Specification and Efficient Monitoring of Local Graph-based Constraints in Hypermedia Systems

Stephen Arnold , Leo Mark, and Sham Navathe
College of Computing,
Georgia Institute of Technology
Atlanta, Georgia 30332-0280

GIT-CC-94-56
November 29, 1994

Abstract

The concept of hypermedia has existed for about fifty years. It became a practical technology in the seventies, and widely available in the eighties. The concept has proven quite useful as a paradigm for information presentation, and been applied to information relevant to many diverse fields. However, the networks of semantic connections that exist in hypermedia systems are often so large and complex that they become overwhelming to people trying to find information in them.

This paper presents a number of types of constraints representing application semantics as a way of reducing the complexity of networks of semantic connections in hypermedia. Unlike constraints developed in the past, those presented in this work are graph-based and can be evaluated within local regions of the hypermedia system. In addition, this work presents an algebra on overview graphs of hypermedia systems. To formalize the definitions of the algebra and constraints, this paper presents a data model for hypermedia.

Also, this paper presents an algebra on networks of semantic connections in hypermedia. This algebra, in itself, can be used to define overview graphs on networks of semantic connections. In addition, the algebra increases the expressive power of the constraints by allowing the definition of overview graphs to which the constraints can be applied.

Keywords: Hypermedia, Algebra, Views, Constraints, Semantic Data Model

1 Motivation

Hypermedia is a very powerful technology for storing and accessing information which is highly, but not regularly, related. The concept was developed by Bush [Bush45]. He envisioned a system where information was stored on microfilm and some mechanical system was used to traverse related pieces of information. Though the concept is the same as the one found in modern hypermedia systems, microfilm technology did not provide a practical platform for implementing it.

The development of the computer provided a machine able to support Bush's concept. The high speed of information access, the ability to keep track of numerous and complex cross references between information, and the powerful methods of displaying information possible in computers today make these machines ideal platforms for hypermedia.

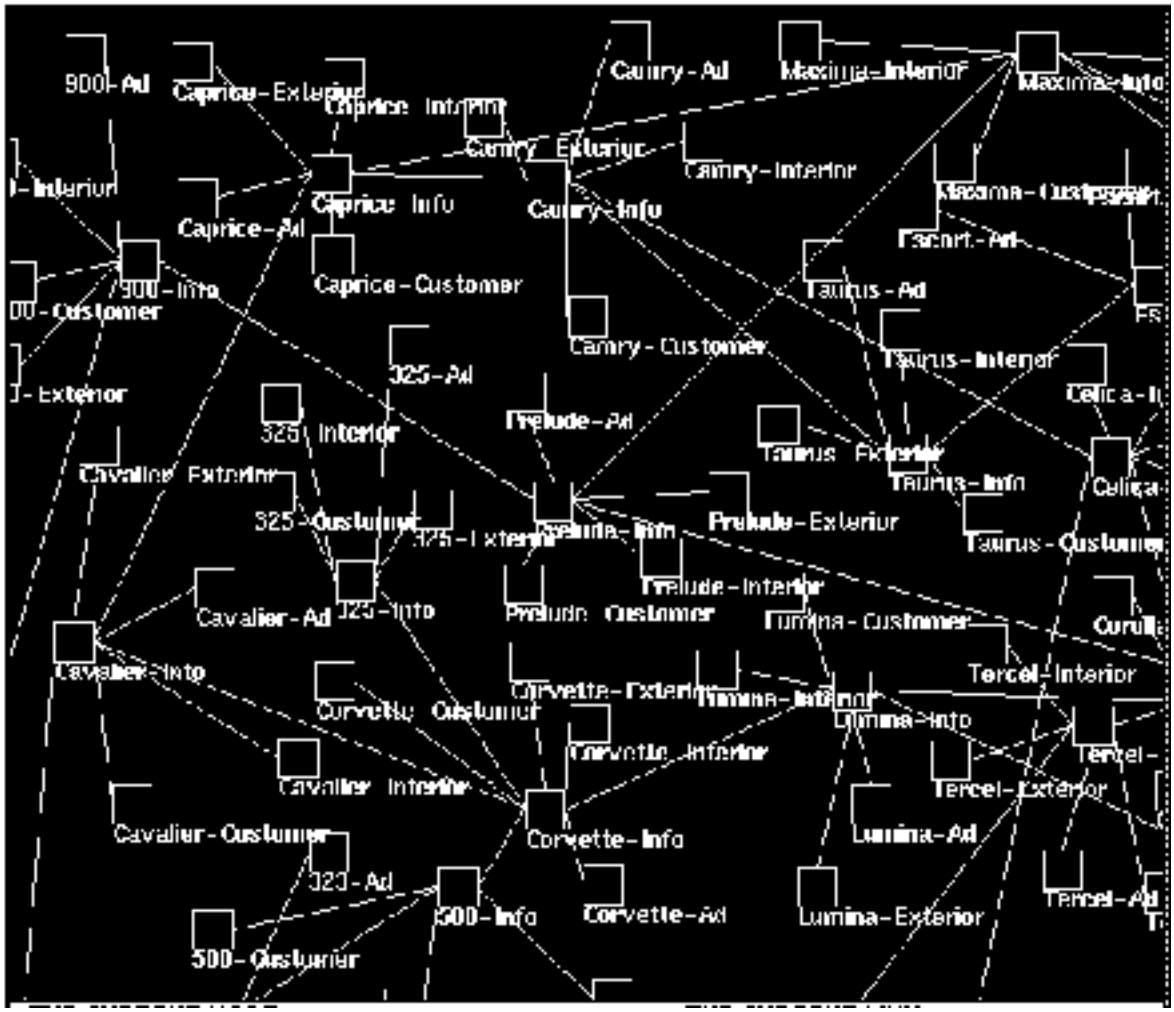
Once it became feasible to implement, hypermedia technology was used to develop a wide array of diverse applications. Hypermedia systems have been created to support patient care, intelligence gathering and analysis, car sales, education in a broad range of subjects, etc. It appears to be useful in any domain where pieces of information need to be related in some irregular pattern.

With the development of hypermedia systems, however, a major problem appeared. The problem stems from the fact that complex webs of information are to complex to be effectively used - even in hypermedia! Often, the networks of semantic connections in hypermedia become very large and complicated. This problem manifests itself in two ways: users can not find the specific information they want, and overviews of the semantic connections are too complex to be understood by users. The different descriptions reflect two different perspectives on hypermedia: users navigating from one piece of information to another and users trying to understand the structure of a hypermedia system as a whole. The different perspectives highlight the problem that the network of semantic connections in hypermedia systems can become too complex to be useful.

This problem is ironic considering that the objective of hypermedia technology is to present information in a structured and easily browsable form. But the problem is very real and limits the usefulness of these systems. The figure below, showing the semantic connections found in a sim-

ple auto broker application, clearly illustrates the problem. The complexity of this overview is so great that it makes the overview almost useless.

Figure 1. Overview graph of an autobroker network of semantic connections [Mukerjea94]



Nielsen surveyed users of a Guide document [Nielsen90] and found that 44% of them agreed with a statement that they could not find where they had previously been in the document. They were presented with too many choices of semantic connections to follow.

Different approaches to showing the connections have been tried. The fish-eye view is one of the most successful. But even the best of these techniques is limited, because no matter how well a confusing structure is shown, it is still a confusing structure. No matter how well a confusing network of semantic connections in a hypermedia system is shown, it is still a confusing network. This work tries to attack the problem at its root, by controlling the structure of the network of semantic connections.

In considering how to control the structure of networks of connections, two approaches have been considered: templates and constraints. The use of templates has been explored [Marshall91] [Marshall92]. In a discussion with Frank Halasz [Halasz93], he described the disadvantage of the template approach. Specifically, users dislike templates and, in many cases, refuse to used them. So this work focus on the other alternative, that of constraints.

Since a hypermedia system can be considered a type of database, it is natural to apply results from database research to hypermedia. This is especially helpful for the present work, because the use of constraints in databases is a well explored area - much research has focused on using constraints in databases to incorporate application semantics. From this research, the amount of time it takes to detect a constraint violation is one major difficulty in using constraints in databases. Many types of constraints are impractical because they make the transaction time in a database unacceptably long.

The biggest problem in constraint violation detection is that the detection algorithms may have to access every page of the database. The bottleneck in database systems is the time it takes to transfer data from disk to memory and back.

Thus, taking into consideration the disk access problem and the objective of simplifying the network of connections, this work explores local graph-based constraints. These constraints are local because for a single transaction only a local region of the database needs to be accessed to determine if a constraint is violated. Graph-based means that these constraints can be expressed as networks of connections.

