Language Definitions

Sujit Kumar Chakrabarti
sujitkc@iiitb.ac.in

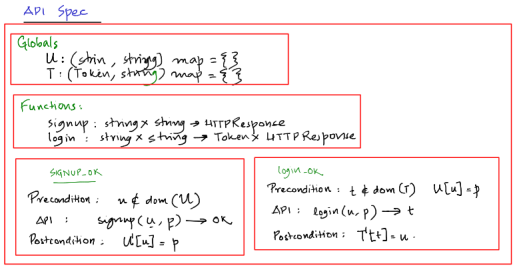
January 2025

1 API Specification Language

1.1 Abstract Syntax

1.2 Example

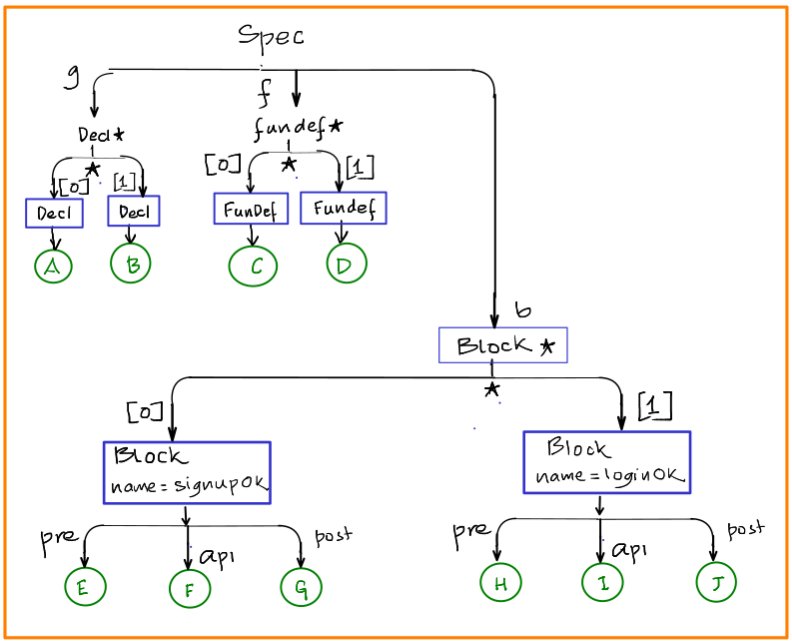
1.3 Example Specification – Signup-Login API

