A HyperNet Model for Large Hypertext Systems

Yi Deng, Shi-Kuo Chang
Department of Computer Science
University of Pittsburgh
Pittsburgh, PA 15260, USA

ABSTRACT

Two major problems with large hypertext systems are the user disorientation and lack of search ability. The problems, we believe, are mainly caused by the lack of semantics and modularity in hypertext data models. In this paper, a HyperNet model is proposed as a data model for large hypertext systems aimed at the above problems, which features strong modularity, attribute-based link semantics and capability of organizing information in various levels of abstraction. Based on the model, User View (UV), Local-HyperNet and Global GSP View (GGSVPV) are proposed as the user's access contexts, each of which emphasizes different aspects of the user access and compensate the weakness of the others. Algorithms are also provided to automatically generate and update user views. Two browsing schemes, namely automatic browsing and user-controlled multi-level browsing, are presented to provide flexible navigation in the database and to remit the disorientation problem. At last, a concurrent search algorithm is proposed to provide efficient search/query in (potentially distributed) hypertext systems.

1. Introduction

Two major problems with current hypertext systems are the user disorientation problem and the lack of search/query ability (3,9). The so called disorientation problem is the one that when a user faces a large, unfamiliar, heterogeneously structured network, he tends to get lost while wandering around in the network looking for some target information. Often these users can describe exactly what information they are looking for, but simply cannot find it in the network. As indicated by Halasz [9], even in a 500 node single user network, navigational access can be difficult as the network changes and its structure becomes heterogeneous. A problem closely related to the disorientation is cognitive overhead [3], i.e. when the network becomes large, it tends to cause tremendous user effort to examine a large number of links in order to determine which link to follow next. The problem of the lack of searching ability is, on one hand, considered as one of the major reasons causing disorientation [1,3,4,9], on the other hand, significantly affect the efficiency of user access to the database.

The major causes of the above problems, we believe, are the following: (1) the network is too large, where large amount of nodes and links are out of the user's concern. (2) Links are the only medium to describe the relations among the concepts of the nodes In hypertext systems. To support efficient search/query, the data model for hypertext systems should provide enough information about the relations of the nodes. Furthermore, failure to provide enough information about the relations among the links makes it more difficult for the user to determine the direction of browsing. (3) The lack of modularity in hypertext data models makes a database composed of a huge net and not be able to reflect the logical hierarchy and abstraction of the underlying data.

The above problems are the major focus of this paper. We present HyperNet model as a representation of hypertext, which can be either superimposed on the existing systems or used independently. The model features strong modularity, attribute-based link semantics and capability of organizing information in various levels of abstraction. The model not only provides a guideline to construct highly modular, hierarchical hypertext database, but also provides effective support of flexible navigational access of hypertext databases and concurrent and attribute-based search in the database.

The rest of the paper is organized as follow: The HyperNet model is presented in section 2. In section 3, the concept of User View (UV) is discussed, where a UV is a substructure of the global hypertext database (which is called global view in this paper), which is associated with a user of the database and only contains the user interested information. The algorithms for automatically generating and updating a UV are also presented in the section. In section 4, it is discussed how convenient navigational access can be performed based on the HyperNets. In particular, three browsing contexts, namely UV, Local-HyperNet and Global GSP View (GGSVPV) are proposed, their features and relationship are discussed, upon which two browsing modes, namely, automatic browsing and user-controlled multi-level browsing are presented as vehicles to achieve flexible browsing. Section 5 is dedicated to discuss the search/query mechanism on HyperNets. A current search algorithm based on the user provided attribute list is presented. At last, the discussion and conclusion of the paper are in section 6.

2. HyperNet Model

HyperNet model is based on the concept of the G-Net model [5], which, in turn, is a Petri-Net based representation model [8,10,11,12]. The basic building blocks of a HyperNet are places, transitions, and links which connect places and transitions, where places are information storage units and transitions specify the relationship among places. As an introductory example, figure 1 shows a fragment of the HyperNet organization of this paper. It is assumed that the paper is described by a HyperNet $H_{UV}$. $P = \{p_i, ISP_i(3), ISP_i(5), ISP_i(8), ISP_i(13) \mid i = 1..6\}$ are set of places, where $p_i, i = 1..6$. 

