Organising an Information System as Stratified Hypermedia

C.A.J. Burgers, H.A. Proper and Th.P. van der Weide

Computing Science Institute, University of Nijmegen,
Toernooiveld, NL-6525 ED Nijmegen, The Netherlands,
{coenb,erikp,tvdw}@cs.kun.nl

PUBLISHED AS:

C.A.J. Burgers, H.A. Proper, and Th.P. van der Weide. Organising an Information System as Stratified Hypermedia. In H.A. Wijshoff, editor, *Proceedings of the Computing Science in the Netherlands Conference*, pages 109–120, Utrecht, The Netherlands, EU, November 1993.

Abstract

In this paper we investigate a relation between modern hypertext approaches and conventional data modelling techniques, such as PSM. This relation provides us with the mechanism of query by navigation, in order to facilitate the query formulation problem.

1 Introduction

The information disclosure problem begins with a person having an information need that they wish to fulfil (figure 1). Formulation of this need leads to a *request* q, which has to be matched against the *characterization* of *information objects* which are available in the information base (also referred to as information carriers or documents). This process of *matching* yields a measure for the relevance of objects to the request. The formulation of a request is known to be a cause of problems ([SM83]).

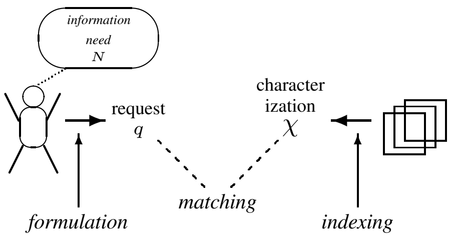


Figure 1: The information disclosure paradigm

Next we focus on information modelling problems, in which the analyst has to deal with information structures of sizes, matching that of wallpaper (even for medium sized modellings problems). Such structures easily result in diagrams which are unreadable for both domain experts and system analysts. If a user wants to know something about objects in this information structure, it is very likely that they have problems in formulating an adequate request. As a consequence, they will retrieve irrelevant (or even wrong) objects and may miss relevant objects. Retrieving irrelevant objects leads to a low *precision*, missing relevant objects has a negative impact on the *recall* ([SM83]).

	relevant objects	irrelevant objects
retrieved objects	ideal	precision
unretrieved objects	recall	ideal

A way to overcome the query formulation problem is the concept of *query by navigation*, allowing the user to navigate through the information characterizations, and meanwhile formulating their request interactively. This interaction mechanism between a searcher and the system is well-known from Information Retrieval, and has proven to be usefull ([BW92b, Bru93]).

The Predicator Set Modelling technique (PSM) has been introduced as a common denominator for object-role modelling techniques, covering well-known techniques such as ER ([Che76]) and NIAM ([NH89]). This makes the presented ideas applicable to object-role modelling techniques in general. Note that the approach is related to the natural language paradigm of NIAM.

In this paper a PSM information structure will be represented as a stratified hypermedia architecture, leading to an implementation of the concept of query by navigation in the context of information modelling. Stratified hypermedia architecture ([BW92b]) features a descriptive level, the so-called *hyperindex*, comprising a hypertext of characterizations, indexing the lower level, the so-called *hyperbase*. The hyperbase contains the actual information. Stratified hypermedia architecture offers, by seperating description and instantiation, the possibility to formulate information needs by an interactive process of navigation through the descriptive level.

As a result, the hyperindex provides a domain, both for the characterization and query language. The characterization function χ relates hyperindex and hyperbase, and forms the basis for determining the relevancy (matching) of information objects (hyperbase) to a given query (descriptor from the hyperindex).

2 Traversing an information structure

Before describing a data modelling technique such as PSM as a stratified hypermedia architecture, we present a demonstration of the benefits of the resulting system. In this paper, we concentrate on query by navigation, and therefore *omit* how this architecture can be helpful during the construction process of information systems. For a description of this latter use, see [HPW92] and [HPW93].

The examples in this section are based on the so-called presidential database, which served as a unified example in a special issue of Computing Surveys [FS76]. The example was first enuntiated in [WBGW73]. An excerpt of the presidential database is shown in figure 2. The examples show how the system supports the formulation of queries. The process of query formulation corresponds to a search through the information system in order to fulfil some information need. The request of the searcher is formulated by stepwise refining or enlarging the current description (the *focus*) of this need, until the searcher recognises the current description as sufficiently describing (part of) the information need. Usually the best description will result from a set of such descriptions (a so-called *guide*).

The examples, in this paragraph, give an idea of what a searcher subsequently has to do, and what screens they will encounter, when formulating their information need. Each screen contains, in its header, the current focus, and in its body the direct environment of the current focus. The first example, figure 3, shows the screen which corresponds to the starting point of a search in which the searcher has not yet revealed anything. This screen will be referred to as the *starting node* of the system.
The starting node contains all object types of the information structure at hand. The searcher can now choose one of the objects as focus for further processing. This selection then is the first refinement of the searcher's information need. A selection of an item in a node is denoted by the symbol $\leftarrow$ in the figures. The symbol $\bigtriangledown$ is a button for a refinement step, while $\bigtriangleup$ is used for an enlargement step. Finally $\rhd$ is a button used for an associative link.

In the second example, figure 4, the searcher wants to find those presidents married with someone involved in politics. The search begins with the starting node. We describe only one path leading to a descriptor of his information need in which the searcher is directly heading for the goal, without any backtracking. The searcher starts with selecting *president* as focus. The associated screen shows about the direct environment of *president*. Thereupon the searcher selects *president is married* as next focus. In the resulting screen the searcher continues with *president is married with person*. The screen associated with this focus shows the sentence *president is married with politician*, which is a proper description of the original information need. Now the searcher can satisfy their information need by requiring the result of the request. The system will present the answer in a standard tabular format, or in the following more verbose form:

president x is married with politician y

where x denotes a president and y a politician.

In the third example, figure 5, the searcher is interested in the birthyear and age of death of some person. In order to find these properties of some persons the searcher first has to choose *president*, as the birthyear and age of death are only recorded for presidents. Now the searcher can refine their focus by choosing either *president has birthyear* or *president has died* as focus for further processing. When selecting the former as focus, using A, the sentence *president has birthyear in year* appears. This satisfies the first part of the information need. In order to satisfy the second part, the searcher must create another context. They now focus, using B, on the age of death of persons. After choosing *president has died at age*, they have fulfilled their information need and can require the respons by this guide. The system will present all answers in the following verbose way:

president x has birthyear y,
president x has died at age z

where x denotes a president, y a birthyear and z an age of death.

