Principia Nota

Henry Blanchette

Contents

1 Introduction

The point of this compilation is to record and formalize my notations. I am very intrigued by the aesthetics of code, where I mean code in the most general sense:

DEFINITION. A **code** is a set of consistent and formal rules for recorded expression.

Note that I didn't restrict code to only *computer* code. I think that, although there are obvious and important distinctions between code meant to be read by humans and code meant to read by machines, the distinctions are merely superficial. In my taxonomy, I label *Machine Language* the section for programming languages (usually written by humans) and machine/assembly-esque code.

There is a distinction between what I mean by *code* and what I mean by *language*. In fact, the term *language* actually has a technical meaning in computer science.

2 Plaintext

3 Archaic

4 Philosophy

4.1 Logic

4.1.1 Truth Table

4.1.2 Variable

4.2 Metaphysics

4.3 Epistemology

4.4 Ethics

5 Mathematics

5.1 Logic

5.2 Set Theory

5.3 Measure Theory

5.4 Number Theory

5.5 Algebra

5.6 Analysis

5.7 Calculus

5.8 Graph Theory

6 Computarelogy

6.1 System

6.1.1 File System

6.1.2

6.2 Programming

6.2.1 Special Symbol

Notation in programming languages often uses special symbols that resemble the look and function of mathematics ones, as well as adding a few new

function to available ASCII characters. Some symbols have multiple distinct meanings, which are seperated by commas in the right column of the table below.

Note that many programming languages devise their own special symbols or combinations of symbols that are language-specific and non-standard. This list records only the standard associations of the symbols across languages and academia. Additionally, many language-specific instances draw on these standards. For example, Python uses `//` for integer division, whereas it is standardly used to begin in-line comments. Why do languages differ is such odd, arbitrary ways? I don't know, however I may address this question later.

Symbol	Meanings
`+`	numeric addition, list join
`-`	numeric subtraction, cabob-case
`/`	numeric division
`*`	numeric multiplication, type product, pointer type
`^`	numeric power-of, bitwise XOR
`%`	numeric modulo
`&`	bitwise AND, reference-of (pointers context)
`|`	bitwise OR, case matching divider
`&&`	boolean AND
`||`	boolean OR
`!`	boolean NOT
`~`	bit inversion
`<<`	bits shift left
`>>`	bits shift right
`<`	numberic less than
`<=`	numeric less than or equal to
`>`	numeric greater than
`>=`	numeric greater than or equal to
`?`	conditional ternary
`=`	value assignment, token equality
`:=`	value assignment
`==`	value equality
`->`	function type constructor
`=>`	anonymous function, "maps to"
`,`	tuple
`;`	end of statement
`#`	in-line comment begin
`//`	in-line comment begin
`/* */`	multi-line comment block
`\`	begin escaped word
`' '`	a single of character type, unformatted string
`" "`	formatted string
`( )`	function call with enclosed parameters, associative grouping
`[ ]`	indexing operator (comes after target, encloses index value)
`{ }`	block with new scope, object body
`< >`	special parameters, type parameters
`$`	associative divide
`.`	scope tree name divider

6.2.2 Comment

6.3 Language

DEFINITION. A **letter** is a member of an alphabet. Letters are denoted by single ASCII characters. σ is associated with a generic letter.

DEFINITION. An **alphabet** is a finite set of letters. Σ is often used as the name of either a specific or generic alphabet.

DEFINITION. A **string** is a n-tuple of letters, where n is the length of the string.

DEFINITION. A **language** is set of strings that all have letters in some alphabet Σ. L is ogten used as the name f either a specific or generic language. $L(X)$ represents the language yielded by the X, where X is an automata, grammar, or other language-yielding construct.

6.3.1 Context-Free Grammar

DEFINITION. A context-free grammar (CFG) is a set of rules that yield a language (a context-free language (CFL)) with an alphabet Σ. Formally, a grammar G is given by

$$G = (S, \Sigma \cup \{\epsilon\}, V, R)$$

where S is the start token, $\Sigma \cup \{\epsilon\}$ is a set of terminals, V is a set of non-terminals, and R is the set of rules.

The language is yielded by applying staring with S and then applying the rules in every valid combination and order. In other words, a string is in the yielded language if it can be produced exactly by applying rules to S.

A **token** is either a letter in either Σ or V.

DEFINITION. If a token is **terminal**, then it cannot be manipulated by rule applications. If a terminal is produced by a rule application in a string, then it will persist till the string is completed. When a string contains only terminals, it is completed.

DEFINITION. If a token is **non-terminal**, then it must be manipulated by rule applications. A string is not completed until all of the non-terminals have been transformed into terminals by rule appliations.

The rules follow a few conventions.

1. Capital letters reprents non-terminals.
2. Rules are writted $A \to x$ where A is a non-terminal and x is a string of tokens. This represents the rule "any A can be replaced with x".
3. The left-hand-side of each rule must be a single non-terminal. The right-hand-side of each rule must be a string of tokens.
4. There must be at least one rule $S \to x$, where x is a string of tokens.
5. There must be at least one rule $A \to x$, where A is a non-terminal and x is a string of only terminals.

6. For each terminal in $A \in V$, there must be at least one rule $A \to x$, where x is a string of tokens.

7. A subset of R of the form $\{A \to x_1, A \to x_2, \ldots, A \to x_n\}$ can be written as $A \to x_1 \mid x_2 \mid \cdots \mid x_n$.

6.3.2 Regular Expression

Regular expressions are a rare instance of practicality derived from this subfield of Computarelogy. A regular expression represents a set of strings. The academic standard for regular expressions is the simple grammar

$S \to$	$\mid \epsilon$	empty string
	$\mid \sigma_i \mid \cdots \mid \sigma_n$	letter of Σ
	$\mid SS$	concatenation
	$\mid (S \cup S)$	set union
	$\mid (S)^*$	Kleene star: $\{x_1 x_2 \cdots x_n \mid \forall i \geq 0, x_i \in S\}$

Where $\Sigma = \{\sigma_i, \ldots, \sigma_n\}$ is the alphabet of the grammar. In addition to these standard rules, there are many other common rules:

$S+$	S^* while additionally requiring $i > 0$
$S?$	S^* while aditionally requiring $i = 0 \vee i = 1$
$S \mid S$	set union
$[\sigma_1 \cdots \sigma_n]$	letter union: $\cup_{i=1}^{n} \sigma_i$
$[\sigma_0 - \sigma_1]$	letters from σ_0 to σ_1 in the standard ordering of Σ
.	anything except newline
\$	end of the string

6.3.3 Automata

6.4 Computability

6.5 Complexity

6.6 Artificial Intelligence

7 Game Design

8 Physics