DATABASE SYSTEMS FOR ADVANCED APPLICATIONS '91
Ed. A. Makinouchi
©World Scientific Publishing Co.
1.6. represents the i'th section of the paper and the rest of the places are called Instantiated Switch Places (ISP), which will be discussed shortly. \( T = \{ t_{ij} \mid j = 1 \ldots 16 \} \) is the set of transitions representing the relations between the places, where:

- \( t_1 = (\text{ORG}, \{ \text{HyperNet}, \text{link-semantics}, \text{hypertext-data-model}, \text{next} \}, \{ \}) \),
- \( t_2 = (\text{REF}, \{ \text{next} \}, \{ \}) \),
- \( t_3 = (\text{ORG}, \{ \text{UV}, \text{disorientation} \}, \{ \}) \),
- \( t_4 = (\text{REF}, \{ \text{next}, \text{navigation}, \text{UV} \}, \{ \}) \),
- \( t_5 = (\text{ORG}, \{ \text{UV}, \text{navigation}, \text{disorientation} \}, \{ \}) \),
- \( t_6 = (\text{REF}, \{ \text{next}, \text{disorientation} \}, \{ \}) \),
- \( t_7 = (\text{REF}, \{ \text{next}, \text{search} \}, \{ \}) \),
- \( t_8 = (\text{ORG}, \{ \text{search}, \text{disorientation} \}, \{ \}) \),
- \( t_9 = (\text{ORG}, \{ \text{summary} \}, \{ \}) \),
- \( t_{10} = (\text{REF}, \{ \text{UV}, \text{navigation} \}, \{ \}) \),
- \( t_{11} = (\text{REF}, \{ \text{navigation}, \text{disorientation} \}, \{ \}) \),
- \( t_{12} = (\text{REF}, \{ \text{HyperNet}, \text{hypertext-data-model}, \text{next} \}, \{ \}) \),
- \( t_{13} = (\text{REF}, \{ \text{disorientation}, \text{link-semantics}, \text{navigation}, \text{search} \}, \{ \}) \),
- \( t_{14} = (\text{REF}, \{ \text{disorientation}, \text{search} \}, \{ \}) \),
- \( t_{15} = (\text{REF}, \{ \text{concurrent-browsing-paths}, \text{PetriNet-Based-hypertext} \}, \{ \}) \),
- \( t_{16} = (\text{REF}, \{ \text{PetriNet-based-representation}, \text{link-semantics}, \text{complex-nets} \}, \{ \}) \).

Figure 1. An example of HyperNets.

A place may be associated with one of following two types of information: applicational data or structural information, where the structural information specifies relations and connections between two different parts of a HyperNet. A place, which stores applicational data, is called Normal place (or simply place), and a place, which stores structural information, is called a Instantiated Switch Place (ISP). In figure 1, the places \( p_i, i = 1 \ldots 6 \) are normal places, and the places \( ISP(j), j = 3,5,8,13 \) are ISPs. From dynamic point of view, each place can be in one of the three states: current, active and passive. A place is current if it is currently being accessed; a place is active if it is currently under display, otherwise it is said passive.

The relationship among places are described by semantic relation augmented by attributes. According to Conklin [3], there are basic two types of relationships among the information units in hypertext. i.e. organizational relation and referential relation. The former describes hierarchical (parent-child) relationship among the units, and the latter describes non-hierarchical relationship. Similarly, there are two basic semantic relations in HyperNet model, ORG and REF, which stand for organizational and referential relationship respectively. Attribute list associated with each semantic relation plays an important role in specifying inter-place relationship, which further describes in which sense two or more information units have a given relationship in given application domains. Therefore, two basic components of a transition are a semantic relation and an augmented attribute list. In the transition lists given in figure 1, the first field of each transition is its associated relation, and the second field is the augmented attribute list.

