Appendix C: Example of Object-Z

This appendix contains an Object-Z specification of the two generic relationship types composition and containment, the object types Subnetwork and Link and the relationships between them.

Comments are given in the text to explain the notation.

The following rules are used for indentation and lineshift in assertions:

1) lineshift without indentation means conjunction (whenever possible).

lineshift with indentation means begin parenthesis "(".

3) lineshift with outdentation means end parenthesis ")".

Composition[X, Y]

Generic relationship type: "an X is composed of some Y's". X and Y are generic parameters, i.e. sets that are used, but provided later.

```
\begin{array}{l} ComposedOf: X \leftrightarrow Y \\ Composites: \mathbb{P} \ X \\ Components: \mathbb{P} \ Y \end{array}
```

ComposedOf is a binary relation between X and Y. Composites is a set of X members and Components is a set of Y members.

```
Composites \supseteq \operatorname{dom} ComposedOf
Components \supseteq \operatorname{ran} ComposedOf
```

The domain of *ComposedOf* is a subset of *Composites*. The range of *ComposedOf* is a subset of *Components*

The interpretation of Composites and Components may vary. Later we will give an example where they represent available instances of X and Y (the Information Base).

The \supseteq relations indicates that zero role-cardinality is possible.

. Add CompositesAn operation $\Delta(Composites, Components, ComposedOf)$ Composite?: XAddComposites changes the variables in the Δ list. *Composite*? is an input to the operation *Composite*? \notin *Composites* Composite? is not a member of the Composites set $Composites' = Composites \cup \{Composite?\}$ The new value of Composites is the old value with *Composite*? inserted (set union). $ComposedOf \subseteq ComposedOf'$ $\forall x : X \bullet \forall y : Y \mid x \mapsto y \in ComposedOf' \setminus ComposedOf \bullet$ $x = Composite? \land y \in Components$ \forall means "for all", \exists means "there exist". Only tuples with first element Composite? and second element in Components are inserted into ComposedOf $_RemoveComposites_$ $\Delta(Composites, Components, ComposedOf)$ Composite?: X $Composite? \in Composites$ $Composites' = Composites \setminus \{Composite?\}$ $ComposedOf' = \{Composite?\} \triangleleft ComposedOf$

The *Composite*? is removed from the *Composites* and the relation.

AddComponents_____ $\Delta(Components, Components, ComposedOf)$ Component?: Y $Component? \notin Components$ $Components' = Components \cup \{Component?\}$ $ComposedOf \subseteq ComposedOf'$ $\forall x : X \bullet \forall y : Y \bullet$ $\forall x \mapsto y : X \leftrightarrow Y \mid x \mapsto y \in ComposedOf' \setminus ComposedOf \bullet$ $y = Component? \land x \in Composites$ _RemoveComponents _____ $\Delta(Components, Components, ComponentOf)$ Component?: Y $Component? \in Components$ $Components' = Components \setminus \{Component?\}$ $ComposedOf' = ComposedOf \Rightarrow \{Component?\}$ Add $\Delta(ComposedOf, Composites, Components)$ NewComponent?: YNewComposite?: X $NewComposite? \mapsto NewComponent? \notin ComposedOf$ $ComposedOf' = ComposedOf \cup$ $\{NewComposite? \mapsto NewComponent?\}$ $NewComposite? \in Composites'$ $New component? \in Components'$

 $\begin{array}{c} Remove _ \\ \Delta(ComposedOf, Composites, Components) \\ RemoveComponent?: X \\ RemoveComposite?: Y \\ \hline RemoveComposite \mapsto RemoveComponent? \in ComposedOf \\ ComposedOf' = ComposedOf \\ \{RemoveComposite \mapsto RemoveComponent?\} \\ Composites' \subseteq Composites \\ Components' \subseteq Components \\ \end{array}$

```
Containment[X, Y]
Composition[X, Y]
A generic relationship between X members and Y members.

"an X contains some Ys". Inherits Composition[X, Y].

\hline Components = ran \ ComposedOf
All components have to have a container

\forall y : Y \mid y \in Components \bullet
\exists_1 x : X \mid x \in Composites \bullet
x \mapsto y \in ComposedOf
All components are in at most one container
```

This could also have been modelled as a function from Y to X.

The invariant has implications for the inherited operations: AddComposites: ComposedOf and Components are left unchanged. *RemoveComposites*: All components of the removed composite are removed from Components and all mappings involving the removed composite are removed from ComponentOf. AddComponents: Composites may grow or shrink and the map (The Container, *NewComponent*?) is added to *ComposedOf* RemoveComponents: Composites may grow or shrink and the map (The Container, *Component*?) is removed from *ComposedOf* Add: Composites may grow or shrink and the new component is added to Components *Remove:* Composites and Components may shrink.

