14 HYPERMILANO: HYPERMEDIA NAVIGATION IN A CITY INFORMATION POINT

Franca Garzotto, Luca Mainetti and Paolo Paolini
Politecnico di Milano
Department of Electronics and Information
Italy

This paper discusses design issues and information access paradigms in HyperMilano, a hypermedia information point about Milano. The static design of HyperMilano follows the guidelines of HDM - the Hypertext Design Model - which provides powerful constructs to define consistent and well organise hypermedia structures. HyperMilano supports different paradigms of information access. *Free navigation* allows exploration of the network without any constraint. *Guided navigation*, particularly suitable for novice users, to leads the reader across a set of information structures in a rigid manner, dynamically hiding the most complex connections. Various guided navigation patterns have been defined in HyperMilano: *linear* guided tours, *nested* guided tours, *branched* guided tours, *lattice* guided tours, and *automatic* guided tours. Finally, *query-based navigation* allows the definition of a starting point for exploration through a query, and allows the user to navigate the query results in free or guided mode.

HyperMilano provides multimedia information - texts, pictures, video, and data - about the most interesting aspects of Milano, including art, history, shopping, restaurants, and hotels. HyperMilano is bi-lingual, i.e., text and voice contents are available in Italian and English. Initially developed for the 2nd ACM European Conference on Hypertext and Hypermedia (held in Milano in December 1992), this application has been significantly improved, with the purpose of being located in protected public areas (e.g., the Gallery, the City Hall), hotels, or department stores, for free consultation by Italian and foreign tourists and casual visitors to the town.

The intended users of HyperMilano are (in general) non specialists in the application domain nor are particularly used to hypermedia. For this category of "naïf" users, the risk of disorientation, i.e., the feeling of getting lost in the hyperspace, is particularly high. Since disorientation obviously affects usability and acceptance of the application, we have paid a particular attention to reduce this negative effect, by increasing the quality of design, by distinguishing among different navigation patterns and making them intelligible to the users, and by integrating navigation with query based access.

A number of studies (Thuring, Haake, and Jannemann, 1991, or Garzotto, Paolini, Schwabe, and Berstein, 1991) have shown that bad design, i.e., inconsistent nodes-links organisation and lack of coherency in their visualisation, increases the disorientation effect. The more is a navigation environment regular and predictable, the more are hypermedia networks intelligible to the user and is the risk of losing the cognitive control of navigation reduced. However, good design might not avoid disorientation totally, since

even a well organised and consistent hypermedia might be too complex for novice users. The application should be able to guide exploration, by providing simplified navigation paths which lead users across (portions of) the network in rigid, but less confusing, manner. Finally, users should be allowed to directly identify a set of items of interest, by expressing their goals through queries, and to navigate from the query results to related items. This raises the issue of interplaying query based access with navigation.

HyperMilano: design and navigation

The representation structures of HyperMilano have been designed according to the primitives of HDM - Hypertext Design Model (Garzotto, Paolini, and Schwabe, 1992, and Garzotto, Mainetti, and Paolini, 1993). HDM does not address all the design phases of hypermedia application development. It focuses on "authoring-in-the-large", i.e., on the task of defining the representation structures of an application, abstracting from the development of the content ("authoring-in-the-small") and from the design of the lay-out features ("visual design").

In HDM, a hypermedia application is described by a set of structured information segments, called *entities*, and a set of connection structures of various categories, called *webs*, that glue together entities, or portions of entities, or other webs. HDM objects are typed, and the collection of their types is defined in the *application schema*. The regularity of HDM structures induce regularity in the navigation patterns, the semantics of which should be explicit to the user, in order to increase his understanding of the application.

Entities and entity types

An *entity* is a structured segment of information, that represents a physical or abstract object (e.g., a shop, a museum, a monument). Entities are organised collections (sequences or trees) of *components*.

