Programming Languages: Theory and Practice

(WORKING DRAFT OF AUGUST 28, 2002)

Robert Harper
Carnegie Mellon University

Spring Semester, 2002

Preface

This is a collection of lecture notes for Computer Science 15–312 *Programming Languages*. This course has been taught by the author in the Spring of 1999 and 2000 at Carnegie Mellon University, and by Andrew Appel in the Fall of 1999, 2000, and 2001 at Princeton University. I am grateful to Andrew for his advice and suggestions, and to our students at both Carnegie Mellon and Princeton whose enthusiasm (and patience!) was instrumental in helping to create the course and this text.

What follows is a working draft of a planned book that seeks to strike a careful balance between developing the theoretical foundations of programming languages and explaining the pragmatic issues involved in their design and implementation. Many considerations come into play in the design of a programming language. I seek here to demonstrate the central role of type theory and operational semantics in helping to define a language and to understand its properties.

Comments and suggestions are most welcome. Please send any you may have to me by electronic mail.

Enjoy!

Contents

Part I

Preliminaries

Chapter 1

Inductive Definitions

Inductive definitions are an indispensable tool in the study of programming languages. In this chapter we will develop the basic framework of inductive definitions, and give some examples of their use.

1.1 Informal Overview

In this section we give an informal overview of inductive definitions, with an emphasis on how they are used in practice.

1.1.1 Judgements and Rules

An inductive definition consists of a collection of *inference rules* defining one or more *judgements*. A judgement is an assertion stating that a property holds of some object. For example, the judgement x nat might state that x is a natural number, and the judgement t tree might state that t is a binary tree.

The inference rules determine the conditions under which a judgement may be *inferred*, or *derived*. An inference rule has the form of an implication, stating that a judgement is inferrable whenever some other judgements (possibly none) are inferrable. Rules are written in the form

$$\frac{J_1 \ \ldots \ J_n}{J}$$

where J and each J_i $(1 \leq i \leq n)$ are judgements. The judgement J is called the *conclusion* of the rule, and the judgements $J_1, \ldots, J_n$ are its *premises*. If a rule has no premises (*i.e.*, $n = 0$), the rule is called an *axiom*.

A rule of this form states that the judgement J is inferrable, provided that each of the judgements $J_1, \ldots, J_n$ is inferrable. Thus axioms state that a judgement is inferrable unconditionally, whereas rules with premises state the conditional inferrability of a judgement. For example, the following set of rules, $\mathcal{R}_N$, constitute an inductive definition of the judgement x nat:

$$\frac{}{\texttt{zero}\ \mathsf{nat}} \qquad \frac{x\ \mathsf{nat}}{\texttt{succ}(x)\ \mathsf{nat}}$$

The first rule states that `zero` is a natural number. The second states that *if* x is a natural number, *then* so is $\texttt{succ}(x)$.

Rules may be *composed* to form a *derivation* of a judgement J from *premises* $J_1, \ldots, J_n$. A derivation is a tree whose nodes are judgements such that the children of a node are the premises of some rule ending with the judgement at that node. Such a tree is a derivation of a judgement J from premises $J_1, \ldots, J_n$ iff the root of the tree is J and its leaves are the judgements $J_1, \ldots, J_n$.

Derivation trees are normally depicted as "stacked" inference rules. For example, here is a derivation of the judgement $\texttt{succ}(\texttt{succ}(\texttt{zero}))$ nat:

$$\frac{\dfrac{\dfrac{}{\texttt{zero}\ \mathsf{nat}}}{\texttt{succ}(\texttt{zero})\ \mathsf{nat}}}{\texttt{succ}(\texttt{succ}(\texttt{zero}))\ \mathsf{nat}}$$

To take another example, here is an inductive definition of the judgement t tree, stating that t is a binary tree:

$$\frac{}{\texttt{empty}\ \mathsf{tree}} \qquad \frac{x\ \mathsf{tree} \quad y\ \mathsf{tree}}{\texttt{node}(x, y)\ \mathsf{tree}}$$

