APPENDIX B – Grammars

This appendix describes the concrete textual syntax of the SysADL and π-ADL languages by using the Extended Backus-Naur Form (EBNF) meta-language, a notation for formally describing the context-free grammar of a language, in an adapted version for *xtext*. Section A.1 presents the notation elements used hereinafter; Section A.2 shows the production rules forming the SysADL grammar, whereas Section A.3 gives the production rules forming the π-ADL grammar.

B.1 Grammar notation

The EBNF/Xtext meta-language consists of terminal symbols, a sequence of characters forming an irreducible element of the language, and non-terminal production rules dictating how a syntactic element can be rightfully formed in terms of terminal symbols. Syntactic elements have names used in production rules, and they are distinguished from names and reserved words in the language. Furthermore, the EBNF/Xtext meta-language uses a set of meta-symbols summarized in Table 9

Table 9 – EBNF/Xtext meta-symbols

Meta-symbol	**Usage**
Colon (:)	Definition of production rule: A:B is read as A is defined as B
Pipe symbol (|)	Alternative choice between elements in production rule
Asterisk character (*)	Multiple occurences of element in production rule
Plus character (+)	At least occurence of element in production rule
Question mark character (?)	At most one occurrence of element in production rule
Equal character (=)	Represents a straightforward assignment, and is used for features which take only one element.
Plus equal character (+=)	Expects a multi-valued feature and adds the value on the right hand side to that feature, which is a list feature
Question mark equal character (?=)	Expects a feature of type `EBoolean` and sets it to true if the right hand side was consumed, independently from the concrete value of the right hand side
Brackets ([element])	Optional occurences of element in production rule
Single quotes ('text')	Represents keywords, that is a kind of terminal rule literals.
Parentheses	Element grouping or precedence

B.2 SysADL grammar

```
grammar org.sysadl.SysADL with org.eclipse.xtext.common.Terminals

grammar org.sysadl.SysADL with org.eclipse.xtext.common.Terminals

import "http://org.sysadl"
import "http://www.eclipse.org/emf/2002/Ecore" as ecore

Model returns Model:
  'Model' name=ID ';'
    packages+=Package
  ;

Package returns Package:
  'package' name=QualifiedName '{'
    ('import' imports+= QualifiedName ';')*
    (definitions+=ElementDef | definitions+=ArchitectureDef)*
  '}';

ElementDef returns ElementDef:
  DataTypeDef | ValueTypeDef | Enumeration | ComponentDef |
   ConnectorDef | CompositePortDef | SimplePortDef;

DataTypeDef returns DataTypeDef:
  'datatype'
  name=ID
  '{'
    ('attributes' ':' attributes+=TypeUse+)?
  '}';

ValueTypeDef returns ValueTypeDef:
  'value' 'type'
  name=ID
  '{'
    ('unit' '=' unit=[UnitDef])?
    ('dimension' '=' dimension=[DimensionDef])?
  '}';

Enumeration returns Enumeration:
  'enum'
  name=ID
```

```
  ('{'
    literals+=EnumLiteralValue (',' literals+=EnumLiteralValue)*
  '}');

ArchitectureDef returns ArchitectureDef:
  'architecture' 'def' name=ID
  '{'
    'ports' ':' ports+=PortUse*
    (composite=Configuration)
  '}'
;

ComponentDef returns ComponentDef:
  (isBoundary?='boundary')? 'component' 'def' name=ID
  '{'
    ('ports' ':' ports+=PortUse*)?
    (composite=Configuration)?
  '}'
  ;

ConnectorDef returns ConnectorDef:
  'connector' 'def' name=ID
  '{'
    ('participants' ':' ports+=PortUse_Reverse*)
    ('flows' ':' flows+=Flow*)?
  '}';

PortDef returns PortDef:
  CompositePortDef | SimplePortDef;

CompositePortDef returns CompositePortDef:
  'port' 'def' name=ID
  '{'
    'ports' ':' ports+=PortUse*
  '}';

SimplePortDef returns SimplePortDef:
  'port' 'def' name=ID
  '{'
    'flow' flowProperties=FlowProperty flowType=[TypeDef]
  '}';
```

```
TypeUse returns TypeUse:
  name=ID ':' definition=[TypeDef]
    ('{' '}' | ';' )
  ;

TypeDef returns TypeDef:
  DataTypeDef | ValueTypeDef | Enumeration;

PortUse returns PortUse:
  name=ID ':' definition=[PortDef]
  ('{' '}' | ';');

PortUse_Reverse returns PortUse:
  '~' name=ID ':' definition=[PortDef]
  ('{' '}' | ';');

Flow returns Flow:
  type=[TypeDef] 'from' source=[PortUse] 'to' destination=[PortUse]
  ;

enum FlowProperty returns FlowProperty:
  in = 'in' | out = 'out' | inout='inout';

Configuration returns Configuration:
  'configuration'
  '{'
    ('components' ':' components+=ComponentUse*)?
    ('connectors' ':' connectors+=ConnectorUse*)?
    ('delegations' ':' delegations+=Delegation*)?
  '}';

ConnectorUse returns ConnectorUse:
  name=ID ':' definition=[QualifiedName] 'bindings'
   bindings=[ConnectorBinding] ';';


ConnectorBinding returns ConnectorBinding:
  source=[QualifiedName] '=' destination=[QualifiedName];

ComponentUse returns ComponentUse:
  name=ID ':' definition=[QualifiedName]
  '{'
```

```
    ('using' 'ports' ':' ports+=PortUse*)?
  '}';

Delegation returns Delegation:
  source=[PortUse] 'to' destination=[PortUse]
;

EnumLiteralValue returns EnumLiteralValue:
  name=ID;

DimensionDef returns DimensionDef:
  'dimension'
  name=ID
  ('{' '}')?;

UnitDef returns UnitDef:
  'unit'
  name=ID
  ('{'
    ('dimension' '=' dimension=[DimensionDef])?
  '}')?;

QualifiedName:
  DotQualifiedName
  | ID
;

DotQualifiedName:
  ID ('.' ID)+
;

ActivityDef returns ActivityDef:
  'activity' 'def' name=ID ('(' inParameters+=Pin (','
   inParameters+=Pin)* ')')*
    (':' '('outParameters+=Pin (',' outParameters+=Pin)* ')')'{'
      ActivityBody
  '}';

ActivityBody returns ActivityBody:
  'body'
  '{'
    ('actions' ':' actions+=ActionUse*)?
```

```
    flows+=ActivityRelation*
    dataObjects+=DataObject*
  '}';

Pin returns Pin:
  {Pin}
  name=ID ':' (isFlow?='flow')? definition=[QualifiedName]
  ;

ActionUse returns ActionUse:
  name=ID ':' definition=[QualifiedName] ('{'
    ('using' 'pins' ':' (pinIn+=Pin ';')* )
  ';');

ActionDef returns ActionDef:
  {ActionDef}
  'action' 'def'
  name=ID '(' inParameters+=Pin ( "," inParameters+=Pin)* ')' ':'
   returnType=[QualifiedName]
  '{'
    'constraint' ':' constraints+=ConstraintUse*
    delegations+=ActivityDelegation*
  '}';


ActivityRelation returns ActivityRelation:
  ActivityDelegation | ActivityFlow;

ActivityDelegation returns ActivityDelegation:
  'delegate' source=[QualifiedName] 'to' (target=[QualifiedName]

ActivityFlow returns ActivityFlow:
  'flow'
  'from' source=[QualifiedName]
  'to' (target=[QualifiedName]
;

ConstraintUse returns ConstraintUse:
  kind=ConstraintKind (definition=[QualifiedName])
  ;

ActivityFlowable returns ActivityFlowable:
```

```
  ActionUse | DataStore | DataBuffer;

enum ConstraintKind returns ConstraintKind:
  preCondition = 'pre-condition' | postCondition = 'post-condition'
   | invariant = 'invariant';


ConstraintDef returns ConstraintDef:
  {ConstraintDef}
  'constraint'
  name=ID ('(' ( inParameters+=Pin (',' inParameters+=Pin)* )? ')')
   (':' '(' outParameters+=Pin (',' outParameters+=Pin)* ')')?
  '{'
    ('equation' '=' equation=Expression)?
  '}';

DataStore returns DataStore:
  'datastore'
  name=ID ':' type=[QualifiedName]
  ('{' '}')?;

DataBuffer returns DataBuffer:
  'databuffer'
  name=ID ':' type=[QualifiedName]
  '{' '}';
```

