Role Change in Database Domains

Roel Wieringa
Department of Mathematics and Computer Science
Free University
De Boelelaan 1081
1081 HV Amsterdam
The Netherlands
uucp address: roelw@cs.vu.nl

ABSTRACT

In data modelling the universe of discourse (UoD) is divided up into classes having a taxonomic structure which is intended to express some of the structure inherent in the UoD. Some of these classes, for example the class of persons or departments, may be called "natural kinds," in that they are a fixed set of possible objects, existing in some possible state of the UoD, and all of which have a similar structure and behavior. Others have a more dynamic nature, such as the class of students. Whereas an object is created as as person and, when it ceases to be a person, ceases to exist, an object may come to be a student and cease to be one without coming into existence or passing away. A class like persons is a natural kind, and a class like students will be called roles in this report. This report studies the formal definition of roles and the resulting taxonomy of natural kinds and roles.
1. Introduction

1.1.
In a recent proposal for a data model, Abiteboul & Hull [1987] distinguish two kinds of subtype relations, called specialization and generalization, respectively. For example, the types STUDENT and EMP loyee specialize the type PERSON in that STUDENT and EMP are roles of a PERSON. One the other hand, CARs and MOTORBOATs may be generalized to VEHICLEs; the division of VEHICLES into CARs and MOTORBOATs is fixed in that a CAR is not a role of VEHICLE but a kind of VEHICLE.

Before carrying on with this example, we want to make a philosophical remark about the status of the classes of vehicles, cars and motorboats as natural kinds. In as far as cars and motorboats are social or legal constructs, labels attached to objects by habit, convention, convenience or legal decision, it is easy to envision a socially possible or legally admissible world in which cars can turn into motorboats and vice versa. In as far as "car" and "motorboat" are names, not so much of kinds of objects but of the use we make of objects, migration between these classes of objects is really a change of use of these objects and not a change of natural kind. By calling cars and motorboats natural kinds, we treat the distinction between cars and motorboats as a real division in nature, i.e. one exists independently of our use of them, a division which would exist even if there were no human beings on this planet. For the purpose of conceptual modeling, this is not an unreasonable attitude, since we may regard categorizations found in a UoD as facts of nature, not to be altered by the information analyst. For the sake of the current example, we assume that the classes of CARs and MOTORBOATs are natural kinds, so that CARs do not turn into MOTORBOATs or vice versa. When we study change of use of objects, which is a special case of change of the role which objects play, we will use other examples.

Instead of using the terms generalization and specialization to indicate the difference hinted at by Abiteboul and Hull, we will use the terms natural kind and role. A natural kind is a set of object identities to which a fixed set of attributes is applicable, called the types of the kind. A role is a set of attributes which characterize a role which an object may play. Later we will give precise definitions, here we explain our intuitions about roles in terms of the example of Abiteboul and Hull. Names of
natural kinds are written with an initial capital letter, role names are written in capital letters only.

We regard the sets of all possible Persons, Vehicles, Cars, and Motorboats as natural kinds and the sets of STUDENTs and EMPs as roles of Persons. At least the following differences exist between these natural kinds and roles.

1. During its existence, an object does not become a member of a natural kind like Person or Car. Rather, it is born as an object of a natural kind; when it exists, it exists as a Person or a Car. Similarly, when an object ceases being a Person, it ceases to exist; it cannot cease to be a Person and continue existing.

2. An object may start playing a role like STUDENT or EMP immediately at birth or at a later point in its life, and similarly it may stop playing a role without ceasing to exist.

3. At any moment it is possible for a STUDENT to become an EMP and vice versa. By contrast, no Car can become a Motorboat or vice versa.

4. If at a particular moment all Persons are STUDENTs or EMPs, then we may not conclude that this is always the case. But if at a particular moment all Vehicles are Cars or Motorboats, then this is the case at all moments.

There are two interesting ways to define a role. First, one may define an attribute emp# which in each state of the world is applicable to some persons and define the people to which it is applicable to be employees. There are now two types of events, one which change the value of an applicable attribute, and one which makes an attribute applicable or terminates its applicability to an individual object. Role change is then formalized as change in attribute applicability.

A second way to define a role is to define a subrange of values of an attribute. For example, one may define the role ANTIQUE–CAR of Car, defined by the attribute age being in the subrange $[70, \infty)$ of $\mathbb{N}$. Role change in this case is a change in attribute value which causes the value to enter or leave the subrange of attribute values connected with a particular role.

This kind of range restriction is to be sharply distinguished with what we may call attribute specialization, where for example it is known that the value of max–speed on the natural kind of Cars lies between 0 and 1000 (measured in a certain unit). A vehicle does not start to play the role of car when its maximum speed enters this subrange, nor ceases it to play the role of a car when its maximum speed leaves this subrange. max–speed is not definitive for the role Car, for Car is not a role. Rather, what is intended is that for all cars, max–speed happens (or ought to) lie in this subrange. Specialization by restriction of attribute values thus sometimes, but not always, introduces the possibility of role change.

Specialization by attribute restriction has been adopted by most data models which allow one to define a taxonomic hierarchy. Most of these, like TAXIS (Mylopoulos et al. [1980]), SDM (Hammer & McLeod [1981]), and the Event Model (King & McLeod [1984]), do not develop a formal basis for their models. In particular, they have no formal definition of role change and consequently cannot distinguish the two kinds of attribute restriction, one defining a role and the other associated with a subkind. IFO (Abiteboul & Hull [1987]) is a formal study of data model structures which has been used to study updates, but the problem of role change has not been dealt with explicitly.

In this paper we propose to study class change in more detail. In the rest of this introduction we summarize some terminology from Wieringa [1987] and in section 2, we list the relevant definitions relating to partial functions needed in the rest of this paper. In section 3 we distinguish universal attributes, which are universally applicable and are used to define natural kinds, from partial attributes, which may not be always applicable and are used to define roles. We also assimilate role change by entry or exit of an attribute value into or from a subrange to the case of an attribute.
becoming to be or ceasing to be applicable. Section 4 treats the changes to be made to the theory of natural types to account for this new situation, and section 5 through 7 define events, in particular role changes, in this new framework. Section 7 also shows how to express constraints related to roles which objects can play, and shows which constraints need not be specified anymore because they are built into the system. Some constraints can only be specified in terms of processes because they concern more than one event. Examples are constraints on the playability of a role, e.g. on entering a role only when another role is already being played, etc. Sections 8 and 9 deal with these process-related constraints. Section 10 ends with a summary of the results and directions for future research.

1.2.

A database domain is an abstract representation of a universe of discourse (UoD) and has the form of a rooted directed graph. The nodes of the graph are called worlds or states of the domain, the arrows are called events. An event is a function from possible worlds to possible worlds. A world is a set of objects, and an object is a pair \((s, \sigma)\) where \(s\) is called the identity of the object and \(\sigma\) the internal state or state vector of the object. For example, in an employee domain we may have an object

\[(p_1, \text{name}: \text{John}, \text{age}: 35),\]

where \(p_1\) is the unique identity of this object and \((\text{name}: \text{John}, \text{age}: 35)\) the state of this object. The object

\[(p_1, \text{name}: \text{John}, \text{age}: 36),\]

is considered to be (mathematically) a different object but is said to be identical to the previous one, because it has the same identity. An event never changes the identity of an object.

The set of all possible object identities is \(S\), so \(p_1 \in S\). The elements of \(S\) are also called surrogates, because they are abstract representations of identities of UoD entities. For this reason, the model presented here is called ABSURD (ABstract SURrogate Domain).

An arbitrary subset of \(S\) is called a kind. The set of all kinds is called \(K\), so \(K = \mathcal{P}(S)\). Metavariable over \(K\) is \(k\). A natural kind is a set of identities which have the same aggregation structure and behavior. Two identities have the same aggregation structure if the objects they identify have isomorphic state vectors. For example,

\[(p_1, \text{name}: \text{John}, \text{age}: 36),\]
\[(p_1, \text{name}: \text{John}, \text{age}: 36)\] and
\[(p_2, \text{name}: \text{Karin}, \text{age}: 30)\]

have the same aggregation structure. Two identities have the same behavior if the same set of events applies to them and these events are subject to the same constraints (see Wieringa & van de Riet [1988] for details on this). An example of a natural kind is \(\text{Person} \subset S\). The set of all natural kinds is called \(\mathcal{X}\) and is a complete lattice under set inclusion. This lattice is also called a generalization hierarchy. Metavariable over \(\mathcal{X}\) is \(k\) (as opposed to \(k\) for \(K\)). If \(k_1, k_2 \in \mathcal{X}\) and \(k_1 \subseteq k_2\), then \(k_1\) is a specialization of \(k_2\) and \(k_2\) is a generalization of \(k_1\).

An attribute is a set of partial functions from \(S\) into \(S\) which all have the same domain. For example,

\[\text{age} = [\text{Person} \rightarrow \mathbb{N}]\] and
\[\text{name} = [\text{Person} \rightarrow \text{NAME}],\]

where \(\text{NAME}\) is a set of possible names.
A world is a set of objects with different identity. The set of all possible words is $PW$ and metavariable over $PW$ is $w$.

If $a$ is applicable to $s$, and $s$ exists in $w$ (it identifies an element of the set $w$), then $a_w(s)$ is the value of $a$ on $s$ in $w$. For example, if $w_1 \in PW$ and

$$(p_1, (name: John, age: 36)) \in w_1,$$

then

$$name_{w_1}(p) = John$$
$$age_{w_1}(p) = 36.$$ 

The set of all attributes in a domain is $A$. A finite set of attributes is called a type and the set of all types is $T$, so $T = \mathcal{P}(A)$. Two functions are defined,

$$kind: T \rightarrow K,$$

which assigns to a set of attributes the set of identities to which they are all applicable, and

$$type: K \rightarrow T,$$

which assigns to a set $k$ of identities the set of all attributes applicable to all identities in $k$. If $t = type(k)$ for an arbitrary non-empty $k$, then $t$ is called a natural type. The set of all natural types is called $\mathcal{S}$. Like $\mathcal{K}$, $\mathcal{S}$ is a complete lattice. $kind$ and $type$ are each other’s inverse when restricted to $\mathcal{S}$ and $\mathcal{K}$. Thus, a kind $k$ is a natural kind because there is a $t \subseteq A$ such that $k = kind(t)$, just as a type $t$ is a natural type because there is a $k \subseteq A$ such that $t = type(k)$. $kind$ and $type$ form a Galois connection between $\mathcal{K}$ and $\mathcal{S}$, which means that suprema are mapped to infima and vice versa.