Using these rules, we may construct a derivation of the judgement

$$\texttt{node}(\texttt{empty}, \texttt{node}(\texttt{empty}, \texttt{empty}))\ \mathsf{tree}$$

as follows:

$$\frac{\dfrac{}{\texttt{empty}\ \mathsf{tree}} \quad \dfrac{\dfrac{}{\texttt{empty}\ \mathsf{tree}} \quad \dfrac{}{\texttt{empty}\ \mathsf{tree}}}{\texttt{node}(\texttt{empty}, \texttt{empty})\ \mathsf{tree}}}{\texttt{node}(\texttt{empty}, \texttt{node}(\texttt{empty}, \texttt{empty}))\ \mathsf{tree}}$$

In practice, we find a derivation of a judgement J by starting with J and working "backwards", looking for a rule ending with J with premises $J_1, \ldots, J_n$, then finding derivations of each of the J_i's by the same procedure. This process is called *goal-directed search*; the judgement J is the *goal*, and each of the J_i's are *subgoals*. Note that there may be *many* rules ending with J; if we fail to find a derivation by using one rule, we may have to abandon the attempt, and try another rule instead. If J is, in fact, derivable, then this process will eventually find a derivation, but if not, there is no guarantee that it will terminate! We may, instead, futilely apply rules forever, introducing more sub-goals each time, and never completing the derivation.

Often we give a *simultaneous* inductive definition of several judgements at once. For example, here is a simultaneous inductive definition of the judgements t tree, stating that t is a *variadic* tree, and f forest, stating that f is a *variadic forest*. By "variadic" we mean that the number of children of any given node in a tree varies with each node.

$$\frac{f \ \mathsf{forest}}{\mathtt{node}(f) \ \mathsf{tree}} \qquad \frac{}{\mathtt{nil} \ \mathsf{forest}} \qquad \frac{t \ \mathsf{tree} \quad f \ \mathsf{forest}}{\mathtt{cons}(t, f) \ \mathsf{forest}}$$

1.1.2 Rule Induction

What makes an inductive definition *inductive* is that the rules are *exhaustive* in the sense that a judgement is defined to hold iff it can be inferred by these rules. This means that if a judgement J is inferrable from a rule set $\mathcal{R}$, then there must be a rule in $\mathcal{R}$ ending with J such that each of its premises are also inferrable. For example, if n nat is inferrable according to the rules $\mathcal{R}_N$, then either it is inferrable by the first rule, in which case $n = \mathtt{zero}$, or by the second, in which case $n = \mathtt{succ}(m)$ and m nat is itself inferrable. Similarly, if t tree is inferrable according to the rules $\mathcal{R}_T$ given above, then either $t = \mathtt{empty}$ or $t = \mathtt{node}(t_1, t_2)$, where t_1 tree and t_2 tree are both inferrable.

This observation provides the basis for reasoning about derivable judgements by *rule induction* (also known as *induction on derivations*). For any set of rules, $\mathcal{R}$, to show that a property P holds of every inferrable judgement, it is enough to show that for every rule

$$\frac{J_1 \quad \dots \quad J_n}{J}$$

in $\mathcal{R}$, if $J_1, \dots, J_n$ all have property P, then J also has property P. By doing this for every rule in $\mathcal{R}$, we cover all the cases, and establish that P holds for every inferrable judgement.

The assumption that P holds for each premise of a rule is called the *inductive hypothesis*. The proof that P holds for the conclusion, under these assumptions, is called the *inductive step*. In the case of axioms the inductive hypothesis is vavuous; we must simply establish the conclusion outright, with no further assumptions to help us. If we can carry out the inductive step for each rule in $\mathcal{R}$, we thereby establish that P holds for every inferrable judgement, since the inference must arise by the application of some rule whose premises are derivable (and hence, by inductive hypothesis, have the property P).

For example, consider again the rule set $\mathcal{R}_N$. The principle of rule induction for $\mathcal{R}_N$ states that to show $P(n\ \mathsf{nat})$, it is enough to show

1. $P(\texttt{zero}\ \mathsf{nat})$;
2. if $P(n\ \mathsf{nat})$, then $P(\texttt{succ}(n)\ \mathsf{nat})$.

This is, of course, the familiar *principle of mathematical induction*.