B.3 π-ADL grammar

```
grammar fr.irisa.archware.PiADL with
    org.eclipse.xtext.common.Terminals

generate piADL "http://www.irisa.fr/archware/PiADL"

ArchitectureDescription: //ok
  archElements+=ArchitecturalElement*
  archs+=Architecture+
  cbehavior=BehaviorDeclaration
;

ArchitecturalElement:
  Component | Connector
```

```
;

Connector://ok
  'connector' name=ID 'is'
  'abstraction()' '{'
    typeDecl+=TypeDeclaration*
    connections+=ConnectionDeclaration*
    protDecl=ProtocolDeclaration?
    behavior=BehaviorDeclaration
  '}'
;

Component://ok
  'component' name=ID 'is'
  'abstraction()' '{'
    typeDecl+=TypeDeclaration*
    connections+=ConnectionDeclaration*
    protDecl=ProtocolDeclaration?
    behavior=BehaviorDeclaration
  '}'
;

Architecture://ok
  'architecture' name=ID 'is'
  'abstraction()' '{'
    'behavior' 'is' '{'
      compose=Composition
    '}'
  '}'
;

TypeDeclaration:
  'type' name=ID 'is' type=ValueType
;

ConnectionDeclaration:
  'connection' name=ID 'is' direction=ConnectionMode '('
   type=ValueType ')'
;

ProtocolDeclaration:
  'protocol' 'is' '{'
```

```
    '('
    protocol+=ProtocolAction*
    ')' '*'
  '}'
;

ProtocolAction:
  '('* 'via' connectionName=ID action=Action type=ValueType ')'*
;


enum Action:
  send='send' | receive='receive'
;


enum ConnectionMode:
  in='in' | out='out'
;

BehaviorDeclaration:
  {BehaviorDeclaration}
  'behavior' 'is' '{'
    body+=BehaviorClause*
  '}'
;

BehaviorClause:
  ConnectionDeclaration | Composition

;

Composition:
  'compose' '{'
    clause+=(BehaviorClause | ElementInstantiation)+
    (('and' | 'and') clause+=(BehaviorClause |
   ElementInstantiation)+)+
  ('}' | '}') uc=UnificationClause?
;

UnificationClause:
  {UnificationClause}
```

```
  'where' '{'
    (unifications+=Unification)*
  '}'
;

ElementInstantiation:
  elementName=ID 'is' elementType=ID
  '(' (parameterName+=ID (',' parameterName+=ID)*)? ')'
;

Unification:
  fromc=ConnectionAccess 'unifies' toc=ConnectionAccess
;

ConnectionAccess:
  elementName=(ID | 'self') '::' connectionName=ID
;

Unobservable:
  'unobservable'
;

FunctionDeclaration:
  functionName=ID 'is' 'function' '('
  (parameters+=Parameter (',' parameters+=Parameter)*)? ')'
  (':' returnType=ValueType)? '{'
    block+=BehaviorClause*
  '}'
;

Parameter:
  name=ID ':' type=ValueType
;

ValueType:
  BaseType | ConstructedType | {TypeRef} idt=ID
;

BaseType:
  NaturalType | IntegerType | RealType | BooleanType | StringType |
   AnyType
;
```

```
NaturalType:
  {NaturalType} type='Natural'
;

IntegerType:
  {IntegerType} type='Integer'
;

RealType:
  {RealType} type='Real'
;

BooleanType:
  {BooleanType} type='Boolean'
;

StringType:
  {StringType} type='String'
;

AnyType:
  {AnyType} type='Any'
;

ConstructedType:
  View
;

View:
  'view''[' labt+=LabeledType (',' labt+=LabeledType)* ']'
;

LabeledType:
  label=ID ':' type=ValueType
;
```

APPENDIX C – Denotational Semantics for SysADL using π-ADL

This appendix presents the necessary elements to define the denotational semantics of SysADL in π-ADL. Section C.1 presents the normal form defined for input file with an architectural description in SysADL. Section C.2 describes the notation used in the definition of denotation semantics, which is finally presented in section C.3.

C.1 Normal Form

To simplify the definition and understanding of the denotational semantics of SysADL in π-ADL, we propose a normal form to place the elements in a textual architectural description in SysADL in order more favorable to the expression of the semantics. This section describes the normal form for architectural descriptions in SysADL.

C.1.1 Architectural elements order

The architectural elements are all enclosed in a unique package ordered as follows:

1. Basics ValueType
2. Created ValueType
3. Enumerations
4. DataTypes: Datatypes are ordered by dependency. A datatype appears only if all types of its attributes have seemed before.
5. PortDef: we consider only simple port because a sequence of simple ports can replace any composite port.
6. ConnectorDef: Similarly to ports, we consider only simple connectors because a sequence of simple connectors can replace any composite connector.
7. ComponentDef: Components are ordered by dependency like DataTypes. A component can only use connectors and other components that were previously defined. Therefore, in the proposed normal form, the composite component only appears after its used connectors and components are declared.
8. ArchitectureDef

C.1.2 Primitives ValueTypes

These types are considered primitive in SysADL. They are the first and appear in the same order generated by the SysADL Studio tool:

1. Int
2. Boolean
3. String
4. Real
5. Void

C.1.3 Created ValueType

Value Types no primitives created without subtypes, dimension, and unit.

C.1.4 Enumerations

There are no Enum changes. Anyway, the initial version of the transformation is not considered the value list of enums.

C.1.5 DataTypes

In the Normal Form, the datatype attributes are ordered according to their types:

1. Basics ValueType
2. Created ValueType
3. Enumerations
4. DataTypes: If the type X of an attribute depends on a type Y of another attribute on the same datatype, then the attribute of type Y appears before the attribute of type X.

C.1.6 PortDef

There is no change in the structure of PortDef. However, it is essential to note that this version of the study does not consider composite ports and that this does not reduce our scope since composite ports can be redefined as a succession of simple ports.

C.1.7 ConnectorDef

Similar to the definition of PortDef, there are no changes in the structure of connectors. This version of the study does not consider composite connectors also.

C.1.8 ComponentDef

In the Normal Form, the ComponentDef are ordered according to the below list:

1. Boundary Components
2. Simple Components
3. Composite Components: If Component X depends on Component Y, component Y is defined before component Y.

C.1.9 ArchitectureDef

There is no change in the ArchitectureDef structure. But, they must appear after all the ComponentDef.

C.1.10 Behavior viewpoint definitions

The activity must have the same name as the associated component and describe the behavior, adding the letters "AC" to the end.

C.1.11 Preprocessing

The transformation defined in our study should happen in its phases:

1. SysADL Model transformation in a SysADL Normal Model (preprocessing)
2. Model in Normal Form SysADL transformation to Model π-ADL.