To distinguish between the objects in a world $w$ which exist and those which do not exist, there is an object $(db, (alive: aa, dead: dd))$. In each world, $aa$ is the set of object identities which exist in that world and is called the existence set, and $dd$ is the set of object identities which have existed in a previous world. (Note that attribute values may be sets of identities.) An object comes into existence by entering $aa$ and ceases to exist when it leaves $aa$ and enters $dd$ (which is one event). Object identities may reincarnate by moving from $dd$ back to $aa$.

The database domain is specified by a set of axioms which define the attributes, natural kinds, etc. The appendix gives an example and Wieringa [1987] gives details.

2. Partial functions

An attribute has been defined as a set $[k_1 \rightarrow k_2]$, i.e. the set of all possible total functions from $k_1$ into $k_2$ (since $k_1$ will in general be genuinely contained in $S$, the functions are partial functions on $S$).

The main idea in the definition of roles is to allow an attribute to contain some, not necessarily all, partial functions from $k_1$ into $k_2$. If $a$ is such an attribute, then for any $s \in k_1$, there will be worlds where $s$ has a value for $a$ and others where it has not. Sets of partial functions thus allow us to define roles. In the case of a partial function, there may be some misunderstanding about what exactly is the domain and range of the function and when two partial functions are equal. The following definitions take care of this.
2.1. Definition

A partial function from $k_1 \subseteq S$ to $k_2 \subseteq S$ is denoted $k_1 \leftarrow k_2$. $k_1$ is called the source and $k_2$ the range of $f$, denoted $\text{src}(f) = k_1$ and $\text{range}(f) = k_2$. The domain of $f$ is the set of elements in $\text{src}(f)$ for which $f$ has a value, written $\text{dom}(f)$. For any $k$, the image of $k$ under $f$ is

$$f[k] = \{x \in \text{range}(f) \mid \exists y \in k [f(y) = x]\}.$$ 

The image of $f$ is $\text{im}(f) = f[\text{dom}(f)]$. The inverse image of $k$ under $f$ is

$$f^{-1}[k] = \{x \in \text{dom}(f) \mid f(x) \in k\}.$$ 

Two functions $f_1$ and $f_2$ are equal iff

$$\text{dom}(f_1) = \text{dom}(f_2) \text{ and } \forall x \in \text{dom}(f_1) [f_1(x) = f_2(x)].$$

We thus have an extensional view of functions. Equal functions may have different sources and ranges, but their domains and effects on the elements of their domain are equal.

For example, if $\text{ZIP} \subseteq \mathbb{N}$, then $f_1: \text{Person} \leftarrow \mathbb{N}$ and $f_2: \text{Person} \leftarrow \text{ZIP}$ are equal if $f_1$ and $f_2$ happen to assign the same numbers to the same persons. This is so even if $f_1 \in \text{age}$ assigns an age to a person and $f_2 \in \text{zip}$ assigns a zip code to a person.

Note, incidentally, that a total function is a special case of a partial function, where $\text{dom}(f) = \text{src}(f)$.

Some elementary properties of partial functions we need later on are:

2.2. Lemma

1. $f^{-1}[k] \subseteq \text{dom}(f) \subseteq \text{src}(f)$ and $f[k] \subseteq \text{im}(f) \subseteq \text{range}(f)$.
2. If $k_1 \subseteq k_2$, then $f[k_1] \subseteq f[k_2]$ and $f^{-1}[k_1] \subseteq f^{-1}[k_2]$.
3. If $f_1 = f_2$, then $\text{im}(f_1) = \text{im}(f_2)$.
4. = is an equivalence relation on $[S \leftarrow S]$. □

The following definition is needed to distinguish the two cases of attribute restriction.

2.3. Definition

Let $k \subseteq S$ be non-empty.

1. The domain restriction of $f$ to $k$ is the function $f \downarrow k: \text{src}(f) \cap k \rightarrow \text{range}(f)$ defined by $g(x) = f(x)$ for $x \in k \cap \text{dom}(f)$. Instead of $f \downarrow k$, we also write $f_k$.
2. The range restriction of $f$ to $k$ is the function $f^k = f \downarrow f^{-1}[\text{range}(f) \cap k]$. □

2.4. Lemma

1. $\text{dom}(f_k) \subseteq \text{dom}(f)$ and $\text{im}(f_k) \subseteq \text{im}(f)$.
2. $\text{dom}(f^k) \subseteq \text{dom}(f)$ and $\text{im}(f^k) \subseteq \text{im}(f)$.
3. If $\text{dom}(f) \subseteq k$, then $f_k = f$.
4. If $\text{im}(f) \subseteq k$, then $f^k = f$. □

We will use domain restrictions to model attribute specialization related to natural subkinds, and image restriction to model attribute specialization related to roles. For this reason we define a partial ordering on functions determined by image restriction.
2.5. Definition

The binary relation $\leq$ on partial functions $S \subseteq S$ is defined by

$$f_1 \leq f_2 \iff \left( \text{dom}(f_1) \subseteq \text{dom}(f_2) \land \text{im}(f_1) \subseteq \text{im}(f_2) \land f_1 = f_2 \downarrow \text{dom}(f_1) \right).$$

2.6. Lemma

1. $\leq$ is a partial order on the set of partial functions $S \subseteq S$.
2. $f^k \leq f$ for any $k$.
3. $f_k \leq f$ for any $k$. □

3. Partial and universal attributes

3.1. Definition

The set of all partial functions with source $k_1$ and range $k_2$ is denoted $[k_1 \leftarrow k_2]$.

1. A non-empty set $a \subseteq [k_1 \leftarrow k_2]$ containing at least one non-total function is called a partial attribute with $\text{src}(a) = k_1$ and $\text{range}(a) = k_2$.
2. The set $a = [k_1 \rightarrow k_2]$ of all total functions $k_1 \rightarrow k_2$ is called a universal attribute with $\text{src}(a) = k_1$ and $\text{range}(a) = k_2$.
3. The domain of a partial or universal attribute $a$ is the set

$$\text{dom}(a) = \bigcup_{f \in a} \text{dom}(f)$$

and the image is

$$\text{im}(a) = \bigcup_{f \in a} \text{im}(f).$$

4. $a_1$ and $a_2$ are equal iff $\forall f \in a_1 \iff f \in a_2$. □

The term universal is derived from classical logic, where it means "valid for all regions of space-time." Our universal attributes are in all worlds applicable to elements in their domain. Universal attributes are identical to the attributes defined in Wieringa [1987]. Note that universal attributes are not a special case of partial attributes, because a partial attribute must contain at least one non-total function whereas a universal attribute contains only total functions.

Some elementary facts about attributes are:

3.2. Lemma

1. If $a$ is partial, then $\text{dom}(a) \subseteq \text{src}(a)$ and $\text{im}(a) \subseteq \text{range}(a)$.
2. If $a$ is universal, $\text{dom}(a) = \text{src}(a)$ and $\text{im}(a) \subseteq \text{range}(a)$.
3. $\text{dom}([k_1 \leftarrow k_2]) = k_1$. □

Examples of a partial and a universal attribute are

$$\text{stud\#} = [\text{Person} \leftarrow \text{STUD\#}],$$

$$\text{age} = [\text{Person} \rightarrow \mathbb{N}].$$

$\text{dom(age)} = \text{Person}$ is the set of surrogates to which $\text{age}$ is applicable, but $\text{dom(stud\#)} \subseteq \text{Person}$ is the set of surrogates to which $a$ may be applicable. In this case, we have $\text{dom(stud\#)} = \text{Person}$ so
that \textit{stud\#} may be applicable to any set of persons. In any world, we can view the applicability of \textit{stud} as definitive for students; an object is a student, plays the role of student, if \textit{stud\#} is applicable to it. There may be states of the domain where a total function in \{\textit{Person} \models \textit{STUD\#}\} is actual. In those worlds all persons are students.

From now on we assume a set \(A^P\) of partial attributes and \(A^u\) of universal attributes, \(A^P \cap A^u = \emptyset\) and \(A^P \cup A^u = A\). The appendix gives an example of \(A\).

We now define three kinds of partial attributes.

3.3. Definition

Let \(k_1, k_2 \subseteq S\).

1. The set \([k_1 \vdash k_2]\) is called a \textit{role attribute}.

2. Let \(a \subseteq [k_1 \vdash k_2]\) be any partial attribute and let \(k_3 \subseteq S\) be non-empty. Then the \textit{range restriction} of \(a\) to \(k_3\) is the attribute \(a^{k_3} = \{f^{k_3} \mid f \in a\}\). \(a^{k_3}\) is called an \textit{range-restricted} or simply \textit{restricted} attribute.

3. Let \(a\) and \(k_3\) be as in 2. Then the \textit{domain restriction} of \(a\) to \(k_3\) is the attribute \(a^{k_3} = \{f^{k_3} \mid f \in a\}\). \(\Box\)

An example of a role attribute is \textit{stud\#} defined above.