A transition may have a set of input places and a set of output places \([5,8,10,12]\). A transition \( t_i \) with relation ORG indicates that the input places of \( t_i \) are the parents of the output places in the hierarchy. A transition \( t_i \) with relation REF specifies the cross-reference relation between the input places and the output places, with the input places function as references and the output places as referents. For example, in figure 1, \( p_1 \) is a parent of \( p_2 \) in terms of attribute list \( \{ \text{HyperNet}, \text{link-semantics}, \text{hypertext-data-model}, \text{next} \} \), because that \( t_1 \) is an ORG transition with \( p_1 \) as its input place and \( p_2 \) as its output one. Similarly, \( p_2 \) and \( p_3 \) are related by attribute list \( \{ \text{UV}, \text{next} \} \) although they don't have parent-child relationship. The attribute, next, is a special attribute defined in the net, which indicates the next place(s) to be accessed if it is not otherwise specified.

The node marked as GSP(HN) in figure 1 is a special place called Generic Switch Place (GSP). Each HyperNet has one GSP, which uniquely identify the net. A GSP is composed of three fields, an unique net ID (the ID for the net in figure 1 is HN), a brief description of the information stored in the net (e.g. the abstract of this paper may be stored in GSP(HN)), and the last field is called a marking of the net, which indicates which units to be first displayed when the net is accessed, if not otherwise specified. For instance, the dashed line in figure 1 indicates \( p_1 \) will be the default first units to be accessed.

There are two classes of HyperNets, elementary HyperNets and complex HyperNets. An elementary net is one with only one GSP, and a complex net is one with more than one GSPs. In other words, a complex HyperNet is composed of a set of inter-connected elementary nets. For example, the net shown in figure 1 is an elementary net. The connections between elementary nets are made possible by those ISPs. An ISP is always associated with a GSP (it is an instantiated copy of the GSP), the marking in the ISP can be different from the one in the GSP, and the ISP always belongs to another net rather than the one identified by the GSP. Since an ISP is a place, it can be put in any net and connected with other places in the net using transitions. A net HN, which contains an ISP of another net HN, is said connected to the net HN. Since an ISP is a place, it can be related to other places by relations ORG and REF. Therefore, the relationship between the two nets connected by the ISP can also be described using relation ORG/REF augmented by an attribute list.
For instance, given two nets $HN_i$ and $HN_j$, where $HN_i$ dominates $HN_j$ according to attribute [a], $HN_j$ dominates $HN_i$ according to attributes [b,c], and furthermore, $HN_j$ can be cross referenced from $HN_i$ according to attributes [d,e]. The two nets can be connected using three ISPs, $ISP_1$, $ISP_2$ and $ISP_3$, as shown in figure 2. The $ISP_1$ of $HN_j$ is connected to transition $i_1 = (ORG, [a], (1))$; the $ISP_2$ of $HN_i$ is connected to transition $i_2 = (ORG, [d,e], (1))$ in $HN_i$; and the $ISP_3$ of $HN_i$ is connected to $i_3 = (ORG, [b,c], (1))$ in $HN_i$. Two ISPs of $HN_i$ created in $HN_i$ implies that $HN_j$ may be accessed in different ways from $HN_i$.

![Figure 2. Example of connecting two HyperNets](image)

Suppose each of the references of this paper is represented by a HyperNet, figure 1 shows the connections between this paper and the referenced papers. $ISP_3(3)$, $ISP_3(5)$, $ISP_3(8)$, and $ISP_3(13)$ are the instantiated ISP of the HyperNets corresponding to the paper [3], [5], [9], and [14], respectively. The transitions $t_{11}$, $t_{16}$, $t_{19}$ and $t_{15}$ represent the relationship between this paper and the papers [3], [5], [9] and [14], correspondingly. Furthermore, if each of the sections of this paper is represented by a lower level HyperNet, the places, $p_i$, $i = 1..6$ in the net will be replaced by a set of ISPs.

The formal definition of HyperNet model is given below. Refer [6] for detailed discussions of each parts in the model.

A HyperNet $HN = (GSP, P, T, I, O, D, A)$, where,

$GSP = (NID, M, B)$ is a special place called generic switch place of the net, where NID is the unique ID of the HyperNet, and $M = (tk_i, i = 1..n)$ is the marking vector, where $tk_i$ is a token list associated with place $p_i$, and $B$ is the data associated with $GSP$, which is the brief description of the information stored in the net.

