a \cdot y) \multimap f(y)$$

So we have to show

$$v_s : \forall y.\ s(a \cdot y) \multimap f(y); s(a \cdot y) \vdash f(y)$$

which follows in three steps.

$v_s : \forall y.\ s(a \cdot y) \multimap f(y); s(a \cdot y) \vdash s(a \cdot y)$	Linear hypothesis
$v_s : \forall y.\ s(a \cdot y) \multimap f(y); \cdot \vdash \forall y.\ s(a \cdot y) \multimap f(y)$	Unrestricted hypothesis
$v_s : \forall y.\ s(a \cdot y) \multimap f(y); \cdot \vdash s(a \cdot y) \multimap f(y)$	By rule $\forall$E
$v_s : \forall y.\ s(a \cdot y) \multimap f(y); s(a \cdot y) \vdash f(y)$	By rule $\multimap$E

Case: $r = r_1 \cdot r_2$ where $\mathcal{L}(r_1 \cdot r_2) = \{x_1 \cdot x_2 \mid x_1 \in \mathcal{L}(r_1) \text{ and } x_2 \in \mathcal{L}(r_2)\}$. Let $\Gamma_1(s_1, f_1)$ and $\Gamma_2(s_2, f_2)$ be the translations of r_1 and r_2 respectively. We construct

$$\begin{array}{rcl} \Gamma(s, f) & = & v_s : \forall x.\ s(x) \multimap s_1(x), \\ & & \Gamma_1(s_1, f_1), \\ & & v_{f_1} : \forall z.\ f_1(z) \multimap s_2(z), \\ & & \Gamma_2(s_2, f_2), \\ & & v_{f_2} : \forall y.\ f_2(y) \multimap f(y) \end{array}$$

We have to show

$$\Gamma(s, f); s(x_1 \cdot x_2 \cdot y) \vdash f(y)$$

for a new parameter y.

$\Gamma(s, f); s(x_1 \cdot x_2 \cdot y) \vdash s(x_1 \cdot x_2 \cdot y)$	Linear hypothesis
$\Gamma(s, f); s(x_1 \cdot x_2 \cdot y) \vdash s_1(x_1 \cdot x_2 \cdot y)$	From v_s by $\forall$E and $\multimap$E
$\Gamma(s, f); s(x_1 \cdot x_2 \cdot y) \vdash f_1(x_2 \cdot y)$	By i.h. on r_1, weakening and subst.
$\Gamma(s, f); s(x_1 \cdot x_2 \cdot y) \vdash s_2(x_2 \cdot y)$	From v_{f_1} by $\forall$E and $\multimap$E
$\Gamma(s, f); s(x_1 \cdot x_2 \cdot y) \vdash f_2(y)$	By i.h. on r_2, weakening and subst.
$\Gamma(s, f); s(x_1 \cdot x_2 \cdot y) \vdash f(y)$	from v_{f_2} by $\forall$E and $\multimap$E

Case: $r = \mathbf{1}$ where $\mathcal{L}(\mathbf{1}) = \{\epsilon\}$. Then

$$\Gamma(s, f) = v_s : \forall y.\ s(y) \multimap f(y)$$

This case is trivial, since $\epsilon \cdot y = y$.

$\Gamma(s, f); s(\epsilon \cdot y) \vdash s(\epsilon \cdot y)$	Linear hypothesis
$\Gamma(s, f); s(\epsilon \cdot y) \vdash f(y)$	From v_s by $\forall$E, $\multimap$E since $\epsilon \cdot y = y$

Case: $r = r_1 + r_2$ where $\mathcal{L}(r_1 + r_2) = \mathcal{L}(r_1) \cup \mathcal{L}(r_2)$. Let $\Gamma_1(s_1, f_1)$ and $\Gamma_2(s_2, f_2)$ be the translations of r_1 and r_2 respectively. We construct

$$\begin{array}{rcl} \Gamma(s, f) & = & v_s : \forall x.\ s(x) \multimap (s_1(x) \& s_2(x)), \\ & & \Gamma_1(s_1, f_1), \Gamma_2(s_2, f_2), \\ & & v_{f_1} : \forall y.\ f_1(y) \multimap f(y), \\ & & v_{f_2} : \forall y.\ f_2(y) \multimap f(y) \end{array}$$