Though these constraints by themselves look promising for being able to limit the complexity of networks of semantic connections in hypermedia systems, there usefulness can be increased. Often users do not wish to view the raw data in a database, but rather work with abstractions on the raw data. A traditional approach to defining these abstractions is an algebra. An algebra, once defined, can allow the user to constrain abstractions of the base semantic connections in addition to the base connections themselves. This paper presents an algebra to can be used to construct views which, in turn, can be constrained by the local graph-based constraints.

To be able to define both the constraints and the algebra, a formal description of semantic connections in a hypermedia system is needed. Hence, this paper presents a formal model for semantic connections in hypermedia systems.

In summary, hypermedia systems are here, and here to stay. The networks of semantic connections in these system can become too complex to be useful. This problem is addressed by developing constraints, an algebra, and a data model. The constraints cannot be allowed to slow transaction processing of the hypermedia system too much, so this work will explore efficiency issues such as localization and caching. The algebra increases the power of the constraints by allowing them to be used on both raw data and on abstractions on the raw data. Lastly, to be able to define the constraints and the algebra, a formal model of hypermedia is needed.

The structure of the remainder of this paper is as follows. In section 2, the background is discussed. This section is divided into three parts: constraints, hypermedia, and other related work. In section 3, this paper presents high level description of hypermedia concepts. In section 4, this paper describes the object-oriented data model on which the work here is based. In section 5, this

paper presents a formal data model of the semantic connections in hypermedia. In section 6, this paper defines an algebra for the semantic connections in hypermedia. Section 7 presents constraints on the semantic connections. This section consists of two parts: definitions of constraints and algorithms for constraint violation detection. In section 8, this paper presents the direction which the authors plan to continue this research. Section 9 describes the impact of this work.

2 Background

This work represents an intersection between two different areas: constraints and hypermedia. To the authors' knowledge, no work dealing with this intersection exists other than that presented here, but a great deal of research has been done in each of the individual areas. This work is based on results and approaches from several areas within computer science including formal algebras, views, and data modeling. This paper addresses the background in each of these areas below.

2.1 Constraints

Constraints are used in the area of artificial intelligence, but the emphasis and thus the results are not closely related to this work. Constraints in this area aim to capture information and adapt information to changes. The goal of this work is to develop efficient algorithms for constraint violation detection, and the constraints in this work simply reject changes that would violate constraints. Some examples of artificial intelligence systems that use constraints are the planning system of MOLGEN [Stefik81.1] [Stefik81.2] and the common sense reasoning system Cyc [Lenat90].

Graphical user interfaces and simulation systems use constraints in very similar ways. In both areas, the constraints are arithmetic equations with minor extensions such as conditionals. The emphasis of the work is to quickly recompute values to adapt to changes.

Within the graphical user interface area, there is a significant quantity of work in the area of inferring constraints. In this work, constraint equations are inferred from demonstrations or examples. DEMO II [Fisher92] and Chimera [Kurlander93] are two systems that infer constraints.

In addition, computer languages extended to support use of constraints have been used to create interfaces and support simulations. Hudson and Yeats [Hudson91] have developed a visual language for constraint specification. Eval/vite [Hudson93], ThingLab [Borning81] [Borning87] and its extensions [Freeman89] [Freeman90], and RENDEZVOUS [Hill93] are extensions of C++, Smalltalk, and CLOS (an object-oriented LISP), respectively, which allow these languages to include constraints.

Vander Zanden [Zanden91] and Gleicher [Gleicher93] have investigated issues in constraint implementation. Vander Zanden extended constraint definitions to support indirect referencing order, leading to an increase in the expressiveness of constraints. Gleicher used an event-based model for resolution of constraint violations. His approach is very similar to the constraint work in databases.

The work presented here takes a different approach to constraints than is used in the areas of user interface research and simulation systems. This work is not concerned with adapting to changes that violate constraints, nor are these constraints arithmetic in nature. The aim of constraints in both user interfaces and simulation systems is to adapt to changes and to represent and enforce arithmetic relationships.

Constraints in databases are handled through the use of triggers. Cochrane [Cochran92] presents the event-condition-action model of triggers, and a taxonomy with fourteen axes: control strategy, event type, event specification, processing granularity, condition scope, termination condition, event-condition binding, condition-action binding, transition value reference, net effect,

conflict resolution and programming support. The authors are aware of at least seventeen different systems that are effectively categorized within this taxonomy [Anwar93], [Cervesato92], [Delcambre89], [Dia91], [Eick93], [Gala93], [Garcia94], [Gertz93], [Han91], [Hanson89], [Ishikawa93], [Jagadish92], [Mark91], [McCarthy89], [Stonebraker88], [Stonebraker89],[Widom90], [Zhou90]. The constraints used in the work presented here can be classified in this taxonomy as well. However, five of the fourteen axes do not apply. These are the following: termination condition, transition value references, net effect, conflict resolution, and programming support.

Use of constraints in object-oriented databases has been explored. There are two approaches to constraints that can be taken in such systems: either constraint definitions are placed inside of objects[Formica92], [Martin92], or are kept external to them[Shack93], [Yoon92]. This distinction is important in looking at the degree of support for inter-object vs. intra-object constraints. Slack and Unger [Slack93] present convincing arguments that constraints should exist at the schema level. The issue of constraint bind time has been explored[Martin92], [Karadimce93],as in Cochrane's work, but under the name of soft constraints verses hard constraints. Soft constraints are those that can be violated during a transaction, but not between transactions. Hard constraints are never allowed to be violated. The constraints in the work presented here are schema level and soft.

The object-centered approach to constraints proposed by Delcambre et. al. [Delcambre91] is fundamental to the work presented here. An object-centered constraint is one for which the set of objects needing to be examined to detect constraint violation can be discovered by tracing a path originating at any member of that set. Whereas Delcambre applied this concept to Prolog Databases, this work applies it to graphs.

2.2 Hypermedia

A great deal of work has been done in modeling hypermedia. One of the most significant models is the Dexter Hypertext Reference Model (DHRM) [Halasz89] [Halasz94] [Leggett94] [Gronbaek94.1]. This model is based on Z [Z] and divides hypermedia into three layers: run-time layer, storage layer, and internal information chunk structure layer. The practical issues of implementing hypermedia systems in this model have been explored [Gronbaek93] [Gronbaek94.2]. It has also been extended to address temporal aspects of display [Hardman94].

The author's previous work [Arnold94] differs from the DHRM in proposing a schema level for semantic connections, which DHRM does not do. In addition, this work maintains a distinction between relationships and all other information, whereas DHRM allows the structures that store relationships to contain arbitrary data.

There are several other model of hypermedia present in the literature, besides DHRM. Each model addresses different issues. HAM [Campbell88], Conklin[Conklin88], Garg[Garg88], Lange[Lange90], and Parunak [Parunak91] all present general models of hypermedia. Unlike other models, Parunak presents a model of semantic connections represented as sets rather than edges. None of these models has a schema level nor addresses controlling the network of semantic connections like this work does.

Many specialized models have been developed as well. Afrati and Koutras [Afrati90] look at querying in hypermedia. αTrellis [Stotts90] models information traversal and timing issues in such traversals. Bra et al. [Bra92] present a metaphor for capturing aggregation of information. Zheng and Pong [Zheng92] use state charts to model browsing semantics. The author has developed the Grain-Node-Role-Link model for hypermedia [Arnold94] from observing user interactions.

There has been some previous work on the topic of controlling networks of semantic information in hypermedia. Templates were developed to aid users in structuring the hypermedia links needed by applications[Marshall91] [Marshall92]. A disadvantage of this approach is that users seem to have difficulty using template structures, and avoid using them in practice[Marshall92] [Halasz93]. This has led the author to conclude that a better approach might be to allow users to structure information in a flexible manner (without the use of templates), and use constraint checking to enforce structure requirements.

2.3 Closely Related Work from Other Areas

The algebra described below is very similar to an object-oriented algebra. Yu and Osborn [Yu91] have developed a framework for evaluation of object-oriented algebras based on the following criteria: object-orientedness, expressiveness, degree of formality, performance, and how well database issues are handled. The algebra described does well when evaluated with these criteria.

Several object-oriented algebras have been created. Bertino at al. [Bertino92] create a calculus based on first order logic and recursion. Liu [Liu93] develops an algebra based on recursion and aggregation. Su, Guo, and Lam [Su93], [Gou91] develop an algebra based on associations. In contrast to these other algebras, the one presented in this work strives for similarity to the relational algebra, so as to achieve ease-of-learning and ease-of-use.

Agrawal and DeMichiel [Agrawal94] explore type derivation for the project operation, although their work violates the object-oriented principle of encapsulation. Alhajj and Arkun [Alhajj92] use sets to emulate relational and nested algebras. Kim [Kim92] develops yet another relational based algebra, in which selection is used to produce a form of semi-join.