Similarly, the principle of rule induction for $\mathcal{R}_T$ states that if we are to show that $P(t\ \mathsf{tree})$, it is enough to show

1. $P(\texttt{empty}\ \mathsf{tree})$;
2. if $P(t_1\ \mathsf{tree})$ and $P(t_2\ \mathsf{tree})$, then $P(\texttt{node}(t_1, t_2)\ \mathsf{tree})$.

This is called the *principle of tree induction*, or *induction on the structure of a tree*.

As a notational convenience, when the judgements in question are all of the form $x\ l$ for x an object and l is a property of the object x, we often write $P_l(x)$, rather than the more cumbersome $P(x\ l)$. If there is only one form of judgement, $x\ l$, then we often drop the subscript entirely, writing just $P(x)$, rather than $P_l(x)$ or $P(x\ l)$. Thus, instead of writing $P(n\ \mathsf{nat})$, we may write $P_{mathsfnat}n$, or just $P(n)$, when it is clear from context that we are working the $\mathcal{R}_N$. Similarly $P(t\ \mathsf{tree})$ is often written $P_{\mathsf{tree}}(t)$, or just $P(t)$.

Rule sets that define more than one judgement give rise to proofs by *simultaneous* induction. For example, if we wish to show $P_{\mathsf{tree}}(t)$ for all t such that t tree and $P_{\mathsf{forest}}(f)$ for all f such that f forest, then it is enough to show

1. if $P_{\mathsf{forest}}(f)$, then $P_{\mathsf{tree}}(\texttt{node}(f))$.
2. $P_{\mathsf{forest}}(\texttt{nil})$.
3. if $P_{\mathsf{tree}}(t)$ and $P_{\mathsf{forest}}(f)$, then $P_{\mathsf{forest}}(\texttt{cons}(t, f))$.

It is easy to check that this induction principle follows from the general principle of rule induction by simply working through the rules $\mathcal{R}_{TF}$, taking account of the notational conventions just mentioned.

1.1.3 Defining Functions by Rule Induction

A common use of rule induction is to justify the definition of a function by a set of equations. For example, consider the following recursion equations:

$$\begin{array}{rcl} hgt_{\mathsf{tree}}(\mathtt{empty}) & = & 0 \\ hgt_{\mathsf{tree}}(\mathtt{node}(t_1, t_2)) & = & 1 + \max(hgt_{\mathsf{tree}}(t_1), hgt_{\mathsf{tree}}(t_2)) \end{array}$$

We prove by rule induction that if t tree then there exists a unique $n \geq 0$ such that $hgt_{\mathsf{tree}}(t) = n$. In other words, the above equations determine a *function*, hgt.

We consider each rule in $\mathcal{R}_T$ in turn. The first rule, stating that $\mathtt{empty}$ tree, is covered by the first equation. For the second rule, we may assume that hgt_{tree} assigns a unique height to t_1 and t_2. But then the second equation assigns a unique height to $t = \mathtt{node}(t_1, t_2)$.

Similarly, we may prove by simultaneous induction that the followng equations define the height of a variadic tree and a variadic forest:

$$hgt_{\mathsf{tree}}(\mathtt{node}(f)) = 1 + hgt_{\mathsf{forest}}(f)$$

and

$$\begin{array}{rcl} hgt_{\mathsf{forest}}(\mathtt{nil}) & = & 0 \\ hgt_{\mathsf{forest}}(\mathtt{cons}(t, f)) & = & \max(hgt_{\mathsf{tree}}(t), hgt_{\mathsf{forest}}(f)). \end{array}$$

It is easy to show by simultaneous induction that these equations determine two functions, hgt_{tree} and hgt_{forest}.

1.1.4 Admissible and Derivable Rules

Given an inductive definition consisting of a set of rules $\mathcal{R}$, there are two senses in which a rule

$$\frac{J_1 \quad \cdots \quad J_n}{J}$$

may be thought of as being *redundant*.

