

Language Definitions

Sujit Kumar Chakrabarti
`sujitkc@iiitb.ac.in`

January 2025

<i>spec</i>	→	$(g : Decl*, t : TypeDecl*, i : Init, f : Function*, b : Block*)$
<i>Decl</i>	→	$(n : \text{string}, t : TypeExpr)$
<i>FunDecl</i>	→	$(n : \text{string}, p : TypeExpr, r : TypeExpr)$
<i>TypeDecl</i>	→	<i>VariantDecl</i> <i>RecordDecl</i>
<i>VariantDecl</i>	→	<i>VariantConstructor</i> +
<i>RecordDecl</i>	→	<i>recname</i> : string , <i>fields</i> : <i>Decl</i> +
<i>TypeExpr</i>	→	<i>TypeConst</i> <i>TypeVariable</i>
		<i>FuncType</i>
		<i>MapType</i>
		<i>TupleType</i>
		<i>SetType</i>
		(More will be added as we get more examples)
<i>TypeConst</i>	→	$(n : \text{string})$
<i>FuncType</i>	→	$(p : TypeExpr*, r : TypeExpr)$
<i>MapType</i>	→	$(d : TypeExpr, r : TypeExpr)$
<i>TupleType</i>	→	$(et : TypeExpr+)$
<i>SetType</i>	→	$(et : TypeExpr)$
<i>Init</i>	→	$(v : \text{string}, e : Expr)$
Functions		
<i>FunDecl</i>	→	$(n : \text{string}, p : Decl*, r : \text{pair} < HTTPResponseCode, TypeExpr >)$
APIs		
<i>API</i>	→	$(pre : Expr, call : FuncCall, resp : (ret : HTTPResponseCode, resp : post : Expr))$

Figure 1: Abstract Syntax: API Specification Language

<i>Expr</i>	→	<i>Var</i> <i>FuncCall</i> <i>Num</i> <i>Set</i> <i>Map</i> <i>Tuple</i>
		(More will be added as we get more examples)
<i>Var</i>	→	$(n : \text{string})$
<i>FuncCall</i>	→	$(n : \text{string}, a : Expr*)$
<i>Num</i>	→	$(v : \text{int})$
<i>Set</i>	→	$(e : Expr*)$
<i>Map</i>	→	$(v : \text{pair} < Var, Expr >)$
<i>Tuple</i>	→	$(e : Expr)$

Figure 2: Abstract Syntax: Expressions

API Spec

Globals

$U: (\text{string}, \text{string}) \text{ map} = \{\}$
 $T: (\text{Token}, \text{string}) \text{ map} = \{\}$

Functions:

$\text{signup}: \text{string} \times \text{string} \rightarrow \text{HTTPResponse}$
 $\text{login}: \text{string} \times \text{string} \rightarrow \text{Token} \times \text{HTTPResponse}$

signup-OK

Precondition: $u \notin \text{dom}(U)$
API: $\text{signup}(u, p) \rightarrow \text{OK}$
Postcondition: $U[u] = p$

login-OK

Precondition: $t \notin \text{dom}(T) \quad U[u] = p$
API: $\text{login}(u, p) \rightarrow t$
Postcondition: $T[t] = u$