Among all the algebras referenced above, the one presented in this paper is most similar to Kim's in that both use traditional relational operators. There are significant differences, however. For example, the algebra described below maintains encapsulation better than that of Kim. In addition, while Kim's algebra functions on a domain of objects, the one developed here functions on graphs composed of objects.

Hy+ [Consens93] and the work of Catarci et al. [Catarci93] develop algebras for graphs. Hy+ uses a different subset of hypergraphs than used in this work. Catarci overlays graphs onto existing data models. The work presented here targets the specific application of hypermedia instead of functioning as a layer above another data model.

The Classic database [Borgida89] addresses issues similar to those addressed in this paper. Classic generalizes data into classes, but the nature of these generalizations is limited because of

tractability issues. Similarly, the work presented here attempts to find patterns in base data, but limits the kinds of patterns due to tractability issues.

3 Hypermedia Concepts

The concepts in this work are based in part on the Grain-Node-Role-Link (GNRL) model described by the author in a previous paper [Arnold94]. Rather than merely trying to model the semantic linkages existing in a hypermedia system, the GNRL model takes a user's point of view, and attempts to describe the kinds of relationships that users have in mind as they traverse hypermedia links. Using this model as a starting point, the authors can then define constraints which will reflect the types of semantic relationships that users are interested in.

The GNRL model was developed based on physician interactions with a hypermedia system when performing patient evaluations. The authors believe that the concepts of grains, nodes, roles, and links capture the information relationships required to model user interaction in hypermedia. A node is a chunk of information. A link is a named semantic connection between nodes. The name of a link denotes the relationship existing between the nodes. A link is composed of roles, each of which is a named set of nodes. The role name denotes the role that the nodes in the role play in the link. A grain indicates a section of a node which is related by a link. The concept of a grain is very closely related to the known hypermedia concept of anchors.

The algebra and constraints developed in this paper utilize the concepts of nodes, roles, and links. Thus, these concepts of hypermedia and those in an object-oriented data model provide the formal foundation of this work.

4 Object-Oriented Data Model

The abstract concepts of hypermedia provide a model for information structures, but not a model for the storage and behaviors needed in hypermedia systems. There are a wide variety of models which could provide a practical platform for implementation, such as the relational data model, the entity-relationship data model, and third generation programming languages. However, none of these effectively captures both information structures and desired behavior, such as the ability to store, display, and traverse information. Since these behaviors are critical for any hypermedia system, this paper adopts a model that allows specifying them along with the information structures. This combination of data storage and behavioral needs is best modeled with an object-oriented data model.

Objects support the storage and display of information, and can hold arbitrary pieces of information such as nodes. They can also have any desired programmable behavior such as display routines. They are not good for supporting links, but this weakness can be overcome by defining objects that specifically support the links needed to allow users to traverse networks of semantic connections.

Many different object-oriented data models exist. This paper does not use a specific model in this work, but work within the set of models that have classes and attributes which take values. Classes are abstract descriptions of objects that include the definition of objects' attributes and methods. The attributes of an object hold the values of the object's state. The methods of an object define the behaviors of an object. When receiving a method list, an object returns a value. C++ and ORION [Kim92] are examples of object-oriented systems that have these characteristics.

This paper uses an object-oriented data model having classes because this work relies on the able to refer to groups of objects based on a common abstract description. Subclasses enhance the ability to refer to groups of objects. This work uses attributes that can take values because the work relies on the able to refer to the content of the state of an object, but can allow some objects' states to contain arbitrary values.

5 Formal Model

The GNRL model and an object-oriented data model with classes and attributes that take values provide the basis for the formal model presented in this paper. The model is based on hypergraphs [Berge73]. The model is augmented with both blocks and behaviors to create labeled hypergraphs with blocks (LHGB). This graph theoretic formalism provides a model on which the algebra and constraints can be defined.

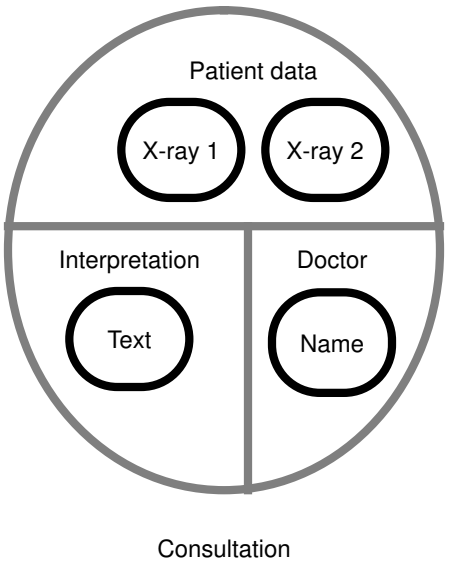
A **LHGB** consists of nodes, blocks, and links. A **node** is a labeled vertex with behaviors. Its label is part of a hierarchy with "node" as the label at the root of the hierarchy. A method list can be sent to a node, and the node will respond with a value. A node responds with the same value every time it is sent the same method list. A node in a LHGB represents a node from the GNRL model with the same label. A node responds to messages like objects do in an object-oriented data model.

A **block** is a labeled set of nodes. The label is the same as the label of roles from the GNRL model it represents.

A **link** is a labeled set of blocks, where each block within the set is uniquely labeled. A link must contain at least two blocks, and at least two of the blocks in a link must contain at least one node each. A link represents a link from the GNRL model and provides a structure to overcome the weakness of object-oriented data models in supporting relationships. Links respond to method lists the same way nodes do.

The figure below shows a consultation link. It has three blocks: *patient data*, *interpretation*, and *doctor*. Each block contains nodes. The patient data block contains two *x-ray* nodes, the interpretation block contains a *text* node, and the doctor block contains a *name* node.

Figure 2. Example of nodes and links in LHGB notation



6 Algebra

The algebra was designed using the following principles:

- operators in the algebra should resemble traditional database operators,
- the algebra should be closed on the set of LHGBs
- the results of operations should be strongly typed
- the results of an operation should be no more complex than the original operands, where the level of complexity of an LHGB is measured by the number of links, blocks, and nodes it contains.

The algebra has three kinds of operators: node, link, and graph. Node operators act on nodes and induce a set of links. Link operators act on links and induce a set of nodes. Graph operators act on graphs as a whole.

The syntax is simple and traditional. Node and link operators are unary and prefix, and, in some cases, take lists as subscripts. The graph operators are binary and postfix. Operators are evaluated from left to right, except when parenthesis are used to specify otherwise. For operators which take a method list as input, evaluation is done in a sequential fashion; the first method in the list is applied to the original operand, the next method is applied to the results of that operation, and so on until all methods in the method list have been applied. The result of the operation as a whole consists of the output from the application of the final method.

The node operations are **node select** $\sigma_{\mathbf{n}}$ and **group** γ. The nodes in $\sigma_{n\ [method\ list]}$ (G) are all of the nodes in G which return the value *true* in response to the method list. The induced set of links in $\sigma_{n\ [method\ list]}$(G) consists of links in G which contain at least one node in each of two different blocks. These links are missing any node that is not present in the nodes of $\sigma_{n\ [method\ list]}$ (G). This operator is similar to the selection operator in Kim's algebra because it can be used to select elements based on their characteristics and do semi-joins along paths of methods [Kim92].

To define the group operator, an aggregation node needs to be defined. An **aggregation node** is labeled "aggregation". It contains two attributes: *value* and *nodes*. The value attribute contains a value and the nodes attribute contains a set of nodes.

The group operator groups nodes based on a common characteristic. The result of applying $\gamma_{[method\ list]}$ to a graph G is a set of aggregation nodes which contains one node for each different value produced by sending the method list to every node in G. Each aggregation node's value attribute is one of these produced values, and its nodes attribute is the set of nodes in G which responded with that value. The links in $\gamma_{[method\ list]}$ (G) are induced as follows: if a node N appears in an aggregation node's nodes attribute, then the aggregation node is contained by a block with the same block label as N's block, and participates in links with the same link labels as N.

The link operators are **link select** $\sigma_{\mathbf{l}}$, **project** π, and **join** χ. The links in $\sigma_{l\ [method\ list]}$ (G) are those links in G which respond *true* to the method list. The induced set of nodes in link select $\sigma_{l\ [method\ list]}$ (G) is the set of the nodes participating in the links of $\sigma_{l\ [method\ list]}$ (G). This operator is similar to the selection operator in Kim's algebra because it can be used to select elements based on a characteristics and do semi-joins along paths of methods [Kim92].

The **project** operator π projects over a set of blocks. It has two forms, project-on and project-off. The link set of $\pi_{[\text{ label, label,...}]}(G)$ is a set of links derived from the set of links in G as follows: for each link L in G that contains at least two blocks labeled with labels from the list and having at least one node each, there is a link L' in $\pi_{[\text{ label, label,...}]}(G)$. Each link L' contains only those blocks from L which have labels found in the label list. The link set of $\pi_{[\text{ - , label,...}]}(G)$ is defined similarly, except that blocks in the result are those which do not match the labels in the list. The induced nodes in both results are those that are present in the links of the result.