Let $x \in \mathcal{L}(r)$. Then there are two subcases. We show the case where $x \in \mathcal{L}(r_1)$; the other subcase is symmetric.

$$\begin{array}{ll}
\Gamma(s,f); s(x \cdot y) \vdash s(x \cdot y) & \text{Linear hypothesis} \\
\Gamma(s,f); s(x \cdot y) \vdash s_1(x \cdot y) \& s_2(x \cdot y) & \text{From } v_s \text{ by } \forall\text{E}, \multimap\text{E} \\
\Gamma(s,f); s(x \cdot y) \vdash s_1(x \cdot y) & \text{By rule } \&\text{E}_\text{L} \\
\Gamma(s,f); s(x \cdot y) \vdash f_1(y) & \text{By i.h. on } r_1\text{, weakening and subst.} \\
\Gamma(s,f); s(x \cdot y) \vdash f(y) & \text{From } v_{f_1} \text{ by } \forall\text{E}, \multimap\text{E}
\end{array}$$

Case: $r = \mathbf{0}$ where $\mathcal{L}(\mathbf{0}) = \{\,\}$.

$$\Gamma(s,f) \quad = \quad v_s : \forall x.\ s(x) \multimap \top$$

with no rule for f. Then the conclusion follows trivially since there is no $x \in \mathcal{L}(\mathbf{0})$.

Case: $r = r_1^*$ where $\mathcal{L}(r_1^*) = \{x_1 \cdots x_n \mid x_i \in \mathcal{L}(r_1) \text{ for } 1 \leq i \leq n \text{ and } n \geq 0\}$.

$$\begin{array}{rcl}
\Gamma(s,f) & = & v_s : \forall x.\ s(x) \multimap (f(x) \& s_1(x)), \\
& & \Gamma_1(s_1, f_1), \\
& & v_{f_1} : \forall y.\ f_1(y) \multimap s(y)
\end{array}$$

Assume $x = x_1 \cdots x_n \in \mathcal{L}(r_1^*)$ where each $x_i \in r_1$ for $1 \leq i \leq n$. We conduct an auxiliary induction on n.

Subcase: $n = 0$. Then, by definition, $x_1 \cdots x_n = \epsilon$.

$$\begin{array}{ll}
\Gamma(s,f); s(\epsilon \cdot y) \vdash s(\epsilon \cdot y) & \text{Linear hypothesis} \\
\Gamma(s,f); s(\epsilon \cdot y) \vdash f(y) \& s_1(x) & \text{From } v_s \text{ by } \forall\text{E}, \multimap\text{E} \\
\Gamma(s,f); s(\epsilon \cdot y) \vdash f(y) & \text{By rule } \&\text{E}_\text{L}
\end{array}$$

Subcase: $n > 0$. Then

$$\begin{array}{ll}
\Gamma(s,f); s(x_1 \cdots x_n \cdot y) \vdash s(x_1 \cdots x_n \cdot y) & \text{Linear hypothesis} \\
\Gamma(s,f); s(x_1 \cdots x_n \cdot y) \vdash f(x_1 \cdots x_n \cdot y) \& s_1(x_1 \cdots x_n \cdot y) & \text{From } v_s \\
\Gamma(s,f), s(x_1 \cdots x_n \cdot y) \vdash s_1(x_1 \cdots x_n \cdot y) & \text{By rule } \&\text{E}_\text{R} \\
\Gamma(s,f), s(x_1 \cdots x_n \cdot y) \vdash f_1(x_2 \cdots x_n \cdot y) & \text{By i.h. since } x_1 \in \mathcal{L}(r_1) \\
\Gamma(s,f), s(x_1 \cdots x_n \cdot y) \vdash s(x_2 \cdots x_n \cdot y) & \text{From } v_{f_1} \\
\Gamma(s,f), s(x_1 \cdots x_n \cdot y) \vdash f(y) & \text{From i.h. on } n-1
\end{array}$$

We now also show the correctness for the encoding of intersection and all strings.