Figure 3: Example: API specification

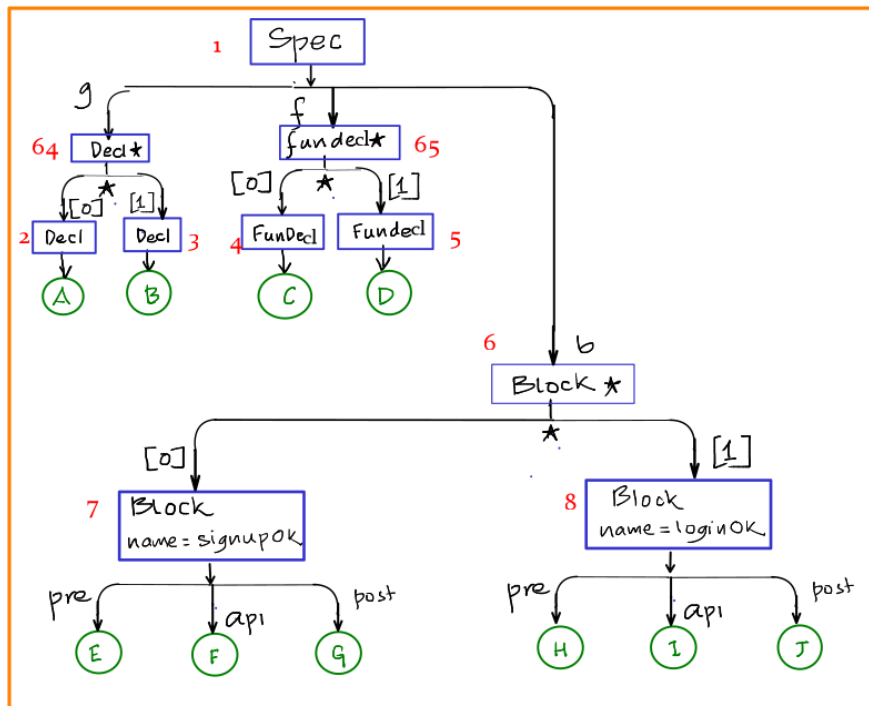
1 API Specification Language

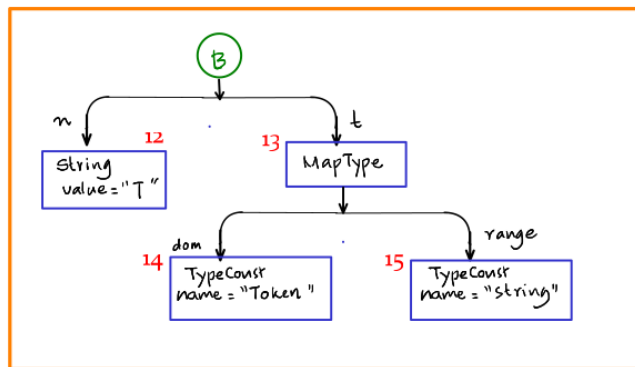
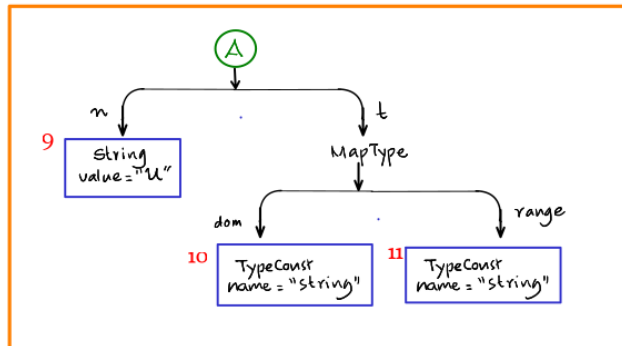
1.1 Abstract Syntax

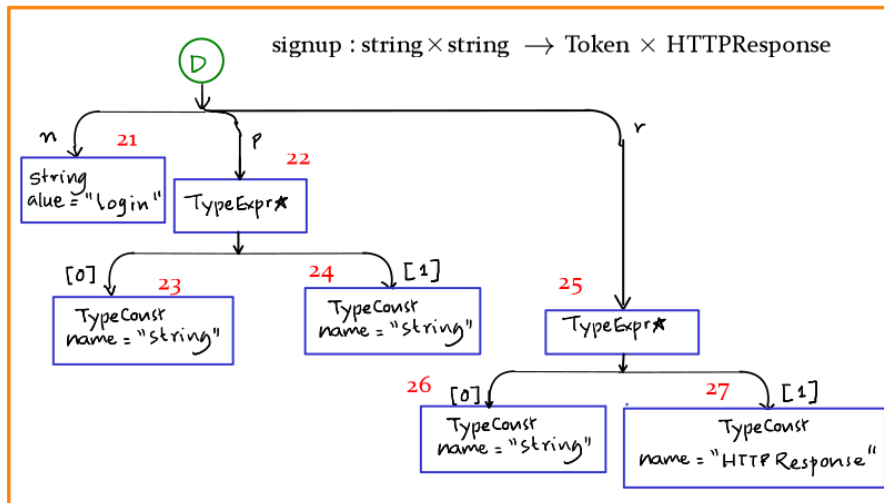
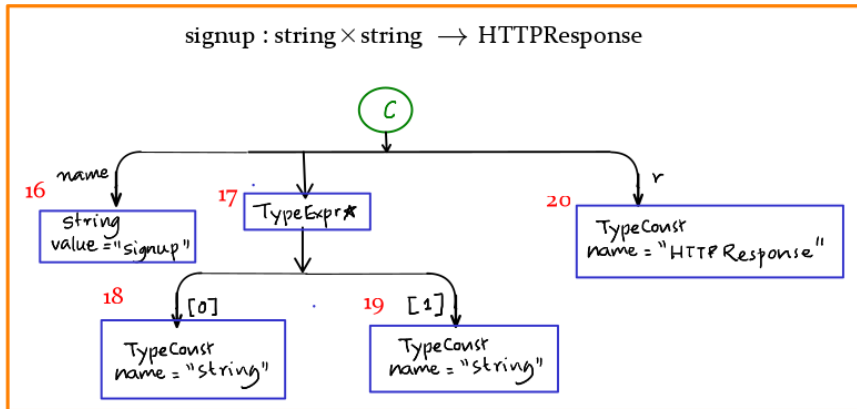
1.2 Example