Before defining the join operation, a path pattern needs to be defined, and a matching path pattern. A **path pattern** is a list of labels in the following pattern:

<block label> [<node label> <block label> <link label> <block label>] * <node label> <block label>

A **path** is a list of blocks, nodes, and links with the following characteristics. A block is either followed by a node which is contained in the block or a link which contains the block. A link is only followed by a block which it contains. A node is only followed by a block that contains the node. A block with a node in front of it is only followed by a link. A **path pattern matches a path** if the labels in the path pattern match the labels of the elements in the path.

Join χ joins two links which are connected by a specified path into one link. The set of links in $\chi_{[\text{ link type, link type, path pattern}]}(G)$ is a subset of the union of the blocks in each element of the cartesian-product of the links of the first link type with the second link type. If there exists a path from the first link in an element of the cross-product to the second link, and the path matches the path pattern, then that element is in the result. The nodes in $\chi_{[\text{ link type, link type, path pattern}]}(G)$ are the nodes contained in the links of the result.

To preform a link operation such as project or join on a graph, some scheme is needed for deriving names for the new links. This scheme works in the following manner: each link L from G that participates in the derived link L' is assigned an extended name that consists of listing the blocks found in that link, and labelling this list with the name of the link L. For example, the Consultation link shown previously can be named Consultation-{Patient Data, Doctor, Interpretation}. Then, when a link operation is preformed such as project or join, this extended name can be used to derive a new link name for the result. When project on G is preformed, the block list is merely updated to contain the names of those blocks that appear in the result. Hence, if the Consultation link is found in G, then $\pi_{[\text{ lDoctor,Interpretation}]}(G)$ will result in a new link called Consultation-{Doctor, Interpretation}. If a join is preformed the extended names are concatenated. Hence, a join on links Consultation-{Patient Data, Doctor, Interpretation} and Education-{Student, School} will result in a new link called {Consultation-{Patient Data, Doctor, Interpretation}, Education-{Student, School}}

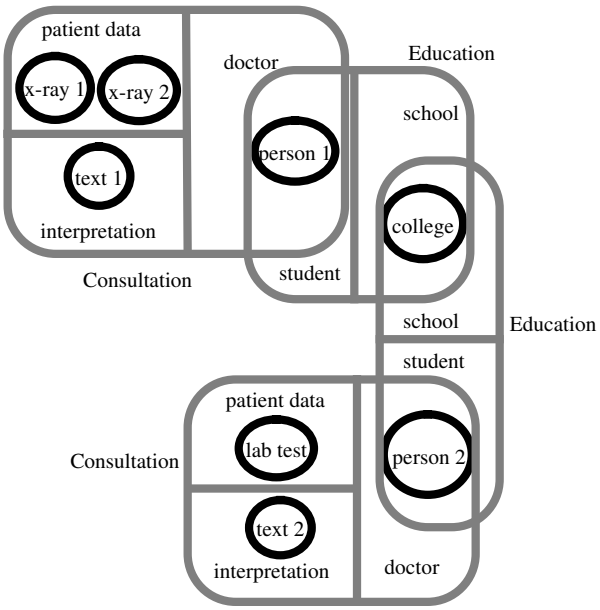
Before defining the graph operations, a definition of link equality is needed. Two **links are equal** if they have the same label and contain identical blocks. Blocks are identical if they have the same block label and contain the exact same nodes.

Union + combines two LHGBs together. The set of links in $G_1 + G_2$ is the union of the set of links in G_1 with the set of links in G_2. The set of nodes in $G_1 + G_2$ is the union of the set of nodes in G_1 with the set of nodes in G_2.

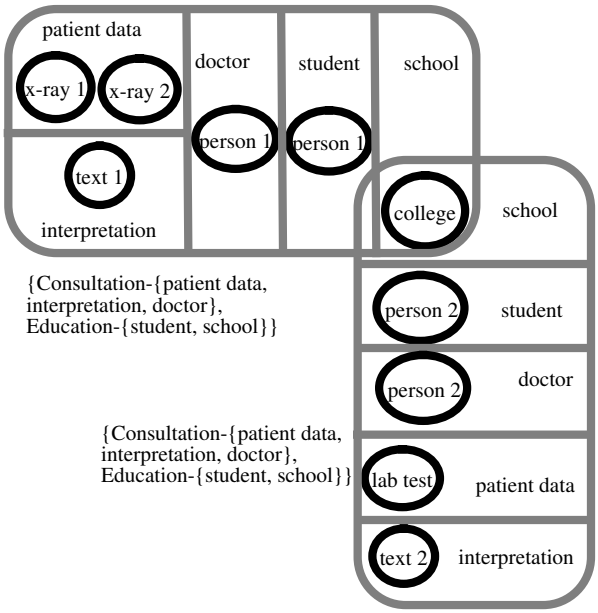
Set subtraction removes one LHGB from another. The set of nodes in G_1 - G_2 is the set of nodes in G_1 minus the set of nodes in G_2. A link *l'* appears in G_1 - G_2 for every link *l* such that *l* is in the set difference between links in G_1 and links in G_2, and at least two nodes from different blocks in *l* appear in G_1 - G_2. For every block *b'* in *l'*, there exists a block *b* in *l* such that *b* and *b'* have the same label and the set of nodes in *b'* is the intersection of the set of nodes in *b* and the set of nodes in the result G_1 - G_2.

Below, this paper provides an example of these the join and project operators applied to a small graph.

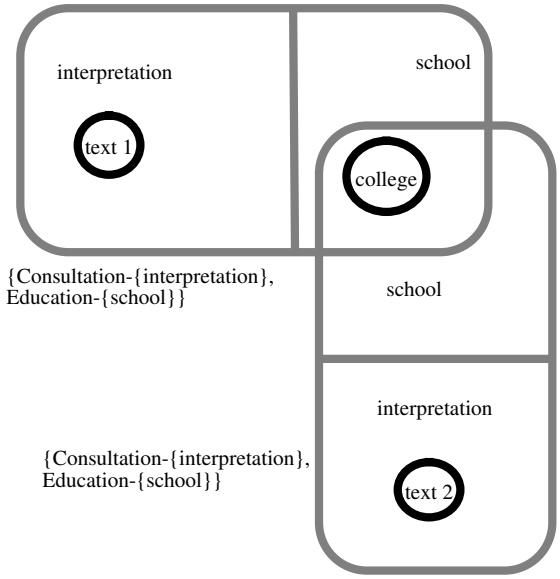
G:



$\chi_{[consultation,\ education,\ doctor\text{-}person\text{-}student]}(G)$:



$\pi_{[interpretation,\ school]}(\chi_{[consultation,\ education,\ doctor\text{-}person\text{-}student]}(G))$:



7 Constraints

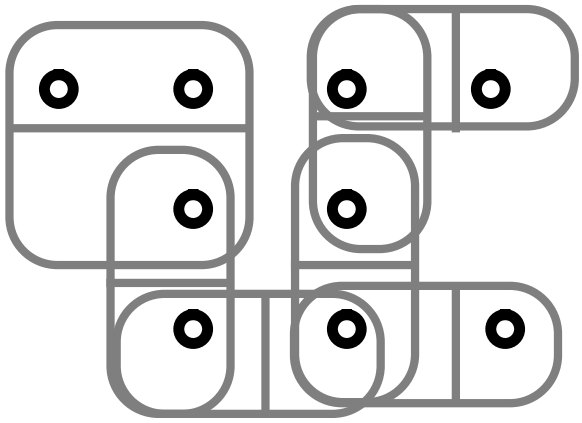
The formalism of labeled LHGBs is used to represent the constraints placed on networks of semantic connections in hypermedia systems. For reasons of efficiency, the constraints will be defined in such a way that they only require access to a local region of the graph representing a network of semantic connections in order to determine if they are violated.

Below this paper will define three different types of constraints: density, graph exclusion, and subgraph containment. This paper will also present algorithms to detect violations of these constraints when a link is added or deleted.