The preprocessing step serves to adapt the SysADL model to the Normal Form now defined to help transform SysADL into π-ADL.

C.2 Notation

- Semantic Functions are defined with double brackets $[\![SysADL!C]\!]$ and map an element of a syntactic/semantic Class C to a corresponding syntactic/semantic element in π-ADL.
- Semantic functions of the form $[\![es]\!]^*$ represent a mapping of the correspondent semantic function $[\![\ ]\!]$ to all elements of the syntactic element sequence es.

- The expression $env[n]$ signifies a list in the *nth element* position defined in the *env* list defined in section C.3.2.

C.3 Formal Semantic

C.3.1 Function Signature

- $Env : Package \rightarrow \langle e_1, ..., e_{10} \rangle$, where

  $e1$: $seq(SysADL!ValueTypeDef)$

  $e2$: $seq(SysADL!Enumerate)$

  $e3$: $seq(SysADL!DataTypeDef)$

  $e4$: $seq(SysADL!TypeDef)$

  $e5$: $seq(SysADL!SimplePortDef)$

  $e6$: $seq(SysADL!ConnectorDef)$

  $e7$: $seq(SysADL!ComponentDef)$

  $e8$: $seq(SysADL!ActivityDef)$

  $e9$: $seq(SysADL!ActionDef)$

  $e10$: $seq(SysADL!ArchitectureDef)$

  $e11$: $seq(SysADL!ComponentUse)$

- $ValuetypesP : Package \rightarrow seq(SysADL!ValueTypeDef)$
- $Valuetypes : seq(SysADL!ElementDef) \bigcup seq(SysADL!ArquitectureDef) \rightarrow seq(SysADL!ValueTypeDef)$
- $Valuetype : SysADL!ElementDef \rightarrow SysADL!ValueTypeDef$
- $EnumtypesP : Package \rightarrow seq(SysADL!Enumeration)$
- $Enumtypes : seq(SysADL!ElementDef) \bigcup seq(SysADL!ArchitectureDef) \rightarrow seq(SysADL!Enumeration)$
- $Enumtype : SysADL!ElementDef \rightarrow SysADL!Enumeration$
- $DatatypesP : Package \rightarrow seq(SysADL!DataTypeDef)$
- $Datatypes : seq(SysADL!ElementDef) \bigcup seq(SysADL!ArchitectureDef) \rightarrow seq(SysADL!DataTypeDef)$
- $Datatype : SysADL!ElementDef \rightarrow SysADL!DataTypeDef$

- $TypedefsP : Package \rightarrow seq(SysADL!TypeDef)$
- $Typedefs : seq(SysADL!ElementDef) \bigcup seq(SysADL!ArchitectureDef) \rightarrow seq(SysADL!TypeDef)$
- $Typedef : SysADL!ElementDef \rightarrow SysADL!TypeDef$
- $SimplePortsP : Package \rightarrow seq(SysADL!SimplePortDef)$
- $SimplePorts : seq(SysADL!ElementDef) \bigcup seq(SysADL!ArchitectureDef) \rightarrow seq(SysADL!SimplePortdDef)$
- $SimplePort : SysADL!ElementDef \rightarrow SysADL!SimplePortDef$
- $ConnectorsP : Package \rightarrow seq(SysADL!ConnectorDef)$
- $Connectors : seq(SysADL!ElementDef) \bigcup seq(SysADL!ArchitectureDef) \rightarrow seq(SysADL!ConnectorDef)$
- $Connector : SysADL!ElementDef \rightarrow SysADL!ConnectorDef$
- $ComponentsP : Package \rightarrow seq(SysADL!ComponentDef)$
- $Components : seq(SysADL!ElementDef) \bigcup seq(SysADL!ArchitectureDef) \rightarrow seq(SysADL!ComponentDef)$
- $Component : SysADL!ElementDef \rightarrow SysADL!ComponentDef$
- $ActivitiesP : Package \rightarrow seq(SysADL!ActivityDef)$
- $Activities : seq(SysADL!ElementDef) \bigcup seq(SysADL!ArchitectureDef) \rightarrow seq(SysADL!ActivityDef)$
- $Activitie : SysADL!ElementDef \rightarrow SysADL!ActivityDef$
- $ActionsP : Package \rightarrow seq(SysADL!ActionDef)$
- $Actions : seq(SysADL!ElementDef) \bigcup seq(SysADL!ArchitectureDef) \rightarrow seq(SysADL!ActionDef)$
- $Action : SysADL!ElementDef \rightarrow SysADL!ActionDef$
- $ArchsP : Package \rightarrow seq(SysADL!ArchitectureDef)$
- $Archs : seq(SysADL!ElementDef) \bigcup seq(SysADL!ArchitectureDef) \rightarrow seq(SysADL!ArchitectureDef)$
- $Arch : SysADL!ArchitectureDef \rightarrow seq(SysADL!ArchitectureDef)$

- $ComponentsUseP : Package \rightarrow seq(SysADL!ComponentUse)$
- $ComponentsUse : seq(SysADL!ElementDef) \bigcup seq(SysADL!ArchitectureDef) \rightarrow seq(SysADL!ComponentUse)$
- $ComponentUseF : SysADL!ElementDef \rightarrow seq(SysADL!ComponentUse)$
- $ComponentsUseExtractor : SysADL!Configuration \rightarrow seq(SysADL!ComponentUse)$
- $[\![ElementS]\!] : SysADL!ElementS \rightarrow PIADL!ElementP$
  - $[\![SysADL!Model]\!] : SysADL!Model \rightarrow PIADL!ArchitectureDescription$
  - $[\![SysADL!ConnectorDef]\!] : SysADL!ConnectorDef \rightarrow PIADL!Connector$
  - $[\![SysADL!ComponentDef]\!] : SysADL!ComponentDef \rightarrow PIADL!Component$
  - $[\![SysADL!ArchitectureDef]\!] : SysADL!ArchitectureDef \rightarrow PIADL!Architecture$
  - $[\![SysADL!PortUse\_Reverse]\!] : SysADL!PortUse\_Reverse \rightarrow PIADL!ConnectionDeclaration$
  - $[\![SysADL!PortUse]\!] : SysADL!PortUse \rightarrow PIADL!ConnectionDeclaration$
  - $[\![SysADL!TypeDef]\!] : SysADL!TypeDef \rightarrow PIADL!TypeDeclaration$
  - $[\![SysADL!TypeUse]\!] : SysADL!TypeUse \rightarrow PIADL!LabeledType$
  - $[\![SysADL!QualifiedName]\!] : SysADL!QualifiedName \rightarrow String$
- $getNameTypeDefFlows : seq(SysADL!Flow) \rightarrow SysADL!QualifiedName$
- $getLstNameTDefPUse : seq(SysADL!PortUse) \rightarrow seq(SysADL!QualifiedName)$
- $getTypeDef : SysADL!QualifiedName \rightarrow SysADL!TypeDef$
- $getDDependency : SysADL!TypeDef \rightarrow seq(SysADL!QualifiedName)$
- $getTypeDefDep : seq(SysADL!QualifiedName) \times seq(SysADL!TypeDef) \rightarrow seq(SysADL!TypeDef)$
- $makeInjSeq : seq(SysADL!QualifiedName) \rightarrow seq(SysADL!QualifiedName)$
- $distCat : seq(seq(SysADL!QualifiedName)) \rightarrow seq(SysADL!QualifiedName)$
- $getAttTypes : seq(SysADL!TypeUse) \rightarrow seq(SysADL!QualifiedName)$