Such a rule is said to be *derivable* iff there is a derivation of J from premises $J_1, \ldots, J_n$. This means that there is a composition of rules starting with the J_i's and ending with J. For example, the following rule is derivable in $\mathcal{R}_N$:

$$\frac{n\ \mathsf{nat}}{\mathtt{succ}(\mathtt{succ}(\mathtt{succ}(n)))\ \mathsf{nat}.}$$

Its derivation is as follows:

$$\dfrac{\dfrac{\dfrac{n\ \mathsf{nat}}{\mathtt{succ}(n)\ \mathsf{nat}}}{\mathtt{succ}(\mathtt{succ}(n))\ \mathsf{nat}}}{\mathtt{succ}(\mathtt{succ}(\mathtt{succ}(n)))\ \mathsf{nat}}.$$

Such a rule is said to be *admissible* iff its conclusion is derivable from no premises whenever its premises are derivable from no premises. For example, the following rule as *admissible* in $\mathcal{R}_N$:

$$\frac{\mathtt{succ}(n)\ \mathsf{nat}}{n\ \mathsf{nat}}.$$

First, note that this rule is *not* derivable for any choice of n. For if $n = \mathtt{zero}$, then the only rule that applies has no premises, and if $n = \mathtt{succ}(m)$, then the only rule that applies has as premise m nat, rather than n nat. However, this rule *is* admissible! We may prove this by induction on the derivation of the premise of the rule. For if $\mathtt{succ}(n)$ nat is derivable from no premises, it can only be by second rule, which means that n nat is also derivable, as required.

While this example shows that not every admissible rule is derivable, the converse holds. For a rule to be derivable means precisely that if its premises are derivable, then so is its conclusion!

The distinction between admissible and derivable rules can be hard to grasp at first. One way to gain intuition is to note that if a rule is derivable in a rule set $\mathcal{R}$, then it remains derivable in any rule set $\mathcal{R}' \supseteq \mathcal{R}$. This is because the derivation of that rule depends only on what rules are available, and is not sensitive to whether any other rules are also available. In contrast a rule can be admissible in $\mathcal{R}$, but inadmissible in some extension $\mathcal{R}' \supseteq \mathcal{R}$! For example, suppose that we add to $\mathcal{R}_N$ the rule

$$\frac{}{\mathtt{succ}(\mathsf{junk})\ \mathsf{nat}}.$$

Now it is no longer the case that the rule

$$\frac{\mathtt{succ}(n)\ \mathsf{nat}}{n\ \mathsf{nat}}.$$

is admissible, because if the premise were derived by the additional rule, there is no way to obtain a derivation of junk nat!

Since admissibility is sensitive to which rules are *absent*, as well as to which are *present*, a proof of admissibility almost always proceeds by induction on one or more of its premises. This constitutes an exhaustive analysis of how the premises might have been derived, and concludes that in each case the conclusion must also have been derived. Adding an additional rule requires that we add an additional case to the proof, and there is no assurance (as we have just illustrated) that this will go through.

1.2 A More Rigorous Development

In this section we will give a more rigorous account of inductive definitions of a subset of a given set. This will include as a special case the foregoing treatment of inductive definitions of judgements, and will make clear the mathematical underpinnings of the principle of rule induction.

1.2.1 Universes

We will consider inductive definitions of subsets of some fixed *universe* of objects. In principle we may consider inductive definitions over any set of objects we like, but in practice we confine ourselves to sets of *finitary objects*, which can be put into one-to-one correspondence with the natural numbers. Given such a correspondence, it suffices to make all inductive definitions over the set of natural numbers. However, doing so requires that we explicitly define the encoding of each object of interest as a natural number, called its *Gödel number*. To avoid this complication we take a more liberal approach in which we admit inductive defnitions over any specified set of objects.

For example, we will make use of the set of *(finite) strings* over a given alphabet as a universe for inductive defitions. Let Σ be a countable set of *symbols*, or *letters*, or *characters*. For example, Σ might be the set of ASCII or UniCode characters. The set of *strings* over Σ, written Σ^*, consists of the finite sequences of symbols from Σ. We write $s_1\,s_2$ for the concatenation of the string s_1 followed by the string s_2, write ε for the null string, and treat every $a \in \Sigma$ as string of length 1.