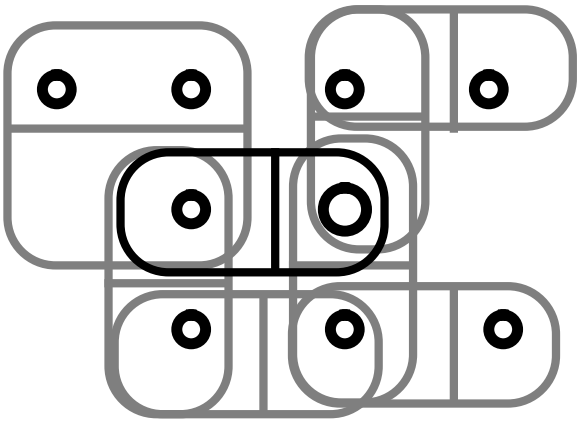
7.1 Definitions

A **density constraint**, as its name implies, puts a limit on the density of nodes appearing in a graph. The specification for a density constraint is a fraction t/n, where t is a number of traversals, and n is a number of nodes. A network of semantic connections satisfies the density constraint t/n if, for all nodes in the LHGB which represents that network, no more than n nodes can be reached by all possible paths of t traversals. To fully understand this definition, it is necessary to know precisely what a traversal consists of. A users **current location** in a network of semantic connections is a specific node. Traversal in the network means changing the current location from one node to another. A single traversal means changing the current location from one node within a block to another node in a different block where both blocks are in the same link. For example, specifying that a network of semantic connections satisfies a density constraint of 2/7 means that, in the LHGB representing that network, it is impossible to find a node from which more than 7 nodes can be reached by following all paths of 2 (or fewer) traversals.

A LHGB that satisfies the density constraint of 2/7:



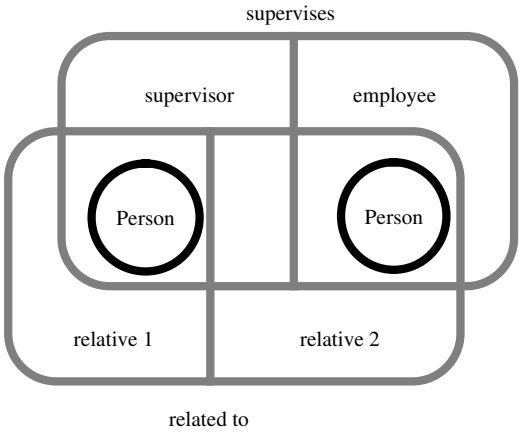
A LHGB that violates the density constraint of 2/7:



The definitions of the graph exclusion and subgraph containment both rely on the concept of **subgraph isomorphism**. This isomorphism is more than a traditional graph isomorphism. For graphs to be isomorphic, they must have the same structure, and elements of the two graphs which correspond in the isomorphism must have the same label. A subgraph isomorphism exists between two labeled hypergraphs if the second hypergraph contains a subgraph that is isomorphic to the first hypergraph.

A **graph exclusion constraint**'s specification is a continuous LHGB. A network of semantic connections satisfies a graph exclusion constraint G if there is no subgraph isomorphism between G and the LHGB which represents the network of semantic connections. For example, given links that represent the semantic connections of 'membership in the same family' and 'employment supervision', a graph exclusion constraint can be constructed to prevent nepotism.

Nepotism is not allowed:

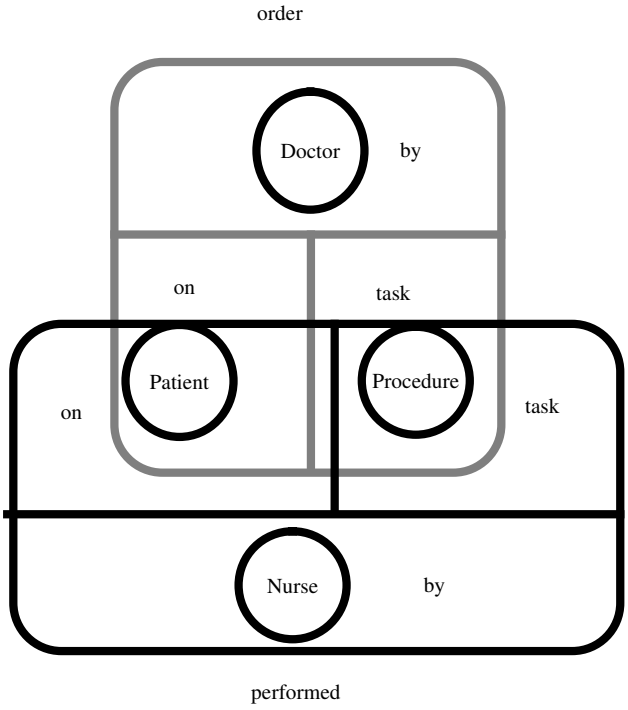


A **subgraph containment constraint**'s specification consists of two continuous LHGBs, the first being a subgraph of the second. A network of semantic connections satisfies the subgraph

containment constraint G_1 in G_2 if all subgraphs of the LHGB which are isomorphic to G_1 are contained in graphs isomorphic to G_2.

For example, in a network of semantic connections containing links for doctors requesting procedures to be performed on patients, and links for procedures having been performed on patients by staff members, a constraint could be specified so that a procedure can not be performed without having been ordered.

All procedures must be ordered by a doctor:



7.2 Algorithms

When a semantic connection is inserted into or deleted from a network of semantic connections, a constraint may be violated. Hence. algorithms are needed for detecting such violations. To simplify the development of these algorithms, the paper shall consider only addition or deletion of a single link.

In considering the density and graph exclusion constraints, deleting a link does not pose the possibility of violation. By deleting a link, more nodes do not become accessible in a given number of traversals. Similarly, by deleting a link, a subgraph which was not originally present in a LHGB, does not appear. Therefore, constraint violation detection algorithms are needed only in the case of addition of links in LGHBs bound only by these types of constraints.

For the subgraph containment constraint G_1 in G_2, algorithms are needed for both addition and deletion of a link. When a link is added, a subgraph isomorphic to G_1 may be created. When a link is deleted, a subgraph isomorphic to G_2 may be removed while a subgraph isomorphic to G_2 still exists.

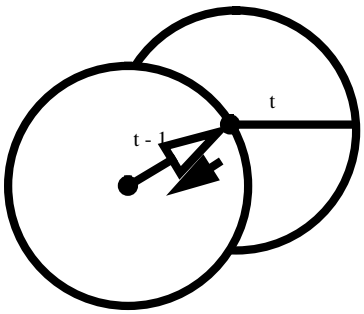
Because of space limitations, this paper will only present informal descriptions of these algorithms.

Algorithm for detecting violation of density constraint, t/n.

Input: t/n, a LHGB G that does not violate the density constraint t/n, a link *l* to add to G.

Output: yes (the constraint is violated) or no (it is not).

Let G' be G with *l* added to it. In G', gather the set of all nodes reachable by paths of t - 1 links from all nodes in *l*. For each node *x* is this set, gather the set of all nodes that can be reached by a path of t links starting at *x*. Count the number of nodes in each of these sets. If that number is greater than n for any of these sets, then the constraint is violated, otherwise it is not.



The algorithms that detect constraint violations for graph exclusion and subgraph containment constraint rely on finding subgraph isomorphisms. To simplify these algorithms, this paper will present an algorithm for finding subgraph isomorphisms and then use this algorithm in the remaining three algorithms.

Algorithm for detecting subgraph isomorphisms (DSI).

Input: A continuous LHGB G_1, another LHGB G_2, and an isomorphism between a continuous subset of links appearing in G_1 and G_2.

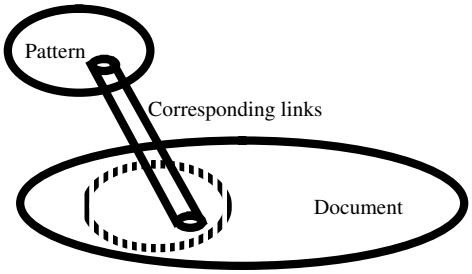
Output: an isomorphism between G_1 and a subset of G_2 or the empty set.

Place an arbitrary total ordering on the links in G_1 and G_2. Create an empty set B which will be used to hold the pairs of elements that cannot be part of the isomorphism. Create an empty list C that will be used to hold pairs of elements from the graph that are part of the isomorphism.

Find the least link l_1 in G_1 which is not part of the input isomorphism and is not in C but which has at least one node in common with a link in the isomorphism. Find the least link l_2 in G_2 which could become part of the isomorphism such that l_1 and l_2 are paired in the isomorphism and (l_1, l_2) is not in B. Add (l_1, l_2) to the end of the list C. Repeat this process until the isomorphism contains all links in G_1, or no pair (l_1, l_2) can be found. When no pair (l_1, l_2) can be found, remove the last element of C and place it in B. Repeat this process until all links in G_1 are in the isomorphism or no pair (l_1, l_2) can be found.

If an isomorphism containing all elements in G_1 is found, that is the output. If no such isomorphism is found, the output is the empty set.

.