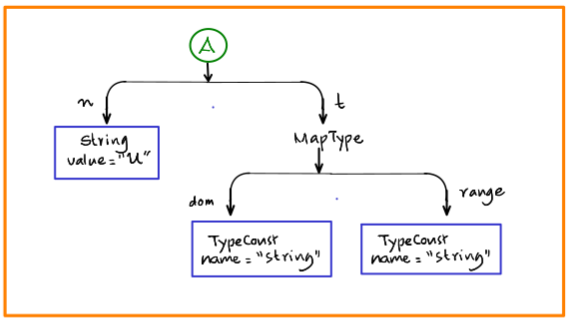


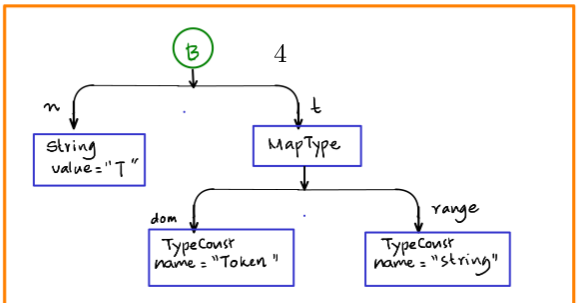
$spec$	$\rightarrow$	$(g : Decl\star, t: TypeDecl\star, i : Init, f : Function\star, a : API\star)$
$Decl$	$\rightarrow$	$(v : String, t : TypeExpr)$
$TypeDecl$	$\rightarrow$	$VariantDecl \mid RecordDecl$
$VariantDecl$	$\rightarrow$	$VariantConstructor+$
$TypeConstructor$	$\rightarrow$	$(constrname : String, TypeExpr\star)$
$RecordDecl$	$\rightarrow$	$recname : String, fields : Decl+$
$TypeExpr$	$\rightarrow$	$TypeConst \mid TypeVariable$ $\mid FuncType$ $\mid MapType$ $\mid TupleType$ $\mid SetType$
$TypeConst$	$\rightarrow$	$String$
$FuncType$	$\rightarrow$	$(args: TypeExpr\star, ret: TypeExpr)$
$MapType$	$\rightarrow$	$(key: TypeExpr, val: TypeExpr)$
$TupleType$	$\rightarrow$	$(elements: TypeExpr+)$
$SetType$	$\rightarrow$	$(elttype : TypeExpr)$
$Init$	$\rightarrow$	$(v : String, e : Expr)$
		Functions
$FunctionDecl$	$\rightarrow$	$(fname$: String, $pars : Decl*, ret : HTTPResponseCode, resp : TypeExpr)$
		APIs
API	$\rightarrow$	$(pre : Expr, call: FuncCall, resp : (ret : HTTPResponseCode, resp : post: Expr)$

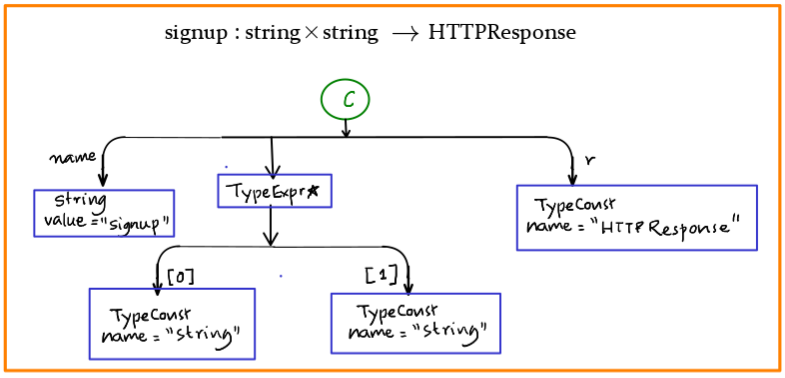
Figure 1: Abstract Syntax: (a) API Specification Language; (b) Abstract Test Case Language; (c) Common Parts