Case: $r = r_1 \cap r_2$ where $\mathcal{L}(r_1 \cap r_2) = \mathcal{L}(r_1) \cap \mathcal{L}(r_2)$. Let $\Gamma_1(s_1, f_1)$ and $\Gamma_2(s_2, f_2)$ be the translation of r_1 and r_2.

$$\begin{array}{rcl}
\Gamma(s,f) & = & v_s : \forall x.\ s(x) \multimap (s_1(x) \otimes s_2(x)), \\
& & \Gamma_1(s_1, f_1), \Gamma_2(s_2, f_2), \\
& & v_{f_{12}} : \forall y.\ (f_1(y) \otimes f_2(y)) \multimap f(y)
\end{array}$$

Assume $x \in \mathcal{L}(r_1) \cap \mathcal{L}(r_2)$. Then

$\Gamma(s,f); s(x \cdot y) \vdash s(x \cdot y)$ Linear hypothesis
$\Gamma(s,f); s(x \cdot y) \vdash s_1(x \cdot y) \otimes s_2(x \cdot y)$ From v_s
$\Gamma(s,f); s_1(x \cdot y) \vdash f_1(y)$ From i.h. since $x \in \mathcal{L}(r_1)$
$\Gamma(s,f); s_2(x \cdot y) \vdash f_2(y)$ From i.h. since $x \in \mathcal{L}(r_2)$
$\Gamma(s,f); s_1(x \cdot y), s_2(x \cdot y) \vdash f_1(y) \otimes f_2(y)$ By $\otimes$I
$\Gamma(s,f); s(x \cdot y) \vdash f_1(y) \otimes f_2(y)$ By $\otimes$E
$\Gamma(s,f); s(x \cdot y) \vdash f(y)$ From $v_{f_{12}}$

Case: $r = \mathbf{T}$ where $\mathcal{L}(\mathbf{T}) = \Sigma^*$, the set of all strings over the alphabet Σ.

$$\Gamma(s,f) \quad = \quad v_s : \forall x.\, \forall y.\, s(x \cdot y) \multimap f(y)$$

Let x by an arbitrary string. Then

$\Gamma(s,f); s(x \cdot y) \vdash s(x \cdot y)$ Linear hypothesis
$\Gamma(s,f); s(x \cdot y) \vdash f(y)$ From v_s

The other direction is much harder to prove. Assume we have a regular expression r, its encoding $\Gamma(s,f)$, and a deduction of $\Gamma; \cdot \forall y.\, s(x \cdot y) \multimap f(y)$. We need to show that $x \in \mathcal{L}(r)$. In order to do this, we need to analyse the structure of the given deduction. In some sense, we are showing that the deduction we gave in the other direction above are "inevitable". To illustrate the difficulty, consider, for example, the $\multimap$ rule.

$$\frac{\Gamma; \Delta_1 \vdash A \multimap B \qquad \Gamma; \Delta_2 \vdash A}{\Gamma; (\Delta_1, \Delta_2) \vdash B} \multimap\mathrm{E}$$

If we are trying to prove B from Γ and Δ, two problems arise. First, we have to decide how to split Δ into Δ_1 and Δ_2. More importantly, however, how do we choose A?

If we look back at the development of our logic, we introduced this rule to answer the question "*How do we use the knowledge that $A \multimap B$ true*". Above, however, we think about how to prove B. It is this mismatch which makes rules like $\multimap$E intractable.

In the next section we will introduce a restriction on the application of the inference rules whereby introduction rules are only applied bottom-up while elimination rules are applied only top-down. With this restriction it will be possible to show to prove the more difficult direction of adequacy for the encoding of regular expressions.

2.6 Normal Deductions

The judgmental approach to understanding propositions and truth defines the meaning of the connectives by giving the introduction and elimination rules. We claim, for example, that $A \& B$ *true* if A *true* and B *true*. We see this as a defining property for alternative conjunction (once the hypotheses have been added to the rule). But does our final deductive system validate this interpretation of alternative conjunction? For example, it could be the case $C \multimap (A \& B)$ *true* and C *true* but there is no direct way of deriving A *true* and B *true* without the detour through C.