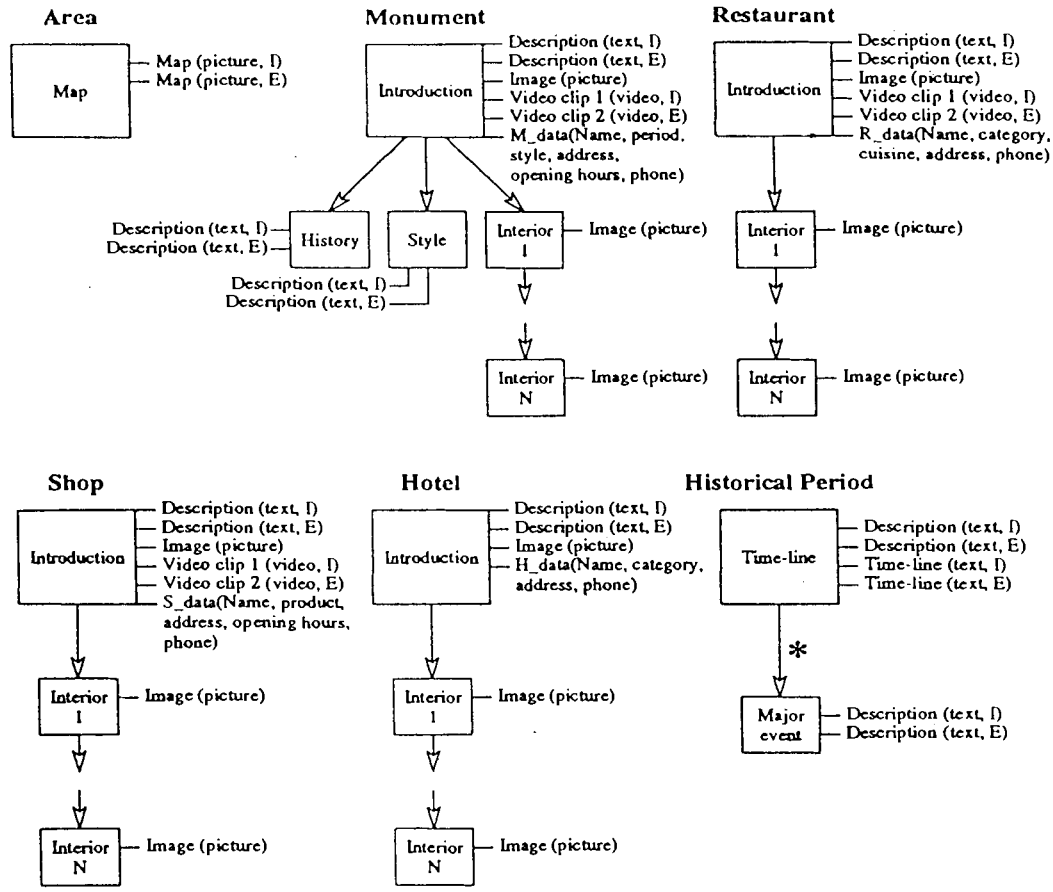
A monument, for example, has a root component that provides a short introduction to the monument, and a number of children components that describe, respectively, the monument history, the architecture style, and details about the interiors.

A component consists of a set of *units*, each one representing the same piece of information under a different *perspective*. A perspective defines the type of the data content, e.g., text, video, picture, formatted data, or its rhetorical style, e.g., "for beginner", "for specialist", or the language, e.g., Italian, English, French. The root component of a monument, for example, consists of six units: the picture of the monument, its brief description in Italian, its short description in English, some formatted data (address, opening hours, etc.), and two short video presentations of the monument, in Italian and in English. The component "monument history" have two text descriptions (units), in Italian and in English.

Entities are grouped into *entity types*. An entity type describes a class of domain objects (e.g., Monuments, Hotels, Restaurants). All entities of the same entity type have the same *structure*. A structure describes the organisation pattern of components and units within entities.

The entity types of HyperMilano are described in Fig.1. Terms in bold denote entity type names. Names inside rectangles denote the role of the corresponding components. Perspective definitions are specified outside the rectangles. The symbol "*" denotes multiple occurrences of a substructure within a given structure.

Fig. 1 Entity types of HyperMilano



Legenda

I = Italian

E = English

Navigation semantics of entities

The high level organisation of entities in components and, in turn, in units, induces a number of logical connections inside entities, which do not need to be defined explicitly by the designer, since they can be *derived from* the topology of entities and from the structure of components.

