Part III
Chapter 11

Names

11.1 Introduction

In its simplest form a name is a symbol which is used to refer to some object. There are no constraints on the nature of the symbols that are used for names, although individual systems will restrict the symbol set that can be used. There are no constraints whatsoever on what may be named. Any object of any type, including objects external to a system, can be named. All names in the context of computer systems are denotational, that is, names are only used to refer or point to things.

A **named object** appears to know its own name. This appearance is false however, when it is examined in the context of physical realizability. The name is part of the interactions between an object and the named object and the named object will only respond in the desired way when the proper symbol or name is supplied.

An object may have one or several names. When an object has several names these are often called aliases. A name may also refer to several objects. In such a case, a policy is required to determine which of the several objects is actually referred to. The policy may specify that an arbitrary choice be made. An object may use a name for which there is no corresponding object. Again a policy will be required to resolve the conflict.

A named object can be modelled by a **name** and a **nameable object.** The latter is an object that can be given one of many names. Which name is given depends on the name that is attached to the object as a property.

A **sentinel** can make an unnamed object look as if it is a named object. The sentinel links a name and a policy to the unnamed object and intervenes in the unnamed objects interactions to give the appearance of a named object.

A **name interpreter** is similar to a sentinel, but where a sentinel deals with a single name and a single unnamed object, the name interpreter can deal with many objects and many names. The name interpreter requires policies for duplicate names and for anonymous objects. These policies, together with the collection of names, are collected in the **naming model.**

ometimes it is beneficial to use a structure of name interpreters and aming models to provide a certain route to an unnamed object. The iteractions between the objects in the structure of name interpreters, with ieir naming models is governed by the provision of a **structured name**. he way in which the structured name is interpreted, is reflected in the ructure of the name interpreters and their naming models.

litorial: In this chapter it is assumed that a naming context exists, that is implicitly scribed by the extent of the interaction alphabets and the content of the the naming odel. This assumption is satisfactory at the moment, but a future version of this chapter ill have to include the naming context explicitly.

1.2 Reference section

he manual pages that follow contain the description of the concepts of amed object, name, nameable object, sentinel, name interpreter and aming model.

ı the future, separate entries for node, naming context and alias can be cpected.

litorial: The entries in this chapter will need to be aligned with those in the chapter on odels. In particular, the naming model is really a denotation rather than a model as fined in chapter 9.

NAME

Named object

PURPOSE

To provide an object that can distinguish its own name as part of its interaction with other objects

SYNOPSIS

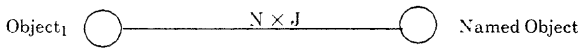
A named object can impose one of two constraints on its interactions with other objects. The named object imposes one constraint when its name is supplied with the interaction. It places another constraint when its name is not supplied. A named object is thus able to distinguish its name that is supplied as part of the interaction it has with other objects.

A named object may have more than one name. These different names are often referred to as *aliases*.

CANONICAL FORM

A named object is represented in the same way as any other object and as illustrated in Figure 11.1. The interactions between a named object and any other objects consists of a composite alphabet.

Figure 11.1 Interactions with a named object



SPECIFICATION

A named object interacts with other objects using a composite alphabet. In Figure 11.1 the named object interacts with $Object_1$ using the alphabet $N \times J$. The named object can impose one of two constraints on alphabet J, denoted by J_1 and J_2.

Alphabet N consists of two parts N_1 and N_2 that do not have any common members. The relation maintained by the named object is such that when a member of N_1 is chosen, constraint J_1 is imposed and when a member of N_2 is chosen, constraint J_2 is imposed. The relation maintained by the named object can thus be written as

$R = N_1 \times J_1 \cup N_2 \times J_2$

When N_1 contains only few members and N_2 contains a great m members then the members of N_1 can be interpreted as the nam the named object. Consequently J_1 will be the desired constra imposed when the correct name of the named object is used. Wl any other name is used constraint J_2 is imposed. When $J_2 = J$ tl the named object does not place any constraints when its nam not used.