$P = \{p_i | i = 1..n\}$ is a set of places. Each place $p_i$ can be in one of the two types: normal place or instantiated switch place (ISP). Each normal place $p_i = (tk_i, AL_i, CON_i)$, where $tk_i \in N$, $N$ is the set of natural numbers, is a user that accessed when the place is last accessed; $AL_i$ is an attribute list which is used to describe the properties of $CON_i$; and $CON_i$ is the data or information associated with the place. Each ISP is an instantiated copy of another HyperNet's GSP, which serves as a connection between the net and another HyperNet.

$T = \{t_j | j = 1..m\}$ is a set of transitions, where each $t_j = (REL, AL, TL)$, where $REL \in D$ is the relation associated with $t_j$, $AL \subseteq A$ is a list of attributes associated with $t_j$, $TL = \{c(t_i, c(t_j)) | c_i \in NAME(P), c_i \in N, i = 1..L\}$ is a token list of $t_j$, where $NAME(P)$ is the set of place names. The meaning of $TL$ is to be defined.

$T : 2^T \rightarrow T$ is the output function.

$O : T \rightarrow 2^T$ is the output function.

$D = \{ORG, REF, GLOBAL\}$ is the set of semantic relations, where $ORG$ stands for organizational relation, $REF$ stands for referential relation, and $GLOBAL$ stands for global relation, as will be explained shortly.

$A$ is a set of attributes, which is domain specific and remains undefined in the model.

As a hypertext data model, HyperNet model has the following features. (1) Modularity. Elementary HyperNets provide the basic building blocks of large hypertext systems. Based on these building blocks, arbitrary large network can be constructed, and still, each logically independent part of the network can be managed to reasonable size and logical clarity. The concept of ISP provides a flexible way to inter-connect and represent the semantic relationship within subnetworks of a large complex HyperNet uniformly as representing the semantic relations of primitive information units. (2) Abstraction mechanism. HyperNet model is capable of representing information based on different levels of abstraction. If we view each elementary net in a large HyperNet as a single node, and view the underlying hypertext database as a space containing information for certain application domain, the ORG connections among the nodes will show a hierarchy where nodes in higher levels cover larger grained subspaces or represent more general information chunks, and the nodes in lower levels cover small grained subspaces or represent more detailed information chunks. This feature not only ease the task of organizing information in hypertext database, but also to tasks of updating and re-organizing the hypertext network. For instance, if, in figure 1, some section of this paper becomes too long to store in a single place, another lower level net may be built to represent the information in that section, and the corresponding place in figure 1 can be replaced by an ISP of the new net. (3) Attribute-based semantic relations. The relationship among the places or between different elementary nets are not only described in terms of broadly-sensed relations, but also further clarified or justified by the augmented attributes, which allows not only to describe the relationships more accurately, but also to the relationships from different perspectives.

3. User Views

One of the major causes of user disorientation is that the space or amount of information the user face is too large to handle. A natural way to solve the problem is to reduce the amount of information the user has to face in accessing the database, which is the focus of this section.

The strategy used in this paper is to build User Views (UV) [2]. A UV belongs to one user and is a portion of the database network, which only contains information which is presumably interested by the user. The goals are to provide the user a small and handy working space and hope that most of the user's navigational access can be done in his UV, and at meantime, allow the user to traverse and access whatever information he wants to in the database.

The key issue for building a UV, of course, is what information should be kept in the UV, which is determined based on the following rules: (1) The places accessed recently by the user, as well as the paths of the accesses, should be included in the UV, which provides the user the history of his accesses, and allows the user to backtrack. (2) The active places (including the current place) should always be included in the UV, which is part of the context of the user's access. (3) The adjacent places of the current place should be included in the UV, under the assumption that the next place to be accessed by
the user is most likely one of the adjacent places of $p_2$. (4) The adjacent places of the previous current place, except the ones which have been accessed, should be removed from the UV to reduce its size.