Range restrictions and domain restrictions formalize the difference between attribute value restrictions which define roles and those which are related to a natural kind. An example of a domain-restricted attribute is

\begin{align*}
1 & \quad \text{truck-weight} = \text{weight}_{\text{Truck}} \\
\end{align*}

with

\begin{align*}
\text{weight} &= [\text{Vehicle} \rightarrow \mathbb{N}], \\
\text{Truck} &\subseteq \text{Vehicle}.
\end{align*}

We assume here that the natural kind \textit{Truck} \subseteq \textit{Vehicle} has been defined elsewhere. \textit{truck-weight} is simply the \textit{weight} attribute restricted to \textit{Truck}. This is not very interesting, unless we know that, for example,

\begin{align*}
2 & \quad \text{weight}_{w}[\text{Truck}] = \mathbb{N}^{\geq 3000} \text{ for all } w \in PW.
\end{align*}

This statement is either an empirical generalization ("as a matter of fact, all trucks weigh more than 3000 tonnes") or a legal stipulation ("trucks must weigh more than 3000 tonnes"). In both cases, it can be used as a static integrity constraint on the value of the \textit{weight} attribute for trucks. The constraint allows us to derive

\begin{align*}
3 & \quad \text{truck-weight} = \text{weight}^{\geq 3000}.
\end{align*}

In most treatments of attribute restriction, 3 is treated as the definition of \textit{truck-weight}. However, this blurs the fact that there are really two statements made in 3, the definition of \textit{truck-weight} as the \textit{weight} of \textit{Trucks} in 1, and the constraint that \textit{Trucks} weigh more than 3000 tonnes in 2. 1 and 2 are not derivable from 3, unless we assume that only trucks weigh more than 3000 tonnes,

\begin{align*}
\text{weight}_{w}[\text{Vehicle} \cdot \text{Truck}] = \mathbb{N}^{< 3000} \text{ for all } w \in PW.
\end{align*}

Separating these 1 from 2, we make clear that no vehicle can become a truck by increasing its weight to above 3000 tonnes.

An example of a range-restricted attribute is
senior-age = \text{age}^\text{N+55},

with \text{age} = [\text{Person} \rightarrow \mathbb{N}]. For every \text{Person} identity, \text{age} may have value 66 or more, and there is no natural subkind of \text{Person} for which all \text{age} values are over 65. Instead, senior-age may be used to define a role of objects, which they play when they are in a state with \text{age} > 65.

We now want to claim that all interesting attributes are total attributes, role attributes or domain- and range restrictions of total or role attributes, in the sense that any domain specification with other attributes than these can be altered such that all attributes are of one of these types. For example, take the partial attribute

\text{boss} = \{ f : \mathcal{F}(\text{Person}) \downarrow \text{Person} \mid \text{for } f(pp) \in \mathcal{F}(\text{Person}) \text{ with } |pp| \geq 4 \}.

\text{boss} assigns a person to each group of at least 4 people. This is a partial attribute, because

\text{boss} \subseteq [\mathcal{F}(\text{person}) \downarrow \text{Person}],

and it is not a role-, domain-, or range restriction attribute. However, the domain can be respecified so that \text{boss} is a role attribute, by defining the natural kind

\text{Large-groups} = \{ pp \subseteq \text{Person} \mid |pp| > 3 \}

and defining

\text{boss} = [\text{Large-groups} \downarrow \text{Person}].

Not only this eliminates a partial attribute which is not a role-, domain-, or range restriction, it also expresses the intention of the designer in a better way, because it shows that there are groups of people where the \text{boss} predicate may be applicable.

In general, a partial attribute \text{a} is a set of functions \([k_1 \rightarrow k_2]_k\), and this can be converted in a role attribute by introducing a natural kind \text{dom}(a). We can therefore restrict our attention to partial attributes which are role attributes or range- or domain restrictions.

We can reduce the kinds of partial attributes to consider even further, by the following theorem.

**3.4. Theorem**

1. \([k_1 \rightarrow k_2]_k = [k_1 \cap k_3 \rightarrow k_2].\)
2. \([k_1 \uparrow k_2]_k = [k_1 \cap k_3 \downarrow k_2].\)
3. \([k_1 \rightarrow k_2]^{k_3} = [k_1 \downarrow k_2 \cap k_3].\)
4. \([k_1 \uparrow k_2]^{k_3} = [k_1 \uparrow k_2 \cap k_3].\)

**Proof.**

1. \(f \in [k_1 \rightarrow k_2]_k\) iff there is a \(g : k_1 \rightarrow k_2\) with \(f = g \downarrow k_1 \cap k_3\), which is the case iff \(f \in [k_1 \cap k_3 \rightarrow k_2].\)
2. As 1.
3. \(f \in [k_1 \rightarrow k_2]^{k_3} \iff [f = g \downarrow g^{-1}[k_3 \cap \text{range} (g)]\) for a \(g : k_1 \rightarrow k_2\) \(\iff f \in [k_1 \rightarrow k_2 \cap k_3].\)
4. As 3. \(\square\)

Examples are
working-age = [Person \rightarrow \mathbb{N}]_{16-65}^{16-45} = [Person \times \mathbb{N}]_{16-65},
senior-age = [Person \times \mathbb{N}]_{65},
truck-weight = [Vehicle \rightarrow \mathbb{N}]_{Truck} = [Truck \rightarrow \mathbb{N}],
heavy-truck-weight = truck-weight^{N^{10000}} = [Truck \times \mathbb{N}]^{N^{10000}}.

(See the appendix for the working-age attribute.)

We can conclude from the theorem that we only need to consider role attributes. Starting with a
database domain specification containing only universal- and role attributes, we can at most produce
more role attributes by range restriction and never produce other kinds of attributes.

4. The natural type lattice

We now generalize the definition of the partial ordering on types by allowing a type to be specialized
by adding an attribute or by restricting an attribute. To define this, we need a partial ordering on
attributes.

4.1. Definition

The relation \( \leq \) is defined on \( A \) by

\[ a_1 \leq a_2 \iff a_1 \text{ is a domain- and/or range restriction of } a_2. \]

\( a_1 \) is called a **specialization** of \( a_2 \) and \( a_2 \) is a **generalization** of \( a_1 \). \( \square \)

Examples are

\[ [Person \times \mathbb{N}]^{16} \leq [Person \rightarrow \mathbb{N}], \]
\[ [Truck \times \mathbb{N}]^{3500} \leq [Truck \rightarrow \mathbb{N}] \leq [Vehicle \rightarrow \mathbb{N}], \]
\[ \text{heavy-truck-weight} = ([Vehicle \rightarrow \mathbb{N}]_{Truck})^{N^{10000}} \leq [Vehicle \rightarrow \mathbb{N}] = \text{weight}. \]

4.2. Lemma

If \( a_1 \leq a_2 \) then \( \text{dom}(a_1) \subseteq \text{dom}(a_2) \) and \( \text{im}(a_1) \subseteq \text{im}(a_2) \).

4.3. Theorem

\( \leq \) is a partial ordering on \( A \).

**Proof.**

1. \( a \leq a \), for \( a_{\text{dom}(a)}^\text{im}(a) = a \).

2. We do this only for range restriction. If \( a_1 \leq a_2 \leq a_3 \), then there are \( k_1 \) and \( k_2 \) with \( a_1 = a_2^{k_1} = (a_3^{k_2})^{k_1} = a_3^{k_2 k_1} \), so \( a_1 \leq a_3 \). The case with domain restriction is similar.

3. Finally, again ignoring domain restriction, if \( a_1 \leq a_2 \) and \( a_2 \leq a_1 \), then \( a_1 = a_2^{k_1} \) and \( a_2 = a_1^{k_2} \), so \( a_1 = a_2^{k_1 k_2} \). By the lemma above this implies that \( \text{im}(a_1) \subseteq k_1 \cap k_2 \subseteq k_1 \), so that \( a_2 = a_1^{k_1} = a_1 \). The case with domain restriction is analogous. \( \square \)

We can now define the partial ordering on types.
4.4. Definition
The relation $\leq$ on $\mathcal{F}(A)$ is defined by

$$t_1 \leq t_2 \iff \forall a_2 \in t_2 \exists a_1 \in t_1[a_1 \leq a_2].$$

If $t_1 \leq t_2$, then $t_2$ is called a generalization of $t_1$ and $t_1$ is called a specialization of $t_2$. □

For example, taking the attributes of the University example,

$$\{\text{name, age, working-age}\} \leq \{\text{name, working-age}\} \leq \{\text{name, age}\} \leq \{\text{name}\}.$$ 

To specialize a type is thus to add more attributes or to restrict existing attributes. To generalize is to lift a restriction or to drop an attribute. $\{\text{name, age}\}$ is a more general type than $\{\text{name, working-age}\}$. This is the common view of specialization, as formalized for example by Cardelli [1984]. The special feature of our view is that we can distinguish the natural kind defined by a type from the role(s) defined by a type, as detailed in the following sections.

4.5. Lemma
$\leq$ is a partial order on $\mathcal{F}(A)$ which coincides with $\supseteq$ for elements of $\mathcal{F}(A^u)$.

Proof.
Reflexivity and transitivity follow from the reflexivity and transitivity of $\leq$ on attributes. Antisymmetry follows from the finiteness of the types. If $t_1 \leq t_2$ and $t_2 \leq t_1$ then for each $a_1 \in t_1$ there is an $a_2 \in t_2$ with $a_1 \leq a_2$, for which there is an $a_3 \in t_1$ with $a_2 \leq a_3$, etc. Because there are finitely many attributes in $t_1$ and $t_2$, there will be a cycle

$$a_1 \leq a_2 \leq \ldots \leq a_1.$$ 

By antisymmetry of $\leq$ on attributes, $a_1 = a_2$. □

The theory of natural types of Wieringa [1987] holds also for the new partial ordering, as shown in the following theorem.

4.6. Theorem
kind: $\mathcal{F}(A) \rightarrow \mathcal{F}(S)$ and type: $\mathcal{F}(S) \rightarrow \mathcal{F}(A)$, with $\mathcal{F}(A)$ ordered by $\leq$, form a Galois connection.

Proof.
We only have to prove that (Herrlich & Hušek [1986], with the dual ordering on $\mathcal{F}(A)$)

$$\text{type}(k_1 \sqcup k_2) = \text{type}(k_1) \sqcap \text{type}(k_2).$$

It is easy to verify this, for

$$a \in \text{type}(k_1 \sqcup k_2) \iff $$
$$k_1 \subseteq \text{dom}(a) \land k_2 \subseteq \text{dom}(a) \iff $$
$$a \in \text{type}(k_1) \sqcap \text{type}(k_2).$$

It is also easy to show that kind, type is a Galois connection between $\mathcal{X}$ and $\mathcal{F} \cap \mathcal{F}(A^u)$. The complete lattice $\mathcal{F} \cap \mathcal{F}(A^u)$ is not a sublattice of $\mathcal{F}$, because the lub of two types is different in both lattices.