3 PSM represented as stratified hypermedia

In order to represent PSM as a stratified hypermedia architecture, a PSM information structure is divided into two levels of abstraction first. For more details on PSM, the reader may refer to [HW93] or [HPW93].

The *concrete level* consists of all concrete object types, i.e. label types. The non concrete object types form the *abstract level*, and corresponds to a defoliated information structure (see [BW92a]). This level thus consists of the following components:

- entity types ($\mathcal{E}$), fact types ($\mathcal{F}$),
  power types ($\mathcal{G}$), sequence types ($\mathcal{S}$),
  schema types ($\mathcal{C}$)
- generalisation (Gen), specialisation (Spec)

These two levels of abstraction can be transformed into the layers constituting a stratified hypermedia architecture (figure 6). The hyperbase is constructed from the instantiation of an information structure (its population). The instantiations of label types form the fragments of the hyperbase ($\mathbb{F}_b$), while the instantiations of abstract object types form the molecules ($\mathbb{M}_b$).

Object role models allow the verbalisation of queries by so-called path expressions, built from names of object types and role names. These path expressions are used to construct the hyperindex. For example in figure 7, the expression

$$\mathsf{Person\ being\_manager\_of\ Project\ having\_budget\ 10000}$$

describes all persons which are manager of some project with a budget of 10000 dollars. Rather than role names (such as being_manager_of), we will use predicator names (such as is and has in figure 7). A role name can be seen as a representation mechanism for (binary) fact types. For example, role name being_manager_of corresponds to the representation of fact type Manager by deep structure sentences [NH89] of the form:

$$\langle \mathsf{Person} \rangle \text{ being\_manager\_of } \langle \mathsf{Project} \rangle$$

A role name also corresponds to a connector [HPW93] through fact type $\mathsf{Manager}$. In non-binary fact types, however, connectors can not be uniquely derived from a unique predicator, as predicators in this case does not have a unique co-role. For this reason predicator names are used to uniquely identify paths.

Predicators and object types are the elementary parts from which the set of *linear path expressions* ($\mathcal{PE}_{lin}$) is constructed. The hyperindex is constructed from this set $\mathcal{PE}_{lin}$ in the obvious way:

- the elementary paths form the fragments ($\mathbb{F}_i$) of the hyperindex,
- the others form the molecules ($\mathbb{M}_i$).

In the remainder of this section, we will describe this in more detail.

3.1 Constructing the hyperindex

In [HPW93] the concept of path expression is introduced. A path expression corresponds to a path through the information structure, via predicators, beginning and ending in some object type. In this paper, we will restrict ourselves to linear paths. Linear path expressions are constructed from object types $O \in \mathcal{O}$ and predicators $p \in \mathcal{P}$.

Predicator p corresponds to a path from object type $\mathsf{Base}(p)$ to object type $\mathsf{Fact}(p)$. For example, in figure 7, $\mathsf{Base}(\mathsf{of}) = \mathsf{Budget}$ and $\mathsf{Fact}(\mathsf{of}) = \mathsf{Budgets}$. The reverse path is denoted as $p^{\leftarrow}$. Note that a predicator forms a connection between two object types. Linear path expressions are concatenated using the composition operator $\circ$. For instance in figure 7:

$$\mathsf{Person} \circ \mathsf{is} \circ \mathsf{Manager} \circ \mathsf{Manager.has}^{\leftarrow} \circ \mathsf{Project}$$

is a path from Person to $\mathsf{Project}$.

In the above example, object type Convoy is a so-called power type with underlying *element type* Ship. A power type is instantiated with subsets of the (current) instantiation of the underlying element type. The fact type connecting a power type with its element type is usually omitted in the information structure. Such a fact type is therefore called an *implicit fact type*. In the hyperindex, however, implicit fact types (and their implicit predicators) are taken into account.

Sample sessions in the hyperindex

At this point the basic concepts of the hyperindex layer are defined. In the remainder of this subsection some examples are given, demonstrating the mechanism of query by navigation.

The following example shows how to deal with objectification. An objectified fact type can be:

- used as an entity type, abstracting from the composition into predicators,
- used as a fact type, taking the decomposition into predicators into account.

The hyperindex makes no difference between both views, the continuation of the linear path leads for both views to an extension with a predicator and an object. As a concrete example, consider figure 9. When arriving in $\rho(\mathsf{Part} \circ \mathsf{is} \circ \mathsf{Used})$, the system shows as possible continuations of this path $\rho(\mathsf{Part} \circ \mathsf{is} \circ \mathsf{Used} \circ \mathsf{in}^{\leftarrow} \circ \mathsf{Project})$, using Used as a fact type, and $\rho(\mathsf{Part} \circ \mathsf{is} \circ \mathsf{Used} \circ \mathsf{and} \circ \mathsf{Supplied})$, using Used as an entity type. The function ρ is a represents linear path expressions as readable sentences. This function will be introduced in section 4. Note that unification of these views is not generally applicable. In subsection 3.3 it is shown how these different views lead to a different treatment in case of characterization.

Two PSM schema's, containing generalisation and specialisation respectively, are shown in figure 10. In order to demonstrate the differences between specialisation and generalisation, figure 10 shows a part of the hyperindex, located around the starting molecule. From this example, the assignment of the associative links, and the constructed path expressions will become clear. When generalisation is involved, the extension of A with $r \circ g$ results in a substitution of A by B. This substitution denotes that instances of A, playing g, are actually instances of B. This substitution origines from the fact that generalised objects inherits all of their properties of its specifiers [HW93].

In specialisation, subtypes inherit their properties from its associated supertype(s) [HW93]. A path expression however is represented by the most specialised name, so in the representation $\rho(B \circ p \circ f)$, B is *not* substituted by A.

3.2 Constructing the hyperbase

The translation of an instantiation of a PSM information structure into a hyperbase will be done bottom-up. The fragment base ($\mathbb{F}_b$) is defined first, followed respectively by the node base ($\mathbb{N}_b$), the schema ($\mathbb{G}_b$) and the views ($\mathbb{V}_b$).

Instances of abstract object types are composed values. Such instances are represented as structural elements in the hyperbase, thus contributing to the set $\mathbb{M}_b$ of molecules. These, however, are not the only molecules in the hyperbase. For fact types with arity > 2, it will be usefull to have disposal of all non-empty projections $\wp^+(f)$ of each instance of f. This is motivated by the possibility that a path expression walks via a connector through part of a fact type. This will occur when the searcher is not interested in the complete fact instance.