Another example is the set of *(first-order) terms* over a given set of operators. Let $\mathcal{O}$ be a countable set of *operators*, and let $\alpha : \mathcal{O} \to \mathbb{N}$ be an assignment of *arities* to each of the operators. An operator $o \in \mathcal{O}$ of arity n (*i.e.*, for which $\alpha(o) = n$) is said to be n-ary; the 0-ary operators are called *constants*. The set $\mathcal{T}$ of *ast's*, or *terms*, consists of all expressions of the form $o(t_1, \ldots, t_n)$, where o is an n-ary operator, and $t_1, \ldots, t_n$ are themselves ast's. Such a term may be depicted as an ordered tree with root labelled by the operator o, and with n children corresponding to the terms $t_1, \ldots, t_n$.

We often work with combinations of these basic universes. For example, we may consider inductive subsets of $\mathcal{T} \times \mathcal{T}$, the set of ordered pairs of ast's, and so forth. Generally we will leave implicit the exact choice of the universe for a particular inductive definition.

1.2.2 Inference Rules

An inductive definition of a subset of a universe $\mathcal{U}$ consists of a collection of *rules* over $\mathcal{U}$. A rule over $\mathcal{U}$ has the form

$$\frac{x_1 \;\ldots x_n}{x}$$

where $x \in \mathcal{U}$ and each $x_i \in \mathcal{U}$ $(1 \leq i \leq n)$. Thus a rule consists of a finite subset of $\mathcal{U}$ and an element of $\mathcal{U}$. The element x is called the *conclusion* of the rule; the elements $x_1, \ldots, x_n$ are called the *premises* of the rule. A *rule set* is, quite obviously, a set of rules.

A subset $A \subseteq \mathcal{U}$ is *closed under* $\mathcal{R}$, or $\mathcal{R}$*-closed*, iff $x \in A$ whenever

$$\frac{x_1 \;\ldots\; x_n}{x}$$

is a rule in $\mathcal{R}$ and each $x_i \in A$ for every $1 \leq i \leq n$.

The subset $I = I(\mathcal{R})$ *inductively defined by* $\mathcal{R}$ is given by the equation

$$I(\mathcal{R}) = \bigcap \{\, A \subseteq \mathcal{U} \mid A \text{ is } \mathcal{R}\text{-closed} \,\}.$$

As we shall see shortly, this is the smallest set closed under $\mathcal{R}$.

For example, here is a set, $\mathcal{R}_P$, of rules for deriving strings that are, as we shall prove later, are palindromes:

$$\frac{}{\varepsilon} \qquad \frac{}{a} \qquad \frac{s}{a\,s\,a}$$

The set of rules $\mathcal{R}_P$ just given has $2 \times |\Sigma| + 1$ rules, where $|\Sigma|$ is the cardinality of the alphabet Σ. In particular, if Σ is infinite, then there are infinitely many rules! Since we cannot expect to write down infinitely many rules, we need some means of defining large (or even infinite) rule sets. Here we have specified these using *rule schemes*. A rule scheme is a rule involving one or more *parameters* ranging over a specified set (by default, the universe). For example, the third rule above is a rule scheme with two parameters, a and s. The rule scheme determines one rule for each possible choice of character $a \in \Sigma$ and $s \in \Sigma^*$.

A simultaneous inductive definition of one or more judgements can be considered a single inductive definition of a subset of a suitable universe by a simple "labelling" device. A simultaneous inductive definition of the judgements $x_1\, l_1, \ldots, x_n\, l_n$, where each x_i ranges over a universe $\mathcal{U}$, may be thought of as a simple inductive definition of a subset of the *disjoint union* of n copies of $\mathcal{U}$, namely

$$\mathcal{U} \times \{\, l_1, \ldots, l_n \,\} = \{\, x\, l_i \mid x \in \mathcal{U},\ 1 \leq i \leq n \,\},$$

where we write $x\ l_i$ for the ordered pair (x, l_i). The rules defining these judgements emerge as rules over this enlarged universe. Thus the rules $\mathcal{R}_{TF}$ given above may be seen as defining an inductive subset of the universe $\mathcal{T} \times \{\, \mathsf{tree}, \mathsf{forest} \,\}$.