Perspective connections relate units of the same component (e.g., the monument picture and its description in English or Italian).

Structural connections denote structural relationships among components of the same entity, A connection *father-of*, for example, holds among a monument's root component and the component containing the monument history. *Child-of* is the inverse connection, with respect to *father-of*. *Next-brother* holds among a component and another child of the same father (e.g., between the monument history and its architecture style), etc.

Structural connections among components induce similar relationships among the corresponding units. For example, the connection *father-of* holding between the

introductory root component of a monument and its history induces a similar connection - let's call it *father-of /text-English* between the English text introduction and the monument history presentation in the same language.

In general, not all these logical connections correspond to links, i.e., have a navigational counterpart. Before establishing which of them will be perceived by the user as navigable links, the designer must decide how entities, components, and units, are perceived by the user as nodes, i.e., visual loci of navigation. The simplest case is that only units correspond to nodes. In a monument, we will have therefore six nodes corresponding to the root component.

In a more sophisticated case, nodes can correspond to components, or to a subset of units under given perspectives. In this case, we will have *composite* nodes, whose constituents correspond to the units. For example, the designer might decide that the following units of a monument root, picture, introductory Italian text, and introductory English text, all are mapped in the same node, while Italian video, English video, and formatted data, each one corresponds to an individual node.

Depending on his choice, the designer can select, for each entity type, which perspective and structural connections will be provided by an application. In general, two navigation patterns are available to explore information about a given entity:

- *Perspective navigation:* it is activated by selecting a link corresponding to a perspective connection. An example would be navigating from a video in English to the video in Italian concerning the same monument. The effects of perspective navigation are, from a cognitive point of view, very simple for a user. Activating a perspective connection, in fact, corresponds to visiting the same information element under a different perspective, without changing the current focus of attention.
- *Structural navigation:* it is activated by selecting a link corresponding to a structural connection. Structural navigation allows the reader to browse across chunks of information belonging to the same entity, i.e., concerning the same topic. Navigationally, it is a little more complex than perspective navigation, but is still cognitively simple, since it does not change the current topic of interest.

Webs and web types

Beside having structural and perspective connections, HDM objects can be interconnected in various ways and for different purposes. We will use the term "web" to denote a HDM structure that glues together two or more pre- existing HDM objects - units, components, entities, or webs.

HDM supports several categories of webs; each one has a different representational role and different navigational semantics. Webs of the various categories all share the same syntactic structure, made of a (possibly empty) *centre*, and a set of destinations coupled with selector names. The centre is a component that stores information about the whole web (e.g., a comment, an explanation, a set of attributes, etc.) under various perspectives. A destination can be an entity, a component, a unit, or another web. A selector name identifies the role of the destination in the connection structure.

Application webs and their navigation semantics

Application webs express domain relationships among components, entities, or units. A monument, for example, has been *built-during* a town history period. These relationships, in general, are not just binary. A monument, a shop, a hotel, for example, all are *located-in* a given area.

In application webs, a selector name identifies the role of a destination in the relationship. Application webs are grouped in types. An *application web type* describes a class of application webs that denote the same relationship.

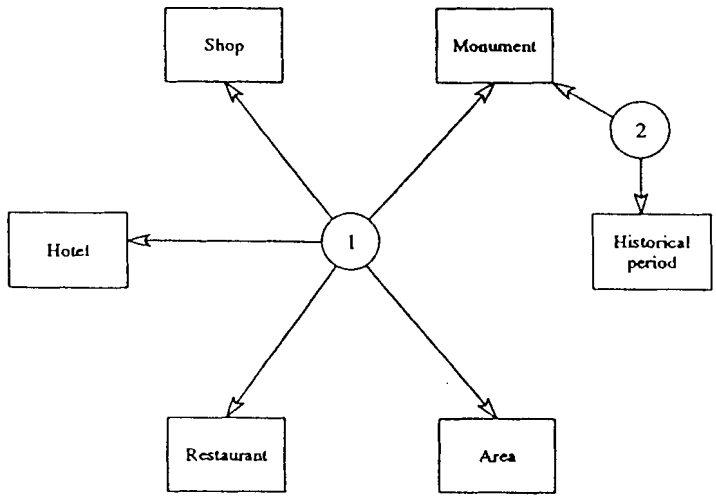
Some application webs are *base webs*, in the sense that their instances must be explicitly inserted by the application developer. Other application webs can be *derived* from other webs (transitive connections are typically used for derivation). In addition, from webs defined among components or entities it is possible to derive webs connecting units[1].

