Appendix 20: Structural Induction

The relation **r** defined in the previous lecture is a function, but how could we prove it? Here's a semi-formal argument: The relation **r** is defined by a pair of constraints. The first constraint applies to a • expression that starts with `f`, and the result is determined uniquely by the right-hand part of the expression. The second constraint applies to a • expression that starts with `t`, and the result is determined uniquely as `t`. Since an expression can start with either `f` or `t`, then only one constraint will apply, and the result is unique.

The above argument is fairly convincing because it relies on a directly observable property of the expression: whether it is a • expression that starts with `f` or `t`. Other claims about relations over B may not be so simple to prove. For example, it is not immediately obvious that $eval_{\mathbf{r}}$ is a function. The fundamental reason is that B is recursively defined; there is no way to enumerate all of the interesting cases directly. To prove more general claims, we must use **induction**.

Mathematical induction arguments are based on the natural numbers. A claim is proved for 0, then for $n+1$ assuming the claim for n, and then the claim follows for all n. For most of the proofs we will consider, no natural number n is readily provided for elements of the set (such as B), but one could be derived for an expression based on the number of steps in an argument that the expression belongs to the set (as in §1.1).

Instead of finding appropriate numbers for set elements to enable mathematical induction, however, we will use **structural induction**, which operates directly on the grammar defining set elements. The underlying principle is the same.

20.1 Detecting the Need for Structural Induction

Structural induction applies when proving a claim about a recursively-defined set. The set is well-founded, of course, only if there are atomic elements in the definition; those elements constitute the **base cases** for induction proofs. The self-referential elements of the definition constitute the **induction cases**.

For example, suppose we have the following definition:

$$\begin{array}{rcl} P & = & \alpha \\ & | & (\beta \otimes P) \\ & | & (P \odot P) \end{array}$$

Example members of P include α, $(\beta \otimes \alpha)$, $(\alpha \odot \alpha)$, and $((\beta \otimes (\beta \otimes \alpha)) \odot (\beta \otimes \alpha))$. Here is a claim about elements of P:

> **Theorem 20.1**: For any P, P contains an equal number of βs and $\otimes$s.

The truth of this claim is obvious. But since Theorem 20.1 makes a claim about all possible instances of P, and since the set of Ps is recursively defined, *formally* it must be proved by structural induction.

> **Guideline:** To prove a claim about all elements of a recursively defined set, use structural induction.

The key property of a correctly constructed induction proof is that it is guaranteed to cover an entire set, such as the set of Ps. Here is an example, a proof of the above claim:

> **Proof for Theorem 20.1**: By induction on the structure of P.
>
> - Base cases:

  - **Case** α
    α contains 0 βs and 0 $\otimes$s, so the claim holds.

- Inductive cases:
  - **Case** $(\beta \otimes P)$
    By induction, since P is a substructure of $(\beta \otimes P)$, P contains an equal number—say, n—of βs and $\otimes$s. Therefore, $(\beta \otimes P)$ contains $n+1$ βs and $\otimes$s, and the claim holds.
  - **Case** $(P_1 \odot P_2)$
    By induction, P_1 contains an equal number—say, n_1—of βs and $\otimes$s. Similarly, P_2 contains n_2 βs and $\otimes$s. Therefore, $(P_1 \odot P_2)$ contains $n_1 + n_2$ βs and $\otimes$s, and the claim holds.

The above proof has a relatively standard shape. The introduction, "by induction on the structure of P" encapsulates a boilerplate argument, which runs as follows:

> The claim must hold for any P. So assume that an arbitrary P is provided. If P has the shape of a base case (i.e., no substructures that are Ps) then we show how we can deduce the claim immediately. Otherwise, we rely on induction and assume the claim for substructures within P, and then deduce the claim. The claim then holds for all P by the principle of structural induction.

The proof for Theorem 20.1 contains a single base case because the definition of P contains a single case that is not self-referential. The proof contains two inductive cases because the definition of P contains two self-referential cases.

> **Guideline:** For a structural induction proof, use the standard format. The section for base cases should contain one case for each base case in the set's definition. The section for induction cases should contain one case for each remaining case in the set's definition.

The standard shape for a structural induction proof serves as a guide for both the proof writer and the proof reader. The proof writer develops arguments for individual cases one by one. The proof reader can then quickly check that the proof has the right shape, and concentrate on each case to check its correctness separately.

Occasionally, within the section for base cases or induction cases in a proof, the case split uses some criteria other than the structure used for induction. This is acceptable as long as the case split merely collapses deeply nested cases, as compared to a standard-format proof. A non-standard case split must cover all of the base or inductive cases in an obvious way.