Besides, it will be usefull to have disposal of joins of (different) molecules. Two molecules can be joined if they share some sub-molecule. This makes it possible to deal with the construction of linear paths over multiple facts. If a searcher traverses several fact types subsequently, and then performs a beam-down, the hypermedia will return the combination of the associated fact type instances as a result.

3.3 Characterization

In this section, we relate hyperbase and hyperindex by the introduction of a characterization χ. This function maps information objects (fragments, nodes, molecules) from the hyperbase onto the fragments of the hyperindex [BW92b].

The characterization function will be defined bottom up, starting with the fragments of the $\mathbb{F}_b$. Fragment $y \in \mathbb{F}_b$ is characterized by all entity types that are associated to y via a bridge type. As a corollary, note that subtyping and generalisation is covered by this rule as follows:

- $X \,\mathsf{Spec}\, Y \wedge X \in \chi(x) \Rightarrow Y \in \chi(x)$
- $X \,\mathsf{Gen}\, Y \wedge Y \in \chi(x) \Rightarrow X \in \chi(x)$

The characterization of node N is directly defined in terms of the contained in its presentation. If, however, N presents a molecule containing an instance of a power type, a sequence type or a schema type then the characterization has to deal with it differently. This because these object types are presented using fragments who are concretisations of their elementary types. The characterization of a node N can be defined using the following four rules. The characterization of a node containing an instance of a power type G yields:

$$\chi(\sigma\, \pi_b(\{x_1, \ldots, x_n\})\tau) = \chi(\sigma) \cup \{G\} \cup \chi(\tau)$$

where π_b is the presentation function for molecules from the hyperbase b, assigning to each molecule a sequence of fragments. If a node contains an instance of a sequence type S then the characterization yields:

$$\chi(\sigma\, \pi_b(\langle x_1, \ldots, x_n\rangle)\tau) = \chi(\sigma) \cup \{S\} \cup \chi(\tau)$$

The characterization of a node containing an instance of a schema type C yields:

$$\chi(\sigma\, \pi_b(v)\tau) = \chi(\sigma) \cup \{C\} \cup \chi(\tau)$$

In all the other cases the characterization of a node yields:

$$\chi(N) = \bigcup_{f \in N} \chi(f)$$

For instance, the characterization of

project 1026 has a budget of 10000 dollars

yields $\{$Project, Budget$\}$, because:

$$\chi(\text{'project'}) \quad = \quad \varnothing$$

$$\begin{array}{rcl}\chi(1026) & = & \{\text{Project}\} \\ \chi(\text{`has a budget of'}) & = & \varnothing \\ \chi(10000) & = & \{\text{Budget}\} \\ \chi(\text{`dollars'}) & = & \varnothing\end{array}$$

Note that the structure of the presentation of a molecule is not taken into account.

The characterization of molecule x, representing an instance of object type X is found in two steps. Firstly, the characterization of the presentation $\pi_b(x)$ is taken into account:

$$\chi_w(x) = \chi(\pi_b(x))$$

This is known as the weak characterization of x. Secondly, the properties of x are included (strong characterization). These properties are found by the predicators based on X.

$$\chi_s(x) = \{p \in \mathcal{P} \mid t(p) = X \vee p \in X\}$$

where t is an instance of some fact type f.

Molecules representing joins are characterized by the union of the characterizations of their sub-molecules.

3.4 The beam-down operation

Now that the characterization of the hyperbase molecules is completed, we focus on the beam-down operation. The beam-down operation transfers the searcher from an expression in the hyperindex to the result of that expression, i.e. molecule(s) in the hyperbase. The hyperbase molecules are characterized using the fragments within the hyperindex ($\mathbb{F}_i$), so, in order to establish a beam-down result, the molecules of the hyperindex ($\mathbb{M}_i$) must also be described using these fragments.

The function *dissect* (Di) yields for each linear path expression its associated set of fragments:

$$\mathsf{Di} : \mathcal{PE}_{lin} \rightarrow \wp(\mathbb{F}_i)$$

The function dissect is defined inductively according to the structure of linear path expressions.

- An object X is dissected in its own name and the entity types on its identification path. The entity types on an identification path of object type X is denoted by $\mathsf{Id}(X)$. A fact type X however is dissected in its predicators and the bases of those predicators. If a fact type X is connected to an objectified fact type Y then the dissection of Y is also included in the dissection of X, so:

$$\mathsf{Di}(X) = \begin{cases} \bigcup_{p \in X} \left(\{p\} \cup \mathsf{Di}(\mathsf{Base}(p)) \right) & \text{if } X \text{ is a fact type} \\ \{X\} \cup \bigcup_{y \in \mathsf{Id}(X)} \{y\} & \text{otherwise} \end{cases}$$

- The dissection of a linear path must deal with the possibility that this path contains objectified fact types (used as entity types). If some path Y contains an objectified fact type Z, used as an entity type, then the dissection of Y includes the dissection of Z also, so:

$$\mathsf{Di}(P \circ p \circ \mathsf{Fact}(p)) = \begin{cases} \mathsf{Di}(P) \cup \{p\} \\ \qquad \cup\, \mathsf{Di}(\mathsf{Base}(p)) & \text{if } \mathsf{Base}(p) \in \mathcal{F} \\ \mathsf{Di}(P) \cup \{p\} & \text{otherwise} \end{cases}$$

- If a linear path P is extended with a predicator p and an object type $\mathsf{Base}(p)$, then $p^{\leftarrow}$ and the dissection of $\mathsf{Base}(p)$ are added to the dissection of P.