The application web types defined in HyperMilano are described in Fig.2. Webs of type (1) connect all monuments, hotels, restaurants, shops, located in a given area. Webs of type (2) connect monuments to the period(s) of Milano history in which they have been built.

There are several navigation patterns corresponding to application webs[2]. For each application web with centre, the user can navigate from a destination node to the web centre, and from here to any other destination. For each application web without centre, the user can move from any selector destination to each other destination.

From the user standpoint, this type of navigation can be the most disorienting. As the reader, in fact, traverses application webs, he will perceive that his information environment (i.e., the topic he is exploring), has abruptly changed (e.g., from a monument to the town area where it is located). In fact, application webs, in general, connect nodes generated from different entities.

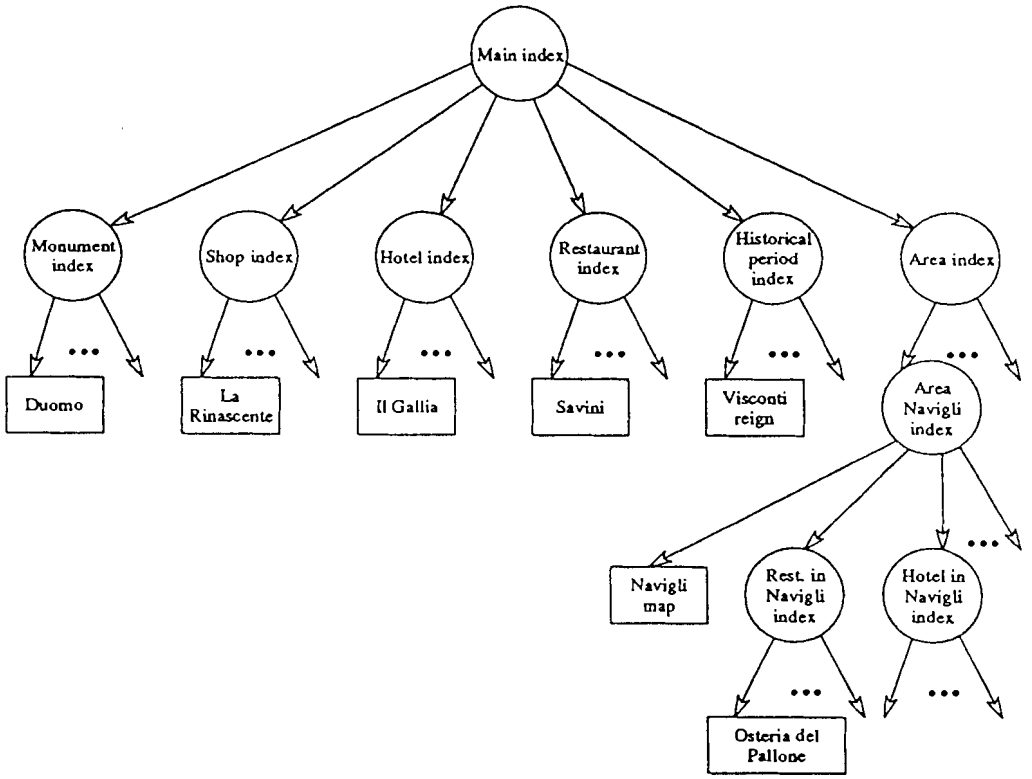
Fig. 2 Application Web types of HyperMilano



1 For lack of space, we will not elaborate further about derivation definitions and rules (Garzotto et al., 1993).

2 Not all HDM structures correspond to nodes. As a consequence, once the designer has defined how entities, components, and units are interpreted in terms of nodes, only webs connecting objects corresponding to nodes are actually used for naviation.