Local soundness and completeness are important tools to verify the correctness of our system. They verify that if we introduce and then immediately eliminate a connectives, this detour can be avoided. This is a local property of a derivation. However, it is possible that we introduce a connective and eliminate it at some later point in a proof, but not immediately. For example,

$$\dfrac{\dfrac{\dfrac{}{A \otimes B \Vdash A \otimes B}\,\mathsf{hyp} \qquad \dfrac{\begin{matrix}\mathcal{D}\\ A, B \Vdash C\end{matrix} \qquad \begin{matrix}\mathcal{E}\\ A, B \Vdash D\end{matrix}}{A, B \Vdash C \& D}\,\&\mathrm{I}}{A \otimes B \Vdash C \& D}\,\otimes\mathrm{E}}{A \otimes B \Vdash C}\,\&\mathrm{E_L}$$

This deduction contains a detour, since we first introduce $C \& D$ and then later eliminate it. In a derivation of this form, we cannot carry out a local reduction because $C \& D$ is introduced above and eliminated below the application $\otimes$E. In this case it is easy to see how to correct the problem: we move the application of $\&$E to come above the application of $\otimes$E, and then carry out a local reduction.

$$\dfrac{\dfrac{}{A \otimes B \Vdash A \otimes B}\,\mathsf{hyp} \qquad \dfrac{\dfrac{\begin{matrix}\mathcal{D}\\ A, B \Vdash C\end{matrix} \qquad \begin{matrix}\mathcal{E}\\ A, B \Vdash D\end{matrix}}{A, B \Vdash C \& D}\,\&\mathrm{I}}{A, B \Vdash C}\,\&\mathrm{E_L}}{A \otimes B \Vdash C}\,\otimes\mathrm{E}$$

This then reduces to

$$\dfrac{\dfrac{}{A \otimes B \Vdash A \otimes B}\,\mathsf{hyp} \qquad \begin{matrix}\mathcal{D}\\ A, B \Vdash C\end{matrix}}{A \otimes B \Vdash C}\,\otimes\mathrm{E}$$

What we eventually want to show is global soundness and completeness.

Global soundness states that every evident judgment $\Gamma; \Delta \vdash A$ *true* has a derivation in which we only apply introduction rules to conclusions that we are trying to prove and elimination rules to hypotheses or consequences we have derived from them. We all such derivations *normal.* Normal derivations have the important *subformula property*: every judgment occurring in a normal derivation of $\Gamma; \Delta \vdash A$ *true* refers only to subformulas of Γ, Δ, and A. This means our definition of truth is internally consistent and well-founded. It also means that our connectives are orthogonal to each other: we can understand each connective in isolation, falling back only on judgmental notions in their definition.

Global completeness means that every evident judgment $\Gamma; \Delta \vdash A$ *true* has a derivation in which every conclusion is eventually inferred by an introduction rule. For this to be the case, the elimination rules need to be strong enough so we can decompose our hypothesis and reassemble the conclusion from the atomic constituents, where a proposition is atomic if it doesn't have a top-level logical connective. We call such a derivation a *long normal* derivation, because it corresponds to the notion of *long normal form* in λ-calculi.

In order to prove these properties for our logic, we need to define more formally what *normal* and *long normal* deductions are. We postpone the discussion of long normal derivation and just concentrate on normal derivations in this section. We express this by two mutually recursive judgments that reflect the nature of hypothetical reasoning with introduction and elimination rules.

$\Gamma; \Delta \vdash A \uparrow$	A has a normal derivation, and
$\Gamma; \Delta \vdash A \downarrow$	A has an atomic derivation,

These formalize an intuitive strategy in constructing natural deductions is to apply introduction rules backwards to break the conclusion into subgoals and to apply elimination rules to hypotheses until the two meet. These judgments are defined by the following inference rules.