  Because there is no difference between connecting two object types using a p or $p^{\leftarrow}$, $p^{\leftarrow}$ can be substituted (in the result) by p, so:

$$\begin{aligned} &\mathsf{Di}(P \circ p^{\leftarrow} \circ \mathsf{Base}(p)) \\ &\quad = \mathsf{Di}(P) \cup \{p\} \cup \mathsf{Di}(\mathsf{Base}(p)) \end{aligned}$$

The BeamDown operator, on some guide G, retrieves hyperbase molecules in the following way. The function Rel calculates the relevancy of hyperbase molecules for the actual guide. This function yields a pair as a result, the first component denoting the relevancy of the object types from the dissection of the guide, the second that of the predicates. Only hyperbase molecules that exceed some treshold ϵ will be retrieved, and are presented in their (alphabetic) order of relevance.

$$\mathsf{BeamDown}(G) = \{M \in \mathbb{M}_b \mid \mathsf{Rel}(M, G) > \epsilon\}$$

The calculation of the relevancy of some hyperbase molecule M for some guide G yields:

$$\begin{aligned} &\mathsf{Rel}(M, G) = \\ &\quad (\text{relevance object types}, \ \text{relevance predicators}) \end{aligned}$$

The calculation of the relevance of the object types takes type relatedness into account. For each object type X, in some characterization or dissection, only $\sqcap(X)$ is regarded relevant for the calculation. This leads to the following rule:

$$\begin{aligned} &\text{relevance object types} \\ &\quad = \frac{\left| \left(\mathsf{Cr}(M) \cap \mathsf{Pr}(G) \right) \cup \mathsf{Corr}(M) \right|}{\left| \mathsf{Cr}(M) \cup \mathsf{Pr}(G) \right|} \end{aligned}$$

The function Cr is based on the weak characterization (containing object types):

$$\mathsf{Cr}(M) = \sqcap(\chi_w(M))$$

Similarly, the function Pr yields the set containing the pater familias of each object type from the dissection of G:

$$\mathsf{Pr}(G) = \sqcap \left(\bigcup_{p \in G} \mathsf{Di}(p) \cap \mathcal{O} \right)$$

The function Corr corrects for noise, resulting from multiple entity type instances being represented by a single label type instance. For example in figure 2, object types Politician and President are both identified by label type Person-name.

$$\mathsf{Corr}(M) = \{X \mid \exists_{x \in \mathsf{Pop}(X) \wedge y \in \pi_b(M)} [\mathsf{Concr}(x) = y]\}$$

where $\mathsf{Concr}(x)$ denotes the concrete value used for representing abstract instance x.

The calculation of the relevance of the predicators is done analogously, with the exception that no noise occurs in this case.

$$\text{relevance predicators} = \frac{\left| \mathsf{Cp}(M) \cap \mathsf{Pp}(G) \right|}{\left| \mathsf{Cp}(M) \cup \mathsf{Pp}(G) \right|}$$

The function Cp is defined by:

$$\mathsf{Cp}(M) = \chi_s(M)$$

The predicators, associated with the dissection of a guide, are found by:

$$\mathsf{Pp}(G) = \bigcup_{p \in G} \mathsf{Di}(p) \cap \mathcal{P}$$

Beam-down example

In this paragraph we will give two examples of the beam-down process. Consider the following linear path expression in figure 8:

$$\mathsf{Project} \circ \mathsf{Budgets.has} \circ \mathsf{Budgets}$$

The dissection belonging to this path expression is:

$$\begin{aligned}&\mathsf{Di}(\mathsf{Project} \ \circ \ \mathsf{Budgets.has} \ \circ \ \mathsf{Budgets}) = \\ &\quad \big\{\mathsf{Project}, \mathsf{Budgets.has}\big\}\end{aligned}$$

Suppose Budgets and Manager are populated as follows:

$$\begin{aligned}&\mathsf{Pop}(\mathsf{Budgets}) \\ &\quad = \quad \big\{\big\{\mathsf{of}: e_1, \mathsf{has}: c_1\big\}, \big\{\mathsf{of}: e_2, \mathsf{has}: c_2\big\}\big\} \\ &\mathsf{Pop}(\mathsf{Manager}) \\ &\quad = \quad \big\{\big\{\mathsf{is}: d_2, \mathsf{has}: c_2\big\}, \big\{\mathsf{is}: d_1, \mathsf{has}: c_3\big\}\big\}\end{aligned}$$

The characterizations of some hyperbase molecules are:

- $\chi(\langle c_1 \rangle) = \{\mathsf{Project}, \mathsf{Budgets.has}\}$
- $\chi(\langle c_2 \rangle) = \{\mathsf{Project}, \mathsf{Budgets.has}, \mathsf{Manager.has}\}$
- $\chi(\langle c_3 \rangle) = \{\mathsf{Project}, \mathsf{Manager.has}\}$
- $\chi(\langle d_2 \rangle) = \{\mathsf{Person}, \mathsf{is}\}$

When performing the beam-down operation on guide $G = \big\{\mathsf{Project} \ \circ \ \mathsf{Budgets.has} \ \circ \ \mathsf{Budgets}\big\}$, the hypermedia system computes the relevancy of these molecules with respect to this guide. Thus:

$$\begin{aligned}\mathsf{Rel}(\langle c_1 \rangle, G) &= (1, 1) \\ \mathsf{Rel}(\langle c_2 \rangle, G) &= (1, \frac{1}{2}) \\ \mathsf{Rel}(\langle c_3 \rangle, G) &= (1, 0) \\ \mathsf{Rel}(\langle d_2 \rangle, G) &= (0, 0)\end{aligned}$$

Using $(\frac{1}{2}, \frac{1}{4})$ as a treshold, only molecules $\langle c_1 \rangle$, $\langle c_2 \rangle$ are retrieved. These molecules are presented respectively as $\pi_b(c_1)$ and $\pi_b(c_2)$ to the searcher.

In the next example, we consider the following guide:

$$\begin{aligned}&G = \\ &\quad \{\mathsf{Project} \circ \mathsf{Budgets.has} \circ \mathsf{Budgets}, \\ &\quad\ \ \mathsf{Project} \circ \mathsf{Manager.has} \circ \mathsf{Manager} \circ \mathsf{is}^{\leftarrow} \circ \mathsf{Person}\}\end{aligned}$$

The dissection of this guide is:

$$\begin{aligned}&\mathsf{Di}(G) = \\ &\quad \big\{\mathsf{Project}, \mathsf{Budgets.has}, \mathsf{Manager.has}, \mathsf{is}, \mathsf{Person}\big\}\end{aligned}$$

The relevancy of some molecules to this guide is:

$$\begin{aligned}\mathsf{Rel}(\langle c_1 \rangle, G) &= (\frac{1}{2}, \frac{1}{3}) \\ \mathsf{Rel}(\langle c_2 \rangle, G) &= (\frac{1}{2}, \frac{2}{3}) \\ \mathsf{Rel}(\langle c_3 \rangle, G) &= (\frac{1}{2}, \frac{1}{3}) \\ \mathsf{Rel}(\langle d_2 \rangle, G) &= (\frac{1}{2}, \frac{1}{3})\end{aligned}$$

Using as treshold $(\frac{2}{3}, \frac{1}{4})$, none of these molecules will be retrieved. However, the retrieval result will not be empty for this guide. For example, the relevancy of $\langle c_2 d_2 \rangle$ is (1,1), which clearly exceeds the treshold, and will be presented as $\pi_b(c_2 d_2)$.

