15-312 Foundations of Programming Languages
Recitation 2: Rule Induction

Daniel Spoonhower
spoons+@cs
September 3, 2003

1 Matching Parentheses

Recall from lecture our original definition (through inference rules) of the language of matching parentheses.

\[
\begin{align*}
\varepsilon & \xrightarrow{M} M_1 \\
\frac{s_1 M \quad s_2 M}{s_1 s_2 M} & \xrightarrow{M_2} \quad \frac{s M}{(s) M} \quad (s) M
\end{align*}
\]

Recall also our “parser” for this language, given in terms in the following judgment and inference rules.

\[
\begin{align*}
0 & \not\vdash \varepsilon \quad \vdash_1 \\
\frac{k + 1 \not\vdash s}{\vdash_2} \\
\frac{k \not\vdash (s)}{\vdash_3} \\
\frac{k > 1}{(k > 1)}
\end{align*}
\]

We would like to show that the languages defined by \(M\) and \(\not\vdash\) are one and the same, and we began in lecture with a proof of the following: if \(s M\) then \(0 \not\vdash s\).

We will continue today by showing inclusion in the opposite direction. In particular, that

**Theorem 1.** If \(0 \not\vdash s\) then \(s M\).

**Proof.** By rule induction on the derivation of \(0 \not\vdash s\). We consider each case in turn.

**(Rule \(\not\vdash_1\))** Then \(s = \varepsilon\).

\[s M\]

By \(M_1\)

**(Rule \(\not\vdash_2\))** Then \(s = (s')\).

\[k + 1 \not\vdash s'\]

Subderivation
What’s gone wrong? Normally, this is the point of the proof where we’d cleverly apply the induction hypothesis – why can we not do so in this case? What, according to a recent lecture, are our alternatives if we find ourselves stuck in such a situation?

The first answer here is to generalize the induction hypothesis. To claim the equivalence of the languages \( M \) and \( \sqcup \), our theorem is strong enough, but it is not strong enough for us to carry out our proof. Let’s try it again.

**Theorem 1 (Revised).** If \( k \vdash s \) then \( (\cdots (s \ M). \kern-1.5em k) \).

**Proof.** By rule induction on the derivation of \( k \vdash s \). We consider each case in turn.

(Rule \( \vdash_1 \)) (as above)

(Rule \( \vdash_2 \)) Then \( s = (s') \).

\[
\begin{array}{c}
k + 1 \vdash s' \\
\underbrace{\cdots (s' \ M}_{k+1} \\
\underbrace{\cdots (s \ M}_{k}
\end{array}
\]

Subderivation

By i.h.

Since \( s = (s') \).

(Rule \( \vdash_3 \)) Then \( s = )s' \) and \( k > 1 \).

\[
\begin{array}{c}
k - 1 \vdash s' \\
\underbrace{\cdots (s' \ M}_{k-1} \\
\underbrace{\cdots () \ M}_{k-1}
\end{array}
\]

Subderivation

By i.h.

By \( M_1, M_3 \)

We’re so close this time! We’d like to conclude \( (\cdots (()s' \ M (\text{equivalently } (\cdots (s' \ M). \kern-1.5em k) \), but we’re not quite there yet. Intuitively, this should work out: we should be able to add a pair of (balanced) parentheses anywhere within a string of whose parentheses are already matched. (Are you convinced? Try some examples.) We’ll use another strategy from lecture: we’ll prove a lemma! To keep the syntax under control, I’ll use \( t, r, \) and \( c \) instead of just \( s \) to stand for strings of parentheses.

**Lemma 2.** If \( l \ r \ M \) and \( c \ M \) then \( l \ c \ r \ M \).

(Before you read on, think about how we will go about proving this? By induction? Over what?)
Proof. By rule induction on the derivation of $lr \ M$. We consider each case in turn.

(Rule $M_1$) Then $lr = \varepsilon$.

$c \ M$ By assumption

$lcr \ M$ Since $l = r = \varepsilon$

(Rule $M_2$)

One might think that the derivation of $lr \ M$ looks something like this:

\[
\vdots \vdots
\]
\[
l M \quad r M
\]
\[
\frac{}{lr M}
\]

Why is this not the case? Just because we have chosen to break our string into two parts $l$ and $r$ doesn’t mean that they each have matching parentheses. (Think about where we’d like to use this lemma and about a statement of the form $l \triangleright r$. Must $l$ (in particular) and $r$ have matching parentheses?)

To complete this case, we must consider a number of subcases, one for each way that $lr$ might be broken down into two strings of matching parentheses. First we take the case where $l$ is split.

(Rule $M_2$, Subcase 1) Let $l = l_1 l_2$.

\[
\vdots \vdots
\]
\[
l_1 M \quad l_2 r M
\]
\[
\frac{}{l_1 l_2 r M}
\]

$l_2 cr \ M$ By i.h.

$l_1 l_2 cr \ M$ By $M_2$

$l cr \ M$ Since $l = l_1 l_2$

(Rule $M_2$, Subcase 2) Let $r = r_1 r_2$.

\[
\vdots \vdots
\]
\[
lr_1 M \quad r_2 M
\]
\[
\frac{}{lr_1 r_2 M}
\]

(as above)

(What if the split really was between $l$ and $r$? Do we need a separate case for this?)
(Rule M₃)

Again, we must consider each of the ways that \( lr \) might be split in a derivation that ends with

\[
\begin{array}{c}
\vdots \\
 s M \\
 (s) M
\end{array}
\]

(Rule M₃, Subcase 1) Let \( l = (l' \text{ and } r = r') \).

\[
\begin{array}{c}
\vdots \\
l' r' M \\
(l' r') M \\
l c r M
\end{array}
\]

Subderivation

By i.h.

By M₃

Since \( l = (l' \text{ and } r = r') \)

(Rule M₃, Subcase 2) Let \( l = \epsilon \) and \( r = (r') \).

\[
\begin{array}{c}
l r M \\
r M \\
c r M \\
l c r M
\end{array}
\]

By assumption

Since \( l = \epsilon \)

By M₂

Since \( l = \epsilon \)

(Rule M₃, Subcase 3) Let \( l = (l') \) and \( r = \epsilon \). (as above)

Given this lemma, we can now return to our main theorem. In fact, we now have all the right tools to complete the proof: the last case goes through easily using our new lemma.

1.1 Alternatives to Rule Induction?

We have focused this time on rule induction, but there are other properties of strings that we might reason about. In many cases, we might want to carry out some proof by reasoning inductively over the lengths of strings. (Quick: think of a handful from 212!) Reconsider the case from our lemma where we split \( l \) into two pieces \( l_1 \) and \( l_2 \). The end of the derivation looked something like this:

\[
\begin{array}{c}
\vdots \\
l_1 M \\
l_2 r M
\end{array}
\]

\[
\frac{l_1 l_2 r M}{l_1 l_2 r M}
\]
What if $l_1 = l_2 = \varepsilon$? Then $l_2 r$ is not any shorter than $l_1 l_2 r$! If we were to reason about the lengths of the strings in this case, we could not apply the induction hypothesis. Here (and in many proofs in this class) rule induction will prove to be the better choice.