Now follows the Subnetwork and Link object types and the SubnetworkSubnetwork, SubnetworkLink and ConnectedBy relationship types.

_ Subnetwork _____

 $\boxed{\begin{array}{c} level: \mathbb{N} \\ \exists s: Subnetwork \bullet s. level = 0 \end{array}}$

The *level* attribute indicates the level of nesting of the subnetwork within the layered network.

	RootSubnetwork
	Subnetwork
ſ	
	level = 0

_NonMaxSubnetwork ______ Subnetwork

level > 0

_ *Link* _____

We are focussing on relationships so Links do not have any attributes or invariants at this time.

_SubNSubN _____

Containment[*Subnetwork*, *NonMaxSubnetwork*]

This is a relationship class describing the Subnetwork/Subnetwork relationship type. It gives actual parameters *Subnetwork* and *NonMaxSubnetwork* to the generic parameters in *Containment*, which it inherits.

 $\begin{array}{l} \forall \, s \, : \, Components \, \bullet \, \forall \, s1 \, : \, Composites \, \bullet \\ s \, \mapsto \, s1 \, \in \, Component \, \Rightarrow \, s1.level \, = \, s.level \, + \, 1 \end{array}$

_SubNLink

Containment[Subnetwork, Link]

This is a relationship class describing the Subnetwork/Link relationship type. Note that it inherits a containment relationship so that all properties of containment are preserved.

```
\forall l : Link \bullet l \in Components
```

For every link l there exist exactly one Subnetwork \boldsymbol{s} such that \boldsymbol{s} contains l.

_ ConnectedBy _

This is a relationship class describing the connected-by relationship type. Exactly two Subnetworks are connected by a link in this relation.

 $SLS: \mathbb{P}(NonMaxSubnetwork \times Link \times NonMaxSubnetwork)$ $\forall s_1, s_2 : NonMaxSubnetwork \bullet \forall l : Link$ $(s_1, l, s_2) \in SLS \bullet$ $(s_1 \neq s_2 \land s_1.level = s_2.level > 0)$ $\forall s_3, s_4 : NonMaxSubnetwork \bullet$ $(s_1, l, s_2) \in SLS \land (s_3, l, s_4) \in SLS \Rightarrow$ $s_1 = s_3 \land s_2 = s_4$ if $SLS(s_1, l, s_2)$ and $SLS(s_3, l, s_4)$ then $s_1 = s_3$ and $s_2 = s_4$. $\exists s : Subnetwork \bullet$ $\exists SS : SubNSubN \bullet$ $s \mapsto s1 \in SS.ComposedOf$ $s \mapsto s2 \in SS.ComposedOf$ $\exists SL : SubNLink \bullet$ $s \mapsto l \in SL.ComposedOf$ If $(s_1, l, s_2) \in SLS$ then there exist some subnetwork s containing s1 and s2 and s contains l in the SubNLink relationship.

"The link connects two subnetworks in a larger subnetwork containing the subnetworks and the link" The following describes an example application of the previous modeling. It uses the classes to create a layered network.

[Layered, Links, SL, SS, CB, InformationBase]

These are given sets

 $InformationBase = Layered \cup Links \cup SL \cup SS \cup CB$

The Layered set contains the Subnetwork objects of the layered network.

The Links set contains the Link objects of the layered network.

The SS set contains the SubnetworkSubnetwork relationship objects within the layered network.

The SL set contains the SubnetworkLink relationship objects within the layered network.

the CB set contains the ConnectedBy relationship objects within the layered network.

Only object identities are contained in these sets.

LayeredNetwork

Root : *RootSubnetwork* $S: \mathbb{P}$ NonMaxSubnetwork $L: \mathbb{P} Link$ SNSN : SubNSubNSNLN: SubNLinkCONB : ConnectedByRoot.Level = 0 $Layered = \{Root\} \cup S$ SNSN.Components = S $SNSN.Composites \subset Layered$ SNLN.Composites = LayeredSNLN.Components = Links $\forall s1, s2 : Subnetwork \bullet \forall l : Link$ $|(s1, l, s2) \in CONB.SLS \bullet$ $s1 \in S \land s2 \in S \land l \in Links$ $\forall \, s : S \, \bullet \, \exists \, s2 : S \, \bullet \, \exists \, l : Links \, \bullet$ $(s, l, s2) \in CONB.SLS \lor (s2, l, s) \in CONB.SLS$