4 Presenting linear path expressions

In this section the function ρ is introduced. This function maps linear path expressions into readable sentences. Before defining ρ we introduce the concepts of *object type naming* and *predicator naming*.

4.1 Object type and predicator naming

In order to verbalise linear path expressions, we must first verbalise object types and predicators. Explicit object types are referenced by a unique name:

$$\mathsf{ONm} : \mathcal{O} \rightarrow \mathcal{N}_s$$

Explicit predicators may have assigned a so-called *predicator name* via the (partial) function:

$$\mathsf{PNm} : \mathcal{P} \cup \mathcal{P}^{\leftarrow} \rightarrow \mathcal{N}_s$$

These naming functions, however, do not assign names for implicit object types and implicit predicators. For the latter categories, a general naming mechanism (using keywords) is introduced. For instance in figure 7, the predicator q could be verbalised in two directions as follows:

$$\begin{aligned} \mathsf{PNm}(\mathsf{Manager.has}) &= \text{has} \\ \mathsf{PNm}(\mathsf{Manager.has}^{\leftarrow}) &= \text{of} \end{aligned}$$

The predicator name of a predicator may yield an empty string. The motivation is that an empty string sometimes enables us to present better readable sentences.

4.2 Verbalising linear paths

Linear path expressions are presented as readable sentences via the function: $\rho : \mathcal{PE}_{lin} \rightarrow \mathcal{N}_s$. This function is defined recursively as follows:

$$\rho(O) = \mathsf{ONm}(O)$$

$$\rho(P \circ p \circ X) = \begin{cases} \rho(P)\ \mathsf{PNm}(p)\ \mathsf{ONm}(X) \\ \quad \text{if } p \text{ and } X \text{ have names} \\ \rho(P)\ \textit{involved in}\mathsf{ONm}(X) \\ \quad \text{if } p \text{ has no name, and } X \text{ has} \\ \text{some keyword, see below} \\ \quad \text{otherwise} \end{cases}$$

$$\rho(P \circ p^{\leftarrow} \circ X) = \begin{cases} \rho(P)\ \mathsf{PNm}(p^{\leftarrow})\ \mathsf{ONm}(X) \\ \quad \text{if } p^{\leftarrow} \text{ and } X \text{ have names} \\ \rho(P)\ \textit{of}\,\mathsf{ONm}(X) \\ \quad \text{if } p^{\leftarrow} \text{ has no name, and } X \text{ has} \\ \text{some keyword} \\ \quad \text{otherwise} \end{cases}$$

For instance in figure 7:

$$\begin{aligned}
&\rho(\text{Project} \circ \mathsf{Manager.has} \circ \text{Manager}) \\
&\quad = \mathsf{ONm}(\text{Project})\ \mathsf{PNm}(\mathsf{Manager.has}) \\
&\qquad \mathsf{ONm}(\text{Manager}) \\
&\quad = \text{project has manager}
\end{aligned}$$

Now we will introduce keywords. Keywords serve as a naming mechanism for handling implicit object types and predicators. In figures 11, 12, 13, and 14 the keywords are summarised. For each keyword an arrow denotes the path associated with that keyword. (For more information about keywords see [HPW93].)

These keywords also can be used when no predicator names are defined. For instance, if in figure 7 $\mathsf{PNm}(\mathsf{Manager.has}^{\leftarrow})$ is not defined:

$$\begin{aligned}
&\rho(\mathsf{Manager} \circ \mathsf{Manager.has}^{\leftarrow} \circ \mathsf{Project}) \\
&\quad = \text{manager } \textit{of} \text{ project}
\end{aligned}$$

When dealing with implicit facts it can be usefull to use keywords instead of defining predicator names and the name of the implicit fact. If in figure 12:

$$\begin{aligned}
\mathsf{ONm}(\mathsf{Convoy}) &= \text{convoy} \\
\mathsf{ONm}(\mathsf{Ship}) &= \text{ship}
\end{aligned}$$

Then the denotation of the following linear paths yields:

$$\begin{aligned}
&\rho(\mathsf{Convoy} \circ \in^{c}_{\mathsf{Convoy,Ship}} \circ \in_{\mathsf{Convoy,Ship}}) \\
&\quad = \text{convoy } \textit{has elements}
\end{aligned}$$

$$\begin{aligned}
&\rho(\mathsf{Ship} \circ \in^{e}_{\mathsf{Convoy,Ship}} \circ \in_{\mathsf{Convoy,Ship}} \circ \\
&\quad (\in^{p}_{\mathsf{Convoy,Ship}})^{\leftarrow} \circ \mathsf{Convoy}) \\
&\quad = \text{ship } \textit{in} \text{ convoy}
\end{aligned}$$

5 Conclusions

In this paper we investigated an integration between stratified hypermedia architecture and information systems. This integration makes it possible to use the concept of query by navigation in information structures based on object-role modelling techniques. Due to this integration a user can interactively formulate a query.

The presented mechanism offers several opportunities for further refinement. Firstly, at the moment a guide connects the comprising linear path expressions like conjunction for boolean expressions. It might be worthwile to investigate the effects of an extension to full boolean retrieval for combining linear path expressions.

Secondly, the extension to (general) path expressions seems to be a natural extension of this approach.

Thirdly, applying this mechanism on other database query languages (such as SQL) may be investigated.

References

[Bru93] P.D. Bruza. *Stratified Information Disclosure: A Synthesis between Information Retrieval and Hypermedia*. PhD thesis, University of Nijmegen, Nijmegen, The Netherlands, EU, 1993.

[BW92a] P. van Bommel and Th.P. van der Weide. Reducing the search space for conceptual schema transformation. *Data & Knowledge Engineering*, 8:269–292, 1992.

[BW92b] P.D. Bruza and Th.P. van der Weide. Stratified Hypermedia Structures for Information Disclosure. *The Computer Journal*, 35(3):208–220, 1992.

[Che76] P.P. Chen. The entity-relationship model: Towards a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, March 1976.

[FS76] J.P. Fry and E.H. Sibley. Evolution of Data-Base Management Systems. *Computing Surveys*, 8(1):7–42, 1976.