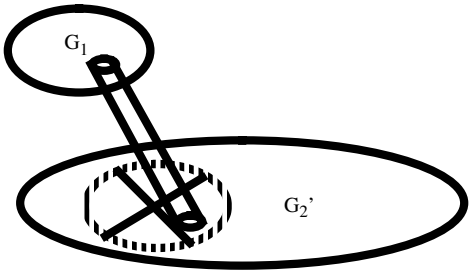
The algorithm for detecting a violations of a graph exclusion constraint is a direct application of the DSI algorithm.

Algorithm for detecting a violation of the graph exclusion constraint, G_1.

Input: A graph exclusion constraint, G_1, a LHGB G_2 that does not violate the graph exclusion constraint G_1, and a link *l* to be added to G_2.

Output: yes (the constraint is violated) or no (it is not).

Let G_2' be G_2 with *l* added to it. Let G_3 be the LHGB consisting of *l* and the nodes within *l*. Find all possible subgraph isomorphisms between G_3 and G_1. For each isomorphism, apply the DSI algorithm to find G_1 in G_2' using the isomorphism between G_3 and a subset of G_1. If the DSI algorithm ever finds a subgraph isomorphism between G_1 and G_2', adding *l* violates the constraint, otherwise it is not.



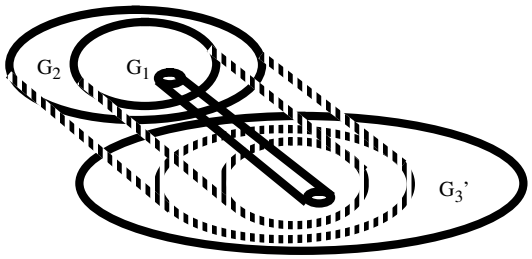
The algorithm for detecting a violation of a subgraph containment constraint when a link is added simply applies the DSI algorithm twice, first to find the inner graph pattern G_1, the second to find the outer graph pattern G_2. It also produces isomorphisms which will be used by the algorithm for detecting a violation of the subgraph containment constraint G_1 in G_2 when a link is deleted.

Algorithm for detecting a violation of the subgraph containment constraint S in G when a link is added.

Input: A subgraph containment constraint G_1 in G_2, a LHGB G_3 that does not violate the subgraph containment constraint G_1 in G_2, and a link *l* to be added to G_3.

Output: yes (the constraint is violated); or no (it is not), and a set of paired subgraph isomorphisms. The first element of the pair is a subgraph isomorphism between G_1 and G_3'; the second between G_2 and G_3'.

Let G_3' be G_3 with *l* added to it. Let G_4 be the LHGB consisting of *l* and the nodes within *l*. Find all possible subgraph isomorphisms between G_4 and G_1. For each isomorphism, apply the DSI algorithm to find G_1 in G_3' using the isomorphism between G_4 and G_1. For each subgraph isomorphism between G_1 and G_3', apply the DSI algorithm to find G_2 in G_3' using the isomorphism with G_1. If an isomorphism with G_2 is not found for each isomorphism with G_1, the constraint is violated, otherwise it is not. If the constraint is not violated, create pairs of all isomorphisms of G_1 with the isomorphism with G_2 which was found using the isomorphism with G_1. Gather these pairs into a set.



In simpler terms, find all occurrences of G_1 around the added link. Then determine if each instance of the pattern G_1 is contained by a pattern G_2. If a pattern G_1 is found in a pattern G_2, keep the pair for use in the algorithm for detecting a violation of the subgraph constraint G_1 in G_2 when a link is deleted.

The algorithm for detecting a violation of the subgraph containment constraint G_1 in G_2 when a link is deleted assumes that the network of semantic connections G_3,which the algorithm is applied to, was built in a certain manner. Specifically, it starts with an empty network of semantic connections and add links, one at time, from G3. Between each link addition, the above algorithm is applied and the isomorphism pairs are associated with G_3. This will ensure the availability of all pairs of subgraph isomorphisms needed for the algorithm presented below to function correctly.

Algorithm for detecting a violation of the subgraph containment constraint G_1 in G_2 when a link is deleted.

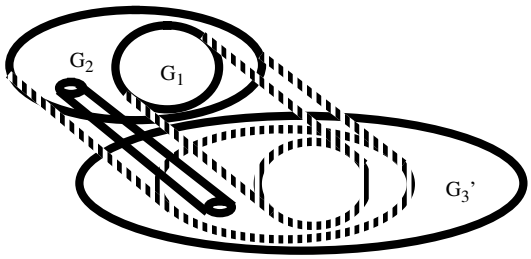
Input: A subgraph containment constraint G_1 in G_2, a LHGB G_3 that does not violate the subgraph containment constraint G_1 in G_2, all the pairs of subgraph isomorphisms P that was produced when G_3 was constructed as described above, and a link *l* to be deleted from G_3.

Output: yes (the constraint is violated); or no (it is not) and a set of pairs of subgraph isomorphisms. The first element of the pair is a subgraph isomorphism with G_1; the second with G_2.

Let G_3' be G_3 with *l* deleted from it. Gather all (I_1, I_2) elements of P where there exists some l_1 such that (l_1, *l*) is an element of I_2 but not I_1. Let this set of pairs of isomorphisms be called P_1. For each I_1 where (I_1, I_2) is an element of P_1, apply the DSI algorithm to find G_2 in G_3' using I_1 as the isomorphism. Let the output of DSI be called I_3. If DSI algorithm does not find a subgraph

isomorphism, then the constraint is violated. If it is not violated, create a set of pairs (I_1, I_3). Let this set be called P_2.

Gather all (I_1, I_2) elements of P where there exists some l_1 such that (l_1, l) is an element of I_1. Let this set of pairs of isomorphisms be called P_3. Return the set of isomorphisms (P - (P_1 + P_2)) + P_2, where - is set subtraction and + is union.



In simpler terms, if a link is removed that was of some pattern G_2, but not part of the associated pattern G_1, then try to find another pattern G_2 containing that pattern G_1. If it is found, remove the old isomorphism pair and replace it with the new one. If a link is removed from some pattern G_1, then remove all pairs of isomorphisms containing that instance of the pattern G_1.

8 Future Work

This work will be continued by three different task. The formalisms developed will be extended and the theory will be developed. A practical example will be explored. And this design will be integrated into a transaction oriented system.

The first task will continue the development of the formal model, the algebra, and the constraints. This work will consist of developing formal definitions (based on set theory) and mathematical proofs. The work will develop a set theoretical definition of LHGB model for networks of semantic connections in hypermedia. This formalization will help us in the exact specification of the algebra and the constraints and provide a context to perform formal analysis of the data structures and algorithms related to the algebra and the constraints.

Transformation rule will be developed for the algebra. These rules will be similar to the equivalence rule for relational algebra. The correctness of these rules will be verified by formal proofs based on graph theory.

The constraints will also be defined in terms of the graph theoretic formalisms mentioned above. This formalism will be used as a basis to study the decideability of determining if a constraint is violated in a network of semantic connections.

The second task will be to identify a compelling example of a hypermedia system which would be improved by the use of the above defined algebra and constraints. This task will first show the difficulties that occur when the network of semantic connections is not controlled. Then it will justify the use of the methods of control (the algebra and constraints) as a way to reduce these difficulties.

The last task is to take the developed theory mentioned above an show how it can be used in a realistic system. The constraint violation detection algorithms are defined in an unrealistic environment. They are defined assuming that only one semantic connection is added or deleted at a time, and only one constraint will be enforced. A more realistic environment can have many semantic connections added and deleted in a single transaction and will have multiple constraints applied at once. The algorithms mentioned above will be extended to work in this environment.

Maintaining views is a very costly operation. In order to evaluate constraints on views, some view maintenance will be needed. For constraints function on views, these views must be maintained constituently. This task will address view maintenance. This work will be based on the existing work done by [Blakeley86], [Blakeley89], [Tampa88], and [Roussopoulos91]. Though there work was developed for a different kind of database, it appears that much of it will transfer into view maintenance on hypermedia systems.

Constraint violation detection may cause transaction times in database to be so long that the database becomes unusable. Specifically, the database may become unusable because the number of page accesses for constraint violation detection becomes very large. This work addresses this issue by developing a cashing strategy. A local region cashing policy can be developed based on the object-centered nature of the constraints.

This work will also develop cost formulas for the view maintenance and the constraint violation detection algorithms. These formulas will be verified by a combination of formal proofs and simulations.

9 Effect of This Work

Finding a "specific address" of a piece of information in a hypermedia system is difficult. Developing an understanding of a whole network of information is almost impossible. An example of such difficulties occurs with the World Wide Web. Navigating though the World Wide Web, for instance, requires a strong set of computer skills and a great deal of persistence. As more information becomes available and more people try to access information, the problems of finding the desired information and understanding the structure of the available information will only get worse. The advantages of access to highly connected information provided by hypermedia will be severely reduced by the difficulties produced by the complex networks of semantic connections within these systems.

