1. Declaration of program and validity of syntax

$sT = \{c \mid c \text{ is a character}\}$

$sR = \{\text{sT} \mid \text{sT is a statement}\}$

$Let\ syntaxRules = \{sR \mid sR \text{ is a rule}\}$

Assume semantic of a language

$Let\ L = (syntax, semantic)$, where syntax = syntaxRules

$compile(expression, syntaxRules) \rightarrow \begin{cases} 1, if\ expression \in syntaxRules \\ 0, otherwise \end{cases}$

$Let\ exp = \{c \mid c \text{ is a character}\}$

$Let\ expression = \{exp \mid exp \text{ is a chain of characters}\}$

$correctSyntax(expressions) = \forall expression \in expressions \mid \text{compile(expression, syntax)}$

$Let\ isSyntaxValid(P) \rightarrow \begin{cases} 1, if\ syntax(expressions) \\ 0, otherwise \end{cases}$

$SyntaxValid(P)\ iff\ isSyntaxValid(P) = 1$

P = {expressions}

2. Correctness of the program by using unit tests

$Let\ inputs_{ft} = \{v \mid v \text{ is a value}\}$

$Let\ func_{ft}: inputs_{ft} \rightarrow output_{ft}, where\ func_{ft} \text{ is a function}$

$Let\ t_j = (func_{ft}, inputs_{ft}, output_{ft})$

$Let\ ts_i = \{t_j \mid t_j \text{ is a test}\}$

$Let\ inputs_{fpb} = \{v \mid v \text{ is a value}\}$

$Let\ func_{fpb}: inputs_{fpb} \rightarrow output_{fpb}, where\ func_{fpb} \text{ is a function}$

$Let\ pb_i = (func_{fpb}, inputs_{fpb}, output_{fpb})$

$Let\ P = a\ program\ to\ test$

$Let\ Blocks(P) = \{pb_i \mid pb_i \text{ is a program block of } P\}$

$Let\ TS_{pb_i} = \{ts_i \mid ts_i \text{is a related set of tests for } pb_i\}$

$Let\ TS_P = \bigcup_{pb_i \in Blocks(P)} TS_{pb_i}$

$success(t_j, pb_i) \rightarrow \begin{cases} 1, if\ [t_j]_{output_{ft}} = [pb_i]_{output_{fpb}} \\ 0, otherwise \end{cases}$

$Let\ p = boolean\ variable\ s.t\ p \rightarrow \begin{cases} 1, if\ True \\ 0, otherwise \end{cases}$

Let $\varphi = p \wedge \neg p$

$$correct(ts_i, pb_i) = \forall t_j \in ts_i \mid success(t_j, pb_i)\ and\ success(t_j, pb_i) \vDash \varphi$$

$$correctBlock(TS_{pb_i}, pb_i) = \forall ts_i \in TS_{pb_i} \mid correct(ts_i, pb_i)$$

$$correctProgram(TS_P, Blocks(P)) = \forall pb_i \in Blocks(P), \forall TS_{pb_i} \in TS_P \mid correctBlock(TS_{pb_i}, pb_i)$$

3. Requirements Satisfaction

$$pre-condition = \{v \mid v\ is\ a\ value\}$$

$$post-condition = is\ a\ value$$

$$Let\ r_i = (pre-condition, post-condition)$$

$$Let\ RS_{pb_i} = \{r_i \mid r_i\ is\ a\ requirement\}$$

$$Let\ RS_P = \{RS_{pb_i} \mid RS_{pb_i}\ is\ a\ set\ of\ requirements\ for\ a\ program\ block\ i\}$$

$$conditionInput = [t_j]_{input_{ft}} = [pb_i]_{input_{fpb}} = [r_i]_{\ pre-condition}$$

$$conditionOutput = success(t_j, pb_i) = 1\ and\ [t_j]_{output_{ft}} = [pb_i]_{output_{fpb}} = [r_i]_{\ post-condition}$$

$$satisfy(success(t_j, pb_i), r_i) \rightarrow \begin{cases} 1, if\ conditionOutput \text{^} conditionInput \\ 0, otherwise \end{cases}$$

$$satisfaction(pb_i, ts_i, r_i) = \forall t_j \in ts_i \mid satisfy(success(t_j, pb_i), r_i)$$

$$satisfyBlock(TS_{pb_i}, pb_i, RS_{pb_i}) = \forall ts_i \in TS_{pb_i}, r_i \in RS_{pb_i} \mid satisfaction(pb_i, ts_i, r_i)$$

$$satisfyP(TS_P, Blocks(P), RS_P) = \forall pb_i \in Blocks(P), \forall TS_{pb_i} \in TS_P, \forall RS_{pb_i} \in RS_P \mid satisfyBlock(TS_{pb_i}, pb_i, RS_{pb_i})$$