Hypotheses.

$$\frac{}{\Gamma; u{:}A \vdash A\ \downarrow}\,u \qquad \frac{}{(\Gamma, v{:}A); \cdot \vdash A\ \downarrow}\,v$$

Multiplicative Connectives.

$$\frac{\Gamma; \Delta_1 \vdash A\ \uparrow \qquad \Gamma; \Delta_2 \vdash B\ \uparrow}{\Gamma; (\Delta_1, \Delta_2) \vdash A \otimes B\ \uparrow}\,\otimes\mathrm{I}$$

$$\frac{\Gamma; \Delta \vdash A \otimes B\ \downarrow \qquad \Gamma; (\Delta', u{:}A, w{:}B) \vdash C\ \uparrow}{\Gamma; (\Delta, \Delta') \vdash C\ \uparrow}\,\otimes\mathrm{E}$$

$$\frac{\Gamma; (\Delta, u{:}A) \vdash B\ \uparrow}{\Gamma; \Delta \vdash A \multimap B\ \uparrow}\,{\multimap}\mathrm{I} \qquad \frac{\Gamma; \Delta \vdash A \multimap B\ \downarrow \qquad \Gamma; \Delta' \vdash A\ \uparrow}{\Gamma; \Delta, \Delta' \vdash B\ \downarrow}\,{\multimap}\mathrm{E}$$

$$\frac{}{\Gamma; \cdot \vdash \mathbf{1}\ \uparrow}\,\mathbf{1}\mathrm{I} \qquad \frac{\Gamma; \Delta \vdash \mathbf{1}\ \downarrow \qquad \Gamma; \Delta' \vdash C\ \uparrow}{\Gamma; (\Delta, \Delta') \vdash C\ \uparrow}\,\mathbf{1}\mathrm{E}$$

Additive Connectives.

$$\frac{\Gamma; \Delta \vdash A \uparrow \qquad \Gamma; \Delta \vdash B \uparrow}{\Gamma; \Delta \vdash A \& B \uparrow}\,\&\mathrm{I} \qquad \frac{\Gamma; \Delta \vdash A \& B \downarrow}{\Gamma; \Delta \vdash A \downarrow}\,\&\mathrm{E}_{\mathrm{L}} \qquad \frac{\Gamma; \Delta \vdash A \& B \downarrow}{\Gamma; \Delta \vdash B \downarrow}\,\&\mathrm{E}_{\mathrm{R}}$$

$$\frac{}{\Gamma; \Delta \vdash \top \uparrow}\,\top\mathrm{I} \qquad \textit{No } \top \textit{ elimination rule}$$

$$\frac{\Gamma; \Delta \vdash A \uparrow}{\Gamma; \Delta \vdash A \oplus B \uparrow}\,\oplus\mathrm{I}_{\mathrm{L}} \qquad \frac{\Gamma; \Delta \vdash B \uparrow}{\Gamma; \Delta \vdash A \oplus B \uparrow}\,\oplus\mathrm{I}_{\mathrm{R}}$$

$$\frac{\Gamma; \Delta \vdash A \oplus B \downarrow \quad \Gamma; (\Delta', u{:}A) \vdash C \uparrow \quad \Gamma; (\Delta', w{:}B) \vdash C \uparrow}{\Gamma; (\Delta, \Delta') \vdash C \uparrow}\,\oplus E$$

$$\textit{No } \mathbf{0} \textit{ introduction rule} \qquad \frac{\Gamma; \Delta \vdash 0 \downarrow}{\Gamma; (\Delta, \Delta') \vdash C \uparrow}\,\mathbf{0}\mathrm{E}$$

Quantifiers.

$$\frac{\Gamma; \Delta \vdash [a/x]A \uparrow}{\Gamma; \Delta \vdash \forall x.\ A \uparrow}\,\forall\mathrm{I}^a \qquad \frac{\Gamma; \Delta \vdash \forall x.\ A \downarrow}{\Gamma; \Delta \vdash [t/x]A \downarrow}\,\forall\mathrm{E}$$

$$\frac{\Gamma; \Delta \vdash [t/x]A \uparrow}{\Gamma; \Delta \vdash \exists x.\ A \uparrow}\,\exists\mathrm{I} \qquad \frac{\Gamma; \Delta \vdash \exists x.\ A \downarrow \qquad \Gamma; (\Delta', u{:}[a/x]A) \vdash C \uparrow}{\Gamma; (\Delta', \Delta) \vdash C \uparrow}\,\exists\mathrm{E}^a$$