All proofs over the structure of P, including Theorem 20.1, have the same outline:

> **Proof for Theorem** : By induction on the structure of P.
>
> - Base cases:
>   - **Case** α
> $\cdots$
> - Inductive cases:
>   - **Case** $(\beta \otimes P)$
> $\cdots$ By induction, [claim about P] ...
>   - **Case** $(P_1 \odot P_2)$
> $\cdots$ By induction, [claim about P_1] and, by induction, [claim about P_2] $\cdots$

Only the details hidden by "$\cdots$" depend on the claim being proved. The following claim is also about all Ps, so it will have the same outline as above.

Theorem 20.2: For any P, P contains at least one α.

▷ **Exercise 20.1.** Prove Theorem 20.2.

20.2 Definitions with Ellipses

Beware of definitions that contain ellipses (or asterisks), because they contain hidden recursions. For example,

$$\begin{array}{rcl} W & = & \alpha \\ & | & (\beta W W \ldots W) \end{array}$$

allows an arbitrary number of Ws in the second case. It might be more precisely defined as

$$\begin{array}{rcl} W & = & \alpha \\ & | & (\beta Y) \\ Y & = & W \\ & | & YW \end{array}$$

Expanded this way, we can see that proofs over instances of W technically require mutual induction proofs over Y. In practice, however, the induction proof for Y is usually so obvious that we skip it, and instead work directly on the original definition.

Theorem 20.3: Every W contains α.

Proof for Theorem 20.3: By induction on the structure of W.

- Base case:
  - **Case** α
    The claim obviously holds.
- Inductive case:
  - **Case** $(\beta W_0 W_1 \ldots W_n)$
    Each W_i contains α, and W contains at least one W_i, so the claim holds.

The following theorem can also be proved reasonably without resorting to mutual induction.

Theorem 20.4: For any W, each β in W is preceded by an open parenthesis.

▷ **Exercise 20.2.** Prove Theorem 20.4.

20.3 Induction on Proof Trees

The following defines the set $\triangle P$, read "P is pointy":

$$\begin{array}{ll} \triangle\alpha & \text{[always]} \\ \triangle(P_1 \odot P_2) & \text{if } \triangle P_1 \text{ and } \triangle P_2 \end{array}$$

As usual, the set of $\triangle P$s is defined as the smallest set that satisfies the above rules. The first rule defines a base case, while the second one is self-referential.

Another common notation for defining a set like $\triangle P$ is derived from the notion of proof trees:

$$\triangle\alpha \qquad\qquad \frac{\triangle P_1 \quad \triangle P_2}{\triangle(P_1 \odot P_2)}$$

When $\triangle P$ appears above a line in a definition, we understand it to mean that there must be a pointy proof tree in its place with $\triangle P$ at the bottom. This convention is used in the self-referential second rule.

Both notations simultaneously define two sets: the set of pointy indications $\triangle P$, and the set of pointy proof trees. Nevertheless, both sets are simply patterns of symbols, defined recursively. Examples of pointy proof trees include the following:

$$\triangle\alpha \qquad \dfrac{\triangle\alpha \quad \dfrac{\triangle\alpha \quad \triangle\alpha}{\triangle(\alpha \odot \alpha)}}{\triangle(\alpha \odot (\alpha \odot \alpha))}$$

We can now write claims about $\triangle P$ and its pointy proof trees:

> **Theorem 20.5**: If $\triangle P$, then the pointy proof tree for $\triangle P$ contains an odd number of $\triangle$s.

The proof for this claim works just like the proof of a claim on P, by structural induction:

> **Proof for Theorem 20.5**: By induction on the structure of the pointy proof of $\triangle P$.
>
> - Base cases:
>   - **Case** $\triangle\alpha$
>   The complete tree contains $\triangle\alpha$, a line, and a check mark, which is one $\triangle$ total, so the claim holds.
> - Inductive cases:
>   - **Case** $\triangle(P_1 \odot P_2)$
>   The complete tree contains $\triangle(P_1 \odot P_2)$, plus trees for $\triangle P_1$ and $\triangle P_2$. By induction, since the tree for $\triangle P_1$ is a substructure of the tree for $\triangle(P_1 \odot P_2)$, the $\triangle P_1$ tree contains an odd number of $\triangle$s. Similarly, the $\triangle P_2$ tree contains an odd number of $\triangle$s. Two odd numbers sum to an even number, so the trees for $\triangle P_1$ and $\triangle P_2$ combined contain an even number of $\triangle$s. But the complete tree for $\triangle(P_1 \odot P_2)$ adds one more, giving an odd number of $\triangle$s total, and the claim holds.

20.4 Multiple Structures

Many useful claims about recursively-defined sets involve a connection between two different sets. For example, consider the following claim:

> **Theorem 20.6**: For all $\triangle P$, P contains no βs.