This work will help people find a "specific address" and develop an overall understanding of the structure of the "information highway" by providing a method of reducing the complexity of the networks of semantic connections that populate it. As new hypermedia systems are constructed, the designers of these systems could use a combination of views and constraints to reduce their complexity. Thus, when users try to find information in a hypermedia system, they will be presented with a less complex network of semantic connections. These better organized networks will be easier to comprehend, so people can develop an overall understanding of the structure of the information which is available to them, and thus utilize it more productively and efficiently.

Bibliography

[Afrati 90], Foto Afrati and Constantinos Koutras, A Hypertext Model for Supporting Query Mechanisms, *Hypertext Concepts, Systems and Applications*, Ed. N. Streitz and A. Rizk and J. Andre, p.52-66, Cambridge University Press, 1990.

[Agrawal94], R. Agrawal, and L. G. DeMichiel, Type Derivation Using the Projection Operation, *Advances in Database Technology - EDBT '9*4, International Conference on Extending Database Technology, Cambridge, United Kingdom, March 1994, Springer-Verlag, 1994, pages 7-14.

[Alhajj93], Reda Alhajj and M. Erol Arkun, A Query Model for Object-Oriented Databases, *Ninth International Conference on Data Engineering*, April 19-23, Vienna, Austria, 1993, pages 163-72.

[Anwar93], E. Anwar, L. Maugis, and S. Chakravarthy, A New Perspective on Rule Support for Object-Oriented Databases, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, Washington, DC, May 26-28, 1993, 22(2):99-108.

[Arnold94] Stephen C. Arnold, Leo Mark, and Sham Navathe, A Model of Hypermedia Systems for Administration of Semantic Connections, *Proceedings of the 5th International Hong Kong Computer Society Database Workshop, Next Generation Database Systems,* Hong Kong Computer Society, Kowloon Shangri-La, Hong Kong, pages 155-66, Feb. 26 1994.

[Berge73], Claude Berge, Graphs and Hypergraphs, North-Holland Publishing Company, New York, 1973.

[Bertino92], E. Bertino, M. Negri, G. Pelagatti, and L. Sbattella, Object-Oriented Query Languages: The Notion and the Issues, *Knowledge and Data Engineering*, June 1992, 4(3):223-237.

[Blakeley86] Jose A. Blakeley, Neil Coburn, and Per-Ake Larson, Updating Derived Relations,: Detecting Irrelevant and Autonomously Computable Updates. Technical report, CS-86-17, University of Waterloo, Computer Science Department, May 1986.

[Blakeley89] Jose A. Blakeley, Neil Coburn, and Per-Ake Larson, Updating Derived Relations,: Detecting Irrelevant and Autonomously Computable Updates. *ACM Transactions on Database Systems*, 14(3):369-400, September 1989.

[Borgida89], Alexander Borgida, Ronald J. Brachman, Deborah L. McGuinness and Lori Alperin Resnick, CLASSIC: A Structural Data Model for Objects, A*CM SIGMOD International Conference on Management of Data*, Portland Oregon, May-June, 1989.

[Borning81] Alan Borning, The Programming Language Aspects of ThingLab, a Constraint-Oriented Simulation Laboratory, *ACM Transactions on Programming Languages and Systems*, 3(4):353-387, October 1981

[Borning87], Alan Borning and Robert Duisberg and Bjorn Freeman-Benson and Axel Kramer and Michael Woolf, Constraint Hierarchies, *OOPSLA '87 Proceedings*, pages 48-60, October, 1987

[Bra92], Paul De Bra, Geert-Jan Houben, and Yoram Kornatzky, An Extensible Data Model for Hyperdocuments, Echt '92, *The Second ACM Conference on Hypertext*, Milano, Italy, November 30-December 4, 1992, pages 222-231.

[Bush94], V. Bush, "as we may think", *Atlantic Monthly*, July 1945, p101-108.

[Campbell 88], Brad Campbell and Joseph M. Goodman, HAM: A General Purpose Hypertext Abstract Machine, *Communications of the ACM*, v.31, n.7, p.856-861, July, 1988.

[Catarci93], Tiziana Catarci, Guiseppe Santucci, and Michele Angolaccio, Fundamental Graphical Primitives for Visual Query Languages, *Information Systems*, 18 (2), March 1993, pages 75-98.

[Catlin91], Karen Smith Catlin, L. Nancy Garrett and Julie A Launhardt, Hypermedia Templates: An Author's Tool, *The Third ACM Conference on Hypertext*, San Antonio, Texas, December 15-18, 1991, pages 147-160.

[Cervesato92], I. Cervesato and C. F. Eick, Specification and Enforcement of Dynamic Consistency Constraints, *Information and Knowledge Management CIKM-92*, Baltimore, MD, U. S. A., November 8-11, 1992, pages 193-99.

[Cochrane92], R. J. Cochrane, Advisor Leo Mark, Issues in Integrating Active Rules into Database Systems, Ph.D. Thesis Report, University of Maryland, College Park, 1992.

[Conklin88], Jeff Conklin, *Computer-Supported Cooperation, A Book of Readings*, Hypertext: An Introduction and Survey, p.423-475, Morgan Kaufmann, 1988.

[Consens93], Mariano P. Consens and Alberto O. Mendelzon, Hy+: A Hypergraph-based Query and Visualization System, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, Washington, DC, May 26-28, 1993, 22(2):511-522.

[Delcambre89], L. M. L. Delcambre and J. N. Etheredge, The Relational Production Language: A Production Language for Relational Databases, In L. Kerschberg, editor, *Expert Database Systems - Proc. From the Second Int. Conf.*, pages 333-351, Redwood City, California, 1989. Benjamin/Cummings.

[Delcambre 91], Lois M. L. Delcambre and Billy B. L. Lim and Susan D. Urban, Object-Centered Constraints, *Proceeding of the Seventh International Conference on Data Engineering*, p.368-377, April 1991.

[Diaz91], Oscar Diaz, Norman Paton and Peter Gray, Rule Management in Object Oriented Databases: A Uniform Approach, *17th International Conference on Very Large Databases,* September 3-6 1991, Barcelona (Catalonia, Spain), pages 317-326.

[Eick93], Christoph F. Eick and Paul Werstein, Rule-Based Consistency Enforcement for Knowledge-Based Systems, *Knowledge and Data Engineering*, February 1993, 5(1):52-64.

[Fisher92], Gene L. Fisher and Dale E. Busse, Adding Rule-Based Reasoning to a Demonstrational Interface Builder, *Proceedings of the ACM Symposium on User Interface Software and Technology*, ACM Press, Monterey, California, November 15-18, 1992, 89-98.

[Freeman89], Freeman, A Module Mechanism for Constraints in Smalltalk, *SIGPAN Notices*, v.24, n.10, p.389-396, Oct. 1989.

[Freeman90], Bjorn N. Freeman-Benson and John Maloney and Alan Borning, An Incremental Constraint Solver, *Comunications of the ACM*, 33,(1): 54-63, January,1990

[Formica92], A. Formica and M. Missikoff, Adding Integrity Constraints to Object-Oriented Databases, *Information and Knowledge Management CIKM-92*, Baltimore, MD, U. S. A., November 8-11, 1992, pages 593-601.

[Garcia94], C. Garcia, M. Celma, L. Mota, and H. Decker, Comparing and Synthesizing Integrity Checking Methods for Deductive Databases, *Tenth International Conference on Data Engineering*, February 14-18, Houston, Texas1994, pages 214-222.

[Garg88], Pankaj K. Garg, Abstraction Mechanisms in Hypertext, *Communications of the ACM*, v.31, n.7, p.862-870, July 1988.

[Gertz93] Michael Gertz and Udo W. Lipeck, Deriving Integrity Maintaining Triggers from Graphs, *Ninth International Conference on Data Engineering*, April 19-23, Vienna, Austria, 1993, pages 22-29.

[Gleicher93], A Graphical Toolkit Based on Differential Constraints, *Proceedings of the ACM Symposium on User Interface Software and Technology*, ACM Press, Atlanta, Georgia, November 3-5, 1993.

[Gronbaek93], Kaj Gronbaek, Jens A. Hem, and Ole L. Madsen, Designing Dexter-Based Cooperative Hypermedia System, *Proceedings of the Fifth ACM Conference on Hypertext*, Seattle, Washington, November 14-18, 1993, pages25-39.

[Gou91], M. Guo, S. Y W. Su, and H. Lam, An Association Algebra for Processing Object-Oriented Databases, *Seventh International Conference on Data Engineering*, April 8-12, Kobe, Japan, 1991, pages 23-33.

[Gronbaek94.1], Kaj Gronbaek and Randall H. Trigg, Design Issues for a Dexter-Based Hypermedia System, *Comunications of the ACM*, ACM Press, 37(2):30-39, Feb. 1994.

