BACHELOR'S THESIS | LUND UNIVERSITY

Inventory of Robotic Skills in the Knowledge Integration Framework Knowledge Base

Isabella Gagner

Department of Computer Science
Faculty of Engineering LTH

ISSN 1650-2884
LU-CS-EX 2018-23

Inventory of Robotic Skills in the Knowledge Integration Framework Knowledge Base

Isabella Gagner

July 2, 2018

Bachelor's thesis work carried out at
the Department of Computer Science, Lund University.

Supervisor: Jacek Malec, jacek.malec@cs.lth.se

Examiner: Elin Anna Topp, elin_anna.topp@cs.lth.se

Abstract

By using knowledge based robotics, the complexity of the tasks of AI enabled robots can be enhanced. However, the knowledge base used with the AI enabled robots at LTH is hard to understand and fully grasp for a human user. It lacks functions to, for example, sort out certain information about what the knowledge base consists of.

The goal of this thesis is to present the theory behind the knowledge base and start the development of a program dedicated to facilitate human interaction with the knowledge base and its contents.

The developed program queries the knowledge base and all its data, sorts out the required information and displays it in a browser.

Even though the developed program could be useful as is, it has a lot of potential and would benefit from more functionality added to it, such as a user interface to simplify entering required information or a graphical presentation of the contents.

Keywords: Knowledge based robotics, RDF4J, KIF, robot skill, ontology

Contents

Chapter 1
Introduction

The last couple of years, artificial intelligence (AI) used in robots has been a hot topic within the world of technology. As experiments advance rapidly, different potential areas of use of AI enabled robots have expanded as well. The main difference between ”ordinary“ robots and robots using AI is that the latter need to be able to make decisions about actions by themselves, whereas robots in the classical sense have all their actions programmed beforehand. In order for a machine to be able to make decisions by itself, it needs knowledge about how the decision could affect for example future decisions or current states [10]. This may be achieved by using knowledge based robotics, where the robot has online access to a knowledge base, from where it can download skills - which would be all the knowledge needed about how to perform certain tasks.

Knowledge based robotics exists because of the need to expand and refine the robots' assignments and working environments. Traditional industrial robots will execute the same movements day after day, year after year. These movements are precise and defined in advance, where the programming time of the robots is long, as well as the execution time. As the need for new types of robots expand, the variation of their assignments do as well. By using knowledge based robotics, the complexity of the assignments of the robots can be enhanced, but the programming time reduced at the same time [17]. This would be cost efficient in two ways - robots can perform high complexity tasks but do not need an expert programmer to program them.

1.1 Problem definition

The knowledge base developed at LTH is the result of several projects, carried through by several research groups, almost each and every one of these groups adhering to different semantic logic when representing the robotic skills. This has led to the knowledge base containing several skills, represented in a multitude of ways and not following any standard. As is imaginable, this leads to unnecessary confusion and complexity when trying

to make sense of what the knowledge base contains. In short, the question the users could ask themselves was: ”Given this knowledge base, what skills could my robot perform? “, and be unable to realize this immediately. The appearance of the knowledge base is not intuitive and the structure of it is hardly obvious even to the practised user, and even less to the uninitiated.

A program that would help the user to easily navigate the content of the knowledge base, displaying only relevant information that the user has requested, would be of much help. It could be used as a form of confirmation that the knowledge base is structured the way it is supposed to be, as well as helping users working with the robots and knowledge base visualize current skills located in the knowledge base.

1.2 Project goal

To simplify for the human user to see what parts of the knowledge base are relevant and add the functionality of being able to specify what types of information to show, an additional applet needed to be designed. The thesis is built around the design of this applet, or small program, by querying the database, retrieving the information and only displaying what was requested.

The goals of this thesis are therefore to simplify the user interaction with the knowledge base by;

- letting the user specify which information to be shown,
- presenting the result in a straightforward manner,
- keep the program design modular, to simplify possible future changes,
- have the program execute in a reasonable amount of time.

By letting the user specify what information to be shown, the user can easily see what parts of the knowledge base are relevant for their interests. A user could for example want to know where each robot skill is located in the knowledge base and what this skill consists of. Presenting the result in a straightforward manner means giving the user the information needed, careful to not present too much information as the risk of cluttering increases [1]. Keeping the program design modular simplifies future modifications and improvements. Having the program execute in a reasonable amount of time is also directly connected to the user's final experience with the applet.

1.3 Related work

Using knowledge bases to achieve AI in robots is an area full of research and there are a multitude of different approaches. RoboEarth presented in [22] has a knowledge base equivalent to the KIF knowledge base at LTH, where the goal of the project is to create a World Wide Web for robots. However, the approach used for RoboEarth is not suitable for industrial robots since the system has real-time requirements and since the software is proprietary [17].

Another knowledge based robotics project is KnowRob, presented in [19], however in the KnowRob project, the end focus has been autonomous personal robots, which makes

for different requirements than for industrial robots. The methodology used in [19] is also based of the OWL (Ontology Description Language), as is the KIF knowledge base.

A program called Protégé allows users to build ontologies, providing logic reasoning and thus aid in building a knowledge base [11]. This is useful as it can provide a simple visualization of the content in a knowledge base, however the program comes with restrictions. Should a user want to see a sequential function chart (further explained in section 2.1.7), an external program or applet needs to be developed. When importing the knowledge base into Protégé, some information is also lost, such as context ID:s (explained in section 2.2). Another restriction is that all information to be handled by Protégé should be expressed in pure OWL, however not all content at the KIF knowledge base at LTH fulfills this, which leads to additional problems. An applet that complements current software such as Protégé by visualizing the content of the knowledge base as well as giving the potential of further additions is presented in this thesis.

In [17], the author introduces a knowledge-based robotic system architecture, which is the same one that the applet for this thesis is built on. The master thesis of Jacobsson [5] presents an ontology hierarchy which new skills added in the KIF knowledge base should follow. It is from these two papers that the theory has been built of how to best design the applet of this thesis.

1.4 Thesis outline

The following chapters are structured as follows; chapter 2 will present some theory behind knowledge based robotics and the way the knowledge is stored, specifically in the KIF knowledge base at LTH. Chapter 3 will present the outline of the program design and some details about the implementation (such as frameworks and integrated development environments). In chapter 4 we will evaluate the resulting program and discuss different methods and problems realized along the way, and this is followed by a conclusion in chapter 5.

Chapter 2
Background

This chapter will introduce the reader to the concepts used in knowledge based robotics, as well as the specific structures of the LTH knowledge base that this thesis is built on.

2.1 Vocabulary

For readers not yet familiar with the terms often used in contexts surrounding the semantic web (further explained in section 2.1.1), the following section will present some basic concepts and go through the outline of the theory behind knowledge representation.

2.1.1 Semantic web

The word semantics means "the study of meaning" [6], most often referred to in linguistics - thus concerning the meaning of linguistic expressions. These expressions can be in natural languages such as English or Swedish, but also artificial languages, such as programming languages [20].

As the content on the internet web expands exponentially, so does the need to categorize and integrate the content. It is further explained in [15] that scientists, among others, would benefit from expressing the vast amount of data on the web by using some sort of semantics. Linking data with the use of semantics gives us the potential to see connections between different categories of content that otherwise might have gotten lost. This could aid for example to connect different areas of research by providing "the bigger picture".

The first report on the expression "Semantic Web" in the Scientific American was released in 2001 [15]. As opposed to the web that most of us are used to - the web that was developed for humans - the Semantic Web contains data for computers to query and manipulate [15]. The content on the semantic web is made up by ontologies (see section 2.1.5), expressing the data using RDF (see section 2.1.2), giving the content meaning using semantics. The computers can then use the meaning when querying the Semantic Web.

2.1.2 Triple store RDF

RDF stands for Resource Description Framework, and is a framework for expressing resources. These resources could be anything (i.e. documents, abstract concepts), where relations between each resource are expressed using properties. The property is directional - from subject to object - and can be expressed in the same way as the resources (see section 2.1.4)[16].

These three parts, namely the subject, predicate and object, make up a statement in the form of a triplet. The resources are the subject and the object, and the predicate is the relation between these resources [16]. Figure 2.1 illustrates an example of how a statement might be constructed, to give the reader a sense of the form.

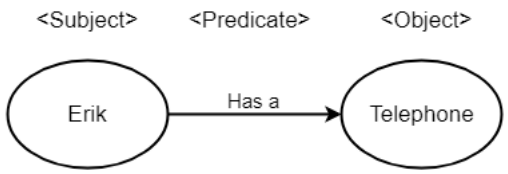

Figure 2.1: A simple example of a triple, a statement consisting of a subject, predicate and object. The arrow between the subject and object indicates the predicate.

What makes RDF statements powerful, as with linked data in general, is that they convey a lot of information about relations between things - knowledge. If you have information about one thing, the relations can lead you to discover more related data [16].

2.1.3 RDF classes

We will keep building on the example in figure 2.1 to give the reader an understanding of how this knowledge could look with more information. In figure 2.2, more resources have been added and also the relationships between them. Of course, these are just examples to give the reader a better understanding of the concept and the examples may not necessarily be useful in relation to robotics.

Those familiar with object oriented programming may recognize the similarities between what is presented in figure 2.2 and how classes work in for example Java [3]. As an example, Erik's mobile telephone (the Telephone in figure 2.2) is an instance - it is of type *Telephone*. The class *Telephone* in turn is a subclass of the class *Machine*. This would also imply that Erik's *Mobile Phone* is a *Machine*, as it inherits this from *Telephone*. Even though there are certain similarities between RDF classes and object oriented programming, the structure in object oriented programming is usually stricter than the classes expressed in RDF [3].

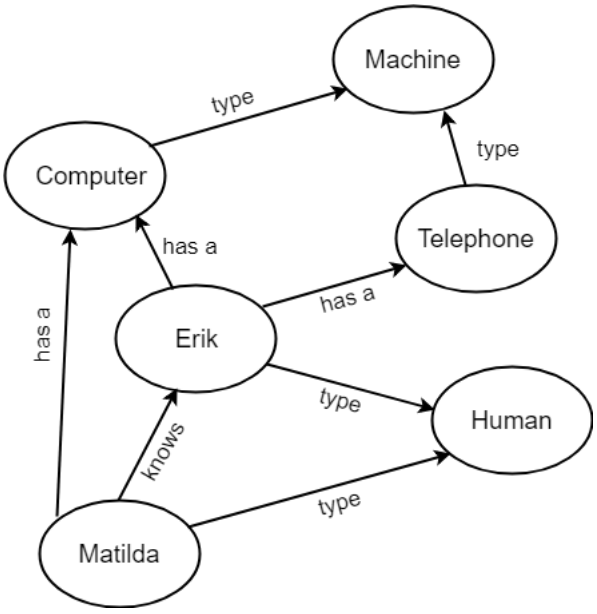

Figure 2.2: An example of how different statements (resources) could be linked together.

2.1.4 IRI:s

It is through IRI:s (or, in older versions, URI:s) that the resources and the relations between them are expressed, formats confusingly alike URL:s. In fact, a URL is a sort of IRI, which explains the similarity. IRI stands for International Resource Identifier, which also explains quite well what it is - something that identifies resources [16]. These resources can be either physical or abstract, and the IRI:s are a set of characters that identify these. To show this with an example, consider the statement "*<Matilda> <Type> <Human>*", that is shown in picture 2.2. Using IRI:s, this could be expressed as following:
Subject: `<http://example.org/Matilda#me>`
Predicate: `<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>`
Object: `<http://xmlns.com/foaf/0.1/Human>`

The different parts of the IRI refer to different things. For example, the `http://` part contains information about the used scheme. The following characters, `example.org` refers to the domain name, and the rest - `Matilda#me` - is the path. As such, the IRI points to where the actual data is kept on the server, from the server root [21].

IRI:s are global, which means that anyone can use and reuse the same IRI:s [16]. What this means is that IRI:s that look the same refer to the same resource, even if they are located in different repositories or even in different knowledge bases. For example, the predicate IRI in the example above is frequently used in the knowledge base at LTH and is a World Wide Web Consortium standard, so it is most likely used in completely different knowledge bases as well.

2.1.5 Ontology

There are several definitions of what an ontology is, but one widely used one is defined by Tom Gruber [4]. On his website he states, with the support of the Encyclopedia of Database Systems, that ”an ontology defines a set of representational primitives with which to model a domain of knowledge or discourse“. This means that a set of classes referring to knowledge about one domain, as well as relations between the different classes, will make up an ontology. As an example, the figure 2.2 could be thought of as an ”Erik ontology“. An ontology, or several, together with certain instances of classes will make up a knowledge base - however in practice, the line between ontologies and a knowledge base is often blurred [8]. As another example, consider an ontology that would describe furniture; a certain class of this ontology would then be chairs, and instances of this class would be specific types of chairs.

2.1.6 Skill

Described in [17], “a reusable robot program is called skill which is the capability of performing an action resulting in a meaningful outcome”. As in [18], a skill is defined according to the product, process, resources triangle (also known as the PPR triangle). To make the concepts concrete to grasp, imagine a robot that would prepare a cup of tea. The product would in this case be the cup of tea, the process would define all relevant information about how to prepare the cup of tea (where the robot should position its limbs when for example pouring the water into the cup), and the resources would be the material requirements for the robot to have to be able to prepare the cup of tea (such as the limbs).

Skills can be compound or primitive (also known as non-divisible). In the tea example, a non-divisible skill could be tilt kettle of water, hold mug, and so on. A simple compound skill could be to locate tea bags and bring one or heat water in a kettle (thus consisting of several non-divisible skills). The final complex compound skill would then be to prepare the tea, which consists of multiple simple skills. The way the skills are stored in the KIF knowledge base, and how to identify them, will be further discussed in section 2.2.2.

2.1.7 Sequential Function Chart (SFC)

Sequential Function Chart, SFC, is a graphical robot programming language, used for programming PLC:s (programmable logic controllers) [17]. In figure 2.3, an example of how an SFC could look is shown.

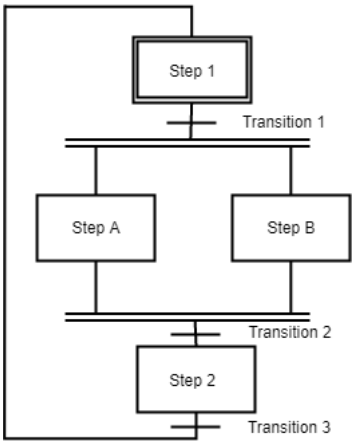

Figure 2.3: Step 1 is the initial step, followed by steps A and B that will be executed parallel, followed by the last step 2

In figure 2.3, each step has an action that will be executed until the following transition condition is fulfilled. The initial step (shown with double lines), will therefore be executed until the first transition is fulfilled. The parallel lines surrounding step A and B means they will be executed at the same time, until the second transition condition is fulfilled, and lastly step 2 will be executed until the third transition is fulfilled [17].

To give the reader a sense of how a real SFC could look, figure 2.4 shows a real SFC.

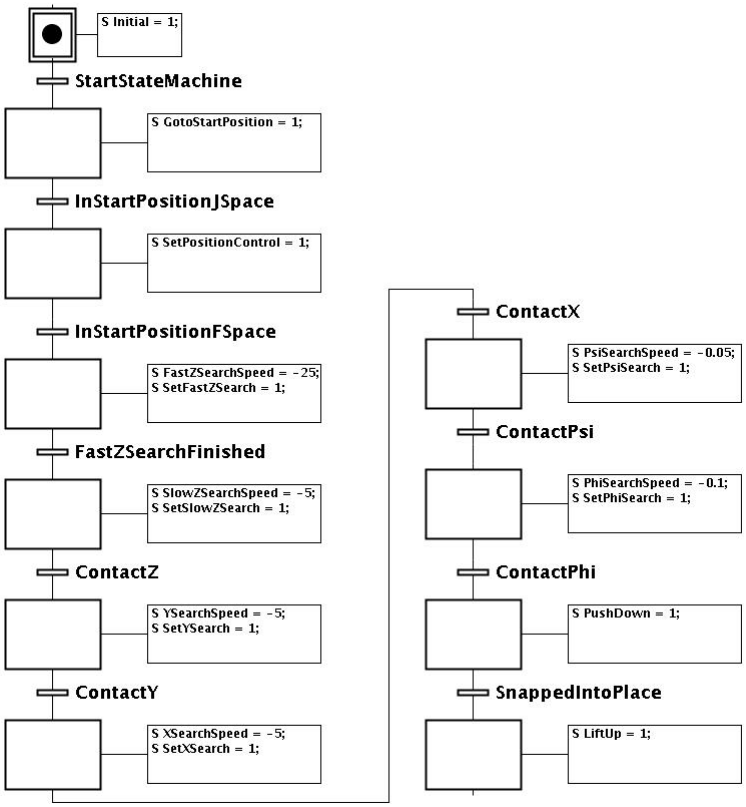

Figure 2.4: Example of a real SFC [7].

The SFC in this figure is the one of a skill called “snapfit”, which consists of snapping a piece of plastic into a small opening - something that sounds trivial but actually contains several steps for the robot to achieve. As in the previous example, the horizontal lines paired with text in the figure above are transition states. The current state of the SFC is the box containing the black circle.

Some of the skills stored in the knowledge base can be described using SFC:s. They then have an SFC stored within the statements of that certain skill. Apart from being essential for programming robot skills, the SFC:s can also be used to show how a skill is executed, serving as a graphical explanation to the human.

2.2 The knowledge base

The knowledge base developed at LTH is called the Knowledge Integration Framework - or simply KIF [1]. The knowledge base is, as the name may give away, the core of knowledge based robotics. It is here that skills, ontologies and other data related to the robots are stored. According to Stenmark [17], the KIF knowledge base mainly consists of robotics ontologies, dynamic data repositories and several services to reach this knowledge and data.

The knowledge base as it is presented when accessing the data directly from the server is organized as shown in figure 2.5 - it consists of repositories, each containing one or more contexts and the contexts containing several RDF statements. These RDF statements convey the knowledge in the knowledge base, and the contexts can explain robotic skills or ontologies. In section 2.2.2, the relation between skill and ontology is explained.

What does not show in figure 2.5 are the relations between different statements of different repositories, or the structure of the ontologies used in the knowledge base. The latter will be explained further in section 2.2.1.

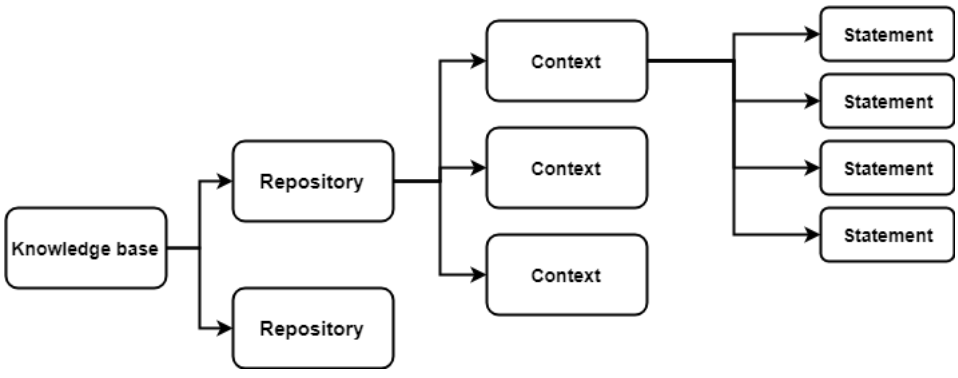

Figure 2.5: A schematic view of how the KIF knowledge base is built.

[1] Not to be confused with the Knowledge Interchange Format described in [2]

2.2.1 Ontology structure

As suggested by [5], the structure of the ontologies in the knowledge base KIF is built according to figure 2.6. Each ontology describes different concepts that the robot needs to execute a skill. The rosetta ontology is a continuously developed project, where the aim is to create a generic ontology for industrial robots [18]. This ontology is accompanied by several others, among these the *configuration.owl* and *coordination.owl* ontologies.

The configuration and coordination ontologies describe skill behaviors and skill parameterizations, where the ontology *coordination.owl* describes the former and *configuration.owl* the latter. Together with the *rosetta.owl* ontology, they complete the information needed for skill execution.

As can be seen in figure 2.6, the concrete robotic skills refer to terms in coordination and configuration ontologies, which in turn use ones from the *rosetta.owl* ontology (in some cases however, skills refer directly to the *rosetta.owl* ontology, which is not shown in the picture).

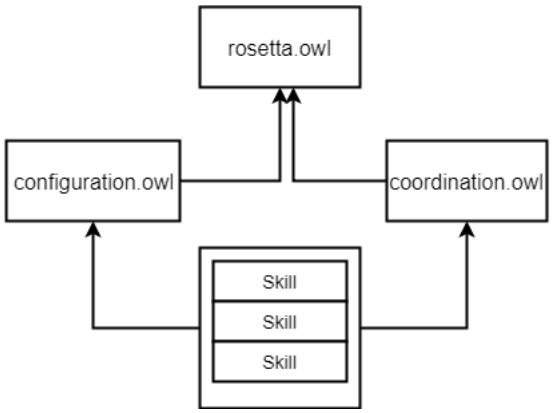

Figure 2.6: The ontology structure of the KIF knowledge base according to [5].

2.2.2 Skill in relation to ontologies

A skill for the purpose of this thesis will be any instance or subclass of the concept "`http://kif.cs.lth.se/ontologies/rosetta.owl#Skill`". Figure 2.7 shows a stripped outline of how a skill in the KIF knowledge base could be structured. This could be thought of as another perspective of figure 2.6, with details such as parameters added.

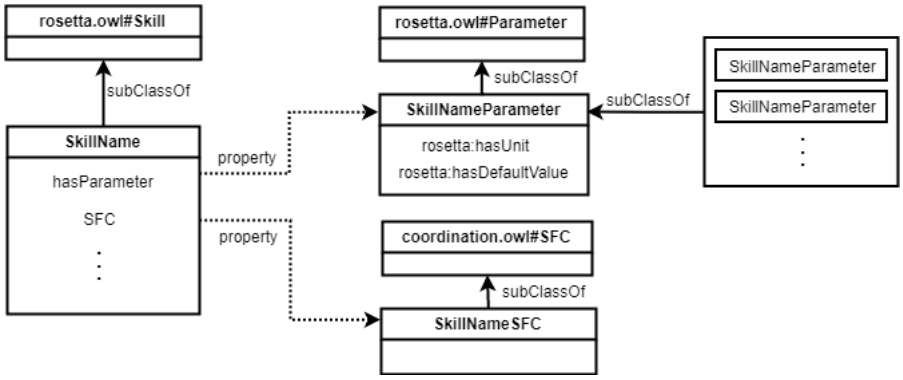

Figure 2.7: A simplified version of a skill outline.

As is obvious from figure 2.7, each skill is a subclass of the *rosetta.owl#Skill* class. To identify which contexts that fulfill this condition, the IRI:s are used. As an example, a skill of, say, 120 statements, will have exactly one statement that says ”<thisSkill> <subclassof> <rosetta.owl#Skill>“.

The information conveyed in the model is low level information on how the robot should behave in order to execute a certain skill. This information is explained using the ontologies, that can be viewed as the models explaining concepts for the robot to be able to perform skills. For example, it could contain information about where the robot should position its limbs and at what time.

2.2.3 SPARQL

To query the information that this type of knowledge base consists of, namely RDF triples, a language called SPARQL (short for “SPARQL Protocol and RDF Query Language”[9]) is used [12]. It enables the programmer to easily specify what type of data that should be retrieved from the knowledge base. As an example the programmer could enter that all statements where the predicate describes “type” and the object describes “human” (i.e. all statements where the subject is of type human) should be retrieved from the knowledge base, or all statements containing the subject “Erik”.

Chapter 3
Program design

This chapter will present the RDF4J (Resource Description Framework for Java) [13], which was used when developing the program, followed by a short outline of the program design. The full implementation can be found at `https://git.cs.lth.se/jacek/new-skills-inventory`.

3.1 The RDF4J framework

The RDF4J framework is a Java framework for processing RDF data, and has been used when developing the program in this thesis. The framework allows the user to query data stored in RDF triples using SPARQL, as well as providing suitable data structures to store the results in. All information about the RDF4J framework can be found in [13]. Because of RDF4J being easily incorporated with the Eclipse Integrated Development Environment, this was used to develop the program.

3.2 Program design outline

The KIF knowledge base server at LTH can be reached through the URL `http://vm25.cs.lth.se/rdf4j-server`. This information - the server name to access the knowledge base - together with what IRI:s the user intends to find and ontologies to be sorted out, need to be entered in a configuration document.

The reason for the ontologies to be sorted out is because they most likely will contain the IRI:s that the user enters, however if the user intends to find for example the robotic skills present in the knowledge base, the ontologies describing the skills are not relevant.

When this information has been entered in the configuration document, the user launches the main program. See figure 3.2 for a schematic view of the program design. In figure

3.1 an excerpt of the configuration file is shown. It is completed with instructions on how the user should enter the information.

The program will start by reading the specifications in the configuration file, followed by querying the server to pick up all statements available. The RDF4J framework will use SPARQL to query the database and fetching the relevant statements. The program first reads all repository names, then establishes a connection with each repository and fetches all contexts and their statements in that repository (as in figure 2.5).

To sort out the contexts containing statements with the specified IRI:s, each statement in the context will be examined to see if its object consists of the IRI. Those contexts that contain statements with the IRI will be sorted out and shown in the end result, which is an HTML table launched in the computer's browser. If the context also contains an SFC, the link to this is added in another column. Each context - each skill - will be clickable, to let the user go directly into the knowledge base to examine it further.

If the context name contains one of the ontology names specified, this context will not be shown in the end result. This was the easiest way to sort out the ontologies.

An excerpt of the end result is shown in figure 3.3. Each skill is presented in a column, and in cases where an SFC is linked to it, that is also shown through a clickable link. This table lets the user see directly what IRI:s are present in which contexts, instead of manually searching through.

```
 /** This is the document in which to specify the details
for visualizing information found in an RDF knowledge base
/** 1.  Server name:
http://vm25.cs.lth.se/rdf4j-server
/** 2.  IRI:s for interesting elements:
http://kif.cs.lth.se/ontologies/rosetta.owl#Skill,
http://kif.cs.lth.se/ontologies/rosetta.owl#Parameter,
http://kif.cs.lth.se/ontologies/sfc.owl#SFC
/** 3.  Ontologies to be sorted out:
rosetta.owl, coordination.owl, configuration.owl, sfc.owl
```

Figure 3.1: An excerpt of the configuration file used in the developed program.

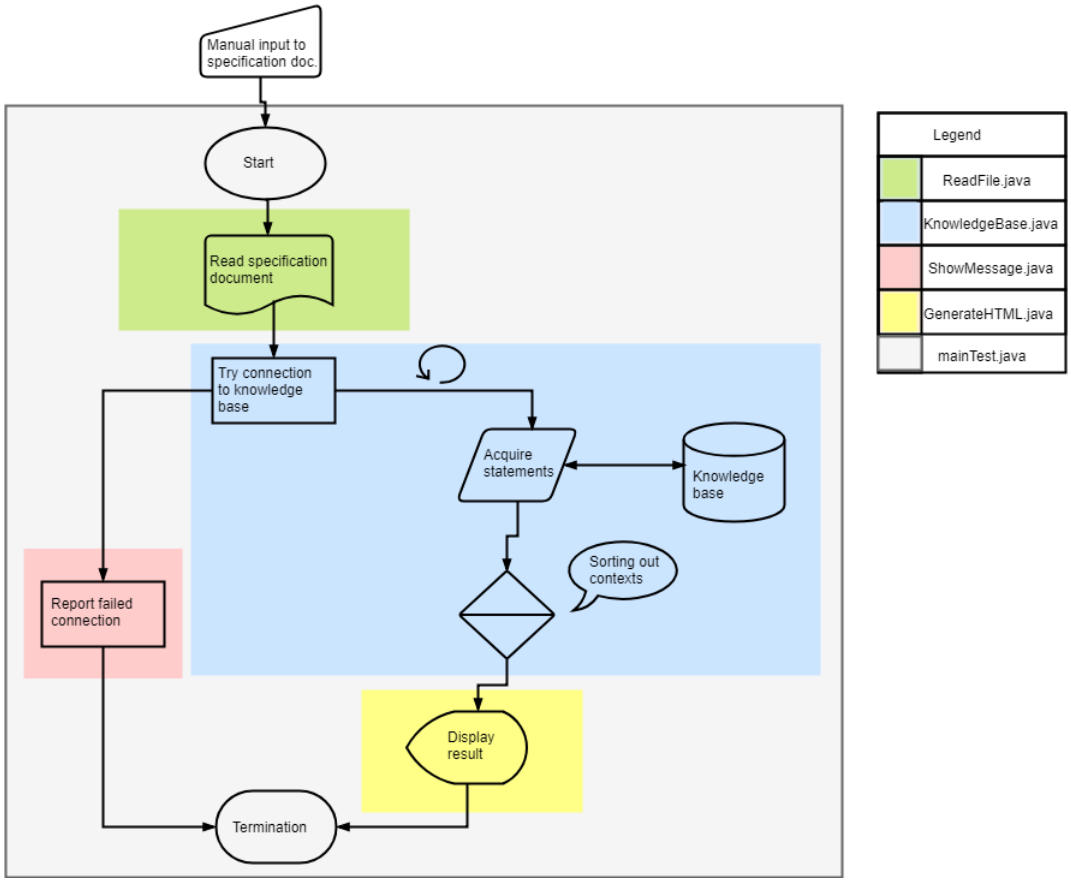

Figure 3.2: The program design outline. Each class is represented by the colorful blocks, and the main program is the beige box surrounding most of the classes.

Server: http://vm25.cs.lth.se/rdf4j-server

Repository: majId

Contains URI: http://kif.cs.lth.se/ontologies/sfc.owl#SFC

SKILL	CONTEXT	SFC
file://checkPenLength.xml	file://checkPenLength.xml	http://kif.cs.lth.se/ontologies/sfc.owl#PenCalibrationsSFC
file://checkPenLength34.xml	file://checkPenLength34.xml	http://kif.cs.lth.se/ontologies/sfc.owl#PenCalibrationSFC
http://checkPenLength.xml	http://checkPenLength.xml	http://kif.cs.lth.se/ontologies/sfc.owl#snapFitSkillSFC

Repository: SFC_tests_ver4

Contains URI: http://kif.cs.lth.se/ontologies/sfc.owl#SFC

SKILL	CONTEXT	SFC
file://output.xml	file://output.xml	http://kif.cs.lth.se/ontologies/sfc.owl#AlignSFC
file://../JGraphChartFiles/snapfittest2.xml	file://../JGraphChartFiles/snapfittest2.xml	http://kif.cs.lth.se/ontologies/sfc.owl#snapFitSkillSFC
file://../JGraphChartFiles/snapfittest.xml	file://../JGraphChartFiles/snapfittest.xml	http://kif.cs.lth.se/ontologies/sfc.owl#SnapFitSFC

Figure 3.3: The table launched in the computer browser.

Chapter 4
Evaluation and discussion

This chapter will evaluate the methods used when developing the program as well as some details about the implementation, discussing them as they are presented.

The project goal was to simplify human interaction with the knowledge base, by letting the user specify requirements, presenting the result in a straightforward manner while keeping the program design modular and execute within a reasonable amount of time.

To "simplify the human interaction with the KIF knowledge base" is a subjective task, which makes it hard to objectively evaluate the result from this goal - especially since no user evaluation has been carried through. It can be argued that the interaction has been simplified, since the user now could enter information and let the program query the database (thus fulfilling the goal of letting the user specify information), instead of reading and making sense of statements manually. However, since the entire purpose of the project is to simplify the human interaction with the KIF knowledge base, further investigations need to be done concerning the front end. The future user interface would need to be developed after a thorough user study, where potential users of the program would have their say on how the program could be designed for it to be as usable as possible for them.

The solution of letting the user enter information in a text document is not to be thought of as final. This is because it requires the user to already have quite a bit of knowledge about the KIF knowledge base and its content - the user must know what IRI:s is, which IRI:s stands for what as well as which ontologies the KIF knowledge base contains.

The time for the project was enough to make the back-end of the program, however the front-end needs serious improvements, to make the program usable. An idea concerning the future front-end could be making a user interface pop-up that launches when the program starts, where the user would then enter the specifications, instead of in a text document. This information could be entered in a multitude of ways - maybe the interface would contain some IRI:s, that the user could choose to query for just by pressing them. This way the user would not have to enter information explicitly but rather have a multitude of IRI:s to choose from (maybe together with explanations of what each IRI would mean).

The current implementation takes for granted that each context contains one and ex-

actly one skill. This is of course a weakness, which could lead to skills being lost and not presented in the accurate way. The reason for this is because when a skill was sorted out according to the logic explained in section 2.2.2, only one skill (i.e. one statement stating "subClassOf" "*rosetta.owl#Skill*") was found per context in the current knowledge base. However, if a context would contain more than one skill, the implementation is fairly easy to modify, by adding a method to double check this. One of the hardships to overcome in that case would be to realize which SFC belongs to which skill. To be able to do this, one would have to (again) look at the statements in the context to be able to distinguish this - it is therefore doable, however not trivial.

When evaluating the end result, the table shown in figure 3.3, where the goal was to present the information in a straightforward manner, some things are realized. The the end result is very primitive, however it does specify the end result information. Before further development of the front end of the end result, a user study should be of interest, realizing in what way the end users would like the information to be presented. A idea for the future is that full SFC:s should be attainable directly from the end result table, instead of having a link referring back into the knowledge base. This should be complemented with skill parameters also directly attainable.

The program is built modular, making further development simplified. For example, as can be seen in figure 3.2, to change the way the program reads the specifications, the class ReadFile could be replaced.

The main concern when evaluating the goals is the running time. For a computer with excellent performance, connected to internet through a cable, the time taken for the program to compile is roughly five seconds. However, depending on the computer and the connection, the compiling could take as long as 60 to 80 seconds. This is due to the multiple connections to the server, as it makes a new connection to each repository, retrieves the statements, closes the connection and moves on to the next repository. The more stable the internet connection, the faster the program execution. The method of multiple connections was necessary to be able to present the repository names in the HTML table, as it otherwise would have been complicated to reach which context was found in which repository. However, if another method would be chosen, it is not certain that the running time would improve greatly, which is mainly due to the large amount of statements in the knowledge base.

Chapter 5
Conclusion

The project goal was to develop a program that would sort out relevant information and present it in a simple way, which has been fulfilled. The main draw-back of the developed program is the running time, which may or may not be able to be improved. The program could definitely improve by adding a user interface, with focus on the end user. An idea of an interface could be one that presents some choices to the user - such as often searched for IRI:s, having a default server name (that could be changed, should need arise), and a clickable list of the ontologies present in the knowledge base, where marked ontologies will not be shown. This could be presented together with a button to launch the program, instead of explicitly having to enter Eclipse and launch it.

As for the sequential function charts, in the third column of the end result, the goal would be to make the links show charts directly. The clickable skill links would, instead of bringing the user into the knowledge base, show skill parameters. The overall impression of the presented table could be enhanced with not too much work.

In conclusion, this has become the base of a program with a lot of more potential and development ideas. Hopefully it will someday be a useful tool when examining the knowledge base and its content.

Bibliography

[1] Crim-Tumelson, R., *Eye on Design: Lost in Space: Clutter Control*, [PDF], LOEX Quarterly, vol. 34, issue 4, 2008.

[2] Genesereth, M., Fikes, R., *Knowledge Interchange Format Version 3.0 Reference Manual*, Stanford University, 1992.

[3] GraphDB, *Introduction to the Semantic Web*, [website], `http://graphdb.ontotext.com/documentation/standard/introduction-to-semantic-web.html?highlight=basic%20rdf%20components#resource-description-framework-rdf`, (accessed May 19, 2018).

[4] Gruber, T., *Ontology*,[website], 2007, `http://tomgruber.org/writing/ontology-definition-2007.htm` (accessed May 19, 2018).

[5] Jacobsson, L., *A Module-Based Skill Ontology for Industrial Robots*, Master Thesis, Lund University 2015.

[6] Merriam-Webster, [website] `https://www.merriam-webster.com/dictionary/semantics` (accessed on June 13 2018)

[7] Malec, J., Nilsson, K., and Bruyninckx, H., *Describing assembly tasks in declarative way*, IEEE ICRA 2013 Workshop on Semantics, Identification and Control of Robot-Human-Environment Interaction

[8] Noy, N., McGuiness, D. *Ontology Development 101: A Guide to Creating Your First Ontology*,[PDF], `https://protege.stanford.edu/publications/ontology_development/ontology101.pdf` (accessed May 21, 2018).

[9] Ontotext, *What is SPARQL?*, [website], https://ontotext.com/knowledgehub/fundamentals/what-is-sparql/ (accessed June 13, 2018).

[10] Poole, D., and Mackworth, A., *Artificial Intelligence Foundations of Computational Agents*, Cambridge University Press, 2010.

[11] Protégé, [website], https://protege.stanford.edu/, (accessed on June 15, 2018)

[12] Prud'hommeaux, et al. *The World Wide Web Consortium*, [website], 2013, https://www.w3.org/TR/sparql11-overview/, (accessed May 30, 2018).

[13] RDF4J, *RDF4J*, [website], http://rdf4j.org/ (accessed March-June 2018)

[14] Riazuelo, L. et al. *RoboEarth Semantic Mapping: A Cloud Enabled Knowledge-Based Approach*, IEEE Transactions on Automation Science and Engineering, vol. 12, no. 2, 2015

[15] Shadbolt, N., Hall, W., Berners-Lee, T., *The Semantic Web Revisited*, https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1637364 [PDF], IEEE Intelligent Systems, vol. 21, issue 3, 2006.

[16] Schreiber, G. and Raimond,Y. *The World Wide Web Consortium*, [website], 2014, http://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624, (accessed May 19, 2018).

[17] Stenmark, M., *Intuitive Instruction of Industrial Robots: A Knowledge-Based Approach*, PhD Thesis, Lund University 2017.

[18] Stenmark, M. et al., *Supporting Semantic Capture During Kinesthetic Teaching of Collaborative Industrial Robots*, International Journal of Semantic Computing, vol. 12, no. 1, 2018, pp. 167-186.

[19] Tenorth, M., Beetz, M., *KNOWROB — Knowledge Processing for Autonomous Personal Robots*, 2009, [PDF], https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5354602, (accessed on June 15, 2018)

[20] Thomason, R., *What is semantics?* [website], 2012, https://web.eecs.umich.edu/~rthomaso/documents/general/what-is-semantics.html (accessed on June 13 2018).

[21] W3C, *An Introduction to Multilingual Web Addresses*, [website], https://www.w3.org/International/articles/idn-and-iri/ (accessed on June 16 2018).

[22] Weibel, et al. (2011) *RoboEarth, a World Wide Web for Robots* [PDF] https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5876227&tag=1 (accessed on June 15 2018).

EXAMENSARBETE Inventory of Robotic Skills in theKnowledge Integration FrameworkKnowledge Base
STUDENT Isabella Gagner
HANDLEDARE Jacek Malec (LTH)
EXAMINATOR Elin Anna Topp (LTH)

Framtidens industrirobotar och deras kunskaper

POPULÄRVETENSKAPLIG SAMMANFATTNING **Isabella Gagner**

De senaste åren har robotar som byggs med hjälp av artificiell intelligens varit ett hett ämne, kanske främst för att många ser en postapokalyptisk framtid framför sig där robotar har tagit över världen. YuMi-roboten [1](se bild), en så kallad smart, kollaborativ robot, är ett mindre skräckinjagande exempel på en robot som bygger på en typ av artificiell intelligens. Denna typ av artificiella intelligens kallas för kunskapsbaserad robotik.

Kunskapsbaserad robotik innebär kort sagt att roboten har en kunskapsbas, där färdigheter ligger som informationspaket, redo att bli nedladdade av roboten. Tänk dig till exempel en färdighet som är "förbered en kopp te" - det paketet skulle innehålla bland annat information om hur roboten ska bete sig för att koka vatten eller måtta teet.

[1]https://new.abb.com/products/robotics/sv/industrirobotar/yumi hämtad 2018-05-29

Denna kunskapsbas innehåller, förutom de informationspaket som är direkt knutna till en färdighet, även ytterligare information som robotarna kan ha nytta av. För robotarna är kunskapsbasen lätt att navigera, men på grund av att många olika forskningsprojekt har bidragit till samma bas är strukturen mindre lätt att navigera som människa. Den är dessutom inte speciellt lättläst även då en har lärt sig strukturen. Detta bidrar till svårigheter då det trots allt är människan som designar färdigheterna för robotarna - en kunskapsbas som är svår att få en överblick av bidrar till att onödig tid går till att söka igenom kunskapsbasen manuellt efter specifik information. Det är detta problem som detta arbete ämnat lösa.

Genom att bygga ett program som låter människan specificera information som hen vill ha direkt tillgång till, som sedan söker igenom kunskapsbasen och presenterar resultatet på ett enkelt sätt, kan människans interaktion med kunskapsbasen förenklas avsevärt och mer tid kan läggas på att faktiskt utveckla nya robotfärdigheter. Resultatet blev ett program som gjorde just detta - lät användaren ange vilka informationspaket hen ville se och presenterade dessa i en lista med klickbara länkar. Länkarna leder användaren direkt in i rätt ställe i kunskapsbasen.

Förhoppningen med programmet är att det ska kunna vidareutvecklas och bli ett essentiellt verktyg för alla som kommer att arbeta med denna kunskapsbas i framtiden!