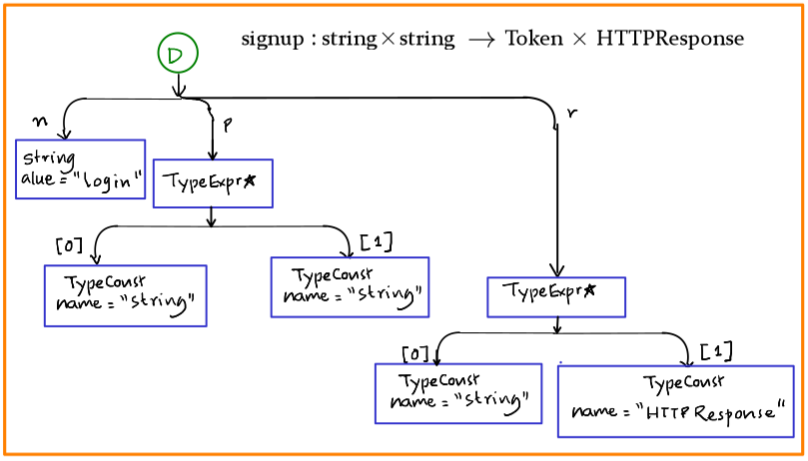
1.3.1 Abstract Syntax Tree

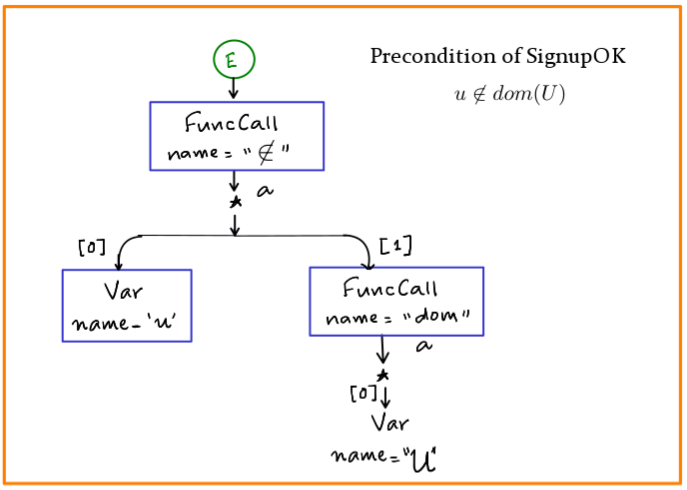


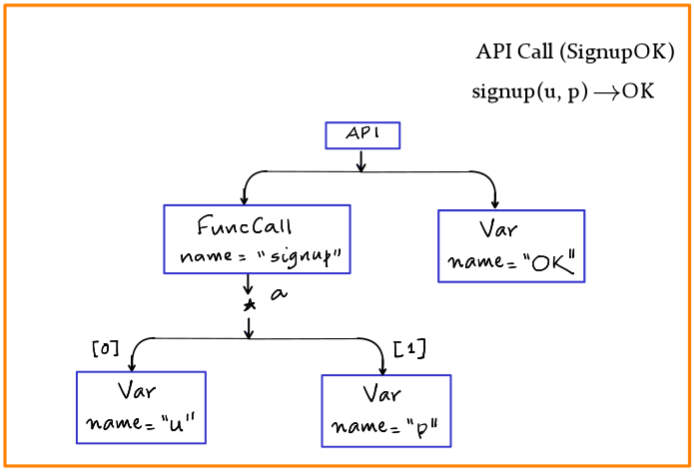


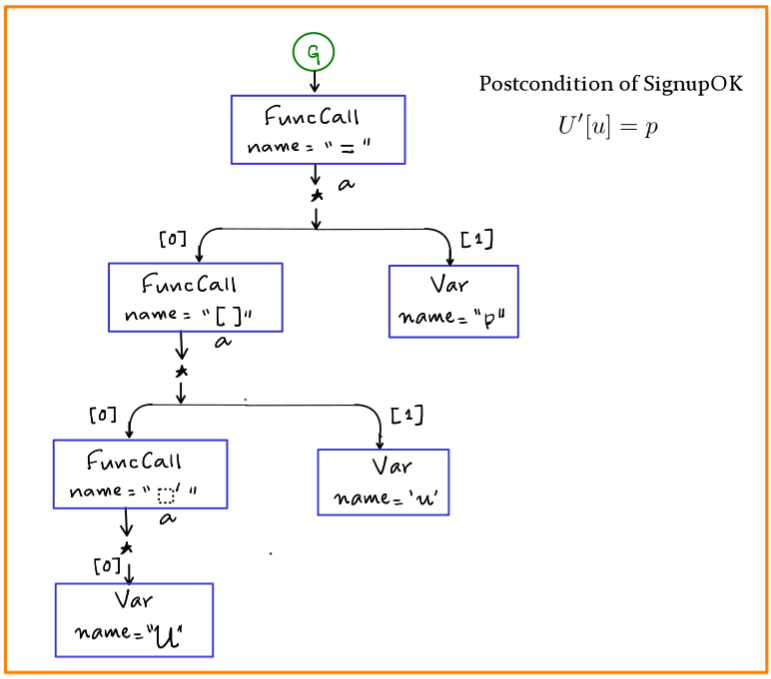


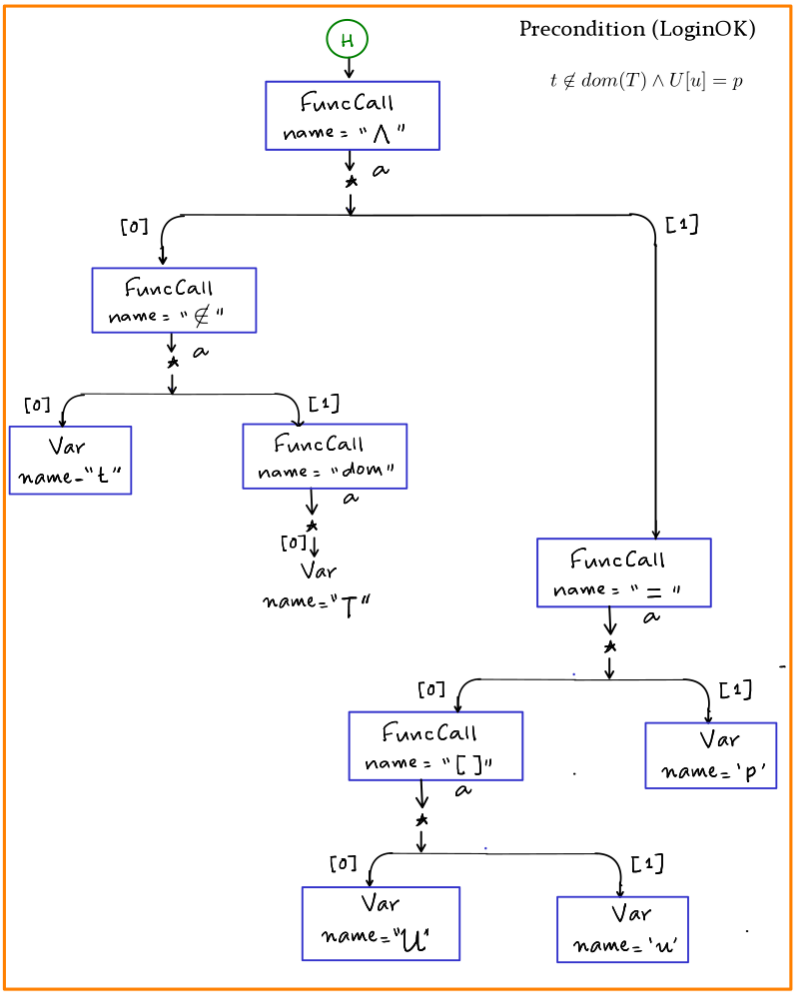

signup : string × string ⟶ HTTPResponse
C
name
r
String
value = "signup"
TypeExpr★
TypeConstr