- $removeQName : SysADL!QualifiedName \times seq(QualifiedName) \rightarrow seq(SysADL!QualifiedName)$
- $getDependencies : SysADL!QualifiedName \times seq(SysADL!TypeDef) \rightarrow seq(SysADL!TypeDef)$
- $getDependLst : seq(SysADL!QualifiedName) \times seq(SysADL!TypeDef) \rightarrow seq(SysADL!TypeDef)$
- $setProtocolDeclaration : SysADL!PortUse\_Reverse \rightarrow PIADL!ProtocolAction$
- $setProtocolDeclarationPU : SysADL!PortUse \rightarrow PIADL!ProtocolAction$
- $setCompLstElementInstantiation : seq(SysADL!ComponentUse) \rightarrow seq(PIADL!ElementInstantiation)$
- $setConLstElementInstantiation : seq(SysADL!ConnectorUse) \rightarrow seq(PIADL!ElementInstantiation)$
- $getPortDef : SysADL!QualifiedName \rightarrow SysADL!SimplePortDef$
- $getDirection : SysADL!SimplePortDef \rightarrow String$
- $getAction : SysADL!SimplePortDef \rightarrow String$
- $setConUnification : seq(SysADL!ConnectorUse) \rightarrow seq(PIADL!Unifications)$
- $setConnectionAccesFrom : SysADL!ConnectorBinding \rightarrow PIADL!ConnectionAccess$
- $setConnectionAccesTo : SysADL!ConnectorBinding \rightarrow PIADL!ConnectionAccess$
- $setCAccessFromCon : SysADL!ConnectorDef \rightarrow PIADL!ConnectionAccess$
- $setCAccessToCon : SysADL!ConnectorDef \rightarrow PIADL!ConnectionAccess$
- $getConnectorDef : SysADL!QualifiedName \times seq(SysADL!ConnectorDef) \rightarrow SysADL!ConnectorDef$
- $getComponentUsePU : SysADL!QualifiedName \times seq(SysADL!ComponentUse) \rightarrow SysADL!QualifiedName$
- $getPortUseReverseOut : SysADL!ConnectorUse \rightarrow SysADL!QualifiedName$
- $getPortUseReverseIn : SysADL!ConnectorUse \rightarrow SysADL!QualifiedName$
- $setDelUnification : SysADL!Delegation \rightarrow PIADL!Unifications$

- $setBehavior : SysADL!ActivityDef \times seq(SysADL!ActionDef) \rightarrow seq(PIADL!FunctionDeclaration)$
- $setFunctions : seq(SysADL!ActionUse) \times seq(SysADL!ActionDef) \rightarrow seq(PIADL!FunctionDeclaration)$
- $getActionDef : SysADL!ActionUse \times seq(SysADL!ActionDef) \rightarrow SysADL!ActionDef$
- $setFunction : SysADL!ActionUse \times SysADL!ActionDef \rightarrow PIADL!FunctionDeclaration$
- $setInParam : seq(SysADL!Pin) \rightarrow seq(PIADL!Parameters)$
- $getActivity : SysADL!QualifiedName \rightarrow SysADL!ActivityDef$

C.3.2 Enviroment

$$
\begin{aligned}
&\underline{\text{Env}}(Package) = \\
&\quad \langle \underline{\text{ValuetypesP}}(Package), \\
&\quad \underline{\text{EnumtypesP}}(Package), \\
&\quad \underline{\text{DatatypesP}}(Package), \\
&\quad \underline{\text{TypedefsP}}(Package), \\
&\quad \underline{\text{SimplePortsP}}(Package), \\
&\quad \underline{\text{ConnectorsP}}(Package), \\
&\quad \underline{\text{ComponentsP}}(Package), \\
&\quad \underline{\text{ActivitiesP}}(Package), \\
&\quad \underline{\text{ActionsP}}(Package), \\
&\quad \underline{\text{ArchsP}}(Package), \\
&\quad \text{ComponentsUseP}(Package)\rangle
\end{aligned}
$$

$$
\begin{aligned}
&//\text{ValueTypes} \\
&\underline{\text{ValuetypesP}}(\text{'package'}\ QualifiedName\ \text{'\{'}(\text{'import'}\ QualifiedName\ \text{';'})^* \\
&\quad (ElementDef \mid ArchitectureDef)^*\text{'\}'}) = \underline{\text{Valuetypes}}((ElementDef \mid \\
&\quad ArchitectureDef)^*) \\
&//\text{---} \\
&\underline{\text{Valuetypes}}(\epsilon) = \langle\rangle \\
&\underline{\text{Valuetypes}}(ElementDef\ (ElementDef \mid ArchitectureDef)^*) = \\
&\quad \underline{\text{Valuetype}}(ElementDef) \frown \underline{\text{Valuetypes}}((ElementDef \mid ArchitectureDef)^*) \\
&\underline{\text{Valuetypes}}(ArchitectureDef\ (ElementDef \mid ArchitectureDef)^*) = \\
&\quad \underline{\text{Valuetypes}}((ElementDef \mid ArchitectureDef)^*) \\
&//\text{---} \\
&\underline{\text{Valuetype}}(ValueTypeDef) = \langle ValueTypeDef\rangle \\
&\underline{\text{Valuetype}}(Enumeration \mid DataTypeDef \mid SimplePortDef \mid ConnectorDef \\
&\quad \mid ComponentDef \mid \text{ActivityDef} \mid \text{ActionDef}) = \langle\rangle
\end{aligned}
$$