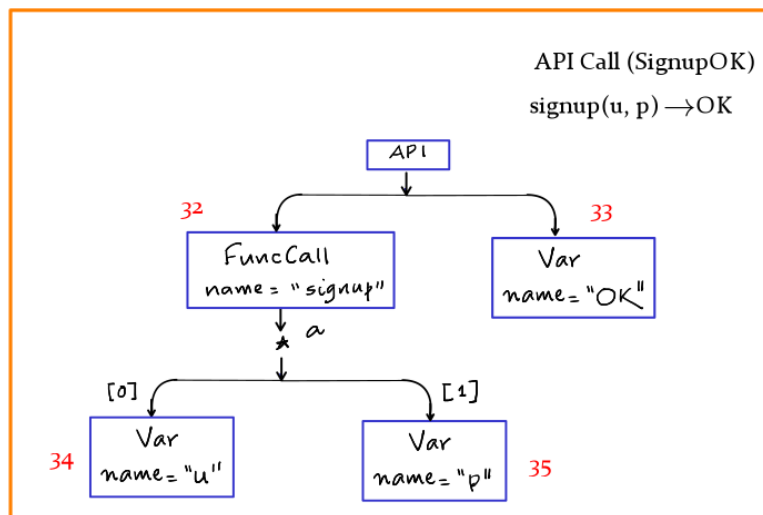
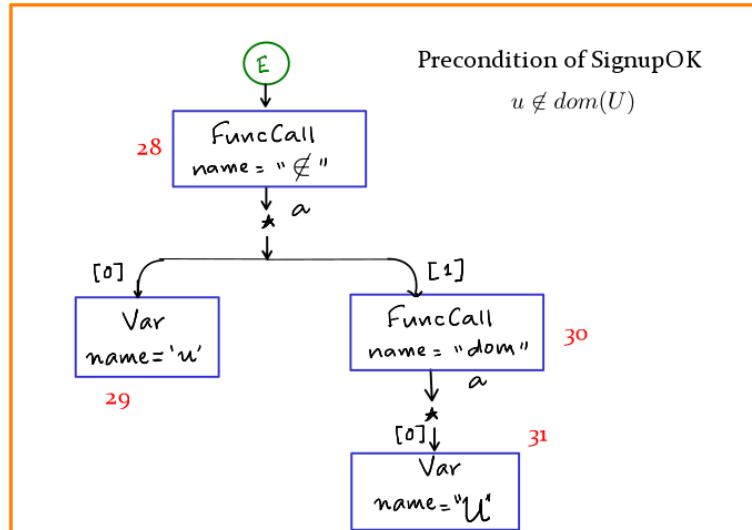
1.3 Example Specification – Signup-Login API