name = "HTTPResponse"
[0]
[1]
TypeConstr
name = "string"
TypeConstr
name = "string"

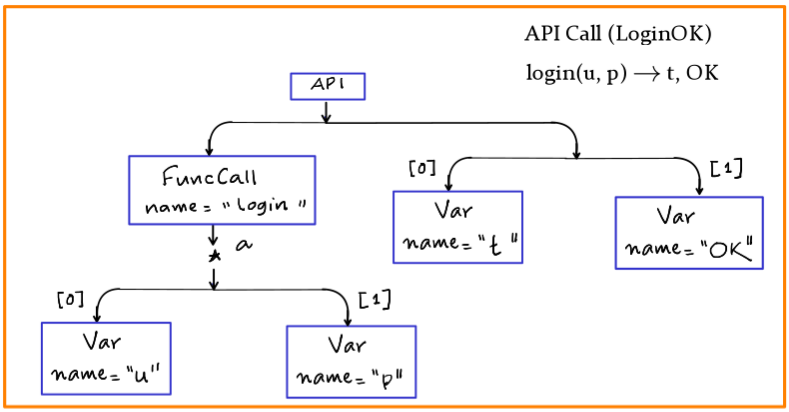

signup : string × string ⟶ Token × HTTPResponse
D
n
p
r
String
alue = "login"
TypeExpr★
[0]
[1]
TypeConstr
name = "string"
TypeConstr
name = "string"
TypeExpr★
[0]
[1]
TypeConstr
name = "string"
TypeConstr
name = "HTTPResponse"