Exponentials.

$$\frac{(\Gamma, v{:}A); \Delta \vdash B \uparrow}{\Gamma; \Delta \vdash A \supset B \uparrow}\,\supset\mathrm{I} \qquad \frac{\Gamma; \Delta \vdash A \supset B \downarrow \qquad \Gamma; \cdot \vdash A \uparrow}{\Gamma; \Delta \vdash B \downarrow}\,\supset\mathrm{E}$$

$$\frac{\Gamma; \cdot \vdash A \uparrow}{\Gamma; \cdot \vdash {!A} \uparrow}\,!I \qquad \frac{\Gamma; \Delta \vdash {!A} \downarrow \qquad (\Gamma, v{:}A); \Delta' \vdash C \uparrow}{\Gamma; (\Delta', \Delta) \vdash C \uparrow}\,!\mathrm{E}$$

Coercion.

$$\frac{\Gamma ; \Delta \vdash A \downarrow}{\Gamma ; \Delta \vdash A \uparrow} {\downarrow\uparrow}$$

The coercion $\downarrow\uparrow$ states that all atomic derivations should be considered normal. From the point of view of proof search this means that we can complete the derivation when forward and backward reasoning arrive at the same proposition. We obtain long normal derivation if we restrict the coercion rule to atomic propositions. It easy to see that these judgments just restrict the set of derivations.

Property 2.1 (Soundness of Normal Derivations)

1. *If* $\Gamma ; \Delta \vdash A \uparrow$ *then* $\Gamma ; \Delta \vdash A$.
2. *If* $\Gamma ; \Delta \vdash A \downarrow$ *then* $\Gamma ; \Delta \vdash A$.

Proof: *By simultaneous induction on the given derivations. The computational contents of this proof are the obvious structural translation from* $\mathcal{N} :: (\Gamma ; \Delta \vdash A \uparrow)$ *to* $\mathcal{N}^- :: (\Gamma ; \Delta \vdash A)$ *and from* $\mathcal{A} :: (\Gamma ; \Delta \vdash A \downarrow)$ *to* $\mathcal{A}^- :: (\Gamma ; \Delta \vdash A)$. *Note that the coercion* $\downarrow\uparrow$ *disappears, since the translation of the premise and conclusion are identical.* □

The corresponding completeness theorem, namely that $\Gamma ; \Delta \vdash A$ implies $\Gamma ; \Delta \vdash A \uparrow$, also holds, but is quite difficult to prove. This is the subject of the Normalization Theorem **??**. Together with the two judgments about atomic and normal derivations, we have refined substitution principles. Since hypotheses are atomic, they permit only the substitution of atomic derivations for hypotheses.[2]

Lemma 2.2 (Substitution Principles for Atomic Derivations)

1. *If* $\Gamma ; \Delta \vdash A \downarrow$ *and* $\Gamma ; (\Delta', u{:}A) \vdash C \uparrow$ *then* $\Gamma ; (\Delta, \Delta') \vdash C \uparrow$.
2. *If* $\Gamma ; \Delta \vdash A \downarrow$ *and* $\Gamma ; (\Delta', u{:}A) \vdash C \downarrow$ *then then* $\Gamma ; (\Delta, \Delta') \vdash C \downarrow$.
3. *If* $\Gamma ; \cdot \vdash A \downarrow$ *and* $(\Gamma, v{:}A) ; \Delta \vdash C \uparrow$ *and then* $\Gamma ; \Delta \vdash C \uparrow$.
4. *If* $\Gamma ; \cdot \vdash A \downarrow$ *and* $(\Gamma, v{:}A) ; \Delta \vdash C \downarrow$ *and then* $\Gamma ; \Delta \vdash C \downarrow$.

Proof: By straightforward inductions over the structure of the derivation we substitute into, appealing to weakening and exchange properties. □

[2]We have not formally stated the substitution principles for natural deduction as theorems of the complete system. They can be proven easily by induction on the structure of the derivation we substitute into.