The basic requirements of a UV are the abilities of dynamical growth and shrinkage. In order to present the currently user interested information, it is essential to dynamically add information to the UV. In order to keep the size of the UV small enough to avoid user cognitive overload, the information which the user has not accessed for certain threshold period of time should be removed from the UV. The structure of the UV should also be compatible with the structure of the global view of the underlying hypertext database (A UV is compatible with the GV iff the UV is a subgraph of the global view (GV)). In doing so, the system will exhibit the same behavior when a user performs navigation or search in the user view, as in the global view.

Two algorithms, a UV insertion algorithm and a UV pruning algorithm, have been designed to dynamically add or remove information to or from a UV, and the invocation of both algorithms is transparent to the user. The insertion algorithm is invoked whenever the user makes a successful move and inserts the access path into the UV based on the insertion rules mentioned above. The UV pruning algorithm is invoked periodically by the system to remove the units, which has not been accessed for certain threshold period of time, from the UV. Assume of segment of the global view is as figure 1, an example of insertion is shown in figure 2, where figure 2(a) is the UV structure before the insertion, figure 2(b) shows a successful access path, and figure 2(c) shows the UV structure after the insertion. For the details for the algorithms, refer to [6].

A UV can be considered as a kind of partial global view [17], which has the following features. (1) This partial global view is personalized, i.e. it provides the accessing history personal to the user and the hints for the future access according to the user’s previous access. (2) The UV is dynamic. The UV is built up incrementally, and its content is dynamically changed as the user’s access to reflect the user’s current interests. (3) The structure of this partial global view is compatible to the one of the global view.

4. Navigation in HyperNets

In order to navigate through a large hypertext system efficiently without getting lost, it is important for a user to know the answers of the following questions: What are in the system and where do I stand in the system? How do I get to the current place, and where to go next?

One solution to the first question is to provide the user the global map and/or local map of the system [17]. The global map approach has been suggested not appropriate because it is too large, and therefore, too difficult for the user to examine [17]. The local map, with current active nodes marked, approach has been proven helpful [17]. It, however, is not sufficient because that the local map only gives the user an isolated local view of the system. In order to determine his current status of access and determine his next move, it is sometimes important for the user to know the information of beyond the local map.

In this section, we describe two browsing modes, namely automatic browsing and user controlled multi-level browsing, based on the concept of complex HyperNets. These browsing methods provide flexible ways of database access, and each of which may be used as a complement of the other.

The first browsing mode is automatic browsing. It is so called because that the system will determine for the user about which place(s) to be accessed next based on the user provided information and the history of the user access. An invocation of automatic browsing is similar to a guided tour to the hypertext database. The automatic browsing may proceed in one of the two ways: (1) This guided tour may be conducted based on some specific topics (attributes). In this case, the user has to provide which topics he/she wants to look into. The execution of the automatic browsing algorithm will then only visit the places related to the given attributes. (2) If there is no specific attribute provided, the automatic browsing will traverse the database in depth-first fashion. An automatic browsing may stop whenever the user issues an stop command or the algorithm can not proceed.

The default context for automatic browsing is UV. If there is a current place $p_s$, (i.e. the place whose state is current), the browsing will start from $p_s$. (Note: the current place for a user is always in his UV.) based on the assumption that a person's work has its continuity and locality. If there is no current place for the user, the browsing will start from the root (or a default starting point if the database is not organized in a hierarchy) of the global hypertext database, and the root then becomes the current place.

Once having a current place $p_s$, the next place to be visited is an adjacent place ¹ of $p_s$. If an attribute list AL is provided, the automatic browsing algorithm will then examine the adjacent output.

---

1 A place is an adjacent place of a given place if they are connected by one transition.
transitions to see if the attribute list of any of the transitions has common attributes with AL and has not been visited before. If so, the output place of the transition with maximal number of attributes matched with the ones in AL will be selected as next current place. (Note: the next place and its adjacent places will be automatically inserted to the UV. Refer to section 3.) If there is no such output transition, the automatic browsing algorithm will automatically backtrack up to the point where there is such a transition, otherwise, the algorithm will exit. The third field, token list, in a transition is used to record the access paths, and used as a reference for backtracking. If no attribute list provided, the algorithm will simply pick the next unvisited adjacent place of \( p_C \) as the next current place. The traversal will stop when the current place is the starting place and all of its adjacent places have been visited.