$$
//\text{EnumTypes}
$$

```
EnumtypesP('package' QualifiedName '{'('import' QualifiedName ';')*
    (ElementDef | ArchitectureDef)*'}') = Enumtypes((ElementDef |
    ArchitectureDef)*)
//-----------------------------------------------------------
Enumtypes(ϵ) = ⟨⟩
Enumtypes(ElementDef (ElementDef | ArchitectureDef)*) =
    Enumtype(ElementDef) ⌢Enumtypes((ElementDef | ArchitectureDef)*)
Enumtypes(ArchitectureDef (ElementDef | ArchitectureDef)*) =
    Enumtypes((ElementDef | ArchitectureDef)*)
//-----------------------------------------------------------
Enumtype(Enumeration) = ⟨Enumeration⟩
Enumtype(ValueTypeDef | DataTypeDef | SimplePortDef | ConnectorDef |
    ComponentDef | ActivityDef | ActionDef) = ⟨⟩

//DataTypes
DatatypesP('package' QualifiedName '{'('import' QualifiedName ';')*
    (ElementDef | ArchitectureDef)*'}') = Datatypes((ElementDef |
    ArchitectureDef)*)
//-----------------------------------------------------------
Datatypes(ϵ) = ⟨⟩
Datatypes(ElementDef (ElementDef | ArchitectureDef)*) =
    Datatype(ElementDef) ⌢Datatypes((ElementDef | ArchitectureDef)*)
Datatypes(ArchitectureDef (ElementDef | ArchitectureDef)*) =
   Datatypes((ElementDef | ArchitectureDef)*)
//-----------------------------------------------------------
Datatype(DataTypeDef) = ⟨DataTypeDef⟩
Datatype(ValueTypeDef | Enumeration | SimplePortDef | ConnectorDef |
    ComponentDef | ActivityDef | ActionDef) = ⟨⟩

//TypeDefs
TypedefsP('package' QualifiedName '{'('import' QualifiedName ';')*
    (ElementDef | ArchitectureDef)*'}') = Typedefs((ElementDef |
    ArchitectureDef)*)
//-----------------------------------------------------------
Typedefs(ϵ) = ⟨⟩
Typedefs(ElementDef (ElementDef | ArchitectureDef)*) =
    Typedef(ElementDef) ⌢Typedefs((ElementDef | ArchitectureDef)*)
Typedefs(ArchitectureDef (ElementDef | ArchitectureDef)*) =
   Typedefs((ElementDef | ArchitectureDef)*)
//-----------------------------------------------------------
Typedef(ValueType) = ⟨ValueType⟩
Typedef(Enumeration) = ⟨Enumeration⟩
Typedef(DataTypeDef) = ⟨DataTypeDef⟩
Typedefs(SimplePortDef | ConnectorDef | ComponentDef | ActivityDef |
    ActionDef) = ⟨⟩

//SimplePorts
```

```
SimplePortsP('package' QualifiedName '{'('import' QualifiedName ';')*
    (ElementDef | ArchitectureDef)*'}') = SimplePorts((ElementDef |
    ArchitectureDef)*)
//-----------------------------------------------------
SimplePorts(ϵ) = ⟨⟩
SimplePorts(ElementDef (ElementDef | ArchitectureDef)*) =
    SimplePort(ElementDef) ⌢SimplePorts((ElementDef | ArchitectureDef)*)
SimplePorts(ArchitectureDef (ElementDef | ArchitectureDef)*) =
    SimplePorts((ElementDef | ArchitectureDef)*)
//-----------------------------------------------------
SimplePort(SimplePortDef) = ⟨SimplePortDef⟩
SimplePort(ValueTypeDef | Enumeration | DataTypeDef | ConnectorDef |
    ComponentDef | ActivityDef | ActionDef) = ⟨⟩

//Connectors
ConnectorsP('package' QualifiedName '{'('import' QualifiedName ';')*
    (ElementDef | ArchitectureDef)*'}') = Connectors((ElementDef |
    ArchitectureDef)*)
//-----------------------------------------------------
Connectors(ϵ) = ⟨⟩
Connectors(ElementDef (ElementDef | ArchitectureDef)*) =
    Connector(ElementDef) ⌢Connectors((ElementDef | ArchitectureDef)*)
Connectors(ArchitectureDef (ElementDef | ArchitectureDef)*) =
    Connectors((ElementDef | ArchitectureDef)*)
//-----------------------------------------------------
Connector(ConnectorDef) = ⟨ConnectorDef⟩
Connector(ValueTypeDef | Enumeration | DataTypeDef | SimplePortDef |
    ComponentDef | ActivityDef | ActionDef) = ⟨⟩

//Components
ComponentsP('package' QualifiedName '{'('import' QualifiedName ';')*
    (ElementDef | ArchitectureDef)*'}') = Components((ElementDef |
    ArchitectureDef)*)
//-----------------------------------------------------
Components(ϵ) = ⟨⟩
Components(ElementDef (ElementDef | ArchitectureDef)*) =
    Component(ElementDef) ⌢Components((ElementDef | ArchitectureDef)*)
Components(ArchitectureDef (ElementDef | ArchitectureDef)*) =
    Components((ElementDef | ArchitectureDef)*)
//-----------------------------------------------------
Component(ComponentDef) = ⟨ComponentDef⟩
Component(ValueTypeDef | Enumeration | DataTypeDef | SimplePortDef |
    ConnectorDef | ActivityDef | ActionDef) = ⟨⟩

//ActivityDef
ActivitiesP('package' QualifiedName '{'('import' QualifiedName ';')*
    (ElementDef | ArchitectureDef)*'}') = Activities((ElementDef |
```

```
    ArchitectureDef)*)
//-----------------------------------------------------------
Activities(ϵ) = ⟨⟩
Activities(ElementDef (ElementDef | ArchitectureDef)*) =
    Activity(ElementDef) ⌢Activities((ElementDef | ArchitectureDef)*)
Activities(ArchitectureDef (ElementDef | ArchitectureDef)*) =
    Activities((ElementDef | ArchitectureDef)*)
//-----------------------------------------------------------
Activity(ActivityDef) = ⟨ActivityDef⟩
Activity(ValueTypeDef | Enumeration | DataTypeDef | SimplePortDef |
    ConnectorDef | ComponentDef | ActionDef) = ⟨⟩

//ActionDef
ActionsP('package' QualifiedName '{'('import' QualifiedName ';')*
    (ElementDef | ArchitectureDef)*'}') = Actions((ElementDef |
    ArchitectureDef)*)
//-----------------------------------------------------------
Actions(ϵ) = ⟨⟩
Actions(ElementDef (ElementDef | ArchitectureDef)*) =
    Action(ElementDef) ⌢Actions((ElementDef | ArchitectureDef)*)
Actions(ArchitectureDef (ElementDef | ArchitectureDef)*) =
    Actions((ElementDef | ArchitectureDef)*)
//-----------------------------------------------------------
Action(ActionDef) = ⟨ActionDef⟩
Action(ValueTypeDef | Enumeration | DataTypeDef | SimplePortDef |
    ConnectorDef | ComponentDef | ActivityDef) = ⟨⟩

//Archs
ArchsP('package' QualifiedName '{'('import' QualifiedName ';')*
    (ElementDef | ArchitectureDef)*'}') = Archs((ElementDef |
    ArchitectureDef)*)
//-----------------------------------------------------------
Archs(ϵ) = ⟨⟩
Archs(ArchitectureDef (ElementDef | ArchitectureDef)*) =
    Arch{ArchitectureDef} ⌢Archs((ElementDef | ArchitectureDef)*)
Archs(ElementDef (ElementDef | ArchitectureDef)*) =
    Archs((ElementDef | ArchitectureDef)*)
//-----------------------------------------------------------
Arch(ArchitectureDef) = ⟨ArchitectureDef⟩

//ComponentsUse
ComponentsUseP('package' QualifiedName '{'('import' QualifiedName ';')*
    (ElementDef | ArchitectureDef)*'}') = ComponentsUse((ElementDef |
    ArchitectureDef)*)
//-----------------------------------------------------------
ComponentsUse(ϵ) = ⟨⟩
ComponentsUse(ElementDef (ElementDef | ArchitectureDef)*) =
```

```
    ComponentUseF(ElementDef) ⌢ComponentsUse((ElementDef |
    ArchitectureDef)*)
ComponentsUse(ArchitectureDef (ElementDef | ArchitectureDef)*) =
    ComponentsUse((ElementDef | ArchitectureDef)*)
//-------------------------------------------------------
ComponentUseF(ValueTypeDef | Enumeration | DataTypeDef | SimplePortDef |
    ConnectorDef | ActivityDef | ActionDef) = ⟨⟩
ComponentUseF('boundary component def' QualifiedName '{'PortUse*'}') = ⟨ \;
    ⟩
ComponentUseF('component def' QualifiedName '{'PortUse*'}') = ⟨ \; ⟩
ComponentUseF('component def' QualifiedName '{'PortUse* Configuration'}')
    = ComponentsUseExtractor(Configuration)
//-------------------------------------------------------
ComponentsUseExtractor('configuration { }') = ⟨⟩
ComponentsUseExtractor('configuration {components :' ComponentUse* '}') =
    ⟨ComponentUse*⟩
ComponentsUseExtractor('configuration {connectors :' ConnectorUse* '}') = ⟨⟩
ComponentsUseExtractor('configuration {delegations :' Delegation* '}') = ⟨⟩
ComponentsUseExtractor('configuration {components :' ComponentUse*
    'connetors :' ConnectorUse* '}') = ⟨ComponentUse*⟩
ComponentsUseExtractor('configuration {components :' ComponentUse*
    'delegations :' Delegation* '}') = ⟨ComponentUse*⟩
ComponentsUseExtractor('configuration {connectors :' ConnectorUse*
    'delegations :' Delegation* '}') = ⟨⟩
ComponentsUseExtractor('configuration {components :' ComponentUse*
    'connectors :' ConnectorUse* 'delegations :' Delegation* '}') =
    ⟨ComponentUse*⟩
```