It can easily be shown that

$$\text{obj} \circ \text{kind}: \mathcal{F} \rightarrow \mathcal{C},$$
$$\text{type} \circ \text{id}: \mathcal{C} \rightarrow \mathcal{F}$$

form a Galois connection. Instead of assigning the set of all possible surrogates to which the
attributes in \( t \) are applicable, \( obj.kind \) assigns the set of all possible objects in which these attributes occur. This gives all objects with \( t \) as their natural kind plus all objects playing roles contained in \( t \).

5. Events

5.1. Definition

1. A universal type is a set of universal attributes. Metavariable over the set of universal types is \( ut \). The universal type of \( k \) is defined as

\[
\text{type}^u(k) = \text{type}(k) \cap A^u.
\]

2. A role is a set of partial attributes. The set of all possible roles is \( Q = \mathcal{P}(A^p) \) and metavariable over \( Q \) is \( q \).

We define a role as a set of attributes because it are objects in a certain state which play a role. Object identities play no role, they just have a natural kind which, in the current conception, is characterized by a set of universal attributes. There may be a set of partial attributes defined for the natural kind as well, so that some objects of the kind will have these attributes in their state vector and others will not. Rather than defining a role as a set of objects which have certain partial attributes in their state vector, we simply define it as the set of relevant partial attributes.

If \( t \) is universal, all attributes in \( t \) are applicable to all surrogates in \( \text{kind}(t) \) in all worlds. If \( t \) is not universal, this will not be the case in all worlds. All types can be decomposed into a universal type and a role,

\[
t = ut \cup q \quad \text{with} \quad q \cap ut = \emptyset,
\]

where one of \( ut \) and \( q \) may be empty. This is just a convenient notation. For example, if we take as attribute set

\[
A_1 = \{\text{age, working-age, senior-age, junior-age, name, stud#, courses, emp#}\}
\]

where the attributes are defined in the appendix, then

\[
\text{type}(\text{Person}) = \text{type}^u(\text{Person}) \cup q, \quad \text{with} \\
\text{type}^u(\text{Person}) = \{\text{age, name}\}, \\
q = \{\text{working-age, senior-age, junior-age, stud#, courses, emp#}\}.
\]

Our problem is to decompose \( q \) into "realistic" roles, i.e. roles which a person can play. Intuitively, some possible roles in this example are

- student \( \{\text{stud#, courses}\} \)
- employee \( \{\text{emp#}\} \)
- junior \( \{\text{junior-age}\} \)
- work-force \( \{\text{working-age}\} \)
- senior \( \{\text{senior-age}\} \)

We may also want to allow combined roles, such as student employees and senior students. In addition, there may be constraints, like that all employees must be in the work-force and that juniors cannot be work-force or senior.

Our solution to the problem of defining realistic roles is based upon the fact, noted in the introduction, that there must be an event which causes an object to play the role or to stop playing the role.
We therefore define a class of events called role-changes, which make a set of partial attributes applicable or makes them unapplicable. For example, what distinguishes \{\text{stud#}, \text{courses}\} from \{\text{stud#}\} is that there is an event, \textit{become\,-\,-student}, which makes all and only attributes in the first set applicable and there is no event which makes only the attributes in the second set applicable. An event like \textit{become\,-\,-student} which makes a set of partial attributes applicable to an object will be called a role change. Role changes have as only function the addition or removal of attributes from a state vector.

To be able to define role changes, we must re-define events to allow for changes in attribute applicability, which is what we will do in this section. We start with the definition of objects with variable-length state vectors.

5.2. Definition

1. For \(t = \{a_1, \ldots, a_n\} \subseteq A\),
   \[\text{tuples}(t) = \{(a_1; s_1, \ldots, a_n; s_n) \mid s_i \in \text{range}(a_i)\} \].

2. The state space of \(t = ut \cup q \in \mathcal{F}(A)\) is the set of tuples with minimally the attributes in \(ut\) and maximally all the attributes from \(ut \cup q\):
   \[
   \text{space}(ut \cup q) = \bigcup_{ut \leq s \leq (ut \cup q)} \text{tuples}(t).
   \]

3. The universe is the set
   \[
   U = \{(s, \sigma) \mid s \in \mathcal{S} \land \sigma \in \text{space}(t), \text{for all } t \subseteq \text{type}([s])\}.
   \]

4. The class of kind \(k\) is
   \[\text{obj}(k) = \{(s, \sigma) \mid s \in k, \sigma \in \text{space(type}([s]))\}\].

The \(t\)-class of kind \(k\) is
   \[\text{obj}_t(k) = \pi_t[\text{obj}(k)]\].

Thus, \(\text{tuples}(t)\) contains all vectors of fixed length \(\|t\|\) and \(\text{space}(t)\) contains all possible variable-length state vectors.

In the university example (appendix) we have
   \[
   \text{space}([\text{name, age, working\,-\,-age, senior\,-\,-age}]) = \\
   \text{tuples}([\text{name, age}]) \cup \\
   \text{tuples}([\text{name, age, working\,-\,-age}]) \cup \\
   \text{tuples}([\text{name, age, senior\,-\,-age}]) \cup \\
   \text{tuples}([\text{name, age, working\,-\,-age, senior\,-\,-age}]).
   \]

\text{obj}(k)\) is the set of objects with their identity in \(k\) and whose state vectors minimally contain the universal attributes of \(k\) and maximally contain all universal and partial attributes of \(k\). In \(\text{obj}_t(k)\) the vectors are truncated to attributes in \(t\). \(U\) is the set of all objects with a state vector containing zero or more applicable attributes. Thus, \(\bigcup_{k \subseteq \mathcal{S}} \text{obj}(k) \subseteq U\).

Our definition of restriction attributes makes certain objects impossible, such as
(p, (junior-age: 12, working-age: 12)) \notin \text{obj}(\text{Person}) \text{ and } (p, (working-age: 21)) \notin \text{obj}(\text{Person}).

However, other objects are possible, but inconsistent, such as

(p, (age: 20, working-age: 21)) \notin \text{obj}(\text{Person}).

On the other hand,

(p, (age: 20, working-age: 20))

is consistent but redundant, and equivalent to

(p, (working-age: 20)) as well as to

(p, (age: 20)).

We will define events in such a way that they preserve consistency. Role changes add or remove an attribute; in the case of a restriction attribute, they add or remove redundancy (e.g. addition or removal of working-age: 20). We first define consistent objects.

5.3. Definition
1. \(a_1\) and \(a_2\) are called compatible if \(a_1 \subseteq a_2\) or \(a_2 \subseteq a_1\).

2. An object is called redundant if its state vector contains two compatible attributes.

3. An object is called inconsistent if its state vector contains two compatible attributes with different values.

In order to account for communication events, we have to extend these concepts to tuples of objects. To do this, we partially order \(U\).

5.4. Definition
The relation \(\leq\) on \(U\) is defined by

\[(s, (a_1: s_1, ..., a_m: s_m)) \leq (s, (a'_1: s'_1, ..., a'_n: s'_n)) \triangleq \{a_1, ..., a_m\} \subseteq \{a'_1, ..., a'_n\} \land \{a_i \in \{a_1, ..., a_m\} \Rightarrow s_i = s'_i\}.\]

It is easily seen that this is a partial order. Note that only identical objects may be related by \(\leq\). An example is

\[(p_1, \text{name: John}) \leq (p_1, \text{name: John, age: 30}).\]

The lub of two objects in \(\leq\), if defined, is their natural join:

\[(p_1, \text{name: John, emp#: e#}) \cup (p_1, \text{name: John, age: 30}) =
(p_1, \text{name: John, emp#: e#, age: 30}),
(p_1, \text{name: John, age: 31}) \cup (p_1, \text{name: John, age: 30}) \text{ is undefined.}\]

5.5. Definition
A tuple \((o_1, ..., o_n)\) is consistent iff each subtuple of identical objects has a lub. □

In its simplest form, an event is a function \(\text{obj}(k) \rightarrow \text{obj}(k)\) which changes the state of an object, but leaves the identity and kind of the object invariant. For example, if \(\text{type}(k) = \{a\}, \)
is a simple event in the life of objects of kind \( k \). In general, an event can be parametrized, such as in

\[
\text{add}: \mathbb{N} \rightarrow (\text{obj}(k) \rightarrow \text{obj}(k)) = \text{add}(m)(s, (a: n)) = (s, (a: m+n)).
\]

Generalizing further, an event may change the state of more than one object simultaneously, in which case it works on an \( n \)-tuple of objects. The general definition of an event follows.

5.6. Definition

An event is a function

\[
e: k_1 \times \ldots \times k_m \rightarrow (\text{obj}_{i_1}(k_1) \times \ldots \times \text{obj}_{i_n}(k_n) \rightarrow \text{obj}_{r_1}(k_1) \times \ldots \times \text{obj}_{r_n}(k_n)).
\]

In an application \( e(s_1, \ldots, s_m)(o_1, \ldots, o_n) \), \( s_i \) are called the actual parameters and \( o_i \) the subjects of the application. If \( n = 1 \), \( e \) is called a solitary event, if \( n \geq 1 \), it is called a communication event.