Notice that the boundary of elementary nets will not cause any problem in the browsing. Since the elementary nets in a complex one are connected through ISPs and the ISPs can be accessed just like normal places. Whenever an ISP is accessed, the net associated with ISP is then inserted into the UV, and a new current place will be chosen in the net. (Refer to [6] for the details of insertion.) The browsing will then naturally extends to the newly inserted net. When the browsing in the net is done, the browsing algorithm will set the next current place to the ISP in another net, from where the current net is entered, if it exists.

As an example, suppose that the automatic browsing algorithm is invoked with attribute list \( AL = \{UV, navigation\} \); there is no current place for the user who invokes the browsing; there is no UV for the user, and the default starting point is the net of figure 1, then a UV for the user is created; \( p_1 \) will become the first current place according to the UV insertion algorithm of [6]; and the adjacent places of \( p_1 \) are inserted into the UV as shown in figure 4(a). In the next step, \( p_4 \) becomes the current place because \( AL \) is a set of the attribute list of \( t_1 \), which connects \( p_1 \) and \( p_4 \). The corresponding UV is shown in figure 4(b). Examining the adjacent transitions of \( p_4 \), only \( t_5 \) has common attributes with \( AL \). Therefore, the browsing is extended to net \( HN_{13} \) associated with \( ISP_{13} \). When the browsing in net \( HN_{13} \) is done, the place \( ISP_{13} \) becomes the current place again through backtracking. At this point, backtracking in net \( HN_0 \) occurs, through \( p_1 \) back to \( p_1 \), where \( p_1 \) is selected as the next current place, as shown in figure 4(c). In each of the above steps, the next place to visit is either selected from the adjacent places of the current place, which are located in UV, or through backtracking. Also notice that at each step, the adjacent places of the previous current place, except for the ones which have been visited before, are deleted from the UV. For the details of the automatic browsing algorithm, refer to [6].

The user controlled browsing is perhaps the most popular browsing scheme [1,3,9,15,17]. Our second browsing mode, user-controlled multi-level browsing, belongs to this category. In this browsing mode, the most important thing is the browsing context, i.e. the information provided to the user about how data are organized in the database, based which the user's browsing decision is made. Appropriate information provided can facilitate the user's decision, on the other hand, inappropriate information is the direct cause of the problems of cognitive overhead and disorientation.

The browsing environment provided by our scheme is called Adaptive Browsing Environment (ABE). The ABE provides the underlying network information in three different forms, namely, User View (UV), local HyperNet and Global GSP View (GGSPV), and a user is entitled to choose any of them based on a specific browsing situation or need. A UV emphasizes the history, locality and continuity of the user's access. In certain occasions, however, the UV is not adequate to be used alone. For instance, when a user accesses the system for the first time, his UV is empty; when the user wants to start a new subject, the information in his UV is likely to become useless; or when the user needs to view the global structure.
of the system. A local HyperNet is an elementary net which a user is accessing. Since each elementary net is composed of a set of logically closely related information units, a local net can provide detailed local structural information. Since the local HyperNet is connected to the others through its ISPs, the local net also provides summary information about its adjacent nets. Figure 1 is such an example.

Getting global structural information of a large hypertext database is a difficult task, because the size of the network is too large to display on screen and too complicated for the users to scan, and yet knowing the global structure could be important in many situations [17]. The third form of browsing context, GGSPV, is aimed to solve the problem. In GGSPV, each elementary HyperNet, HN, is shrunken into a single place, its GSP(HN). The inter-HyperNet connections are now represented by inter-GSP connections. More specifically, given two nets HN1 and HN2, where each ISP1, i = 1..k, of HN1 can be connected through its ISPs to HN2 by a transition r, with relation REF and attribute list AL1, i.e. there are k REF connections from HN1 to HN2 with attribute lists AL1..ALk, i.e. ISP1..ISPk, i = 1..k. These connections are now combined into one connection from GSP(HN1) to GSP(HN2) by a transition r, with relation REF and attribute list AL12 = AL1 \cup AL2 \cup ... \cup ALk. This technique is called connection clustering. The ORG connections between HN1 and HN2 and the REF connections from HN2 to HN1 can be clustered in the same way.