Fig. 3 Nested indexes in HyperMilano



Index webs and their navigation semantics

Index webs (indexes, for short) have the same structure as application webs - a (possibly empty) centre, and a number of destinations (called "items") with the corresponding selectors. An index connects a set of objects of interest to the users. Index items can be objects selected according to arbitrary criteria (say, for example, the index "Our favourite restaurants"), or can identify a subclass of objects that share some common properties or relationships (say, the top ten hotels, the Renaissance buildings, the monuments closed to Duomo, etc.). Thus indexes allow to introduce non intrusive classifications, without modifying the schema.

An index item can be either an information object (entity, component, or unit) or a web (typically, a sub-index). Indexes might be *typed* or non typed, i.e., *constant*. Constant indexes have the same structure as typed indexes, but they are totally arbitrary, with no type constraints on the destination items.

Differently from application webs, indexes are mainly used as entry points to the application, and have a different navigation semantics. The typical navigation on an index is from the centre to an item. Once the user has reached an item, he can return to the centre in order to select another item, or start any navigation pattern available from that item. Beside containing information concerning the whole index, or explaining its meaning, the main purpose of the index centre is to allow the selection of the item(s) of interest.

HyperMilano provides, among others, a main index which lists all entity types of the schema (Monument, Historical Period, Hotel, Restaurant, Shop, Area); each selector points to the index of all the entities of the corresponding type, the items of which are entities of the corresponding type, but for the Area index. The Area index points to a further index, which allows either to access the description of the selected zone, or to chose a specific topic in that area (e.g., its monuments) in order to access the index of the entities of the selected type. This situation is described in Fig.3.

Guided navigation

The patterns discussed so far can be defined as "free navigation", since there is no constraint in the way users explore the network. Free navigation in HyperMilano corresponds, metaphorically, to visit the town in a totally arbitrary way, moving from one place to another at will.

Arbitrary navigation can be disorienting for a novice user even in a very well organised application. Thus free navigation should be complemented with guided navigation, i.e., mechanisms that lead the user across simplified structures in a rigid manner. Metaphorically, guided navigation in HyperMilano would correspond to visit the town strictly according to the plan proposed by a guide book, or under the guidance of a human guide.

To support the design of guided navigation, HDM allows to define webs that have *guided tours*[3] semantics and provides several guided tour patterns, with different degrees of complexity and flexibility.

Linear guided tours

The simplest guided tour is the *linear guided tour*. As other categories of webs, linear guided tours are composed by a centre and a list of destinations with the corresponding selectors, but differently from application webs and indexes, they are intended to be traversed linearly.

The navigation starts from the centre, and from here, the user can navigate to the first destination, and from each element to the *next* element, to the *previous* element, the *first* element, and to the *last* element. Differently from indexes, in a linear guided tour the reader cannot select any item of interest at will: he has to make the selection according to a predefined order, starting from the first item.

Most importantly, while a user is located on a guided tour item, perspective, structural and application links outgoing from the current node *cannot be traversed*. They are inhibited (hidden to the user, or visible but disabled), unless the guided tour is explicitly *suspended*. Only from a suspended guided tour can the user activate these links, and reach any node (directly or indirectly) connected to his current position. At any point, during this exploration, the user can *resume* the guided tour from the item at which it was suspended, and restart his linear navigation from here[4]. If the user desires to stop guided navigation, he can either *stop* or *close* the tour. With a *stop* command, the reader is left in the current position, but the commands of the guided tour cannot be activated any more, while all links outgoing from the current node become active. With a *close* command, instead, the reader returns to the guided tour centre, still deactivating the guided tour.

---

3 Guided tours have been initially defined in Trigg (1986)

4 The implementation of guided tours requires a sophisticated definition of the run time model of navigation, which must dynamically keep track of the navigation status. The interested reader is referred to Garzotto, Mainetti, and Paolini (1993) for details.