[HPW92] A.H.M. ter Hofstede, H.A. Proper, and Th.P. van der Weide. Data Modelling in Complex Application Domains. In P. Loucopoulos, editor, *Proceedings of the Fourth International Conference CAiSE'92 on Advanced Information Systems Engineering*, volume 593 of *Lecture Notes in Computer Science*, pages 364–377, Manchester, United Kingdom, EU, May 1992. Springer Verlag, Berlin, Germany, EU. ISBN 3540554815

[HPW93] A.H.M. ter Hofstede, H.A. Proper, and Th.P. van der Weide. Formal definition of a conceptual language for the description and manipulation of information models. *Information Systems*, 18(7):489–523, October 1993.

[HW93] A.H.M. ter Hofstede and Th.P. van der Weide. Expressiveness in conceptual data modelling. *Data & Knowledge Engineering*, 10(1):65–100, February 1993.

[NH89] G.M. Nijssen and T.A. Halpin. *Conceptual Schema and Relational Database Design: a fact oriented approach*. Prentice-Hall, Sydney, Australia, 1989. ASIN 0131672630

[SM83] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill New York, NY, 1983.

[WBGW73] S.E. Willner, A.E. Bandurski, W.C. Gorhan, and M.A. Wallace. COMRADE data management system. In *Proceedings of the AFIPS National Computer Conference*, pages 339–345, Montvale, New Jersey, 1973. AFIPS Press.