\( e \) may be applied to any tuple of objects with identities in \( k_1 \times \ldots \times k_n \). The effect of \( e \) on \( (o_1, \ldots, o_n) \) when \( o_i \notin \text{obj}_{i}(k_i) \) is defined as follows:

1. The effect on attributes in \( t_i \) is defined in the function body,
2. The effect on attributes \( a' \) compatible with an attribute \( a \) in \( t_i \) is the same as that on \( a \), unless the new value is not defined for \( a' \),
3. The effect on all other attributes is that of the identity function. \( \square \)

Remarks:

1. Note that the parameter sets \( R_{k_i}, i = 1, \ldots, m \) can be any kind, but the objects must be of natural kinds \( k_i, i = 1, \ldots, n \).
2. The subjects are projected on arbitrary types \( t_i, i = 1, \ldots, n \). In most cases, \( t_i = \text{type}^a(k_i) \), and we write \( \text{obj}_{i}(k_i) \) instead of \( \text{obj}_{i}(k_i) \).
3. As usual, if \( t_i = \text{type}(k_i) \), i.e. contains all universal and partial attributes of \( k_i \), we write \( \text{obj}(k_i) \) instead of \( \text{obj}_{i}(k_i) \).
4. In all cases, in the body of \( e \) the universal attributes which are not changed by \( e \) are omitted.

A simple example of a solitary event is

\[
\text{change-name}: \text{NAME} \rightarrow (\text{obj}_u(\text{Person}) \rightarrow \text{obj}_u(\text{Person})) = \text{change-name}(n_0)(p, (\text{name}: n_1)) = (p, \text{name}: n_0).
\]

\text{change-name} only affects the universal attribute \text{name}, other universal attributes (in this case \text{age}) remain constant, which is why we omitted them from the specification of the effect. Another example of a solitary event is

\[
\text{inc-age}: \text{obj}_{\text{age}}(\text{Person}) \rightarrow \text{obj}_{\text{age}}(\text{Person}) = \text{inc-age}(p, (\text{age}: n)) = (p, (\text{age}: n+1)).
\]

\text{inc-age} and \text{change-name} can be used to define a communication event
inc-age | change-name: NAME \rightarrow (\text{obj}_u(Person) \times \text{obj}_u(Person)) \rightarrow \text{obj}_u(Person) \times \text{obj}_u(Person)
=
inc-age | change-name(n_0)((p_1, (name: n_1)), (p_2, (age: n)))
= ((p_1, (name: n_0)), (p_2, (age: n+1))).

So

inc-age | change-name(John)((p_1, (name: Fred)), (p_2, (age: 30)))
= ((p_1, (name: John)), (p_2, (age: 31))).

This communication event can be applied to a tuple consisting of several states of a single object and then yields a consistent tuple,

inc-age | change-name(John)((p, (name: Fred)), (p, (age: 30)))
= ((p, (name: John)), (p, (age: 31))).

We would like to treat this result as equivalent to

(p, (name: John, age: 31)),

which is what we do in the next definition.

5.7. Definition
1. If \( o = o_1 \cup \ldots \cup o_n \) exists, then the tuple \( (o_1, \ldots, o_n) \) is called equivalent to \( o \).
2. If \( o' = o_1 \cup \ldots \cup o_m \) exists, where \( o_i, \ldots, o_m \) occurs as a subtuple in the tuple \( (o_1, \ldots, o_n) \), then
\( (o_1, \ldots, o_n) \) is called equivalent to the tuple resulting from replacing \( o_i, \ldots, o_m \) by \( o' \).
3. A tuple of objects is called consistent if every set of identical objects in it has a lub. □

Consistent tuples can be replaced by equivalent tuples in which all objects have different identity. The effect of an event on such an equivalent tuple is then easily found. For example, the tuple

\(((p, (name: Fred)), (p, (age: 30)))\)

is consistent and equivalent to

\((p, (name: Fred, age: 30)).\)

(This is a tuple of 1 object & we have omitted the outer brackets.) The communication event can be applied to this object,

inc-age | change-name(John)(p, (name: Fred, age: 30)) = (p, (name: John, age: 31)).

The tuple

\(((p, (name: John, age: 30)), (p, (name: Fred, age: 30)))\)

is inconsistent, so the effect of inc-age | change-name on it is not defined.

6. Role changes

6.1. Definition
1. The dimension of \( o = (s, (a_1; s_1, \ldots, a_n; s_n)) \) is \( \dim(o) = \{a_1, \ldots, a_2\} \).
2. Let \( \dim(o) = ut \cup q \). Then we say that \( o \) plays role \( q \). □

We now define role changes for solitary events.
6.2. Definition

If

\[ e : k_1 x \ldots k_m \rightarrow (\text{obj}_j(k) \rightarrow \text{obj}_r(k)) \]

then \( t \) is called the **input dimension** of \( e \) and \( t' \) is called the **output dimension**. If \( t = ut \cup q \) and \( t' = ut' \cup q' \), then the roles defined by \( e \) are

\[ rl(e) = \{q, q'\}. \]

\( q \) is called the **input role** and \( q' \) the **output role** of \( e \). If \( q = q' \), \( e \) is called **role-invariant**, otherwise it is called a **role change**. □

Because communication events are composed of solitary events, they introduce no new role changes and we can leave them out of consideration in the determination of roles.

The events **change-name** and **inc-age** of the previous section are role-invariant,

\[ rl(\text{change-name}) = rl(\text{inc-age}) = \{\emptyset\}. \]

Universal events always define the universal role \( \emptyset \).

An example of a role change is

\[ \text{become-student} : \text{STUD}\# \rightarrow (\text{obj}_{\emptyset}(\text{Person}) \rightarrow \text{obj}_{\{\text{stud}#, \text{courses}\}}(\text{Person}))^*, \]

\[ \text{become-student}(s#)(p, e) = (p, (\text{stud}#: s#, \text{courses}: \emptyset)). \]

This event defines the role \( \{\text{stud}#, \text{courses}\} \) and we have

\[ rl(\text{become-student}) = \{\emptyset, \{\text{stud}#, \text{courses}\}\}. \]

\( e \) is the empty (zero-length) state vector. The effect of **become-student** on any \( \text{Person} \) object is simply the identity on all attributes not in \( e \), i.e. on all attributes. For example,

\[ \text{become-Student}(s#)(p, (\text{name}: \text{John}, \text{emp}#: \text{e}#)) = (p, (\text{name}: \text{John}, \text{emp}#: \text{e}#, \text{stud}#: s#', \text{courses}: \emptyset)). \]

This application is valid because the result is in \( \text{obj}(\text{Person}) \). On the other hand, the following application is not valid,

\[ \text{become-Student}(s#)(p, (\text{name}: \text{John}, \text{emp}#: \text{e}#, \text{stud}#: s#', \text{courses}: \text{cc})) = (p, (\text{name}: \text{John}, \text{emp}#: \text{e}#, \text{stud}#: s#', \text{courses}: \text{cc}, \text{stud}#: s#', \text{courses}: \emptyset)), \]

because the result is not an object in \( \text{obj}(\text{Person}) \). Later we will see that this is a special case of the general problem when role changes can occur in the life of an object. A student cannot become a student when s/he already is a student, but only a student can become a graduate student, so that the **graduate** role change must occur after **become-student** and before **finish-student**.

Another example of a role change is

\[ \text{become-emp} : \text{EMP}\# \rightarrow (\text{obj}_{\emptyset}(\text{Person}) \rightarrow \text{obj}_{\{\text{emp}\}}(\text{Person}))^*, \]

\[ \text{become-emp}(e#)(p, e) = (p, (\text{emp}#: e#)). \]

\( \text{stud}\# \) and \( \text{emp}\# \) are role attributes. For restriction attributes like **working-age** the treatment is more complicated. For example,

\[ \text{enter-workforce} : \text{obj}_{\emptyset}(\text{Person}) \rightarrow \text{obj}_{\{\text{working-age}\}}(\text{Person})^*, \]

\[ \text{enter-workforce}(p, e) = (p, (\text{working-age}: 16)). \]

We have
The rules for application of an event say that the effect of \( \text{enter-workforce} \) on any attribute not compatible with an attribute in \( \emptyset \) is the identity. This gives us

\[
\text{enter-workforce}(p, (\text{age}: 12)) = (p, (\text{age}: 12, \text{working-age}: 16)),
\]

so that clearly something extra must be defined for roles which contain restriction attributes. In this case, we want \( \text{enter-workforce} \) to be executed only in synchronization with \( \text{inc-age} \), when the input object has age 15. Thus,

\[
\text{inc-age} \parallel \text{enter-workforce}(p, (\text{age}: 15)) = (p, (\text{age}: 16, \text{working-age}: 16)).
\]

Subsequent applications of \( \text{inc-age} \) will then by the rules of event application simultaneously update age and working-age. This formalizes the intuition that age and working-age are in one sense the "same" attribute, because one is a restriction of the other, and on the other hand different attributes, because one becomes applicable as a result of an update of the other.

For the simultaneous update to work as intended, we must demand that

1. \( \text{inc-age} \) can occur on its own if the age of its input object is not equal to 15, but
2. must be executed synchronously with \( \text{enter-workforce} \) if the input object has age 15.
3. \( \text{enter-workforce} \) cannot occur on its own and must be executed synchronously with \( \text{inc-age} \).

These conditions will be specified as past of the process specification in section 9.

Similarly, we have events which finish a role, like

\[
\text{finish-student}: \text{obj}_{\text{stud#}}(\text{Person}) \to \text{obj}_{\emptyset}(\text{Person}) =, \\
\text{finish-student}(p, (\text{stud#}: s#)) = (p, \varepsilon).
\]

To define exit from the role \( \{\text{working-age}\} \), would like to define the event

\[
\text{leave-workforce}: \text{obj}_{\text{working-age}}(\text{Person}) \to \text{obj}_{\emptyset}(\text{Person}) =, \\
\text{leave-workforce}(p, (\text{working-age}: 64)) = (p, \varepsilon).
\]

A synchronized application with \( \text{inc-age} \) is

\[
\text{inc-age} \parallel \text{leave-workforce}(p, (\text{age}: 64, \text{working-age}: 64)) = (p, (\text{age}: 65)).
\]

The problem with this is that exit of the role \( \{\text{working-age}\} \) is also entry into the role \( \{\text{senior-age}\} \).