Nested guided tours

In nested guided tours, some items themselves are guided tours. An example is a nested guided tour about Milanese historical buildings; odd items are building facade pictures, while even items are linear guided tours, each one having a building short description in its centre, and a list of items showing selected pictures about the building interiors. The navigation in the upper level guided tour lead the user from a building front picture to its description, and from here to the picture and description of the next building, and so on. The navigation in an inner guided tour allows to explore the interiors of the current building, starting from the text description.

Nested guided tours are slightly more disorienting than linear guided tours, unless the user is aware of which navigation patterns should be expected.

At any time, the user must be able to distinguish which guided tour has the navigation control. Once the upper guided tour is activated, the navigation commands *next*, *previous*, *first*, *last*, move the user from one item to the other at upper level. If the current item is in turn a guided tour, the user is located on its centre. In order to navigate inside the corresponding lower level guided tour, he must explicitly activate it.

From an item of an inner guided tour, the user cannot jump directly to the next, previous, first or last item of the upper guided tour. In order to continue the navigation in the upper guided tour, he must explicitly return it the control to the upper level, by a *close* command that deactivates the inner guided tour, and move the user to its centre, i.e., to the current stage of the upper guided tour.

Navigation in a nested guided tour of historical buildings would metaphorically correspond to a guided visit that plans to show the various monuments only from outside, but, upon tourist's request, guides him inside. Obviously, the visitor must exit and return outside, in order to continue the visit to the next building.

Branched guided tours

Branched guided tours have at least one branching point, from which the user must select where to go next among two or more (linear, nested, or branched) alternative items[5].

Structurally speaking, a branched guided tour is represented as a web, called a *main web*, in which the last item is in turn a web having guided tours of any kind (linear, nested, or branched in turn) as its items.

The navigation in a branched guided tour starts from the centre of the main web, and proceeds with *next* commands. Once the last item, i.e., the branching point, is reached, the user is located in the centre of the corresponding web. From here, the user can only go to the previous or first item, or stop or close the tour, but, in order to continue, he must *explicitly* select where to go next, among the various guided tours starting from there.

The navigation control is different from nested guided tours, where the control is hierarchical - by default, the navigation commands are interpreted in the upper level guided tour, unless the inner guided tour is explicitly activated. On a branching point of a branched guided tour, the simple next command is not allowed, since the user must specify which guided tour he wants to take from that point. Once a "direction" is selected, the effect is to concatenate the main web items with all items of the selected tour, temporarily discharging all other "options". Commands *next*, *previous*, *first*, *last*, *stop*, and *close* are interpreted in the result. A *close* command, for example, returns the user to the

5 By definition, there are no cycles in branched guided tours. Whatever the user's *next* choice, he will never return to a previously visited item.

centre of the main web. In addition, after the user selects a given path from a branching point, a *back-branch* command becomes available, which allows to return to the last visited branching point, and to select an alternative path if desired.

Lattice guided tours

Lattice guided tours are a variant of branched guided tours. Metaphorically, lattice guided tours correspond to a set of town visits which all start from the same "place", say, The Duomo, and end to a common place, say, The Castle, but at some point follow different itineraries[6]. Structurally speaking, a lattice guided tour is the same as a branched guided tour, but it has an additional property, namely, *all guided tours outgoing from a branching point share the last N items* (N>0).

The trouble with lattice guided tours is in the *previous* command. Once the user is located in the first of the shared items, he must explicitly specify which branch is wants to take back, i.e., which guided tour he wants to traverse in the inverse order.

Automatic guided tours

We use *automatic guided tours* to model, and navigate within, active media such as slide-shows, video clips, animations, sound-tracks, etc. Automatic guided tours are linear guided tours in which the *next* command is automatically determined by the system. In HyperMilano, for example, we have a video showing the most interesting places in town. The video sequence presents different topics (say, The Duomo, La Scala Theatre, Cavour Square, La Rinascente Department Store, etc.), each of which, in turn, is a video unit, and has connections to related information segments (say, the textual description, the city area, etc.)

We model this situation with an automatic guided tour, in which items are video units associated with the different topics. Starting the video corresponds to a *start* command on the guided tour. Running the video corresponds to performing automatically a sequence of *next's*, each one activated at the end of a video unit. If a video unit is connected to other information segments, the user can access these connected information segments by *suspending* the automatic guided tour, i.e., the video currently being played. From any point reached during this deviation outside the guided tour, he can resume and restart the video exactly in the position where it was "paused".

