

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.

$$\frac{}{\varepsilon M} M_1 \quad \frac{s_1 M \quad s_2 M}{s_1 s_2 M} M_2 \quad \frac{s M}{(s) M} M_3$$

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

$$\frac{}{\varepsilon \triangleright \varepsilon} \triangleright_1 \quad \frac{k(\triangleright s)}{k \triangleright (s)} \triangleright_2 \quad \frac{k \triangleright s}{k(\triangleright)s} \triangleright_3$$

We would like to show that the languages defined by M and \triangleright are one and the same, and we began in lecture with a proof of the following: if $s M$ then $\varepsilon \triangleright s$.

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

Theorem 1. *If $\varepsilon \triangleright s$ then $s M$.*

Proof. By rule induction on the derivation of $\varepsilon \triangleright s$. We consider each case in turn.

(Rule \triangleright_1) Then $s = \varepsilon$.

$s M$

By M_1

(Rule \triangleright_2) Then $s = (s'$.

$(\triangleright s'$
?

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 \triangleright , 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 $k \triangleright s$ then $k s M$.*

Proof. By rule induction on the derivation of $k \triangleright s$. We consider each case in turn.

(Rule \triangleright_1) (as above)

(Rule \triangleright_2) Then $s = (s'$

$k (\triangleright s'$	Subderivation
$k (s' M$	By i.h.
$k s M$	Since $s = (s'$

(Rule \triangleright_3) Then $s =)s'$ and $k = k'($

$k' \triangleright s'$	Subderivation
$k' s' M$	By i.h.
$() M$	By M_1, M_3
$?$	

We're so close this time! We'd like to conclude $k' () s' M$, 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 l , r , and c instead of just s to stand for strings of parentheses.

Lemma 2. *If $lr M$ and $c M$ then $lcr M$.*

(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 lrM looks something like this:

$$\frac{\begin{array}{c} \vdots \quad \vdots \\ lM \quad rM \end{array}}{lrM}$$

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 $l = l_1 l_2$.

$$\frac{\begin{array}{c} \vdots \quad \vdots \\ l_1M \quad l_2rM \end{array}}{l_1l_2rM}$$

l_2crM

l_1l_2crM

$lcrM$

By i.h.

By M_2

Since $l = l_1 l_2$

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

$$\frac{\begin{array}{c} \vdots \quad \vdots \\ lr_1M \quad r_2M \end{array}}{lr_1r_2M}$$

(as above)

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

(Rule M_3)

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

$$\frac{\begin{array}{c} \vdots \\ sM \end{array}}{(s)M}$$

(Rule M_3 , Subcase 1) Let $l = (l'$ and $r = r')$.

$$\frac{\begin{array}{c} \vdots \\ l' r' M \end{array}}{(l' r') M}$$

$l' r' M$	Subderivation
$l' c r' M$	By i.h.
$(l' c r') M$	By M_3
$l c r M$	Since $l = (l'$ and $r = r')$

(Rule M_3 , Subcase 2) Let $l = \varepsilon$ and $r = (r')$.

$l r M$	By assumption
$r M$	Since $l = \varepsilon$
$c r M$	By M_2
$l c r M$	Since $l = \varepsilon$

(Rule M_3 , Subcase 3) Let $l = (l')$ and $r = \varepsilon$. (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:

$$\frac{\begin{array}{cc} \vdots & \vdots \\ l_1 M & l_2 r M \end{array}}{l_1 l_2 r M}$$

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.