1.2.3 Rule Induction

As we mentioned earlier, the set $I(\mathcal{R})$ is the *least* set closed under $\mathcal{R}$.

Theorem 1
Let $\mathcal{R}$ be a rule set over $\mathcal{U}$, and let $I = I(\mathcal{R})$.

1. *I is $\mathcal{R}$-closed.*

2. *If A is $\mathcal{R}$-closed, then $I \subseteq A$.*

Proof:

1. Suppose that

   $$\frac{x_1\ \ldots\ x_n}{x}$$

   is a rule in $\mathcal{R}$, and that

   $$X = \{\, x_1, \ldots, x_n \,\} \subseteq I.$$

   Since I is the intersection of all $\mathcal{R}$-closed sets, $X \subseteq A$ for each $\mathcal{R}$-closed set A. But then $x \in A$ for each such A, by the definition of $\mathcal{R}$-closure, and hence x is an element of their intersection, I.

2. If A is $\mathcal{R}$-closed, then it is among the sets in the intersection defining I. So $I \subseteq A$.

■

The importance of this theorem is that it licenses the principle of *proof by rule induction* for a rule set $\mathcal{R}$:

To show that $I(\mathcal{R}) \subseteq X$, it suffices to show that X is $\mathcal{R}$-closed.

That is, if we wish to show that $x \in X$ for every $x \in I(\mathcal{R})$, it is enough to show that X is closed under the rules $\mathcal{R}$.

Returning to the inductively defined set P above, suppose we wish to show that every $s \in P$ is in fact a palindrome. That is, we wish to show that

$$P \subseteq \{ s \in \Sigma^* \mid s = s^R \}.$$

For this to hold, it is enough to show that the set of palindromes is closed under the rules $\mathcal{R}$. We consider each rule in turn, showing that if the premises are palindromes, then so is the conclusion.

1. $\varepsilon = \varepsilon^R$, so ε is a palindrome.

2. $a = a^R$ for every $a \in \Sigma$, so a is a palindrome.

3. Assume that $s = s^R$. Observe that

$$\begin{array}{rcl} (a\,s\,a)^R & = & a\,s^R\,a \\ & = & a\,s\,a. \end{array}$$

This completes the proof.[1]

The *parity* of a palindrome is either 0 or 1, according to whether its length is either even or odd. Now that we know that the set of palindromes is inductively defined by the rules given earlier, we may define the parity function by the following equations:

$$\begin{array}{rcl} parity(\varepsilon) & = & 0 \\ parity(a) & = & 1 \\ parity(a\,s\,a) & = & parity(s) \end{array}$$

Notice that we include one clause of the function definition for each rule defining the domain of the function.

Why does this define a function? We must prove that if s is a palindrome, then there exists a unique $x \in \{0, 1\}$ such that $parity(s) = x$. This

[1]You might also like to prove that every palindrome is a member of P. This can be achieved by (strong) induction over the length of palindrome s.

may be proved by rule induction by showing that the property $P(s)$ given by the formula

$$\exists!\, x \in \{\, 0, 1 \,\}\ \mathit{parity}(s) = x$$

is closed under the rules $\mathcal{R}_P$. The first two rules, for the null string and the single-letter strings, are covered by the first two clauses of the definition of parity. For the third rule, we assume that parity is well-defined for s (*i.e.*, there exists a unique x such that $\mathit{parity}(s) = x$). But then it follows directly from the third clause of the definition of parity that it is uniquely defined for $a\,s\,a$.

1.2.4 Admissibility and Derivability

Using this machinery we may shed additional light on admissibility and derivability of inference rules. Fix a rule set $\mathcal{R}$ over some universe $\mathcal{U}$. A rule

$$\frac{x_1 \quad \cdots \quad x_n}{x}$$

is *derivable* iff $x \in I(\mathcal{R} \cup \{\, x_1, \ldots, x_n \,\})$. That is, we take x_1, ..., x_n as new axioms, and ask whether x is derivable according to this expansion of the rule set. The same rule is *admissible* iff $x \in I(\mathcal{R})$ whenever $x_i \in I(\mathcal{R})$ for each $1 \leq i \leq n$. That is, we check whether x is in the set inductively defined by $\mathcal{R}$, whenever the x_i's are in that same set.