Queries and navigation

A query (in any environment) specifies conditions which must be satisfied from objects of a given type. The result of a query is an (ordered) list of information elements. As opposed to navigation, which retrieves one node at the time, query based access allows to retrieve several objects at the time, and can be helpful in situations where the user is able to express precisely his (initial) goal and wants to collect in one shot all information that can be useful for his task. For example, if he desires to select all restaurants serving Milanese cuisine, he will find easier to express this request through a query, rather than searching in the network all nodes of type Restaurant that have a Milanese menu.

However, pure query based access have a number of drawbacks. In our example, if a person wants to know where each Milanese restaurant is located, he has to reformulate the query, which can be quite tricky for a non expert user. A more intuitive solution would be to interplay query with navigation, allowing to access each node corresponding to the retrieved elements, and to explore its outgoing connections. In the above depicted situation, the user will access the node describing a Milanese restaurant, visit his picture

---

6 The expression "lattice guided tour" is inspired by mathematical lattices, which are acycled directed graphs with an upper and lower node, as defined for example in Tremblay and Manohar (1975).

and his description (perspective navigation), and then traverse the link to the area where the restaurant is located.

The notion of schema, the existence of various categories of links, and the possibility of having formatted data as unit contents, allow to introduce a powerful query language for HDM. Rather than presenting this query language, which is outside the scope of this paper, we shall briefly discuss the navigational interpretation of queries in HDM.

In HDM we interpret the result of a query as a *dynamic web*. The centre of this web contains the list of the identifiers of the objects selected by the query; the selectors of the web point to the selected objects. A dynamically created web can be used as an index (directly choosing the items to be explored) or as a linear guided tour (visiting all the selected items one after the other).

Conclusions and future work

In this paper we have described the design rationale of HyperMilano - a hypermedia information point about Milano - and a set of sophisticated patterns of information access defined in this application. HyperMilano has been specified in HDM - Hypertext Design Model - which allows to describe hypermedia applications at a high level of abstractions, and to enforce consistency and regularity of design by imposing a strong typing on the representation structures. In HyperMilano, we distinguish among free, i.e., unconstrained, navigation, guided tour navigation, and query based navigation. While free navigation allows to user to explore the application in a totally arbitrary way, guided tour navigation leads the user across (portions of) the network in a rigid manner, and, as such, is particularly suitable for novel users. The semantics of various patterns of free navigation (perspective, structural, application, and index navigation) and various categories of guided tours (linear, nested, branched, lattice, and automatic guided tours) has been informally presented. The navigational use of queries has been discussed, by interpreting the query result as an index or a guided tour.

The previous version of HyperMilano has been successfully tested during ECHT'92, the 2nd ACM International Conference on Hypertext and Hypermedia (held in Milano in December 1992) where the system was available to all conference participants. In its current version, HyperMilano provides most of the navigation features described along the paper. It is important to notice that the burden of implementing them, guided tours and query based access in particular, is totally left to the application developer. We have used Asymetrix Toolbook (under Windows), exploiting the Toolbook script language and data structures, and integrating them with a relational DBMSs. Current system today only support limited structuring facilities and naïf navigation and query patterns. Our belief is that modelling navigation is a design task, and should not be confined to the application writing level. We expect that, in future development systems, sophisticated navigation features will be implemented directly in the hypermedia engine. We are currently working on the specification of a hypermedia run time model (initially defined in Garzotto et al., 1993) aimed to provide the theoretical foundation for this work.

A development environment with system-level support for a relevant subset of the features proposed in this paper has been developed within the ESPRIT project HYTEA (HYTEA Working Team, 1993), started in 1989 and ended in March 1993, and is available at prototype level. Another ESPRIT project, HIFI, descrived in Cavallaro (1993), has started in mid 1992 and will continue the HYTEA work towards a complete integration of queries within a hypermedia application. In a related ESPRIT project, (MINERS Working Team, 1992) we are also further exploring the general implementation of guided tours.