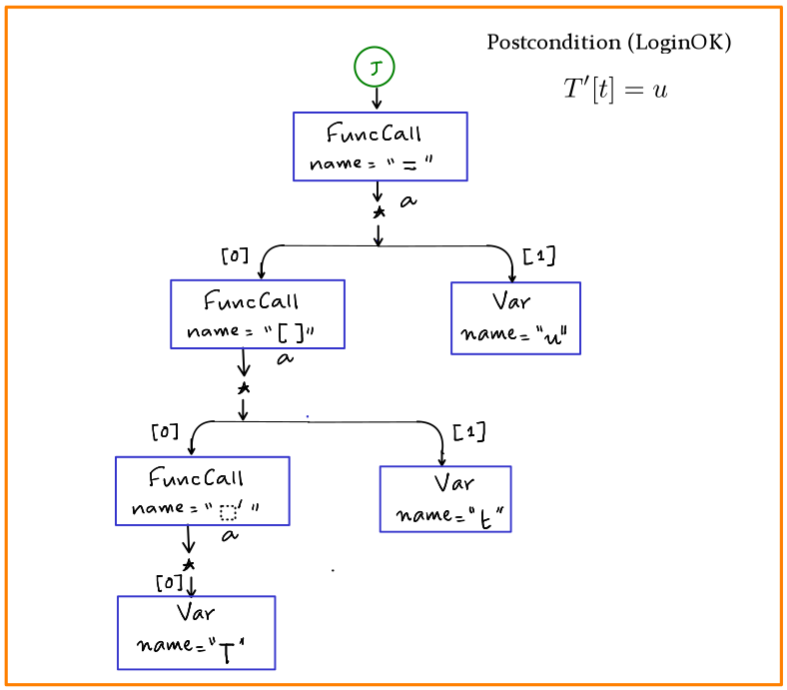

Precondition of SignupOK
$u \notin dom(U)$
E
FuncCall
name = "∉"
a
[0]
[1]
Var
name = 'u'
FuncCall
name = "dom"
a
[0]
Var
name = "u"


API Call (SignupOK)
signup(u, p) ⟶ OK
API
FuncCall
name = "signup"
Var
name = "OK"
a
[0]
[1]
Var
name = "u"
Var
name = "p"


FuncCall
name = " = "
a
[0]
[1]
FuncCall
name = " [ ] "
a
Var
name = "p"
[0]
[1]
FuncCall
name = " ⬚' "
a
Var
name = 'u'
[0]
Var
name = "u"
Postcondition of SignupOK
$U'[u] = p$


H
Precondition (LoginOK)
$t \notin dom(T) \wedge U[u] = p$
FuncCall
name = "∧"
a
[0]
[1]
FuncCall
name = "∉"
a
[0]
[1]
Var
name="t"
FuncCall
name = "dom"
a
[0]
Var
name="T"
FuncCall
name = "="
a
[0]
[1]
FuncCall
name = "[ ]"
a
Var
name='p'
[0]
[1]
Var
name="U"
Var
name='u'


API Call (LoginOK)
login(u, p) ⟶ t, OK
API
FuncCall
name = "login"
a
[0]
Var
name = "u"
[1]
Var
name = "p"
[0]
Var
name = "t"
[1]
Var
name = "OK"


Postcondition (LoginOK)
$T'[t] = u$
J
FuncCall
name = "="
a
[0]
FuncCall
name = "[]"
a
[0]
FuncCall
name = "□'"
a
[0]
Var
name = "T"
[1]
Var
name = "t"
[1]
Var
name = "u"