(Meta)Knowledge modeling for inventive design

Wei Yan

UNIVERSITÉ DE STRASBOURG

INSA STRASBOURG

N° d'ordre:

École Doctorale Mathématiques, Sciences de l'Information et de l'Ingénieur

UDS - INSA - ENGEES

THÈSE

présentée pour obtenir le grade de

Docteur de l'Université de Strasbourg

Discipline: INFORMATIQUE

Spécialité : Informatique

par

Wei YAN

(Meta)Knowledge Modeling for Inventive Design

(Modélisation des (Méta)Connaissances pour la Conception Inventive)

soutenue publiquement le 07 février 2014 devant le jury:

Directeur de thèse:	DENIS CAVALLUCCI	Professeur, INSA de Strasbourg
Co-Directeur de thèse:	PIERRE COLLET	Professeur, Université de Strasbourg
Rapporteur externe:	ROSSI SETCHI	Professeur, Cardiff University
Rapporteur externe:	JOOST DUFLOU	Professeur, Katholieke Universiteit Leuven
Examinateur:	CAROLE BOUCHARD	Professeur, Arts et Métiers ParisTech
Examinateur:	CECILIA ZANNI-MERK	Maître de conférences, INSA de Strasbourg

Acknowledgements

This dissertation was made possible by the help of many individuals during the last three years and a half in France. Foremost, I would like to thank China Scholarship Council (CSC) for the financial support during my Ph.D. study and research in the Design Engineering Laboratory (LGeCO) at the National Institute of Applied Science (INSA) of Strasbourg, and the Engineering Science, Computer Science and Imaging Laboratory (ICube) in the University of Strasbourg.

I would like to express my sincere gratitude to my supervisor, Prof. Denis Cavallucci, for his continuous support of my Ph.D. study and research, and for his motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. My sincere thanks also goes to my co-supervisor, Prof. Pierre Collet, for his open-minded mindset as a leader and his trust in my choices throughout this research. I would like to thank my assistant supervisor, Prof. Cecilia Zanni-Merk, for her continuous engagement, invaluable insights and careful inspection for every detail during my Ph.D. study. I have greatly benefited from her supervision style that provides intense training on how to think carefully and how to present logically. I would also like to thank Prof. François Rousselot for his fruitful discussions and previous suggestions for my work. All of them provide a creative work atmosphere in the research group and always encourage one to strive beyond one's limits. Thank you for all your discussions and guidance!

I would like to express my appreciation to the wonderful faculty members and stuff of the laboratory LGeCO and the laboratory ICube, for their support, understanding, and kindness throughout the process of my doctoral study.

I would especially like to thank Dr. HuiChao Sun, Dr. Qiang Zhang, Dr. Huan He, Dr. Kai Li, Dr. Sarra Mamoghli and Dr. Alban Meffre for their

commitment to our common research interest of inventive design and computer science.

Last but not the least, I would like to thank my parents and brother for their unconditional support throughout my life. Especially I would like to thank my husband, Dr. Liang Zhang, for his patience, understanding and encouragement that made me complete the Ph.D. research.

Abstract

An increasing number of industries feel the need to formalize their innovation processes. In this context, quality domain tools show their limits as well as the creativity assistance approaches derived from brainstorming. TRIZ (Theory of Inventive Problem Solving) appears to be a pertinent answer to these needs. Developed in the middle of the 20th century by G. S. Althshuller, this methodology's goal was initially to improve and facilitate the resolution of technological problems.

According to TRIZ, the resolution of inventive problems consists of the construction of models and the use of the corresponding knowledge sources. Different models and knowledge sources were established in order to solve different types of inventive problems, such as the forty inventive principles for eliminating the technical contradictions. These knowledge sources with different levels of abstraction are all built independent of the specific application field, and require extensive knowledge about different engineering domains.

In order to facilitate the inventive problem solving process, the development of an "intelligent knowledge manager" is explored in this thesis. On the one hand, according to the TRIZ knowledge sources ontologies, the manager offers to the users the relevant knowledge sources associated to the model they are building. On the other hand, the manager has the ability to fill "automatically" the models of the other knowledge sources.

These research works aim at facilitating and automating the process of solving inventive problems based on semantic similarity and ontology techniques. At first, the TRIZ knowledge sources are formalized based on ontologies, such that heuristic inference can be executed to search for specific solutions. Then, methods for calculating semantic similarity are explored to search and define the missing links among the TRIZ knowledge sources.

In order to solve inventive problems, the TRIZ user firstly chooses a TRIZ knowledge

source to work for an abstract solution. Then, the items of other knowledge sources, which are similar with the selected items of the first knowledge source, are obtained based on semantic similarity calculated in advance. With the help of these similar items and the heuristic physical effects, other specific solutions are returned through ontology inference.

Finally, a software prototype is developed based on semantic similarity and ontology inference to support this automatic process of solving inventive problems.

Keywords: Innovation; TRIZ (Theory of Inventive Problem Solving); Knowledge source; Semantic similarity; Case-based reasoning; Ontology; Ontology inference

Résumé

Un nombre croissant d'industries ressentent le besoin de formaliser leurs processus d'innovation. Dans ce contexte, les outils du domaine de la qualité et les approches d'aide à la créativité provenant du "brain storming" ont déjà montré leurs limites. Afin de répondre à ces besoins, la TRIZ (Acronyme russe pour Théorie de Résolution des Problèmes Inventifs), développée par l'ingénieur russe G. S. Altshuller au milieu du 20ème siècle, propose une méthode systématique de résolution de problèmes inventifs multi-domaines.

Selon TRIZ, la résolution de problèmes inventifs consiste en la construction du modèle et l'utilisation des sources de connaissance de la TRIZ. Plusieurs modèles et sources de connaissances permettent la résolution de problèmes inventifs de types différents, comme les quarante Principes Inventifs pour l' élimination des contradictions techniques. Toutes ces sources se situent à des niveaux d'abstractions relativement élevés et sont, donc, indépendantes d'un domaine particulier, qui nécessitent des connaissances approfondies des domaines d'ingénierie différents.

Afin de faciliter le processus de résolution de problèmes inventifs, un "Système Intelligent de Gestion de Connaissances" est développé dans cette thèse. D'une part, en intégrant les ontologies des bases de connaissance de la TRIZ, le gestionnaire propose aux utilisateurs de sources de connaissance pertinentes pour le modèle qu'ils construisent, et d'autre part, le gestionnaire a la capacité de remplir "automatiquement" les modèles associés aux autres bases de connaissance.

Ces travaux de recherche visent à faciliter et automatiser le processus de résolution de problèmes inventifs. Ils sont basés sur le calcul de similarité sémantique et font usage de différentes technologies provenantes de domaine de l'Ingénierie de Connaissances (modélisation et raisonnement basés sur les ontologies, notamment). Tout d'abord, des méthodes de calcul de similarité sémantique sont proposées pour rechercher et définir

les liens manquants entre les bases de connaissance de la TRIZ. Ensuite, les sources de connaissance de la TRIZ sont formalisées comme des ontologies afin de pouvoir utiliser des mécanismes d'inférence heuristique pour la recherche de solutions spécifiques.

Pour résoudre des problèmes inventifs, les utilisateurs de la TRIZ choisissent dans un premier temps une base de connaissance et obtiennent une solution abstraite. Ensuite, les éléments des autres bases de connaissance similaires aux éléments sélectionnés dans la première base sont proposés sur la base de la similarité sémantique préalablement calculée. A l'aide de ces éléments et des effets physiques heuristiques, d'autres solutions conceptuelles sont obtenues par inférence sur les ontologies.

Enfin, un prototype logiciel est développé. Il est basé sur cette similarité sémantique et les ontologies interviennent en support du processus de génération automatique de solutions conceptuelles.

Mots-clés: Innovation; TRIZ (Théorie de Résolution des Problèmes Inventifs); Source de connaissance; Similarité sémantique; Case-based reasoning; Ontologie; Inférence de l'ontologie

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter gives a general introduction of this thesis that is realized at the Laboratoire du Génie de la Conception (LGECO) and the Laboratoire des Sciences de l'Ingénieur, de l'Informatique et de l'Imagerie (ICUBE). Firstly, the general background of this thesis is given. Then, the overall research problem, the motivation and research scope of this thesis are discussed. Finally, the structure of this thesis is presented.

1.1 General Background

Confronting the trend of globalization of economic activities, companies operate today, more than ever before, in a very competing and complex environment with rapidly changing market conditions. Thus, to play an important role in the global market, it is necessary to increase the capability to innovate and combine customer needs, productivity and competitiveness in the development of new products, services or business models. It is reported that successful companies produce seventy-five percent of their revenues from new products or services that did not exist five years ago [1]. This especially applies to companies that obtain their competitive advantages by technological lead.

Innovation is defined as the process of making changes, large and small, radical and incremental, to products, processes, and services that results in the introduction of something new for the organization that adds value to customers and contributes to the knowledge store of the organization [2]. Applying innovation is to use practical tools and techniques that can make these changes to products, processes, and services.

Innovation consists of the generation of a new idea and its implementation into a new

product, process or service, leading to the dynamic growth of the national economy and the increase of employment as well as to a creation of pure profit for the innovative business enterprise. Innovation is never a one-time phenomenon, but a long and cumulative process of a great number of organizational decision-making process, ranging from the phase of generation of a new idea to its implementation phase. New idea refers to the perception of a new customer need or a new way to produce. It is generated in the cumulative process of information-gathering, coupled with an ever-challenging entrepreneurial vision. Through the implementation process the new idea is developed and commercialized into a new marketable product or a new process with attendant cost reduction and increased productivity [3].

The innovation in industries can also be considered as a complex process where creative ideas have been turned into new products and services in the market [4]. The innovation process consists of seven steps [5]:

- *Step 1 Strategic Thinking* The innovation process begins with the goal to create strategic advantage in the market, so in this stage, the way innovation adds value to the strategic intents and the areas innovation has the greatest potential to provide strategic advantage are considered.