Adding the attribute \( \text{senior-age} \), we have

\[
\text{become-senior}: \text{obj}_{\emptyset}(\text{Person}) \to \text{obj}_{\text{senior-age}}(\text{Person}) =, \\
\text{become-senior}(p, \varepsilon) = (p, \text{senior-age}: 65)).
\]

The synchronization requirement now becomes that \( \text{become-senior} \) and \( \text{leave-workforce} \) can only occur synchronized with each other and with \( \text{inc-age} \) when this is applied to a person of age 65.

\[
\text{inc-age} \parallel \text{leave-workforce} \parallel \text{become-senior}(p, (\text{age}: 64, \text{working-age}: 64)) = \\
(p, (\text{age}: 65, \text{senior-age}: 65)).
\]

Since which roles there are depends upon which events are defined, we now fix a set \( E_1 \) of generic events. For the sake of the example let us define the set

\[
E_1 = \{\text{become-student, enter-grad-school, become-emp}\}
\]

with
enter-grad-school: Person \rightarrow (\text{obj}_{\text{stud}}(\text{Person}) \rightarrow \text{obj}_{\text{stud}^\#, \text{advisor}, \text{papers}}(\text{Person}))
\begin{equation}
\text{graduate}(a)(s, (\text{stud}^\#: s\#)) = (s, (\text{stud}^\#: s\#, \text{advisor}: a, \text{papers}: \emptyset)),
\end{equation}

where \text{advisor} is the thesis advisor of a graduate student and \text{papers} is the set of papers written as a graduate student.

\text{type}(\text{Person}) can now be decomposed as in figure 1, with

\text{type}^u(\text{Person}) = \{\text{age}, \text{name}\},
\text{type}(\text{Person}) = \text{ut} \cup \{\text{emp}^\#, \text{stud}^\#, \text{advisor}, \text{courses}, \text{papers}\},
\text{STUDENT} = \{\text{stud}^\#, \text{courses}\},
\text{GRAD-STUDENT} = \text{STUDENT} \cup \{\text{advisor}, \text{papers}\},
\text{EMP} = \{\text{emp}^\#\}.

\begin{figure}
\centering
\includegraphics[width=\textwidth]{diagram}
\caption{Decomposition of \text{type}(\text{Person}) in basic roles.}
\end{figure}

The arrows in figure 1 follow the direction of the \(\leq\) on finite types, i.e. there is an edge \(t_1 \rightarrow t_2\) iff \(t_1 \geq t_2\).

\(E_1\) defines more roles than shown in figure 1, because the event definitions allow (graduate) students to become employees and vice versa. For example, the role

\(\text{EMP} \cup \text{STUDENT} = \{\text{stud}^\#, \text{courses}, \text{emp}^\#\}\)

is defined because

\(\text{become-student}(s\#)(p, (\text{emp}^\#: e\#))\)

is defined for all \(e\# \in \text{EMP}^\#\). More generally, \(q_1 \cup q_2\) is a role if at least one event which lets objects of \text{kind}(q_1) play (or stop) role \(q_1\) is applicable to objects of dimensionality \(q_2\), or vice versa.

Similarly, \(q_0 = q_1 \cap q_2\) is an attainable role if there is an event initiating or stopping it.

We first define the set of basic roles of objects of a kind with respect to a set \(E\) of events.
6.3. Definition
The function \( \text{roles}: \mathcal{P}(S) \rightarrow Q \) is defined by \( \text{roles}(k) = \bigcup_{e \in E_k} r(e) \). The elements of \( \text{roles}(k) \) are called \textit{basic roles} of \( k \). □

To define composite roles like \( \text{EMP} \cup \text{STUDENT} \), we must know if an event is applicable to an object playing a role. To find this out, we make a graph of the possible roles attainable from a universal type and the events effecting the transitions. From this graph we can the read the set of basic and composite roles which objects of a kind can reach.

6.4. Definition
For each natural type \( t \) the \textit{role change graph} \( RCG(t) \) is a directed graph with labeled nodes and edges, constructed as follows:

1. Assign a node to each subset \( q \) of partial attributes in \( t \).
2. For each \( e \in E_{\text{kind}(t)} \) with input role \( q_1 \) and output role \( q_2 \), connect the nodes labeled \( q_1 \), \( q_2 \subseteq t \) with an edge labeled \( e \).
3. Omit disconnected nodes. □

Obviously, there is a unique \( RCG(t) \) for any \( t \). The role change graph of our example is given in figure 2.

\[
\begin{align*}
\varnothing & \overset{\text{become-student}}{\rightarrow} q_1 \overset{\text{graduate}}{\rightarrow} q_2 \\
q_3 & \overset{\text{become-emp}}{\rightarrow} q_1 \cup q_2 \cup q_3
\end{align*}
\]

\( q_1 = \text{STUDENT} \)
\( q_2 = \text{GRAD-STUDENT} \)
\( q_3 = \text{EMP} \)

Figure 2. Role change graph for student/employee example.

The root of the graph represents the persons playing the universal role \( \varnothing \), any other node represents people playing more specialized roles.

Writing \( q \in RCG(t) \) if \( q \) is a label of a root node in \( RCG(t) \), we can now define the set of attainable roles of objects of a kind. (Note that \( \mathcal{Z} \) is \( Q \) in commercial script and \( \varphi \) is \( q \) in the same script.)

6.5. Definition

1. The function \( \text{roles}: \mathcal{P}(S) \rightarrow Q \) is defined by \( \text{roles}(k) = \{ q \mid q \text{ is a node in } RCG(\text{type}(k)) \} \). Non-basic elements of \( \text{roles}(k) \) are called \textit{composite roles}

2. The set \( \mathcal{Z} \) of \textit{reachable} roles defined by \( E \) is \( \mathcal{Z} = \bigcup(\text{roles}(X)) \). Metavariable over \( \mathcal{Z} \) is \( \varphi \).
3. We assume that our event language contains a set $CON_2$ of names for basic roles. Role names are written in capital letters. Metavariable over $CON_2$ is $\overline{q}$. For each $\overline{q}$ there is a role axiom in the event theory of the form

$$\overline{q} = \{\overline{a}_1, ..., \overline{a}_n\},$$

where $\overline{a}_i$ are partial attributes. □

"Reachable" as defined here does not include a starting node but should be construed as "reachable from any node in $RCG(t)$ different from the current one."

We can now decompose type"(Person) in basic as well as composite roles, as shown in figure 3, which we call a role decomposition graph of type(Person), written $RDG(type(Person))$.

![Role decomposition graph of type(Person)](image)

**Figure 3.** Role decomposition graph of type(Person).

If $t = ut \cup \overline{q}_1 \cup ... \cup \overline{q}_n$, then $RDG(t)$ is simply a graph of the set $(t, ut) \cup \text{roles}(\text{kind}(t)) = \{t, ut, \overline{q}_1, ..., \overline{q}_n\}$, ordered by the specialization relation on types, where $ut$ is the universal type of $t$. To avoid cluttering up the graph we have omitted the information that each node should be unioned with $ut$. The smallest element in the graph is $ut$ and the largest is $t$. $RDG(t)$ and $RCG(t)$ are related in that they have the same node labels. $RCG(t)$ shows how an object in kind($t$) can change roles, $RDG(t)$ shows the relation between these roles with respect to the $\leq$ ordering.

7. Role-dependent constraints

Some constraints concerning roles are built into our structure and do not need be specified explicitly. The following theorem summarizes a few.
7.1. Theorem

1. \( q \in \text{roles}(\text{kind}(q)) \).
2. \( q \in \text{roles}(k) \Rightarrow q \leq \text{type}(k) \).
3. \( \cup(\text{roles}(k)) \leq \text{type}(k) \).
4. \( q \in \text{roles}(k) \Rightarrow k \subseteq \text{kind}(q) \).
5. \( k_1 \subseteq k_2 \Rightarrow \text{roles}(k_2) \subseteq \text{roles}(k_1) \).
6. \( q_1 \leq q_2 \Rightarrow \text{kind}(q_2) \subseteq \text{kind}(q_1) \).

Proof.

1 and 2 follows immediately from the definition of \( \text{roles} \). 3 and 4 immediately follow from 2 and 5 follows from

\[
q \in \text{roles}(k) \land k_1 \subseteq k_2 \Rightarrow q \in \text{roles}(k_2).
\]

6 is a theorem about types already proven in Wieringa [1987]. □

Part 5 says that roles are inherited downward: If \( \text{Car} \subseteq \text{Vehicle} \), then all attainable roles of Vehicles are attainable roles of Cars. Part 6 says that extension of a role may delimit the kind of things which can play the role. If \( \text{STUDENT} \leq \text{GRAD-STUDENT} \), then all objects which can be graduate students can be students, but we may have that the kind of objects which can be graduate students is a subkind of the kind of objects which may be students. For example, in a 19th century setting only Male persons are graduate students.

Part 1 is not true for all possible roles. \( q \) is not always an element of \( \text{roles}(\text{kind}(q)) \), because this depends upon the repertoire of \( \text{obj}(k) \). For example,

\[
\{\text{stud#}\} \in \text{roles}(\text{kind}([\text{stud#}])) = \text{roles}([\text{Person}]).
\]

Although the set of all possible students is the set of all possible persons, in a particular world there will in general be less students than persons. We would like to know which relation there is between the extension of a kind and the set of existing objects playing a role. To find this out, we must define the extension of a role in a world. We first review the definitions.

A world is a function \( w : s \mapsto \sigma \in \text{space}(\text{type}([s])) \) for all \( s \in S \). In particular a world assigns a state to \( db \in S \) with type \( \{\text{alive, dead}\} \). The surrogates in \( \text{alive}_w(db) \) are said to exist in \( w \).

