UNIVERSITY OF OSLO
Department of Informatics

# Knowledge Engineering using The Weak-Transitive-Closure algorithm – A case study of The University of Oslo's IT system.

Ephrem Tewolde

Network and System Administration

Oslo University College

May 24, 2011

# Knowledge Engineering using The Weak-Transitive-Closure algorithm – A case study of The University of Oslo's IT system.

Ephrem Tewolde

Network and System Administration
Oslo University College

May 24, 2011

**Abstract**

The complexity of human-computer systems nowadays requires the aid of manageable and simplified, machine-readable representation of those systems. In addition, the need to accommodate appropriate and mandatory changes to legacy systems is an inherent challenge in system administration tasks. Thus, a state-of-the-art knowledge management system must be capable of balancing both these needs.

A reasoning method that works via a simplified calculus of facts and rules was suggested by professors Alva L. Couch of Tufts University and Mark Burgess of Oslo University College. The name 'weak-transitive-closure algorithm' or the WTC algorithm for short, is adopted here for this method owing to the key technique applied in the system. This method visualizes complex systems as directed graphs, and applies graph theory techniques for inference of causal dependence between components of the system.

This thesis is an investigative study of the application of the WTC algorithm, using University of Oslo's IT system as the application domain. A topic map representation already exists for this system. The application of the WTC algorithm requires a thorough study of the problem domain, so as to construct a knowledge base and a set of rules as the embodiment of the abstractions, modeling and representation of the system. These sets of facts and rules are then input to a prototype engine, which uses them for supplying answers to queries about the system. In contrast, the topic map representation of the system is as undirected graph, comprised of discrete components called topics connected by the edges known as associations. Evaluation of the two representations at various levels is done thoroughly, since both enforce some constraints on how to model and represent the system. In the process, the new opportunities of the weak-transitive-closure algorithm in supplementing and/or replacing the topic map representation are investigated.

We demonstrate that the WTC algorithm has the advantage of discovering connections with specific properties, by generating the paths automatically, which is more optimized for troubleshooting. In addition, the WTC algorithm's presentation is more suitable for learning about legacy systems.

# Acknowledgments

I sincerely thank God the Almighty, for the gift of life and for being gracious unto me.

I am specially indebted to prof. Alva L. Couch, who gave me insights into many ideas, answered many of my questions and helped me with corrections in many parts of the thesis.

My gratitude goes to my advisor, Hårek Haugerud, for being helpful in many ways, and for commenting on the writings of the thesis.

I thank prof. Mark Burgess for his encouragement at the start of the thesis, and for his useful feedback.

Special thanks to Jarle Bjørgeengen of USIT, who first proposed the project idea, for facilitating contact and access with USIT and for helpful co-operations, including providing system knowledge data from USIT.

Thanks to Are Gulbrandsen of the XML group at USIT for introducing me to the Houdini topic map.

I would like to thank USIT for allowing me to work with the infrastructure's data in this project.

I would like to thank UiO/HiO for giving me the opportunity to study, and for being such a wonderful place of learning.

I would like to thank the ever cooperative and enthusiastic instructors at HiO's Master in Network and System Administration department for providing diverse knowledge in Network and System Administration.

My special thanks go to Addisu Tesfaye and Eskedar Kefialew, for encouraging me to apply for the program and for being supportive friends all along.

Last but not least, I extend my affectionate gratitude to my wife Yetnayet, for her support, patience and love.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

With the increasing complexity of human-computer systems, the task of system administration is getting more difficult by each year and technological advance. One of the key challenges facing system administration is knowledge management , i.e., how the facts required for managing a system are collected, updated, and maintained.

For a start, even stand-alone computers are by themselves complex. When several of them are in a network, infrastructure and inter-dependence complexity follows. All of this complex infrastructure is intended for human use, i.e., the objective is human-computer interaction, creating an environment of constant change and unpredictability. Added to all of these factors, there is pressure to adapt and conform to the ever changing innovation in the field and the changing needs of users.

Thus, a large amount of knowledge of complex nature is involved in human-computer systems. In fact, human-computer systems are rightfully called knowledge-based systems, owing to their reliance on a large amount of procedural or factual expertise to carry out their function[34].

First, there is the knowledge of the system and its complex structure and configuration, including the intentions of the 'why the structure is made so in the first place.' Second, there is expert know-how and experience of the professionals in the form of best practices, proven procedures, documentation of changes and 'cook book' solutions. Third, there can be many experts with specializations in some aspect, so that knowledge includes who knows what. This 'who knows what' knowledge is important for seamless cooperation.

As a matter of expediency – due to shortness of time – little of this knowledge is recorded in a formal and efficiently re-usable manner. Most resides in the minds of the experts and scattered in numerous other documents. This scattering results in loss of business continuity when experts leave an organization. Even when they are there, the lack of a formal, organized body of knowledge exposes for uncertainty and avoidable repetitions of solution finding to old problems, which causes loss of man hours [18]. This is another

reason why knowledge management is viewed as a key challenge of system administration[29]. In addition, when the inevitable change is required in complex systems, making the change without catastrophe requires a state-of-the-art knowledge management.

Part of the solution to the problem of knowledge management is representing systems knowledge, representing 'what there is in the system', in machine-readable knowledge bases. Knowledge bases include data and configuration information which try to capture, in some form, a useful representation of a domain for some purpose. Thus, developing machine-readable knowledge bases and investigating available representation methods is a crucial piece of research area in system administration.

## 1.1 Motivation

The main motivation of this project is the tenet that *the way we view, model and represent human-computer networks knowledge bases* has a profound effect on our understanding of the complex system. Topic Maps, an ISO standard technology for describing knowledge structures, can be used and are used for knowledge management of human-computer systems. A topic map visualizes and presents a knowledge domain as a set of discrete components linked to each other. Another seemingly distinct approach is the WTC algorithm. The WTC algorithm is an automated reasoning program which views the knowledge domain as comprised of entities related to each other by causal chains. This thesis investigates the new opportunities that the WTC algorithm might provide, in supplementing and/or replacing a topic maps representation.

In applying either Topic Maps or the WTC algorithm, the whole orchestration from the representation of the knowledge domain to the front-end presentation can roughly be divided into two main components.

First, there is the knowledge base, which contains the modeling, representation and abstraction of the system. This part requires a thorough study of the problem domain, and is specifically dependent on the system being represented.

Second, there is a software analysis component that uses the knowledge base and gives output in a desired format to the end user. This part has a generic name 'engine', and is generally similar for one kind of representation, except minor customizations of the original programs or tools. In topic maps, this second part is called a topic map engine, and in the WTC algorithm it is called the prototype engine. This leads us to the fundamental question: How is the knowledge base represented in Topic Maps, and, in the WTC algorithm? Besides, the components involved in the processing and presentation of the knowledge base, in both Topic Maps and in the WTC algorithm, and the customizations required on the software analysis component of each, need to be addressed.

## 1.2 Problem Statement

The main purpose of this thesis is the application of the WTC algorithm to the IT infrastructure at the University of Oslo as a field application and case study. Topic Maps has already been applied to model it. The XML-group at the University of Oslo's Center for Information Technology Services (USIT) has developed a Topic Map based knowledge base, called the Houdini topic map, for systems operation, administration and maintenance documentation, which is used by the Information and Communication Technology (ICT) control center, codenamed Houston[24]. Houdini is a web application, and serves as a topic map portal for service documentation. Still, there are a number of reasons to try the WTC algorithm in this problem domain:

- The WTC algorithm is newer, specifically designed with human-computer systems in mind and seems specially suited for them. Topic Maps are richer, but arise from distinct traditions in library science and general information indexing and search.
- Topic Maps focus on links between components of a knowledge domain, while the WTC algorithm focuses on chains of reasoning and/or causal chains. Chains of reasoning and causal chains, as opposed to mere linkage between components, is more useful in administration and management of human-computer networks.

### The problem statement of this research is:

To develop a new representation using selected use cases of the University of Oslo's IT infrastructure knowledge base via the WTC algorithm, taking this complex system as the new problem domain to be thoroughly studied and modeled, and investigate what benefits or improvements this method will provide in either supplementing or replacing the Houdini topic map.

These two representations, the Houdini topic maps and the WTC algorithm representation, might differ in many stages of the knowledge representation of the system. One difference may be at the very basic level, on how to conceptualize the system and on the choice of how to describe relationships between the different entities comprising the system. This means in general terms, the difference within the knowledge base level. But, they also differ at a higher level, even when sharing the same conceptualization of a system. The two representations differ in semantic richness and in consequence, in discovery abilities. In order to address the differences in both levels, two approaches will be used to implement the WTC algorithm representation. For convenience and ease of identification, we name the approaches as the 'topic-map approach' and the 'system-description approach'.

1. The topic-map approach

   In this approach, the *existing topic map knowledge base is transformed into a suitable knowledge base for the WTC algorithm.* The knowledge base of

the topic map will be transformed to the WTC representation as-is, in its entirety. Thus, in this approach, both share the same conceptualization of the system. This approach will help in investigating the benefits of the WTC algorithm even when sharing the same kind of conceptualization of the system as the topic map.

2. The system-description approach

   In this approach, *an available detail of systems knowledge is used to create a new knowledge base for the WTC algorithm.* This will help to investigate the benefits of the WTC algorithm at the very basic level. How will WTC conceptualize the knowledge domain? What choice of relationships will be best for the WTC algorithm? and what are the benefits of those? Such questions will be addressed in this second approach.

## 1.3 Thesis Structure

This thesis is structured as follows.

Chapter 2 describes Background and Literature, which has three main parts. In the first part, a selected review of the most relevant ideas pertaining to this work are presented. In the second part, the ideas are elaborated in a practical example application. Then, in the third part, the Houdini topic map is explained in a summarized way.

Chapter 3 starts with an introduction of the model used in the WTC algorithm. Then, the WTC algorithm itself is explained in detail, followed by an elaborate example. Finally, the approach used in this work is outlined in detail.

In Chapter 4, results of the use-cases are presented for both approaches, with explanations.

Discussion of the results is done in detail in chapter 5, followed by the conclusion in chapter 6.

Finally, the bibliography has the list of references, and is preceded by the appendix.

# Chapter 2

# Background and literature

## 2.1 Introduction to Knowledge

What is knowledge?

This question is broad and generally there is no single answer to it , but the approach taken here is to avoid the philosophical debate and refrain from trying to give a general definition of what knowledge is. Rather, an attempt is made to lay a working definition from a pragmatic, engineering discipline approach. By this is meant the idea that is useful for solving the practical problem at hand.

First of all, useful knowledge has to be acquired somehow, and then summarized and stored[34]. Secondly, though there is a relationship between knowledge and information, information only is not knowledge. Thus, a working definition is that, in knowledge, one finds the functional associations between items of information and/or data expressed explicitly[3].

Knowledge is more than a static configuration of facts. Having a knowledge of something means the ability to form a mental model that accurately represents the thing as well as the actions that can be performed by it and on it. Therefore, knowledge is the configuration of ideas in a representation medium that is somewhat formalized and can be transferred[17]. The representation medium can be human mind, computer memory or a piece of paper.

If a system administrator has a good knowledge about the complex human-computer system that he or she is responsible for, that means he or she has a good mental model of what is there, of the different kinds of relations, and what can be done by it all and on it. In this there is the underlying assumption that knowledge is some aspect of something, and not everything that can be known, even about a domain of interest. Thus knowledge refers, in this context, *the knowledge about a domain of interest necessary and hopefully sufficient from the perspective of a certain objective or accomplishment.*

This paper consciously differentiates between the 'knowledge that there is and remains' and 'the dynamic knowledge that is not predictable or cannot be

assured beforehand'. The focus is on the former, which is synonymous with, for the purpose of this work, *system's knowledge*.

This leads us to the next question of how this can be represented into a digital system. The general answer to this problem is: *by developing knowledge based systems.*

Knowledge based systems are computer programs that contain a large amount of *knowledge base*, *rules* and *reasoning system for making inferences.* When we say that knowledge based systems contain large amounts of knowledge, we mean a large amount of detail about the configuration and facts of the domain in question.

Two things come to attention. One, if domain knowledge is encoded in a computer, and a knowledge based system is formed, this requires that the knowledge based system behave as some sort of a knowledgeable agent concerning that domain. It will be able to answer some questions, giving kinds of answers that normally require human expertise. Two, the encoding should also be understandable by a human, so that it can be used, verified and expanded. In order to develop such knowledge based systems, one needs to *model the knowledge* and also be able to *represent it.*

## 2.2 Knowledge Engineering

Knowledge Engineering is an engineering discipline that involves integrating knowledge into computer systems[2]. Integration of knowledge into computer systems is primarily done in order to solve complex problems normally requiring a high level of human expertise. Thus, Knowledge Engineering is a field of engineering that develops knowledge based systems. Put another way, knowledge engineering applies logic , conceptualization and way of expression and representation to build computable models of some domain for some purpose. It analyzes knowledge about some subject and transforms it to a computable form for some purpose[36].

### 2.2.1 Principles of Knowledge Engineering

Knowledge engineers have developed a number of principles, methods and tools that have considerably improved the process of knowledge acquisition and ordering. Some of the key principles are summarized as follows[7]:

- There are different types of knowledge, and the proper approach and technique should be used for the knowledge required.

  For example, in human-computer systems, we have systems knowledge and expert knowledge.

- There are different ways of representing knowledge.

  For example, we consider two types in this thesis, Topic Maps and the input to the WTC algorithm.

- There are different ways of using knowledge, so that the acquisition process can be guided by the project aims, meaning it will be goal-oriented.

  Some of the goals can be to document searching, troubleshooting, or system administrative tasks.

### 2.2.2 Knowledge Representation

> The term knowledge representation was originally used in artificial intelligence (AI) to refer to the encoding of knowledge that an intelligent program would seem to require in order to plan, observe or draw conclusions. It is now understood more broadly to refer to *any organized body of general knowledge*, including large-scale repositories of information intended largely for human use. ...knowledge representation (KR) is usually taken to refer to the representation of general knowledge that can support some nontrivial reasoning. To represent knowledge, one must choose a suitable collection of concepts, a notation and a system of inference rules or processes that use the notation[19].

## 2.3 Ontology

The term "ontology" is used in many disciplines with slightly different but related meanings. Its origin is philosophy, and ontology is defined as *the study of being, or of 'what exists'*. Ontology's most general definition is as the study of existence of all things whether abstract or concrete that make up the world. It provides the vocabulary to describe the things that exist[36].

When applied to different disciplines, the difference comes, partly, in the definition of 'the world' and 'what exists'. For philosophers and metaphysicists, 'the world' is the universe, and 'what exists' is everything. Usually, philosophers build their ontologies from the top down. They start with grand conceptions, seeking a general and sufficient way of explanation about everything in heaven and earth[36].

For the Artificial Intelligence community, where the term was first borrowed from philosophy and customized to their field[13], the world is the subject of study or a domain of interest and 'what exists' is that which can be represented. From this perspective, we have [10]

> A body of formally represented knowledge is based on a conceptualization: the objects, concepts, and other entities that are

> presumed to exist in some area of interest and the relationships that hold them. Every knowledge base, knowledge-based system, or knowledge-level agent is committed, explicitly or implicitly, to some conceptualization, which is an abstract, simplified view of the world that we wish to represent for some purpose. *An ontology is an explicit specification of a conceptualization.*

In fact, [36] has also a more general definition which confirms this:

> The subject of ontology is the study of the categories of things that exist or may exist in some domain. The product of such a study, called an ontology, is a catalog of the types of things that are assumed to exist in a domain of interest D from the perspective of a person who uses a language L for the purpose of talking about D. The types in the ontology represent the predicates, word senses, or concept and relation types of the language L when used to discuss topics in the domain D.

We need ontology in knowledge representation for many benefits. First, ontological analysis clarifies the structure of knowledge since it involves the conceptualizations that underlie knowledge. Given a domain, what forms the heart of any system of knowledge representation for that domain is in its ontology. In other words, there cannot be a vocabulary for representing knowledge without ontologies. Second, ontologies enable knowledge sharing[14].

Thus, [10] was later broadened to an inclusive definition, that ontology is *a formal, explicit specification of a shared conceptualization*[11], to accommodate this 'knowledge sharing concept'.

In summary, if we keep in mind the distinction between 'what exists' and 'the domain', and the different contexts the terms are used, ontology

- Is used for systematic account of 'what exists' in 'the domain'. This is achieved through conceptualization; defining the types and subtypes of concepts and the relationships necessary to describe the perceived things in the domain.
- Provides vocabulary for describing 'what exists' in 'the domain'.
- Enables knowledge sharing through formalization of expression.

## 2.4 Topic Maps

### 2.4.1 General Background

Topic Maps[1] describes a paradigm centered around efficiently locating and searching for digital information. It is also used for knowledge management. It is a data model by which information is made more manageable and findable[6]. The approach that Topic Maps provide have advanced techniques of linking and addressing, comprising the best from several worlds like traditional indexing, library science and knowledge representation[22].

According to the ISO standard definition, Topic Maps can be used[31]:

- To qualify the content and/or data contained in information objects as topics to enable navigational tools such as indexes, cross-references, citation systems, or glossaries.
- To link topics together in such a way as to enable navigation between them. This capability can be used for virtual document assembly, and for creating thesaurus-like interfaces to corpora, knowledge bases, etc.
- To filter an information set to create views adapted to specific users or purposes. For example, such filtering can aid in the management of multilingual documents, management of access modes depending on security criteria, delivery of partial views depending on user profiles and/ or knowledge domains, etc.
- To structure unstructured information objects, or to facilitate the transformation of topic-oriented user interfaces that provide the effect of merging unstructured information bases with structured ones. The overlay mechanism of topic maps can be considered as a kind of external markup mechanism, in the sense that an arbitrary structure is imposed on the information without altering its original form.

As an approach, Topic Maps conceptualize the world out there, a domain of some sort, as a collection of 'atomic abouts'. These 'atomic abouts' are called 'subjects', which are at the heart of the Topic Maps paradigm [21].

For any information source, the topic map tries to manage the meaning of the information conveyed, rather than the information itself. This is done by taking the key concepts in the information source and relating them together[23]. This is demonstrated in figure 2.1, in the context of the Houdini topic map.

[1]To differentiate between the standard and the applications, the established convention is "using initial capitals ('Topic Maps') when referring to the standard itself or the technology in general, and lower case ('topic maps') when referring to the document- like artifacts created through the application of that technology"[21].