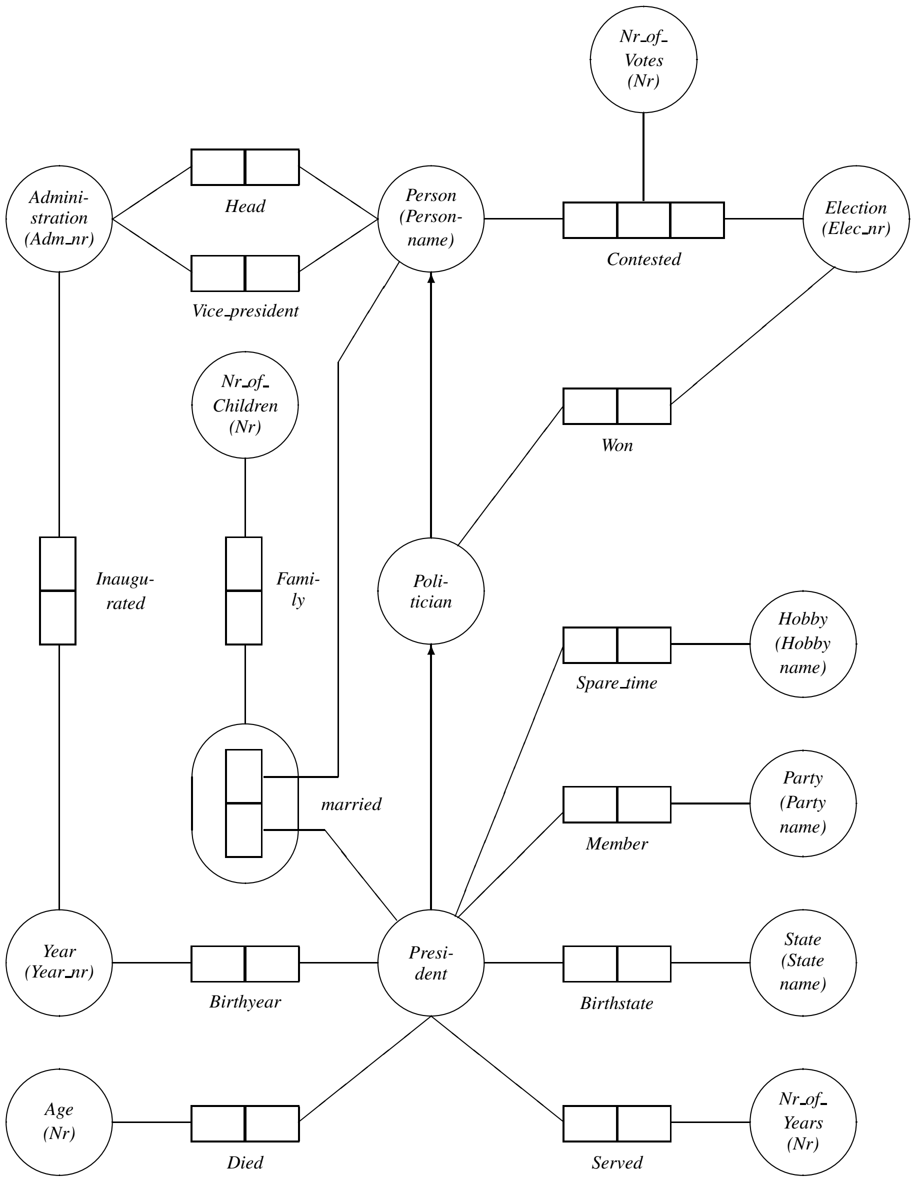


Figure 2: Part of the American president information structure

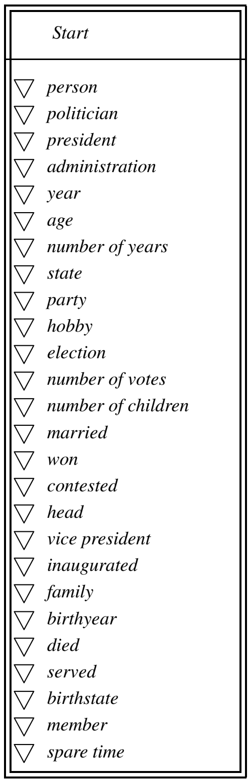


Figure 3: The starting node

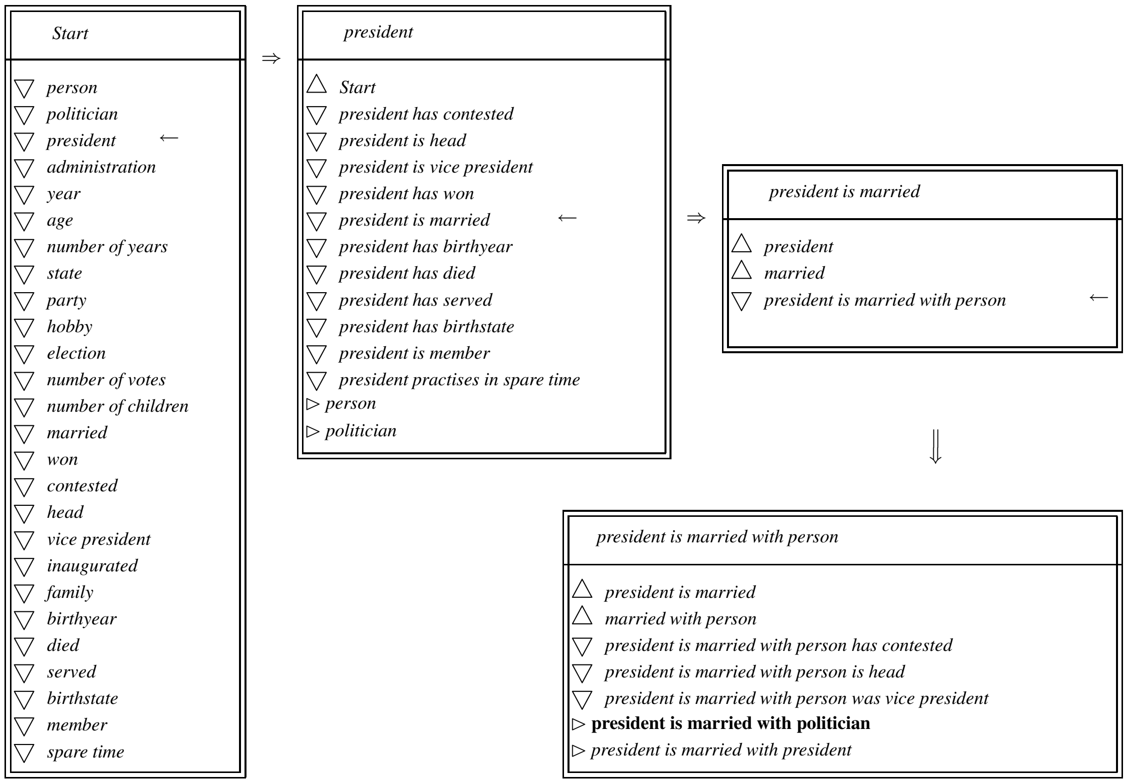


Figure 4: The quest for a president who is married with a politician

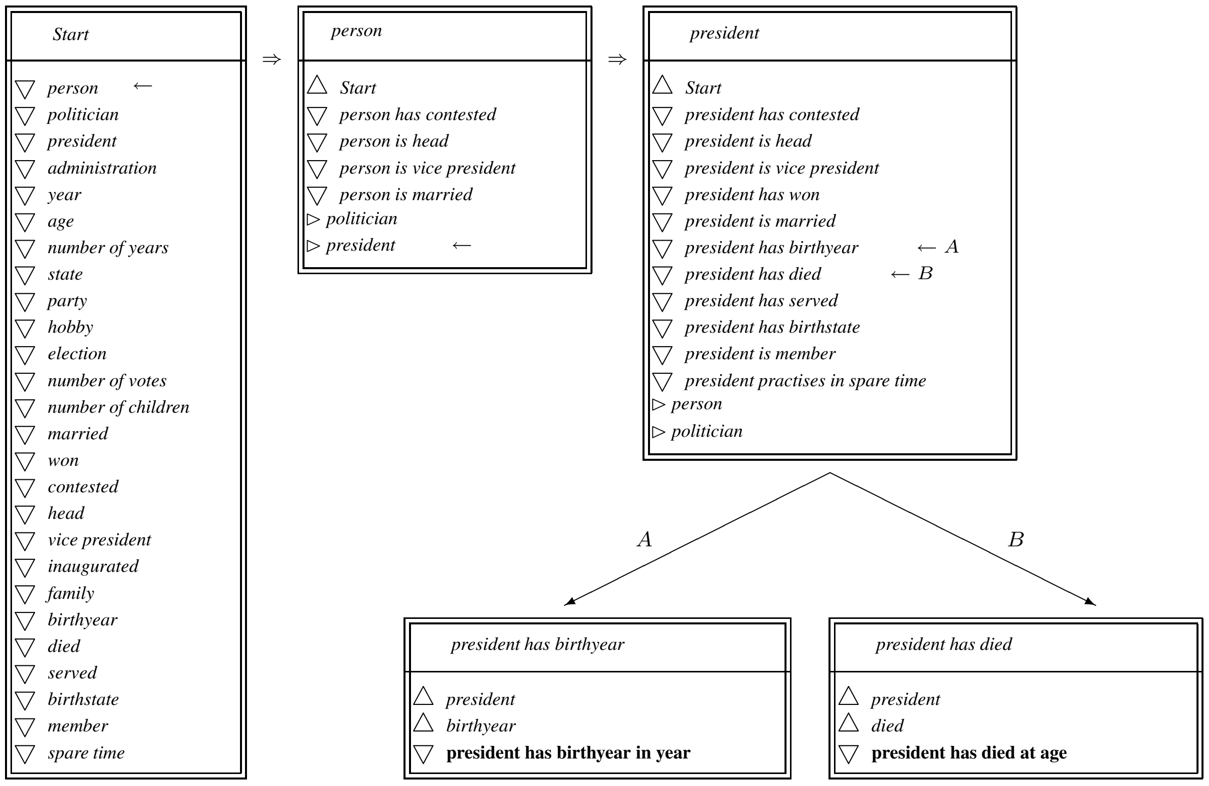
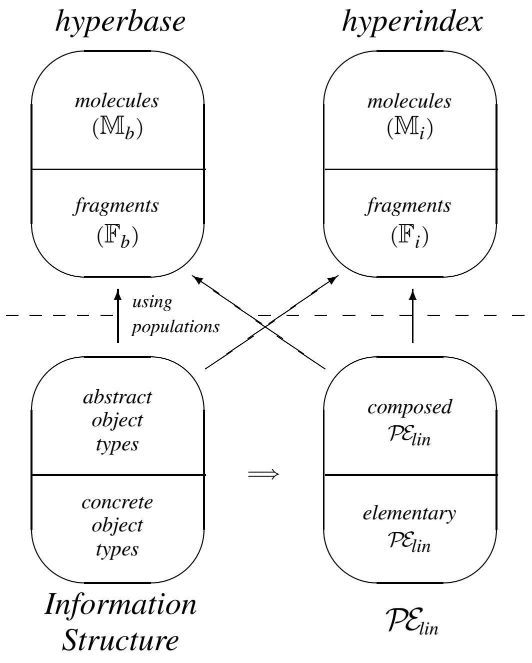


Figure 5: The quest for a persons birthyear and age of death



hyperbase
hyperindex
molecules ($\mathbb{M}_b$)
fragments ($\mathbb{F}_b$)
molecules ($\mathbb{M}_i$)
fragments ($\mathbb{F}_i$)
using populations
abstract object types
concrete object types
composed $\mathcal{PE}_{lin}$
elementary $\mathcal{PE}_{lin}$
⟹
Information Structure
$\mathcal{PE}_{lin}$