For each kind \( k \) we defined the extension of \( k \) in \( w \) as the set

\[
k_w = k \cap \text{alive}_w(db).
\]

We now want to define the extension of a role in a world such that the extension of, for example, \( \text{Student} \) is precisely the set of existing identities of objects which have \( \text{Student} \) attributes in their state vector. Now, in

\[
\pi_{\text{Student}}[\text{obj}(\text{Person})]
\]

the objects with an empty state vector are precisely the objects not playing the role of student. Conversely, the set of all possible objects playing the role of student is that subset of \( \pi_{\text{Student}}[\text{obj}(\text{Person})] \) which consists of objects having a non-empty state vector. Intersecting the set of identities of these objects with \( \text{alive}_w(db) \), we get the set of existing persons playing the role of \( \text{STUDENT} \), which is the extension of \( \text{STUDENT} \) in \( w \).
7.2. Definition

1. The function \( \text{players} : Q \rightarrow \mathcal{P}(U) \) is defined by
   \[
   \text{players}(q) = \{(s, \sigma) \mid (s, \sigma) \in \pi_q[U] \land \sigma \neq \epsilon \land q \subseteq \text{dim}(\sigma)\}.
   \]

2. The extension of \( q \) in \( w \) is the set
   \[
   \text{ext}_w(q) = id[\text{players}(q)] \cap \text{alive}_w(db).
   \]

\( \text{players}(q) \) contains all possible objects playing role \( q \), projected on the attributes in \( q \), and \( \text{ext}_w(q) \) is the set of existing identities of objects in \( \text{players}(q) \).

The following theorem formulates the relation between role- and kind extensions.

7.3. Theorem

1. \( id[\text{players}(q)] \subseteq \text{kind}(q) \).
2. If \( \text{kind}(q) = k \), then for all \( w \), \( \text{ext}_w(q) \subseteq k_w \).
3. \( q_1 \subseteq q_2 \Rightarrow \text{ext}_w(q_2) \subseteq \text{ext}_w(q_1) \).

Proof.

1. \( (s, \sigma) \in \pi_q[U] \Rightarrow s \in \text{kind}(q) \), so \( id[\text{players}(q)] \subseteq \text{kind}(q) \).
2. Follows from 1.
3. From 7.1, part 6. \( \square \)

For example, from 1 it follows that the identities of objects playing the role of \( \text{STUDENT} \) are in \( \text{Person} \),

\[
\text{id[players(STUDENT)]} \subseteq \text{kind(STUDENT)} = \text{Person}.
\] (1)

From this we know that in every world all students are persons,

\[
\text{ext}_w(\text{STUDENT}) \subseteq \text{Person}_w.
\] (2)

Thirdly, since \( \text{STUDENT} \leq \text{GRAD-STUDENT} \), we know that in each world \( w \)

\[
\text{ext}_w(\text{GRAD-STUDENT}) \subseteq \text{ext}_w(\text{STUDENT}).
\] (3)

So far, we have considered constraints which follow from the definition of partial and universal attributes only. We now consider role-related constraints which must be specified explicitly. A simple constraint is that only people in the workforce can become employee. We now consider the attribute set

\[
A_2 = \{\text{age}, \text{name}, \text{junior-age}, \text{working-age}, \text{senior-age}, \text{stud#}, \text{courses}, \text{advisor}, \text{emp#}\}
\]

and extend our example event set to

\[
E_2 = \{\text{become-student}, \text{enter-grad-school}, \text{finish-study}, \text{become-emp}, \text{fire}, \text{grow-up}, \text{enter-workforce}, \text{leave-workforce}, \text{become-senior}\},
\]

where the events are defined in the appendix. \( RCG(\text{type}(\text{Person})) \) now becomes much more complicated, as shown in figure 4. Remarks:

1. Since every person has an age, and an age value falls in the range of \( \text{junior-age}, \text{working-age} \) and \( \text{senior-age} \), every person plays role \( \text{JUNIOR}, \text{WORKFORCE} \) or \( \text{SENIOR} \). Assuming a person can be born into our domain in any state, we must connect the root with any other node
by the \textit{birth} event. This is represented by the double arrow leaving from the root.

2. The definition of \textit{become-emp} implies that only people in the workforce can become \textit{Emp}. One can leave a job by resigning or being fired ($e_3$) or retiring ($e_4$). There are complex
synchronization requirements between these events and the aging role-changes.

3. Some events in the graph are individual and some generic. Synchronizations such as $e_1$ and $e_2$ usually require a detailed specification of the synchronization conditions. In the example this is just the state of the object to which they are applied.

8. Actions and events

Role-invariant events are related to the universal type of a kind or to a specific role. In order to specify processes, we must know the difference between the two.

8.1. Definition

If $e$ is role-invariant, then $\text{rl}(e) = \{q_1, \ldots, q_n\}$, $e$ is called an action of roles $q_1, \ldots, q_n$. The actions of $q \in Q$ are

$$\text{acts}(q) = \text{rl}^{-1}(\{q\}).$$

Examples of universal events are $\text{change-name}$ and $\text{inc-age}$, because $\text{rl}(\text{change-name}) = \text{rl}(\text{inc-age}) = \emptyset$. Universal events define the universal role $\emptyset$. Apart from that, they are not connected to any role in particular but belong to a natural kind.

An example of a STUDENT action is

$$\text{enroll-course} : \text{Course} \rightarrow (\text{obj}_{\text{STUDENT}}(\text{Person}) \rightarrow \text{obj}_{\text{STUDENT}}(\text{Person})) =, \quad \text{enroll-course}(c)(p, (\text{stud} : s#, \text{courses} : cc)) = (p, (\text{stud} : s#, \text{courses} : cc \cup \{c\})).$$

We have

$$\text{rl}(\text{enroll-course}) = \{\text{STUDENT} \} \text{ and } \text{enroll-course} \in \text{acts(}\text{STUDENT}').$$

Because we define communication events always in terms of solitary events, we do not have to consider $\text{rl}(e)$ for a communication event and $\text{acts}(e)$ will always contain solitary events (not decomposable in more elementary events).

**Figure 5.** Actions belonging to a role.

If a universal event is to take on a special meaning or extra effect in the case that its subject is playing
In this section, we assume familiarity with process algebra as applied to domain specification in for example Wieringa [1988a, b]. For each natural kind \( k \) we define a *generic process* \( k \) executed by all members of the kind. An individual \( (s, \sigma) \) of kind \( k \) executes an *individual process* which is an instantiation of \( k \), denoted \( \lambda s^\sigma k \), i.e. \( k \) executed by \( s \) starting in state \( \sigma \).

Analogous to the distinction between partial and universal attributes and between roles and natural kinds, we now distinguish *generic processes* executed by all members of a natural kind from *generic plays* executed by all objects playing a role. Because all objects playing a role have a smallest natural kind, we will embed the generic play of a role in the generic process of that kind. This embedding will specify synchronization constraints that enforce the proper entry and exit from a role when certain events or event instances occur (e.g. in our example, enter the role of workforce when age becomes 16). We now show how this is done for our example.

We start by defining a generic process, called a *generic play*, for each role in our specification. A student play can be initially defined as a bag of courses: every course entered in the bag \( \text{enroll-course} \) can be taken out at a later time \( \text{finish-course} \), see figure 6.

\[
\text{STUDENT} = \text{become-student} \cdot \text{STUDY}, \\
\text{STUDY} = (\text{enroll-course} \cdot \text{finish-course}) \parallel \text{STUDY}.
\]

![Figure 6.](image)

The instance of \( \text{STUDENT} \) by \( p \in \text{Person} \) is

\[
\Lambda^{p(x: a, \ldots)}(\text{become-student} \cdot \text{STUDY}) = \\
\sum_{s^\# \in \text{STUDY}} \text{become-student}(p, (\text{age}: a, \text{name}: n)) \cdot \Lambda^{p(x: a, \ldots)}(\text{STUDY}).
\]

There is no condition in the definition of \( \text{become-student} \) that \( s^\# \) should differ from other people’s student numbers, but this can easily be added.

To this bag of courses we add the options to stop the study or to graduate. After graduation, the student can stop or continue with a graduate study (figure 7). There is no constraint on the order or nature of the courses followed; a student can graduate without ever having followed a course. The graduate study is modeled as a competitive publish-or-perish process (figure 8).

\[
\text{STUDENT} = \text{become-student} \cdot \text{STUDY}, \\
\text{STUDY} = (\text{enroll-course} \cdot \text{finish-course}) \parallel \text{STUDY} + \text{graduate} \cdot (\text{stop-study} + \text{GRAD-STUDY}) + \text{stop-study},
\]
GRAD-STUDY = enter-grad-school · PUBLISH-OR-PERISH,  
PUBLISH-OR-PERISH = publish · PUBLISH-OR-PERISH + perish + stop-study + finish-grad.

An EMP play is an overly simplistic iteration of promotions, as shown in figure 9. Finally, there is the sequence of age roles, JUNIOR, WORKFORCE and SENIOR. The basic aging process is
Figure 9.

GROW = \textit{inc-age} \cdot \textit{GROW}.

The three roles played at different life stages have simple plays,

\begin{align*}
\text{JUNIOR} &= \textit{grow-up}, \\
\text{WORKFORCE} &= \textit{enter-workforce} \cdot \textit{leave-workforce}, \\
\text{SENIOR} &= \textit{become-senior}.
\end{align*}

These plays must be synchronized with the GROW process such that the appropriate role changes occur when the subject becomes 16 years of age and 65 years of age, respectively. If every person is instantiated as a \textit{JUNIOR}, this can be accomplished by encapsulating the parallel composition

\[
\text{AGE} = \text{GROW} \parallel \text{JUNIOR} \parallel \text{WORKFORCE} \parallel \text{SENIOR}
\]

using the communications

\[
\begin{align*}
\textit{inc-age}(p, (\text{age: 15})) \parallel \textit{grow-up}(p, (\text{age: 15})) \parallel \textit{enter-workforce}(p, (\text{age: 15})) &= T, \\
\textit{inc-age}(p, (\text{age: 64})) \parallel \textit{become-senior}(p, (\text{age: 64})) \parallel \textit{leave-workforce}(p, (\text{age: 64})) &= T, \\
\textit{inc-age}(p, (\text{age: 64})) \parallel \textit{become-senior}(p, (\text{age: 64})) \parallel \textit{leave-workforce}(p, (\text{age: 64})) &\parallel \textit{retire}(p, (\text{emp#: e#})) = T,
\end{align*}
\]

and any other synchronization between these events deadlocks.