For example, consider the rule set $\mathcal{R}_P$ defining the palindromic strings. It is easy to see that the rule

$$\frac{s}{a\,b\,s\,b\,a}$$

is derivable, since if we add s as a new axiom, then we can apply the third rule of $\mathcal{R}_P$ to this axiom twice to obtain a derivation of $a\,b\,s\,b\,a$. On the other hand, the rule

$$\frac{a\,s\,a}{s}$$

is admissible, since if $a\,s\,a \in I(\mathcal{R})$, then so we must also have $s \in I(\mathcal{R})$. This is easily proved by rule induction, showing that the set

$$\{\, t \mid t = a\,s\,a \text{ \textit{and} } s \in I(\mathcal{R}) \,\}$$

is $\mathcal{R}$-closed.

1.3 Exercises

1. Give (simultaneous) inductive definitions of various languages.
2. Prove properties of these languages, including well-definedness of certain functions over them.
3. Fill in missing proofs.

Chapter 2

Transition Systems

Transition systems are fundamental to the study of programming languages. They are used to describe the execution behavior of programs by defining an abstract computing device with a set, S, of *states* that are related by a *transition relation*, $\mapsto$. The transition relation describes how the state of the machine evolves during execution.

2.1 Transition Systems

A *transition system* consists of a set S of *states*, a subset $I \subseteq S$ of *initial* states, a subset $F \subseteq S$ of *final* states, and a binary *transition relation* $\mapsto \subseteq S \times S$. We write $s \mapsto s'$ to indicate that $(s, s') \in \mapsto$. It is convenient to require that $s \not\mapsto$ in the case that $s \in F$.

An *execution sequence* is a sequence of states $s_0, \ldots, s_n$ such that $s_0 \in I$, and $s_i \mapsto s_{i+1}$ for every $0 \leq i < n$. An execution sequence is *maximal* iff $s_n \not\mapsto$; it is *complete* iff it is maximal and, in addition, $s_n \in F$. Thus every complete execution sequence is maximal, but maximal sequences are not necessarily complete.

A state $s \in S$ for which there is no $s' \in S$ such that $s \mapsto s'$ is said to be *stuck*. We require that all final states are stuck: if $s \in F$, then $s \not\mapsto$. But not all stuck states are final; these correspond to "run-time errors", states for which there is no well-defined next state.

A transition system is *deterministic* iff for every $s \in S$ there exists at most one $s' \in S$ such that $s \mapsto s'$. Most of the transition systems we will consider in this book are deterministic, the notable exceptions being those used to model concurrency.

The *reflexive, transitive closure*, $\mapsto^*$, of the transition relation $\mapsto$ is inductively defined by the following rules:

$$\frac{}{s \mapsto^* s} \qquad \frac{s \mapsto s' \quad s' \mapsto^* s''}{s \mapsto^* s''}$$

It is easy to prove by rule induction that $\mapsto^*$ is indeed reflexive and transitive.

The *complete* transition relation, $\mapsto^!$ is the restriction to $\mapsto^*$ to $S \times F$. That is, $s \mapsto^! s'$ iff $s \mapsto^* s'$ and $s' \in F$.

The *multistep* transition relation, $\mapsto^n$, is defined by induction on $n \geq 0$ as follows:

$$\frac{}{s \mapsto^0 s} \qquad \frac{s \mapsto s' \quad s' \mapsto^n s''}{s \mapsto^{n+1} s''}$$

It is easy to show that $s \mapsto^* s'$ iff $s \mapsto^n s'$ for some $n \geq 0$.

Since the multistep transition is inductively defined, we may prove that $P(e, e')$ holds whenever $e \mapsto^* e'$ by showing

1. $P(e, e)$.
2. if $e \mapsto e'$ and $P(e', e'')$, then $P(e, e'')$.

The first requirement is to show that P is reflexive. The second is often described as showing that P is *closed under head expansion*, or *closed under reverse evaluation*.

2.2 Exercises

1. Prove that $s \mapsto^* s'$ iff there exists $n \geq 0$ such that $s \mapsto^n s'$.