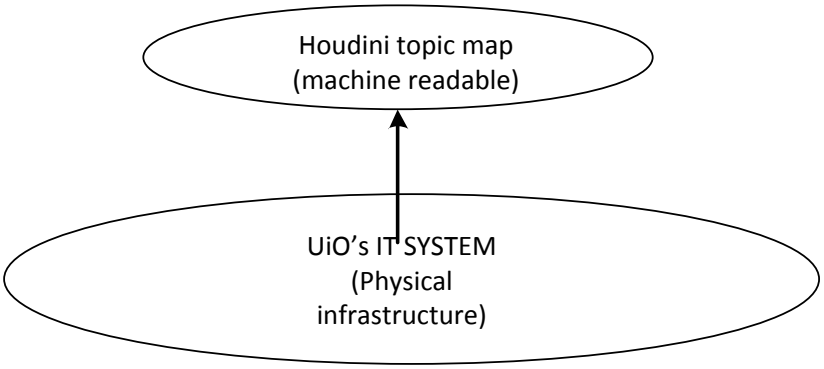


Figure 2.1: Topic Maps create an index of the information outside the information source.

Each subject can be conceptualized as having a type, which is a generic name which 'bears' instances. Each instance can then be represented by a symbol called topic. Topics can be components, called 'topics' or, they can be the concepts of relations among them. Any concept of the interaction between components, itself a type having topics, has a special name, 'association'. Where and how in the domain the topics occur is their 'occurrence'. This is called the TAO of topic maps[22].

In general, the structural information conveyed by topic maps includes[31]:

- groupings of addressable information objects around topics (occurrences), and
- relationships between topics (associations).

Knowledge representation using Topic Maps has a formal model, ISO Standard Topic Maps. In fact, there exist a family of ISO Topic Maps standards currently[31][32][33]. The complexity of digital information, including how to attach meaning and context to tokens, has led to more other concepts than the core topic map concepts, i.e., the TAO.

### 2.4.2 Main concepts in Topic Maps

- Subject
  A subject is "anything whatsoever, regardless of whether it exists or has any other specific characteristics, about which anything whatsoever may be asserted by any means whatsoever"[31].

  The subjects of a topic map are thus the 'atomic abouts' , in the view of the author of the topic map, that compose the domain in question. By articulating the subjects, the author claims , in a sense, that what is

embodied in the domain can be thought of as the set of those discrete subjects; and, the meaning of the subjects mean exactly the assertions given about each of them.

- Topic
  While subjects are conceptual, they are represented by topics. A topic is a "symbol used within a topic map to represent one, and only one, subject, in order to allow statements to be made about the subject"[31].

  Thus, a topic serves as a surrogate or symbolic representation of a subject. Topics have three characteristics: name, occurrence and role.

  A topic may have zero or more names, that can be given to it. During association with other topics, a topic will have a role in the association. Occurrence is a characteristics of a topic in relation to an information source that is about the topic.

- Collocation objective and Identity of subjects
  Since a domain of interest is conceptualized as comprised of atomic subjects and each subject is represented by one and only one topic , each subject can be assigned an 'identity' to uniquely identify it. By this, all about a subject can be reached from a single point, a concept known as the collocation objective. This can be applied even across domains, so that subjects which represent the same thing will be universally identified. Using identifiers, it will be possible to know when two or more topics across domains represent the same subject. This is useful for merging of different topic maps.

- Associations
  Associations express relationships among topics. They correspond to 'see also' listings in traditional back-of-book indexes. In describing relationships, the nature of the relationship can be made explicit by "association type", while the "association role" is used to express the nature of the topic's involvement in the relationship. As such, the order of the topics in a relationship is not important in topic maps. In other words, a relationships "what kind is it?" is explained by the "association type", while "as what is a topic involved in this relationship?" is expressed by the "association role" of a topic. Since assertions can be made about them, association types and association roles are themselves subjects and are represented by topics.

- Types in Topic Maps
  A type is a specification for a set or collection of entities that exist or may exist in some domain of discourse[36]. Topics can be categorized according to their kind. In a topic map, any given topic is an instance of zero or more topic types. In other words, the relationship between a topic and its type is a typical class-instance relationship[22].

  Thus, we understand that topics are instances of topic types, associations are instances of association types and association roles are instances of association role types.

- Occurrence
  A special kind of relationship between a subject's topic and an information source is represented by "occurrence" in Topic Maps. A subject X occurs in information source Y is another way of stating that subjects X and Y are related in the sense that the information source Y is about or mentions about the subject X. The information source is usually represented by a URL. Any occurrence can be viewed as an instance of an "occurrence type", which is itself a topic.

- Scope and Reification
  The meaning of an expression is usually validated in a certain context. For example, the statement "3 is not divisible by 2" is true if the context is entirely about the set of natural numbers, but is generally not true. We always presume a certain context when we make statements, otherwise communication would be difficult.

  In topic maps, contextual validity is reinforced by stating "Scope". "Scope is expressed as a set of topics which qualify a statement (i.e., a name, occurrence, or association) and indicate the context in which the assertion represented by the statement may be considered valid. If no scope is explicitly specified, the scope is said to be 'unconstrained"'[21].

  A more general concept is reification, which tries to assert any other thing (including relations and the topic map itself) in the topic map. "Reification is making a topic represent the subject of another topic map construct in the same topic map"[31].

To understand the above concepts, let us consider an example of how an informal knowledge specification is represented into topic maps.

### 2.4.3 Knowledge Representation using Topic Maps

Consider the following informal statements, which explain a very tiny portion of the complex system at the University of Oslo's IT infrastructure. The system can be explained by several thousand such statements.

'A unix server provides login service to users. A switch is connected to it. The unix server is called login1.uio.no. The switch is called sw-248-191.uio.no. For the server to provide its service, the switch must also provide its service.'

Topic maps starts by extracting the concepts involved. In other words, we ask what the best way of thinking about the system is. Both the server and the switch are there for some purpose. The unix server provides login service. The switch is there for the server to provide its service. Thus, the concepts involved in the above statements are 'service providing by service providers' and 'one service-provider depending on another to function - a dependence relationship between the service-providers.' Thus, our subjects are 'service-providing' and 'strong dependence-a concept of the relationship between the objects'. These are then our 'atomic abouts'.

Next, we define the concepts, making the assertions about the subjects[27]:

- Service-giving : is a useful IT functionality provided by a hardware device or a software running on such a device.
- Strong dependence: is a relationship between service providers so that the dependent cannot provide service if the depended upon is not providing its service.

The next step is to model the domain as a representation of the concepts defined. This means that we can now view the above statements as two service-providers and their dependence relationship. We define the types in the domain, which will have instances. Those types are 'service-providing' and 'strong dependence'. By this steps, we have created the ontology of the domain.

**Creating the knowledge base**

Create the digital representation of the things involved. This is how they go into topic maps representation.

- represent the physical unix server by a token = 'login1.uio.no'
- represent the physical switch by a token = 'sw-246-191.uio.no'
- represent the concept of the relationship between the server and the switch, the strong dependence, by a token= 'strong dependence'

The types we defined help us to categorize topics according to their kind since every topic is considered an instance of zero or more topic types in any topic map[21]. The 'service-providing' topic type has two instances, the topics representing the unix server and the switch. The 'strong dependence' topic type has one instance, the topic representing the relationship concept between the server and the switch.

In Topic Maps, the tokens or symbols which represent things in the computer are called 'topics'. The topics can represent physical things, concepts of relationships etc. We have three topics up to now: two topics represent physical things while one represents a relationship concept or an association. The association type 'strong dependence' categorizes all such kind of relationships between topics in this topic map.

We now formulate the above into topic|association|topic format. (Note: the use of the "|" symbol between the tokens is done only as a matter of notation in these examples. This symbol is part of the syntax in the WTC algorithm, but not in Topic Maps.)

login1.uio.no|strong dependence|sw-248-191.uio.no

**Contextual validity in topic maps, scope:**

We see that the above relationship is true only as far as service providing is concerned. It does not mean that the unix server depends on the switch for everything. For example, if the power of the switch is down, the power of the unix server will not be down. The unix server will fail from the perspective of providing "login service" due to the failure of the switches power, but in other ways, the unix server is still there. To address such contextual validity, we use scope. (The sentence is written in two lines for space consideration, but it is one line of tokens.)

$$login1.uio.no_{\textit{as service provider}}|strongdependence_{\textit{as association}}$$
$$|sw-248-191.uio.no_{\textit{as service provider}}$$

**Disambiguating the relationship and making direction unnecessary, Using roles in topic maps:**

In topic maps, direction is made unnecessary. This is possible by adding the 'role' of a topic in an association. Thus, by adding the role of the topics in the above,we have (again, written in two lines for space consideration)

$$login1.uio.no_{\textit{as service provider, dependent}}|strongdependence_{\textit{as association}}$$
$$|sw-248-191.uio.no_{\textit{as service provider, depended upon by}}$$

This relationship is true whether we write 'login1.uio.no' first or 'sw-248-191.uio.no' first. This is how direction is made unnecessary in topic maps. This has the advantage that, login1.uio.no is reachable both from its token or from any other topic related to it.

The whole thing about the knowledge base in topic maps is the exhaustive recording of such relations of every topic in the domain. The complete topic map does this for all topics in the domain. This is stored in files of a standard format, text files in either XTM or LTM formats, and constitutes the knowledge base of the topic map.

The processing program, called the topic map engine, parses such knowledge base files and the front end application presents the result to end users. As an example, a screen shot taken from the web-applications front-end at Houdini, displayed in fig 2.2, shows how the topic login1.uio.no is presented by the topic map.

As can be seen in the figure, figure 2.2, this way of writing all relations of login1.uio.no to any other topic in the domain makes anything about 'login1.uio.no' reachable from a single click of the topic 'login1.uio.no'. These are called 'occurrences' of the topic 'login1.uio.no' in the domain. Since this topic is a symbol representing an actual physical server in the complex system, we have achieved a way of knowing everything relevant about the actual server in the domain. The complete topic map does this for all topics in the domain. By doing this to all topics in the domain, Topic Maps achieves the information

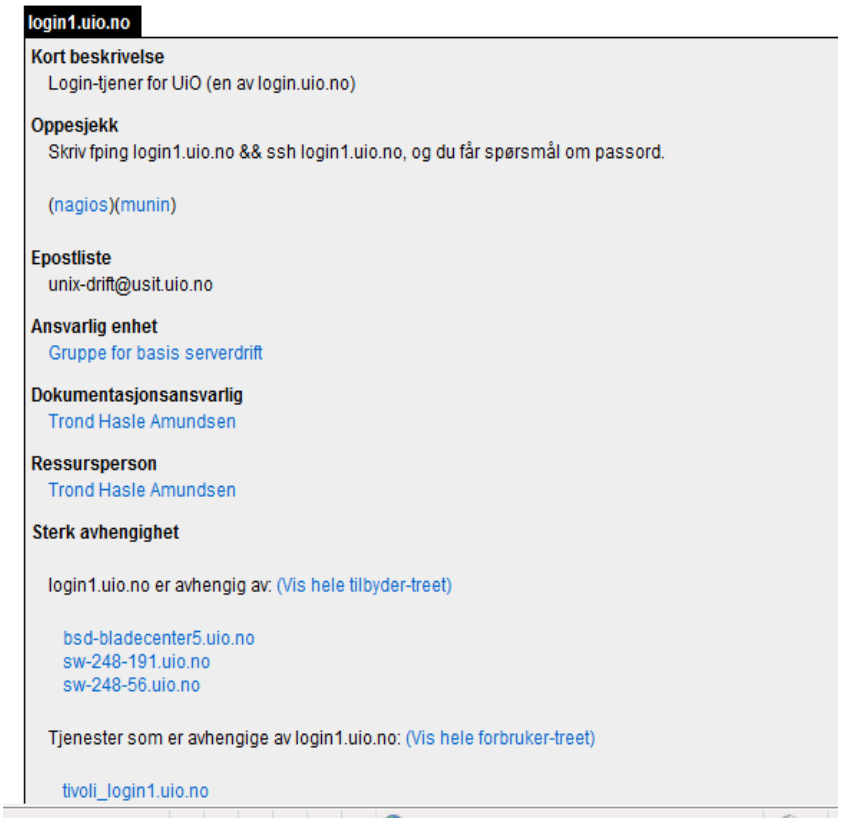


Figure 2.2: topic maps enables everything about login1.uio.no to be accessible from a single click.

find-ability and knowledge management of a domain possible. In fact, something that collects the key concepts in the domain and ties them all together is created[6].

## 2.5 Principles of Knowledge Representation

Knowledge Representation as a discipline has few widely accepted principles. Those widely accepted principles of knowledge representation, first outlined in[8], are explained below.

- The surrogate nature of knowledge representation.

  By being a substitute for the actual thing, a knowledge representation enables to have something to reason about and study consequences, than doing actions on the actual thing.

  In the topic maps representation above, the whole goal was to represent the physical things and their connection and relation in a digital form.

This enables many things like planning of downtime for maintenance and studying system dependencies, possible[27].

- A knowledge representation addresses the fundamental question of how to think about a domain, and the representation is committed to the fulfillment of this chosen conceptualization.

  Again in the example, the choice was to think the domain as 'service-givers and their dependence relationships.'

- In a knowledge representation, the underlying tenet is that the domain can be thought of as some kind of interaction between fundamental components. In other words, a knowledge representation assumes that the most useful aspects of a domain can be viewed as a form of relationship and dependence among key components of the domain. Therefore it is a fragmentary theory of intelligent reasoning.

  As the example shows, the existence of the server and the switch was defined not in entirety but in one aspect, in 'service-giving' aspect. Then the inference was made that the switch is there for the server's goal to be met.

- By a knowledge representation, we create a medium of efficient computation.

  Due to the specific conceptualization and the representation we made, it is *easier to see from the model* that the whole setup is there for a sustained login service to the unix server, and that the switch is there for the server's functioning.

- In a knowledge representation, we are actually putting forth a description of the domain we want to represent. Therefore, it is a medium of human expression.

## 2.6 The Houdini topic map

### 2.6.1 Background

The University of Oslo is one of the largest universities in northern Europe, with approximately 30000 students and more than 4600 employees. The University's Central IT Unit is called USIT. USIT has an Information and Communication Technology control center called Houston. Houston monitors and administers a wide range of ICT-service types, spanning from gateways to Digital Library services. ICT–Information and Communication Technology– are responsible for network and telephone system as well as other IT-services.

One of the groups at USIT, the XML-group, have developed a Topic Map based knowledge base, the Houdini topic map, for systems operation, administration and maintenance documentation which is used by Houston.[24]

Next is presented a summary of the documentation, limited to the interest areas of the thesis. The documentation is mostly an informal translation of the department's documentations[27], which are in Norwegian.

Houdini is USIT's service registry and operations documentation system. Basic information about the services, responsibilities and the dependencies between the services will be stored. Documentation already available in the web are linked by Houdini.

The data model is built on the ISO standard Topic Maps. This makes it possible to delegate responsibility and information to the individual responsible groups at USIT and seamlessly merge together different types of information from different sources.

According to the projects document, the purpose of the topic map is to have a binding point of documentation and systems operation. This will be used as the core of the operations center, in order to plan maintenance and downtime if something goes wrong, and, to avoid duplication of information at different sources.

### 2.6.2 Houdini's ontology / data model

According to the designers, *USIT operates services, and it is these that need to be represented. So it is service which is the basic entity that the whole system is modeled around. All services must be rooted in a group, and there shall be one person who is responsible for the documentation of a service.* The main entity types defined are organizational unit, person and service[25].

There were main guiding goals in designing the data model:

- Finding a common set of information that must or may be registered for a service. Each single service will be represented by a topic. Each topic will have mandatory and optional characteristics.
- Different service types form a class hierarchy, which is useful for arranging the services. Each subclass can add special attributes of its own.
- Document and show how different systems are based on and depend on each other.
- Integration of information from existing sources.

Some of the definition of the subjects at Houdini and their published subject identifiers are shown below.

**Published subjects for operating documentation**

This is a collection of published subjects for operational documentation, with emphasis on services and dependencies between them, developed and published by USIT.

**Subject Indicators:**

- The concept of 'service': A service is something that offers a useful IT function; a computer, other hardware device or software running on one. (PSI:http://psi.uio.no/usit/ddok/ # service )
- The concept of 'person': A person is an individual. In other words, 'person' as defined here is the same as the everyday understanding of the word.
  (PSI:http://psi.uio.no/usit/ddok/ # person )
- The concept of 'strong dependency': A strong dependence is a relationship between two services, one of which is 'dependent' and the other is 'depended'. The dependence is such that if the depended does not function, the dependent cannot provide its service.
  (PSI: # http://psi.uio.no/usit/ddok/ strong dependence )
- The concept of 'weak dependency': A weak dependence is a relationship between two services, one of which is 'dependent' and the other is 'depended'. The dependence is such that if the dependent fails, the dependent will lose some of its service or functionality, but still provides service.
  (PSI: # http://psi.uio.no/usit/ddok/ weak dependence )
- The concept of 'resource person': A resource person is a person who has thorough knowledge of a service.
  (PSI: http://psi.uio.no/usit/ddok/ # resource person )
- The concept of 'doc-responsible': A doc-responsible is a person who has written, or have a thorough knowledge of, the documentation for a service.
  (PSI:http://psi.uio.no/usit/ddok/ # doc manager )
- The concept of 'short-description': A short description is a short and concise description of a service.
  (PSI: http://psi.uio.no/usit/ddok/ # short-description )
- The concept of 'long-description': A long description is an exhaustive description of a service.
  (PSI: http://psi.uio.no/usit/ddok/ # long description )
- The concept of 'up-check command': An up-check command is a description of how a person can test if a service works.
  (PSI: http://psi.uio.no/usit/ddok/ # top check )

- The concept of 'mailing list': A mailing list is a collection of email addresses, so that all who are on the list receive an email sent to the list. A mailing list has an email address.

  (PSI: http://psi.uio.no/usit/ddok/ # mailing list )

- The concept of 'common problems': Common problems are a list of the most common mistakes that a service suffers.
  (PSI: http://psi.uio.no/usit/ddok/ # common problems )

- The concept of 'Box': A network box is a piece of hardware that is placed as a node in a network and manages network traffic through this point.
  (PSI: http://psi.uio.no/usit/ddok/ # this box )

- The concept of 'machine': A machine is a piece of hardware that offers one or more services.
  (PSI: http://psi.uio.no/usit/ddok/ # machine )

- The concept of 'webapp': A webapp, or web application is an application where the user interface is available on the web.
  (PSI: http://psi.uio.no/usit/ddok/ # webapp )

The main ontology can be summarized as follows:

- The main entity types are ICT-service or simply Service, Person and OrgUnit, short for Organizational Unit.
- An ICT-service, which is an instance of the type "ICT-service" or simply "Service", must have
  - A person responsible for its documentation.
  - An OrgUnit which is its owner.
- An ICT-service may have one or more service experts, called responsible person.
- The association between services is either of "strong dependence" or "weak dependence".

A Service will have a set of mandatory and optional characteristics which describe it, all of which are published subject identifiers. The following are mandatory characteristics of any service at Houston.

- A string of short description
- An email address, which is a collection of the email addresses of all those who will receive mail on behalf of the service.

- A service code, and
- A service name.

A person must have a full name as mandatory characteristics, and optionally a user name and short name.

An OrgUnit must have a name, an email address, a homepage URL, and a URI showing a telephone list as mandatory characteristics. Optionally, It might have child units or a parent unit.

The topics , the symbols representing the actual things or concepts, are the binding points of all information about the corresponding subject. At Houdini, there are more than a thousand different topics, representing services, people, organizational units and associations among them. The topic for people is usually their full name.

A web applications screen shot, as shown in figure 2.2 shows the topics as binding points of information in the topic map. The topic clicked here is a unix server represented by the topic 'login1.uio.no'.

The main associations of interest in the topic map can be grouped into the following types:

- Those association types between a service and an OrgUnit: these are defined by the concept of "responsible orgunit".
- Those associations between different services: these are the associations defined by the concepts of "strong dependence" and "weak dependence". There is also the special "Subclass, Superclass" relationship due to the class hierarchy of services.
- Those association types between persons and services. We have the following associations:
  - The associations between a service and the documentation responsible person is explained by the concept of "doc-responsible".
  - The associations between a service and the experts of the service is explained by the concept of "resource person".
- There are also those associations between the entities and their characteristics. Examples are the concepts of "short description" and "email-list" of a service.

# Chapter 3

# Model, Methodology and Approach

## 3.1 Model

The basic model used in the WTC algorithm is the graph model. This chapter begins with a very selected introduction of set theory and the graph model.

### 3.1.1 Sets

A set is a collection of things. All manner of useful collections can be represented or denoted using sets[34]. In set representations, the elements of the collection can be real or imaginary, physical or abstract. Abstract things like numbers and points can be represented by sets. Real physical thing collections like the set of apples or of people can also be elements of a set. In computer science, useful representations by sets are composed of bits, bytes, pointers and blocks of storage. Usually, the significance of set representation is not in defining the elements, but in *forming abstractions of representation*. These abstractions of representation can take many forms in different media[37].

A set consisting of finitely many elements is called a finite set. We usually write such sets as A = $\{a_1, a_2, \ldots, a_n\}$. In such notation, the set is denoted by A while the representations of the elements are listed inside the braces.

### 3.1.2 Set Relations

Sets can have different relations with one another. One of the set relations is the subset relationship. If we have two finite sets A and B, we say A is the subset of B, written $A \subseteq B$, if all elements of A are also elements of B.

Set theory is used for an exact description of verbal statements. For example, the subset relationship can be used to denote containment of concepts within another, or for event descriptions as shown in table3.1.

| verbal statement | set theory expression |
|---|---|
| A more general concept B contains a specific concept A | $A \subseteq B$ |
| if A occurs, so does B | $A \subseteq B$ |

Table 3.1: Set theory as a precise expression of verbal statements.

### 3.1.3 Graphs

A graph consists of a set of vertices, or nodes, connected by edges, or lines or curves. The usual way of drawing a graph is by representing the nodes as circles or points, and connecting them by lines. We say a graph G has the set of vertices V and the set of edges E, and denote it as G(V,E).

Definition: A graph is a pair (X, $\gamma$) that consists of a set of nodes X and a mapping $\gamma : X \rightarrow X$, formed by the arcs or lines between the points x $\epsilon$ X[34].

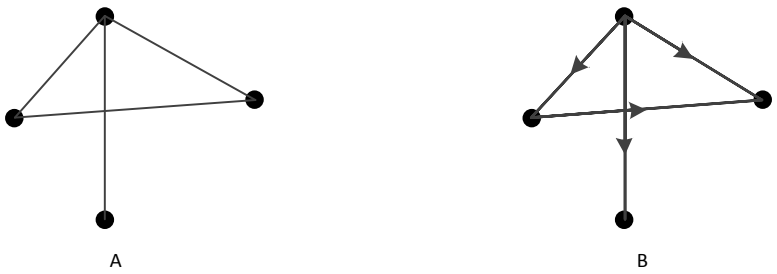


Figure 3.1: some graphs. A is undirected, while B is directed

The graphs at figure 3.1 show some elementary graphs. They depict only one sort of graphs, the labeled graph. But graphs can have different representations and can be quit complex.[34]

We categorize graphs in the way the nodes are connected as directed or undirected graphs. The graphs as shown in figure 3.1 (A) where the lines connecting them are not directed are called undirected graphs. When the lines connecting the nodes are arrows, as in figure 3.1 (B), we say the graph is directed.

Graphs can also be categorized as cyclic or acyclic. A cyclic graph is a graph in which one can return back to the starting point. An acyclic graph goes away from the starting point, and provides no direction for returning back to the starting point.

For two graphs G(V,E) and G'(V',E'), we say that G' is a subgraph of G (and G a supergraph of G')if V' is subset of V and E' is subset of E. A subgraph can often be built by deleting specific vertices and/or edges from a graph[4].

### 3.1.4 Human-computer systems and graphs

As with all problem representations, a graph based representation is used to provide a particular perspective on a problem. We draw graphs to denote some kind of information which requires the depiction of entities and relationships among them. The important thing here is that, graphs enable us to represent the aspects of a system that we want to emphasize clearly. In computer science, those aspects we want to emphasize are usually aspects of relationships between items of data.

An example of a common graph in computer science is a tree, which is a graph with one and only one path between every two nodes[5]. One common application of a tree graph is in describing file systems as depicted in diagram 3.2.

In human-computer networks, the aspects we want to emphasize are usually aspects of relationships between the components of the system. For example, we depict membership i.e., certain things belong together, and, causalities,i.e., certain things depend on other things, using graphs[35]. The connecting edges between the nodes often represent some qualities of those relationship aspects,such as[34],

- A dominates B (directed),
- A depends on B (directed),
- A is associated with B (usually undirected),

and so on. When the relationships are one-way, the joining lines will be directed, and when the relationships are multi-way or symmetric, the joining lines will be undirected.

Many concepts and information in human-computer systems are represented by graphs. It is, for example, most common to represent network topologies using graphs in human computer systems. This is shown in fig 3.3.

A particular need in discussing human-computer systems, where elementary graph theory helps us a lot, is the depiction of the system by joining the items having a certain relationship, with the arrow going in certain direction. These direction oriented depictions are useful, because the items will be joined only if certain criteria are met. On this basis, the example we had in the background, in section 2.4.3, can be depicted using graphs. This is shown in fig 3.4.

### 3.1.5 Graph problems

There are many graph problems that have interesting applications in many fields of study. In this section, two of them, *the shortest path problem* and *the transitive closure algorithm*, are discussed.

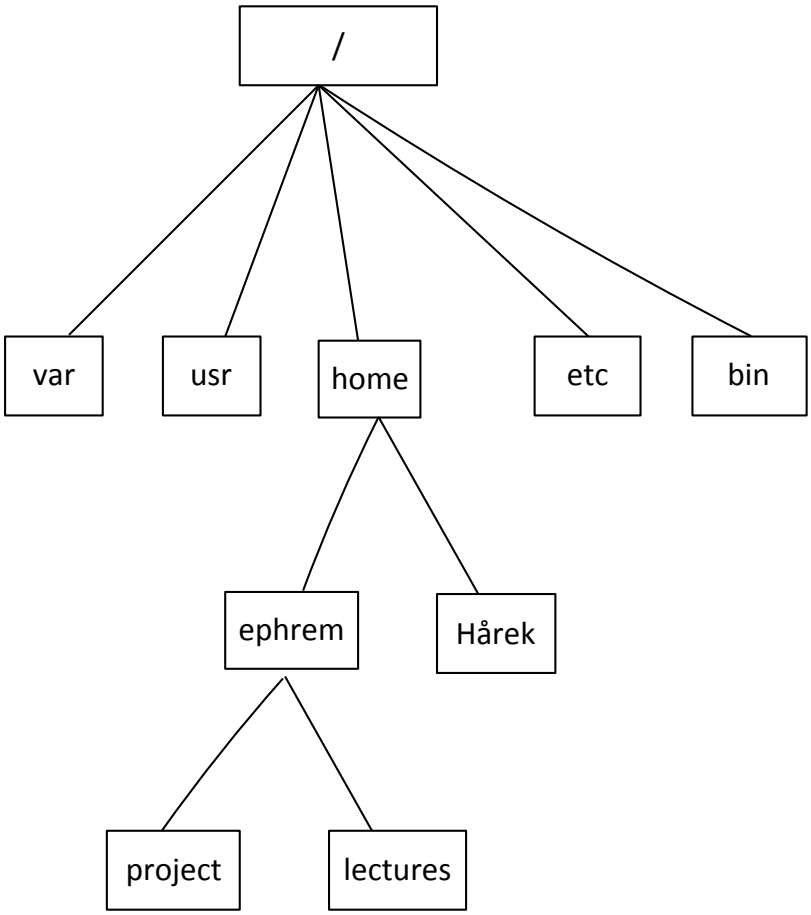


Figure 3.2: part of a unix file system represented by a tree

The shortest path problem is a problem of finding the shortest total path between two given nodes in a graph. There are many algorithms to solve the shortest path problem. One of the well known is the Dijkstra algorithm[5]. Such algorithms are useful when there exist more than one path between nodes, which is the case in graph representation of human-computer systems. The basic fact used in Dijkstra's shortest path algorithm is that, if the minimal path between two nodes A and B is found, it implies knowledge of the minimal path between the starting node A and any intermediate node between A and B[5]. The steps can be rephrased as follows:

1. Start at A, and find the shortest distance node connected to it.
2. Make the new shortest distance node from A as a starting point and find the shortest distance node from it.
3. Repeat step 2 until you reach node B.

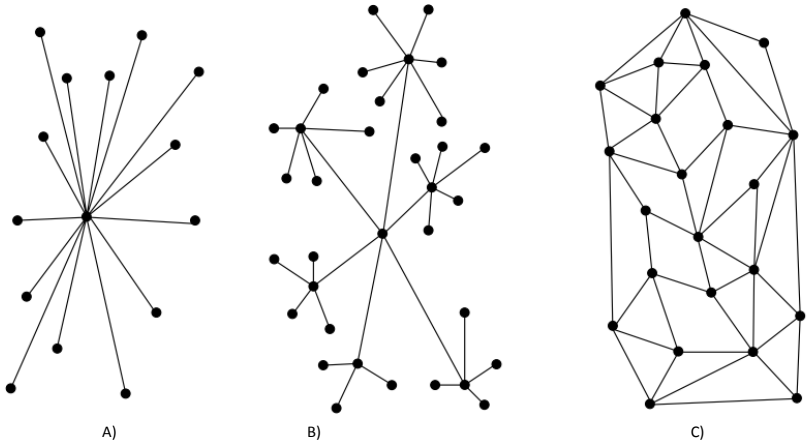


Figure 3.3: graph representation of network topologies, A) centralized, B) decentralized or hierarchical and C) distributed mesh. Shown here with permission from the author of [34]


Figure 3.4: graph representation of the connection characteristics of a server and a switch

4. The shortest path between A and B is the sum of all paths found in steps 1, 2, and 3.

In mathematics, a binary relation R is transitive over a set S if it fulfills the following: for elements a, b and c in the set, if 'a R b' and 'b R c', then 'a R c'. For example, the binary relation 'is greater than' is transitive on any set of numbers. For any three numbers a, b and c, if 'a $>$ b' and 'b $>$ c' then 'a $>$ c'.

The transitive closure of a binary relation R over a set S is another transitive relation that contains the binary relation R and is minimal. The transitive closure of R is given by the intersection of all transitive relations containing the binary relation R. A very important point in transitive closure relation is that, if the binary relation R is transitive, the transitive closure relation will be the same transitive relation R. Otherwise, the transitive closure is a different transitive relation.

The application of the transitive algorithm to graphs is to answer reachability questions. For example, consider three nodes in a graph as shown in fig 3.5.

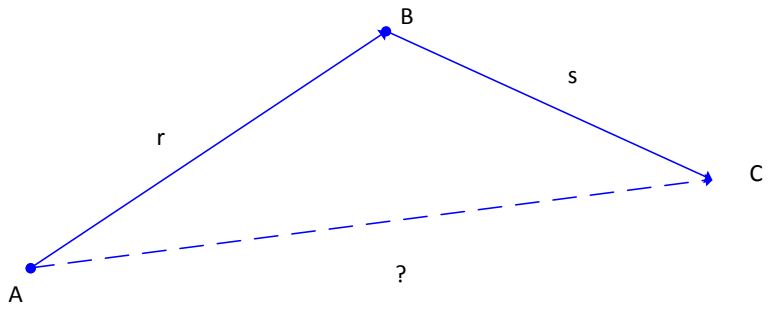


Figure 3.5: application of transitive closure to graphs

The binary relation tells us about the connection between two nodes. But, if we have a connection between A and B that it r, and, a connection between B and C that is s, what is the connection between A and C ? A transitive closure algorithm is an algorithm used to find such a path.

## 3.2 Methodology

In this section, the methodology used in the project, the WTC algorithm, is explained. This section has three parts. The first section explains general concepts specifically relevant to the methodology. The second section is a detailed elaboration of the methodology. This elaboration is entirely based on the two papers that the originators of the algorithm published to describe their work, [28] and [29]. In the third section, the elaborated ideas are demonstrated using examples.

### 3.2.1 General concepts

**Causality:** The principle of causality states that every change or effect happens due to a certain cause. The main issue with this intuitive and simple principle is the fact that finding cause and effect in a dynamic and complex system is not always an easy matter. Specially, establishing a causal link between the resulted change in a system and the cause for that change is a much desired outcome. But, such discoveries of cause and effect are difficult in complex systems. Such skills are, however, key in many occasions like troubleshooting[34].

**Reasoning System:** A reasoning system is a system which gives answers or conclusions that are results of some sort of reasoning or deduction done by itself. Among other things, reasoning requires figuring out what one needs to know from what one already knows. Thus, whenever a computer system is required to do things that it has not been told,explicitly, how to, the system must reason. In such scenarios, the system must be able to deduce and verify many new facts from those it has explicitly been told[20].

**Inference:** Inference is a formalized reasoning to draw conclusions from given facts. In computing, it is a reasoning performed on data through rules.

**Abductive inference:** Abduction is a kind of inference to generate hypothesis that are best explanations of a conclusion. While deduction arrives at the consequence or conclusion from given facts or hypothesis, abduction goes the opposite, and finds the best explanation of a conclusion or an end result.

**Rules of inference:** The constraints of the reasoning process which limit what kind of conclusions can be drawn from given facts are called the rules of inference. Rules of inference preserve truth in such a way that, if the starting formulas are true, the result of performing a rule of inference on them must also be true.

### 3.2.2 Elaboration of The WTC algorithm

The WTC algorithm is an automated reasoning method for the generation of *human-understandable* inferences of causal relations. *Human-understandable* statements follow the usual language rules. In fact, in the WTC algorithm, the statements are limited to subject-verb-object constructs only.

The WTC algorithm is designed to aid humans to understand causal relationships between components in a complex system. This reasoning method has three main parts[30]:

- A text file used as the knowledge base, the facts file. This text file follows a specific syntax, and is filled with statements of assertion about the system. Each statement is called a base fact, and is an entity–relationship–entity construct.
- Another text file, the rules file, which contains the rules of inference.
- A Perl CGI program, a prototype, that uses the above two files as inputs and provides an interactive web interface for answering queries about the system.

This reasoning method first conceptualizes complex systems as directed graphs, with 'entities' as nodes and 'relationships' as edges. The basic idea is to utilize a description of system knowledge for inferring dependencies and causal relations of a more useful and subtle nature.

A statement, written as entity-relationship-entity triple, that describes any invariant and positive system property is called a fact. The collection of all facts, taken to be true from the outset, is what are called the base facts of a problem domain and is stored in the facts file.

### Entities

An entity is a component in the system that can be named and represented by a noun that does not change with time. In the preferred visualization of a system, anything that can have a name and can be viewed as a component of the system is called an entity. Kinds of entities include:

- Physical machines, eg., 'login1.uio.no' , 'sw-248-196.uio.no'.
- Software, eg., Unix, windows explorer.
- Services, eg., 'login service', 'dns service'.
- Classes of physical machines, eg., 'unix servers' ,'windows servers'.

### Relationships

A relationship expresses the concept that connects entities, and is usually a verb. In other words, a concept depicting the interaction between entities in the system is called a relationship. Such concepts include:

- Dependencies, including 'requires', 'provides', 'strongly depends on', 'weakly depends on'.
- Containment, including 'is a part of', 'is an instance of', 'is a subclass of'.
- Causality, including determines, influences.
- Connectivity, including connected to.
- Intent, including promises, uses.

### Facts

Facts are recorded by studying the system and writing the descriptions in entity–relationship–entity triples. The specific syntax of the system requires that facts be subject–object–verb, and be written as:

```
entity|relationship|entity
```

In such facts, the subject and object are system entities while the verb is a relationship between them. There are a number of constraints on the facts that can be recorded in the WTC algorithm's facts file:

- Facts must describe invariant properties of entities of the system and, generally, variation over time is not supported. In the fact sentence, there are two entities and a relationship. The relationship type must therefore be something that does not change over the lifetime of the entities involved.

As an example, consider facts that describe the address of machines in a LAN. If the LAN uses a dhcp-service, it is not useful to record ip–addresses of machines in the fact file, since this would lead to erroneous conclusions. In this sense, the mac address would be a good choice.

- Facts cannot be negatives, only positive assertions.

  For example, if an item is part of another item A, we would state that in a fact. But if it is not, we simply omit it. However, stating that the item 'is not part of A' is not supported in the system.

  The above two points means that, in the WTC algorithm, not every kind of fact can be represented.

- Facts, to the prototype, are syntactic tokens, and nothing more.

  Thus, tokens are known to be entity tokens or relationship tokens by the syntax in a fact. In any fact, the first and third tokens are entities, while the second token is a relationship. But the meaning of an entity token is implicitly found in the cumulation of the facts that describe it. For a relationship token, the meaning is found in the cumulation of the rules that describe it. This is the ontology of the system.

  For example, what does the token 'host01' mean? This is implicit in the set of facts that describe the entity. It means, it is the result of what all facts about the entity are stating about it. If this 'host01' is stated as "host 01 is an instance of workstation" in one fact , as "host 01 is influenced by dns service" in another, and as "host 01 has part external ATA drive" by still another third fact, then those collection of facts are the meaning of the entity so that it would mean the particular 'host01' physical entity in the domain.

  Similarly, what does a relationship, say 'describes', mean? The meaning is implicit within what this relationship concept implies, what its inverses are, etc.

  This understanding of how meaning is stated in the system is fundamental so that the stating of facts is done with thorough understanding of the system. Facts must be carefully selected in their construct, so that they are suitable and useful.

- Facts about an entity are facets of its meaning, and additive to other facts about it.

  According to the system, if a fact is stated about an entity, it is one facet that does not contradict with any other facts about the same entity. This means that the system does not detect contradictory facts, and this part

should be taken care of in the formation of the facts. Unless care is taken not to have contradictory facts in the fact base, misleading and wrong conclusions will be drawn by the reasoning system.

- In this system, it is not assumed that facts are exhaustive from the outset.

  The system is based on an open model of knowledge, in that, new facts can be added later when known. The absence of a fact means only that, it is not known to be true.

- Deleting a fact is handled by out-dating it.

- Inverses are true, but are not recorded as facts. They are represented by rules as described below.

**Rules**

Rules in the WTC algorithm are written by the way they describe relationships. While a relationship describes the interaction between entities, a rule on the other hand describes the interaction between the relationship concepts themselves. Thus, the rules file is full of rules, each of which describes how the relationships interact with each other. The meaning of a relationship concept is determined by the rules that describe it in the rules file.

- Rules are used to infer new facts from base facts and also new rules from existing rules.

  Rules are crafted so that only certain kinds of facts can be implied. This is a core concept in the WTC algorithm. This truth means that, what kind of dependences and causal connection we can grasp as the end result depends on the formulation of the rules.

  A set of rules with specific properties also affect the speed of computation.

- By applying rules repeatedly, it is assumed that all facts will be represented.

  In other words, by some finite application of rules, the system achieves a fixed point state in which no further operations add new facts or rules. This is the principle of convergence as applied in the reasoning system.

- Like facts, rules are deleted by out-dating them. If no inferences are made using the rule, the rule will be as good as deleted.

There are four kinds of rules in the system:

1. Canonical rules

   These rules are used for both saving typing and ensuring that representation of facts are sufficiently precise to be useful. The notation for canonicalization is

   ```
   r=>s
   ```

   This means, everywhere in the file where an r is found, the system should replace it with an s. In the text file which contain the rules, these rules are placed first, since they apply to everything in the file and influence readability.

2. Implication rule

   Implication is a rule used for depicting the relationship between concrete and more general concepts. By the implication rule, denoted by the symbol `->`, we create derived facts that demonstrate a more general relationship between the entities. Every relationship between entities must be a subset of a more general relationship. This rule is written as

   ```
   relationship->implied relationship
   ```

   This means that, if entities X and Y are related by "relationship", they can also be related by "implied relationship" without risking error. Also, the latter will be a more general relationship between entities X and Y than the former concrete relationship.

   As an example, let us consider the concepts 'influences' and 'determines'. If A determines B, we can also say that A influences B. But if A influences B, we cannot generalize that A determines B. This is because determining is a type of influence. But influence is not only determining. It can be any other type of influence than determines. In this context, we can say that if an entity determines another, it also influences it. Thus, determines implies influences, but influences does not necessarily imply determines. In the notation of the system, this is written as

   ```
   determines->influences
   ```

   This can also be represented as a set relationship between the two concepts. We can say, in other words, that the concept of 'determines' is contained within the more general concept of 'influences'. Therefore, It is a subset relationship, where 'determines' is the subset of 'influences'. This idea is shown in a Venn diagram in fig 3.6.

   By similar analysis, all implication relationships in the WTC algorithm are also reckoned as having the subset relationship between them.

3. The inverse rule

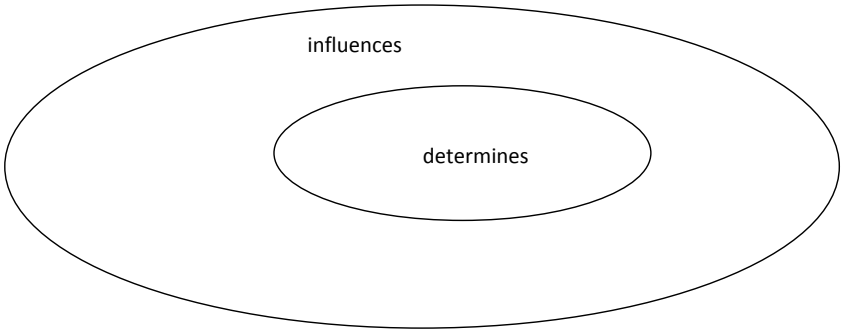


Figure 3.6: A set relationship between two concepts shown in a Venn diagram

The inverse rule is used to handle direction. Every stated fact must have an inverse fact which is also true. The handling of such facts is done using the inverse rule in the reasoning system, rather than writing them down in the knowledge base. The inverse rule is denoted by <>. We write

`relationship<>its inverse`

This is understood to mean that for any entities X and Y,

if

`X|relationship|Y`

is a fact, then

`Y|its inverse|X`

is also a fact.

An inverse rule can also be explained using implication. Using implication, the above rule can be rewritten as 'for every X and Y, if X *relationship* Y, then Y *inverse relationship* X and vice versa.' Therefore, inverse rule can be viewed as a special kind of implication. Some facts are self-inverse, e.g., if 'X is a peer of Y', then 'Y is a peer of X'.

4. Weak Transitive rules

   Weak transitive rules are used for addressing connectivity problems in the system. This is a rule which shows what the relation between entities X and Z will be if entities X and Y are related by a relation 'r' and entities Y and Z are related by a relation 's'. This rule is a way of addressing complex systems interdependence which is usually built by transitivity. The notation for weak transitivity is the symbol `^` .

   In transitive rules, we involve three entities and three rules. `r^s^t` means that for every X , Y and Z, if X r Y and Y s Z , then X t Z . As an example, consider the following two facts

```
server1|requires|network-connectivity
network-connectivity|is provided by|sw1
```

If we want to know how server1 and sw1 are related based on only those facts, we cannot formulate the relationship. But if we have the weak transitive rule

```
requires^is provided by^strongly depends on
```

The rule will guide us that, they are related as follows:

```
server1|strongly depends on|sw1
```

When the consequent relationship does not match at least one of the antecedents and is usually more general than at least one of them, the transitive rule is called 'weak transitive.' As such,

```
requires^requires^requires
```

is a (strong) transitive rule but,

```
has part^controls^controls
```

is a weak one.

As shown by the above example, these rules are the key to turning a logic computation into a graph computation when computing queries about facts, which is done using the prototype engine of the system.

**The prototype engine**

The engine of the WTC algorithm is a Perl-CGI program written by professor Alva Couch of Tufts. This program is available as a free software under the GNU license at [30]. In this program, abstract mathematical principles are transformed into result yielding computations.

- Sets, Graphs and Perl

  Sets and Graphs are represented by hashes in Perl[35]. Sets are hashes with values of 1, and keys would be the members of the set. For graphs, we may use different ways of representation. One usual way is the adjacency list, where the neighbors of each edge are listed. This way, operations in graphs and sets are converted into operations in hashes. This is made use of extensively in the prototype.

- Abductive inference is changed to graph computation

  First, binary relations between entities, which are our facts, are considered as labeled graphs.

```
A|provided by|B
B|depends upon|C

provided by     depends upon
A----------->B------------>C
```

Second, the transformation of a new relation between A and C by the (weak) transitive rule is considered to be an action on the graph to create a new edge connecting A and C.

`The rule: provided by^depends upon^depends upon`

The action: Adds edge, as shown below in fig 3.7

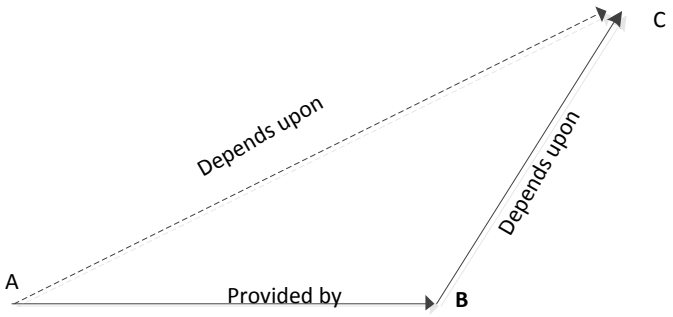


Figure 3.7: Abductive inference to graph computation

Thus, abductive inference is changed to graph computation, something that computers can do using well established algorithms. Also, since we can "measure" the length of the new edge by counting *how many inferences are done to reach at it*, a problem of logic, i.e., inferring `A^B^C` , is turned to a problem of distance, i.e., how many `A^B^C` applications are needed.

This operation has many benefits. Once the change of abductive inference to graph computation is achieved, answering queries about the most related entities to a chosen one is possible. In addition, any derived fact can be explained as a shortest chain of base facts that were used to infer it. This explanation is human-readable, thus the person will understand it without knowing the details of how it was inferred.

**The pivotal relationship of a problem domain**

In order for the reasoning system to work, we must have a pivotal relationship. This is something the program runs to achieve, thus performing all the inferences. In other words, in the complex graph that represents the system, looking for a pivotal relationship means routing the way towards a specific

edge. All the ways leading to the edge are then explanations of the dependence or causal connection path in the network, in different ways.

Typically, the pivotal relationship is the most abstract relationship that remains interesting. Therefore, to have this pivotal relationship, we think of a set which encompasses all the other relationship concepts. Put another way, the pivotal relationship should be implied by all the relationship concepts in the problem domain. If we have a relationship concept that does not imply the pivotal relationship, then we can find no result pertaining to it because the system is doing all inferences that lead it to the pivotal relationship.

The reasoning system assumes a thorough study of a problem domain,so that to craft 'the central relationship of a problem domain.' This is something the user wants to know. The rule set describes many relationships, in fact, an intractable number of them. The pivotal relationship is a statement of what one wants to know, and that *limits the graph computation*, since, when looking for a pivotal relationship, it means looking for a specific kind of edge in the graph.

The system is then asked to "explain" the edge. This is what is provided as the explanation of any relationship between entities. Often a less specific pivot is better than a strict one, for example, "can influence" is better than "determines". This is because, less specific relationships are at a higher abstraction level than more strict ones. A higher abstraction level means it is the result of many lower leveled relationships, thus providing explanation in terms of them.

As an example, if we have relationship concepts like 'determines', 'contains', and 'is part of' in our facts, then 'influences' can serve as the pivot since it is implied by all of those concepts. This is equivalent to say that, if we choose 'determines' to be the pivot, facts that have 'is part of' as relationship will have no result or no explanation, because 'determines' is not implied by 'is part of'.

**Queries**

There are many kinds of queries that can be answered in this system. In the first kind of query, the goal is to find all the entities related to a chosen one, with a known relationship concept. This means that, if the chosen entity is X, and the known relationship is r, then we find as much entities Y such that X r Y is satisfied.

Consider the relation to be 'depends upon' and the chosen entity to be node A in fig 3.7. The first query is what answers the "which?" in `' A|depends upon|which?'`. Put another way, this query means that, given an entity A, which entities do we reach at using an edge named 'depends upon'? In our simple example, the answer is node C.

The system answers such queries as follows:

- First, calculates all new facts and new rules by using the rules and base facts.
- A new collection of facts is generated, base facts plus derived facts.
- The facts that satisfy the particular query are then presented.

The second kind of query is explaining a particular fact X r Y in terms of all the other facts that were used to reach at it. By the graph computation, this means that, if we have the two nodes X and Y related by a derived or inferred relationship r, we had other intermediate nodes and edges that were base facts. The second kind of query is the presentation of those facts in human-readable format. In short the second kind of query asks this: 'when we reach at an inferred fact 'X r Y', what minimum number of weak transitive rules and what number of base facts did we apply to reach at it?'.

In our example, figure 3.7, this would mean answering or explaining how we reached at the conclusion

``A|depends upon|C''.

The answer is two facts and one inference, which are facts:

```
A|provided by|B
B|depends upon|C
```

and the rule

```
provided by^depends upon^depends upon
```

Thus, the first query is about what an entity is related to, given an entity and a relationship, while the second query is about the fact bases and number of inferences needed to reach the new inferred relationship between two entities, put forth as the 'distance' between the entities. This distance is measured by the minimum number of weak transitive rules that must be applied to base facts to reach at this inferred fact. The result of this query is what we call an explanation or history of the relationship.

The basic conceptualization is that the binary relation between entities is seen as labeled graphs. An inferred kind of relationship is an action of adding a new edge between nodes, to connect them directly.

### 3.2.3 Demonstration of Methodology

This section will demonstrate how the WTC algorithm is applied with an example. We refer to the example we used in topic maps representation, in chapter 2, section 2.4.3. We can use both the two approaches to represent this using the WTC algorithm.

1. Based on topic maps knowledge base, the topic-map approach:

   This methodology utilizes the topic maps knowledge base as follows:

   - Take the topics to be entities, and represent the association as a directional relation.

     Direction of relation is one difference between the two approaches. In topic maps, direction is made unnecessary and the role of each topic in the association is included for that. In the WTC algorithm, relations are directional.

     Thus, we will start from one direction of the association. Instead of 'strong dependence' which is a noun phrase, we use 'strongly depends on', a verb showing the dependence of the server on the switch.

     `login1.uio.no|strongly depends on|sw-248-191.uio.no`

     This is a one directional relationship and is an entity-relationship-entity triple, a fact. Facts like this are taken to be true from the outset and are therefore the base facts. But from this fact, the reverse fact is also true,

     `sw-248-191.uio.no|is strongly depended upon by|login1.uio.no`

     This is handled by one of the rules in this approach, called the inverse rule.

   - Craft rules:

     Rules are expressed in terms of the relationships they manipulate. Therefore, we can have them as follows:

     **The canonicalization rule:**

     Since this rule is used for shorthand notations to save typing and also to make facts precise, for the above example, we can have the canonicalization rule:

     strongly depends on=>strong , and write the above sentence as

     `login1.uio.no|strong|sw-248-191.uio.no`

     The system will understand this to mean the same as the tokens

     `login1.uio.no|strongly depends on|sw-248-191.uio.no`

     When we have several hundred facts to type, this rule is beneficial.

     **The inverse rule:**

     In the above, we have seen that the base fact

     `login1.uio.no|strongly depends on|sw-248-191.uio.no`

     has the reverse fact

     `sw-248-191.uio.no|is strongly depended upon by|login1.uio.no`

     which is also true. Thus, we will have the inverse rule

```
strongly depends on<>is strongly depended upon by
```

This is understood by the system to mean that for any entities X and Y in the system, if

```
X|strongly depends upon|Y
```

is a fact, then

```
Y|is strongly depended upon by|X
```

is also a fact.

**The implication rule:**

In the above, 'strongly depends on' is a subset of a more general kind of relationship 'is influenced by', for example. We say that 'strongly depends on' implies 'is influenced by'. Therefore, any entities X and Y related by 'strongly depends on', can also be related by 'is influenced by' without risking error. The above rule is written as

```
strongly depends on-->is influenced by
```

**The weak transitive rule:**

Consider, in the example above, there were a second switch called sw2 that is not directly connected to login1, but on which the first switch is related to by 'is influenced by'. The relation of login1 to sw2, will be the result of the fact that login1 is related to sw1 and sw1 is related to sw2. It can be seen that login1 and sw2 can be related by 'is influenced by'', because

```
login1.uio.no|strongly depends on|sw-248-191.uio.no
```

and

```
sw2|is influenced by|sw-248-191.uio.no
```

are facts. To code this as rule, we write

```
strongly depends on^is influenced by^is influenced by
```

which means that if

```
X|strongly depends on|Y
```

and

```
Y|is influence by|Z,
```

then

```
X|is influence by|Z
```

2. Based on architectural base facts and corresponding rules, the system-description approach:

   In this approach, a knowledge base is created from scratch, based on system architecture. An attempt is made to limit facts only on the architectural level, so that the associations depicted by the topic maps construct are inferred from them. In the above example, we resort to record the lower level relationships in the system that caused 'strong dependence'

between the server and the switch. Instead of trying to put the relationship between the switch and the server in a concept of relationship like what the topic map does, we simply record the assertions about the server and the switch. We then let the system infer the kind of dependence by using these base facts and the rules. The fact that the server 'cannot function without the switch giving its service' can then be derived by the prototype, using the lower level facts and the rules crafted accordingly.

For example, consider the following restatement of the above:

```
login1.uio.no|requires|network-connectivity
sw-248-191.uio.no|provides|network-connectivity
```

These are simple architectural facts that can be recorded to the knowledge base.

Now consider the reverse rule,

```
provides<>is provided by
```

This rule enables the system to add the new fact

```
network-connectivity|is provided by|sw-248-191.uio.no
```

to the knowledge base.

Let us also consider the following weak transitive rule:

```
requires^is provided by^strongly depends on
```

This rule is stating that, for three entities X, Y and Z in the system, if entities X and Y are related by 'requires' and Y and Z are related by 'is provided by', then X and Z are related by 'strongly depends on'. In other words, according to this rule, the facts

```
X|requires|Y
```

and

```
Y|is provided by|Z
```

give rise to the fact

```
X|strongly depends on|Z
```

Therefore, using this rule, the system reaches at the inferred fact that

```
login-service|strongly depends on|sw-248-191.uio.no
```

since we have the fact

```
login-service|requires|network-connectivity
```

as base fact, and

```
network-connectivity|is provided by|sw-248-191.uio.no
```

as a derived fact using the reverse rule. This is demonstrated in fig 3.8 below.

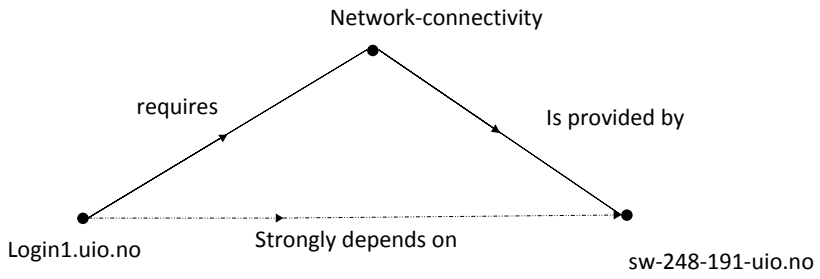


Figure 3.8: In the system-description approach, 'strongly depends on' is inferred from concrete architectural relationships. The solid lines are the concrete relationships, while the dotted line is the inferred relationship.

## 3.3 Approach in detail

The project uses similar procedures of the two approaches, as demonstrated above, for selected use cases of the IT system at University of Oslo. The thorough study of the domain, both from the topic map data and the available detail of system knowledge, will be performed for creating useful facts, a pivotal relationship for each case, and appropriate rules of inference.

### 3.3.1 Topic-map approach, Using the Houdini topic maps Knowledge Base

In this case, the conceptualizations used for the representation of the system by the topic map will be preserved. What is done is transform the topic maps knowledge base to be the knowledge base of the WTC algorithm. Topics will be nodes, and the relationships derived from the associations will be used as edges. Thus, in this approach, the Houdini system is seen as a directed graph of 'topics' and 'relationships.'

- The topics *defined by the topic map* will be taken to be entities.
- The associations of the topic map between the topics will be translated to one directional relations between the entities.
- These `entity|relationship|entity` triples will form the base facts of the system.
- The inverse rules will be crafted to create the inverse facts from the base facts.
- The implication rules will be crafted to create new set of rules and facts that are more general than the base relationships and facts.
- Weak transitive rules will be crafted to take account of transitive relations between topics.

- A pivotal relationship will be selected so that it will both present the outcome we want and will also be a universal abstraction of all the relationships.

### 3.3.2 System-description approach, Using Systems Knowledge to Create the Knowledge Base

In this case, we focus on the architecture of the system to create the knowledge base. We will conceptualize the system as a graph of named entities and relationships among them. The named entities are components in the system like physical machines, services, people and the like. The relationships we focus on will be architectural relationships between those named entities. The expectation from this approach is that it may result in further dividing of the topic maps associations into lower level relationships. Another deviation is that there will be no limitation to the ontological commitments of the Houdini topic map.

As the example demonstrated by figure 3.8 shows, facts transformed from the topic map are considered to be the result of other, lower level facts of architectural nature in this approach.

In short, this method simply records architectural relations, and lets the inference rules infer the type of dependence chosen by us between the entities. This has an advantage in clarifying the causal relationships for troubleshooting.

The steps we follow will be,

- Write down architectural facts in terms of

  ```
  named-entity|relationship|named-entity
  ```

  triples. These will form the base facts of the new knowledge base. Unlike the topic map approach, where the relationship concepts have to be derived from the associations of the topic map, in this approach relationships will be selected from system knowledge. The same holds true for the pivotal relationship.

- Inverse rules will handle the transformation of the inverse facts from the base facts.

- The implication rules will be crafted so as the lower level architectural relationships will be the more specific relationships which imply higher level abstractions.

- Weak transitive rules will be crafted to take account of transitive relations between topics.

# Chapter 4

# Results

The use case is implemented for a random sample of services documented by the Houdini web application, http://intra.usit.uio.no/houdini/. We select some services as described by the topic maps for the use case, which comprise a total of more than 100 topics and 9 association types with more than 100 associations to deal with.

A table showing the selected services, the associations and all the topics associated with the services, as presented by the topic map, are shown in the appendix. One of the services is shown in table 4.1 as an example.

## 4.1 Results of topic-map approach

The result for this approach is available at the site *http://ephrem.vlab.iu.hio.no/cgi-bin/topic-map.cgi*

### 4.1.1 Finding Relationships

The topic-map approach is solely based on the topic maps knowledge base, therefore associations are changed to relationships, and the topics will automatically be the related entities. Primary focus will be on those aspects that inference is most beneficial. First, focus is on the service–to–person and on service–to–service relationships. According to the topic map, there are four links between those topic types[25]:

1. A service–to–service association, 'strong dependence'. This association will be transformed to 'strongly depends on' in one direction and 'is strongly depended upon by' in the reverse direction when used in the WTC algorithm.

2. A service–to–service association, 'weak dependence'. This association will be transformed to 'weakly depends on' in one direction and 'is weakly depended upon by' in the reverse direction when used in the WTC algorithm.

| Selected topic | Related topics | Association type |
|---|---|---|
| login1.uio.no | 'Login–service for UiO (one of login.uio.no)' | short description |
| login1.uio.no | 'fping login1.uio.no && ssh login1.uio.no' | up-check command |
| login1.uio.no | 'unix–drift@usit.uio.no' | e-post list |
| login1.uio.no | 'M29,25 bsd–bladecenter5.uio.no' | physical location |
| login1.uio.no | 'Dell Inc. PowerEdge M600' | device description |
| login1.uio.no | Trond Hasle Amundsen | doc-responsible |
| login1.uio.no | Trond Hasle Amundsen | resource person |
| login1.uio.no | sw–248–191.uio.no, bsd–bladecenter5.uio.no, sw–248–56.uio.no | strong dependence |
| login1.uio.no | domain_dvergen.uio.no, domain_nissen.uio.no, domain_huldra.uio.no, ypserv_radius1.uio.no, ipv6–connected_virtual, viktighet-2_virtual, nfs_platon.uio.no, tsm_sumo.uio.no | weak dependence |

Table 4.1: details for service login1.uio.no, its related topics and associations.

3. A service–to–person association, 'doc–responsible'. This association will be 'has doc–responsible' when relating from service to person, and 'is doc–responsible for' when going from person to service in the WTC algorithm.

4. A service–to–person association , 'resource person'. This association will be 'has resource person' when going from service to person, and 'is resource person for' when going from person to service in the WTC algorithm.

Thus, what was non–directional in the topic map, has now turned to a directional relationship. The most imminent advantage of the WTC algorithm is the ability to find existing relationships with specific properties using transitivity. The topic map contains binary relations between services that are 'weak dependence' and 'strong dependence', which are respectively changed

to 'weakly depends upon' and 'strongly depends upon'. If we formulate a weak transitivity rule as follows:

```
weakly depends on^weakly depends on^weakly depends on
```

then we can arrive at what the dependence between services that are not directly addressed by the Houdini topic map will be. Similarly,

```
strongly depends on^strongly depends on^strongly depends on
```

will give us another transitive result. For example, we see no relationship between login1.uio.no and viktighet-1_virtual in the Houdini topic map, refer to figure 2.2. But a 2 step weak dependence relationship, shown by the red color code, is inferred between those topics after applying those rules, as shown in the screen shot in figure 4.1.

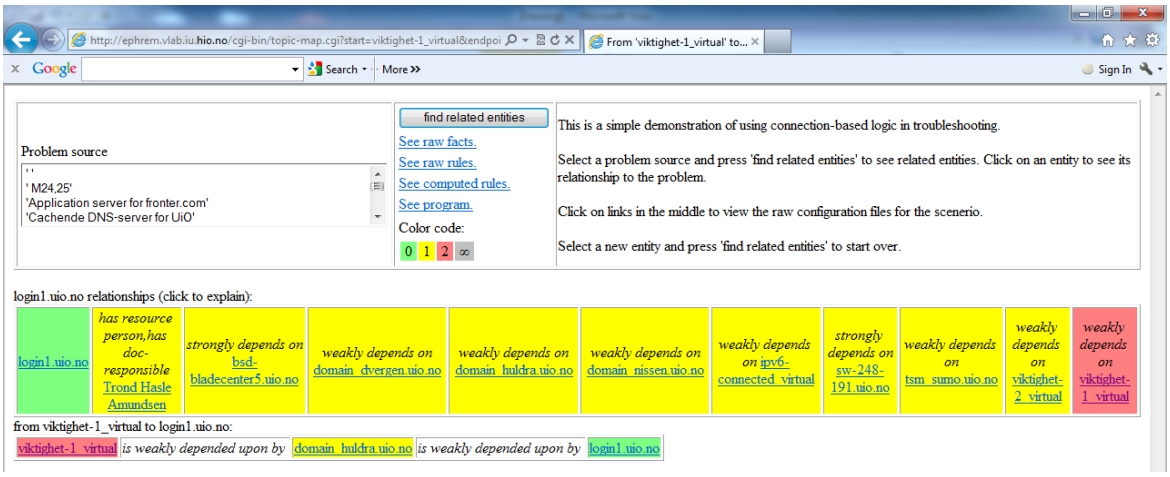


Figure 4.1: A transitive relationship, not depicted by the Houdini topic map, is discovered and presented by the WTC algorithm.

The explanation, 'from viktighet-1_virtual to login1.uio.no', shows that this dependence is inferred from the two base facts namely

```
login1.uio.no|weakly depends on|domain_huldra.uio.no
```

and

```
domain_huldra.uio.no|weakly depends on|viktighet-1_virtual
```

The reasoning system applies the inverse rule for the opposite direction, and then the weak transitive rule is used to reach at the final inference.

Another useful relationship may be between a resource person for one service and the services on which this service depends on. In other words, if person1 is resource person for service1, and service1 weakly or strongly depends on service2 and service3, it is clear that person1 has some interest on those two services. This means, for any resource person, he or she will be able to see which services have dependence upon the service that he or she is responsible for. Let us represent this relationship concept with 'has interest in'. Thus we want inferred facts of the form

```
resource person|has interest in|service,
```
or its inverse which is,
```
service|is interest of|resource person
```

This kind of facts are like the edge labeled by X in fig 4.2.

Another relationship that can be inferred by the WTC algorithm is a person-to-person relationship, not depicted by the Houdini topic map. This relationship can be explained as follows: When a resource person, person1, clicks a service that he is responsible for, say service A, he finds all the services that service A strongly depends on. In the Houdini topic map, this is available. But whom shall he contact about those other services? For that to happen, the person has to click the service, get another window, and read the resource person for that service. This is because, the topic map cannot use inference to infer new relationships. But in the WTC algorithm, we can formulate this as an edge and try to find the edge. This is the edge labeled by Y in fig 4.2.

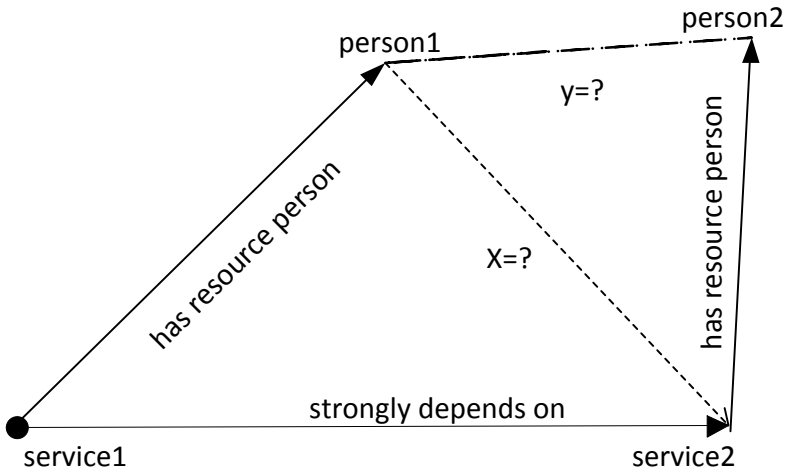


Figure 4.2: In WTC algorithm, to find new relationships, we formulate it into an edge connecting the nodes directly.

The solid lines are the relationships from the topic map. The dotted lines are the new inferences we want to make. Edges X and Y are unknown. Once they are known, a resource person of a service can right away see which other resource people he needs to contact.

The diagram shows how the WTC algorithm simplifies finding such new relationships. First, we notice that if a person is a resource person for service1, and service1 strongly depends on service2, then person1 and service2 have a

relationship X. We can say that the relationship is 'has interest in'. We can now define Y from the need that motivated the formation of the edge, communication between person1 and person2. Thus, Y can be a symmetric relationship 'needs communication with'. This means we have two weak transitive rules to connect person1 and person2.

```
is resource person for^strongly depends on^has interest in
```

and

```
has interest in^has resource person^needs communication with
```

To explain this using the graph: First it makes use of the fact that, going from person1 to service2 was taking two edges in the topic map. These are 'is resource person for', which is the reverse of 'has resource person' because we are going in the reverse direction, and 'strongly depends on'. But now, a new edge, 'has interest in', is created from them using this rule. So, a direct relationship, via edge X, is created. This edge can be explained when clicked.

Once we have X, we can traverse from person1 to service2 via edge X and then to person2 through 'has resource person'. But if that is possible, then going from person1 to person2 directly is possible, according to the transitive closure algorithm. Thus, the second rule which makes this route possible is crafted:

```
has interest in^has resource person^needs communication with
```

Now this new edge Y, 'needs communication with', is symmetric, so that we use the same relationship whether we go from person1 to person2 or vice-versa.

This person-to-person relationships can be done for both doc-responsible and resource person of services that depend on one another whether strongly or weakly using the appropriate weak transitive rules. Those rules are:

```
is resource person for^strongly depends on^has interest in
is resource person for^weakly depends on^has interest in
has interest in^has resource person^needs communication with
is doc-responsible for^strongly depends on^has interest in
is doc-responsible for^strongly depends on^has interest in
has interest in^has doc-responsible^needs communication with
```

The entire rules file for this use-case is available at the appendix.

The result of this is that when a person's name is clicked and the related entities are displayed, we see many relationships right away in the WTC algorithm that require more steps of navigations in the Houdini topic map.

### 4.1.2 Crafting more Rules

Each of the above relationships will have their inverses that will be in the rule file:

```
strongly depends on<>is strongly depended upon by
weakly depends on<>is weakly depended upon by
is doc-responsible for<>has doc-responsible
has resource person<>is resource person for
has interest in<>is interest of
needs communication with<>needs communication with
```

### 4.1.3 The pivotal relationship

The pivotal relationship is that relationship between entities that we are looking for. To have this pivotal relationship, we think of a relationship concept that encompasses all the above concepts. Normally, the pivotal relationship of a problem domain should be a universal abstraction of all the relationships that exist in the system of design.

But in this approach, the aim is to accommodate the generality of the topic map. Thus, we cannot use a relationship in the topic map as the pivotal relationship, since we have one exception, namely, "weak dependence" is not implied by "strong dependence". We therefore aim for a universal synonym. This is a universal relationship that encompasses all the relationships, but is not necessarily a useful abstraction of all of them. To have a universal synonym, we reason that all of the above are a subset of a "relationship concept". Therefore, we can have the implication rules as follows:

```
strongly depends on->is related to
weakly depends on->is related to
has doc-responsible->is related to
has resource person->is related to
has interest in->is related to
needs communication with->is related to
```

The inverse of this new concept is itself, i.e.,

```
is related to<>is related to
```

Now we are ready to form the knowledge file or facts file, the rules file, and modify the prototype for a result. Those files are available at the appendix.

Thus, by crafting appropriate rules, and changing the topic map knowledge base into a suitable facts file for the WTC algorithm, we have the benefit of easily reading a lot of inferred relationships. To reach at such conclusions using the topic map needs many navigations.

## 4.2 Results of system-description approach

The result for this approach is available at the site *http://ephrem.vlab.iu.hio.no/cgi-bin/creation.cgi*

In the system-description approach, system knowledge is directly used to create the knowledge base and the rule set of the WTC algorithm. The entities will be representing devices, services, personnel and usefulness of any sort of network items. While in most cases these will be the topics of the Houdini topic map, we are not bound to that in this approach. The chain of dependence and causal connection among the entities will be represented by the relationship concepts chosen to express the aspects we want to represent, since again we are not bound by the topic map associations. The rules will be crafted so that straight forward binary connections will give rise to inferred connections that traverse the complex system.

The difference between this and the prior approach is that we will honor neither the topic limits nor the associations of the Houdini topic map. We will allow natural implications as in English, and not worry about whether they are faithful to the topic map.

### 4.2.1 Formulation of Entity Representations

The primary idea of entity representation is to think the best representation of the network when conceptualized as a set of individual nodes connected to each other. The sources of our entities are therefore:

- Devices: Includes the servers, switches, ups and so may other network devices.
- Services: The network is about service giving. There are file-service, print-service, dns-service, dhcp-service and the like.
- Personnel: People who have specific tasks and have expertise will be represented by their names as nodes. While the people may change, the expertise and task is more or less 'invariant'.
- Usefulness of Items: like connectivity, power standby, back-up.

Since the system-description approach uses system knowledge, the level of what we know about the system is crucial. In the case of this thesis, very accurate architecture will not be used, due to the sensitivity of such information. A limited level of detail about the system around the selected use cases is shown below in table 4.2. This level of system knowledge will be used as the demonstration of the system-description approach.

Using the above system knowledge, we will use the key relationships "is provided by", "needs" and "utilizes" to formulate our facts. Services with strong dependence will be identified by "needs", while services with weak dependence will be identified by "utilizes". Since "needs" is a more concrete and strong relationship, it implies "utilizes". Thus, in the system-description approach, "utilizes" is the pivotal relationship of the problem domain. That

| Service | Function |
|---|---|
| sw* | all sw-* provide network connectivity. example: sw–248–191.uio.no, sw–248–56.uio.no |
| bsd-blader* | provide physical location and infrastructure. example: bsd–bladecenter5.uio.no |
| ypserv_* | provide nis catalog service. example ypserv_kvernbit.uio.no |
| dvergen,huldra,nissen | provide dns-lookup service |
| domain_dns-cache | provide dns-cache service. example: domain_dns-cache |
| nfs_* | provide nfs file access service.example:nfs_fronter-netapp01.uio.no |
| *_virtual | provide host classification service. examples are ipv6_connected_virtual, viktighet-2_virtual |

Table 4.2: system knowledge from IT infrastructure of UiO for system-description approach use-case.

means, "utilizes" is the universal abstraction towards which the reasoning system converges.

We will have, therefore, facts of the kind shown below. Here it is shortened for readability. The full facts file is found in the appendix.

```
aton.uio.no|needs|disk-shelf-for-aton
aton.uio.no|needs|network-connectivity-for-aton
aton.uio.no|utilizes|dns-lookup-for-aton
aton.uio.no|utilizes|nfs-file-access-for-aton

disk-shelf-for-aton|is provided by|raid-93.uio.no
network-connectivity-for-aton|is provided by|sw-248-146.uio.no
dns-lookup-for-aton|is provided by|domain_dvergen.uio.no
nfs-file-access-for-aton|is provided by|nfs_alruba.uio.no
```

The main connectivity rules will then be as follows, shortened for readability. The full rules file is available in the appendix.

```
needs^is provided by^strongly depends upon
utilizes^is provided by^weakly depends upon
strongly depends upon^strongly depends upon^strongly depends upon
weakly depends upon^weakly depends upon^weakly depends upon
```

The main advantage now is the presentation of explicit systems knowledge at the front-end. For example, consider the screen shot 4.3 shown below.

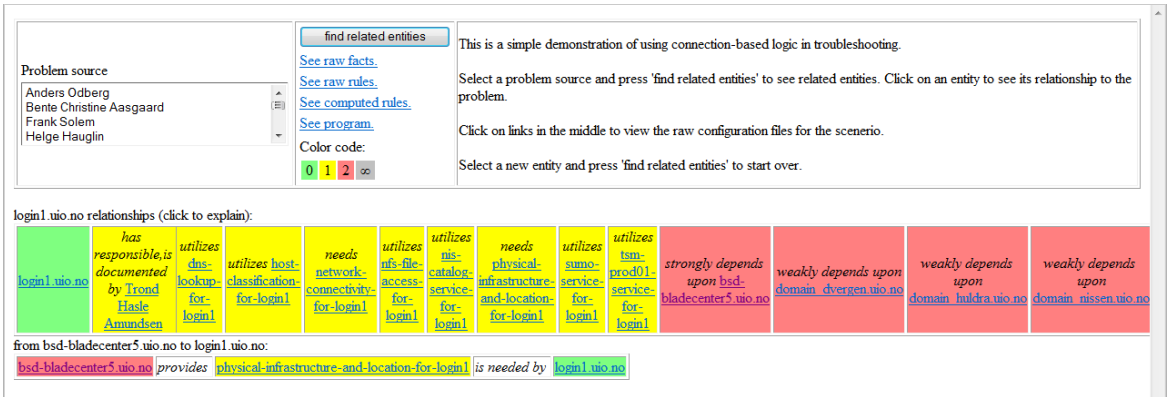


Figure 4.3: In WTC algorithm's system-description approach, the relationships in the topic map are presented as a result of concrete system facts.

We see details of login1.uio.no. Relationships like 'strong dependence' and 'weak dependence' are shown by red color code, since they are the result of more concrete system facts which are shown in yellow. One red code fact says that 'login1.uio.no strongly depends on bsd-bladecenter5.uio.no'. The explanation for it gives us the history of why this dependence exists, in concrete system terms.

In summary, we combine the benefits from both approaches and the WTC algorithm benefits us in many ways:

- The discovery of transitive relationships among services
- The discovery of new, useful person-to-person relationships, related with 'needs to communicate with'.
- The discovery of new person-to-service relationships, related with 'has interest in'.
- The explanation of the Houdini topic map associations in terms of concrete system terms

This result, which combines both benefits, is available at the site *http://ephrem.vlab.iu.hio.no/cgi-bin/general.cgi*

If we were to assign a color code to the Houdini topic map, we will have only one color code, because all we can see is binary associations between topics. But in the output of the WTC algorithm, the color codes display the different levels of inference used to present new relationships that have specific properties.

In addition to those benefits, the front end of the WTC algorithm makes learning about legacy systems very convenient. This is because, all system

knowledge records are available by clicking the 'see raw facts' button. All rules and calculated rules are also available by clicking the respective buttons, as can be seen in figure 4.3. This is something the Houdini topic maps does not give us. All we can do in the Houdini topic map is navigate through the knowledge structure.

# Chapter 5

# Discussion

This thesis work has been an endeavor to a new area not formally covered in the curriculum. Understanding Topic Maps, Understanding the WTC algorithm and dealing with the complex IT system's knowledge base at Oslo University has been a challenge. This required the understanding of mathematical principles and their implementation to practical system administration tasks. Thus, the work has been an investigative study, with open questions to answer. Yet, the questions were within the limit of knowledge management opportunities via a different kind of representation for an already represented system.

Topic Maps allow one to encode all that is and can be known about the system and simplifies navigation within that knowledge realm, serving as a binding point for all the system knowledge. Thus Topic Maps assume somewhat an exhaustive representation, at least for the intended purpose, of the represented system. This means that, what the front end presents is only what has been recorded from the outset.

The WTC algorithm does not assume an exhaustive representation, but relies on its rules for inferring more from the basic facts. The WTC algorithm records as many known facts as possible at first, and, using rules and an engine, infers consequences of the more obvious relations. This means that the front end presents some more relationships than the ones originally recorded.

As a reasoning system, the WTC algorithm is able to infer as many consequences as possible by trying to reach at the universal abstraction of all the relationships, from as many concrete facts as possible. This universal abstraction is the pivotal relationship of a problem domain. This has a consequence, however. The need for a pivotal relationship means that the WTC algorithm is narrow and can accommodate only specific issues in the system. If the pivotal relationship is the universal abstraction of all the relationships in a problem domain, all other relationships must somehow imply it. This means, we will be forced to think differently than the Houdini topic map, or drop the idea that the pivotal relationship is a universal abstraction. This latter decision will in turn limit the ability of the reasoning system in some ways, as demonstrated in the topic-map approach.

In the representation of system knowledge, there are differences as well. In the topic map, the knowledge base that contains the system knowledge is comprised of different text files. The system knowledge is in the ltm, xml or xtm files, while most of the expert knowledge also resides in the ontology. The topic map engine then presents this to the end user. Or, put another way, the meaning of the front end result about the system is not very clear without the knowledge of the ontology. It can be found from the front end that one service is "strongly dependent" on another. But what does this really mean? This meaning is the knowledge known by the expert, and is encoded in the ontology. Thus, in the Houdini topic map, knowledge of the ontology is assumed for the intended use of the front end.

In the WTC algorithm, all the concrete knowledge about the system is represented in the facts file. The facts themselves are simple and straight forward. The rules work in meaningful and logical ways, that give sense even in common semantics. Thus, the front end results are very clear at the outset. When the front end user finds that one service is "strongly dependent" on another, the meaning is usually found in the explanation, there at the front end. The front end user does not necessarily need to know any ontology, for that matter.

These stark differences could have been used, from the outset, as a basis of argument about the pros and cons of each method for knowledge engineering in system administration. But this kind of argument would have assumed the best, ideal utilization of each system. Thus, in the approach, the question was deliberately left open. This has helped us to focus on very pragmatic, detailed and real-life differences between the two. When the methods are compared on such basis, there arise practical limitations of each approach, bringing their significance to practical solutions into more rigorous test. This has been evident in the results as explained in the coming sections.

## 5.1 Generality

To accommodate all practical needs of USIT, the semantic richness of Topic Maps has been an advantage. Things like "weak dependence" and "strong dependence" has been defined in the ontology and used to define dependences of different nature. According to the Houdini topic map, these two dependences are exclusive. If service A is strongly dependent on service B, it cannot function without the service depended upon, service B, functioning. But if it is weakly dependent on B, A can still function without the service depended upon, B. Thus, strong and weak dependence are exclusive. A service that strongly depends on another cannot be said to be weakly dependent as well.

In the Houdini ontology, "strong dependence" doesn't imply "weak dependence" because of the model of knowledge. In their view of the world, "strong dependence" is a completely separate thing from "weak dependence".

Because of the semantic richness of Topic Maps, those kind of seemingly illogical definitions caused no problem and could be accommodated by the Houdini topic map. Thus, the advantage of Topic Maps is the ability to accommodate such a generality.

When we try to accommodate this generality in the WTC algorithm using the topic-map approach, there arose a need to restrict its full potential, which was achieved by making the pivotal relationship a concept of universal synonym instead of making it a universal abstraction. Thus, in practical usability in a certain field, this is one of the rigorous differences between the Houdini topic map and the WTC algorithm.

This difference shows how Topic Maps are richer in semantics, and are more general. It is understandable, for a human, to accommodate these dependence types as exclusive. Since the Houdini topic map is not a reasoning system, it can represent such ideas, ideas that the human mind can abstractly understand, into the computer system.

But as a reasoning system, the WTC algorithm cannot accommodate such semantic width. The reasoning system works by trying to reach at valid conclusions, using the rules, from the facts. When concrete, stronger kinds of relations cannot imply weaker kinds, there will be an unbridgeable break to the reasoning system. Thus, we think differently about the domain of interest in the WTC algorithm.

When thinking differently than the Houdini topic map, what we can accommodate are relationships in the domain that are subsets of the pivotal relationship. The semantic width will be limited. Concrete relationships always imply their weaker kinds. As an example, “requires” implies “might require”. This is the true essence of WTC algorithm.

On the other hand, if we decide to accommodate everything in the topic map by the WTC algorithm, we can reduce the pivotal relationship to a universal synonym. The effect of such a choice is the limitation of the reasoning system. Due to this choice, most of the explanations will not be useful for understanding the facts. But, we can compensate this by making some of the facts in no need of further explanation.

Consider the results for the topic-map approach. The explanation for those facts labeled yellow is not useful. This is indeed a limitation, since explanation of the edges is one of the advantages of the WTC algorithm. This limitation is imposed because, the pivotal relationship is not a universal abstraction of all relationships as it should normally be. Rather, it is a universal synonym, “is related to”.

Since the system is told to try to arrive at “is related to” as pivot, the explanation for all of the yellow labeled relationships is “is related to”, not particularly an insight adding explanation. But, the compensation of this limitation

was done so that all yellow colored relationships be self explanatory that need no further elaboration. This was done so that to present everything the topic map had in the WTC algorithm, i.e., for generality purposes. The reasoning system was limited, by the absence of rules, from its normal way that "strong" implies "weak."

Since "weak dependence" is weaker and more general, it should be implied by "strong dependence". This is how the WTC algorithms implication rule normally works. But, when that is so, as is done in the system-description approach, the seemingly exclusive nature of "weak dependence" and "strong dependence" in the Houdini topic map cannot be presented as they are.

Thus the fundamental difference between the two representations owing to semantic generality or richness forces us to choose. To keep what the Houdini topic map had as it is, i.e., making "weak dependence" and "strong dependence" exclusive, the WTC algorithm must be limited to use a universal synonym as pivot. Or, we must decide it is not necessary to make them exclusive, and handle the domain as we did in the system-description approach. This brings the excellence of the WTC algorithm, which is tracing connectivity, to full effect.

## 5.2 Tracing connectivity

The general issue here is that there is a subtle distinction between what is allowable in creating a topic map and what is allowable and meaningful in reasoning about a topic map. Just because relationships A and B have to be mutually exclusive in the Houdini topic map, does not mean that one has to reason about them as if they are mutually exclusive, outside the topic map. In the WTC algorithm, that we place on top of the topic map, the implication

```
strong => weak
```

has a profound effect, even though at the lower level, i.e. the Houdini topic map, it does not.

The system-description approach of the WTC algorithm demonstrates why it is not a reasonable option to separate "weak" and "strong" dependence in an exclusive manner for the WTC algorithm. This exclusive representation of the two dependence types prohibits the WTC algorithm from inferring the consequences of any kind of intermediate dependence between two entities. Thus, the WTC algorithm is allowed to reject this in the system-description approach, and it takes the liberty of its fundamental assumption into full effect. The assumption is that concrete relationships imply more abstract relationships. While concrete relationships are specific and narrow, they are within the domain of weaker, more abstract relationships. Thus, according to the WTC logic, "strong" implies "weak". If weak dependence affects only a certain functionality of a service, then strong dependence can be seen as a cumulation of many weak dependences, the sum effect of which will be affecting all functionality aspects of the depending service. Thus, we don't have unbridgeable

break between the two dependences. This means that, "weak" can be used as a pivotal relationship of the problem domain, being a universal abstraction. The result of this is that, consequences of relations can be inferred between any two entities in the system. Thus, the WTC algorithm has keenness to detail and consequence pin pointing. This is a really good quality for the WTC algorithm's superior advantage for troubleshooting and learning about legacy systems. This semantic difference is shown in figure 5.1.

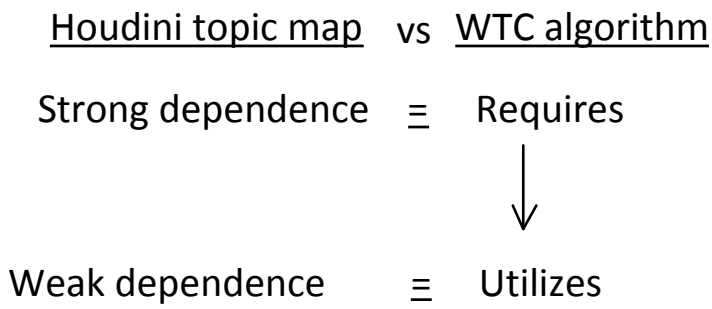


Figure 5.1: What was exclusive and unrelated, by choice of design, in the Houdini topic maps is made to have a relationship in the WTC algorithm

In troubleshooting, understanding the different connections is necessary. Thus, breaks in the connection are not useful for connectivity analysis. As to learning of legacy systems, the whole causal chain is possible to trace using this new connectivity, thus making the acquaintance with the system more friendly. No break means also that new connectivities can be traced by the reasoning system.

Using transitivity rules, it was possible to immediately see how two remote entities may depend on each other, as we saw in the example between login1.uio.no and viktighet-1_virtual.uio.no. There is no direct way to see such transitive dependences in the Houdini topic map. For the WTC algorithm, once there is an appropriate weak transitivity rule, all such dependences are presented to the user.

New connectivity were traced, as another additional example, between resource persons of two strongly dependent services. This simplifies the navigation that has to be done to reach at such connections using the Houdini topic map. While in the topic map, such navigation will require remembering the details by oneself, in the WTC algorithm the detail is vividly presented. The way we navigate between such two persons in the topic map is that, first, a person clicks a service to see who the resource person is. Then, remembering this, one of the services which has a strong dependence is clicked to see who the resource person for this other service is. For more than two services, this

would amount several back and forth clicks and remembering several names. The information is there, but it requires navigating and remembering details. But in the WTC algorithm, this is presented at one click of a resource person.

This discovery of new connectivity as a consequence of existing concrete connections is the result of utilizing graph computation and inference rules. From graph computation, we know that if we can traverse from one node to another using two or more edges, we can also have an edge that directly connects those two nodes. Then, we tell the prototype how to get that new edge by crafting the rules of inference.

If the Houdini topic map was to add this resource person–to–resource person association, it will take a lot more than what the WTC required. A new association type will be defined in the ontology, and every occurrence of this association needs to be written down with the role and scope of each member in the association. This requires the modification of the topic map itself. But in the WTC algorithm, only telling the prototype how to reach such conclusions was necessary, adding two new rules! This is a powerful advantage of the WTC algorithm.

## 5.3 Explicit system knowledge

The WTC algorithm presents explicit system knowledge to end users. First of all, the front-end can supply the raw files knowledge base to the user. Since the facts are made of simple, system descriptive sentences, this is by itself a tool for learning about legacy systems.

In addition, by making more abstract concepts the results of concrete system descriptions, as demonstrated by the results of the system-description approach, we were not only able to present what the Houdini topic map presented, but also with the added advantage of an explanation in terms of concrete system description. The following result is as an example.

```
login1.uio.no|strongly depends on|bsd-bladecenter5.uio.no .
```

This is a fact that we get from both the Houdini topic map and the WTC algorithm. While in the Houdini topic map this is all that we have, in the WTC algorithm we have the explanation of this as a result of two concrete system facts, as shown in fig 5.2:

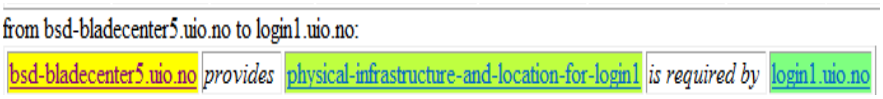


Figure 5.2: The WTC algorithm can be made to provide explicit system knowledge as explanation of more abstract relationships

# Chapter 6

# Conclusion

As knowledge representation technologies, Topic Maps and the WTC algorithm both give us the means to create a digital representation of the system knowledge. As such, both the Houdini topic map and the WTC algorithm encoding for the system achieved the goal of creating a representation where system administration staff have access to study, think and reason about the system without referring to the physical infrastructure.

It has been possible to demonstrate how the WTC algorithm can benefit us using two approaches, the first approach based on the topic map and the second approach based on the system description.

In the first approach, no direct knowledge of the underlying system was assumed. All knowledge about the system is confined to what the Houdini topic map can tell us. Based on the description of the system by the Houdini topic map, the representation of the system by the WTC algorithm provided additional benefits. These benefits include the inference of new connections between items in the system, in ways that the topic map did not depict.

In the second approach, the facts are simple system descriptions and the benefits over the Houdini topic map was that the WTC algorithm results enabled the explicit presentation of system knowledge in more concrete system terms to end users. The somewhat abstract associations of the topic map were shown to be the result of more concrete dependences of an architectural nature among system components.

In general, therefore, discovery of less direct forms of dependence, and the detailed presentation of system knowledge in explicit terms were enabled by the WTC algorithm.

The choice of two approaches was a deliberate attempt to place emphasis on the differences between the two representations at various levels. In the first approach, the differences came even when both have the same system knowledge base, namely, the topic map knowledge base. This is because, the topic map knowledge base was, in its entirety, transformed to a suitable knowledge base for the WTC algorithm. The difference came due to the presence of inference in the WTC algorithm, which allowed the latter to make a better use

of the knowledge base by providing more relationships that are inferred from the basic facts.

In the second approach, a far more fundamental difference was outlined. This is the choice of how to think about the domain. For Houdini topic maps, the choice is to view the system as discrete components related to each other, with one and only one kind of relationship between two entities. Services will depend on another either strongly or weakly, but not both. There has been a policy decision that associations cannot overlap in meaning. They can in topic maps; they can't in Houdini. The reason for this policy decision is that it makes data easier to gather and more reliable, even if many people are gathering it. The other reason is that when depicting the data, one can use one and only one depiction for each kind of edge. There is no requirement as to what kind of relationship the different types of associations may have to one another. In other words, if a concept can express an association between the chosen discrete components, it will be applied with no care as to what its relationship with other association concepts my be. What the Houdini topic map cares most is to capture the domain and enable navigational interconnection between the components. The association-to-association interaction is not a concern in the Houdini topic map.

This is not the choice of thinking in the WTC algorithm. In this representation, relationships between the associations themselves can be represented. Association-to-association interaction is a key concern in the WTC algorithm. These are the rules in this algorithm. The different associations themselves must somehow be related to each other. Therefore, in the WTC algorithm, it is not about capturing the domain and enabling navigation that matters, it is capturing the causal dependence among the components of the domain and representing those chains that matters. While topic map enables navigation and give us paths whenever available, the WTC algorithm "colors" those paths and shows us different connections as consequences of other basic connections. In other words, we are navigating in the Houdini topic map graph, but we are extracting and viewing subgraphs of choice in the WTC algorithm. The choice of which subgraphs to view is dictated by the inference rules and the pivotal relationship in the WTC algorithm.

The Houdini topic map is an excellent match for the purpose of its design, i.e., it serves as a binding point for system administration tasks. These are, among others,

- Serves as seamless connection of different information sources
- Helps personnel to study system dependencies and plan things like downtime during maintenance
- Makes it easy to view responsibility divisions among staff and knowing who to contact in contingencies

The WTC algorithm adds value. It meets all of the Houdini topic map goals, and benefits some more. Among the responsibility divisions and knowing who does what and for eventual contact, the new connections by the WTC algorithm make it more accessible than the Houdini topic map does. By adding transitive rules, new person-to-person and person-to-service_of_interest relationships are inferred.

The detailed system knowledge presentation has more substantial benefit for personnel when studying system dependence. In addition, learning about the system is made easier by the WTC presentation. In the topic map, the knowledge base files are not accessible from the front end. Even if they were, the knowledge base files in the topic map are difficult to read as they are. In the WTC presentation, general relationships acquire detailed explanation in concrete terms. In addition, the facts file, which is the knowledge base of the WTC algorithm, is accessible from the front end. The facts file is full of easy to understand system descriptions, and can enhance acquaintance with a legacy system.

Both Houdini and the WTC algorithm attempt to make Topic Maps useful by imposing limitations. Houdini limits by allowing only one kind of association per pair. The WTC algorithm limits by restricting to two kinds of reasoning, the implication and the weak transitive closure. Houdini is optimal for people coding information, but does not utilize reasoning. The WTC algorithm is optimal for understanding hidden relationships, but not for coding the initial relationships.

# Chapter 7

# Appendix

## 7.1 Details of Selected Topics from Houdini

| Selected topic | Related topics | Association type |
|---|---|---|
| login1.uio.no | 'Login-service for UiO (one of login.uio.no)' | short description |
| login1.uio.no | 'fping login1.uio.no && ssh login1.uio.no' | up-check command |
| login1.uio.no | 'unix-drift@usit.uio.no' | e-post list |
| login1.uio.no | 'M29,25 bsd-bladecenter5.uio.no' | physical location |
| login1.uio.no | 'Dell Inc. PowerEdge M600' | device description |
| login1.uio.no | Trond Hasle Amundsen | doc-responsible |
| login1.uio.no | Trond Hasle Amundsen | resource person |
| login1.uio.no | bsd-bladecenter5.uio.no, sw–248–56.uio.no, sw–248–191.uio.no | strong dependence |
| login1.uio.no | domain_dvergen.uio.no, domain_huldra.uio.no, domain_nissen.uio.no, ipv6-connected_virtual, nfs_platon.uio.no, tsm_sumo.uio.no, tsm_tsm-prod01.uio.no, viktighet-2_virtual, ypserv_radius1.uio.no | weak dependence |

Table 7.1: details for service login1.uio.no, its related topics and associations.

| Selected topic | Related topics | Association type |
|---|---|---|
| aton.uio.no | 'Fail over machine for inti, HINODE data center, astro' | short description |
| aton.uio.no | 'fping aton.uio.no && ssh aton.uio.no' | up-check command |
| aton.uio.no | 'drift@astro.uio.no' | e-post list |
| aton.uio.no | 'M24,25' | physical location |
| aton.uio.no | 'Sun-Fire-T200' | device description |
| aton.uio.no | Kjetil Kirkeba | doc-responsible |
| aton.uio.no | Unni Fuskeland, Kjetil Kirkeba, Torben Leifsen, Stein Vidar Hagfors Haugan | resource person |
| aton.uio.no | raid–93.uio.no, sw–186–105.uio.no, sw–248–146.uio.no | strong dependence |
| aton.uio.no | domain_dvergen.uio.no, ypserv_mistilteinn.uio.no, nfs_sadir.uio.no, nfs_alruba.uio.no, domain_nissen.uio.no, domain_huldra.uio.no, viktighet-3_virtual | weak dependence |

Table 7.2: details for service aton.uio.no, its related topics and associations.

## 7.2 topic-map approach

### 7.2.1 Entities of the WTC algorithm, topic-map Approach

```
Problem source
' '
' M24,25'
'Application server for fronter.com'
'Cachende DNS-server for UiO'
'Dell Inc. PowerEdge M610,417TS4J'
'Dell Inc. PowerEdge R710 7C0LM4J'
'Diskhylle for aton og eltanin'
'Failovermaskin for inti, HINODE datasenter, astro'
'HP ProLiant BL460c ,G1 447707-B21,GB874953F7'
'M11,27'
'M14,27 fronter-bladecenter3.uio.no'
'M23,12'
'M24,25'
'Mail-server for Fronter'
```

| Selected topic | Related topics | Association type |
|---|---|---|
| backup01.fronter.uio.no | 'Server for TSM-backup of fronter-netapp01' | short description |
| backup01.fronter.uio.no | 'fping backup01.fronter.uio.no && ssh backup01.fronter.uio.no' | up-check command |
| backup01.fronter.uio.no | 'fronter-core@usit.uio.no' | e-post list |
| backup01.fronter.uio.no | 'M11,27' | physical location |
| backup01.fronter.uio.no | 'Dell Inc. PowerEdge R710' | device description |
| backup01.fronter.uio.no | Kjetil Kirkeboe | doc-responsible |
| backup01.fronter.uio.no | Frank Solem, Helge Hauglin, Poal Hjelmeseth Myklebust | resource person |
| backup01.fronter.uio.no | sw-248-150.uio.no, sw-248-23.uio.no | strong dependence |
| backup01.fronter.uio.no | domain_dvergen.uio.no, domain_huldra.uio.no, domain_nissen.uio.no, nfs_fronter-netapp01.uio.no, tsm_sumo.uio.no, tsm_tsm-prod01.uio.no, viktighet-2_virtual, ypserv_kvernbit.uio.no, | weak dependence |

Table 7.3: details for service backup01.fronter.uio.no, its related topics and associations.

```
'Server for TSM-backup of fronter-netapp01'
'Sun-Fire-T200'
'drift@astro.uio.no'
'fping aton.uio.no && ssh aton.uio.no'
'fping backup01.fronter.uio.no && ssh backup01.fronter.uio.no'
'fping domain_huldra.uio.no && ssh domain_huldra.uio.no'
'fping mail-imap1.fronter.uio.no && ssh mail-imap1.fronter.uio.no'
'fping raid-93.uio.no && ssh raid-93.uio.no'
'fping web1.fronter.uio.no && ssh web1.fronter.uio.no'
'fronter-core@usit.uio.no'
'postmaster@usit.uio.no'
'storage-core@usit.uio.no'
Anders Odberg
```

```
 Bente Christine Aasgaard
 Dell Inc. PowerEdge M600,B8CBS3J
 Frank Solem
 Helge Hauglin
 Kjell Andresen
 Kjetil Kirkebo
 M29,25 bsd-bladecenter5.uio.no
 Morten Werner Forsbring
 Myklebust
 Poal Hjelmeseth
 Poal Hjelmeseth Myklebust
 Stein Vidar Hagfors Haugan
Torben Leifsen
 Trond Hasle Amundsen
 Unni Fuskeland
 aton.uio.no
 backup01.fronter.uio.no
 bsd-bladecenter5.uio.no
 domain_dns-cache.uio.no
 domain_dvergen.uio.no
 domain_huldra.uio.no
 domain_nissen.uio.no
 fronter-bladecenter3.uio.no
 hostmaster@usit.uio.no
 ipv6-connected_virtual
 login1.uio.no
 mail-imap1.fronter.uio.no
 need-up-sup-in-365_virt
 nfs_alruba.uio.no
 nfs_fronter-netapp01.uio.no
 nfs_sadir.uio.no
 power-kurs-M159
 power-kurs-M160
 power-kurs-M50-5
 power-ups-02-69
 power-ups-02-71
 power-ups-02-74
 raid-93.uio.no
 sw-186-105.uio.no
 sw-186-113.uio.no
 sw-248-141.uio.no
 sw-248-146.uio.no
 sw-248-150.uio.no
 sw-248-169.uio.no
 sw-248-170.uio.no
 sw-248-191.uio.no
 sw-248-23.uio.no
```

```
sw-248-33.uio.no
sw-248-56.uio.no
sw-248-71.uio.no
tsm_sumo.uio.no
tsm_tsm-prod01.uio.no
viktighet-1_virtual
viktighet-2_virtual
viktighet-3_virtual
web1.fronter.uio.no
ypserv_kvernbit.uio.no
ypserv_mistilteinn.uio.no
```

### 7.2.2 Facts file for the topic-map approach

```
#for short description, upcheck command, e-post list, pysical
#location and device description

aton.uio.no|has short|'Failovermaskin for inti, HINODE datasenter, astro'
backup01.fronter.uio.no|has short|'Server for TSM-backup of fronter-netapp01'
login1.uio.no|has short|'Login-service for UiO (one of login.uio.no)'
mail-imap1.fronter.uio.no|has short|'Mail-server for Fronter'
web1.fronter.uio.no|has short|'Application server for fronter.com'
raid-93.uio.no|has short|'Diskhylle for aton og eltanin'
domain_huldra.uio.no|has short|'Cachende DNS-server for UiO'

aton.uio.no|has comm|'fping aton.uio.no && ssh aton.uio.no'
backup01.fronter.uio.no|has comm|'fping backup01.fronter.uio.no
&& ssh backup01.fronter.uio.no'
login1.uio.no|has comm|'fping login1.uio.no && ssh login1.uio.no'
mail-imap1.fronter.uio.no|has comm|'fping mail-imap1.fronter.uio.no
&& ssh mail-imap1.fronter.uio.no'
web1.fronter.uio.no|has comm|'fping web1.fronter.uio.no
&& ssh web1.fronter.uio.no'
raid-93.uio.no|has comm|'fping raid-93.uio.no && ssh raid-93.uio.no'
domain_huldra.uio.no|has comm|'fping domain_huldra.uio.no
&& ssh domain_huldra.uio.no'

aton.uio.no|has epost|'drift@astro.uio.no'
backup01.fronter.uio.no|has epost|'fronter-core@usit.uio.no'
login1.uio.no|has epost|'unix-drift@usit.uio.no'
mail-imap1.fronter.uio.no|has epost|'postmaster@usit.uio.no'
web1.fronter.uio.no|has epost|'fronter-core@usit.uio.no'
raid-93.uio.no|has epost|'storage-core@usit.uio.no'
domain_huldra.uio.no|has epost|hostmaster@usit.uio.no
```

```
aton.uio.no|has phys|'M24,25'
backup01.fronter.uio.no|has phys|'M11,27'
login1.uio.no|has phys|'M29,25 bsd-bladecenter5.uio.no'
mail-imap1.fronter.uio.no|has phys|'M23,12'
web1.fronter.uio.no|has phys|'M14,27 fronter-bladecenter3.uio.no'
raid-93.uio.no|has phys|' M24,25'
domain_huldra.uio.no|has phys|M29,25 bsd-bladecenter5.uio.no

aton.uio.no|has dev|'Sun-Fire-T200'
backup01.fronter.uio.no|has dev|'Dell Inc. PowerEdge R710 7C0LM4J'
login1.uio.no|has dev|'Dell Inc. PowerEdge M600'
mail-imap1.fronter.uio.no|has dev|'HP ProLiant BL460c ,G1 447707-B21,GB874953F7'
web1.fronter.uio.no|has dev|'Dell Inc. PowerEdge M610,417TS4J'
web1.fronter.uio.no|has dev|' '
domain_huldra.uio.no|has dev|Dell Inc. PowerEdge M600,B8CBS3J

# for strong dependence
aton.uio.no|strongly depends on|raid-93.uio.no
aton.uio.no|strongly depends on|sw-186-105.uio.no
aton.uio.no|strongly depends on|sw-248-146.uio.no

backup01.fronter.uio.no|strongly depends on|sw-248-23.uio.no
backup01.fronter.uio.no|strongly depends on|sw-248-150.uio.no

login1.uio.no|strongly depends on|sw-248-191.uio.no
login1.uio.no|strongly depends on|bsd-bladecenter5.uio.no
login1.uio.no|strongly depends on|sw-248-56.uio.no

mail-imap1.fronter.uio.no|strongly depends on|sw-248-170.uio.no
mail-imap1.fronter.uio.no|strongly depends on|sw-248-141.uio.no
mail-imap1.fronter.uio.no|strongly depends on|sw-248-169.uio.no
mail-imap1.fronter.uio.no|strongly depends on|sw-248-33.uio.no

web1.fronter.uio.no|strongly depends on|sw-248-71.uio.no
web1.fronter.uio.no|strongly depends on|fronter-bladecenter3.uio.no
web1.fronter.uio.no|strongly depends on|power-kurs-M159
web1.fronter.uio.no|strongly depends on|power-kurs-M50-5
web1.fronter.uio.no|strongly depends on|power-kurs-M160
web1.fronter.uio.no|strongly depends on|power-ups-02-74
web1.fronter.uio.no|strongly depends on|sw-186-113.uio.no
web1.fronter.uio.no|strongly depends on|power-ups-02-71
web1.fronter.uio.no|strongly depends on|power-ups-02-69
raid-93.uio.no|strongly depends on|sw-186-105.uio.no

domain_huldra.uio.no|strongly depends on|bsd-bladecenter5.uio.no
domain_huldra.uio.no|strongly depends on|sw-248-56.uio.no
```

```
#for weak dependence
aton.uio.no|weakly depends on|domain_dvergen.uio.no
aton.uio.no|weakly depends on|ypserv_mistilteinn.uio.no
aton.uio.no|weakly depends on|nfs_sadir.uio.no
aton.uio.no|weakly depends on|nfs_alruba.uio.no
aton.uio.no|weakly depends on|domain_nissen.uio.no
aton.uio.no|weakly depends on|domain_huldra.uio.no
aton.uio.no|weakly depends on|viktighet-3_virtual

backup01.fronter.uio.no|weakly depends on|tsm_tsm-prod01.uio.no
backup01.fronter.uio.no|weakly depends on|domain_huldra.uio.no
backup01.fronter.uio.no|weakly depends on|domain_dvergen.uio.no
backup01.fronter.uio.no|weakly depends on|domain_nissen.uio.no
backup01.fronter.uio.no|weakly depends on|nfs_fronter-netapp01.uio.no
backup01.fronter.uio.no|weakly depends on|viktighet-2_virtual
backup01.fronter.uio.no|weakly depends on|ypserv_kvernbit.uio.no
backup01.fronter.uio.no|weakly depends on|tsm_sumo.uio.no

login1.uio.no|weakly depends on|domain_huldra.uio.no
login1.uio.no|weakly depends on|domain_nissen.uio.no
login1.uio.no|weakly depends on|domain_dvergen.uio.no
login1.uio.no|weakly depends on|ipv6-connected_virtual
login1.uio.no|weakly depends on|viktighet-2_virtual
login1.uio.no|weakly depends on|ypserv_radius1.uio.no
login1.uio.no|weakly depends on|nfs_platon.uio.no
login1.uio.no|weakly depends on|tsm_sumo.uio.no

mail-imap1.fronter.uio.no|weakly depends on|domain_dns-cache.uio.no
mail-imap1.fronter.uio.no|weakly depends on|domain_nissen.uio.no
mail-imap1.fronter.uio.no|weakly depends on|ypserv_kvernbit.uio.no
mail-imap1.fronter.uio.no|weakly depends on|domain_dvergen.uio.no
mail-imap1.fronter.uio.no|weakly depends on|need-up-sup-in-365_virt
mail-imap1.fronter.uio.no|weakly depends on|domain_huldra.uio.no
mail-imap1.fronter.uio.no|weakly depends on|viktighet-2_virtual
mail-imap1.fronter.uio.no|weakly depends on|tsm_sumo.uio.no

web1.fronter.uio.no|weakly depends on|domain_nissen.uio.no
web1.fronter.uio.no|weakly depends on|viktighet-3_virtual
web1.fronter.uio.no|weakly depends on|tsm_tsm-prod01.uio.no
web1.fronter.uio.no|weakly depends on|domain_dvergen.uio.no
web1.fronter.uio.no|weakly depends on|ypserv_mistilteinn.uio.no
web1.fronter.uio.no|weakly depends on|domain_huldra.uio.no
web1.fronter.uio.no|weakly depends on|tsm_sumo.uio.no

raid-93.uio.no|weakly depends on|viktighet-3_virtual

domain_huldra.uio.no|weakly depends on|domain_dvergen.uio.no
```

```
domain_huldra.uio.no|weakly depends on|domain_huldra.uio.no
domain_huldra.uio.no|weakly depends on|domain_nissen.uio.no
domain_huldra.uio.no|weakly depends on|ipv6-connected_virtual
domain_huldra.uio.no|weakly depends on|tsm_sumo.uio.no
domain_huldra.uio.no|weakly depends on|viktighet-1_virtual

# for documentation of service
Kjetil Kirkebo|is doc-responsible for|aton.uio.no
Kjetil Kirkebo|is doc-responsible for|backup01.fronter.uio.no
Trond Hasle Amundsen|is doc-responsible for|login1.uio.no
Trond Hasle Amundsen|is doc-responsible for|mail-imap1.fronter.uio.no
Trond Hasle Amundsen|is doc-responsible for|web1.fronter.uio.no
Anders Odberg|is doc-responsible for|domain_huldra.uio.no
Morten Werner Forsbring|is doc-responsible for|raid-93.uio.no

#for resource person
Unni Fuskeland|is resource person for|aton.uio.no
Kjetil Kirkebo|is resource person for|aton.uio.no
Torben Leifsen|is resource person for|aton.uio.no
Stein Vidar Hagfors Haugan|is resource person for|aton.uio.no

Poal Hjelmeseth|is resource person for|backup01.fronter.uio.no
Myklebust|is resource person for|backup01.fronter.uio.no
Frank Solem|is resource person for|backup01.fronter.uio.no
Helge Hauglin|is resource person for|backup01.fronter.uio.no

Trond Hasle Amundsen|is resource person for|login1.uio.no

Frank Solem|is resource person for|mail-imap1.fronter.uio.no
Bente Christine Aasgaard|is resource person for|mail-imap1.fronter.uio.no
Trond Hasle Amundsen|is resource person for|mail-imap1.fronter.uio.no

Frank Solem|is resource person for|web1.fronter.uio.no
Poal Hjelmeseth Myklebust|is resource person for|web1.fronter.uio.no

Morten Werner Forsbring|is resource person for|raid-93.uio.no
Anders Odberg|is resource person for|raid-93.uio.no
Kjell Andresen|is resource person for|raid-93.uio.no

Anders Odberg|is resource person for|domain_huldra.uio.no
Morten Werner Forsbring|is resource person for|domain_huldra.uio.no
####################
login1.uio.no$
```

### 7.2.3 Rules file for the topic-map approach

```
# canonical
has short=>has short description
has comm=>has up check command
has epost=>has e-post list
has phys=>has physical location
has dev=>has device description

is comm=>is short command for
is epost=>is epost list for
is short=>is short description for
is phys=>is physical location of
is dev=>is device description for

# inverse
strongly depends on<>is strongly depended upon by
weakly depends on<>is weakly depended upon by
is doc-responsible for<>has doc-responsible
is resource person for<>has resource person

is related to<>is related to
is determined by<>determines
is influenced by<>influences

has interest in<>is interest of
requires communication with<>requires communication with

has comm<>is comm
has epost<>is epost
has short<>is short
has phys<>is phys
has dev<>is dev

# implication

strongly depends on->is determined by
weakly depends on->is influenced by
is determined by->is related to
is influenced by->is related to
#strongly depends on->is related to
#weakly depends on->is related to
has short->is related to
has comm->is related to
has epost->is related to
has phys->is related to
has dev->is related to
```

```
is doc-responsible for->is related to
is resource person for->is related to
has interest in->is related to
requires communication with->is related to

# weak transitive rules
strongly depends on^strongly depends on^strongly depends on
weakly depends on^weakly depends on^weakly depends on
# the effect of these two rules is a service might be both strongly and
#weakly dependent on another, from different routes
#weakly depends on^strongly depends on^weakly depends on
#strongly depends on^weakly depends on^weakly depends on

# new for person to person
is resource person for^strongly depends on^has interest in
is resource person for^weakly depends on^has interest in
has interest in^has resource person^requires communication with

is doc-responsible for^strongly depends on^has interest in
is doc-responsible for^strongly depends on^has interest in
has interest in^has doc-responsible^requires communication with
```

## 7.3 system-description approach

### 7.3.1 Facts file for the system-description approach

```
aton.uio.no|requires|disk-shellf-for-aton
disk-shellf-for-aton|is provided by|raid-93.uio.no

aton.uio.no|requires|network-connectivity-for-aton
network-connectivity-for-aton|is provided by|sw-248-146.uio.no

aton.uio.no|utilizes|dns-lookup-for-aton
dns-lookup-for-aton|is provided by|domain_dvergen.uio.no
dns-lookup-for-aton|is provided by|domain_huldra.uio.no
dns-lookup-for-aton|is provided by|domain_nissen.uio.no

aton.uio.no|utilizes|nfs-file-access-for-aton
nfs-file-access-for-aton|is provided by|nfs_alruba.uio.no
nfs-file-access-for-aton|is provided by|nfs_sadir.uio.no

aton.uio.no|utilizes|host-classification-for-aton
host-classification-for-aton|is provided by|viktighet-3_virtual
```

```
aton.uio.no|utilizes|nis-catalog-service-for-aton
nis-catalog-service-for-aton|is provided by|ypserv_mistilteinn.uio.no

Kjetil Kirkebo|documents|aton.uio.no
Unni Fuskeland|is responsible for|aton.uio.no
Stein Vidar Hagfors Haugan|is responsible for|aton.uio.no
Kjetil Kirkebo|is responsible for|aton.uio.no
Torben Leifsen|is responsible for|aton.uio.no

###################################
login1.uio.no|requires|phys-infra-and-loc-for-login1
phys-infra-and-loc-for-login1|is provided by|bsd-bladecenter5.uio.no

login1.uio.no|requires|network-connectivity-for-login1
network-connectivity-for-login1|is provided by|sw-248-191.uio.no
network-connectivity-for-login1|is provided by|sw-248-56.uio.no

login1.uio.no|utilizes|dns-lookup-for-login1
dns-lookup-for-login1|is provided by|domain_dvergen.uio.no
dns-lookup-for-login1|is provided by|domain_huldra.uio.no
dns-lookup-for-login1|is provided by|domain_nissen.uio.no

login1.uio.no|utilizes|host-classification-for-login1
host-classification-for-login1|is provided by|ipv6-connected_virtual.uio.no
host-classification-for-login1|is provided by|viktighet-2_virtual.uio.no

login1.uio.no|utilizes|nfs-file-access-for-login1
nfs-file-access-for-login1|is provided by|nfs_platon.uio.no

login1.uio.no|utilizes|sumo-service-for-login1
sumo-service-for-login1|is provided by|tsm_sumo.uio.no

login1.uio.no|utilizes|tsm-prod01-service-for-login1
tsm-prod01-service-for-login1|is provided by|tsm_tsm-prod01.uio.no

login1.uio.no|utilizes|nis-catalog-service-for-login1
nis-catalog-service-for-login1|is provided by|ypserv_radius1.uio.no

Trond Hasle Amundsen|documents|login1.uio.no
Trond Hasle Amundsen|is responsible for|login1.uio.no

#####################################
mail-imap1.fronter.uio.no|requires|network-connectivity-for-mail-imap1
network-connectivity-for-mail-imap1|is provided by|sw-248-141.uio.no
network-connectivity-for-mail-imap1|is provided by|sw-248-169.uio.no
network-connectivity-for-mail-imap1|is provided by|sw-248-170.uio.no
network-connectivity-for-mail-imap1|is provided by|sw-248-33.uio.no
```

```
mail-imap1.fronter.uio.no|utilizes|dns-for-mail.imap1
dns-for-mail.imap1|is provided by|domain_dvergen.uio.no
dns-for-mail.imap1|is provided by|domain_huldra.uio.no
dns-for-mail.imap1|is provided by|domain_nissen.uio.no

mail-imap1.fronter.uio.no|utilizes|dnscache-for-mail.imap1
dnscache-for-mail.imap1|is provided by|domain_dns-cache.uio.no

mail-imap1.fronter.uio.no|utilizes|hostclassification-for-mail-imap1
hostclassification-for-mail-imap1|is provided by|need-up-sup-in-365_virt
hostclassification-for-mail-imap1|is provided by|viktighet-2_virtual

mail-imap1.fronter.uio.no|utilizes|tsm-sumo-for-mail-imap1
tsm-sumo-for-mail-imap1|is provided by|tsm_sumo.uio.no

mail-imap1.fronter.uio.no|utilizes|nis-catalog-service-for-mail-imap1
nis-catalog-service-for-mail-imap1|is provided by|ypserv_kvernbit.uio.no

Trond Hasle Amundsen|documents|mail-imap1.fronter.uio.no

Frank Solem|is responsible for|mail-imap1.fronter.uio.no
Trond Hasle Amundsen|is responsible for|mail-imap1.fronter.uio.no
Bente Christine Aasgaard|is responsible for|mail-imap1.fronter.uio.no

################################################
backup01.fronter.uio.no|requires|network-connectivity-for-backup01
network-connectivity-for-backup01|is provided by|sw-248-150.uio.no
network-connectivity-for-backup01|is provided by|sw-248-23.uio.no

backup01.fronter.uio.no|utilizes|dns-for-backup01
dns-for-backup01|is provided by|domain_dvergen.uio.no
dns-for-backup01|is provided by|domain_huldra.uio.no
dns-for-backup01|is provided by|domain_nissen.uio.no

backup01.fronter.uio.no|utilizes|nfs-for-backup01
nfs-for-backup01|is provided by|nfs_fronter-netapp01.uio.no

backup01.fronter.uio.no|utilizes|tsm-sumo-for-backup01
tsm-sumo-for-backup01|is provided by|tsm_sumo.uio.no

backup01.fronter.uio.no|utilizes|tsm-tsm-for-backup01
tsm-tsm-for-backup01|is provided by|tsm_tsm-prod01.uio.no

backup01.fronter.uio.no|utilizes|hostclassification-for-backup01
hostclassification-for-backup01|is provided by|viktighet-2_virtual
```

```
backup01.fronter.uio.no|utilizes|nis-catalog-service-for-backup01
nis-catalog-service-for-backup01|is provided by|ypserv_kvernbit.uio.no

Kjetil Kirkebo|documents|backup01.fronter.uio.no

Frank Solem|is responsible for|backup01.fronter.uio.no
Helge Hauglin|is responsible for|backup01.fronter.uio.no
Poal Hjelmeseth Myklebust|is responsible for|backup01.fronter.uio.no

####################################
web1.fronter.uio.no|requires|phys-infra-and-loc-for-web1
phys-infra-and-loc-for-web1|is provided by|fronter-bladecenter3.uio.no

web1.fronter.uio.no|requires|network-connectivity-for-web1
network-connectivity-for-web1|is provided by|sw-248-71.uio.no

web1.fronter.uio.no|utilizes|dns-for-web1
dns-for-web1|is provided by|domain_dvergen.uio.no
dns-for-web1|is provided by|domain_huldra.uio.no
dns-for-web1|is provided by|domain_nissen.uio.no

web1.fronter.uio.no|utilizes|tsm-sumo-for-web1
tsm-sumo-for-web1|is provided by|tsm_sumo.uio.no

web1.fronter.uio.no|utilizes|tsm-tsm-for-web1
tsm-tsm-for-web1|is provided by|tsm_tsm-prod01.uio.no

web1.fronter.uio.no|utilizes|hostclassification-for-web1
hostclassification-for-web1|is provided by|viktighet-3_virtual

web1.fronter.uio.no|utilizes|nis-catalog-service-for-web1
nis-catalog-service-for-web1|is provided by|ypserv_kvernbit.uio.no

Trond Hasle Amundsen|documents|web1.fronter.uio.no

Frank Solem|is responsible for|web1.fronter.uio.no
Poal Hjelmeseth Myklebust|is responsible for|web1.fronter.uio.no

################################################

huldra.uio.no|requires|phys-infra-and-loc-for-huldra
phys-infra-and-loc-for-huldra|is provided by|bsd-bladecenter5.uio.no

huldra.uio.no|requires|network-connectivity-for-huldra
network-connectivity-for-huldra|is provided by|sw-248-56.uio.no

huldra.uio.no|utilizes|dns-for-huldra
```

```
dns-for-huldra|is provided by|domain_dvergen.uio.no
dns-for-huldra|is provided by|domain_huldra.uio.no
dns-for-huldra|is provided by|domain_nissen.uio.no

huldra.uio.no|utilizes|tsm-sumo-for-huldra
tsm-sumo-for-huldra|is provided by|tsm_sumo.uio.no

huldra.uio.no|utilizes|hostclassification-for-huldra
hostclassification-for-huldra|is provided by|ipv6-connected_virtual
hostclassification-for-huldra|is provided by|viktighet-1_virtual

Anders Odberg|documents|web1.fronter.uio.no

Anders Odberg|is responsible for|web1.fronter.uio.no
Morten Werner Forsbring|is responsible for|web1.fronter.uio.no

#####################################
login1.uio.no$
```

### 7.3.2 Rules file for the system-description approach

```
#canonical
has a=>has instance
is a=>is an instance of

# inverses
has a<>is a
requires<>is required by
utilizes<>is utilized by
documents<>is documented by
is responsible for<>has responsible
is provided by<>provides

strongly depends upon<>is strongly depended upon by
weakly depends upon<>is weakly depended upon by

#implications
strongly depends upon->weakly depends upon
documents->is utilized by
is responsible for->is required by
requires->strongly depends upon
utilizes->weakly depends upon
#weak transitive
```

```
requires^is provided by^strongly depends upon
utilizes^is provided by^weakly depends upon
strongly depends upon^strongly depends upon^strongly depends upon
weakly depends upon^weakly depends upon^weakly depends upon
```

| Selected topic | Related topics | Association type |
|---|---|---|
| mail-imap1.fronter.uio.no | ' Mail-server for Fronter' | short description |
| mail-imap1.fronter.uio.no | 'fping mail-imap1.fronter.uio.no && ssh mail-imap1.fronter.uio.no' | up-check command |
| mail-imap1.fronter.uio.no | ' postmaster@usit.uio.no' | e-post list |
| mail-imap1.fronter.uio.no | 'M23,12' | physical location |
| mail-imap1.fronter.uio.no | 'HP ProLiant BL460c G1' | device description |
| mail-imap1.fronter.uio.no | Trond Hasle Amundsen | doc-responsible |
| mail-imap1.fronter.uio.no | Frank Solem, Trond Hasle Amundsen, Bente Christine Aasgaard | resource person |
| mail-imap1.fronter.uio.no | sw-248-141.uio.no, sw-248-169.uio.no, sw-248-170.uio.no, sw-248-33.uio.no | strong dependence |
| mail-imap1.fronter.uio.no | domain_dns-cache.uio.no, domain_dvergen.uio.no, domain_huldra.uio.no, domain_nissen.uio.no, need-updated-support-in-365-days_virtual, tsm_sumo.uio.no, viktighet-2_virtual, ypserv_kvernbit.uio.no | weak dependence |

Table 7.4: details for service mail-imap1.fronter.uio.no, its related topics and associations.

| Selected topic | Related topics | Association type |
|---|---|---|
| web1.fronter.uio.no | 'Applikasjonserver for fronter.com' | short description |
| web1.fronter.uio.no | 'fping web1.fronter.uio.no && ssh web1.fronter.uio.no' | up-check command |
| web1.fronter.uio.no | 'fronter-core@usit.uio.no' | e-post list |
| web1.fronter.uio.no | 'M14,27 fronter-bladecenter3.uio.no' | physical location |
| web1.fronter.uio.no | 'Dell Inc. PowerEdge M610' | device description |
| web1.fronter.uio.no | Trond Hasle Amundsen | doc-responsible |
| web1.fronter.uio.no | Frank Solem, Poal Hjelmeseth Myklebust | resource person |
| web1.fronter.uio.no | fronter-bladecenter3.uio.no, sw-248-71.uio.no | strong dependence |
| web1.fronter.uio.no | domain_dvergen.uio.no, domain_huldra.uio.no, domain_nissen.uio.no, tsm_sumo.uio.no, tsm_tsm-prod01.uio.no, viktighet-3_virtual, ypserv_kvernbit.uio.no | weak dependence |

Table 7.5: details for service web1.fronter.uio.no, its related topics and associations.

# Bibliography


[1] S. L. Kendal and M. Creen. *An Introduction to Knowledge Engineering* Springer Link DOI 10.1007/978-1-84628-667-4 ISBN 978-1-84628-475-5 (Print) 978-1-84628-667-4 (Online)

[2] Edward A. Feigenbaum, Pamela McCorduck *The fifth generation : artificial intelligence and Japan's computer challenge to the world* OSTI ID: 6089771 Addison-Wesley Pub. Co., 1983.

[3] John K Debenham Normal forms of rule-based knowledge systems DOI:10.1016/0950-7051(89)90019-1 Copyright 1989 Published by Elsevier Science B.V

[4] Reinhard Diestel Graph Theory, Fourth Edition 2010 Springer-Verlag, Heidelberg Graduate Texts in Mathematics, Volume 173 ISBN 978-3-642-14278-9

[5] E. W. DIJKSTRA A Note on Two Problems in Connexion with Graphs Numerische Mathematik , 1959-12-01, Springer Berlin / Heidelberg http://dx.doi.org/10.1007/BF01386390 DOI: 10.1007/BF01386390

[6] Lars Marius Garshol http://www.xml.com/pub/a/2002/09/11/topicmaps.html Accessed in February 2011

[7] Xiaohui Yang To Facilitate Knowledge Management Using Basic Principles of Knowledge Engineering 2009 IEEE DOI 10.1109/KESE.2009.33

[8] Randall Davis, Howard Shrobe, and Peter Szolovits What is a Knowledge Representation? Copyright 1993, AAAI AI Magazine, 14(1):17-33, 1993

[9] Ontology, entry by Thomas Gruber Encyclopedia of Database Systems Springer US, 2009 DOI:10.1007/978-0-387-39940-9

[10] Gruber, Thomas R A translation approach to portable ontology specifications Knowl. Acquis. journal vol. 5, issue 2, June1993 http://portal.acm.org/citation.cfm?id=173743.173747 DOI:10.1006/knac.1993.1008 Academic Press Ltd., London, UK

[11] Nicola Guarino,Daniel Oberle and Steen Staab Handbook on Ontologies, Second Edition What is an Ontology? pages 1-17 Springer Verlag,2009 DOI:10.1007/978-3-540-92673-3

[12] Nicola Guarino Formal Ontology, Conceptual Analysis and Knowledge Representation Int. J. Hum.-Comput. Stud., volume 43, issue 5-6, December 1995 http://portal.acm.org/citation.cfm?id=219666.219668 DOI:10.1006/ijhc.1995.1066 Academic Press, Inc., Duluth, MN, USA

[13] M. Hadzic et al Ontology-Based Multi-Agent Systems, SCI 219, pp. 3760 Springer-Verlag, Berlin Heidelberg ,2009

[14] Chandrasekaran, B. and Jorn R. Josephson, V. Richard Benjamins What Are Ontologies, and Why Do We Need Them? IEEE Intelligent Systems. 14 (1): pp. 20 - 26. 1999

[15] Standard Upper Ontology Working Group http://suo.ieee.org/ Accessed in March 2011

[16] http://www.w3.org/standards/semanticweb/ontology Accessed in February 2011

[17] M. Burgess Knowledge Management and Promises *Lecture Notes in Computer Science 2009 ;Volume 5637. s. 95-107 Oslo University College, Norway*

[18] M. Burgess CFengine Knowledge Management *CFengine Technical white paper,CFengine AS, Norway*

[19] W.B. Norton ACM Digital Library Encyclopedia of Computer Science, 4th edition: 2003 ISBN:0-470-86412-5

[20] ed. Barr, Avron, and Edward A. Feigenbaum The Handbook of Artificial Intelligence, Volume 1, page 146 1981, Stanford, Los Altos, CA: Heuris Tech Press, William Kaufmann, Inc.

[21] Steve Pepper Topic Maps Encyclopedia of Library and Information Sciences, Third Edition DOI: 10.1081/E-ELIS3-120044331

[22] Steve Pepper The TAO of Topic Maps http://www.ontopia.net/topicmaps/materials/tao.html Accessed in February 2011

[23] Lars Marios Garshol http://www.xml.com/pub/a/2002/09/11/topicmaps.html Accessed February 2011

[24] Are Gulbrandsen The XML group, Center for Information Technology Services University of Oslo, Norway Conceptual Modeling of Topic Maps with ORM Versus UML http://www.springerlink.com/content/t15rr815l2n1/#section=495866&page=1&locus=0 First accessed at the end of February 2011

[25] Houdini, Main ontology http://folk.uio.no/areg/topicmaps/HoudiniOntology/houdiniOntology.html First accessed at the end of February 2011

[26] The front-end web application at houdini http://intra.usit.uio.no/houdini/ First accessed at the end of February 2011

[27] Houdini documentation in Norwegian http://www.usit.uio.no/prosjekter/houdini/dokumentasjon/ First accessed at the end of February 2011

[28] Couch, A.L. and Burgess, M Human-Understandable Inference of Causal Relationships Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/IFIP DOI:10.1109/NOMSW.2010.5486560

[29] Alva Couch and Mark Burgess Troubleshooting with Human-readable Automated Reasoning http://www.usenix.org/event/lisa10/tech/full_papers/Couch.pdf Accessed end of January 2011

[30] http://www.cs.tufts.edu/c̃ouch/topics First accessed in February 2011

[31] ISO/IEC 13250 Topic Maps Second Edition 19 May 2002

[32] ISO 18048: Topic Maps Query Language (TMQL)

[33] ISO 19756: Topic Maps Constraint Language (TMCL)

[34] M. Burgess *Analytical Network and System Administration — Managing Human-Computer Systems* J. Wiley & Sons, Chichester, 2004

[35] Jon Orwant, Jarkko Hietaniemi, John Macdonald *Mastering Algorithms with Perl* Copyright 1999 O'Reilly & Associates, Inc.

[36] John F. Sowa *Knowledge Representation - Logical,Philosophical and Computational Foundations* Brooks/Cole: Pacific Grove, CA, 2000.

[37] http://www.jfsowa.com/logic/math.htm#Set Accessed in March 2011