Since a member of N is either a name of the named object or nc name of the named object, the set of names of the named object a the set of symbols that are not the name of the named object must disjoint:

$N_1 \cap N_2 = \{\}$

A named object must have at least one name. If it has more tl one name, these names are often called *aliases*. This is expressed the predicate

$\#N_1 \geq 1$

If an object has a name, it must be possible to distinguish betw what is a valid name and what is not. For this reason there m exist at least one name that is not the name of the named obj which is expressed by the predicate

$\#N_2 \geq 1$

REALIZABILITY ISSUES

The name is merely part of the interaction alphabet. T realizability issues for names are the same as those that apply symbols of an alphabet. The named object only appears to know name; from its perspective the composite alphabet is all tl matters.

SEE ALSO

SYMBOL
ALPHABET
COMPOSITE INTERACTION

FUTURE DIRECTIONS

NAME

Name

PURPOSE

To allow the name of a named object to be separately represented

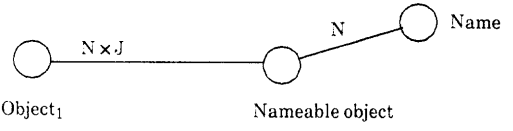
SYNOPSIS

A named object can be decomposed and represented by two parts. One of the parts is the name of the object which constrains the interactions that are possible with that object. The name is an attribute.

CANONICAL FORM

The canonical form is shown in Figure 11.2. A named object has been decomposed into a nameable object and its name.

Figure 11.2 The name is a property



SPECIFICATION

The name is connected with a nameable object. The name imposes a constraint N_1 on the its interactions with the nameable object. N_1 is the set of names that the nameable object will recognize. When $\#N_1 = 1$ then the nameable object will only be able to distinguish one name. If $\#N_1 \geq 1$, then the nameable object has several names or *aliases*.

REALIZABILITY ISSUES

The name is really only part of the interaction alphabet. The realizability issues for names are the same as those that apply to symbols of an alphabet.

SEE ALSO

PROPERTY

FUTURE DIRECTIONS

NAME

Nameable object

PURPOSE

To provide a representation for an object that can have many nam

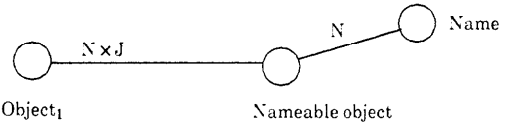
SYNOPSIS

A named object can be decomposed into two parts. One of the pa is the nameable object. The nameable object is an object with attribute that constrains the interactions that are possible with t nameable object.

CANONICAL FORM

The canonical form is shown in Figure 11.3. A named object h been decomposed into a nameable object and its name. T nameable object has its attribute, a name, attached to it.

Figure 11.3 A nameable object associated with a name as a property



SPECIFICATION

The nameable object in Figure 11.3 maintains a relation R that c be defined as

$R \subset N \times N \times J$

when the name that is part of the interaction between $Object_1$ a the nameable object is the same as the name of the object indicat by its property, the nameable object imposes a constraint J_1, or:

$R_1 = \{(n_1,n_2,j) \mid n_1 = n_2 \Rightarrow j \in J_1\}$

which can also be written as

$R_1 = I_{nn} \times J_1$

If the name is not the same, the nameable object imposes constraint J_2. This can be written as

$R_2 = \{(n_1,n_2j) \mid n_1 \neq n_2 \Rightarrow j \in J_2\}$

and by introducing a new notation for this expression R_2 can be rewritten as

$R_2 = \neg I_{nn} \times J_2$

The total relation maintained by the nameable object is the union of R_1 and R_2 :

$R = I_{nn} \times J_1 \cup \neg I_{nn} \times J_2$

REALIZABILITY ISSUES

The name is really only part of the interaction alphabet. The realizability issues for names are the same as those that apply to symbols of an alphabet.