Should this claim be proved by structural induction on P, or on $\triangle P$? We can find the answer by looking closely at what the theorem lets us assume. The theorem starts "for all $\triangle P$", which means the proof must consider all possible cases of $\triangle P$. The proof should therefore proceed by induction on the structure of $\triangle P$. Then, in each case of $\triangle P$, the goal is to show something about a specific P, already determined by the choice of $\triangle P$.

> **Guideline:** A leading "for all" in a claim indicates a candidate for structural induction. An item in the claim that appears on the right-hand side of an implication is *not* a candidate.

In contrast to Theorem 20.6, Theorem 20.5 was stated "if $\triangle P$, then ... ". This "if ... then" pattern is actually serving as a form of "for all". When in doubt, try to restate the theorem in terms of "for all".

Proof for Theorem 20.6: By induction on the structure of the proof of $\triangle P$.

- Base cases:
  - **Case** $\triangle\alpha$
    In this case, P is α, and the claim holds.
- Inductive cases:
  - **Case** $\triangle(P_1 \odot P_2)$
    By induction, P_1 and P_2 each contain no βs, so $(P_1 \odot P_2)$ contains no βs.

Here is a related, but different claim:

Theorem 20.7: For all P, either 1) P contains a β, or 2) $\triangle P$.

▷ **Exercise 20.3.** Prove Theorem 20.7. The theorem must be proved over a different structure than Theorem 20.6.

20.5 More Definitions and More Proofs

We can define more sets and make claims about their relationships. The proofs of those claims will follow in the same way as the ones we have seen above, differing only in the structure used for induction, and the local reasoning in each case.

Here is a new set:

$$
\begin{array}{lcll}
((\beta \otimes \alpha) \odot \alpha) & \diamond & (\beta \otimes \alpha) & \\
(\alpha \odot (\beta \otimes \alpha)) & \diamond & \alpha & \\
(\alpha \odot \alpha) & \diamond & \alpha & \\
(P_1 \odot P_2) & \diamond & (P_1' \odot P_2) & \text{if } P_1 \diamond P_1' \\
(P_1 \odot P_2) & \diamond & (P_1 \odot P_2') & \text{if } P_2 \diamond P_2'
\end{array}
$$

Like our original definition for $\triangle P$, $P \diamond P$ is defined by the least set satisfying the above rules. Examples for this set include $((\beta \otimes \alpha) \odot \alpha) \diamond (\beta \otimes \alpha)$ and $(((\beta \otimes \alpha) \odot \alpha) \beta (\alpha \odot \alpha)) \diamond ((\beta \otimes \alpha) \beta (\alpha \odot \alpha))$. The atomic elements of the definition in this case are the first three rules, and the self-reference occurs in the last two rules.

Theorem 20.8: For all $P \diamond P'$, P contains more $\odot$s than P'.

Proof for Theorem 20.8: By induction on the structure of $P \diamond P'$.

- Base cases:
  - **Case** $((\beta \otimes \alpha) \odot \alpha) \diamond (\beta \otimes \alpha)$
    In this case, P is $((\beta \otimes \alpha) \odot \alpha)$, which has one $\odot$, and P' is $(\beta \otimes \alpha)$, which has zero $\odot$s, so the claim holds.
  - **Case** $(\alpha \odot (\beta \otimes \alpha)) \diamond \alpha$
    In this case, P is $(\alpha \odot (\beta \otimes \alpha))$, which has one $\odot$, and P' is α, which has zero $\odot$s, so the claim holds.
  - **Case** $(\alpha \odot \alpha) \diamond \alpha$
    In this case, P is $(\alpha \odot \alpha)$, which has one $\odot$, and P' is α, which has zero $\odot$s, so the claim holds.
- Inductive cases:

- **Case** $(P_1 \odot P_2) \diamond (P_1' \odot P_2)$
  In this case, P is $(P_1 \odot P_2)$, which means it has as many $\odot$s as P_1 and P_2 combined. P' is $(P_1' \odot P_2)$, which means it has as many $\odot$s as P_1' and P_2 combined. By induction, P_1 has more $\odot$s than P_1', so P has more $\odot$s than P', and the claim holds.
- **Case** $(P_1 \odot P_2) \diamond (P_1 \odot P_2')$
  Analogous to the previous case.

Here is one last definition:

$$\begin{array}{rcl} V & = & \alpha \\ & | & (\beta \otimes V) \end{array}$$

Theorem 20.9: Every V is in P.

Theorem 20.10: If $\triangle P$ and P is not a V, then $P \diamond P'$ for some P'

Theorem 20.11: If $\triangle P$ and $P \diamond P'$, then $\triangle P'$.

▷ **Exercise 20.4.** Prove Theorem 20.9.

▷ **Exercise 20.5.** Prove Theorem 20.10.

▷ **Exercise 20.6.** Prove Theorem 20.11. The proof can proceed in two different ways, since the implicit "for all" applies to both $\triangle P$ and $P \diamond P'$.