C.3.3 Structural Elements

```
//Model and Package
[['Model' ID ';' Package]] ⟨⟩ =
    let env = Env(Package)
    in
        [[env[6]]]* env
        [[env[7]]]* env
        [[env[10]]]* env
        'behavior is {unobservable}'
    end

//SYSADL!ConnectorDef to PIADL!Connector
[['connector def' ID '{' ('participants :' PortUse_Reverse*)
('flows :' Flow*)'}']] env =
```

```
    'connector' ID 'is abstraction() {'
        ⟦
   getTypeDefDep(getDependencies(getNameTypeDefFlows(Flow*),
   env[4]), env[4])⟧* env
        ⟦PortUse_Reverse⟧*env
        'protocol is {
   ('setProtocolDeclaration(PortUse_Reverse*)')*}'
        'behavior is {unobservable}'
    '}'

//SYSADL!ComponentDef to PIADL!Component
⟦'boundary component def' ID '{' ('ports :' PortUse*)'}'⟧ env =
    'component boundary'ID'is abstraction() {'
        ⟦
   getTypeDefDep(getDependLst(getLstNameTDefPUse(PortUse*),
   env[4]), env[4])⟧* env
        ⟦PortUse⟧*env
        'protocol is { ('setProtocolDeclarationPU(PortUse*)')*}'
        'behavior is {unobservable}'
    '}'

⟦'component def' ID '{' ('ports :' PortUse*)'}'⟧ env =
    'component' ID 'is abstraction() {'
        ⟦
   getTypeDefDep(getDependLst(getLstNameTDefPUse(PortUse*),
   env[4]), env[4])⟧* env
        ⟦PortUse⟧*env
        'protocol is { ('setProtocolDeclarationPU(PortUse*)')*}'
        'behavior is {'setBehavior(getActivity(ID), env[8])'}'
    '}'

⟦'component def'ID'{' ('ports :' PortUse*)Configuration'}'⟧ env =
    'component' ID 'is abstraction() {'
        ⟦
   getTypeDefDep(getDependLst(getLstNameTDefPUse(PortUse*),
   env[4]), env[4])⟧* env
        ⟦PortUse⟧*env
        'protocol is { ('setProtocolDeclarationPU(PortUse*)')*}'
        'behavior is {' ⟦Configuration⟧'}'
```

```
    '}'

//SYSADL!ArchitectureDef to PIADL!Architecture
[['architecture def'ID'{' ('ports :' PortUse*) Configuration'}']]
   env =
    'architecture' ID 'is abstraction() {'
        'behavior is {' [[Configuration]]'}'
    '}'

//SYSADL!TypeDef to PIADL!TypeDeclaration
[[ϵ]] = ϵ
[['value type' ID '{ }']]env = 'type' ID 'is Any'
[['enum' ID '{ }']]env = 'type' ID 'is Any'
[['datatype' ID '{'('attributes :' TypeUse*)?'}']]env =
    'type' ID 'is view ['[[TypeUse]]*env ']'

//SYSADL!TypeUse to PIADL!LabeledType}
[[ID':'QualifiedName';']]env = ID':' [[QualifiedName]]

//SYSADL!PortUse_Reverse to PIADL!Connection Declaration}
[['~'ID':' QualifiedName'{''}']]env = 'connection' ID 'is'
   getDirection(getPortDef(QualifiedName, env[5])) '(' [[
   QualifiedName]]')'

//SYSADL!PortUse to PIADL!Connection Declaration}
[[ID':' QualifiedName'{''}']]env = 'connection' ID 'is'
   getDirection(getPortDef(QualifiedName, env[5])) '(' [[
   QualifiedName]]')'

//SYSADL!QualifiedName to ECORE!String
[[ID]]env = ID

//SYSADL!Configuration to PIADL!BehaviorClause(Composition)
[['configuration{components:' ComponentUse+ 'connectors:'
   ConnectorUse+ 'delegations:'Delegation*'}']]env =
'compose {'
setCompLstElementInstantiation(ComponentUse+)
setConLstElementInstantiation(ConnectorUse+)'} where {'
  setConUnification(ConnectorUse+, env[6], env[11])
```

```
  setDelUnification(Delegation*)
'}'

⟦'configuration{components:' ComponentUse+ 'connectors:'
   ConnectorUse+'}'⟧env =
'compose {'
setCompLstElementInstantiation(ComponentUse+)
setConLstElementInstantiation(ConnectorUse+)'} where {'
  setConUnification(ConnectorUse+, env[6], env[11])
'}'
```