These equations state conditions on the state vector of the communicating objects. A graph of the instantiated process $\partial_H(\Lambda^\text{JUNIOR}(\text{AGE}))$ is shown in figure 10, where the messages $H$ are defined by

\[
H^8 = \{ \textit{grow-up}, \textit{enter-workforce}, \textit{leave-workforce}, \textit{become-senior}, \textit{retire} \},
\]

\[
H = \Lambda H^8 \cup \{ \textit{inc-age}(p, (\text{age: 15})), \textit{inc-age}(p, (\text{age: 64})) | p \in \text{Person} \}.
\]

The set $H^8$ of generic encapsulated events contains all role changes defined by restriction attributes, because these must be synchronized with update events of the unrestricted attribute. For example, \textit{grow-up} is in $H^8$ because it must be synchronized with \textit{inc-age}. In the example \textit{retire} is encapsulated as well, because in the example a retirement is always synchronized with a person reaching the age of 65. The instantiations of $H^8$ are the set $\Lambda H^8$. There are more instantiated events encapsulated, viz. the instances of \textit{inc-age} with which \textit{grow-up} etc. must be synchronized.

We must now put the different plays together in a universal generic process executed by each person.
Even in this unencapsulated process, \textit{STUDENT} actions can only be executed when \textit{STUDENT} attributes are present, thus avoiding the problem of updates of non-existing attributes.

To enforce the synchronizations in an individual person process, we must instantiate and encapsulate the process (only the instantiated process can be encapsulated, for synchronizations are defined in terms of instantiated events).

\[ \partial_H \Lambda_{\text{age}} \text{Person} = \partial_H \Lambda_{\text{age}} (\text{birth} \cdot (\text{change-name} \parallel \text{AGE} \parallel \text{STUDENT} \parallel \text{EMP}) \cdot \text{death}). \]
10. Conclusion

We have defined roles as sets of partial attributes which jointly become or cease to be applicable by a role change. Constraints on role changes which concern the roles that are or are not played at the moment of role change can be expressed in the role-changing event. These constraints (and possibilities) are graphically represented in a role change graph. The subset relations between roles are represented by a role decomposition graph; there is no Galois connection between roles and natural kinds as there is between types and natural kinds, but there is one between types and classes. This is natural, for roles are played by objects, not object identities.

Each role has a generic process, called a play, and the plays of all roles playable by objects of a natural kind are composed with the universal process of that kind.

Some topics to be worked out in more detail later include the specification of more complex role-related constraints, such as the constraint that the employee and student roles exclude each other, the specification of a partial ordering on courses, and of requirements for course-entry and for graduation.

Acknowledgements: Thanks are due to Prof. R.P. van de Riet for his detailed comments on the report and to Prof. J.W. Klop for his help with process algebra.

References
Abiteboul S., R. Hull [1987]

Brodie M., J. Mylopoulos, J. Schmidt (eds.) [1984]
On Conceptual Modelling, Springer-Verlag, New York.

Hammer M., D. McLeod [1981]

Herrlich H., M. Hušek [1986]

King R., D. McLeod [1984]

Mylopoulos J., P.A. Bernstein, H.T. Wong [1980]

Wieringa R.J. [1987]

Wieringa R.J. [1988a]

Wieringa R.J. & R.P. van de Riet [1988b]

Appendix: Example University Domain Specification

Atomic data types:

\( \text{NAME} = \text{ADT of printable character strings.} \)

\( \text{STUD\#} = \text{ADT of possible student numbers.} \)

\( \text{EMP\#} = \text{ADT of possible university employee numbers.} \)

\( \mathcal{N}^P = \{ n \in \mathbb{N} \mid P(n) \} \) for a predicate \( P \) on natural numbers.

Natural kinds:

\( \mathcal{X} = \{ \text{Person}, \text{Course}, \text{Paper} \} \)

Universal attributes:

\( \text{age} = [\text{Person} \rightarrow \mathbb{N}] \)

\( \text{name} = [\text{Person} \rightarrow \text{NAME}] \)

Partial attributes:

\( \text{junior-age} = \text{age}^{\text{N}^{16}}, \)

\( \text{working-age} = \text{age}^{\text{N}^{16}-65}, \) where \( 16-65(n) := 16 \leq n \leq 65 \)

\( \text{senior-age} = \text{age}^{\text{N}^{65}}, \)

\( \text{stud\#} = [\text{Person} \downarrow \text{STUD\#}], \)

\( \text{courses} = [\text{Person} \downarrow \mathcal{S}(\text{Course})], \)

\( \text{advisor} = [\text{Person} \downarrow \text{Person}], \)

\( \text{papers} = [\text{Person} \downarrow \mathcal{S}(\text{Paper})] \)

\( \text{emp\#} = [\text{Person} \downarrow \text{EMP\#}] \).

Basic roles:

\( \text{STUDENT} = \{ \text{stud\#, courses} \}, \)

\( \text{GRAD\-STUDENT} = \text{STUDENT} \cup \{ \text{advisor, papers} \}, \)

\( \text{EMP} = \{ \text{emp\#} \}, \)

\( \text{JUNIOR} = \{ \text{junior-age} \}, \)

\( \text{WORK\-FORCE} = \{ \text{working-age} \}, \)

\( \text{SENIOR} = \{ \text{senior-age} \}. \)

Role-invariant events:

\( \text{inc\-age} : \text{obj}_u(\text{Person}) \rightarrow \text{obj}_u(\text{Person}) \equiv \)

\( \text{inc\-age}(p, \sigma) = (p, \sigma(\text{age} \mapsto \text{age} + 1)) \)

\( \text{change\-name} : \text{NAME} \rightarrow (\text{obj}_u(\text{Person}) \rightarrow \text{obj}_u(\text{Person})) \equiv \)
change-name(n_0)(p, (name : n_1)) = (p, name : n_0)

Role changes and actions:

**STUDENT**:

become-student: STUD# x obj(Person) → obj{stud#, courses}(Person)

become-student(st#)(p, e) = (p, (stud#: st#, courses: \{\}))

enrol-course: Course → (objSTUDENT(Person) → objSTUDENT(Person))

enrol-course(c)(p, (stud#: st#, courses: cc)) = (p, (stud#: st#, courses: cc ∪ \{c\}))

finish-course: Course → (objSTUDENT(Person) → objSTUDENT(Person))

finish-course(c)(p, (stud#: st#, courses: cc)) = (p, (stud#: st#, courses: cc − \{c\}))

finish-study: obj(stud#)(Person) → obj(Person)

finish-study(p, (stud#: st#)) = (p, e)

stop-study: obj(stud#)(Person) → obj(Person)

stop-study(p, (stud#: st#)) = (p, e)

**GRAD - STUDENT**:

enter-grad-school: Person → (obj{stud#}(Person) → obj{stud#, advisor, papers}(Person))

enter-grad-school(a)(s, (stud#: st#)) = (s, (stud#: st#, advisor: a, papers: \{\}))

publish: Paper → (obj{papers}(Person) → obj{advisor}(Person))

publish(p)(g, (papers: pp)) = (g, (papers: pp ∪ \{p\})).

perish: obj{stud#, advisor, papers}(Person) → obj(Person)

perish(p, (stud#: st#, advisor: a, papers: pp)) = (p, e).

finish-grad: obj{stud#, advisor, papers}(Person) → obj(Person)

finish-grad(p, (stud#: st#, advisor: a, papers: pp)) = (p, e).

**EMP**:

become-emp: EMP# → (obj{working-age}(Person) → obj{emp#}(Person))

become-emp(e#)(p, (working-age: n)) = (p, (working-age: n, emp#: e#))

resign: obj{emp#}(Person) → obj(Person)

resign(p, (emp#: e#)) = (p, e)

fire: obj{emp#}(Person) → obj(Person)
\[
\text{fire}(p, (\text{emp#}: e#)) = (p, \varepsilon)
\]

\[
\text{retire} : \text{obj}_{\text{emp#}}(\text{Person}) \rightarrow \text{obj}_\varnothing(\text{Person}) = \\
\text{retire}(p, (\text{emp#}: e#)) = (p, \varepsilon)
\]

**JUNIOR:**

\[
\text{grow-up} : \text{obj}_{\text{junior-age}}(\text{Person}) \rightarrow \text{obj}_\varnothing(\text{Person}) = \\
\text{grow-up}(p, (\text{junior-age} : 15)) = (p, \varepsilon)
\]

**WORK-FORCE:**

\[
\text{enter-workforce} : \text{obj}_\varnothing(\text{Person}) \rightarrow \text{obj}_{\text{working-age}}(\text{Person}) = \\
\text{enter-workforce}(p, \varepsilon) = (p, (\text{working-age} : 16))
\]

\[
\text{leave-workforce} : \text{obj}_{\text{working-age}}(\text{Person}) \rightarrow \text{obj}_\varnothing(\text{Person}) = \\
\text{leave-workforce}(p, (\text{working-age} : 64)) = (p, \varepsilon).
\]

**SENIOR:**

\[
\text{become-senior} : \text{obj}_\varnothing(\text{Person}) \rightarrow \text{obj}_{\text{senior-age}}(\text{Person}) = \\
\text{become-senior}(p, \varepsilon) = (p, \text{senior-age} : 65)).
\]

**Synchronizations:**

\[
H^8 = \{\text{grow-up, enter-workforce, leave-workforce, become-senior, retire}\}
\]

\[
H = AH^8 \cup \{\text{inc-age}(p, (\text{age} : 15)), \text{inc-age}(p, (\text{age} : 64)) | p \in \text{Person} \}.
\]

\[
\text{inc-age}(p, (\text{age} : 15)) \mid \text{grow-up}(p, (\text{age} : 15)) \mid \text{enter-workforce}(p, (\text{age} : 15)) = T, \\
\text{inc-age}(p, (\text{age} : 64)) \mid \text{become-senior}(p, (\text{age} : 64)) \mid \text{leave-workforce}(p, (\text{age} : 64)) = T, \\
\text{inc-age}(p, (\text{age} : 64)) \mid \text{become-senior}(p, (\text{age} : 64)) \mid \text{leave-workforce}(p, (\text{age} : 64)) \mid \text{retire}(p, (\text{emp#}: e#)) = T,
\]

and any other synchronization between these events deadlocks.