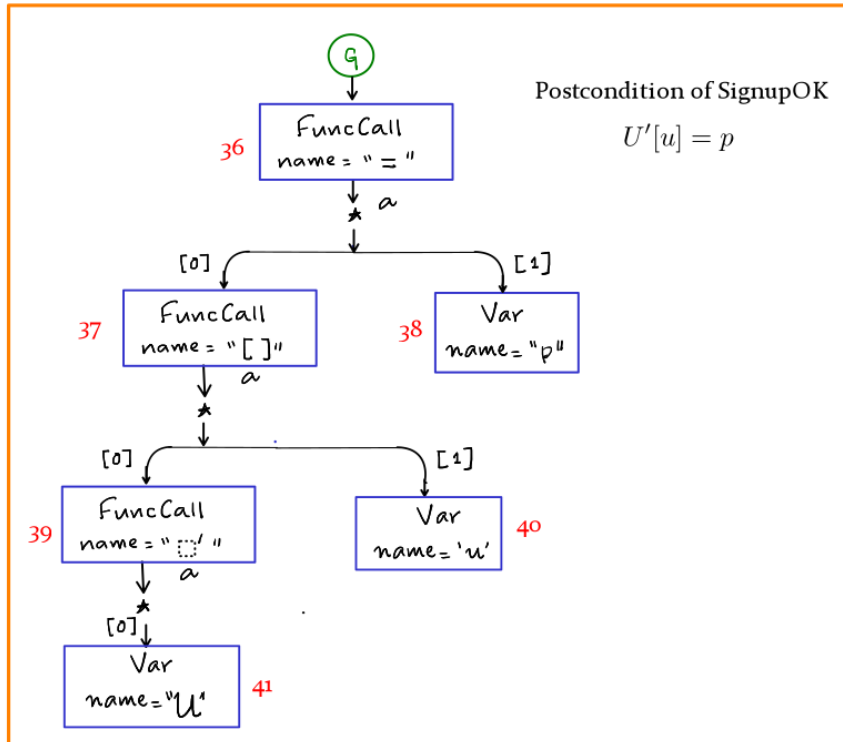
1.3.1 Abstract Syntax Tree

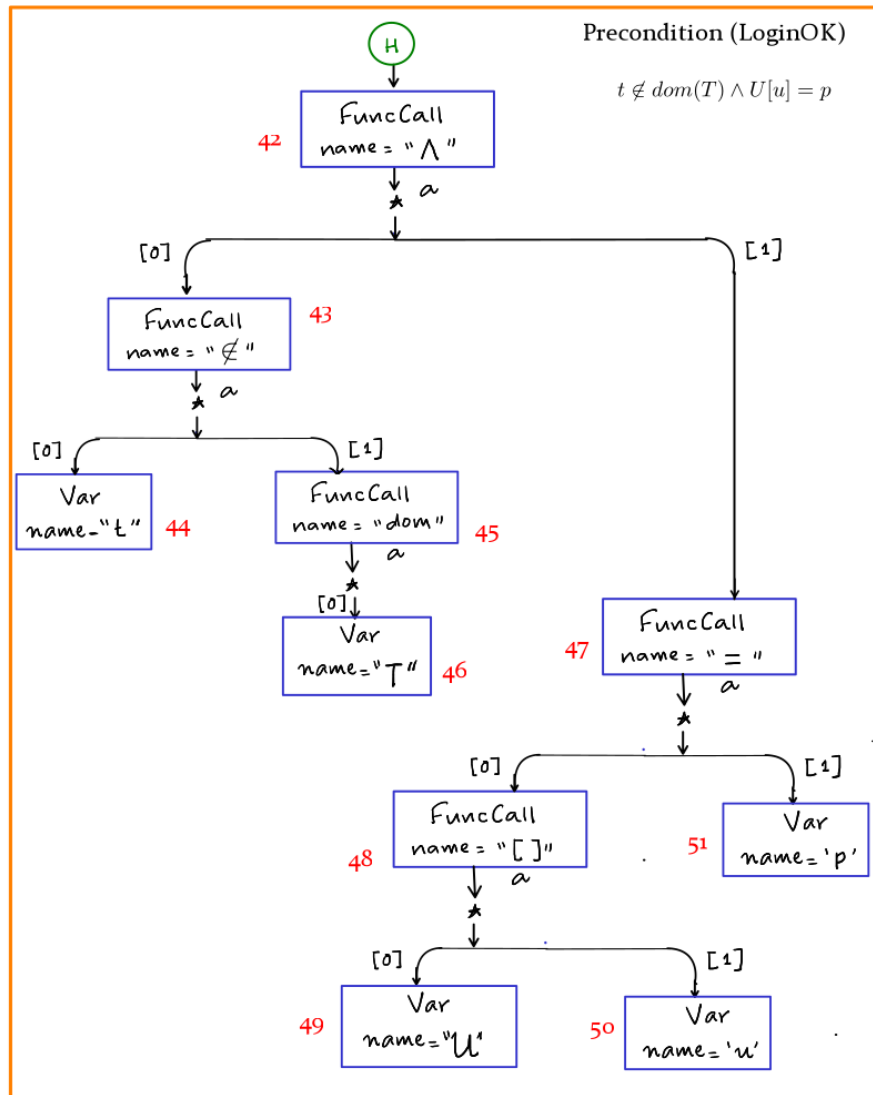


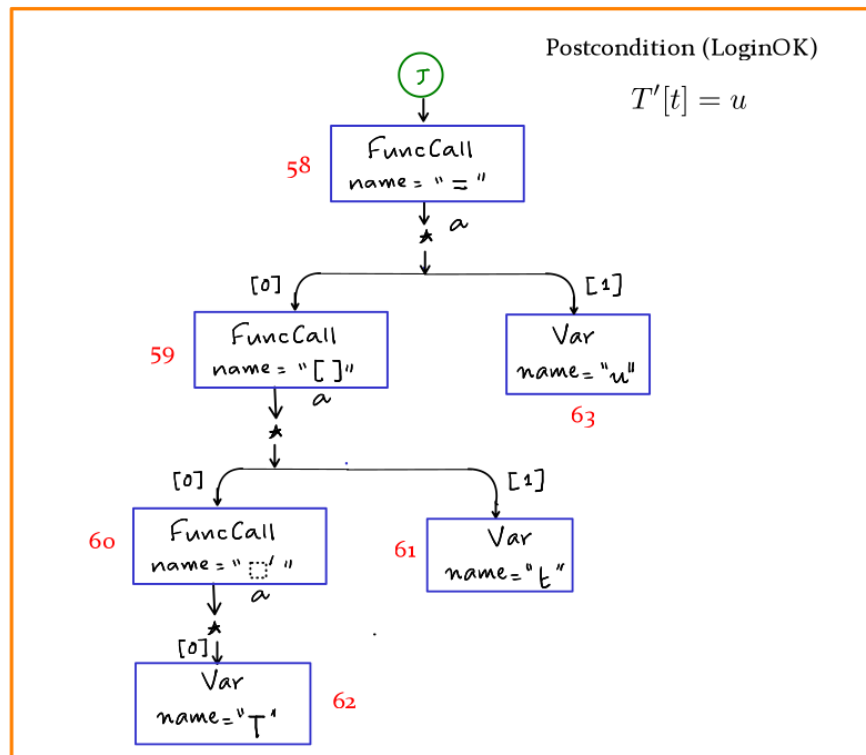
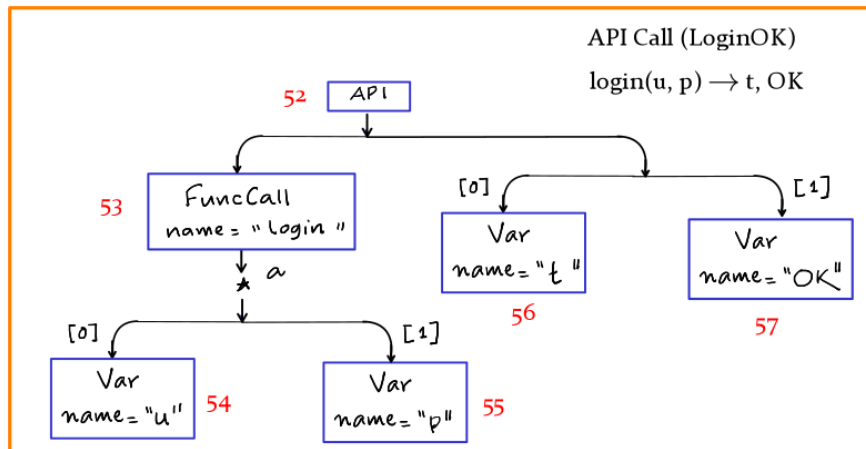












Abstract Test Case

$U := \{\}$
 $T := \{\}$
 $u_1 := \text{input} \langle \text{string} \rangle$
 $p_1 := \text{input} \langle \text{string} \rangle$
 $\text{assume}(u_1 \notin \text{dom}(U))$
 $r := \text{signup}(u_1, p_1)$
 $\text{assert}(U[u_1] = p_1 \wedge r = \text{OK})$

 $u_2 := \text{input} \langle \text{string} \rangle$
 $p_2 := \text{input} \langle \text{string} \rangle$
 $\text{assume}(t \notin \text{dom}(T) \wedge U[u_2] = p)$
 $r := \text{login}(u_2, p_2)$
 $\text{assert}(T[t] = u_2)$

Figure 4: Example – Abstract test case

2 Abstract Test Case

2.1 Example

2.2 Example Abstract Test Case – Signup(OK)→Login(OK)

2.2.1 Abstract Syntax Tree