C.3.4 Auxiliary Functions

```
//-------------------------------------------------------------
getPortDef(QualifiedName, {'port def' ID '{'flow FlowProperty
   QualifiedName'}'}) = 'port def' ID '{'flow FlowProperty
   QualifiedName'}'
getPortDef(QualifiedName, {'port def' ID '{'flow FlowProperty
   QualifiedName2'}'} ∪ pdefSet) = getPortDef(QualifiedName,
   pdefSet)
getPortDef(QualifiedName, {'port def' ID '{'flow FlowProperty
   QualifiedName'}'} ∪ pdefSet) = 'port def' ID '{'flow
   FlowProperty QualifiedName'}'

//-------------------------------------------------------------
getTypeDef(ID: QualifiedName, ⟨⟩) = ⟨⟩
getTypeDef(ID: QualifiedName, ⟨'value' 'type' ID2 '{' '}'⟩) = ⟨⟩
getTypeDef(ID: QualifiedName, ⟨'enum' ID2 '{' '}'⟩) = ⟨⟩
getTypeDef(ID: QualifiedName, ⟨'datatype' ID2 '{'('attributes'
   ':' TypeUse*)?'}'⟩) = ⟨⟩
getTypeDef(ID: QualifiedName, ⟨'value' 'type' ID '{' '}'⟩) = ⟨
   'value' 'type' ID '{' '}'⟩
getTypeDef(ID: QualifiedName, ⟨'enum' ID '{' '}'⟩) = ⟨'enum' ID
   '{' '}'⟩
getTypeDef(ID: QualifiedName, ⟨'datatype' ID '{'('attributes'
   ':' TypeUse*)?'}'⟩) = ⟨'datatype' ID '{'('attributes' ':'
   TypeUse*)?'}'⟩
```

```
getTypeDef(ID: QualifiedName, ⟨'value' 'type' ID2 '{'
    '}'⟩⌢typeDefsList) = getTypeDef(ID: QualifiedName,
    typeDefsList)
getTypeDef(ID: QualifiedName, ⟨'enum' ID2 '{'
    '}'⟩⌢typeDefsList) = getTypeDef(ID: QualifiedName,
    typeDefsList)
getTypeDef(ID: QualifiedName, ⟨'datatype' ID2 '{'('attributes'
    ':' TypeUse*)?'}'⟩⌢typeDefsList) = getTypeDef(ID:
    QualifiedName, typeDefsList)
getTypeDef(ID: QualifiedName, ⟨'value' 'type' ID '{'
    '}'⟩⌢typeDefsList) = ⟨'value' 'type' ID '{' '}'⟩
getTypeDef(ID: QualifiedName, ⟨'enum' ID '{' '}'⟩⌢typeDefsList)
    = ⟨'enum' ID '{' '}'⟩
getTypeDef(ID: QualifiedName, ⟨'datatype' ID '{'('attributes'
    ':' TypeUse*)?'}'⟩⌢typeDefsList) = ⟨'datatype' ID
    '{'('attributes' ':' TypeUse*)?'}'⟩

//-------------------------------------------------------------
getDDependency (⟨'value' 'type' ID2 '{' '}'⟩) = ⟨⟩
getDDependency (⟨'enum' ID2 '{' '}'⟩) = ⟨⟩
getDDependency (⟨'datatype' ID2 '{'('attributes' ':'
    TypeUse*)?'}'⟩) = getAttTypes (TypeUse*)

//-------------------------------------------------------------
getAttTypes (⟨ID ':' QualifiedName';'⟩ ) = ⟨QualifiedName⟩
getAttTypes (⟨ID ':' QualifiedName';'⟩⌢typeUsesList ) = ⟨
    QualifiedName⟩ ⌢getAttTypes(typeUsesList)

//-------------------------------------------------------------
distCat(⟨⟩) = ⟨⟩
distCat(⟨qNameList⟩⌢qNamesList_List) = qNameList ⌢
    distCat(qNamesList_List)

//-------------------------------------------------------------
makeInjSeq(⟨⟩) = ⟨⟩
makeInjSeq(⟨QualifiedName⟩ ⌢qNamesList) = ⟨QualifiedName⟩ ⌢
    remove(QualifiedName,qNamesList)

//-------------------------------------------------------------
```

```
removeQName(qd: QualifiedName,⟨⟩) = ⟨⟩
removeQName(qd: QualifiedName,⟨qd: QualifiedName⟩⌢qNamesList) =
    remove(qd, qNamesList)
removeQName(qd: QualifiedName,⟨qd2: QualifiedName⟩⌢qNamesList) = ⟨
    qd2⟩ ⌢remove(qd, qNamesList)

//-----------------------------------------------------------
getDependencies(QualifiedName, typeDefsList) =
     makeInjSeq(distCat(⟨getDependencies(td,typeDefsList) | td ⟨-
    getDDependency(QualifiedName)⟩))⌢⟨QualifiedName⟩)

//-----------------------------------------------------------
getDependLst(⟨QualifiedName⟩, typeDefsList) =
     makeInjSeq(distCat(⟨getDependencies(td,typeDefsList) | td ⟨-
    getDDependency(QualifiedName)⟩))⌢⟨QualifiedName⟩)

getDependLst(⟨QualifiedName⟩⌢qnList, typeDefsList) =
     makeInjSeq(distCat(⟨getDependencies(td,typeDefsList) | td ⟨-
    getDDependency(QualifiedName)⟩))⌢⟨QualifiedName⟩) ⌢
    getDependLst(qnList, typeDefsList)

//-----------------------------------------------------------
getTypeDefDep(⟨QualifiedName⟩, typeDefsList) = ⟨
    getTypeDef(QualifiedName, typeDefsList)⟩
getTypeDefDep(⟨QualifiedName⟩ ⌢qNameList, typeDefsList) = ⟨
    getTypeDef(QualifiedName, typeDefsList)⟩ ⌢
    getTypeDef(typeDefsList, typeDefsList)⟩

//-----------------------------------------------------------
getNameTypeDefFlows(⟨qf: QualifiedName 'from' qf1: QualifiedName
    'to' qf2: QualifiedName⟩ ⌢flowsList) = qf

//-----------------------------------------------------------
getLstNameTDefPUse(⟨ID ':' QualifiedName';'⟩) = ⟨QualifiedName⟩
getLstNameTDefPUse(⟨ID ':' QualifiedName';'⟩⌢puList) =
⟨QualifiedName⟩⌢getNameTypeDefPortUse(puList)

//-----------------------------------------------------------
setProtocolDeclaration('~'ID':' QualifiedName'{''}') =
```

```
    'via' ID getAction(getPortDef(QualifiedName,
  env[5]))[[QualifiedName]]

//-------------------------------------------------------
setProtocolDeclarationPU(ID':' QualifiedName'{''}') =
    'via' ID getAction(getPortDef(QualifiedName,
  env[5]))[[QualifiedName]]

//-------------------------------------------------------
getDirection('port def' ID '{'flow 'in' QualifiedName'}') = 'in'
getDirection('port def' ID '{'flow 'out' QualifiedName'}') =
  'out'

//-------------------------------------------------------
getAction('port def' ID '{'flow 'in' QualifiedName'}') =
  'receive'
getAction('port def' ID '{'flow 'out' QualifiedName'}') = 'send'

//-------------------------------------------------------
setBehavior('activity def' ID '('Pin*'):('Pin*'){body {actions
  :' ActionUse* ActivityRelation* DataObject*'}}', env[9]) =
  setFunctions(ActionUse*, env[9])

//-------------------------------------------------------
setFunctions(⟨⟩, env[9]) = ϵ
setFunctions(⟨ID':'QualifiedName('{using
  pins:'Pin';'*'}'⟩⌢listActions, env[9]) =
  setFunction(ID':'QualifiedName('{using pins:'Pin';'*'}',
  getActionDef(QualifiedName, env[9]))
  setFunctions(listActions,env[9])

//-------------------------------------------------------
getActionDef(ID:QualifiedName, ⟨⟩) = ϵ
getActionDef(ID:QualifiedName, ⟨'action def'ID2'('Pin*'):'
  QualifiedName'{constraint:'ConstraintUse*
  ActivityDelegation*'}' ⟩) = ϵ
getActionDef(ID:QualifiedName, ⟨'action def'ID'('Pin*'):'
  QualifiedName'{constraint:'ConstraintUse*
  ActivityDelegation*'}' ⟩⌢listAction) = 'action
```

```
    def 'ID' ( 'Pin*' ) : ' QualifiedName '{ constraint : 'ConstraintUse*
    ActivityDelegation*
getActionDef(ID: QualifiedName, ⟨'action def 'ID2' ( 'Pin*' ) : '
    QualifiedName '{ constraint : 'ConstraintUse*
    ActivityDelegation*'}' ⟩⌢listAction) =
    getActionDef(ID: QualifiedName, listAction)

//-------------------------------------------------------------
setFunction(ID':'QualifiedName1('{using
    pins:'Pin';'*'}','action def'ID'('Pin*'):'