- *Step 2 Portfolio Management & Metrics* In this stage, the innovation portfolios are managed aggressively to balance the inherent risks of the unknown with the targeted rewards of success and the pursuit of the ideal with the realities of learning, risking, failing in order to ultimately succeed.

- *Step 3 Research* Through research, new significant opportunities for innovation are proposed in this stage.

- *Step 4 Insight* The innovation and the target are mutually clarified to confirm the right value proposition for the right customer.

- *Step 5 Innovation Development* The process of design, engineering, prototyping, and testing that results in final product, service, and business is designed. Furthermore, manufacturing, distribution, branding, marketing, and sales are also integrated to this multi-disciplinary process.

- *Step 6 Market Development* This universal business planning process with brand identification and development is implemented to help customers to understand and choose the way to innovate and to rapid sales growth.

- *Step 7 Selling* In this stage, the customers benefit from successfully selling the new innovative products and services.

The scope of innovation exists primarily within the realm of the individual and the collective knowledge. This has become increasingly evident as the complexity of technology and markets has increased. The trends of more technology-intensive products or services intensify the necessity of innovation, and long term profitable growth of companies also requires a continuous flow of innovative products and processes. For the most of successful companies, innovation has become an indispensable factor, and therefore, in order to be more competitive in the market, the companies need to create an environment of continuous innovation.

However, innovation is more than an idea or an invention but a complex and multi-dimensional process that should be investigated from different views [6]. During the last decades, innovation has undergone an abrupt change in the following three aspects [7]:

- Innovation has become a condition of survival.

- A continual flow of innovation is a frequent phenomenon.

- Innovation is now spreading throughout the industry.

At the same time, there is a rapid increase of complexity and uncertainty in the process of innovation because:

- There are much more diverse needs from customers and more competitions from the market than before.

- The life cycle is much shorter and the complexity of products is higher than before.

- There are increasing limited resources and strict environmental constraints.

In order to create innovative products or services in a continuous way, an increasing number of industries feel the need to formalize their innovation processes [8]. In this

context, quality domain tools show their limits as well as the creativity assistance approaches derived from brainstorming[9].

TRIZ, the Theory of Inventive Problem Solving (Section 2.1), appears to be a pertinent answer to these needs. Developed in the middle of the 20th century by G. S. Altshuller, this methodology's goal was initially to improve and facilitate the resolution of technological problems [10, 11].

TRIZ relies on a few main ideas:

- Technical artefacts evolve according to certain evolution laws.
- Inventions or patents from different domains often solve the same abstract problem.
- Finding an inventive solution to a problem is not easy, because we are not able to search for the solution beyond the fields we master. It is, therefore, necessary to break the psychological inertia.
- If design is considered as a problem, TRIZ is a method for solving this specific kind of problem.
- The whole makes up a self-designated "theory of inventive design".

The fact that technical artefacts evolve according to certain evolution laws represents an application of dialectics to technical systems that is also applied to organizations. For example, an artefact evolves by optimizing some of its parameters with the goal of getting a maximum of desired values, which is a compromise solution. When there is no solution of this kind, we are in presence of a "contradiction" (Section 2.1.3) in the dialectical sense of the term. It is not about a logical impossibility. The contradiction takes its place in a dynamic framework where the change in the value of a parameter has an influence on the change of another parameter.

TRIZ differs from other techniques by two of its three fundamental principles: problems appear because of and are solved taking into account certain evolution laws that are independent of the designer, and the description of the problem in terms of contradictions eases its resolution. According to the third principle, a problem has to be solved by the maximum use of existing resources, but it is common to the notion of value.

As shown in Figure 1.1, the process of using classic TRIZ can be identified as three different phases:

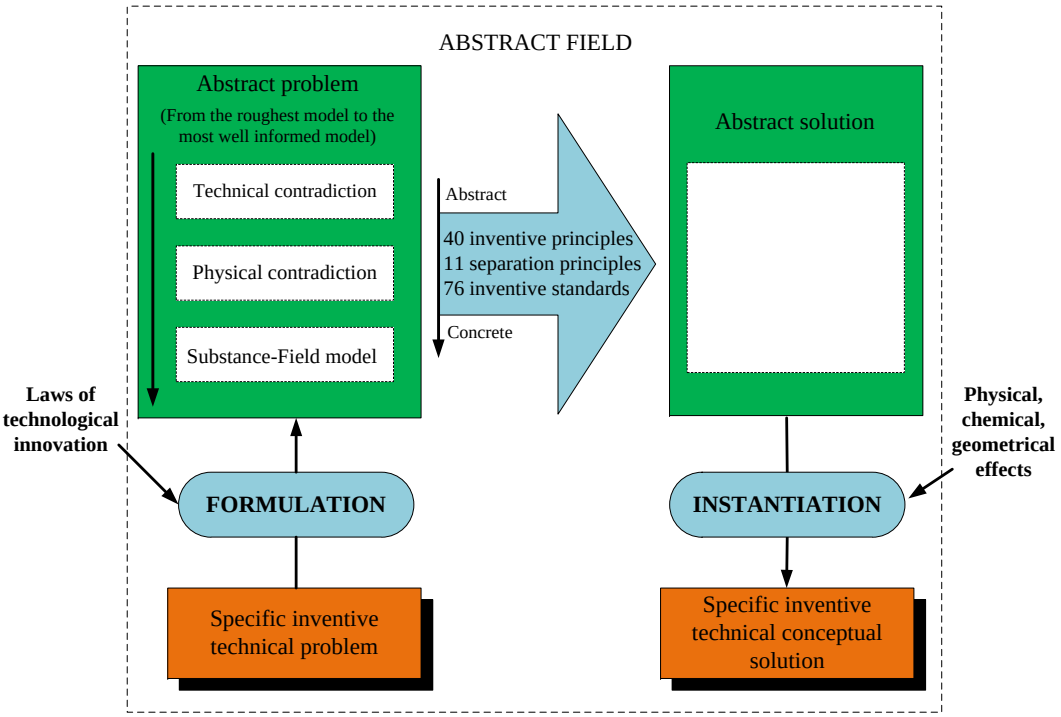


Figure 1.1: The process of using classic TRIZ to solve inventive problems.

- The "formulation" phase, where the expert uses different tools to express the problem in the form of a contradiction network or another model.
- The "abstract solution finding" phase, where access to different knowledge bases is made to get one or more abstract solutions.
- The "interpretation" phase, where these abstract solutions are instantiated with the help of the physical-chemical-geometrical effects knowledge base, to get one or more concept solutions to be implemented in the real world.

1.2 Research Problems

According to TRIZ, the resolution of inventive problems consists in the construction of models and the use of the corresponding knowledge sources (Section 2.1.5). Different models and knowledge sources are built to facilitate the use of TRIZ in different domains. However, there exist several problems in the use of these models and knowledge sources:

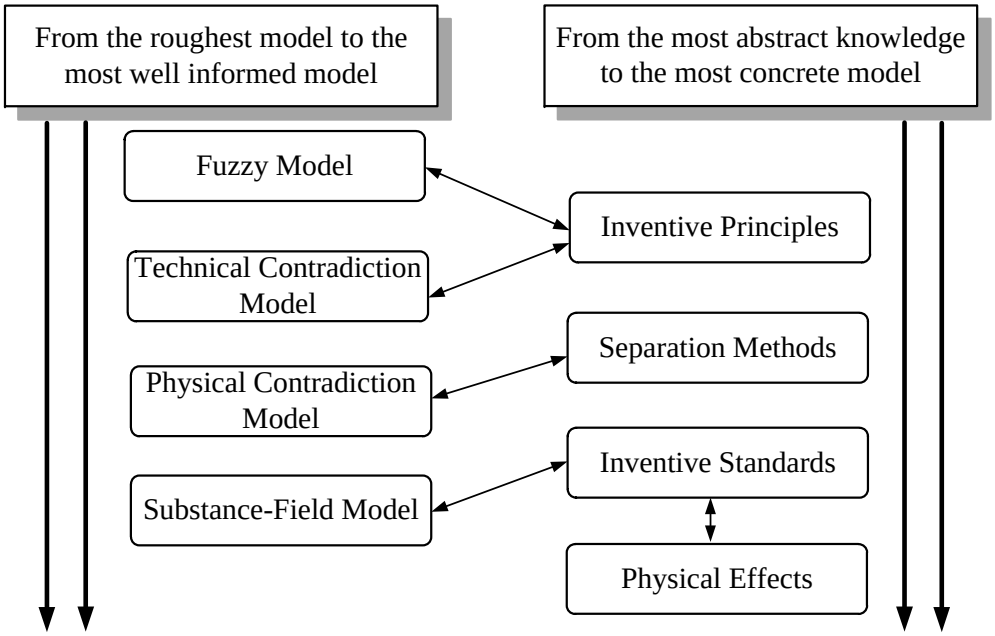


Figure 1.2: The main successive models in TRIZ.

- There are different models at different levels of abstraction. As shown in Figure 1.2, the initial problem being often fuzzy and not clearly expressed, the methodology pushes the designer to the building of a model in terms of a set of predefined TRIZ concepts. This model is called a "systemic model". Then, a second model emphasizes the main contradiction that is at the base of the problem ("technical contradiction" or "physical contradiction").

- Each model is built independently of the specific application field and has its set of associated knowledge sources. The resolution with different models is done starting from the different associated knowledge sources.

- There exists a gap between the high-level abstraction of TRIZ and reality, and TRIZ does not give directives on the way these models have to be used.

As a result, the TRIZ user is brought to work with several models at different levels of abstraction, whose use requires extensive knowledge about different engineering domains. In order to facilitate this innovative process, especially the use of TRIZ knowledge sources, this thesis studies the innovative topic: Automatic Ontology-based Mechanism for Solving

Inventive Problems, which has never been addressed by any other researchers before. In the next section, the motivations for this study will be explained.

1.3 Motivation

The motivation driving these works is to facilitate and automate the process of solving inventive problem based on semantic technologies.

As stated in Section 1.2, different knowledge sources were established in order to solve different types of inventive problems, such as the forty inventive principles (Section 2.1.5.1) for eliminating the technical contradictions (Section 2.1.3.1) and the eleven separation methods (Section 2.1.5.3) for eliminating the physical contradictions (Section 2.1.3.2). These knowledge sources are all built independently of the application field, and their levels of abstraction are very different, making their use quite complicated. The main reasons for this difficulty are:

- The wealth of knowledge sources in TRIZ is available for solving a large variety of inventive problems but access to the needed specific knowledge might be troublesome.
- TRIZ definitions of physical concepts such as substances and fields are ambiguous and cannot be interpreted adequately.
- Using a recommendation proposed by TRIZ for solving a specific problem requires extensive knowledge of different engineering domains and is not currently supported by the methodology. As a consequence, the user is supposed to possess a high degree of expertise in engineering design.

More than that, these knowledge sources for resolutions are situated at different levels of "closeness to reality". In fact, inventive principles, for example, even if they seem to refer to concrete reality (for instance, "inert atmosphere") are conceptually more abstract than inventive standards (Section 2.1.5.2), which refer to concrete substances or fields. The imagination effort to be done by the user to apply an inventive principle is much more important than that the one to be done if he wants to use an inventive standard. Furthermore, even in the same knowledge source the concepts may be different or incompatible, as stated in [12].

Several works have been carried out regarding the formalization of the TRIZ knowledge sources to facilitate its use. For example, Bultey et al. established several ontologies, based on which an ideal TRIZ reasoning environment was proposed to support and simulate the process of Su-Field analysis [13], and Zanni-Merk et al. formalized the Inventive Design Method (IDM) in ontologies, such as the Evolution Hypotheses Ontology and the Problems and Partial Solutions Ontology [14]. However, their research focuses on one or more parts of the TRIZ knowledge souces, not yielding a comprehensive formalization for the TRIZ knowledge sources and their relations.

The main objective of this thesis is the formalization of the TRIZ knowledge sources, to complete the model and make it wholly consistent by the definition of the missing semantic links. This formalization should allow the development of an intelligent management system of these knowledge sources, with the aim of assisting the TRIZ expert during his activity. Indeed, during the processing of a new case, experts are brought to work with various models at different levels of abstraction. The knowledge management system should suggest the experts the use of the relevant knowledge sources, consistent with the level of abstraction of the model they are building. It will also be able to complete "automatically" the rest of the models, by exploiting the semantic links obtained among the different knowledge sources.

As a result, this thesis focuses on the formalization of the TRIZ knowledge sources and the modelling of the inventive problem solving processes. From our point of view, the principal motivation factors can be summarized as follows:

- The TRIZ knowledge sources should be formalized to facilitate their use.
- The missing semantic links among the TRIZ knowledge sources need to be defined to complete the model and make it wholly consistent.
- The development of an intelligent management system of TRIZ knowledge sources facilitates the inventive problem solving processes and provides assistance to the TRIZ expert during his activity.

1.4 Research Scope

The research described in this thesis was conducted at the Laboratoire du Génie de la Conception (LGECO) and Knowledge Engineering Group of Bioinformatique théorique,

Fouille de données et Optimisation stochastique (BFO) team of the Laboratoire des Sciences de l'Ingénieur, de l'Informatique et de l'Imagerie (ICUBE).

The research at LGECO is made in collaboration with industrial companies. It is characterized by its multidisciplinary approach, and the different research topics include diverse fields, such as production systems, mechanical and management sciences. The LGECO is a well known team that has been working for several years now on inventive design aspects. There are three main axes of representation for the research activities at LGECO, that is, the Research-Invention & Development (RID) Process for Early Stage of Innovation, the Human & Management Resources, and the Fields of Application for Inventive Design Process. Several ongoing research works are closely related to ours, such as Souili [15] who uses patent mining techniques to instantiate TRIZ ontologies.

The originality of the works developed within the Knowledge Engineering group, BFO team in ICUBE is founded on the proposal of a layered architecture Knowledge, Rules and Experience to manage the complexity of the development of a Knowledge based System (KBS). The Knowledge layer contains domain ontologies, the Rules layer allows different types of reasoning (monotonic, spatial, temporal, fuzzy, or others) depending on the application; and finally, the Experience layer allows the capitalization and reuse of prior knowledge, either by the use of Case-Based Reasoning (CBR) [16] or Set of Experience Knowledge Structure (SOEKS) and Decisional DNA (DDNA) [17] or by other mechanisms.

Figure 1.3 shows the focus of our research in the laboratories BFO/ICUBE and LGECO. The research in the Knowledge Engineering group of BFO/ICUBE can be divided into four branches according to the different areas of application:

- Semantic Region Labeling for Remote Sensing Image Interpretation.
- Semantic Technologies for Computer Aided Inventive Design.
- Formalization of Theoretical and Experience Knowledge for the Analysis and Diagnosis of SMEs.
- Environment and Sustainable Development.

This thesis focuses on the use of semantic technologies in the area of Computer Aided Inventive Design.

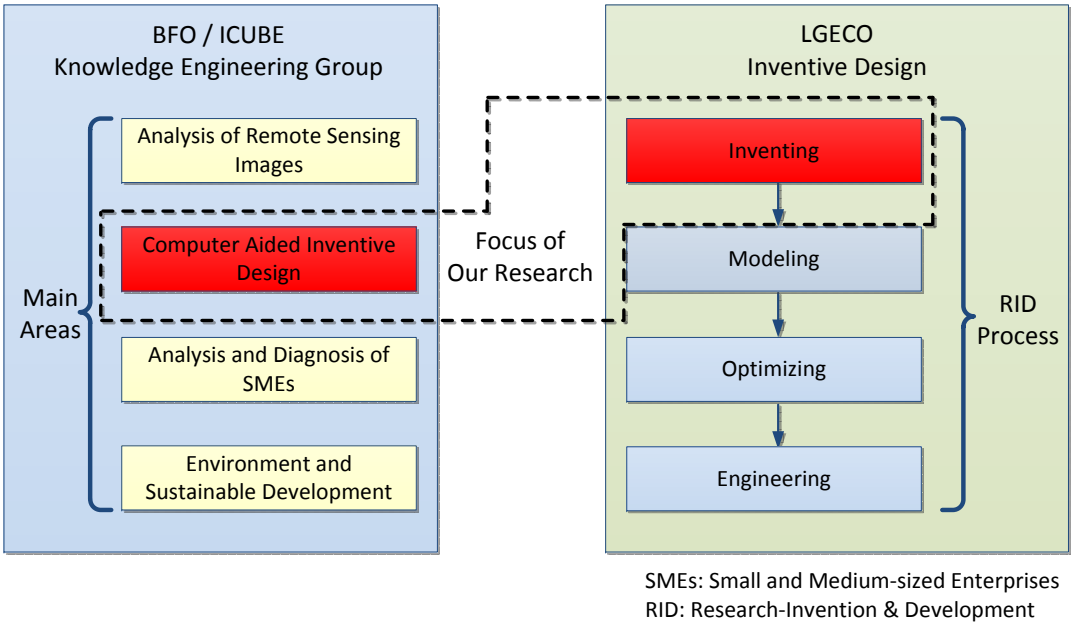


Figure 1.3: The research scope of our study in LGECO and BFO/ICUBE.

At the same time, the research of RID Process in LGECO consists of four phases: Inventing, Modeling, Optimizing and Engineering, in which the user can find opportunities to generate creative ideas, and then transforms these ideas to concepts. However, there are still no standard problem formulation and repeatable procedure of generating ideas in this stage. Thus, this research focuses on Inventing and the transformation from Inventing to Modeling.

1.5 Thesis Structure

Figure 1.4 presents three main technologies used in this thesis, that is, Semantic Similarity, Ontology-based Knowledge Modeling and Ontology Reasoning, and based on these technologies, the automatic ontology-based mechanism for solving inventive problem is made up of three steps:

- Step1-The Matching of Specific Parameters (SP) and Generic Engineering Parameters (GEP) (Section 2.1.5.1): In order to facilitate the use of the contradiction matrix (Appendix B) in the inventive design study, methods of matching SP and

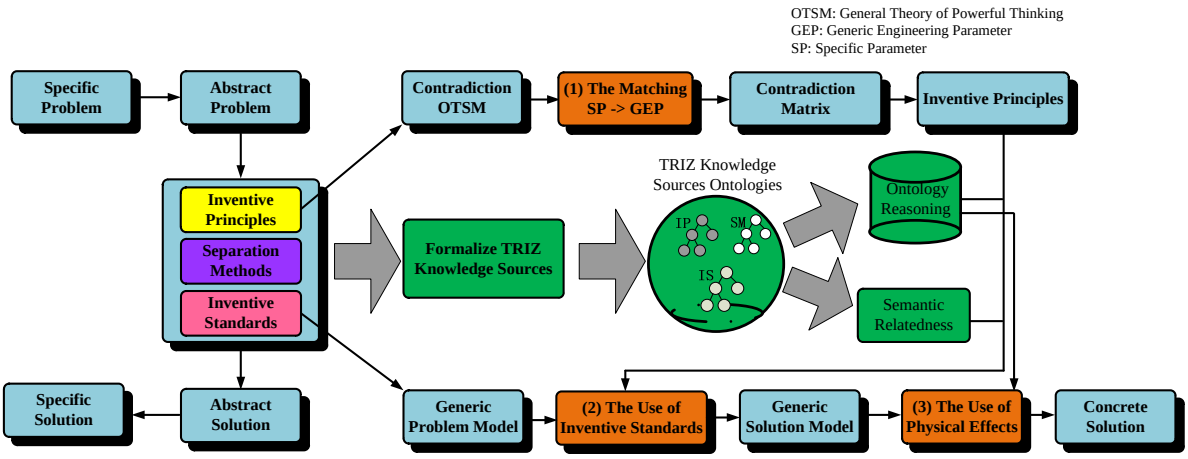


Figure 1.4: The framework of our study.

GEP based on semantic similarity and case-based reasoning are developed.

- Step2-The Use of Inventive Standards: Using the methods of calculating semantic similarity to define the missing links among the TRIZ knowledge sources, formalizing the TRIZ knowledge sources based on ontology and searching for the heuristic abstract solution based on ontology reasoning.
- Step3-The Use of Physical Effects (Section 2.1.6): Formalizing the physical effects based on ontology and Retrieving the heuristic physical effects based on ontology reasoning.

As shown in Figure 1.5, the structure of the whole thesis is:

- **Introduction**
  - Chapter 1 gives a general introduction for the research background, research problem, motivation and research scope.
- **Theoretical Foundations**
  - Chapter 2 presents the theoretical foundations of this thesis, including a detailed literature review on TRIZ, the related knowledge about ontology-based knowledge modeling and ontology reasoning and their application status in the domain of Inventive Design.

- **Contributions**
  - Chapter 3 introduces the proposed methods of calculating semantic similarity in our research, which are used to match SPs and GEPs in the process of using inventive principles and to define the missing links among the TRIZ knowledge sources in the use of inventive standards.
  - Chapter 4 corresponds to the first step of the proposed automatic ontology-based mechanism for solving inventive problems, in which the inventive principles ontology is firstly presented, and then the methods presented in Chapter 3 are used to match SPs and GEPs. In order to improve the performance of matching, the results of previous similar projects are suggested to the users according to a case-based reasoning approach.
  - Chapter 5, corresponding to the second step, gives the detailed process of searching heuristic abstract solutions, including using the methods proposed in Chapter 3 to define the missing links among the TRIZ knowledge sources, the construction of the inventive standards ontology, and the search of heuristic abstract solutions based on ontology reasoning.
  - Chapter 6 discusses the last step, that is, the ontology-based approach of using physical effects, including the re-formalization of physical effects based on ontology and the query of physical effects based on ontology reasoning.
- **Validation**
  - In Chapter 7, a prototype developed based on the research of this thesis is presented, and the resolution of a specific case using this prototype is also given.
- **Conclusions and Perspectives**
  - Chapter 8 concludes our study and recommends several potential directions for future research.

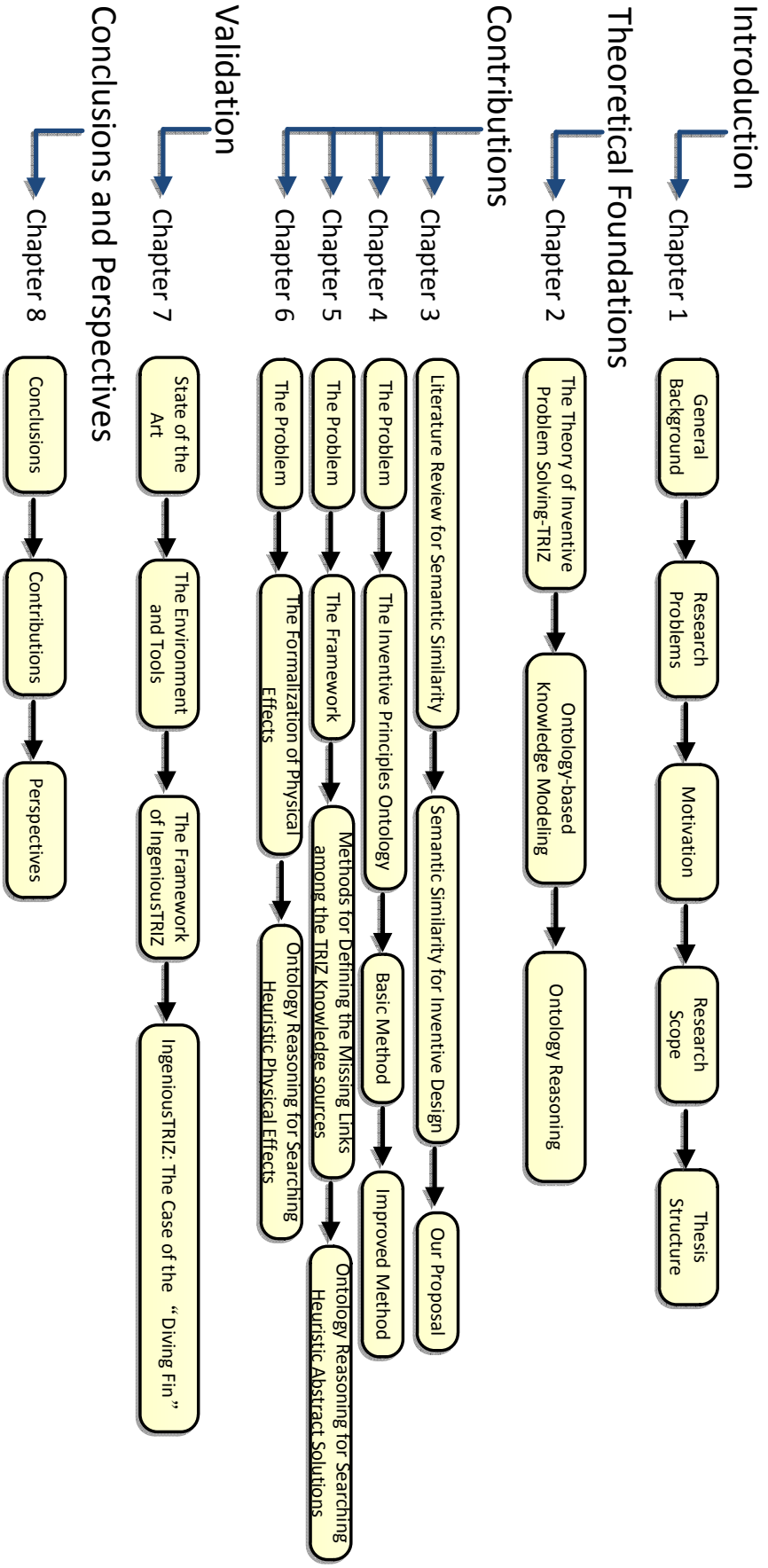


Figure 1.5: The main structure of this thesis.

Bibliography

[1] D. Smith, "Designing an innovative britain," *ESRC The Edge*, vol. 22, 2006.

[2] D. O'Sullivan, *Applying Innovation*. SAGE Publications, Inc, 2009.

[3] K. Urabe, J. Child, and T. Kagono, *Innovation and Management International Comparisons*. Berlin: Walter de Gruyter, 1988.

[4] J. Couger, *Creative Problem Solving and Opportunity Finding*. Danvers, MA, Boyd and Fraser Publishing Co, 1994.

[5] L. Morris, *Serialized Books: The Innovation Master Plan*. [Online]. Available: http://innovationmanagement.se/2013/08/08/how-to-innovate-the-innovation-process/.

[6] R. Garcia and R. Calantone, "A critical look at technological innovation typology and innovativeness terminology: A literature review," *Journal of Product Innovation Management*, vol. 19, 2002.

[7] P. Benghozi, F. Charue-Duboc, and C. Midler, *Innovation Based Competition & Design Systems Dynamics Lessons from French Innovative Firms and Organizational Issues for the Next Decade*. Paris, L'Harmattan: Collection Economiques, 2001.

[8] D. Cavallucci, "A research agenda for computing developments associated with innovation pipelines," *Computers in Industry*, vol. 62.

[9] A.F.Osborn, *Applied Imagination: Principles and Procedures of Creative Problem-Solving*, 1993.

[10] G. Altshuller, *Creativity as An Exact Science*. New York: Gordon and Breach Scientific Publishers, 1984.

[11] ——, *TRIZ: The Innovation Algorithm; Systematic Innovation and Technical Creativity*. Technical Innovation Center Inc., Worcester, MA, 1999.

[12] D. Cavallucci and T. Eltzer, "Parameter network as a means for driving problem solving process," *International Journal of Computer Applications in Technology*, vol. 30, no. 12, pp. 125–136, 2007.

[13] A. Bultey, F. D. Beuvron, and F. Rousselot, "A substance-field ontology to support the triz thinking approach," *International Journal of Computer Applications in Technology*, vol. 30, pp. 113–124, Nov 2007.

[14] C. Zanni-Merk, D. Cavallucci, and F. Rousselot, "An ontological basis for computer aided innovation," *Computers in Industry*, vol. 60, pp. 563–574, Oct 2009.

[15] A. Souili, D. Cavallucci, F. Rousselot, and C. Zanni-Merk, "Starting from patent to find inputs to the problem graph model of idm-triz," in *In: Proceedings of the TRIZ Future Conference 2011*, 2011, pp. 177–189.

[16] A. Aamodt and E. Plaza, "Case-based reasoning : Foundational issues, methodological variations, and system approaches," *AI Communications*, vol. 7, pp. 39–59, 1994.

[17] C. Sanin, L. Mancilla-Amaya, E. Szczerbicki, and P. CayfordHowell, "Application of a multi-domain knowledge structure: The decisional dna. in intelligent systems for knowledge management," *Intelligent Systems for Knowledge Management*, vol. 252, pp. 65–86, 2009.

Chapter 2

Theoretical Foundations

This chapter includes the TRIZ background and the general knowledge about Ontology-based Knowledge Modeling and Ontology Reasoning. At first, the definition of TRIZ, the process of using TRIZ to solve inventive problems and the basic tools of TRIZ, including the contradictions, the patterns of evolution, the TRIZ knowledge sources and the physical-chemical-geometrical effects, will be introduced. Then, in the parts of Ontology Modeling and Reasoning, an overview of their basic methods and concepts will be firstly given, and then the current situation of their applications in the field of inventive design will be illustrated. In the summary, this chapter will be concluded to show the necessity of this thesis.

2.1 The Theory of Inventive Problem Solving - TRIZ

The Theory of Inventive Problem Solving (TRIZ) was developed by G. Altshuller in Russia from 1946 to 1985. After initially reviewing over two hundred thousand patents, Altshuller focused on forty thousand of them as representative of inventive problems, based on which many findings of TRIZ were published. With the increasing exposure in introducing TRIZ, more and more people have been impressed by the power of this systematic innovative approach to creativity.

Now it is known as a powerful methodology for technical problems solving that leads to enhancement of existing technique and strong acceleration of progress [1]. Furthermore, several extensions of TRIZ have been proposed, such as OTSM-TRIZ or IDM (Inventive Design Methodology), which is explored to deal with complex industrial problems and

intended to process problems with hundreds of parameters, contradictions and problems not necessarily linked to inventive product design (for example, in software engineering or in the organization field)[2, 3].

2.1.1 Definition of TRIZ

A definition "TRIZ is a human-oriented knowledge-based systematic methodology of inventive problem solving" is proposed by Savrancky in 2000 [1], and the explanation to this definition is:

- **Human-oriented** It is a human being instead of a machine to orient heuristics since the TRIZ practice depends on the problem itself and socioeconomic circumstances.
- **Knowledge-based** The knowledge about the generic problem solving heuristics is extracted from thousands of patents in different engineering fields. TRIZ uses not only the knowledge in the natural and engineering sciences, but also the knowledge about the domain in which the specific problem occurs.
- **Systematic** TRIZ provides the effective application of the existing resolutions to new problems, and the procedures to creativity are systematically structured.
- **Inventive problem solving** TRIZ aims to solve inventive problems, in which only one main contradiction is to be solved.

2.1.2 The Approach of TRIZ Solving Inventive Problem

During his research, Altshuller proved that a systematic approach to tackle inventive problems following a well-defined process is possible. A major conclusion of Altshuller's studies was that inventions were not the result of unorganized thinking, but instead the products of objective laws and recurrent trends of technology evolution.

Comprehensive studies of patent collections following this discovery resulted in two more findings. Firstly, Altshuller shows that an inventive solution results from the elimination of a contradiction which is caused by attempts to improve preceding products characteristics. Attempts to compromise without eliminating the contradiction do not allow a designer to achieve a degree of improvement qualified as "inventive". The second conclusion is that the majority of the patented inventions comply with a

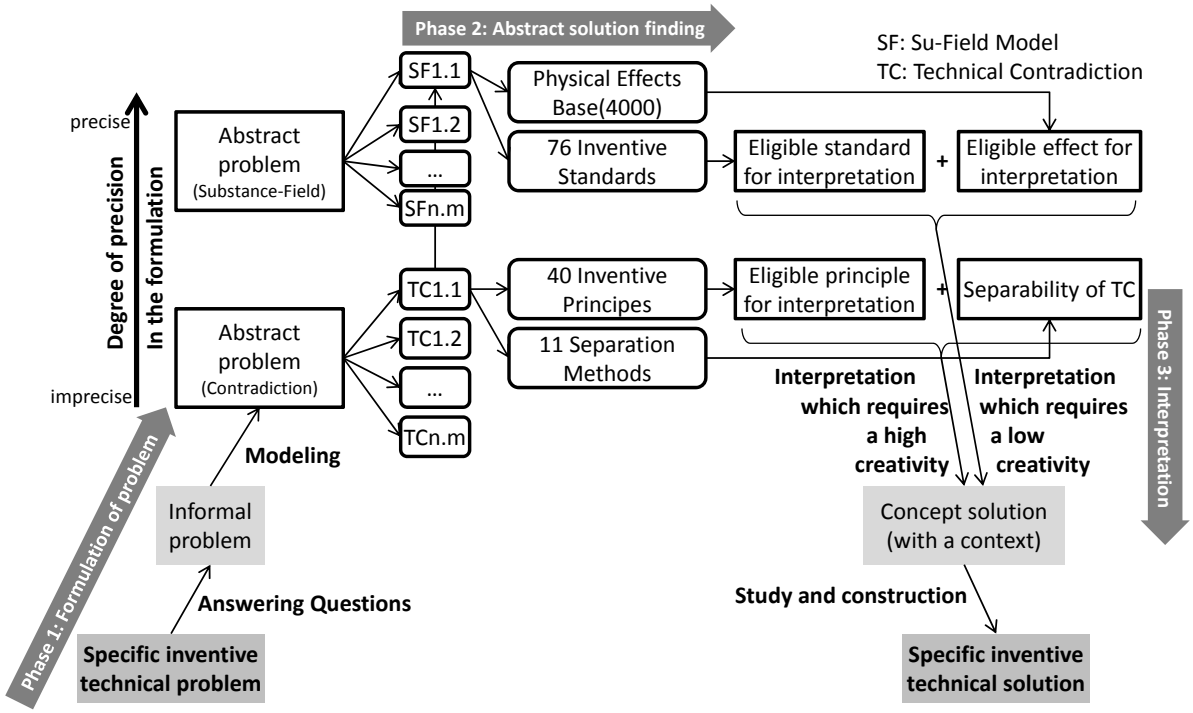


Figure 2.1: The approach of TRIZ solving inventive problem.

relatively small set of basic principles for eliminating the contradictions. Based on these findings, Altshuller has developed a scientifically-based problem solving methodology which codifies numerous inventive principles and, in some techniques, incorporates the laws of engineering system evolution.

Figure.2.1 describes the way classical TRIZ solves problems, and the interrelations among its different parts. Three different phases are clearly identified:

- The "formulation" phase, where the expert uses different tools to express the problem in the form of a contradiction (see later Section 2.1.3) network or another model.
- The "abstract solution finding" phase, where access to different knowledge bases is made to get one or more solution models. Generally, in this step, TRIZ users are required to have wide experience on the TRIZ knowledge sources. They need to be capable of choosing the accurate abstract solution according to the current abstract problem.
- The "interpretation" phase, where these solution models are instantiated with the

help of the physical-chemical-geometrical effects knowledge base, to get one or more solutions to be implemented in the real world.

2.1.3 Contradiction

A goal of the design process is to map a function onto a physical principle that would be capable of performing the function. However, in the situations when it is not available to perform an exact function or none of the previous solutions meet the new specifications, inventive design is difficult to perform due to the uncertainty on how an original problem can be translated into the functional specifications.

From this point of view, the most important TRIZ achievement was that it reveals the common cause of inventive design problems: contradictions. A contradiction arises from mutually exclusive demands that may be placed on the same system where compromising does not produce the required result. Instead of solving inventive problems ad-hoc, TRIZ states that to obtain an inventive solution, a contradiction must be eliminated whereas compromise or optimization methods are not allowed, and it introduces principles for the formulation and elimination of the contradictions in a systematic way.

There are three types of contradictions, that is, administrative contradiction, technical contradiction and physical contradiction.

An administrative contradiction does not reveal any contradictory aspect, and it often describes a desire to improve a characteristic of a system without having an emerging direction of resolution. Its syntax is:

I would like to [*describe the desired characteristic of the studied system*], but I do not know how.

For example, in the case of a "clothespin", an administrative contradiction might be: I would like to [*prevent the clothespin from marking the laundry*] but I do not know how. An analogy can be made at this stage between the part of the sentence regarding willingness to get an improvement and the notion of function as in functional analysis [4].

Taking into account that the administrative contradiction has no heuristic value and cannot indicate a direction to the answer, only technical and physical contradictions are considered because there are more concrete and interconnected.

2.1.3.1 Technical Contradiction

A technical contradiction arises when it is required to improve some feature of the existing prototype but all solutions known within the domain do not produce the required result or their use would cause a negative effect. The impossibility to improve one parameter and to prevent another important parameter from deterioration is the main feature which separates inventive problems from problems that can be solved by a procedure of routine design.

A technical contradiction describes the state of a system where there is an action having a useful effect but causing simultaneously an undesirable effect. In our example of the "clothespin", it could be said that trying not to mark the laundry generates an undesired effect: the laundry will be loosely fixed on the wire.

The resolution of a technical contradiction is mainly performed using reasoning by analogy, facilitated by the use of two TRIZ components: the inventive principles and the contradiction matrix (see later Section 2.1.5).

2.1.3.2 Physical Contradiction

A more precise form of contradictions arising in engineering is a physical contradiction, which indicates that a part of a design prototype should have two mutually exclusive values of the same physical parameter. To eliminate the physical contradiction, it is necessary to change the existing physical principle(s) the prototype is comprised of. Formulation of the physical contradiction makes it possible to identify precisely which part of the prototype is responsible for producing the negative effect. It is important to note that if the given problem may not be solved by using the principles for elimination of technical contradictions, such a situation indicates that a physical contradiction is present in a system.

A technical contradiction requires the definition of several parameters associated to the technical system or of any of its elements. The physical contradiction addresses the part of the technical contradiction centered on that parameter that must have at the same time two opposite values. When acting on a parameter to satisfy a requirement of the problem, it may appear that it must simultaneously have at the same time a value x and $\overline{x}$ (the opposite value of x). In the example of the "clothespin", it can be said that "the stiffness of the spring" must be "hard" to fix the laundry and "soft" not to mark it.

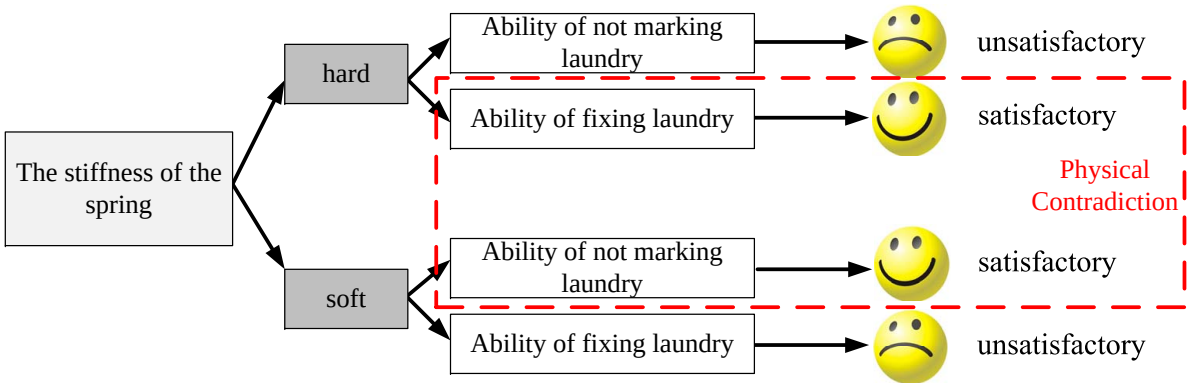


Figure 2.2: The first scheme for the contradictions in the case of a "clothespin".

Contradiction	The stiffness of the spring	
	hard	soft
Ability of fixing laundry		
Ability of not marking laundry		

Figure 2.3: The second scheme for the contradictions in the case of a "clothespin".

There are two main schemes to represent contradictions. As shown in Figure 2.2 and Figure 2.3, the physical contradictions of the case of a "clothespin" are:

- If the stiffness of the spring is hard, the clothespin can fix the laundry better, but it worse prevents marking the laundry.
- If the stiffness of the spring is soft, the clothespin can prevent marking the laundry better, but it worse fixes the laundry.

2.1.4 Patterns of Evolution

Based on a long-term study of patent collections and hundreds of bibliographical sources, G. S. Altshuller discovered that the technical systems do not evolve at random but that

they follow certain patterns, which he called "Laws of Evolution" or "Patterns of Evolution" [5].

These patterns can be revealed from the patent literature and analysis of system development, and purposefully used for systems development without numerous blind trials.

There are eight patterns of evolution, which present a systemic view of the conception of technical products:

- Technology follows a life cycle of birth, growth, maturity, and decline.
- Evolution toward increased ideality.
- Non-uniform development of systems elements.
- Evolution toward increased dynamism and controllability.
- Increased complexity then simplification (reduction) through integration.
- Evolution with matching and mismatching components.
- Evolution toward micro-level and increased use of energy fields to achieve better performance or control.
- Evolution toward decreased human involvement.

These patterns of evolution are potentially very useful because they describe what has repeatedly happened in the past for successful technologies, and consequently indicate what is likely to happen in the future. They can, in other words, be used to predict or at least suggest the next likely and beneficial technical development of a particular industry, for technical systems or their sub-systems [6].

2.1.5 The TRIZ Knowledge Sources

The knowledge sources for solving inventive problems including forty inventive principles, seventy-six inventive standards and eleven separation methods, are used to eliminate technical contradictions, provide common problem-solving methods and eliminate physical contradictions respectively. They are expressed in natural language and, although they

are about related notions, the level of abstraction of the description is very different, making it difficult to understand the links among them[1].

2.1.5.1 Inventive Principles

The first TRIZ problem solving technique was a collection of inventive principles aimed at eliminating technical contradictions [5]. They are heuristic principles based on the accumulated and generalized previous experience of inventors and are available in a form that is independent of any particular engineering domain.

Altshuller formulated thirty-nine generalized engineering parameters (Appendix A), like "the weight of a moving object" or "speed", and built a contradiction matrix (Appendix B), to make the inventive principles (Appendix C) applicable in a systematic way.

A new problem can be solved by the use of a proper inventive principle, after the problem has been formulated as a technical contradiction in terms of predefined generic engineering parameters: "a generic engineering parameter to be improved versus the other generic engineering parameter which deteriorates". Forty inventive principles aimed at resolving contradictions between generic engineering parameters are known, sometimes declined in sub-principles. In this research, these sub-principles are located in order to have a uniform granularity.

The inventive principles can be used in a systematic way by accessing the principles through indices in the contradiction matrix. Along the vertical axis of this matrix the generic engineering parameters which have to be improved are specified while along the horizontal axis the parameters which deteriorate as a result of the improvement are specified. These parameters can be looked up along the vertical and horizontal axes and the matrix suggests up to four principles that can be used to solve the contradiction.

Selected principles are ordered using statistical analysis according to their applicability. The principle chosen most often in the past will most likely solve the next problem and is given first.

Example [2]: Improving the strength of an object (the improvement feature) which consequentially gives rise to an undesirable conflict with the weight of the object (for example, it can be the case of a table). Using the contradiction matrix to identify which

[1] The whole texts of the TRIZ knowledge sources can be found: inventive principles (Appendix C), inventive standards (Appendix D) and separation methods (Appendix E).

[2] Adapted from [6].

inventive principles to use to get strength without increasing weight suggests the following ones:

- IP 40 Composite Materials
- IP 26 Copying
- IP 27 Cheap Short - Living Object
- IP 1 Segmentation

To understand and apply the suggested solutions it is needed to ask which all the desired benefits of the system are and which the priority of those benefits is.

- *InventivePrinciple* 40: Composite Materials suggests using a strong but light material and together with the trigger of IP 1 Segmentation might suggest how to save weight by only putting strength where we need it.
- *InventivePrinciple* 26: Copying suggests that we replace a strong and heavy material with one which is strong but lighter. It might be used a material that looks like wood but is actually a stronger copy material. This principle suggests to copy the heavy material with one which gives the needed strength but has a more appropriate weight. Another example of using this principle copying would be in testing something like a drill which is to be used in remote areas of inaccessible and very hard rock. The tests could be on a copy of rock - artificial rock which simulates natural rock in terms of strength in that it loads the drill with high torque, but could be made of a much lighter composite material which is more easily available than remote rock and easier to handle.
- *InventivePrinciple* 27: Cheap Short - Living Object suggests something like a cardboard box - which assumes that durability is not a requirement. It is needed to understand all the requirements of the system (and the priority of benefits) to judge whether this is an appropriate solution or not.
- *InventivePrinciple* 1: Segmentation represents most of the modern design of tables today: segmented tops for different sizes; segmented table legs for ease of manufacture and assembly; segmented joining of different materials for extra strength.

According to the analysis of principles in [7], the forty inventive principles can be divided into two kinds: thirty-three distinct principles and seven obscure principles. Distinct principles have obvious descriptive information and always share similar or common words with other principles, such as *Inventive Principle 25* "Self-service" with the prefix "self-". Obscure principles share few similar and common words with others and it is difficult to compare with others only by text information. For example, *Inventive Principle 2* "Extraction" describes the extracted objects or systems, without specifying the relationship of extraction by obvious text information. In this research, the distinct principles are directly used while the additional information, such as the similar words, need to be added to the obscure principles to ease their matching.

2.1.5.2 Inventive Standards

They are drawn from the fact that most inventions refer to conceptual modification of physical systems. This means that there should be some common problem solving methods applicable to the whole group of similar inventive problems. If problems from different domains result in identical physical models, this means that the problems are similar. Therefore, they can be solved by applying the same method. Standards are built in the form of recommendations, and generally, formulated as rules like If $< Condition1 >$ and $< Condition2 >$ then $< Recommendation >$. Both conditions permit recognizing the typology of the problem associated to the standard. This way, for a built problem model, there exist a certain number of recommendations allowing the construction of the corresponding solution model. In TRIZ, seventy-six inventive standards (Appendix D) are available. A problem with these inventive standards is that they are formulated abstractly, so their practical use is quite difficult. Some examples include:

- *Inventive Standard 2.4.12*: Applying electrorheologic fluid.
  (a) If a magnetic fluid cannot be used, the electrorheologic fluid may be useful.
- *Inventive Standard 5.3.1*: Changing of phase state.
  (a) Efficiency of the use of substance without introducing other substances is improved by changing its phase.

Although Su-Field analysis provides a simple means to model systems and reveal their problems, more than seventy standard solutions may make users rather confused

and overwhelmed in their search for answers from these many possible solutions. As a result, several classifications of inventive standards were proposed to help the users to find the right solutions in an easy manner.

According to [8], the seventy-six inventive standards are firstly extended to eighty-four standard solutions taking into account that *Inventive Standard 5.1.1* is divided into nine independent standard solutions. Then the eighty-four standard solutions are divided into four classes, that is, A-class: forty inventive standards generalized into seven general solutions, B-class: thirty-one inventive standards identified as some of the existing inventive principles, C-class: six inventive standards considered as new inventive principles, and D-class: seven inventive standards originated from the patterns of evolution (Section 2.1.4).

- *A-class*: Forty inventive standards are condensed and generalized into seven general solutions, which can improve all types of problematic Su-Field models, being incomplete or with harmful, insufficient or excessive impact. These seven generalized standard solutions are discussed as following[1]:
  - General Solution 1: Complete an Incomplete Su-Field Model. Complete a Su-Field model if any of its three components is missing, for example, as shown in Figure 2.4, the missing field F_M needs to be added to connect S_2 to S_1.

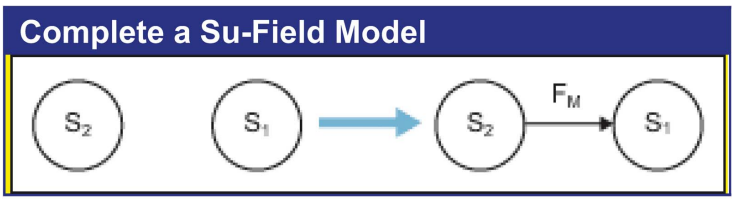


Figure 2.4: Complete a Su-Field model.

  - General Solution 2: Modify Substance S_2 to Eliminate or Reduce Harmful Impact. The physical and/or chemical characteristics of substance S_2 may be changed internally or externally and temporarily or permanently in order to eliminate or reduce harmful impact, as shown in Figure 2.5. Modification can

[1]Please note although these generalized solutions are discussed in the context of a Su-Field model with harmful impact, they are applicable to Su-Field models with other types of problems, for example, excessive and insufficient interactions between substances.

be changing substance S_2 into another form, material or system as long as the Su-Field system with which S_2 is associated carries out the same useful function. Additives may need to be added to the system in this modification.

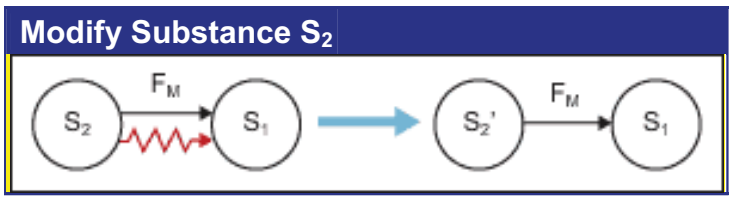


Figure 2.5: Modify substance S_2.

– General Solution 3: Modify S_1 to be Insensitive or Less Sensitive to Harmful Impact. The physical and/or chemical characteristics of substance S_1 may be altered internally or externally, so that it becomes less sensitive or insensitive to a harmful impact, as seen in Figure 2.6. The modification may be either temporary or permanent. Additives may be needed in this modification.

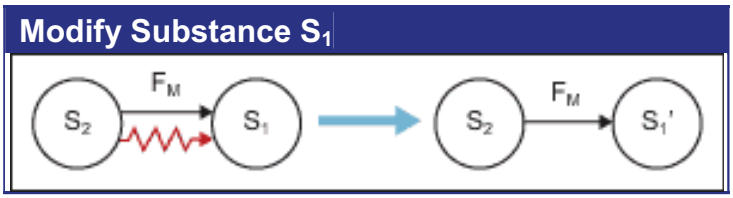


Figure 2.6: Modify substance S_1.

– General Solution 4: Change Existing Field to Reduce or Eliminate Harmful Impact. Changing the existing field while keeping the same substances may be a choice to reduce or remove the harmful impact, as shown in Figure 2.7. Changing the existing field means increasing, decreasing the existing field, or completely removing the existing field and using another one.

– General Solution 5: Eliminate, Neutralize or Isolate Harmful Impact Using Another Counteractive Field F_X. In a system in which harmful impact exists and substances S_1 and S_2 have to co-exist, a counteractive field of F_X may be introduced to remove, neutralize or isolate the harmful impact as shown in Figure 2.8. Neither S_2 nor S_1 will change its physical and chemical

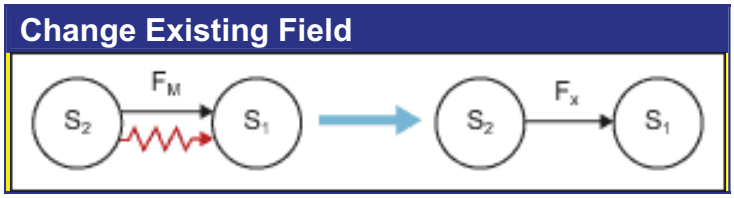


Figure 2.7: Change existing field.

characteristics in this solution. A third substance will be introduced to provide field F_X.

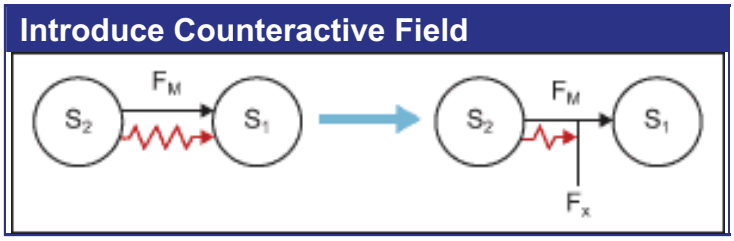


Figure 2.8: Introduce counteractive field.

– General Solution 6: Introduce a Positive Field. Another field may be added to work with the current field in order to increase the useful effect and reduce the negative effect of the existing system while keeping all substances intact, as shown in Figure 2.9.

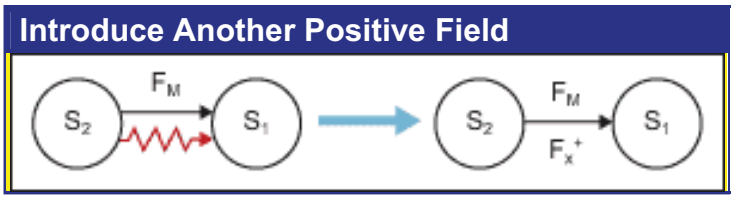


Figure 2.9: Introduce another positive field.

– General Solution 7: Expand Existing Su-Field Model to a Chain. The existing Su-Field model can be expanded to a chain by introducing a new substance S_3 to the system, as seen in Figure 2.10. Instead of directly acting upon S_1, S_2 will interact indirectly with S_1 via another medium, substance S_3.

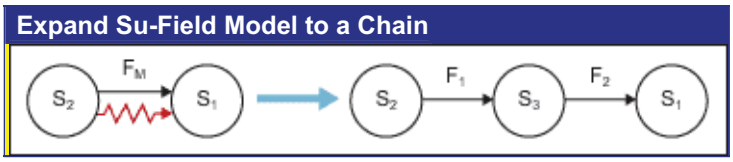


Figure 2.10: Expand Su-Field model to a chain.

- *B-class*: Thirty-one standards are identified as the implementation of existing inventive principles. On the one hand, the matches between the inventive principles and this kind of inventive standards are pre-stored in database, on the other hand, this kind of inventive principles are also considered as the candidates to be matched with the given inventive principles based on the proposed semantic similarity methods.

- *C-class*: Six standards are considered as new inventive principles since they cannot be explained using the Su-Field model but have the nature of inventive principles. The six standards are:

  - *Inventive Standard 3.1.5*: System transition: opposite features of the whole and parts.
  - *Inventive Standard 5.1.1.3*: Use an external additive instead of an internal one.
  - *Inventive Standard 5.1.1.9*: Obtain the required additive by decomposition of either the environment or the object itself.
  - *Inventive Standard 5.2.2*: Use fields that are present in the environment.
  - *Inventive Standard 5.2.3*: Use substances that are the sources of fields.
  - *Inventive Standard 5.4.2*: Strengthen the output field when there is a weak input field.

- *D-class*: Seven standards that are originated from the patterns of evolution.

  - *Inventive Standard 2.2.2* originated from *Pattern of Evolution* 6 (Transform from macro-system to micro-system).
  - *Inventive Standard 2.2.4* originated from *Pattern of Evolution* 7 (Increase dynamism and controllability).

- *Inventive Standard 3.1.2* originated from *Pattern of Evolution* 4 (Match and mismatch).
- *Inventive Standard 3.1.3* originated from *Pattern of Evolution* 7 (Increase dynamism and controllability).
- *Inventive Standard 3.1.4* originated from *Pattern of Evolution* 5 (Increase complexity, then follow with simplicity through integration).
- *Inventive Standard 3.2.1* originated from *Pattern of Evolution* 6 (Transform from macro-system to micro-system).
- *Inventive Standard 4.1.1* originated from *Pattern of Evolution* 2 (Increase ideality).

2.1.5.3 Separation Methods

An advanced form of contradictions are the physical contradictions [5]. To model an inventive problem as a physical contradiction, a physical object of a prototypical design that must have two conflicting properties has to be identified. To solve problems containing physical contradictions, separation methods (Appendix E) for physical contradiction elimination are used. Among them, there are:

- Separation of conflicting properties in time.
- Separation of conflicting properties in space.
- Separation of conflicting properties by system transition.
- Separation of conflicting properties by phase transition.
- Separation of conflicting properties by physical-chemical transition.

As stated above, these methods are very general. Consequently, they are very hard to use without additional support by knowledge that is more specific. The output of using these principles might not be directly in the form of a design concept or in the form of any specific physical function to perform. This problem might be tackled by structuring inventive principles according to more general principles for eliminating physical contradictions.

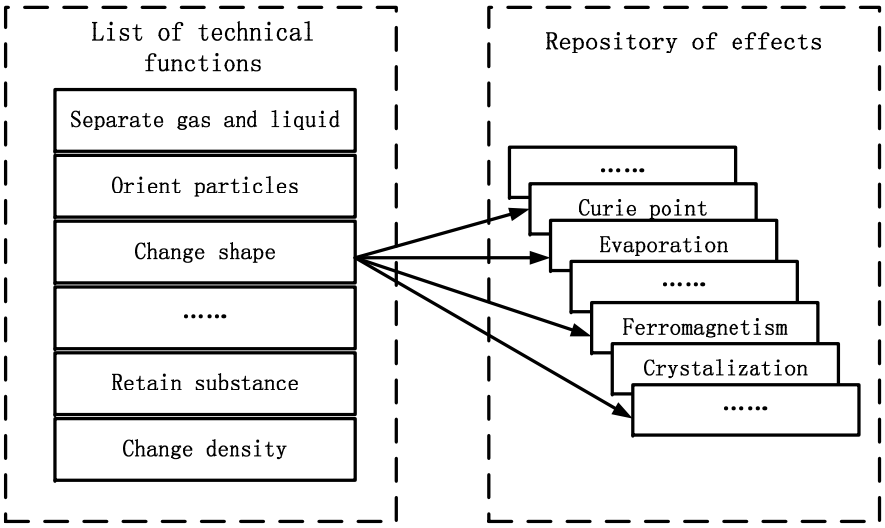


Figure 2.11: The organization of the pointers to the physical-chemical-geometrical effects.