[Gronbaek94.2], Kaj Gronbaek and Hens A. Hem and Ole L. Madsen, and Lennert Sloth, Cooperative Hypermedia Systems: A Dexter-Based Architecture, *Comunications of the ACM*, ACM Press, 37(2):64-75, Feb. 1994.

[Halasz89], Frank Halasz and Mayer Schwartz, The Dexter Hypertext Reference Model, *Proc. of the NIST Hypertext Standardization Workshop*, 1990

[Halasz93], Discussion with Frank Halasz on May 27, 1993, in the GVU Conference room of the College of Computing at Georgia Tech.

[Halasz94] Frank Halasz and Mayer Schwartz, The Dexter Reference Model, *Comunications of the ACM*, ACM Press, 37(2):26-29, Feb. 1994.

[Han91], J. Han, Constraint-Based Reasoning in Deductive Databases, *Seventh International Conference on Data Engineering*, April 8-12, Kobe, Japan, 1991, pages 257-266.

[Hanson89], E. N. Hanson, An Initial Report on the Design of Ariel: A DBMS with an integrated production rule system. In *SIGMOD Record, Special Issue on Rule Management and Processing in Expert Database Systems*, 1989, Pages 12-19.

[Hardman94], Lynda Hardman, Dick C.A. Bulterman, and Guido Van Rossum, The Amsterdam Hypermedia Model: Adding Time and Context to the Dexter Model, *Comunications of the ACM*, ACM Press, 37(2):50-64, Feb. 1994.

[Hill93], Ralph D. Hill, The Rendezvous Constraint Maintenance System, *Proceedings of the ACM Symposium on User Inter face Software and Technology*, November 11-13, 1991, pages 225-234.

[Hudson91], Scott H. Hudson and Andrey K. Yeatts, Smoothly Integrating Rule-Based Techniques Into A Direct Manipulation Interface Builder, *Proceedings of the ACM Symposium on User Inter face Software and Technology*, November 11-13, 1991, pages 145-154.

[Hudson93], Scott E. Hudson, A System for Efficient and Flexible One-Way Constraint Evaluation in C++, Tech. Report GIT-GVU-93-15, Georgia Institute of Technology, 1993.

[Ishikawa93], Hiroshi Ishikawa and Kazumi Kubota, An Active Object-Oriented Database: A Multi-Paradigm Approach to Constraint Management, *19th International Conference on Very Large Databases*, August 24-27 1993, Dublin, Ireland, pages 467-479.

[Jagadish92], H. V. Jagadish and Xiaolei Qian, Integrity Maintenance in an Object-Oriented Database, *18th International Conference on Very Large Databases*, August 23-27 1992, Vancouver, Canada, pages 469-481.

[Karadimce93], Anton P. Karadimce and Susan D. Urban, A Framework for Declarative Updates and Constraint Maintenance in Object-Oriented Databases, *Ninth International Conference on Data Engineering*, April 19-23, Vienna, Austria, 1993, pages 391-398.

[Kim 92], Won Kim, *Introduction to Object-Oriented Databases*, The MIT Press, 1992.

[Kurlander93], David Kurlander and Steven Feiner, Inferring Constraints form Multiple Snapshots, *ACM Transaction on Graphics*, 12(4):277-304, Oct. 1993.

[Lange 90], Danny B. Lange, A Formal Model for Hypertext, *Proceedings of the Hypertext Standardization Workshop*, 1990

[Leggett94], Join J. Leffett and John L. Schnase, Viewing Dexter with Open Eyes, *Comunications of the ACM*, ACM Press, 37(2):76-86, Feb. 1994.

[Lenat90], Douglas B. Lenaat and R. V. Guha, *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project*, Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1990

[Liu93], Ling Liu, A Recursive Object Algebra Based on Aggregation Abstraction for Manipulating Complex Objects, *Data & Knowledge Engineering,* North-Holland, 11:21-60, 1993.

[Mark91], L. Mark, N. Roussopoulos, and R. Cochrane, Updated Dependencies in the Relational Model. Technical Report SRC-TR91-94, Department of COmputer Science, University of Maryland, College Park, Maryland, October 1991.

[Marshall91], Catherine C. Marshall, Frank G. Halasz, Russell A. Rogers, and William C. Janssen Jr., Aquanet: a Hypertext Tool to Hold Your Knowledge in Place, *The Third ACM Conference on Hypertext*, San Antonio, Texas, December 15-18, 1991, pages 261-275.

[Marshall92], Catherine C. Marshall and Russel A. Rogers, Two Years before the Mist: Experiences with Aquanet, *Echt '92, The Second ACM Conference on Hypertext*, Milano, Italy, November 30-December 4, 1992, pages 53-62.

[Martin92], H. Martin, M. Adiba, B. Defuce, Consistency Checking in Object-Oriented Databases: a Behavioral Approach, *Information and Knowledge Management CIKM-92*, Baltimore, MD, U. S. A., November 8-11, 1992, pages 326-334.

[McCarthy89], D. R. McCarthy and U. Dayal, The Architecture of an Active Database Management system, In *SIGMOD89*, pages 215-224.

[Mukherjea94], Sougata Mukherjea, "Visualizing the Information Space of Hypermedia Systems", Ph.D. Proposal, College of Computing, Georgia Institute of Technology, 1994.

[Nielsen 90], Jakob Nielsen, *Hypertext and Hypermedia*, Academic Press, Inc.1990.

[Parunak91], H Van Dyke Parunak, Don't Link Me In: Set Based Hypermedia for Taxonomic Reasoning, *The Third ACM Conference on Hypertext*, San Antonio, Texas, December 15-18, 1991, pages 233-242.

[Roussopolous91], N. Roussopoulos, The Incremental Access Method of ViewCache: Concept and Cost Analysis, *ACM Transactions on Database Systems*, 16(3), 1991.

[Slack93], James M. Slack and Elizabeth A. Unger, Integrity in Object-Oriented Database Systems, *Computers & Security*, Elsevier Science Publishers, Ltd.,12:389-404, 1993.

[Slack93], James M. Slack and Elizabeth A. Unger, Integrity in Object-Oriented Database Systems, *Computers & Security*, Elsevier Science Publishers, Ltd.,12:389-404, 1993.

[Stefik81.1], Mark Stefik, Planning with Constraints (MOLGEN: Part 1), *Artificial Intelligence*, 1981, 16, 111-140.

[Stefik81.2], Mark Stefik, Planning and Meta-Planning (MOLGEN: Part 2), *Artificial Intelligence*, 16:141-169,1981

[Stonebraker88], M. Stonebraker, M. Hearst, and S Potamianos, The POSTGRES rule manager. *IEEE Trans. on Software Engineering*, 14(7):897-907, July 1988.

[Stonebraker89], M. Stonebraker, M. Hearst, and S Potamianos. A Commentary on the POSTGRES Rule System. In *SIGMOD Record, Special Issue on Rule Management and Processing in Expert Database Systems*, 1989, pages 5-11.

[Stotts90], P. David Stotts and Richard Furuta, Browsing Parallel Process Networks, *Journal of Parallel and Distributed Computing*, v9, p.224-235, 1990.

[Su93], Stanley Y. W. Su, Mingsen Guo, and Herman Lam, Association Algebra: A Mathematical Foundation for Object-Oriented Databases, *Knowledge and Data Engineering*, October 1993, 5(5):775-798.

[Tompa88], Frank Wm. Tompa and Jose A. Blakeley, Maintaining Materialized Views Without Accessing Base Data, *Information Systems*, 13(4):393-406, 1988.

[Widom90], J. Widom and S. J. Finkelstein, Set-Oriented Production Rules in Relational Database systems. In *SIGMOD90*, pages 259-270.

[Yoon92], J. P. Yoon and L. Kerschberg, A Framework for Constraint Management in Object-Oriented Databases, *Information and Knowledge Management CIKM-92*, Baltimore, MD, U. S. A., November 8-11, 1992, pages 292-299.

[Z], J. M. Spivey, *The Z Notation*, Prentice-Hall International, 1989

[Zanden91], Brad Vander Zanden, Brad A. Myers, Dario Giuse and Pedro Szekely, The Importance of Pointer Variables in Constraint Models, *Proceedings of the ACM Symposium on User Inter face Software and Technology*, November 11-13, 1991, pages 155-164.

[Zheng92], Yi Zheng and Man-Chi Pong, Using Statecharts to Model Hypertext, *Echt '92, The Second ACM Conference on Hypertext*, Milano, Italy, November 30-December 4, 1992, pages242-250.

[Zhou90], Yuli Zhou and Meichum Hsu, A Theory for Rule Triggering Systems, *Advances in Database Technology - EDBT '90*, International Conference on Extending Database Technology, Venice, Italy, March 1990, Springer-Verlag, 1990, pages 407-21.