    QualifiedName2'{constraint:'ConstraintUse*
    ActivityDelegation*) = ID 'is function ('setInParam(Pin*)')
    :' QualifiedName2'{return}'

//-------------------------------------------------------------
 setInParam(⟨⟩) = ϵ
 setInParam(⟨ ID ':' QualifiedName ⟩) = ⟨ID ':' QualifiedName⟩
 setInParam(⟨ ID ':' QualifiedName ⟩⌢ listParam) = ⟨ID ':'
    QualifiedName⟩ setInParam(listParam)

//-------------------------------------------------------------
(ID: QualifiedName, ⟨⟩) = ϵ
(ID: QualifiedName, ⟨'activity def' ID2 '('Pin*'):('Pin*')
    {'ActivityBody'}' ⟩) = ϵ
getActivity(ID: QualifiedName, ⟨'activity def' ID '('
    Pin*'):('Pin*') {'ActivityBody'}'⟩) = 'activity def' ID
    '('Pin*'):('Pin*') {'ActivityBody'}'
getActivity(ID: QualifiedName, ⟨'activity def' ID
    '('Pin*'):('Pin*') {'ActivityBody'}'⟩⌢listActivity) =
    'activity def' ID '('Pin*'): ('Pin*') {'ActivityBody'}'
getActivity(ID: QualifiedName, ⟨'activity def' ID2
    '('Pin*'):('Pin*') {'ActivityBody'}'⟩⌢listActivity) =
    getActivity(ID: QualifiedName, listActivity)

//-------------------------------------------------------------
setCompLstElementInstantiation(⟨ID':' QualifiedName'{using
    ports:'PortUse*)'}'⟩)= ID'is'QualifiedName'()'
setCompLstElementInstantiation(⟨ID':' QualifiedName'{using
    ports:'PortUse*)'}'⟩⌢CompULst)= ID'is'QualifiedName'() and'
```

```
    setCompLstElementInstantiation(CompULst)

//-------------------------------------------------------
setConLstElementInstantiation(⟨ID ':' QualifiedName' bindings
    'ConnectorBinding*';'⟩) = 'and' ID 'is' QualifiedName'()'
setConLstElementInstantiation(⟨ID ':' QualifiedName' bindings
    'ConnectorBinding*';'⟩⌢ConULst) = 'and'
    ID 'is' QualifiedName() setConLstElementInstantiation(ConUList)

//-------------------------------------------------------
setConUnification(⟨ID ':' QualifiedName 'bindings'
    ConnectorBinding ';'⟩, env[6], env[11])=
setConnectionAccessFrom(ConnectorBinding, env[11]) 'unifies'
    setCAccessFromCon(getConnectorDef(QualifiedName, env[6]))
setCAccessToCon(getConnectorDef(QualifiedName, env[6]))
    'unifies' setConnectionAccessTo(ConnectorBinding, env[11])

setConUnification(⟨ID ':' QualifiedName 'bindings' ConnectorBinding ';'⟩⌢ConU
    env[6], env[11]) =
setConnectionAccessFrom(ConnectorBinding, env[11]) 'unifies'
    setCAccessFromCon(getConnectorDef(QualifiedName, env[6]))
setCAccessToCon(getConnectorDef(QualifiedName, env[6]))
    'unifies' setConnectionAccessTo(ConnectorBinding, env[11])
setConUnification(ConUList)

//-------------------------------------------------------
setConnectionAccessFrom(qnSource: QualifiedName '='
    qnDest: QualifiedName, env[11]) =
    qnSource '::' getComponentUsePU(qnSource, env[11])

//-------------------------------------------------------
setConnectionAccessTo(qnSource: QualifiedName '='
    qnDest: QualifiedName, env[11]) =
    qnDest '::' getComponentUsePU(qnDest, env[11])

//-------------------------------------------------------
setCAccessFromCon('connector def' ID '{' ('participants :'
    PortUse_Reverse*)
```

```
('flows :' Flow*)'}') =
    ID'::'getPorUseReverseOut(PortUse_Reverse*)

//-------------------------------------------------------
setCAccessToCon('connector def' ID '{' ('participants :'
    PortUse_Reverse*)
('flows :' Flow*)'}') =
    ID'::'getPorUseReverseIn(PortUse_Reverse*)

//-------------------------------------------------------
getConnectorDef(ID:QualifiedName, ⟨ 'connector def' ID '{'
    ('participants :' PortUse_Reverse*)('flows :' Flow*)'}'⟩) =
    'connector def' ID '{' ('participants :'
    PortUse_Reverse*)('flows :' Flow*)'}'
getConnectorDef(ID:QualifiedName, ⟨ 'connector def' ID2 '{'
    ('participants :' PortUse_Reverse*)('flows :'
    Flow*)'}'⟩ ⌢ ConDefList) = getConnectorDef(ID:QualifiedName,
    ConDefList)

//-------------------------------------------------------
getComponentUsePU(namePU1: QualifiedName, ⟨ID ':' nameComp:
    QualifiedName '{ using ports :'⟨namePU1:QualifiedName ':'
    QualifiedName'{ }'⟩'}'⟩) = nameComp

getComponentUsePU(namePU1: QualifiedName, ⟨ID ':' nameComp:
    QualifiedName '{ using ports :'⟨namePU1:QualifiedName ':'
    QualifiedName'{ }'⟩ ⌢ puList'}'⟩) = nameComp

getComponentUsePU(namePU1: QualifiedName, ⟨ID ':' nameComp:
    QualifiedName '{ using ports :'⟨namePU2:QualifiedName ':'
    QualifiedName'{ }'⟩ ⌢ puList'}'⟩) =
getComponentUsePU(namePU1: QualifiedName, ⟨ID ':' nameComp:
    QualifiedName '{ using ports :'puList)⟩

getComponentUsePU(namePU1: QualifiedName, ⟨ID ':' nameComp:
    QualifiedName '{ using ports :'⟨namePU1:QualifiedName ':'
    QualifiedName'{ }'⟩'}'⟩ ⌢ compUlist) = nameComp
```

```
getComponentUsePU(namePU1: QualifiedName, ⟨ID ':' nameComp:
    QualifiedName '{ using ports :'⟨namePU1: QualifiedName ':'
    QualifiedName'{ }'⟩ ⌢puList'}'⟩ ⌢compUlist) = nameComp

getComponentUsePU(namePU1: QualifiedName, ⟨ID ':' nameComp:
    QualifiedName '{ using ports :'⟨namePU2: QualifiedName ':'
    QualifiedName'{ }'⟩'}'⟩ ⌢compUlist) =
    getComponentUsePU(namePU1: QualifiedName, compUlist)

getComponentUsePU(namePU1: QualifiedName, ⟨ID ':' nameComp:
    QualifiedName '{ using ports :'⟨namePU1: QualifiedName ':'
    QualifiedName'{ }'⟩ ⌢puList'}'⟩ ⌢CompUlist) = nameComp

getComponentUsePU(namePU1: QualifiedName, ⟨ID ':' nameComp:
    QualifiedName '{ using ports :'⟨namePU2: QualifiedName ':'
    QualifiedName'{ }'⟩ ⌢ puList'}'⟩ ⌢compUlist) =
getComponentUsePU(namePU1: QualifiedName, ⟨ID ':' nameComp:
    QualifiedName '{ using ports :'puList)⟩ ⌢compUlist)

//-------------------------------------------------------
getPorUseReverseOut(⟨'~' ID ':' 'port' 'def' ID2 '{flow' 'out'
    TypeDef'}' '{ }'⟩) = '~' ID

getPorUseReverseOut(⟨'~' ID ':' 'port' 'def' ID2 '{flow' 'out'
    TypeDef'}' '{ }'⟩ ⌢puRevList) = '~' ID

getPorUseReverseOut(⟨'~' ID ':' 'port' 'def' ID2 '{flow' 'in'
    TypeDef'}' '{ }'⟩ ⌢puRevList) =
    getPorUseReverseOut(puRevList)

//-------------------------------------------------------
getPorUseReverseIn(⟨'~' ID ':' 'port' 'def' ID2 '{flow' 'in'
    TypeDef'}' '{ }'⟩) = '~' ID

getPorUseReverseIn(⟨'~' ID ':' 'port' 'def' ID2 '{flow' 'in'
    TypeDef'}' '{ }'⟩ ⌢puRevList) = '~' ID

getPorUseReverseIn(⟨'~' ID ':' 'port' 'def' ID2 '{flow' 'out'
    TypeDef'}' '{ }'⟩ ⌢puRevList) =
```

```
    getPorUseReverseOut(puRevList)

//-------------------------------------------------------
setDelUnification(⟨[ID1 ':' QualifiedName1] 'to' [ID2 ':'
    QualifiedName2]⟩) =
QualifiedName1 '::' ID1 'unifies' QualifiedName2 '::' ID2
```