The concurrent search algorithm is designed for potentially distributed hypertext systems, which conducts search concurrently in multiple machines, and further invokes multiple process to conduct parallel search within each machine. The following assumptions are made for the algorithms: (1) The assumed database is organized by a complex HyperNet, which is distributed on one or more machines, where any elementary HyperNet is always stored in a single machine; (2) There is a root HyperNet for the entire (distributed) database. Note that this assumption does not require the database to be organized as a tree structure; (3) Reliable communication channel, i.e. a message sent is correctly delivered to its destination within certain given time period 6; (4) Within each machine, multiple programming is allowed.

The input of the algorithm is an attribute list, which specifies the topics to be searched. The output is a set of places which contains the related information with the given attribute list, which is stored in a structure called Result Set (RS) of the search. The search is based on the simple attribute matching between the given attribute list and the attribute lists of the transitions in the complex net. It should be noticed, however, that the algorithm can be modified to accept a logical expression over attributes. In addition, the search algorithm does not differentiate the direction of a connection, i.e. a search wave can go through a transition from its input end to the output end, and vice versa, as long as the transition has common attribute(s) with the given attribute list.

The concurrent search algorithm can be sketched as follows. The search is started by creating a search process PROC, which is responsible for searching in the root net HN1. The default marking of HN1, determines from which place(s) the search will start within HN1, and marks those places as active (by depositing a token in each of the places). PROC, then randomly picks up an enabled transition (either forward or backward enabled), and fires the transition. Whenever a token is deposited to a place, the place is added to the result set RS, of PROC,. If a place receiving a token is an ISP(HN1), PROC, will check if the HN1 belongs the local machine. If yes, PROC, will ask the local hypertext system to create a new search process PROC1, which is responsible for search in net HN1. If HN1 belongs to a remote machine, PROC, will send a search request to the remote machine where HN1 resides, where a new search process PROC2, is in turn created. Once PROC, is created or the request message is sent to the remote machine, PROC, will continue its search in net HN1. PROC, is called the parent search process of PROC,. PROC, operates in a similar fashion as PROC,.
In this way, a set of truly concurrent search processes will be created during the search. Once a search process \( \text{PROC}_i \) is done, i.e. its search in net \( \text{HN}_i \) is done, and it has received the replies to every request message it sent out, \( \text{PROC}_i \) takes the union of its result set \( R_i \) with the replies received as the result of its search, reports the result to its parent process, and terminates itself. Once \( \text{PROC}_i \) is done, the whole search process is then accomplished. For the details of the search algorithm refer to [6].

6. Discussion and Conclusion

In this paper, a HyperNet model is presented as a representation of large hypertext systems. The design focuses are to solve the user disorientation problem and to provide mechanisms for convenient and efficient access to the underlying hypertext database. The model has the features of modularity, attribute-based link semantics, the capability to organize information in various levels of abstraction. The model is inherently distributed and can be superimposed on the top of the existing hypertext systems, as long as the attributes associated with the chunks of information are provided. Therefore, the model may be used as a tool to interconnect the existing hypertext systems.

Based on the HyperNet model, we introduced the concept of user view (UV) of the underlying database, and presented algorithms to automatic generate such a UV, which provides a personalized, small and yet powerful accessing environment for user. As alternatives, we also suggested the global GSP view (GGSPV) and local HyperNet as part of the user browsing context. Each of the above emphasize different aspects of the system and compensate the weakness of others. Two browsing schemes, namely automatic browsing and user-controlled multi-level browsing, are presented based on the HyperNet model. The automatic browsing provides a guided tour for the user according to the attributes provided, and the user-controlled multi-level browsing scheme provides user a flexible browsing environment. As another way to solve the disorientation and to conduct efficient access, an attribute based concurrent search algorithm is presented based on the model, which conducts search parallelly in the relevant HyperNets, which may be allocated in multiple machines.

Since the model requires the user to provide the attributes for the places and/or transitions, a potential problem with the model is how to make the users to agree on a unified attribute set, i.e. how to prevent the users from using different attributes to describe the same property, and how much load the users have to take in order to provide such attributes. The study on this issue is still need to be done.

7. References