SEE ALSO

NAME
PROPERTY
COMPOSITE INTERACTION

FUTURE DIRECTIONS

NAME

Sentinel

PURPOSE

To allow a name to be associated with an unnamed object

SYNOPSIS

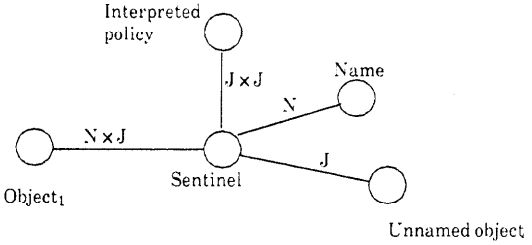
An object that can only interact with a simple alphabet and tl does not have a name directly attributed to it is an unnamed obje It is often desirable to interact with an unnamed object as if it ha name. A sentinel acts as a guard and only allows interactions w a particular unnamed object when the correct name is supplied.

The sentinel has a policy to deal with situations in which a name supplied, but where there is no object associated with that name.

CANONICAL FORM

The canonical form is illustrated in Figure 11.4. An unnamed obj is connected to a sentinel. The sentinel interacts with $Object_1$ a has access to a policy and the name that is now associated with t unnamed object.

Figure 11.4 A sentinel makes an unnamed object appear to be named



SPECIFICATION

From the perspective of $Object_1$ in Figure 11.4 the sentinel behav as a named object.

$R' = N_1 \times J_1 \cup N_2 \times J_2$

The relation maintained by the sentinel is represented by

$R \subset N^2 \times J^4$

The name object imposes a constraint N_1 on its interactions with the sentinel. N_1 is defined as the set of names that the sentinel is willing to accept as names for the unnamed object. When the name in use during interaction with $Object_1$ is taken from N_1 then the sentinel uses the unnamed object to enforce its constraint on the interactions with $Object_1$. The relation can be written as:

$R_1 = \{(n_1, n_2, j_1, j_2, j_3, j_4) \mid n_1 = n_2 \Rightarrow j_2 = j_3\}$

or $R_1 = I_{nn} \times J \times I_{ij} \times J$

When a name is used that is different from the one expected, $Object_1$ does not get any access to the unnamed object. In that case the sentinel will need to perform all interactions with $Object_1$. In order to decide what constraint the sentinel should apply when an incorrect name is supplied, the sentinel has a policy at its disposal. The policy will need to decide the constraints on the interactions with both $Object_1$ and the unnamed object, hence the composite alphabet $J \times J$

The relation maintained by the sentinel when the "wrong" name is used is represented by

$R_2 = \{(n_1, n_2, j_1, j_2, j_3, j_4) \mid n_1 \neq n_2 \Rightarrow j_1 = j_2 \wedge j_3 = j_4\}$

or using the shorthand notation:

$R_2 = \neg I_{nn} \times I_{ij} \times I_{ij}$

The complete specification for the relation maintained by the sentinel is the union of the relations R_1 and R_2, thus

$R = I_{nn} \times J \times I_{ij} \times J \cup \neg I_{nn} \times I_{ij} \times I_{ij}$

REALIZABILITY ISSUES

The sentinel must be able to interpret the name and the policy.

SEE ALSO

INTERPRETER
MODEL
POLICY

FUTURE DIRECTIONS

NAME

Name interpreter

PURPOSE

To associate a name to each of a collection of objects

SYNOPSIS

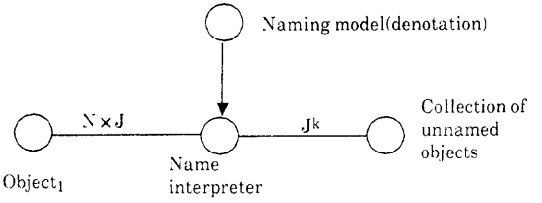
A name interpreter is similar to a sentinel, but where a sentir deals with a single unnamed object, the name interpreter can de with a collection of unnamed objects. The name interpreter c: associate a name to each object in a collection of unnamed objects.

The name interpreter requires a policy to deal with situations which several objects have the same name (*duplicate names*), to de with names that have no corresponding object and to deal wi *anonymous objects.*

CANONICAL FORM

The canonical form is illustrated in Figure 11.5. A collection of unnamed objects is connected to the name interpreter. The nar interpreter also has a connection to a naming model and to an obje labelled $Object_1$ that represents all objects that want to intera with objects in the collection of objects.