Figure 6: Transforming a PSM structure into a stratified hypermedia structure

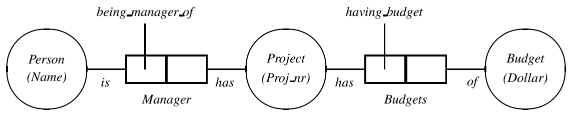


Figure 7: Projects, members and budgets

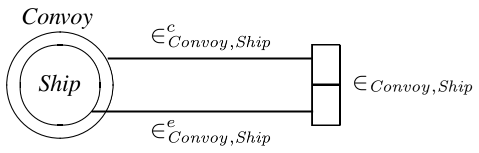


Figure 8: Implicit fact connecting Convoy and Ship

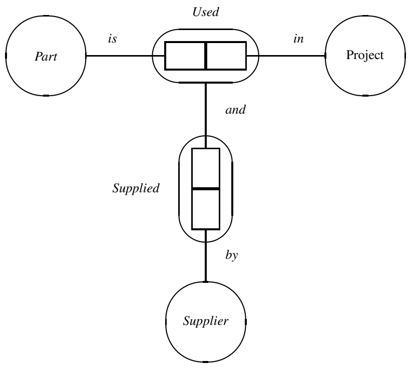


Figure 9: An example of objectification

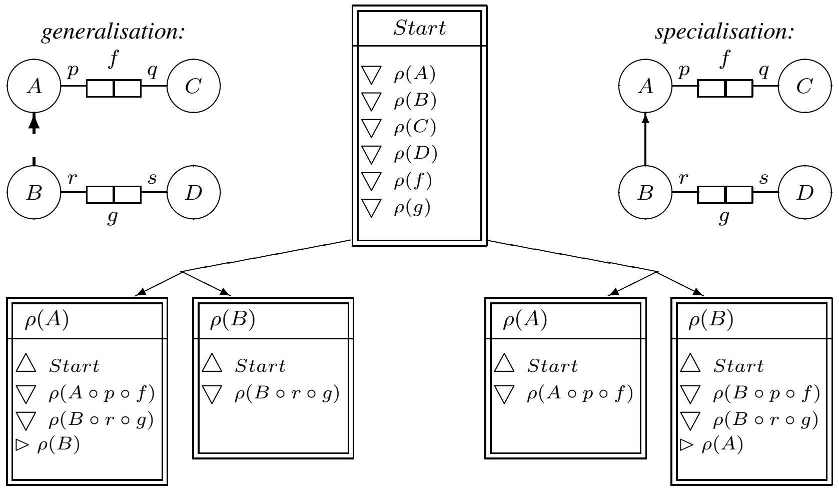


Figure 10: Hyperindex dealing with generalisation and specialisation

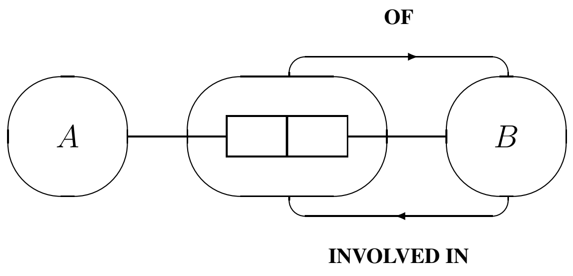


Figure 11: Keywords used in fact verbalisation

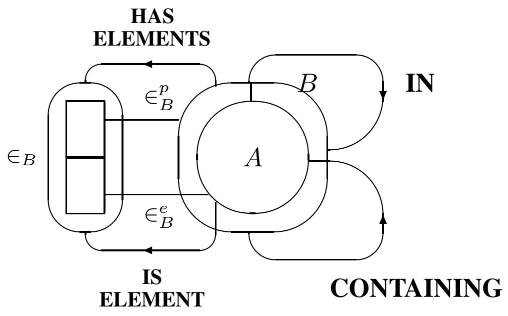


Figure 12: Keywords used in power type verbalisation

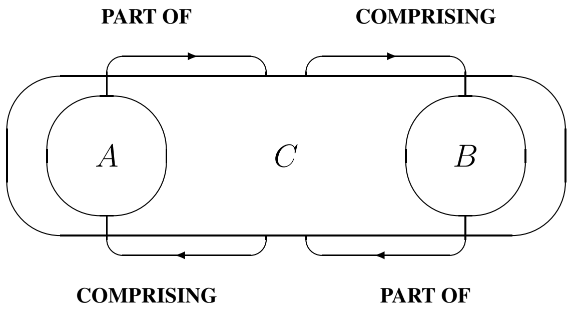


Figure 13: Keywords used in schema type verbalisation

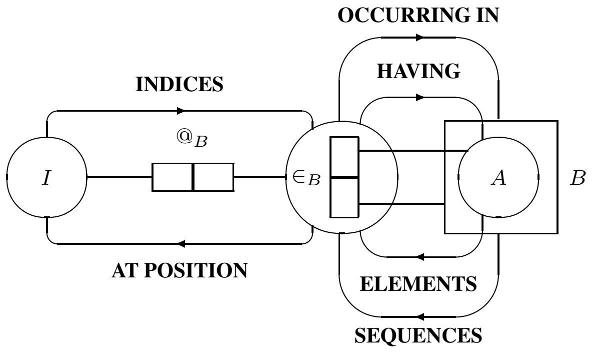
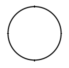
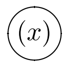
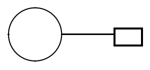
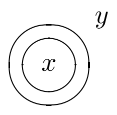
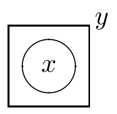
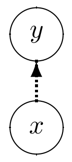
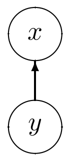
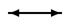
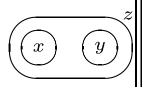


Figure 14: Keywords used in sequence type verbalisation

PSM Graphical Conventions

This appendix contains an overview of the PSM symbols for object types, generalisations and specialisations, and graphical constraints used in this article.

object type:	
label type x:	
role:	
predicator:	
y power type of x:	
y sequence type of x:	
y is generalisation of x:	

y is specialisation of x:	
single-fact uniqueness constraint:	
schema type:	
uniqueness constraint:	Ⓤ
total role or cover constraint:	⊙
exclusion constraint:	⊗
subset constraint:	
equality constraint:	⊜
enumeration constraint: