Universal Information Robots
a way to the effective utilisation of cyberspace

PhD thesis

Brno, September 2006
Filip Procházka
**Declaration**

I declare that this PhD thesis is my original work and that I have written it independently. All sources and literature that I have used during the elaboration of the thesis are correctly cited with complete reference to the corresponding sources.

Supervisor: RNDr. Zdenko Staníček, Ph.D.

Signature: .................................
Abstract

This work describes a way of constructing Universal Information Robot (UIR). We begin with identifying problems preventing us from using accessible data and with identifying how to recognise interesting data. We will summarise today’s situation with data-sources and present an idea of a virtual knowledge network. We will then identify use-cases which can help us to use a tool such as UIR for the more effective utilisation of cyberspace. This will lead us on to nine abilities, which define the tools able to fulfil the use-cases presented. The architecture of such a tool – UIR - will be described. We will then discuss in greater depth three specific abilities. (1) To work with a declarative conceptual system – a basis for universal memory based on self-referential hybrid semantic networks. (2) How to formulate the algorithm enabling the ability that UIR knows what it knows and can therefore absorb information from its environment. (3) How to enrich the system with concept of an attention - energy, which is diffused through a semantic network. We will present UIR and Google vision and the Knowledge garden vision. We will conclude it with a set of open questions.

Keywords

Artificial intelligence, knowledge management, attention, Diamond Modeling Tool, cyberspace, integration, universal information robot, natural language processing, semantic networks, synapse oriented approach, data visualisation
Acknowledgments

It is almost twelve years since my journey in search of the Universal Information Robot began. These have been fruitful years and a lot of important things have happened. The process often reminds me of the way in which an anthill wakes up with the first rays of Spring sunshine. Individual ants are sensitive to warmth in different degrees. When the sun begins to shine, those ants most sensitive to warmth wake up first and begin to move around. Their movement and the sun’s warmth increase the temperature in the anthill still further. More ants are sensitised by the warmth and begin to wake up. The process continues until the whole anthill is awake.

I would like to thank my first comrades-in-ants Zdenko Staníček and Václav Račanský, the whole UIR team and all the other ants who are warming up our anthill. I would also like to thank those ants still sleeping, as they have made life more interesting.

My thanks go also to the earth, my wife Marketa and to the advaitic sun.

I hope that this work can raise the temperature a bit further. My dad David helps me a lot, as I do not want to raise the temperature with my English
# Content

1 Prologue ........................................................................................................... 9

2 Information and information cost ............................................................ 11
   2.1 Available data versus easily accessible data ........................................ 11
   2.2 Problems why data is not easily accessible ........................................ 12
      2.2.1 Problem: WHERE to ask ............................................................... 12
      2.2.2 Problem: HOW to ask ................................................................. 12
      2.2.3 FOCUSING problem .................................................................. 12
      2.2.4 The problem of CHANGE .......................................................... 12
      2.2.5 The Google way of solving these problems ................................. 13
      2.2.6 Ideal solution ............................................................................. 13
   2.3 Information and quality ......................................................................... 14
      2.3.1 Quality based on knowing .............................................................. 14
      2.3.2 Quality based on attention ............................................................ 15
   2.4 Data sources ......................................................................................... 16
      2.4.1 Kinds of data sources .................................................................. 16
      2.4.2 Data sources in the real world ....................................................... 19
   2.1 Virtual knowledge network ................................................................. 19
      2.1.1 Bottom-up approach .................................................................. 21
      2.1.2 Top-down approach ................................................................... 21

3 Main use cases .......................................................................................... 22
   3.1 Question-answer-question ................................................................. 23
   3.2 Monitoring virtual knowledge network .............................................. 25
   3.3 Adapting UIR’s behaviour ................................................................. 26
   3.4 Designing and refocusing Virtual Knowledge Network ................. 28
   3.5 Checking consistency of Virtual Knowledge Network ................... 29
   3.6 Data connecting .................................................................................. 31
   3.7 Explaining why .................................................................................. 33
   3.8 Fundamental functions summary ....................................................... 34

4 UIR - nine abilities .................................................................................... 35
   4.1 UIR abilities – a brief description ...................................................... 36
4.1.1 Ability to communicate in natural language............................. 36
4.1.2 Declarative conceptual system.............................................. 37
4.1.3 Declarative behaviour............................................................ 39
4.1.4 Ability of the cyberspace projection........................................ 40
4.1.5 Ability to plan........................................................................ 41
4.1.6 Ability to access data sources................................................. 42
4.1.7 Ability to be aware and to adapt............................................. 42
4.1.8 Ability of UIR knowing what it knows.................................... 43
4.1.9 Ability to focus attention....................................................... 44
4.2 Defining abilities and addressed problems................................. 45
4.3 Defining abilities and use-cases................................................. 47

5 Architecture for Universal Information Robots.......................... 50
5.1 Three recursive aspects............................................................. 50
5.2 Schematic overview................................................................. 50
5.3 UIR’s anatomy........................................................................... 52

6 Declarative conceptual system....................................................... 55
6.1 Representation requirements....................................................... 55
6.2 Inspirations................................................................................. 55
6.2.1 Semantic networks.................................................................. 55
6.2.2 Nanohouse............................................................................. 57
6.2.3 Diamond Modelling Tool....................................................... 58
6.3 Synapse oriented approach......................................................... 61
6.3.1 Synapse................................................................................ 62
6.3.2 Synapse scheme...................................................................... 63
6.3.3 Synapse cluster...................................................................... 64
6.3.4 Synapse cluster sequence....................................................... 65
6.4 Type system................................................................................. 65
6.4.1 Type/instance relationship...................................................... 65
6.4.2 Sub-type/super-type relation................................................. 66
6.5 Fixing point................................................................................. 66
6.5.1 Fixing point – object viewpoint.............................................. 67
6.5.2 The art of upper fixing point design - object viewpoint.............. 67
6.5.3 Fixing point – connection viewpoint....................................... 69
6.5.4 Fundamental connections.......................................................... 70
6.5.5 Synapse cluster sequence and connections................................. 72
6.6 Conceptual system and behaviour ................................................. 72

7 UIR knows what it knows............................................................. 74
7.1 Motivation .................................................................................. 74
7.2 Introduction to the split cluster problem ........................................ 75
  7.2.1 Problem specification ............................................................... 75
  7.2.2 Variable instantiation task ....................................................... 76
  7.2.3 Synapse constraints ................................................................. 77
  7.2.4 Synapse classification task ....................................................... 79
7.3 Correct and optimal solutions ...................................................... 79
  7.3.1 Solution correctness ............................................................... 79
  7.3.2 Solution optimality ................................................................. 80
7.4 Naive algorithm .......................................................................... 81
  7.4.1 Algorithm specification ........................................................... 81
  7.4.2 Algorithm explanation ............................................................ 81
  7.4.3 Algorithm drawbacks ............................................................. 82
7.5 Split cluster problem complexity ................................................. 85
7.6 The ‘islets of clarity’ algorithm .................................................... 86
  7.6.1 Deterministic steps ............................................................... 86
  7.6.2 Specific solvers ...................................................................... 87
  7.6.3 Non-deterministic steps .......................................................... 89
  7.6.4 Algorithm specification ......................................................... 89
  7.6.5 Algorithm explanation ............................................................ 90
7.7 Split cluster sliding on the identity axis ....................................... 90

8 To focus attention ........................................................................... 92
8.1 Synapse glow ............................................................................. 92
8.2 Object maker .............................................................................. 92
8.3 Connection weaver ...................................................................... 95
8.4 Attention dynamics ..................................................................... 96

9 Visions .......................................................................................... 97
  9.1 UIR and Google ........................................................................ 97
1. PROLOGUE

“Gentlemen,” he cried, “let me introduce you to the famous black pearl of the Borgias.”

Lestrade and I sat silent for a moment, and then, with a spontaneous impulse, we both broke out clapping as at the well-wrought crisis of a play. A flush of colour sprang to Holmes’s pale cheeks, and he bowed to us like the master dramatist who receives the homage of his audience. It was at such moments that for an instant he ceased to be a reasoning machine, and betrayed his human love for admiration and applause. f

“Yes, gentlemen,” said he, “it is the most famous pearl now existing in the world, and it has been my good fortune, by a connected chain of inductive reasoning, to trace it from the Prince of Colonna’s bedroom at the Dacre Hotel, where it was lost, to the interior of this, the last of the six busts of Napoleon which were manufactured by Gelder and Co., of Stepney. [DOYLE1]

Sherlock Holmes is seen as the epitome of human intellectual potential. The mainstream of research into artificial intelligence has tried to build just such a reasoning machine. There were expectations, that we could, for example, build a Sherlock Holmes for medicine who could automatically diagnose patients. These expectations, however, were bound to be disappointed, as artificial intelligence could never fulfil this role. When you go to the doctor, it is highly unlikely, that you will see „intelligent“ software helping the doctor. Would you in fact want to talk to intelligent software if you were worried about your health?

But Sherlock had a brother Mycroft. Let’s have look at Mycroft’s job:

“Mycroft draws four hundred and fifty pounds a year, remains a subordinate, has no ambitions of any kind, will receive neither honour nor title, but remains the most indispensable man in the country.

“His position is unique. He has made it for himself. There has never been anything like it before, nor will be again. He has the tidiest and most orderly brain, with the greatest capacity for storing facts, of any man living. The conclusions of every department are passed to him, and he is the central exchange, the clearinghouse, which makes out the balance. All other men are specialists, but his specialism is omniscience. We will suppose that a minister needs information as to a point which involves the Navy, India, Canada and the bimetallic question; he could get his separate advices from various departments upon each, but only Mycroft can focus them all, and say offhand how each factor would affect the other.
1. PROLOGUE

They began by using him as a short-cut, a convenience; now he has made himself an essential. In that great brain of his everything is pigeon-holed and can be handed out in an instant. Again and again his word has decided the national policy."

[DOYLE2]

Could you employ a Mycroft in a hospital? If you ask doctors, you will obtain an enthusiastic response. They don’t need a machine to do their reasoning work for them. All they need is easy access to data scattered over several data-sources to enable them to reason.

This work describes a way of constructing a virtual Mycroft – Universal Information Robot (UIR). We begin with identifying problems preventing us from using accessible data and with identifying how to recognise interesting data. We will summarise today’s situation with data-sources and present an idea of a virtual knowledge network (chapter 2). We will than identify use-cases which can help us to use a tool such as UIR for the more effective utilisation of cyberspace (chapter 3). This will lead us on to nine abilities, which define the tools able to fulfil the use-cases presented (chapter 4). The architecture of such a tool – UIR - will be described (chapter 5). We will then discuss in greater depth three specific abilities. (1) To work with a declarative conceptual system – a basis for universal memory based on self-referential hybrid semantic networks (chapter 6). (2) How to formulate the algorithm enabling the ability that UIR knows what it knows and can therefore absorb information from its environment (chapter 7). (3) How to enrich the system with concept of an attention - energy, which is diffused through a semantic network (chapter 8). We will present UIR and Google vision and the Knowledge garden vision (chapter 9). We will conclude it with a set of open questions (chapter 10).
2 INFORMATION AND INFORMATION COST

*what is a key aspect which informatics often ignore but which users rightly feel that they are missing* *why don't we use more information when taking decisions* *five problems to solve* *how can UIR be sure, that it is giving the user good service* *how to measure the effectiveness of messages sent from UIR to the user* *does UIR need to know what the user knows* *where the data can be found today* *a virtual knowledge network*

2.1 AVAILABLE DATA VERSUS EASILY ACCESSIBLE DATA

Mycroft works for the British government. Let’s focus on a more everyday place – a hospital. In a hospital you will find more than ten different data sources (hospital information system, lab system, picture archiving system – PATS etc). These systems are typically from different suppliers installed at different times. If we want information on a specific patient, we find, that this information is scattered over a number of data sources. If we want this information presented in a suitable form for browsing we soon see that this is not possible. Having data and having data, which is easily accessible are two quite different things. IT people very often overlook this distinction. They don’t see the problem and think that all is well. If, however, you talk to a hospital doctor, you will hear a different story. “We have lots of data, but do not have the time and knowledge to access this data when working with our patients. When we are treating them we typically use only the most recent records”.

In short, the information is available, but to obtain it you have to click through various systems, various forms, various menus. Obtaining information in this way is expensive, time consuming and demands computer skills.

This problem exists and is pressing. It can be illustrated metaphorically by a spam mail I found recently in my mailbox. The subject was: ‘All questions solved in shortest time’. What an attractive theme and who would not be interested in finding out more about it? The content of the message was:

Viagra $3.4
Lewitra $3.4
Mialis $3.7
Fentermine $3.2
Propecia $0.7
Ambien $2.2
Valium $0.9
Xanax $1.

Koma $2.1
Best regards,
Mr. Pharmac.

Does not this remind you of some of the products and sales material put out by a number of IT companies? You are expecting a solution, but what you really get are anti-depressant pills...

2.2 Problems why data is not easily accessible

Let us break this problem down into five sub-problems – WHERE to ask, HOW to ask, FOCUSING problem, CHANGE problem and the IDENTITY problem. The first four problems are discussed in the following paragraphs; the IDENTITY problem will be discussed later (in section 2.5).

2.2.1 Problem: WHERE to ask

Let’s assume you need an answer to your question. You feel that the answer is present somewhere in an organization’s data-sources. To obtain the answer, you must know where to ask and which information systems hold specific data. You must also have access to these systems.

2.2.2 Problem: HOW to ask

If you have identified WHERE to ask, another challenge raises its head: you must know HOW to ask. Each system works in a different way and you have to know how to interrogate it. Many queries are simple and do not cause problems (for example entering the patient’s name and selecting the last record); many others are, however, more complicated (example, give me the last positive lab test of a given patient).

2.2.3 FOCUSING problem

If you are able to interrogate the system, another problem lies in wait: how to filter out interesting information. Information comes from systems in solid blocks – all medical records, all results from lab tests etc. This is not always the way we think about information or want to have it. Sometimes we need to have specific information from different blocks to be able to decide on the steps in treating the patient, for example one reading from a lab-test together with a given characteristic of the patient.
2.2.4 The problem of Change

‘One thing is sure, things will change’. This motto is well known and widely quoted. Information systems, however, have serious problems with their evolving environment. To wire data-sources is only the beginning of the story. To sustain the function of an entire system takes a lot of energy. One small change in a single data-source can cause serious problems for the whole system. Hard coded solution support is very expensive.

2.2.5 The Google way of solving these problems

Let’s illustrate these problems in a different domain – searching web documents. Google has solved the ‘where to ask problem’. Its solution has more in common with a clever business strategy. It is, however, successful, as most people use Google as their access point to the Internet. The ‘How to ask problem’ has also been solved – you just use a list of keywords. The ‘focusing problem’ has not been solved and represents one of the system’s most serious drawbacks – the answer to your question will typically contain thousands of documents. This is a consequence of the way in which the ‘how to ask problem’ has been solved. If the input is only a list of keywords used for a fulltext search, you cannot expect better results. Focusing on the existing textual documents only and the use of a fulltext search method means that the ‘CHANGE problem’ is not in fact a problem for Google.

2.2.6 Ideal solution

The ideal solution of the ‘where to ask problem’ is a single access point to all relevant data. The core of the solution to the “how to ask problem” lies in using natural language. This way can also help solve the focusing problem. If it is easy to formulate and re-formulate a query more precisely, then the answer will not contain a lot of unnecessary information. As in life, the best format for an answer depends on the situation – written, graph, table, mind-map. The ideal solution should be able to use various answer formats. The specific situation would determine which format would be used. The problem of change can be solved by a conceptual system and behaviour specifications in natural language. If we need to adapt the system to an actual situation we just have to update its conceptual structure and/or behaviour. There is no database re-designing and re-programming.

Designing the content of an answer requires measuring the quality of the messages passed – some messages are ‘more informative’ than others. Let us now address this problem directly by discussing it within the context of a specific system – the Universal Information Robot.
2. INFORMATION AND INFORMATION COST

2.3 INFORMATION AND QUALITY

Communication between the user and UIR is done by exchanging messages. The main goal of communication is for the user to obtain accurate information. The recognition of what is accurate will have to depend to a certain extent on UIR. If not, every question from the user will have to be so detailed that the whole system will in fact be no different to a conventional information system. Some messages are better than other messages. UIR must be able to measure the ‘quality’ of a message. Let’s now focus on how this quality can be defined.

2.3.1 QUALITY BASED ON KNOWING

The quality of a message can be measured by comparing the message content with the level of knowledge of the message’s recipient. The more new facts a message contains, the better this message is. This method of evaluating message quality is part of the theory of information.

One approach is based on ‘possible worlds’. The term ‘possible world’, first coined by Leibnitz, means a maximum consistent collection of thinkable facts. One such collection is the real, ‘actual’ world. We never can say which of the possible worlds is the actual one. An intuitive reason is rather simple: If a possible world is the collection of possible facts, then the actual world is the collection of actual facts. To know which of the possible worlds is the actual one means, therefore, to know all actual facts. So unless we are omniscient we cannot identify the actual world. [MAT98]. The quality of a message is evaluated according to the number of possible worlds, possible from the recipient’s point of view, which the message itself eliminates as being able to play the role of a part of the actual world. Note that, it is necessary to take the recipient’s point of view into account, because the recipient plays a crucial role in the game.

This approach can be formalised into theories of greater practical use. More interesting, however, than elaborating on the detail of such theories, is to emphasise basic suppositions:

- The more new information a message contains, the better it is. If the size of a message is limited, the best message is that where new facts make up its whole content
- A message which excludes no possible world is useless
We can see that this kind of message quality is important. Such suppositions can, however, be problematical. Why does a good lecturer repeat information from a previous lecture? Why do we waste the size of a message on metaphors? Why is what we do not say sometimes more important than what we do say? Why do we need to see information, which we already know? We need to add another viewpoint.

2.3.2 QUALITY BASED ON ATTENTION

User knowledge is represented in neural networks and is inaccessible to UIR. When UIR sends a message to a user, it is not possible to measure the amount of information passed. UIR does not know, what is new for a user and what is superfluous. The process of creating a message has to use other methods. The user has to specify, which messages are appropriate. This specification can be divided into two parts: question and context. Question and context can usefully be seen as instructions defining how the user's attention should be directed within given data sources. The user is telling UIR on which areas or facts to focus. The quality of a message can be measured according to the user's intention expressed in terms of question and context. Question is the more concrete instruction and context the more general instruction. We will return to this point later in the chapter 4.

Let's use this perspective and have a look at suppositions from the previous chapter. The best messages are not those containing all new facts, but those messages containing facts corresponding most closely to the declared direction of attention. It doesn't matter if the user knows all the facts contained in the message. The message can play the role of a basis for subsequent thoughts. Metaphors can be seen as scaffolding for the direction of attention. In their book Mark Johnson and George Lakoff demonstrate that metaphor is not just a special device of literary expression but permeates virtually every aspect of human thought [JOLA80].

UIR does not know, what the user knows. Let us have a look at this fact in reverse. All UIR's knowledge is formalised and is in fact accessible to the user. If a message has come from a user or from another system, it is possible to measure exactly the amount of information contained. This means that UIR must be able to know what it knows (we will discuss this ability in greater depth later – chapter 7). It enables UIR to absorb information from its environment in a very usual way as we human beings do.

To summarize from the UIR’s point of view: the measurement of incoming message quality can be based on the 'knowing principle'. Outgoing message quality can be measured using the 'attention principle'.

2. INFORMATION AND INFORMATION COST

2.4 DATA SOURCES

UIR forms a bridge between the user and data sources. To be able to play this role, UIR has to access various kinds of data sources.

2.4.1 KINDS OF DATA SOURCES

Let’s now describe the typical data-sources, which UIR will meet in cyberspace. We will use mind maps for this description. The pattern of the following mind maps is summarised in figure 1:

![Pattern of data-source description](image)

Figure 1: Pattern of data-source description

2.4.1.1 INFORMATION SYSTEMS

![Information systems](image)

Figure 2: Information systems
2. INFORMATION AND INFORMATION COST

2.4.1.2 REGISTERS

Figure 3: Registers

2.4.1.3 DATA-WAREHOUSES

Figure 4: Data warehouses

2.4.1.4 WEB SERVICES

Figure 5: Web services
2. INFORMATION AND INFORMATION COST

2.4.1.5 HTML PAGES

![Diagram of HTML pages]

**Figure 6: HTML pages**

2.4.1.6 XML FILES

![Diagram of XML files]

**Figure 7: XML files**

2.4.1.7 TOOLS

![Diagram of tools]

**Figure 8: Tools**
2. INFORMATION AND INFORMATION COST

2.4.2 DATA SOURCES IN THE REAL WORLD

There are several non-technical problems connected with data sources in the real world. These problems arise from the status of the data-source. Let us summarize them:

- A data source is often owned. This means, that there is someone who must take the decision to share the information contained. This can be complicated if the data contains information not in the public domain (this is a typical case). There must be a guarantee, that data will be handled with care.
- The data source is used in day-to-day practice. This implies that usage of the data source for UIR purposes cannot cause performance problems in day-to-day use. This problem is also a partial motivation for building data warehouses.
- The data source (typically information systems) is administrated by someone else. The ideal way would be to be able to access the data-source without the need for system administrators. The need to adapt a data source to communicate with a third party system often causes problems.

Any system proposed cannot ignore these problems and must therefore take them into account.

2.1 VIRTUAL KNOWLEDGE NETWORK

There are various technical kinds of data-source. Data-sources are organised using various conceptual systems. Variety is the spice of life but not in this case. If we need to use information from data-sources, we have to unify. There are two possible ways of doing this:

(a) unify the world
(b) unify our way of looking at the world

The first solution was and still is often used in practice but is very expensive and often doesn’t work. This ‘rip and replace’ strategy tries to unify information in one big system and hopes that from this point the world will be unified. This situation, however, doesn’t happen. There are always domains, which occur in an evolving situation and which were not included in ‘big system’ analyses. You cannot also ‘rip and replace’ systems belonging to partners. There is and will, I hope, always be variety in the world.
The second way of solving the variety problem is to unify the view on the world. There should of course be several unified views and not just one unified world. This second solution is much more promising as is confirmed by today’s success of integration platforms and composite applications. Commonly used integration platforms concentrate on business processes going through a number of information systems. UIR’s way is focused on data.

We have to mask the variety of data-sources and provide unified access. This task can be seen as wiring up a virtual knowledge network through data sources. The user can connect to this network and use UIR as a guide.

In the following figure we can see a model situation. There are two data-sources. The first is the geographical information system containing information about devices, their location, purchase dates etc. The second is the intranet containing information about persons, their functions, rooms where they work etc. We can wire a virtual knowledge network through these data-sources and use UIR as a guide. We can ask UIR questions and obtain answers to questions, which necessitate the combining of two separate data-sources.

![Figure 9: Example of the Virtual Knowledge Network](image)

If we begin to think about unification, we meet one big problem – how to recognise that two specific pieces of information, each from a different data source, refer to the same object. For example, how to identify that room and location are in certain situations the same. This is the fifth problem, why data are very often not easily accessible. We will call this problem **Identity problem**. We will focus
A virtual knowledge network can be formed using either a bottom-up or top-down approach.

2.1.1 Bottom-up approach

The principle of this approach is ‘describe what we have and connect what belongs together’.

We have to describe the conceptual systems used in the relevant data-sources and connect these systems together. This approach has several drawbacks. The network copies the conceptual structure of the data-sources. The conceptual systems of many data sources are not very well designed and during their lifetime updates bring tower of Babylon to mind. Some relationships are not present in data and are hidden in program code. They typically focus more on the details of implementation than on a clear conceptual structure. If we put together several unclear conceptual structures, we cannot expect a clear result for the user. This approach ignores the use of a network; we only put together what we have.

2.1.2 Top-down approach

The principle of this approach is ‘design what we want and map what we have’.

This approach begins with designing the basic conceptual scheme of the whole domain. It is not very easy, because we must go to the heart of the matter. Such conceptual schemes are straightforward and quite abstract. They determine the direction, in which we will look at the data. (An example of this scheme used in the domain of banking can be found in [KASP01]). Data-sources are mapped on this straightforward scheme. Mapping is often not an easy task and demands more complicated techniques. But it is more promising way of lowering the cost of a piece of information.

A virtual knowledge network is just a structure. To utilise it properly we need a guide through the network. Let’s focus on how UIR can play the role of this guide and how it can serve us.
3 Main Use Cases

* services which UIR can fulfil * stories from UIR’s life * answering * monitoring * adapting * educating * connecting data * answering the question ‘why’ * identification of UIR’s fundamental functions *

Sir Arthur Conan Doyle wrote 60 stories about Sherlock Holmes. Mycroft Holmes appeared only in two of them. If we begin to think about Mycroft Holmes’s life, we can imagine some interesting stories, which Arthur Conan Doyle perhaps intended to write but never did. However, this is not the correct place to describe them in their entirety. We will use quotations from Sherlock’s life, then focus on the nub of the stories and analyse resulting use-cases. We can, of course, find number of possible use-cases. We will cover those most important from the user’s point of view in this chapter.

The overall situation can be illustrated using following picture:

![Figure 10: User, UIR and data-sources](image)

There are three elements: user, UIR and data-sources. Arrows from left to right represent requests, arrows from right to left represent answers to these requests. UIR works with virtual knowledge networks capturing the knowledge of a given domain. This network is mapped on data-sources. In the following paragraphs we will use a schematic picture of this situation:
3. MAIN USE CASES

Figure 11: User, UIR and data-sources – schematic view

3.1 QUESTION-ANSWER-QUESTION

The Adventure of the Golden Pince-Nez 1904:

I may have remarked before that Holmes had, when he liked, a peculiarly ingratiating way with women, and that he very readily established terms of confidence with them. In half the time which he had named he had captured the housekeeper’s good will, and was chatting with her as if he had known her for years.

“Yes, Mr. Holmes, it is as you say, sir. He does smoke something terrible. All day and sometimes all night, sir. I've seen that room of a morning -- well, sir, you'd have thought it was a London fog. Poor young Mr. Smith, he was a smoker also, but not as bad as the Professor. His health -- well, I don't know that it's better nor worse for the smoking.”

“Ah!” said Holmes, “but it kills the appetite.”

“Well, I don't know about that, sir.”

“I suppose the Professor eats hardly anything?”

“Well, he is variable. I'll say that for him.”

“I'll wager he took no breakfast this morning, and won’t face his lunch after all the cigarettes I saw him consume.”

“Well, you're out there, sir, as it happens, for he ate a remarkable big breakfast this morning. I don't know when I've known him make a better one, and he's ordered a good dish of cutlets for his lunch.”

This dialogue illustrates the question-answer-question situation. Let’s list several examples how Mycroft can be useful.

*Ask Mycroft and obtain the answer. If the answer is not satisfactory refine your question and ask again.*

Mycroft not only knows facts about the world but also how you are accustomed to seeing them. You can thus use shortcuts. Mycroft can easily guess what you mean.
Let us identify fundamental functions of this use-case:

- To understand the question
- To be aware of context
- To respect access rights
- To find out where facts for answering questions can be found
- To connect facts from various data sources
- To decide how the answer will be presented
- To present the answer

We can see that such a simply formulated use-case requires a lot of sophisticated functionality. To solve the 'how to ask problem' we must also handle queries formulated in natural language. We are, however, far from a solution, which understands freely used natural language. We can use UIR in a restricted domain – for example in project management or in health care. This situation can be compared, for example, to machine translation. The complex problem of machine translation has not yet been solved and today's solutions are very funny caricatures [WIKI06]. An example is the project of Wikipedia machine translation from English into Czech. Systems restricted to a specific domain are, however, very successful – TAUM (the translation of aviation manuals) or METEO (the translation of weather reports). The use of context is very important from the user's view and also helps UIR to analyze questions properly. The user can use context to specify the actual focus of his attention. For example, a user is now concentrating on the main information about a patient, the results of cardiological examinations and any similarities with his relatives' case histories. Context should ideally be specified in natural language. There is a need to be able to handle words such as mainly, with respect to, quite etc. The context of a dialogue can be used to handle the problem of disambiguation. Respect of access rights is a necessity. This prob-
lem can be seen as a special usage of context. Some facts you don’t see because you are not allowed to (access rights), some facts you don’t see because they are not relevant for you at that moment (for example you are now asking in your role as a doctor and not as a scientist). Facts necessary to compose an answer are typically not stored in UIR but in various data-sources. UIR must be able to divide a question into sub-questions, interrogate appropriate data-sources for answers to sub-questions and from sub-answers to compose a final answer. Inconsistencies can happen when putting sub-answers together; information from two data-sources can be contradictory. UIR should be able to solve these inconsistencies or warn a user that some part of the answer is contradictory. Finding an answer is not the end of the matter. UIR has to decide how to present the answer. There are several ways of doing this which can also feature together: list, table, graph, form, etc. After designing a format for an answer, this answer will be presented and can be used to refine the question itself. Parts of the present answer can also be used as a context in subsequent questions.

3.2 Monitoring virtual knowledge network

The Adventure of the Three Garridebs, 1924

“I was wondering, Watson, what on earth could be the object of this man in telling us such a rigmarole of lies. I nearly asked him so — for there are times when a brutal frontal attack is the best policy — but I judged it better to let him think he had fooled us. Here is a man with an English coat frayed at the elbow and trousers bagged at the knee with a year’s wear, and yet by this document and by his own account he is a provincial American lately landed in London. There have been no advertisements in the agony columns. You know that I miss nothing there. They are my favourite covert for putting up a bird, and I would never have overlooked such a cock pheasant as that.

Sherlock Holmes monitors several data-sources. One of these data sources is newspapers. Let us list several examples to see how monitoring can be useful.

Send me an email if one of my patients with a specific diagnosis is to be operated on in our hospital.

Scan the company register and send me an e-mail if a given person shows up as shareholder of any private company.

Tell me when the number of my unfinished tasks exceeds ten.
Tell me whether you think that there are inconsistencies in the document base.

The first phase of the use-case is to specify what to monitor. In the second phase UIR monitors given data-sources and alerts if something interesting occurs.

Fundamental functions:

- To understand what to monitor and when to alert
- Periodically check if a given event happened in data sources
- To recognise and understand data-source changes
- Present events to the user

Browsing periodically through data sources to check whether there is something of interest to us is not particularly attractive work. It is far better to use a tool such as UIR to do this for us. We have to inform UIR about events in data-sources, which interest us. The occurrence of some events will trigger certain defined actions. This action could be the monitoring of other parts of the data-sources. There can be a whole network of rules scanning or guarding selected information.

Only a small part of the data-sources can be adapted to send UIR messages about their updates. A situation, which is more likely, is that UIR will have to periodically interrogate the data-source and recognize if there is something new and interesting. The need for an ‘answer or alert’ presentation is similar to in question-answer-question use-case.

3.3 Adapting UIR’s behaviour

Valey of Fear 1904:

„Jean Baptiste Greuze,“ Holmes continued, joining his finger tips and leaning well back in his chair, „was a French artist who flourished between the years 1750 and
27

1800. I allude, of course to his working career. Modern criticism has more than indorsed the high opinion formed of him by his contemporaries."

The inspector’s eyes grew abstracted. “Hadn’t we better --” he said.

“We are doing so,” Holmes interrupted. “All that I am saying has a very direct and vital bearing upon what you have called the Birlstone Mystery. In fact, it may in a sense be called the very centre of it.”

MacDonald smiled feebly, and looked appealingly to me. “Your thoughts move a bit too quick for me, Mr. Holmes. You leave out a link or two, and I can’t get over the gap. What in the whole wide world can be the connection between this dead painting man and the affair at Birlstone?”

“All knowledge comes useful to the detective,” remarked Holmes. “Even the trivial fact that in the year 1865 a picture by Greuze entitled La Jeune Fille a l’Agneau fetched one million two hundred thousand francs -- more than forty thousand pounds -- at the Portalis sale may start a train of reflection in your mind.”

Sherlock’s thought process has often been too quick for others to follow. He therefore had to be careful how he phrased things and to pay particular attention to things, which might not be clear. Examples from UIR’s life can be:

Please pay more attention to financial aspects in projects.

Company ownership is not of interest now.

Let’s use the new version of risk profiling rules of our company.

When we communicate with UIR (first phase) we realize, that we have to ask UIR to adapt its behaviour to our actual needs (second phase). Then we continue communication (third phase).
Fundamental functions:

- To describe UIR's behaviour
- To analyse the request to change behaviour
- To be able to adapt behaviour on-line

This use-case requires that UIR's behaviour not be hardwired in program code but be expressed declaratively. This requirement is very similar to the principle of Business Rules Engines [BRC06], [HKMS94] – not to hardwire business logic into program code but to express it declaratively in a format, which business people can understand and update. The key principle is that UIR's behaviour be modelled in the very same way as information from data-sources. This enables us to query UIR's behaviour, monitor it etc.

3.4 Designing and refocusing Virtual Knowledge Network

"My practice has extended recently to the Continent," said Holmes, after a while, filling up his old brier-root pipe. "I was consulted last week by Francois Le Villard, who, as you probably know, has come rather to the front lately in the French detective service. He has all the Celtic power of quick intuition, but he is deficient in the wide range of exact knowledge which is essential to the higher developments of his art. " ... "He has considerable gifts himself. He possesses two out of the three qualities necessary for the ideal detective. He has the power of observation and that of deduction. He is only wanting in knowledge; and that may come in time. He is now translating my small works into French."

"Your works?"

"Oh, didn't you know?" he cried, laughing. "Yes, I have been guilty of several monographs. They are all upon technical subjects. Here, for example, is one 'Upon the Distinction between the Ashes of the Various Tobaccos.' In it I enumerate a hundred and forty forms of cigar-, cigarette-, and pipe-tobacco, with colored plates illustrating the difference in the ash. It is a point which is continually turning up in criminal trials, and which is sometimes of supreme importance as a clue. If you can say definitely, for example, that some murder has been done by a man who was smoking an Indian lunkah, it obviously narrows your field of search."
Sherlock has been extending his knowledge systematically with new domains. This is a necessity not just for a detective but also for anyone who wants to make use of cyberspace. Examples can be:

Let’s expand our network with a domain on drug contraindications.

Let’s simplify our view of customers’ classifications

This use-case is about communication with UIR about the conceptual system (meta-data) describing a new data source or about changes in the existing conceptual system.

Fundamental functions

- To describe given knowledge in a given domain
- To connect given domain with other domains
- To map domain knowledge on specific data sources

A knowledge network is not something stable and fixed. It must evolve along with the user’s requirements. This means that concepts proposed can be found unsuitable, some important relationships can be discovered as missing or a new data source becomes available for addition to the knowledge network. To do this, we need to describe the concepts and rules (meta-data) of a given domain and to update UIR’s knowledge. This cannot be done by re-designing its database and by re-programming certain functions. UIR looks at meta-data in the same way as data. This implies, that UIR is also able to work with meta-meta-data, etc (see [ODELL95]). The conceptual model update must sometimes handle synonymy (one thing is denoted by more names) and homonymy (one name denotes more things).
3.5 Checking consistency of Virtual Knowledge Network

The Adventure of the Priory School, 1904:

„Let us continue our reconstruction. He meets his death five miles from the school -- not by a bullet, mark you, which even a lad might conceivably discharge, but by a savage blow dealt by a vigorous arm. The lad, then, HAD a companion in his flight. And the flight was a swift one, since it took five miles before an expert cyclist could overtake them. Yet we survey the ground round the scene of the tragedy. What do we find? A few cattle tracks, nothing more. I took a wide sweep round, and there is no path within fifty yards. Another cyclist could have had nothing to do with the actual murder. Nor were there any human footmarks."

„Holmes," I cried, „this is impossible."

„Admirable!“ he said. „A most illuminating remark. It IS impossible as I state it, and therefore I must in some respect have stated it wrong. Yet you saw for yourself. Can you suggest any fallacy?“

Data consistency is an urgent problem in cyberspace. Areas where inconsistency appears are very interesting and, if examined, give us a lot of serious evidence. Examples:

Show me all patients' records where clinical records are not consistent with health insurance records.

Show me all actions in conflict with our agreements on the business plan.

Tell me about all badly planned projects.

During the first phase we teach UIR consistency rules. UIR can check these rules against data-sources (phase two). It will then report results (phase three)

Fundamental functions

• To formulate consistency rules
3. MAIN USE CASES

- To be able to find inconsistencies in data sources
- To report inconsistencies
- To be able to solve inconsistencies

If we are able to integrate data from a number of different data sources, the problem of data consistency then immediately appears. Data sources are typically built in isolation, so if we put data sources together, there can be lot of redundancy. Redundancy goes side by side with inconsistency. There is no general solution, which is domain-independent. Domain knowledge is needed to specify rules for particular situations and to establish which information has precedence (who is right). This knowledge can be expressed by rules. Some rules for detecting inconsistencies along with other rules are able to solve this problem (i.e. preference given to information from a certain data-source and the ability to express the unreliability of information). The presentation of inconsistencies is also very important and there are various ways of doing this: (1) as they happen or (2) by generating reports.

3.6 DATA CONNECTING

The Five Orange Pips 1891

Did you remark the postmarks of those letters?”

“The first was from Pondicherry, the second from Dundee, and the third from London.”

“From East London. What do you deduce from that?”

“They are all seaports. That the writer was on board of a ship.”

“Excellent. We have already a clue. There can be no doubt that the probability—the strong probability—is that the writer was on board of a ship. And now let us consider another point. In the case of Pondicherry, seven weeks elapsed between the threat and its fulfilment, in Dundee it was only some three or four days. Does that suggest anything?”

“A greater distance to travel.”

“But the letter had also a greater distance to come.”

“Then I do not see the point.”

“There is at least a presumption that the vessel in which the man or men are is a sailing-ship. It looks as if they always send their singular warning or token before them when starting upon their mission. You see how quickly the deed followed the sign when it came from Dundee. If they had come from Pondicherry in a steamer they would have arrived almost as soon as their letter. But, as a matter of fact, seven weeks elapsed. I think that those seven weeks represented the difference between the mail-boat which brought the letter and the sailing vessel which brought
How did you trace it, then?"

He took a large sheet of paper from his pocket, all covered with dates and names.

“I have spent the whole day,” said he, “over Lloyd’s registers and files of the old
papers, following the future career of every vessel which touched at Pondicherry in
January and February in ’83. There were thirty-six ships of fair tonnage which
were reported there during those months. Of these, one, the ‘Lone Star,’ instantly
attracted my attention, since, although it was reported as having cleared from Lon-
don, the name is that which is given to one of the states of the Union.”

“Texas, I think.”

“I was not and am not sure which; but I knew that the ship must have an American
origin.”

“What then?”

“I searched the Dundee records, and when I found that the barque ‘Lone Star’ was
there in January, ’85, my suspicion became a certainty. I then inquired as to the
vessels which lay at present in the port of London.”

“Yes?”

“The ‘Lone Star’ had arrived here last week.

When separated data are connected together, interesting things can appear. It
allows us to piece the puzzle together. Typical examples piece together different
viewpoints:

Connect medical treatments to health insurance records.

Connect labour hours to financial outgoings.

During the first phase we teach UIR connecting rules. UIR can use these rules
against data-sources (phase two). It will then report the results to us (phase
three).
3. MAIN USE CASES

Fundamental functions

- To describe connecting rules
- To use connecting rules for different parts of virtual knowledge networks

Related data-sources contain information about the same object but from different points of view. More formally, the conceptual schemes for different data sources differ. Possible examples here are a data-source containing information about patients and a data source containing health-care insurance records. There is a relationship between these two viewpoints. This relationship is not, however, explicit and has to be described using rules.

3.7 EXPLAINING WHY

The Disappearance of Lady Frances Carfax, 1911

„But why Turkish?“ asked Mr. Sherlock Holmes, gazing fixedly at my boots. I was reclining in a cane-backed chair at the moment, and my protruded feet had attracted his ever-active attention.

„English,“ I answered in some surprise. „I got them at Latimer’s, in Oxford Street.“

Holmes smiled with an expression of weary patience.

„The bath!“ he said; „the bath! Why the relaxing and expensive Turkish rather than the invigorating home-made article?“

„Because for the last few days I have been feeling rheumatic and old. A Turkish bath is what we call an alterative in medicine—a fresh starting-point, a cleanser of the system."

„By the way, Holmes,“ I added, „I have no doubt the connection between my boots and a Turkish bath is a perfectly self-evident one to a logical mind, and yet I should be obliged to you if you would indicate it."

„The train of reasoning is not very obscure, Watson,“ said Holmes with a mischievous twinkle. „It belongs to the same elementary class of deduction which I should illustrate if I were to ask you who shared your cab in your drive this morning."

„I don’t admit that a fresh illustration is an explanation,“ said I with some asperity.

„Bravo, Watson! A very dignified and logical remonstrance. Let me see, what were the points? Take the last one first—the cab. You observe that you have some splashes on the left sleeve and shoulder of your coat. Had you sat in the centre of a hansom you would probably have had no splashes, and if you had they would certainly have been symmetrical. Therefore it is clear that you sat at the side. Therefore it is equally clear that you had a companion."

33
3. MAIN USE CASES

„That is very evident.”
„Absurdly commonplace, is it not?”
„But the boots and the bath?”
„Equally childish. You are in the habit of doing up your boots in a certain way. I see them on this occasion fastened with an elaborate double bow, which is not your usual method of tying them. You have, therefore, had them off. Who has tied them? A bootmaker—or the boy at the bath. It is unlikely that it is the bootmaker, since your boots are nearly new. Well, what remains? The bath. Absurd, is it not?

How can others rely on Sherlock Holmes? He has always been able to explain in answer to a question how he came to his conclusion. It is the same with UIR. Examples:

*Why is this insurance claim a fraud?*

*Why is this project inconsistent?*

When you communicate with UIR (phase one), you want to know, why the answer is as it is. Then you ask UIR why (phase two).

Fundamental functions:

- To be aware of the mental operations leading to answer
- To be able to present reasons

This is very similar to Sherlock’s problems. Results have to be backed up with facts otherwise they are not taken seriously. A software tool such as UIR has to be understood as a shortcut – doing things that the user could do himself but has no time and patience for. At any moment it must be possible for the user to check the workings of the shortcut. This feature will cause the user to trust the system.
3. MAIN USE CASES

3.8 FUNDAMENTAL FUNCTIONS SUMMARY

The list of mentioned fundamental functions is:

(1) To understand the question
(2) To be aware of context
(3) To respect access rights
(4) To find out where facts for answering questions can be found
(5) To connect facts from various data sources
(6) To decide how the answer will be presented
(7) To present the answer
(8) To understand what to monitor and when to alert
(9) Periodically check if a given event happened in data sources
(10) To recognise and understand data-source changes
(11) Present events to the user
(12) To describe UIR's behaviour
(13) To analyse the request to change behaviour
(14) To be able to adapt behaviour on-line
(15) To describe given knowledge in a given domain
(16) To connect given domain with other domains
(17) To map domain knowledge on specific data sources
(18) To formulate consistency rules
(19) To be able to find inconsistencies in data sources
(20) To report inconsistencies
(21) To be able to solve inconsistencies
(22) To describe connecting rules
(23) To use connecting rules for different parts of virtual knowledge networks
(24) To be aware of the mental operations leading to answer
(25) To be able to present reasons

This list is may be not fully complete. But can we imagine a guide in cyberspace without any of listed function?
4 UIR - NINE ABILITIES

*abilities UIR needs to satisfy identified use-cases * communication in natural language * declarative conceptual system * declarative behaviour * cyberspace projection * accessing data-sources * awareness of actual situation and adaptation * UIR knows what it knows * focusing attention * how identified abilities help to solve the five problems addressed * how identified abilities are involved in identified use-cases *

In the last chapter we discussed UIR's use-cases – processes, which make UIR useful from the user’s perspective. Let us now look at matters from the constructor's point of view. Which basic abilities have to be designed into UIR and implemented to achieve behaviour, which supports the use-cases described? These abilities can also be seen as the defining properties of systems implementing the concept of Universal Information Robot.

4.1 UIR ABILITIES – A BRIEF DESCRIPTION

4.1.1 ABILITY TO COMMUNICATE IN NATURAL LANGUAGE

This ability is necessary to solve the ‘how to ask problem’. The most user-friendly way to communicate with UIR is to use natural language. We have already seen that there are various requests, which can be passed to UIR in this way (asking a question, updating concepts, expressing rules etc.). The ability to answer in natural language is interesting but not essential, as there are better ways of presenting answers (mind-maps, tables, lists, forms, charts, etc).

UIR has two levels of understanding a natural language message. The first method uses so called identification expressions (IDEXes) for expressing the exact and unambiguous identification of objects. IDEXes have been proposed by Prochazka, for a detailed explanation see [PRO03]. IDEXes can be understood as a kind of a conceptual query language [BLOHAL96].

Any identification expressed can be used as a query – we want to know all objects identified by a given IDEX.

Example:

Task with properties:
* is in category urgent
* is assigned to person with properties:
  ** it is senior consultant

This IDEX identifies an urgent task assigned to a senior consultant. Using the
4. UIR - Nine Abilities

IDEX as a query, UIR can identify all tasks meeting the criteria of this IDEX. Another usage of IDEX is as an update request – we identify an object and inform UIR about it. Using the above example, we inform UIR of an urgent task assigned to a senior consultant without giving UIR the name of the task or the consultant. The information, which we have given UIR, is quite general. Despite this, UIR is able to commit this to memory. We can add more detailed information later.

We will discuss this in greater depth in the following chapters. An IDEX is a very powerful apparatus, but its biggest advantage – un-ambiguity - is taken from another point of view its most serious drawback. You must be precise and use specific syntax.

The second method is a mirror image of the first. It is no longer un-ambiguous but represents a user-friendly way of querying. We can briefly describe this method using the same example as for IDEX. We need only to use key words: task, urgent, senior consultant. UIR is able to reason from its conceptual model that urgent is a property of task, that senior consultant is a category of employee and that there is a relationship between task and employee, which describes ‘persons assigned to a given task’. UIR uses this knowledge to construct the query. We will call this method connection weaving.

The problem of ambiguity, however, arises if there is another relationship between the person and the task – for example ‘persons banned from a given task’. From the input (task, urgent, senior consultant) UIR will not be able to differentiate between these two situations.

We can also use the whole sentence: Please, show me all urgent tasks of senior consultants. UIR will ignore all the words that it doesn't know and will only pick out key points (urgent, tasks, senior consultant). This method looks very primitive but has demonstrated surprisingly good results in everyday use. The method also appears to work well with voice recognition techniques. We should remember how we listen to a foreign language where we use exactly the same method – pick out words we know and from the context weave together the meaning.

Like IDEXes, connection weaving can also be used to inform UIR.

These two methods are used for object identification. UIR has to be able to identify operations too. It has to recognise if a given request is a query, a new rule etc. This is primarily based on pattern matching techniques and is covered by an algorithm called operation parser (see section 5.3). There are a number of more accurate grammars (for example for expressing rules). Operation parser applies these grammars to the appropriate part of the input. The pattern matching technique is very useful for this purpose. More substantial reasons can be found in [ALLEN95].
4.1.2 Declarative conceptual system

We have to specify the initial conceptual system of a given domain and allow this conceptual system to evolve according to the user’s needs. Let us now describe the structure of the conceptual system.

The corner stones are types (examples: person, task). Each individual is an instance of one type only in UIR’s conceptual system. Types are connected to a sub-type/super-type relationship (examples: person-employee, process-task). Individuals can be assigned to one or more categories (examples: senior consultant, urgent task). Each category is assigned to a given categorization (senior consultant → person qualification, urgent task → task status). The properties of individuals can be described by using attributes (example: salary, task description). We use attributes to describe properties that cannot be modelled using categories. For example, to model the property salary we might have to have an infinite number of categories. Individuals can also be placed in time (example: date of birth, task deadline). We can define connections between individuals together with the connection exact and unambiguous semantics (example: persons allocated to given task).

An example of a conceptual system from the testing domain of UIR is illustrated in the following picture.

![Figure 12: Example of a conceptual system](image-url)
4. UIR - Nine Abilities

The assumption that we can fully analyse a user's needs from a given domain is an illusion [JAKTHO03], [STA03], [STPR04a]. A user is able to specify exactly what he needs only when he first begins to use the system. This will inevitably lead to the need to be able to update the conceptual system on the run. UIR on the other hand looks at the conceptual system in the same way as at other data. It is therefore possible to change this data on the run. No database redesign and reprogramming are needed.

4.1.3 Declarative behaviour

The behaviour of such a system has to be as flexible as its conceptual system. The method proposed for specifying UIR's behaviour is to use rules formulated in slightly formalised (using IDEXes and rule grammar) natural language.

Example ($B$ denotes a variable):

For each building $B$ with properties:
* is assigned to category public building
* is assigned to category floodplain
must hold:
there exists document with properties:
* it belongs to documents of given building $B$
* is assigned to category evacuative plan
* its status is actual
if not report message:
"Public building situated in floodplain must have an actual evacuative plan".

Behaviour influenced by rules can be divided as follows:
- Reaction to certain events – activity rules
- The deduction of new properties – implication rules
- The recognition and handling of inconsistencies – consistency rules

Activity rules are very similar to the well-known ECA rules (Event Condition Action) [HKMS94], [HEMY96].
Example: When inserting new project assign this project to category new project.

Implication rules will make deductions from the coincidence of properties.
Example: Task without assigned resource is incomplete.
Consistency rules function as a UIR’s immune system. They describe things that are not correct. UIR can solve this inconsistency in two ways: by refusing a request, which would result in an inconsistency or by noticing an inconsistency and asking for a solution to it.

Example: *Task has to be part of some project*

Rules are in fact the same as any other data and can be categorised, linked together, etc. They are typically organised into rule-sets.

There are three situations where rules are applied. (1) The most typical situation is when there is an update request to UIR’s memory. Activity rules can react to the request by executing certain actions. Implication rules deduce consequences of the request and consistency rules check if the resulting state after update will be consistent and the memory can be updated. (2) Rules can also be applied at the user’s explicit demand. One example could be to check selected data against consistency rules (use case described in section 3.5). (3) A third situation is to apply rules when a question is asked of UIR. These kinds of rules are typically used during complex queries to more data-sources, where some preliminary information has to be obtained and then combined to obtain a final answer.

### 4.1.4 Ability of the cyberspace projection

UIR describes cyberspace in the form of a virtual knowledge network. We cannot see this network directly, because it’s too labyrinthine. UIR can create a way of looking at cyberspace, which allows us to orientate ourselves there. This is a projection composed from various visualisation elements. The presentation of an answer can be seen as a projection of UIR’s memory on these visualisation elements.

This cyberspace projection ability is used primarily to solve the ‘focusing problem’. Presenting an answer must be understood as constructing a view on the data. This view is formed by questions and can be used to browse over data. How the data will be presented depends on several factors:

- Amount of data
- Character of data
- User preferences and demands
There are many ways of presenting an answer: forms, tables, charts, mind maps, graphs, various diagrams etc. Graphs are very suitable for presenting relationships. The presentation of time schedules requires some kind of table. These methods can be combined together – for example forms can be the nodes of a graph.

There are two main projection operations

- Browsing through data. An important feature is to use every element as an entry to another part of cyberspace. The key function is ‘more data’. You can focus on each element and ask for more data. This function is a way of homing in more easily on specific information.
- Reconfigure view. You can ask UIR to be generally more eloquent or to be briefer.

4.1.5 ABILITY TO PLAN

There are many situations in ‘UIR’s life’ that require analysis and the choice of the best way of solving a given problem. This includes planning of dialogues, planning of visualisation, planning of optimal access to data-sources etc. There is always one factor, however, which plays the role of restricted resource. This could be, e.g., time or screen space. Well-known planning algorithms can solve this problem. [AI93], [BRATKO86]. It is not just UIR that can use these algorithms
4. UIR - Nine Abilities

(domain-independent), they are also useful for the user, too. Scheduling is an important task in various domains. Helping to create project plans or operation schedules are representative examples.

4.1.6 Ability to Access Data Sources

This ability allows UIR to access various kinds of data sources. UIR can use not just information stored in its own memory but also information stored in external data-sources. This ability has two levels – logical and technical.

The technical level includes access to relational databases, reading XML files, reading HTML pages, communication with other information systems through adapters, etc. Problems at the technical level are well known and solutions exist (for example commercial integration platforms like Ensemble [ENSE06] or Sonic [SONIC06]). This is not, however, true for the logical level.

We have to be able to describe conceptually what a specific data-source contains and to map the physical model of a data-source on this conceptual model. Another challenge is asynchronous communication with data sources – a particular data source can have a very slow rate of response or cannot reply at all in the time available. Each request (not only from the user but also, for example, an internal query used to verify some rule) must be able to handle a situation, where some parts of the request cannot be evaluated at that particular time. This situation can sometimes be handled by data-source replication (copy whole data-source to UIR's server) or by caching (load data-source directly into UIR's memory).

4.1.7 Ability to Be Aware and to Adapt

Let's start with a simple example. Imagine that you want to travel to Prague and want to use a web tool to find your itinerary. If you ask 'how can I travel to Prague' the tool will not be able to answer. It has no information on when and from where you want to go – information on time and place is missing. This information can be filled in from the context. There is the time context – actual time. There is also the place context – where you are physically located. If the tool knows who you are and where you live – personal context – it will also be able to give an answer to a question posed in such an incompletely formed way. Please note, that this question poses no problem, if asked of a human being at a railway station information office. Being aware of context is a key aspect to solving the “how to ask” problem. The above example shows us the role of context in making dialogue more effective.

This is only one aspect of context usability. Context also maintains the continu-
ity of a dialogue. We can imagine, that during a dialogue we travel through a space of facts. What can be the base of this space? We will use a base contained in the enterprise architecture framework proposed by John Zachman [SOZA92], [ZIFA06]. This base has six dimensions. Each dimension is connected with one of the following basic questions: WHAT, HOW, WHERE, WHEN, WHO, WHY. We can then project facts onto this base. Please note, that these questions also have a very close relationship to the structure of sentences. Sentences can be seen as giving answers to these questions. Let’s now take one step in an abstraction. The theme of a dialogue can be summarized in natural language using these questions too. To take an example, let us talk about our tennis match (WHAT) yesterday (WHEN) with a friend (WHO). Let us imagine these propositions as points in a six-dimensional space. The point representing the specific sentence is moving as quickly as we talk. The points representing the theme of the dialogue are more stable and are not moving so quickly. Let us imagine that these points are connected by a rubber band, i.e. their movement is somehow correlated. There are not just two levels only – the dialogue and its theme. There are more levels without sharp distinctions. We can visualise this situation as a configuration of connected points in the six-dimensional space. During the dialogue the configuration is morphing, some parts more quickly, some parts more slowly. This configuration can be seen as scaffolding for attention diffusion. It allows UIR to understand context sensitive messages.

The aspect of the element WHO in the base at meta-dialogue level concerns access rights. The content of the answer will depend on the person who is asking. Access rights are not a separate area, they are an inherent part of the problem of being aware of context. Some information you don’t see because you can’t – access rights. Some information you don’t see because it does not interest you – user preferences.

Problems of context will be discussed in chapter 8 with respect to attention dynamics.

4.1.8 ABILITY OF UIR KNOWING WHAT IT KNOWS

This ability is not obvious at first sight, but is very important. It allows UIR to absorb information and to link it automatically to its current knowledge. We human beings do not need tags telling us if information is new, will update our knowledge or is of no use to us at all. It’s up to us to recognise this. This is not the case with an information system where you must declare clearly which information is new and which information updates existing information. You cannot tell the system information which is already known, as it can lead to duplicities.

Let us imagine a situation in a human context where we must specifically point
out information, which is clearly new. In this case we will need individual personalised newspapers, books, lectures, films, radio, TV, etc. Such communication would be horribly complicated, because to do this you would have to know, what each other person knows.

UIR knows what it knows, so it can read data-sources in the very same way, as we read newspapers or textbooks. It can monitor these data-sources and track changes. To know what it knows is crucial for the ‘monitoring use case’ (section 3.2). Some data-sources are able to track changes for themselves (for example robust database engines). This mechanism is not, however, generally available (an example are HTML pages or you simply don't have direct access to the database).

This ability is also closely related to the ability, which allows us to specify behavioural rules. Let us focus on implication rules. If a supposition is true, then a consequence must also hold. This consequence is a fact, a proposition, about which UIR will inform itself. It is in fact something akin to talking to oneself. If you specify rules for UIR's behaviour, you don't have to worry if a proposition is known, partially known or totally unknown. UIR's task is to handle this situation correctly.

However, the importance of this ability goes much deeper. This ability implies that UIR is able to identify objects referred to in requests with objects present in its memory. Linguists call this ability contextual interpretation [ALLEN95]. This correspondence is computed using a set of properties for each object as well as meta-information about these properties. An example of meta-information is the classification of some properties as 'stable'. This means that when talking about a specific object we cannot casually change a 'stable' property. The property sex is an example here. You cannot one moment be talking about a man and the next moment start talking about the same person as a woman. If you do this, one interpretation could be that you are in fact talking about two different people. This is how UIR would interpret the situation. This ability will be discussed in greater depth in chapter 7.

4.1.9 ABILITY TO FOCUS ATTENTION

The last ability concerns one of the most important concepts in the attempt to construct programs, which behave in a similar way to human beings. This involves looking at attention as a kind of mental energy that can be focused or diffused in a virtual knowledge network. The similar concept can be found in the cogitoid model [WIED99] or in neural networks. Attention is again a kind of resource, which is limited (see section 4.1.5). Attention will focus our minds on aspects, which may be important, and will also allow us continuity in our thoughts. Basic ideas about attention dynamics can be found in [PRO02].
UIR’s memory is made up of large semantic network. All facts passed to UIR or which UIR presents to a user will also take the form of a (small) semantic network. Attention focusing is a mechanism, which allows us to select the subpart of a network, which is of interest to us.

By focusing attention we select a sub-network and construct an object (for example a patient record). Another very useful function is to select several points in a network and for UIR to ‘weave connections’ between these points.

The attention mechanism is also used to combine answers with context. This ability will be discussed in greater depth in chapter 8.
4.2 Defining abilities and addressed problems

Let us summarise UIR abilities and their contribution to the solution of problems addressed in chapter 2. The table expresses the way, in which UIR’s ability is used to participate in finding a solution to the given problem.

<table>
<thead>
<tr>
<th>Where to ask problem</th>
<th>Natural language</th>
<th>Declarative conceptual system</th>
<th>Declarative behaviour</th>
<th>Cyberspace projection</th>
<th>Plan</th>
<th>Access</th>
<th>Aware and adapt</th>
<th>Knows what it knows</th>
<th>Focus attention</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>UIR has description of data-sources’ conceptual systems</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>UIR can access various data sources</td>
<td></td>
<td></td>
</tr>
<tr>
<td>How to ask problem</td>
<td>UIR understands natural language</td>
<td>UIR is able to catch the semantics of the question</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>UIR can take into account context of a dialogue for better understanding of question</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Focusing problem</td>
<td>UIR proposes easy way to specify precise queries</td>
<td>UIR is able to categorise the conceptual system itself</td>
<td>UIR allows presentation rules to be specified</td>
<td>UIR has various ways of presenting answers</td>
<td>UIR can plan, how the user’s focus will be driven.</td>
<td>In constructing an answer UIR can take into account the context of a question</td>
<td>UIR knows content of and changes in data sources – UIR is able to present changes only</td>
<td>UIR can use attention to understand user’s intentions</td>
<td></td>
</tr>
<tr>
<td>Problem of changes</td>
<td>Natural language</td>
<td>Declarative conceptual system</td>
<td>Declarative behaviour</td>
<td>Cyberspace projection</td>
<td>Plan</td>
<td>Access</td>
<td>Aware and adapt</td>
<td>Knows what it knows</td>
<td>Focus attention</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------</td>
<td>-----------------------------</td>
<td>----------------------</td>
<td>----------------------</td>
<td>------</td>
<td>--------</td>
<td>----------------</td>
<td>---------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>UIK's conceptual system is visible and editable</td>
<td>UIK's rules of behaviour are visible and editable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>UIK is able to recognise a change in the structure of data-source – when incoming messages will loose their sense</td>
<td></td>
</tr>
<tr>
<td>Identity problem</td>
<td>UIK permits the forming of relationships between identical concepts from various conceptual systems</td>
<td>UIK enables rules of identity to be formulated</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>UIK can use various data-sources to check cross-referentially if given individuals are the same</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>UIK can use context to solve the identity problem</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>UIK is able to recognise what is identical with its knowledge</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>UIK can use attention to find relevant associations</td>
<td></td>
</tr>
</tbody>
</table>
## 4.3 Defining abilities and use-cases

Let us summarise UIR’s abilities and its role in the use-cases identified in chapter 3.

<table>
<thead>
<tr>
<th>Use-case</th>
<th>Natural language</th>
<th>Declarative conceptual system</th>
<th>Declarative behaviour</th>
<th>Cyberspace projection</th>
<th>Plan</th>
<th>Access</th>
<th>Aware and adapt</th>
<th>Know what it knows</th>
<th>Focus attention</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Question answer Question</strong></td>
<td>Parsing a question</td>
<td>Semantic interpretation of a question</td>
<td>Mentioning and using linking rules</td>
<td>Answer presentation</td>
<td>Plan how to access data sources, Plan how to present answer</td>
<td>Obtaining facts from data sources</td>
<td>Be aware of question context, Respecting access rights</td>
<td>This ability is used to absorb facts. This has nothing to do with question answer question use-case</td>
<td>Use attention dynamics to design an answer</td>
</tr>
<tr>
<td><strong>Monitoring</strong></td>
<td>Parsing a monitoring request</td>
<td>Semantic interpretation</td>
<td>Mentioning and using Event-condition-action (ECA) rules</td>
<td>Answer presentation</td>
<td>Plan how to monitor data sources</td>
<td>Perpetual data-sources checking</td>
<td>Respecting access rights</td>
<td>Recognise new facts or updates</td>
<td>Use attention dynamics to design an answer</td>
</tr>
<tr>
<td><strong>Adapting UIR’s behaviour</strong></td>
<td>Parsing a behaviour update request</td>
<td>Semantic interpretation of a rule</td>
<td>Mentioning all kinds of rules</td>
<td>Rule visualisation in its context</td>
<td></td>
<td></td>
<td>Respecting access rights</td>
<td>Contextual interpretation of a request</td>
<td></td>
</tr>
<tr>
<td><strong>Designing and refocusing virtual knowledge network</strong></td>
<td>Parsing an conceptual scheme update request</td>
<td>Semantic interpretation of a request and changing the conceptual system</td>
<td>Conceptual system visualisation</td>
<td></td>
<td></td>
<td>Respecting access rights</td>
<td>Contextual interpretation of a request</td>
<td>Use attention dynamics to simulate how the updated conceptual system will be useful</td>
<td></td>
</tr>
<tr>
<td><strong>Consistency checking</strong></td>
<td><strong>Natural language</strong></td>
<td><strong>Declarative conceptual system</strong></td>
<td><strong>Declarative behaviour</strong></td>
<td><strong>Cyber-space projection</strong></td>
<td><strong>Plan</strong></td>
<td><strong>Access</strong></td>
<td><strong>Aware and adapt</strong></td>
<td><strong>Know what it knows</strong></td>
<td><strong>Focus attention</strong></td>
</tr>
<tr>
<td>-------------------------</td>
<td>----------------------</td>
<td>----------------------------------</td>
<td>--------------------------</td>
<td>--------------------------</td>
<td>---------</td>
<td>-----------</td>
<td>------------------</td>
<td>-----------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>Parsing a consistency check request</td>
<td>Parsing a consistency check request</td>
<td>Semantic interpretation of a rule</td>
<td>Mentioning and using consistency rules</td>
<td>Present discovered inconsistencies</td>
<td>Plan accessing of data-sources</td>
<td>Monitoring data sources</td>
<td>Respecting access rights</td>
<td>Contextual interpretation of checked facts</td>
<td></td>
</tr>
<tr>
<td><strong>Data connecting</strong></td>
<td><strong>Parsing a data connecting request</strong></td>
<td><strong>Specification of related concepts</strong></td>
<td><strong>Mentioning and using connecting rules</strong></td>
<td><strong>Present resulting data</strong></td>
<td><strong>Plan accessing of data-sources</strong></td>
<td><strong>Access of data-sources</strong></td>
<td><strong>Respecting access rights</strong></td>
<td><strong>Contextual interpretation of checked facts</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Explaining why</strong></td>
<td><strong>Parsing an explanation request</strong></td>
<td><strong>Specification of the scheme of UIR’s mental steps</strong></td>
<td><strong>Monitoring and storing applied rules</strong></td>
<td><strong>Present reasons</strong></td>
<td><strong>Storing used plans</strong></td>
<td><strong>Storing accessed data-sources</strong></td>
<td><strong>Taking into account who is asking (e.g. user, administrator) Respecting access rights</strong></td>
<td><strong>Storing used contextual interpretations</strong></td>
<td><strong>Storing attention diffusion</strong></td>
</tr>
</tbody>
</table>
5. Architecture for UIRs

5 ARCHITECTURE FOR UNIVERSAL INFORMATION ROBOTS

* how can such a system be decomposed * is it a real decomposition? * recursive architecture * UIR’s anatomy *

In previous chapters we have analysed the main use-cases. We have also identified the abilities, which the system needs to fulfil these use-cases. Let us now focus on the architecture of the whole system. The architecture described is not the only way of doing it. We do, however, believe that the basic principles set out are common to the various approaches to building Universal Information Robots.

5.1 THREE RECURSIVE ASPECTS

The whole system can be made up from three components: intellect, memory and communication. This division is also mirrored in the software engineering. A typical information system consists of database, application logic and presentation layer. If you focus on one component – for example on intellect – you discover that there are parts closely related to communication and also to memory. The same situation occurs in other components. Communication, intellect and memory are in fact three aspects inherently and recursively present in each building block of the system. The main design task is to understand this principle and utilise it. This enables to use, for example, the principles and features of intellect at any level and/or place in the system in the same uniform manner.

5.2 SCHEMATIC OVERVIEW

Figure 14: Schematic overview of UIR’s architecture
KMA – the Knowledge Management Agent serves as UIR’s intellect. Current implementation [ETC06] is in Prolog.

UDS – the Universal Data Store supports the fast access and storage of the synapase networks of UIR’s memory. Current implementation [ETC06] is in Cache by InterSystems.

IB – the Internal Bus interconnects the UDS and KMA using TCP/IP protocol. This solution also allows for memory and intellect to run on separate servers.

CHM – the Channel Manager is responsible for communication with channels. The main communication channels as of today are Natural Language Terminal, Dynamic Mind Maps, Activity Charts and Adaptors. Current implementation [ETC06] is in Java.
5.3 UIR’S ANATOMY

UIR consists of a set of organs. Each has its own specific mission and its own inputs and outputs. There is considerable misunderstanding about what we need to build an “intelligent” machine. A commonly understood way is to find some genial formula, algorithm or structure. The key is not to elaborate one thing aspect brilliantly but to elaborate all things appropriately. The principle lies in many quite simple automats or organs co-operating with each other (cf. [DEN-96], [MINSKY98]). An organ is each piece of functionality with a clear mission, inputs and outputs, which is not the exclusive part of another organ. The following figure illustrates UIR’s most important fundamental organs.

![UIR organs](image-url)
UIR’s memory is implemented as a special kind of hybrid self-referential semantic network (chapter 6) in the database engine Cache. This memory is accessible via a set of routines, using a TCP/IP socket connection managed by the socket server.

UIR’s user interface consists of set of communication channels (section 4.1.4) managed by the channel manager.

We have already mentioned that UIR’s intellect can be subdivided in the same way as the whole UIR: memory, communication and intellect.

The access to UIR’s memory is done by internal bus. The scheme of UIR’s memory is defined in the synapse scheme (section 6.3.2). The synapse scheme is also used as a basis for routine generator, which automatically generates all necessary memory routines. The initial content of UIR’s memory is defined in so called fixing point (section 6.5). Fixing point can be seen as a top level ontology enabling UIR to communicate, understand, and infer.

The communication part of intellect contains channel bus – the connection to communication channels. Operation parser (section 4.1.1) decomposes incoming text requests to operations and their parameters. It recognises if the request is a question, new information, a command etc. IDEX parser (section 4.1.1) decomposes the textual representation of identification – IDEX – into an internal structure representing the IDEX semantics. IDEX render is able to generate the textual representation of IDEX from this internal structure. All UIR’s language knowledge is stored uniformly in UIR’s memory (chapter 6). During UIR’s reviving process, the language educator generates all necessary language structures and stores them in memory. Rule translator translates the textual representation of a rule to UIR’s internal structure. Each UIR functionality is covered and is accessible using certain communication automata. These are interpreted by communication automata interpreter. Communication dispatcher manages UIR’s functioning in its entirety. Every communication channel has its own channel content manager, which can send and receive information from the communication channel.

UIR’s ‘pure’ intellect contains organs able to translate IDEX structures to synapse clusters – parts of the semantic network (chapter 6, 7, 8) and vice versa (idx to synapse cluster and synapse cluster to idex). One of the most important UIR organs is the synapse transmuter – a general, which allows pattern matching over the semantic network and network transformations (transmutations). A lot of other UIR organs are based on the synapse transmuter. An example of this is UIR’s inference engine, which manages the checking of memory consistency, the deduction of new knowledge and reaction to events happening during UIR’s functioning. Object maker (section 8.2) is used for adding appropriate context to a given individual. Connection weaver (section 8.3) can weave connections between several individuals – it is typically used for handling queries. Split cluster (chapter 7) is one of the most important algorithms in UIR. It gives UIR the ability to know
what it knows. *Integration manager* is responsible for accessing data stored in external data-sources. *Time machine* gives UIR the ability to be aware of passing time.

A detailed description of UIR’s organs is beyond the scope of this thesis. This is the subject of non-provisional patents now being prepared on the basis of provisional patents pending since March 2006 at US Patent Office [PAT06a], [PAT06b], [PAT06c].

The Universal Information Robot with the architecture described has been implemented by the UIR team of approximately 25 people, where I head up construction. The first generation of UIR was introduced in 2003. We are now working on generation 5 and are evaluating UIR in a number of research and business projects, amongst which are the UIRON - UIR ONcology project funded by the Czech Academy of Science and the ILAB project using UIR as an assistant in a business consultancy company.

The next three chapters focus on the three most important principles from the theoretical point of view (but also practically evaluated and used). The first is the so-called declarative conceptual system, which is the key principle of UIR’s memory. The second is UIR’s ability “to know what it knows” – which is also probably the foundation of human thinking. The third is the ability to concentrate attention and to use this concentration for the enactive perception of objects and connections in the environment.
6 DECLARATIVE CONCEPTUAL SYSTEM

*what we need to represent knowledge in UIR’s memory* inspirations *semantic networks* Nanohouse *Diamond Modelling Tool* synapse oriented approach of UIR’s memory *synapses* how synapses are organised *bunches of synapses* - synapse clusters *synapse cluster sequences* fundamental structuring of UIR’s knowledge – the type system *two relationships forming the type system* initial UIR’s type system and pitfalls in its design *are objects more important than connections* seven fundamental connections *how behaviour relates to the conceptual system*

In this chapter we will describe in greater depth UIR’s ability to work with a declarative conceptual system leading to the construction of a universal memory able to store everything.

6.1 REPRESENTATION REQUIREMENTS

There are many ways of representing knowledge (approach using logic, semantic networks, frames, ER modelling, universal modelling, object-oriented approach, connectionist models, conceptual graphs...). In our case, choosing or designing a suitable way of representing knowledge requires deeper analysis.

We need a method of knowledge representation, which is used not only to describe knowledge stored in external data-sources but also to store UIR’s own knowledge. We need to store individual user preferences, rules on how to behave, knowledge about data-sources etc.

Our method has to be universally applicable. It will not be practical to use different methods for external and internal knowledge (this distinction is anyway problematical). Furthermore, we will be integrating knowledge structured and stored by third parties, where no assumptions can be made.

Universality also implies self-reference. We have to be able to describe the properties of properties, connections between properties, state rules about rules etc. We need to store not only structure (e.g. the structure of facts in databases) but also behaviour in declarative form. This requirement mirrors the Business Rules Approach. [BRC06]

6.2 INSPIRATIONS

6.2.1 SEMANTIC NETWORKS

An extract from an article by John F. Sowa [SOWA92] basically describes semantic networks:

A semantic network or net is a graphic notation for representing knowledge in pat-
terns of interconnected nodes and arcs. Computer implementations of semantic networks were first developed for artificial intelligence and machine translation, but earlier versions have long been used in philosophy, psychology, and linguistics.

What is common to all semantic networks is a declarative graphic representation that can be used either to represent knowledge or to support automated systems for reasoning about knowledge. Some versions are highly informal, but other versions are formally defined systems of logic. Following are six of the most common kinds of semantic networks, each of which is discussed in detail in one section of this article.

**Definitional networks** emphasize the subtype or is-a relation between a concept type and a newly defined subtype. The resulting network, also called a generalization or subsumption hierarchy, supports the rule of inheritance for copying properties defined for a supertype to all of its subtypes. Since definitions are true by definition, the information in these networks is often assumed to be necessarily true.

**Assertional networks** are designed to assert propositions. Unlike definitional networks, the information in an assertional network is assumed to be contingently true, unless it is explicitly marked with a modal operator. Some assertional networks have been proposed as models of the conceptual structures underlying natural language semantics.

**Implicational networks** use implication as the primary relation for connecting nodes. They may be used to represent patterns of beliefs, causality, or inferences.

**Executable networks** include some mechanism, such as marker passing or attached procedures, which can perform inferences, pass messages, or search for patterns and associations.

**Learning networks** build or extend their representations by acquiring knowledge from examples. The new knowledge may change the old network by adding and deleting nodes and arcs or by modifying numerical values, called weights, associated with the nodes and arcs.

**Hybrid networks** combine two or more of the previous techniques, either in a single network or in separate, but closely interacting networks.

Some of the networks have been explicitly designed to implement hypotheses about human cognitive mechanisms, while others have been designed primarily
for computer efficiency. Sometimes, computational reasons may lead to the same conclusions as psychological evidence.

Network notations and linear notations are both capable of expressing equivalent information, but certain representational mechanisms are better suited to one form or the other. Since the boundary lines are vague, it is impossible to give necessary and sufficient conditions that include all semantic networks while excluding other systems that are not usually called semantic networks.

End of quotation.

6.2.2 NANOHOUSE

We will quote an extract from an article by Dan Lindstedt [LIN04]:

If we think about our minds and our primitive understanding of their function, they store information in both short and long term memory. We need the long-term memory to understand and interpret the context of the short-term memory. Has anyone said that short-term memory uses a different structure of neuron cells than long-term memory? No!

Why then do we insist on separating different structures for processing purposes like the ODS (operational data store), OLTP (on-line transaction processing) and DW (data warehouse)? If we are ever to construct a Nanohouse™ then we must accept the idea that the model (structure) of the data is not dependent on the function or utilization of the information!

The future Nanohouse™ is one type of structure with potentially different types of functions attached. The more we separate our systems, the further from our destination we get. That is, if our destination is a thinking machine, or even a semi-smart device.

Just as short-term memory houses “today’s events” we need the long-term memory to place these events into context. If there are two different applications of the same information, why not use the same physical structure? Why not integrate all our systems information into a single structure with different functionality?

I’m suggesting that thinking systems must evolve from a similar structure (form). I’m also suggesting that in order to get there, our data sets must converge into the same repeatable structures. I also believe that by combining all the information (regardless of utilization) it will allow us to begin to see patterns in business we have never identified.

End of quotation.
6.2.3 DIAMOND MODELLING TOOL

We will now describe the Diamond Modelling Tool (DMT) proposed by Staniček and Procházka [STPR02], [STA03], [STPR04]. This tool represents a universal approach to the representation of knowledge. A conceptual model of the tool is shown in the following figure.

![Diamond Modelling Tool Diagram](image)

**Figure 16: Diamond Modelling Tool**

To describe the feature, we will use a quotation from the article [STPR04a]:

Each concrete/abstract object of our interest is represented in the tool by a dmt-object. Diamond containers play the role of entity classes of classical conceptual models. Elements of any kind can be inserted into the container as well as removed from it.
**Definition: Container (#Object)**

Container (#Object) contains all the dmt-objects that can be mentioned in DMT, i.e. which can be assigned certain properties. If the object we focus on turns to be a connection or an operation or a category or a rule, and we want to use it, then we link it to (#Connection) or (#Operation) or (#Category) or (#Rule), respectively, by an edge R1 or R3 or R2 or R4. Then we say that this object from the class (#Object) represents this connection or operation or category or rule. Note: an object may represent only one of the four mentioned possibilities.

**Definition: Container (#Connection)**

Container (#Connection) contains n-tuples of dmt-objects; every element of the container (#Connection) is called a connection.

**Definition: Container (#Operation)**

Each element of the container (#Operation) is an algorithmically computable transformation from Diamond Tool states to a new DMT state. By the state of Diamond Tool we mean one particular filling of the Diamond Tool (as a container) by elements-instances that may dwell in the Diamond Tool. The elements of the container (#Operation) are called operations.

**Definition: Container (#Category)**

Container (#Category) is defined in such a way that every of its elements has the following properties:

- it is a container for dmt-objects,
- it is one-to-one mapped to the pair <Cn, Op>, where Cn \( \in \) (#Connection), Op \( \in \) (#Operation); and,
- it holds about the operation Op that by means of the connection Cn it can recognize whether a given object is or is not in this container.

The elements of the container (#Category) are called categories. The connection Cn is called a defining connection of this category. The operation Op is called a defining operation or an evaluator of this category.

**Definition: Container (#Rule)**
Container (#Rule) is defined in such a way that every of its elements has the following properties:

- it is a dmt-object,
- it is one-to-one mapped to the pair \(<\text{Cn}, \text{Op}\>\), where \(\text{Cn} \in (#\text{Connection})\), \(\text{Op} \in (#\text{Operation})\); and,
- operation \(\text{Op}\), by means of the connection \(\text{Cn}\), carries out a test whether the rule is valid, i.e. the operation returns the value True if the test is successful, and value False in the opposite case.

The elements of the container (#Rule) are called rules. The connection \(\text{Cn}\) is called a specifying connection of this rule. The operation \(\text{Op}\) is called a specifying (or testing) operation of this rule.

The edges 01 and 02 in Figure 1 link a category with, respectively, a defining connection and an evaluating operation of this category. The edges 03 and 04 link a rule with a specifying connection and the testing operation of this rule. The Diamond Diagram contains, moreover, the so-called \(\hat{P}\)-edges (projection edges) that link individual objects with a connection that describes the given relationship between these objects.

The Diamond Tool contains a family of operations, which enable us to create, delete, and update dmt-objects (CREATE, DELETE, WRITE, READ, OBTAIN, LABEL), to “travel” around the circumference of the Diamond (GET_CONNECTION, GET_OPERATION, GET_CATEGORY, GET_RULE), and to work with \(\hat{P}\)-edges (projection of a static connection onto its component, adding and removing an element to/from a dynamic connection, testing if an element belongs to a connection, etc.).

The most important are those operations that make the Diamond Tool a universal modeling tool. Edges R1, R2, R3 and R4 are jointly called R-edges. By means of R-edges switching between mentioning (operation MENTION) and using objects (operation USE) is realized. If we stand on an object of the class (#Object), which has been mentioned, then in order to use the object (operation USE) we cross along the respective R-edge to the represented connection (#Connection) or operation (#Operation) or category (#Category) or rule (#Rule), which can be directly used. Operation USE may fail on objects belonging to (#Object) that are “mere” objects, i.e., do not represent any category, operation, rule or connection. An example of such an object can be a concrete document.

If we are in a vertex of the Diamond Diagram, then the operation MENTION takes us, along the respective R-edge, to the representing object which can be directly mentioned.
6.1 Operation MENTION applied to an object belonging to (#Object) returns the same object without any change. Thus, this operation cannot fail. Operation USE applied to a connection belonging to (#Connection) makes it possible to use the connection: it returns a list of objects—components—of the given connection. Similarly, operation USE applied to a category belonging to (#Category) makes it possible to use the category (to inform on its current content, or possibly, to add further elements to the category). Operation USE applied to an operation belonging to (#Operation) simply executes this operation. Finally, operation USE applied to a rule belonging to (#Rule) performs a test that checks the validity of the given rule.

Categories are constructed in an extremely universal and flexible way. We can work with categories specified simply by the list of their elements, or with categories that use their relationships to other categories for the evaluation of their own content, with fuzzy categories, etc. Similar flexibility and universality has been applied in case of rules.

Due to representative edges R1, R2, R3 and R4, we can mention any DMT object—relate it with other entities, categorize it and fill its attributes. We can create connections, e.g. between categories (more accurately, between representative objects of these categories), or between categories and rules, we can categorize the categories, etc. This flexibility is of a key importance. Unlike traditional modeling tools, which strictly separate the modeled world from the “tool world”, the Diamond Tool works within one universal world. Using the Diamond it is possible to integrate models on various levels of abstraction. For instance, we can create a model of reality first, then develop another model used for tuning the former, and still another model above the two, which can learn by “observing” their interactions. Switching between USE / MENTION operations makes it possible to use the knowledge and to develop it.

End of quotation.

6.3 Synapse oriented approach

UIR’s memory is designed as a special kind of hybrid semantic network. The organisation of the network is based on the Diamond modelling tool. This network is stored using the database engine Caché from Intersystems. The storage is using low level access to the hierarchical core of Caché. We will focus on main principles and not on technical details. Detailed motivation and description can be found in [OSKY06], [OSKY06a]. Basic principles on which a synapse oriented data store can be built are the following:
Each piece of information can be broken down into a set of elementary binary connections between individuals (see binarization principle [DMKS86]). These connections are called synapses.

- Individuals are ‘naked’ [MAT98]. They only have to be instances of specific types.
- Synapses can be woven together in various ways depending on context. Objects in a synapse-oriented approach are not solid – they represent the pragmatic views of the individual only. This can be summed up by the motto “objects are subjective.”

Let’s now formally describe elements used in a synapse-oriented approach.

### 6.3.1 Synapse

A synapse is an atomic element of information.

**Definition:** Synapse

A synapse is each tuple \((T, M, I, T_1, P_1, T_2, P_2)\), where \(T\) means synapse type, \(M\) means synapse modus, \(I\) means unique synapse identifier. \(T_1\) and \(T_2\) are unique identifiers of types or constrained variables, \(P_1\) and \(P_2\) are unique identifiers or descriptions or constrained variables. \(P_1\) and \(P_2\) are called synapse elements. \(P_1\) is an instance of type \(T_1\), \(P_2\) is an instance of type \(T_2\). Synapse is a record about the specific relationship of given type \(T\) and the modus \(M\) between \(P_1\) and \(P_2\).

Synapses are organised using the type system. Type system design is in essence a pragmatic task. Examples of synapse types used in UIR are

- *name synapse* maintains information between an individual and its name
- *category synapse* maintains information between an individual and the category to which it belongs

The modus of a synapse expresses the reality of information maintained by the synapse. There are two main moduses ‘can be’ and ‘is’: The modus ‘can be’ states that information maintained by the synapse can occur in the UIR’s memory. The modus ‘is’ states that information maintained by the synapse is present in UIR’s memory.
Each synapse is uniquely identified, enabling synapses to be created between synapses (in the case where $P_1$ and/or $P_2$ is identifier of the synapse). This ability is very useful if we want to record the properties of the synapse. We can use it, for example, if we want to add information about the probability of a ‘can be’ synapse.

There are three possible ways of representing individuals in a synapse:

- as the identifier of an individual
- as a description in the form of a number or a text. It is not necessary to have a unique identifier for a description.
- As an abstract individual. An abstract individual is modelled using a free variable. A variable is subject to at least one constraint – the type constraint (for example the individual can be a person only). Another typical constraint is enumeration constraint (for example individual can be John, Terry or Patty).

**Definition:** Closed synapse

*Closed synapse* is a synapse where $T_1$ and $T_2$ are individual identifiers, $P_1$ and $P_2$ are either individual identifiers or one of them is a description.

**Definition:** Open synapse

*Open synapse* is a synapse, which is not closed.

Simply, an open synapse contains at least one constraint variable.

6.3.2 Synapse scheme

We need to organise synapses in an ordered way. This situation is very similar to the database design process, where the database scheme itself is the order. Seen from the viewpoint of ER-modelling, synapses can be regarded as associative entities. They form a relationship between $P_1$ and $P_2$. The other elements of a synapse can be regarded as entity attributes.

Thus we can model a synapse scheme using an ER model. Let us define a simplified synapse scheme:
6. Declarative Conceptual System

Figure 17: Simplified synapse scheme.

This scheme defines five types of synapse: categorisation, super-type/sub-type, type/instance, part/whole and name. It also defines the cardinalities of used synapse types. Let us take for example the name synapse type: one individual can have zero or more names, one name can denote zero or more individuals. The types of elements of each synapse type are also defined. Consider for example the categorisation synapse type: the $P_1$ element will be an individual and the $P_2$ element will be a category. Constraints derived from a synapse scheme will be defined more precisely in chapter 7.

Information capability [DUZI00],[DMKS86] of the system is derived from the conceptual model of the system. The letter U in ‘UIR’ means Universal. We cannot therefore restrict UIR’s information capability. This can be achieved only if UIR’s synapse scheme is universal. UIR’s synapse scheme is therefore derived from the Diamond Modelling Tool to enable this condition of universality to be fulfilled.

6.3.3 Synapse cluster

Let’s now define collections of synapses.

Definition: Synapse cluster
A synapse cluster is a list of synapses.
Definition: Closed synapse cluster

A closed synapse cluster is every synapse cluster, where all synapses are closed synapses.

Definition: Connected synapse cluster

A connected synapse cluster is every synapse cluster, which forms a connected graph.

6.3.4 Synapse cluster sequence

Definition: Synapse cluster sequence:
A synapse cluster sequence is every synapse cluster list.

Synapse cluster sequence can typically be used for capturing time sequences from some part of UIR’s memory. We will discuss this theme in section 6.6.

6.4 Type system

A type system represents the skeleton of a conceptual system. The extent of such a type system is a pragmatic question (think of fish and the variety and complexity in their skeletons which makes them more or less easy to eat). There are two relationships concerned with types, which are extremely important. These relationships are unfortunately often incorrectly mixed up and called an ISA relationship.

6.4.1 Type/Instance relationship

This relationship has the following semantics: individuals, which are instances of a given type.

Example: John is a person.

We will use the following pragmatic assumption: Every individual is an instance of exactly one type. All types are also individuals, so they are also instances of some types. The relationship type-instance forms a hierarchy called TI hierarchy (Type/Instance). Because we need this hierarchy for practical use, the TI hierarchy cannot be infinite. The solution is to create a loop at the top of the TI hierarchy (this looped type must be instance of itself).
The relationship between type and instances is crucial. It is not, however, necessary to maintain this relationship in an un-uniform manner with respect to other relationships. A special type of synapse is used for maintaining a type-instance relationship.

Example: The linguistic categorizations of words. Each word can be seen as the instance of a linguistic type system. This system includes types such as *verb*, *noun*, *adjective* etc. This type system also contains a loop. The word 'noun' is both an individual and an instance of the type 'noun'.

6.4.2 Sub-type/super-type relation

This relationship has the following semantics: sub-types of a given type.

This relationship will typically form a class hierarchy in an object-oriented approach. If type A is the sub-type of type B, all instances of A are also transitive instances of type B. This relationship is also called specialisation/abstraction.

Example: employee is a sub-type of the type person. This classification implies that if John is an employee, he is also a person.

Once again, we will use the assumption: each type has exactly one super-type. Here we have the same problem as with the TI hierarchy – we have to create a loop. The super-type/sub-type relationship also forms a hierarchy. We will call this hierarchy *SA hierarchy* (Specialisation/Abstraction).

Example: The biological classification of beings. Here we have the same loop problem, too – every being is a special kind of being.

6.5 Fixing point

UIR’s conceptual system can evolve but cannot start from nothing. There must be a basic fixed conceptual system – a *fixing point*. The reasons for this are:

- The evolution of UIR’s conceptual system can then be based on a specialisation. Each newly introduced concept is ‘only’ a specialisation of some existing concept.
- The fixing point is the same for every UIR instance and can be used as a common basis in communication between UIRs.
- Structure and behaviour goes side by side. UIR’s organs have to be programmed (e.g. inference engine or natural language parser). These organs use several concepts (e.g. *rule*, *rule-set*, *word*, *synset*, *phrase*) and adding them to the initial conceptual system is very useful. We can then mention them and
use UIR’s functionality on its own internal processes (validate and visualise UIR vocabulary, browse through rule-sets etc.). This relates to section 5.1 – three recursive aspects.

6.5.1 Fixing point – object viewpoint

The task of fixing point design is very closely related to ontology design. When talking about ontologies the main focus is concentrated on various classifications of objects.

The upper part of the well-known ontology Wordnet is formed from concepts like object, artefact, thing, relation [VOS98].

UIR’s ontology is based on five Diamond concepts object, category, rule, connection, operation. The concept object is specialised into the concepts agent, process, product etc. [STA03].

6.5.2 The art of upper fixing point design - object viewpoint

The task of fixing point design is not so easy. The fixing point forms a ‘conceptual’ skeleton of UIR. The above mentioned cardinality restriction (1,1:0,M) in type/instance relationship and super-type/sub-type relationship is crucial – without this restriction the type system will not be a skeleton. We have to design the type system carefully so as not to restrict UIR’s view of the world. We will illustrate this point by using the Zachman framework [ZIFA06] for enterprise architecture as a fixing point.

“This framework has been widely adopted by systems analysts and database designers. It provides a taxonomy for relating the concepts that describe the real world to the concepts that describe an information system and its implementation.” [SOZA92].
Figure 18: Zachman framework
Let us focus on the framework columns. They correspond to the English questions: *what*, *how*, *where*, *who*, *when* and *why*. Column *what* shows what entities are involved, column *how* shows how they are processed, column *where* shows where they are located, column *who* shows who works with the system, column *when* shows when events occur and column *why* shows why these activities are taking place.

This framework is extremely useful. It is, however, important to realize that usage of the framework is closely related to the viewpoint used. Let us illustrate this feature with an example from the automotive industry.

Example: The dimension *what* describes wheels, bodywork, gearbox, chassis etc. The dimension *how* describes varnishing, screwing, assembly etc. This is, however, only a general view. The view of the financial officer is quite different. Wheels, bodywork, etc. as well as varnishing, assembly, etc. will be described in the dimension *what* – they are entities which are interesting because they cost money. The dimension *how* will contain processes like billing, auditing etc.

If you choose one of these perspectives for fixing point design, UIR’s perspective will be fixed to just one point. This is a particular problem for UIR because it has been designed to connect several viewpoints (i.e. general and financial). The fixing point has to contain concepts that are independent of viewpoint. Such concepts can be found for example in the Diamond Modelling Tool – *category*, *connection*, *rule*, *operation*, *object*. Here we can shift our perspective by travelling on the edges, from mentioning (focusing on an individual as an object) to using (focusing on an individual as either a category or a connection or a rule or an operation).

### 6.5.3 Fixing point – connection viewpoint

Designing a fixing point from the viewpoint of object is of course important. There is, however, a much more important viewpoint which is often overlooked – the viewpoint of connections. The essence of the connection viewpoint is also present in object viewpoint. This can be seen in the Diamond Modelling Tool – one of its nodes is the Connection. If we use the technique of type specialisation when dealing with connections and classify these connections, we make the whole fixing point complete. It will be not only a hierarchy of how objects can be specialised but also a hierarchy of connections used for constructing complex situations. The fixing point will contain both dual viewpoints – objects and connections.
6.5.4 Fundamental connections

Connections are the more important part of UIR’s fixing point. There can be various approaches to designing object classifications. Connection classifications, however, will always be very similar. Our world contains a huge amount of various kinds of objects, but these objects are constructed using several fundamental connections. We emphasize seven of them. They are derived not from the ‘objective’ world, but they are derived from our pragmatic views to the world. The question of the ‘completeness’ of the fundamental connections selected is not appropriate, as there is no ‘objective’ method for describing our ‘subjective’ views. Selected fundamental connections are also recursive, which makes them even more important. These are shown on the following figure.

![Fundamental recursive connections](image)

The abstraction / specialisation connection expresses the axis of change in focus from abstract to more concrete. This is the axis, which we use in art where we move from the “concrete” pictures of Vermeer to the “abstract” paintings of Picasso. The axis can also be seen in biological classifications, where our focus can move from the “generic” dog to the “specific” retriever.

The type / instance connection expresses the axis of change in focus from instance to type.

The part / whole connection is sometime called aggregation. It links together objects, which are parts of other objects. Parts can also be made up from other objects. This connection expresses the axis of change in focus from whole to part.

The owner / ownership connection is very important in our own world. This connection is also recursive – for example, the state owns a factory and the fac-
tory owns a machine. This connection expresses the axis of change in focus from owner to ownership.

The superior / subordinate connection is important for all types of organisation (factory, society, processes in our bodies, etc.). This connection expresses the axis of change in focus from superior to subordinate.

The predecessor / successor connection relates to the phenomenon of time and is also very close to causality. This connection expresses the axis of change in focus from predecessor to successor.

The identity connection is somewhat strange but is the most important, as it tells us that two or more things are the same. This connection expresses the axis of change in focus from more objects unified into a single entity to more distinguished objects with individual identity.

Other important properties of these connections are transitivity and cardinality. Let us stress that these properties are general. It is possible to create sub-types of a given connection type, where, cardinality, for example can be more restrictive. General connection properties are illustrated in the following figure.

Let us now discuss another aspect of fundamental connections. UIR’s memory is a huge tangle of synapses. If we travel through this network, our associations take us through various passages, paths and lanes. There are several highways along which our attention often travels – fundamental recursive connections. They determine the direction of our thoughts; we go on the time line to the past or to the future (predecessor/successor), we examine an object by splitting it into parts or vice versa (part/whole), we generalize or specialise our experiences (abstraction/specialisation, type/instance), we look for superior or subordinate processes (superior/subordinate), we travel from owner to ownership or vice versa (owner/ownership) and
we recognise, that many different things are from a certain viewpoint in fact one and the same thing or that one object is from a certain viewpoint a set of different things *(identity)*. UIR must be familiar with the order of our thoughts, as it uses these connections for diffusing and focusing its attention too.

We also use these connections for tricks with language. For example, metonymy (replacing part with the whole) is often used. Type/instance connection is also often used for these tricks – e.g. when talking about ourselves, we often refer to ourselves as a man – e.g.: Don't do it man.

**6.5.5 SYNAPSE CLUSTER SEQUENCE AND CONNECTIONS**

We often use word sequences – for example new-prepared-approved-finished to express changes in the status of a document, egg-larva-pupa-imago to express the stages in the life of an insect, brick-wall-room-house to express the composition of complex objects.

The meaning of such sequences is often expressed in terms of fundamental connections. The recognition of such connections is so tied up with human intelligence that this kind of task features widely in intelligence tests.

Let’s now focus on how these sequences will be represented in UIR’s memory. The element of the sequence - word – denotes something, which can be identified by an IDEX. An IDEX can be transformed into a synapse cluster (section 5.3). We can therefore use sequences of synapse clusters in general. Each element of the sequence will describe a synapse pattern. This gives us a very powerful apparatus for describing various kinds of thread or flow in UIR’s memory. It can be used for discovering causality chains, extracting records about processes, finding patterns of compositions etc.

**6.6 CONCEPTUAL SYSTEM AND BEHAVIOUR**

There is an omnipresent duality between structure (conceptual system) and behaviour. This duality is present in software engineering in the question – is the data model or the process model the more important? How does the world described by the data model differ from the world described by the process model? The most important difference is that time exists in the second world. From UIR’s point of view the predecessor/successor axis is used in this world. The conceptual model describes the world by concentrating on the super-type/sub-type and type/instance relationships over objects and connections. The conceptual system expresses the structure of UIR’s view on the world. Data
are records about viewing the world through this conceptual view. They record the various world states. State changes are events. A state is expressed as a synapse cluster. An event can be expressed as a synapse–cluster sequence containing two clusters (sequence expressing predecessor/successor connection). If we express a sequence of events as a synapse cluster sequence, we will obtain a record about the execution of a process.

When we need to specify UIR’s behaviour, we must describe, how UIR will react to certain events. This reaction will result in some memory update – the reaction will be expressed as a collection of synapses and stored in UIR’s memory. Instruction to change states can be done using rules. These rules can be quite specific or they can be quite abstract. Abstract rules can be parameterised by knowledge expressed as synapses.

The classification of behaviour is based on the classification of events. Basic events are defined by synapse and by the action – “insert”, “update” or “replace”. As an example let us take the owner/ownership connection – say between you and some book. If this connection is established you begin to own this book. Classification can be more precise and can take account of who is establishing the connection – if connection creation was passive (done by a third party) or active (done by you). Putting events in order is very useful in the formulation of rules – e.g.: alert me, when this house changes its owner.

We can see, that well-designed universal memory is also the key to effective behaviour modelling.
In this chapter we will describe the split-cluster algorithm – the centre of UIR’s ability to know what it knows.

7.1 MOTIVATION

Example: Let us assume that an IDEX has informed us of the following situation:

Person with properties:
* has name Jay
* is male
* date_of_birth = 1964-05-17
* function is graphic designer
* is author of the project Mitu_garden

To absorb this information means relating it to our present knowledge of the world. Several situations can happen:

- we know what ‘person’ means, but do not know this particular person
- we know this person and all his characteristics – the information tells us nothing that we do not know already
- we know this person but do not know either his date of birth or his project
- we know this person but we think that his date of birth is different
- we know this person but think that he is not the author of the project
- etc.

As a human being we are able to solve each of the situations listed above. The problem of relating message content to the knowledge we already have is an integral part of our everyday life.

What makes this task difficult is the phenomenon of homonymy. If each word were to have only one meaning, the task would be easy. But can you imagine what such communication would look like?

Almost every word has more than one meaning. We use context to choose word
meanings, which fit together in the sentence and which fit in with our knowledge [ALLEN95]. The same trick is behind the split cluster algorithm.

7.2 INTRODUCTION TO THE SPLIT CLUSTER PROBLEM

7.2.1 PROBLEM SPECIFICATION

Let’s now formulate this problem in terms of UIR.

Split cluster input:
- Synapse cluster P representing message content.
- Closed synapse cluster M representing actual memory state

Split cluster output:
- List of possible solutions L demonstrating how synapses from P can be broken down into four sets:
  - K: Known synapses – synapses that are already present in the cluster M
  - U: Unknown synapses – synapses that are not present in the cluster M
  - R: Synapses that are not present in the cluster M, but will replace existing synapses from the cluster M
  - N: non-deterministically solvable synapses – synapses that cannot be deterministically placed to set K, U or R

Each element of list L is called split cluster solution.

To combine synapses from cluster P with cluster M, we have to choose one possible solution from the list L and then go through the following steps:
- Ignore synapses from set K – they are already present in cluster M
- Add synapses from set U to synapse cluster M – they represent new knowledge by which the current memory content must be extended
- Replace synapses in cluster M with those from cluster P – they represent a change in current knowledge
- Decide what to do with synapses from set N – they can be either known or unknown. At this moment, however, we do not have enough information to make this decision. We can leave it until later. This is the case when we are reading a detective story and are waiting until the end of the story to find out who the murderer is. Another possible solution is to refuse the message because of its un-ambiguity.
7. UIR Knows What It Knows

**Definition:** Closed solution
A split cluster solution is *closed* if the set N is empty.

**Definition:** Open solution
A split cluster solution is *open* if the set N is not empty.

Closed solutions are typically required if the synapse cluster is used for informing UIR. An opened solution can be used for example in rules: The solution becomes closed after the rule is applied to a given individual.

7.2.2 VARIABLE INSTANTIATION TASK

Let’s describe the elementary task of the split cluster algorithm. Suppose the given synapse is \( S = (T, M, I, T_1, P_1, T_2, P_2) \).

If synapse \( S \) is closed (\( T_1, P_1, T_2, P_2 \) are not constrained variables), the task is easy. If synapse \( S \) is present in the synapse cluster \( M \) then synapse \( S \) is known. If the synapse is not present in cluster \( M \) then synapse \( S \) is unknown. In certain conditions synapse \( S \) can replace the appropriate synapse from cluster \( M \) (we will discuss this possibility later).

If synapse \( S \) is not closed, it contains one or more constrained variables. The task is, now, to instantiate the given constrained variable according to the synapse cluster \( M \) or to instantiate this variable with a newly introduced individual.

Example:
Synapse cluster \( P = \)

\[
\begin{align*}
P_1 &= ('categorization synapse', 'is', p1, person, X, status, married ). \\
P_2 &= ('name synapse', 'is', p2, person, X, first name, 'John') 
\end{align*}
\]

These synapses tell us that person \( X \) with the name John is married (for transparency reasons, we have used names instead of unique identifiers for the elements \( T_1, T_2, P_2 \)).

Let’s suppose that the type \( person \), the categorization \( status \) and the category \( married \) are present in synapse cluster \( M \) (they are known). The following situations with element \( X \) can then happen:
X can be instantiated with some individual from synapse cluster M, which fits with synapses P
1 and P
2. There is no appropriate individual from cluster M, which can be assigned in X. In this case we will create a new individual and instantiate X with this individual.

In other words, the algorithm has to decide, if the content of a variable will be known or unknown. Finally, this decision will in general lead to the classification of synapses to the set K, U, R or N.

7.2.3 SYNAPSE CONSTRAINTS

Not all solutions to the split cluster problem are correct. There are several constraints, which synapses have to satisfy.

7.2.3.1 TYPE CONSTRAINT

**Definition:** Synapse element type constraint

*Synapse element type constraint* is the relationship between synapse type S and tuple (T
1, T
2) with the semantics: types (T
1, T
2) describing the permitted types of synapse elements P
1, P
2 in synapse of the given synapse type S.

Example: Synapse type ‘category synapse’ can have the following synapse element type constraint: (individual, category), i.e. T
1=individual, T
2=category, where instances of type ‘individual’ are all individuals and instances of type ‘category’ are all categories.

7.2.3.2 THE CARDINALITY CONSTRAINT

**Definition:** Synapse type cardinality constraint

*Synapse type cardinality constraint* is the relationship between synapse type and tuple (Ml, Mu, Nl, Nu) with semantics: cardinality (Ml, Mu, Nl, Nu) of the given synapse type.

Cardinality is used in the sense usual in HIT data modelling – if we look at a synapse type as being an ordinary binary HIT attribute. [DKMS86].

Example: Synapse type ‘name synapse’ describing the relationship between an individual and its name. The cardinality of the relationship is 0,M:0,N. This means
that each individual can have zero or N names and each name can belong to zero or M individuals. When the cardinality of the ‘name synapse’ synapse type is defined in this manner, both synonymy (individual can have more than one names) and homonymy (one name can be the name of more than one individual) are allowed.

**Definition:** Synapse cluster cardinality constraint
A synapse cluster satisfies synapse cluster cardinality constraint if and only if all its synapses satisfy synapse type cardinality constraint.

The cardinality constraint is the way, in which we can recognise those synapses, which will replace synapses from cluster M. If the synapse is to be added to the cluster M, it must not violate the synapse cardinality constraint.

Example: a person can belong to only one status categorisation: *single, married, divorced or widowed*. If information in synapse cluster M shows that John is single and information in synapse cluster P that he is married, the synapse from cluster P will replace the synapse from cluster M.

### 7.2.3.3 THE EXPECTATION CONSTRAINT

**Definition:** Expectation constraint
Let I is a synapse \((T, 'is', I_1, T_1, P_1, T_2, P_2)\). Let \(T_1\) is an instance of \(TT_1\), \(T_2\) is an instance of \(TT_2\). Synapse I satisfies expectation constraint if and only if synapse C in the form \((T, 'can be', TT_1, T_1, TT_2, T_2)\) exists.

This constraint ensures that each ‘is’ synapse is ‘expected’. This constraint is used for the specification of permitted properties related to a given type of individual. Example: If we store the synapse ('category synapse', 'can be', I, type, document, category type, document status) we permit synapses maintaining a relationship between a specific document and the category describing the document’s status.

### 7.2.3.4 VALIDITY

**Definition:** Valid synapse
A closed synapse is **valid** if and only if it satisfies both the synapse element type constraint and the expectation constraint.
**Definition:** Valid synapse cluster
Closed synapse cluster is *valid* if and only if all its synapses are valid and if it satisfies the synapse cardinality constraint.

7.2.4 Synapse classification task

If the synapse is closed, its classification depends only on its relationship to synapse cluster M. There are three possible situations:
- the synapse is present in M and is known (it is assigned to the set K)
- the synapse is not present and its addition to cluster M will not violate the cardinality constraint (it is assigned to the set U)
- if neither of the above applies, it will be assigned to the set R

If the synapse is open, its classification depends on its variables instantiation.

**Definition:** Deterministic synapse
We will call a synapse *deterministic*, if one of the condition listed below is true.
- There is one instantiation only, where the synapse can be assigned to the set K.
- There is one instantiation only, where the synapse can be assigned to the set U.
- There is one instantiation only, where the synapse can be classified under the set R.

**Definition:** Non-deterministic synapse
We will call a synapse *non-deterministic*, if it is not deterministic.

7.3 Correct and optimal solutions

7.3.1 Solution correctness

The main task for the split cluster algorithm is to solve all variables from the synapse cluster P by an appropriate instantiation. Not all instantiations are, however, acceptable.
Definition: Correct split cluster solution
The split cluster solution is correct if and only if after combining the solution with cluster M, it forms a valid synapse cluster.

Example: We cannot instantiate variable X from the example from chapter 7.2.2 with some device as it violates the expectation constraint – it is not expected that a device can be married.
Cluster validity ensures that resulting synapse cluster M will be consistent. The concept of cluster validity is based on the type system. This makes the validation process efficient (this is one important reason why we have supposed 1,1:0,M cardinality in type/instance relation in section 6.4.1). Not all constraints, however, can be expressed in terms of the type system. More complicated constraints can be expressed by rules.

7.3.2 Solution optimality
As we have seen from the motivation example in chapter 7.1, there are many possible correct solutions. Some of them are highly improbable – such as, where the word ‘person’ means something different from the type person already known, where the words ‘date of birth’ denote a new attribute with the same name as an existing attribute ‘date of birth’ etc.
Some solutions are better than others. The measurement of solution quality can be based on Occam’s razor principle. When comparing two solutions we will prefer the solution needing fewer new synapses. We can also take the quality of information based on knowing as our viewpoint (see section 2.3.1). It will prefer those interpretations, which bring less rather than more information.

Definition: Solution factor
The split cluster solution factor is the sum of |U| + |R|, where |U| is the number of unknown synapses and |R| is the number of replacement synapses in split cluster solution.

Definition: Optimal solution
In the list of solutions L, the split cluster solution S is optimal if there is no solution in L, which has a lesser factor than solution S.
This definition of optimality can be described as a conservative approach – “if you do not have to think about new things, then don’t do it“.

7.4 NAIVE ALGORITHM

Let’s describe the naive algorithm for the split cluster problem. The algorithm is quite inefficient. It is, however, easily understandable and helps us to illustrate several aspects of the split cluster problem.

7.4.1 ALGORITHM SPECIFICATION

The naive algorithm works using the following steps:

1. Assign all synapse from cluster P to the set N
2. Try to find a deterministic synapse in the set N
   a. If the valid deterministic synapse is found, instantiate its variables. If the synapse is known, store this choice point to stack S together with respective synapse. Repeat step 2.
   b. If the deterministic synapse is found but is invalid, backtrack to the last choice point from stack S, pop the respective synapse and re-instantiate its variables so that the synapse now belongs to the set U (i.e. adds a new individual). Repeat step 2.
   c. If the deterministic synapse cannot be found store the actual synapse classification as an item in the list L of possible solutions (see 7.2.1), backtrack to the last choice point from stack S, pop the respective synapse and re-instantiate its variables so that the synapse now belongs to the set U (i.e. adds a new individual). Repeat step 2.
3. If stack S is empty, stop the computation

7.4.2 ALGORITHM EXPLANATION

The result of this algorithm will be a list of all correct solutions showing how synapse cluster P can be split into sets K, U, R, N. Let’s now focus on closed solutions only. We can select optimal solution(s) according to the solution optimality criterion.

The naive algorithm is based on the fix point method – it stops if no progress can be made in finding a solution. A solution is progressed by the instantiation of a synapse – non-deterministic synapses can become deterministic. Solutions found using a naive algorithm are valid – the validity of each synapse is checked in step b.

Let us now discuss steps b and c, where the algorithm backtracks and re-instantiates synapses assigned to set K such that they now belong to set U. This means
that if the algorithm discovers that the solution developed will lead to an invalid synapse cluster, it solves this inconsistency by adding a new individual (re-instantiate its variables so that the synapse belongs to the set U).

This method can be illustrated by the following example:

- We are talking with a friend of ours about an individual called Hector.
- We think that we are talking about the person Hector we both know.
- Our friend tells us the Hector now very often barks.
- This violates the expectation constraint – we don’t expect people to bark!
- We solve this inconsistency be adding a new individual – instance of the type dog called Hector.

7.4.3 ALGORITHM DRAWBACKS

Let’s now discuss the drawbacks of the naive algorithm.

7.4.3.1 IGNORING THE SPECIAL ROLE OF NAMES

UIR using this algorithm will behave similarly to an advanced yogi. It will not differentiate between things – all things will be one for it. Names will be only different labels for this one thing. This way of understanding may be very pleasant for the yogi itself; it is not, however, a realistic method of communication between the yogi and other human beings. To prevent this kind of understanding, we have to introduce ‘new name assumption’.

Definition: New name assumption
If the unknown name appears, it denotes an unknown individual.

Please note, that this assumption does not exclude synonymy. An individual can have more than one name. In this case, however, we have to make a specific declaration stating explicitly that we are giving another name to an existing individual. As human beings we are able to communicate without this assumption, although it makes our communication unclear.

Example: If we talk about our friend Josef and during our dialogue begin to talk about Joe, it cannot be clear if Joe is Josef’s nickname or if Joe is another person. Whether Joe is Josef’s nickname or another person will only be clear from the relevant context.
Another problem can occur with new names. Example: Let us assume that the person we are talking to doesn’t know who John is. We will not be making ourselves clear if, when we say “John is my colleague and John owns a beautiful cottage” and at the same time we are referring to two separate persons – the first John being a colleague and the second John being another person who owns a beautiful cottage.

We will solve this problem by introducing the following assumption

**Definition:** Two new names assumption
If the same new but unknown name appears twice in one message, it refers to the same individual.

This assumption is also typical of normal human communication.

The last name assumption is not necessary but is useful in reducing the number of solutions.

**Definition:** Unique type name assumption
Two different types cannot share the same name.

It says, that ‘basic words’ denoting types are unambiguous. This assumption is more restrictive than in our real world but helps UIR to function more effectively and to avoid misunderstandings.

These assumptions are not restricted to the naive algorithm only. They are helpful in general. Thus we will use these assumptions in the following.

### 7.4.3.2 Incorrect work with context

Let’s now focus on how the naive algorithm recognises deterministic synapses. It uses only actual memory content. It cannot distinguish between synapses whose determinism is caused by facts not dependent on actual memory content (i.e. not dependent on the state of the world) and synapses, whose determinism is caused by the actual situation. Let’s illustrate this drawback using an example: Example: Let’s focus on the following rule: “all graphic designers must be assigned to more than one project”.

---

83
This rule is containing the following IDEX

Person with properties:
* function is graphic designer

It identifies persons, who are graphic designers. If there is only one graphic designer in UIR’s memory – e.g. Jay - the rule will be interpreted by the split cluster as “Jay must be assigned to more than one project”.
In this case the naive algorithm will choose the way, which is deterministic in an actual situation only. The algorithm will not make this error, if it is able to recognise whether the determinism occurs in the actual memory state only.
The second problem is that the naive algorithm ‘is afraid’ to choose ways, which look non-deterministic at this moment but which can be recognised as being deterministic later on.

Example:
Synapse cluster P is constructed from IDEX:

Person with properties:
* has name Jay
* is ill
* is male

Memory (the cluster M) will contain records about two persons with the name Jay. One will be male; the other female. There is also another person Paul, who is male. None of them is ill.
A naive algorithm cannot solve this situation – it cannot recognise who becomes ill - because it is too worried about making a non-deterministic step. The synapse expressing illness is not deterministic – there are four possibilities of who can be ill – Jay-male, Jay-female, Paul, new individual. The synapse expressing name is not deterministic – there is Jay-male and Jay-female. The synapse expressing gender is also non-deterministic – there are two males – Jay and Paul. But the solution exists and is obvious – it is the male Jay who is ill. If the naive algorithm were to have the courage to make a non-deterministic step, it too would find the solution.

7.4.3.3 SINGLE SYNAPSE APPROACH

The previous example illustrates another drawback. The algorithm is focused on single synapses only. This fact can prevent the algorithm from finding solutions, where it is necessary to concentrate on a whole bunch of synapses.
7. UIR Knows What It Knows

7.4.3.4 No Domain Knowledge is Used

A naive split cluster algorithm is independent of domain. Let’s illustrate this fact using the example:

Patient with properties:
* has name Jabez Wilson
* his medical record number is 7453-3423
* has diagnosis R43

A naive split cluster algorithm cannot use the fact: a medical record number uniquely identifies a patient. If there is another Jabez Wilson in the database with the same diagnosis, the algorithm will replace his medical record number with the number 7453-3423 and will merge the two persons together. We need an algorithm, which can use the domain knowledge.

7.4.3.5 Inefficient State Space Traversing

Checking whether a synapse is already deterministic is very often done uselessly, because no instantiation concerning a given synapse was ever made. Because it does not use information about variables shared between synapses, backtracking is also very inefficient. Let’s assume the following sequence of steps (S -> K means that synapse S was assigned to the set K, S->U is analogical):

\[ S_1 \rightarrow K, S_2 \rightarrow K, S_3 \rightarrow K, ..., S_{10} \rightarrow ? \]

Whilst checking the \( S_{10} \) synapse an invalid instantiation occurs. It is necessary to backtrack. Suppose the synapse \( S_{10} \) shares its variable with \( S_1 \) and the invalidity of \( S_{10} \) is caused by the bad decision to assign \( S_1 \) to the set \( K \). The naive algorithm will first change its decision about \( S_9 \), then about \( S_8 \) and so on, before it finds the right way: \( S_1 \rightarrow U \).

7.5 Split Cluster Problem Complexity

We have discussed how the naive algorithm traverses solution space. Let us focus on how large the solution space is and on its relationship to the split cluster input.

Assessing split cluster problem complexity can be done using synapse element variables. Let \( V_1, ..., V_p \) be variables present in the synapse cluster \( P \). Let \( |V_i| \) be the number of possible values of \( V_i \), independent of the values of other variables. Assessment of the greatest possible number of solutions is \( |V_1| \times ... \times |V_p| \). This number can be expressed as \( k^n \), where \( k \) is the average number of variable instan-
7.6 UIR Knows What It Knows

The ‘islets of clarity’ algorithm

We have identified several serious drawbacks of the naive algorithm and we have discovered several inspirations. Let us now concentrate on a robust solution to the split cluster problem.

We will introduce an algorithm called ‘islets of clarity’. The naive algorithm is based on the synapse classification task. The new algorithm is based on the variable instantiation task and pattern matching techniques. The whole algorithm is based on switching between non-deterministic and deterministic steps. The non-deterministic step makes a decision and in deterministic step the consequences of that decision are deduced. Let us now describe the deterministic steps.

7.6.1 Deterministic steps

The name ‘islets of clarity’ describes metaphorically the main idea behind the algorithm. It focuses on variables and tries to find a variable, where there is only one sure instantiation. The deterministic steps then spread clarity to those variables, which occur in the same synapse as the clear variable. There can be several centres of clarity and clarity spreads from these centres to other synapses creating clarity islets, which finally form a ‘clarity continent’.

The algorithm’s flow is not fixed, it is based on recognition of and reaction to specific events.

The specific event is one of the following:

- A variable, which can be instantiated only with a new individual is found. This event is called a U-event (variable is solved by Unknown individual).
- A variable, which can be instantiated with just one individual is found. This event is called K-event (variable is solved by Known individual)
The situation when a variable can be instantiated by more than one individual causes no event. There are several specific functions – so called solvers, which react to events and check the structure of the clarity islet connected with the event. If the structure matches the solver’s conditions, the solver then tries to enlarge this clarity islet. There are two classes of solvers: U-solvers and K-solvers. A U-solver reacts when a U-event appears; a K-solver reacts when a K-event appears. Solver conditions are expressed using open synapse clusters, which try to match synapses from cluster $P$. If the solver successfully and correctly (with respect to the cluster validity condition) enlarges the clarity islet, it classifies the relevant synapses and returns variables, which the solver solved. This will cause other U-events and/or K-events to which other solvers can react.

7.6.2 Specific Solvers

Let’s now describe the most typical solvers:

7.6.2.1 Basic U-Solver:

This solver reacts to the situation where a new individual appears. It uses the fact that each synapse containing a new individual is unknown. This solver will only assign respective synapses to the set $U$. It will not solve any other variable and thus will not generate new events.

7.6.2.2 Enhanced U-Solver

This solver also reacts to a U-event. It will only function with synapses where the following assumption can be used: if a synapse contains a new individual in the $P_1$ element, then the $P_2$ element will also contain a new individual. Example: This assumption can typically be used with associative entities (in the same sense as in ER terminology). An example can be the relationship between delivery, customer and supplier with the semantics: deliveries from given supplier to given customer. If we don’t know the given customer, we cannot know the respective delivery. Another example can be a synapse, which expresses the relationship between a state and its president. If the individual Lesotho is new for you, the president of Lesotho will be also new for you.

7. UIR Knows What It Knows
7.6.2.3 NAME SOLVER

This solver reacts to a K-event – when a description expressing an individual name appears. It will use the new name assumption and the two new names assumption to identify the new individual. In this case the name solver generates a new U-event.

It can also use unique type name assumption to identify a known type. In this case it will generate a K-event.

If the given name is not the name of a type, it will add enumeration constraint to the respective variable. Enumeration constraint expresses that a variable can be instantiated only with an individual present in a given list of individuals. This enumeration will contain all those individuals with the given name and also the possibility that it is the name of a new individual.

7.6.2.4 TYPE SOLVER

The type solver reacts to K-events. The type solver solves the situation, where a variable is constrained by an enumeration constraint, but where there is only one type compatible possible.

7.6.2.5 UNIQUE PROPERTIES SOLVER

The unique properties solver reacts to K-events. The solver uses domain dependent information where certain properties uniquely identify an individual. An example of such a property can be a medical record number or bar code. This fact is very close to the concept of keys in relational databases. The property of a property to form a key is of course stored in UIR’s memory using synapses.

7.6.2.6 STABLE PROPERTIES SOLVER

The stable properties solver reacts to K-events. The solver uses domain dependent information too. The solver is used to solve contradictions between a message and UIR’s memory.

UIR in general uses two methods to manage contradictory information. (1) The first way is to replace information stored in memory with information contained in a message. An example: The memory contains the information that a given person is single, whilst the message contains the information that this same person is married. In this case, the information from the message will replace the information in the memory (R set is filled). This task is done during synapse validity checking (this is not a job of solvers). (2) The second method is to solve the contradiction by introducing a new individual. We have already discussed this
method in the situation where a non-valid synapse appears. There is, however, yet another situation where a contradiction can appear in a valid synapse and the strategy of synapse replacement is not suitable. This is a job for a stable properties solver.

Stable properties solver works on the assumption that certain properties cannot be changed “just like that”. These can be called ‘stable’ properties and they are part of domain knowledge. Example: One such stable property is gender. The solver assumes that this property is not ‘replaceable’. It is not very common for a man to become a woman. If this situation does occur, it is necessary to declare this situation explicitly.

7.6.3 NON-DETERMINISTIC STEPS

A non-deterministic step is much easier than a deterministic one. It is necessary to select the variable – decision point – and to choose one possible instantiation of this variable. The strategy actually used is to select a variable with the minimal number of possible instantiations. Other strategies are also possible - they will typically use domain knowledge.

7.6.4 ALGORITHM SPECIFICATION

The islets of clarity algorithm consists of the following steps:

1. Cause initialisation K-events by identifying all descriptions from cluster P.
2. Derive the consequences of the K-event or U-event using solvers reacting to specific events. This is a deterministic step. It will make progress on actual solution (actual K, U, R, N splitting).
3. Check the actual solution and measure its factor
   a. If the actual solution is closed, store actual solution to the list L with its factor, backtrack to another non-deterministic possible decision and go to step 4. If no non-deterministic decision can be made, stop the computation.
   b. If the actual solution is not closed and its actual factor is equal or less than the factor of the actual known best solution then go to step 4.
   c. If the actual solution is not closed and its factor is greater than factor of the actual known best solution, backtrack to another non-deterministic possible decision and go to step 4.
4. Make a non-deterministic decision and go to step 2
7.6.5 ALGORITHM EXPLANATION

The algorithm begins with descriptions, because they are constant in cluster P (step 1). They form initial clarity islets. Clarity is spread in deterministic steps (step 2).
The solution factor used as a threshold measured in step 3 is used to stop the development of a solution, which cannot be optimal (step 3c). The solution factor grows monotonously; there is no step, which can switch the new synapse to a known synapse. That is why step 3c is correct. If there is a possibility to make a non-deterministic choice, the choice is made (step 4).
The algorithm always stops. Solutions identified by the algorithm are valid – because validity is checked during deterministic steps.
The algorithm solves all the listed drawbacks of the naive version. Domain knowledge is presented in UIR’s memory and solvers use this knowledge for their function. Domain knowledge reduces the number of decision points. In the naive version decision points were all variables present in the cluster P. The ‘Islet of clarity’ does not decide on points where the appropriate instantiation can be derived from other variables.
The ordering and precedence of solvers can be tuned to obtain better results in a given domain and situation. The principles of genetic algorithms can be used.

7.7 SPLIT CLUSTER SLIDING ON THE IDENTITY AXIS

There are two ways in which a split cluster can misunderstand. (1) It can see two or more different individuals as one or (2) it can see the one individual as two or more different individuals. This misunderstanding is caused by missing or incorrect domain knowledge. It can be described as an inappropriate UIR position at a so-called identity axis in a given situation (see section 6.5.4 and the following text). But what does inappropriate mean? What are we human beings doing when we make abstractions or when we forcibly split things up to understand them better? We are in fact sliding on identity axis. Let us describe how this ability can be implemented in UIR using the ‘islet of clarity’ algorithm.
The deterministic part of the ‘islets of clarity’ algorithm can be divided into two parts – a general computing scheme and solvers. If we switch a given solver off or change its functioning, we change UIR’s power of resolution. Some solvers also use domain knowledge, which can be deformed intentionally.
This manipulation can cause UIR to stay at the top of identity axis and to behave in the way we have already mentioned – just like in advanced yogi. All things will be seen as one individual. The bottom of the identity axis can be also reached using this manipulation. UIR will then see all things in a different way and will be extremely pensive and suspicious. The role of a split cluster can also be described as an ‘identity maker’. Enlightened or suspicious robots are probably not very
useful; the ability to slide on the identity axis, however, is.
Let us describe this ability when absorbing information about patients. UIR, for example can unify patients having the same diagnosis and disease progression in one individual. Please note that this in fact differs from the process of abstraction. This one individual will have a number of names, a number of dates of birth etc. The properties of patients absorbed are all assigned to this ‘artificial’ individual. We can make an abstract pattern of this individual by pruning away unimportant properties. This technique can also allow UIR to find discrimination properties sufficient for distinguishing individuals.
On the other hand, if we modify solvers to slide down at identity axis, a patient record will be split over separated individuals – for example individuals holding information on health insurance records and individuals holding clinical records.
Please note that this separation is typically present in disintegrated data sources. UIR can use this ability to test experimentally, how a record was separated. It can then use this information during the integration process.
UIR can slide along an identity axis when perceiving given data in much the same way, as our eye moves very quickly obtain spatial perception. It gives UIR the opportunity of perceiving cyberspace in a manner beyond the capabilities of us human beings.
8 To focus attention

UIR’s ability to concentrate can be described as an ability to cause those synapses relevant to an actual situation to glow. A lot of UIR use-cases use synapse selection driven by attention. For example, using the ability to concentrate in a question-answer-question use-case will mean that the resulting synapse glow will contain information relevant to the user’s focus of attention.

8.1 Synapse glow

We need to express formally a diffusion of attention on a given synapse cluster.

Definition: Synapse glow

A synapse glow is every tuple \((S, A)\), where \(S\) and \(A\) are synapse clusters and \(A\) is a cluster containing synapses, whose first synapse elements are identifiers of synapses from cluster \(S\) and second synapse elements express the amount of attention focused on a particular synapse from cluster \(S\).

A synapse from synapse cluster \(A\) can express the amount of attention in the following way: (1) The second synapse element is a description expressing in some way the amount of attention. (2) The second synapse element is a category expressing in some way the amount of attention.

There are several other ways, but we can now restrict them to these two.

8.2 Object maker

We pointed out in section 6.3. that objects are subjective. Object is just a pragmatic weaving together of synapses. UIR’s ‘object-maker’ organ (section 5.3) is responsible for these pragmatic creations.

Let us start with the description of the object-maker function independent of context. The input for an object maker is one single individual. The object maker uses the global ‘objectification’ rules and the result is a synapse cluster containing the input individual with its selected nearby synapses. These synapses will typically contain information about an individual’s names (name synapses), categories to which the individual belongs (category synapses), nearby connections with other individuals etc.

The attention viewpoint of a context-independent object-maker can be de-
scribed as boolean rule-based attention glowing – the synapse selected by the rules is lit and is included in the resulting synapse cluster. In this case measurement of the attention assigned is binary (lit or not lit).

The following figure illustrates UIR’s reaction after asking for more data on the central node in the view – it gives us more information about the focused task (its author, its status and the relevant project):

![Figure 21: Example of object making](image)

The context-independent approach has several drawbacks. For example, if the individual is a specific patient, the resulting synapse cluster will be quite large. It will contain all patient attributes, all connections to patient treatments, medications, health insurance records etc. To obtain a useful view, we need to use context – for example, we are interested in cardiological aspects and operations. We need to filter out synapses concerning this topic. For the specification of the filters we will use contexts and attention.

By specifying the context, we express how we want to concentrate our attention. We typically use parts of UIR’s conceptual model (cardiological aspects, operations) for context specification. These parts can be translated into synapse clusters.
Let’s summarise our situation:

Input:
- Synapse cluster $S$ containing individual $I$
- List of synapse clusters $P_1, ..., P_n$ (the context). Each $P_i$ has attached number $p_i$ expressing the amount of attention to $P_i$.

Output:
- Synapse glow $G = (S, A)$, where $A$ is synapse cluster expressing attention distribution to synapses from $S$ according to context representation $P_1, ..., P_n$.

There are numerous ways of diffusing attention over a synapse cluster. Let us describe the most straightforward method: **Query by $P_i$ over $S$**. Add amount of attention $p_i$ to all synapses focused by $P_i$.

This is the easiest way of diffusing attention. Synapse cluster $S$ can be the whole of UIR’s memory. Synapse clusters $P_i$ identify interesting synapses. Example of $P_i$ can be synapses expressing attributes belonging to the category ‘cardiological aspect’. Synapse clusters $P_i$ can overlap, synapses in the area of overlap will gain the sum of attention from $P_i$. The measure of attention on each synapse from $S$ also allows UIR to sort synapses by their relevance. We can choose a threshold, which will divide a synapse cluster into those synapses to be presented (e.g. to the user) and those synapses, which will not be presented. We can move with this threshold according to the desired size of solution (cf. section 4.1.4. and 4.1.5.).

There are several technical problems (for example output synapse glow does not have to form a connected synapse cluster). There are also more sophisticated methods for diffusing attention. More interesting, however, than elaborating on the detail, is to extend the ‘object maker’ into a ‘connection weaver’.
8. TO FOCUS ATTENTION

8.3 CONNECTION WEAVER

An object maker concentrates on synapses near to the one single individual selected. Connection weaver input is a list of selected individuals. It concentrates on synapses on paths between these individuals. It will typically create composite objects such as a person and his tasks, patient and his treatments and relevant doctors.

Input individuals can be *mentioned* during the process of weaving connections or they can be *used* to identify other individuals. The difference between *mentioning* and *using* is also about attention diffusion. A given individual, for example type *patient*, can be the final target for attention diffusion (in the case of mentioning) or it can be used as a route for diffusing attention on type instances (in the case of using). This difference has to do with the decision whether the *type/instance* connection is ‘live’ in this case or not. In the situation where it is live, the attention spreading rule is implemented using synapse cluster. The synapse cluster will contain just one single type/instance synapse with a free variable on the instance element. If the synapse cluster is then evaluated, the result will include all individuals lit by the attention on this type. In this case, the synapse cluster is functioning as an attention teleport.

We can also use other connections to diffuse attention such as *specialization/abstraction* (where the focus on an individual’s specific properties will be diffused over its more abstract properties), *part/whole* (where the focus on a specific object will cause attention to be diffused over its parts), *predecessor/successor* (attention will be diffused from specific object to its history) etc. The synapse cluster can, of course, contain more than one synapse – there can be more sophisticated conditions.

The fundamental connections can be used as a specification of world view. We can, for example, concentrate more on time aspects (more closely to process view – predecessor/successor connections) or we can concentrate more on ownership (the ‘lawyer view’ – *owner/ownership* connections).

The following figure illustrates UIR’s answer to a request using connection weaving:
8. TO FOCUS ATTENTION

I want to see all deliverables and persons who are guarantees.

Figure 22: The connection weaving output

8.4 ATTENTION DYNAMICS

There are several important questions about attention. Is attention burned up or is it constantly present in UIR's memory? If it is burned up, where is the attention source? Is the amount of attention in the system constant or does it depend on situation? If it depends on situation, what is the relationship between situations and the amount of attention?

At this stage there are many more questions than answers. It would seem better to model attention as an energy that is burned up. The main source of this energy is the user himself. The user is directing this energy through a virtual knowledge network. Typically it can be made by the user's questions.
9 VISIONS

* UIR as an extrapolation of Google * problem of personalisation * knowledge garden *

9.1 UIR AND GOOGLE

The functionality of UIR can perhaps be seen as an extrapolation from a functionality of Google. The most important improvements are in solutions to the problems “how to ask” and “focusing”. Solutions to these problems need personalisation – a benefit and yet on the other hand a threat. A personalised tool becomes something much more than just a tool because it contains something of us. How much? That will depend on us. It can become our cyberspace mind. It is a tempting idea but also a very dangerous one. Our cyberspace mind can be influenced in such ways that the ability of today’s mass media to influence us will seem like child’s play in comparison. Who can protect us? Damn recursion – the very same tool.

9.2 KNOWLEDGE GARDEN

This vision began with an item of news, which I heard somewhere. It said that there was a specific kind of plant, which could be used to clear minefields. The principle is very simple – the plant is sensitive to iron. If its roots touch the mine, the flower of the plant will change colour. All that needs to be done is to sow the seeds of this plant over the minefield, sit back and wait!

The relationship between the appearance of this plant and the environment is quite involved. The fact, that some plants occur in specific areas also give us a lot of information about the environment.

A virtual knowledge network can be seen as a field, which has huge potential, which we are not yet able to utilise. It is very probable that in the data available we have the key to cure cancer, to recognise the birth of a tsunami or the preparation of a terrorist attack. Let’s imagine the following: a virtual knowledge network is an environment, various kinds of virtual plants are growing in this environment, human attention plays the role of the sun in this world. Let’s work out this metaphor:

A species of virtual plant are questions and a specific plant is the answer. We have not yet been able to formulate these questions, as our imagination is not up to this task. How difficult will it be to design an artificial apple tree from scratch? A better way is to choose from a variety of plants and improve those species, which fit in with our needs. Let’s compare plant tissue transporting nutrients with synapse tissue transporting attention according to context - environment. A suc-
cessful virtual plant can be as complicated as a physical plant. Let us now focus on the appearance of our virtual plant. It will be made up of various visualisation elements projecting some part of a virtual knowledge network. Some plants will look like mind-maps with forms as fruit, others will look like a fish-bone diagram where the main axis will reflect one of the fundamental connections etc. To be able to understand this train of thought better (remember section 4.1.5) and let’s look at a specific document repository shown as a plant.

Figure 23: Visualisation of document repository (adopted from [WALRUS])

Let us now look at attention as we look at the sun. If a human being looks at a certain plant – answer – he causes the plant to grow. The plant can then multiply – generate more answers to the same kind of a question – and be more successful as a species.

Let us transfer our metaphor to a doctor’s view on data. An individual doctor has his own group of patients, his own special interests, his own preferences and his own way of thinking – in the other words his own virtual knowledge garden. He prefers some species to others and one style of garden to another.
Sometimes he sees a newly evolved plant or plants a colleague’s flower. There is no need to customize the garden, it’s all about liking and disliking various kinds of plant.
10 QUESTIONS

* open end *

10.1 SEEKING THE FINAL SOURCE

Where is the source of attention? Is this source limited? Where is the focusing force of attention? Is it a constant stream or a discharge of some attention source? Can attention focusing be recorded in its whole complexity and be replayed; how does it relates to deja-vu? Can some intelligence be found in structure? Is it the combination of structure and attention? What is then attention itself?

10.2 IS UIR TOOL OR YOU?

Imagine you are making intensive use of UIR. Is it possible for UIR to become you or for you to become UIR? What is the difference between a tool such as UIR and our own eyes? Are we not in fact the users of just such a tool embedded in our brain? If it so, who in fact are we?
References


Kaspar, J: Jak se delal a jak je udelan transakcni system eBanky, Datakon 2001, Czech Republic


Prochazka, F., Stanicek Z.: US patent application Nos. 60/785,055 entitled UIR UNIVERSAL INFORMATION ROBOT, owner: eTrium Corporation

Prochazka, F., Stanicek Z.: US patent application Nos. 60/785,142 entitled INTEGRATION BY MEANS OF UIR, owner: Masaryk University

Prochazka, F., Stanicek Z.: US patent application Nos. 60/785,046 entitled UIRON UNIVERSAL INFORMATION ROBOT IN ONCOLOGY, owner: Masaryk University

Procházka, F.: eTrium-ISMS – Univerzální nástroj pro správu přístupových práv, Proceedings of the conference Datakon 2003


List of Definitions

Definition: Container (#Object) 59
Definition: Container (#Connection) 59
Definition: Container (#Operation) 59
Definition: Container (#Category) 59
Definition: Container (#Rule) 59
Definition: Synapse 62
Definition: Closed synapse 63
Definition: Open synapse 63
Definition: Synapse cluster 64
Definition: Closed synapse cluster 65
Definition: Connected synapse cluster 65
Definition: Synapse cluster sequence: 65
Definition: Closed solution 76
Definition: Open solution 76
Definition: Synapse element type constraint 77
Definition: Synapse type cardinality constraint 77
Definition: Synapse cluster cardinality constraint 78
Definition: Expectation constraint 78
Definition: Valid synapse 78
Definition: Valid synapse cluster 79
Definition: Deterministic synapse 79
Definition: Non-deterministic synapse 79
Definition: Correct split cluster solution 80
Definition: Solution factor 80
Definition: Optimal solution 80
Definition: New name assumption 82
Definition: Two new names assumption 83
Definition: Unique type name assumption 83
Definition: Synapse glow 92
List of Figures

Figure 1: Pattern of data-source description 16
Figure 2: Information systems 16
Figure 3: Registers 17
Figure 4: Data warehouses 17
Figure 5: Web services 17
Figure 6: HTML pages 18
Figure 7: XML files 18
Figure 8: Tools 18
Figure 9: Example of the Virtual Knowledge Network 20
Figure 10: User, UIR and data-sources 22
Figure 11: User, UIR and data-sources – schematic view 23
Figure 12: Example of a conceptual system 38
Figure 13: Example of combining visual elements 41
Figure 14: Schematic overview of UIR’s architecture 50
Figure 15: UIR’s organs. 52
Figure 16: Diamond Modelling Tool 58
Figure 17: Simplified synapse scheme. 64
Figure 18: Zachman framework 68
Figure 19: Fundamental recursive connections 70
Figure 20: Properties of fundamental recursive connections 71
Figure 21: Example of object making 93
Figure 22: The connection weaving output 96
Figure 23: Visualisation of document repository 98