Figure 11.5 A name interpreter associates names with each object in a collection of objects



SPECIFICATION

Assuming that the interactions between the naming model and t name interpreter have the form

$J \times J \times (J \times J \times N)^k$

then the relation can be constructed by considering the cases in which

- a name is supplied that is associated to a unique object
- a name is supplied that is associated with several objects
- a name is supplied that is not associated with any objects

The complete relation can be represented by the union of the above cases.

Anonymous objects are dealt with in the same way as objects that are not addressed.

Aliases are dealt with by multiple entries in each part of composite N^k

For every name in the collection of names, there exists an object in the collection of objects. It may be something trivial, e.g. an object that can only say that the name is invalid.

REALIZABILITY ISSUES

The name interpreter must be able to interpret the naming model and the policies.

SEE ALSO

SENTINEL
NAMING MODEL
INTERPRETER
MODEL
POLICY

FUTURE DIRECTIONS

The specification is to be refined. The entry must be aligned with the one for naming model, specially with respect to the policy aspects.

NAME

Naming model

PURPOSE

To describe the set of names that the naming interpreter c interpret and to specify the policy that applies when one na corresponds to more than one object and when a name is used for object that does not exist

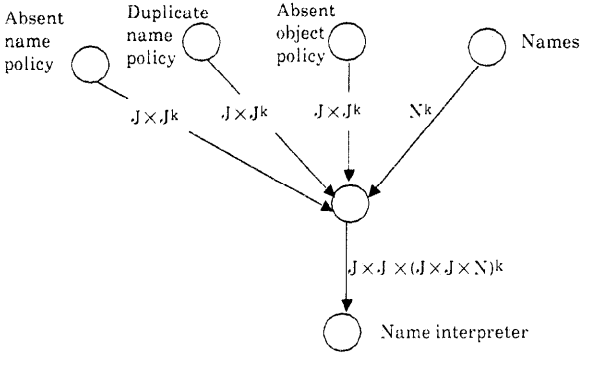
SYNOPSIS

A naming model describes a set of names and their associations w objects in a collection of objects. A naming interpreter is required turn the associations between names and objects into bindings connections.

CANONICAL FORM

The canonical form for a naming model is illustrated in Figure 11 where the naming model is decomposed into a number of objec One of the objects contains the names, and there are three pol objects for absent objects, absent names and for duplicate name The naming model is connected to a name interpreter.

Figure 11.6 A decomposed naming model



SPECIFICATION

REALIZABILITY ISSUES

The naming model and the policies must be interpretable by a name interpreter.

SEE ALSO

DENOTATION
NAME INTERPRETER
ASSOCIATION
INTERPRETER
MODEL
POLICY

FUTURE DIRECTIONS

Provide the specification and align the entry with name interpreter.

Editorial: the naming model is really a denotation, as defined in Chapter 9 on Models.

NAME

Structured name

PURPOSE

To allow a name to consist of several parts

SYNOPSIS

A name may consist of several parts. If it is to be used then all pa must be specified for the name to be valid. A structured name thus be modelled by a composite symbol and a set of structu names by a composite alphabet. It is then possible to decompose name interpreter so that each composite part can interpret onl part of the composite name. The structure of the decompo interpreter and naming model will depend on whether structured name is a sequence of composite parts, that need to sequentially interpreted or whether it consists of parallel parts.

CANONICAL FORM

SPECIFICATION

REALIZABILITY ISSUES

SEE ALSO

NAME INTERPRETER
NAMING MODEL
COMPOSITE SYMBOL
COMPOSITE ALPHABET

FUTURE DIRECTIONS

A separate entry must be provided for sequenced composite nai Sequenced composite names are most often used in compu systems, because the parts of a name are usually interpreted i certain order, one part at the time, thus defining a route throug network of name interpreters.

11.3 Examples

Editorial: Examples are to be provided.