2.1.6 Physical-Chemical-Geometrical Effects

While inventive principles, separation methods and inventive standards do not produce recommendations in terms of what physical substances or fields should be used, the collection of physical-chemical-geometrical effects (also called physical effects for short) provides the mapping between technical functions and known natural laws.

Studies of the patent collections indicated that the best and, as a consequence, more ideal inventive solutions are obtained by the use of natural phenomena previously unused in the engineering domain. Knowledge of natural phenomena often makes possible to avoid the development of complex and unreliable designs.

Example: Instead of a mechanical design including many parts for the precise displacement of an object for a short distance, it is possible to apply the effect of thermal expansion to control it.

As shown in Figure 2.11, the pointers to physical effects are defined to represent the way to use these effects. In order to facilitate this process, several different approaches and software were proposed. Sushkov [9] proposed an approach to model physical knowledge in terms of generic components and integrated inventive standards and sharable physical effects based on ontology. However, only the TRIZ knowledge sources are formalized using ontology while the further development, such as, using ontology inference to obtain

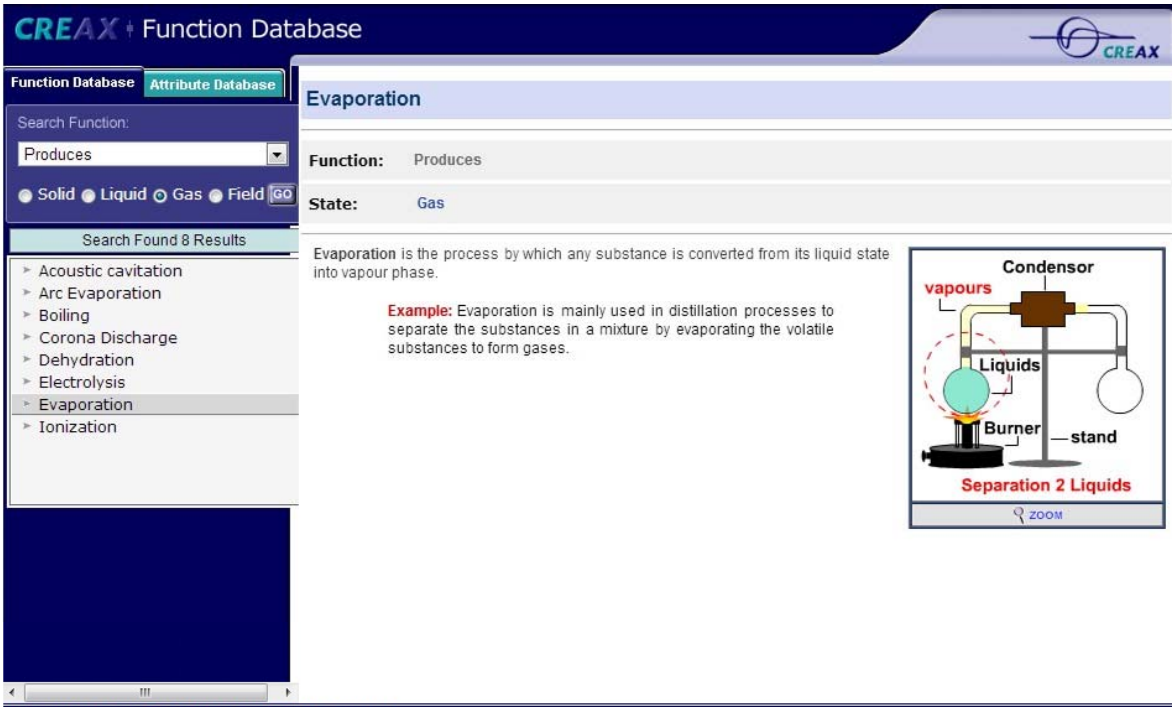


Figure 2.12: The CREAX database.

potential effects in Su-Field analysis, has not been considered.

The CREAX database [1] organises a database of effects by function, and uses a web-based application to support the search of effects. As shown in Figure 2.12, it consists of two databases, that is, the Function Database and the Attribute Database. In the Function Database, a list of Functions (Pointers), such as, "Absorb" and "Produce", and four objects - "Solid", "Liquid", "Gas" and "Field" are pre-defined. The user needs to choose a Function and an object, such as, "Absorb Gas", and then all the related physical effects are obtained. In the Attribute Database, there are a series of Attributes, which describe the different attributes of technical systems or objects, such as, "Colour" and "Speed", and five kinds of behaviour - "Changing", "Decreasing", "Measuring", "Increasing" and "Stabilising" to define the different modifications of the Attribute. Similarly, the user also needs to select an Attribute and a kind of behaviour to search for the appropriate physical effects. However, this system responds to user-entered query by processing the query to a combination of two pre-stored key words, which is quite limited in the specific

[1] http://function.creax.com/.

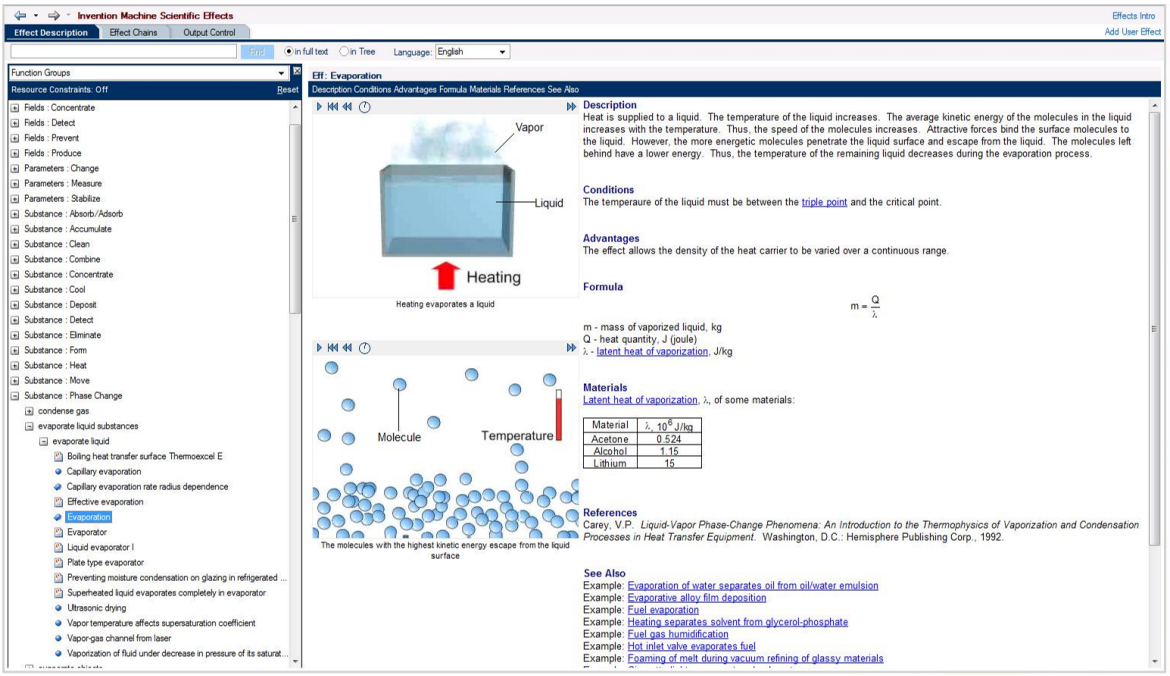


Figure 2.13: The scientific effects in the Invention Machine's Goldfire.

applications. For example, if the user query comprises words that lack all the key words pre-stored, then the system cannot respond with any stored answer or it may respond with an incorrect pre-stored answer. Furthermore, the classification of objects - "Solid", "Liquid", "Gas" and "Field" is at high level of abstraction, and too many physical effects are eligible for a specific application. In this application, the query with key word is considered as the supplementary means.

Invention Machine's Goldfire [1], a commercial software product from Invention Machine, provides tools for the organization's management of technology and for the individual's researching and engineering tasks. As shown in Figure 2.13, the similar research in GoldFire is the scientific effects tool, which helps to stimulate creative problem solving by browsing and searching effects in the Invention Machine Scientific Effects Database.

The physical effects are divided into twenty-six classes according to different functions, such as, "Substance: Absorb/Adsorb", and then for each class, there are several sub-classes, for example, for the class "Substance: Absorb/Adsorb", the sub-classes are:

[1]http://inventionmachine.com/products-and-services/innovation-software/.

- "absorb liquid substances"
- "absorb/adsorb chemical compounds"
- "absorb/adsorb gas"
- "absorb/adsorb molecular and sub-molecular particles"
- "absorb/adsorb structured substances"
- "absorb/adsorb technical objects and substances"
- "adsorb particles"

Each sub-class also consists of several sub-subsidiary classes, for example, "absorb/adsorb gas" is made up of "absorb/adsorb gas" and "absorb/adsorb vapor". As a result, the user needs to locate the specific case level-by-level to obtain the physical effects. With the good performance of the proposed semantic methods and the standard representative way of problem and solution, the results are almost satisfied. However, the hierarchy of the physical effects is stable and any modification of the classification needs to be maintained manually. Furthermore, the semantic search, only depending on the matching among two words with the same role in the sentence, cannot provide more extensive semantic information, such as, a physical effect related to "absorb vapor" is also an effect to "absorb gas".

As stated above, various existing approaches to choose physical effects are still completely or partly operated by TRIZ users, requiring a high expertise in TRIZ usage to appropriately manipulate these concepts. In order to facilitate this process, a new way to formalize physical effects and a new method of using physical effects are required, aiming at obtaining better results while requiring less expertise to complete the tasks.

2.2 Ontology-based Knowledge Modeling

2.2.1 Definition of Ontology

The term ontology originated from Philosophy, where it means a description of the nature of being (a "theory of existence"). In Artificial Intelligence (AI), the most frequently

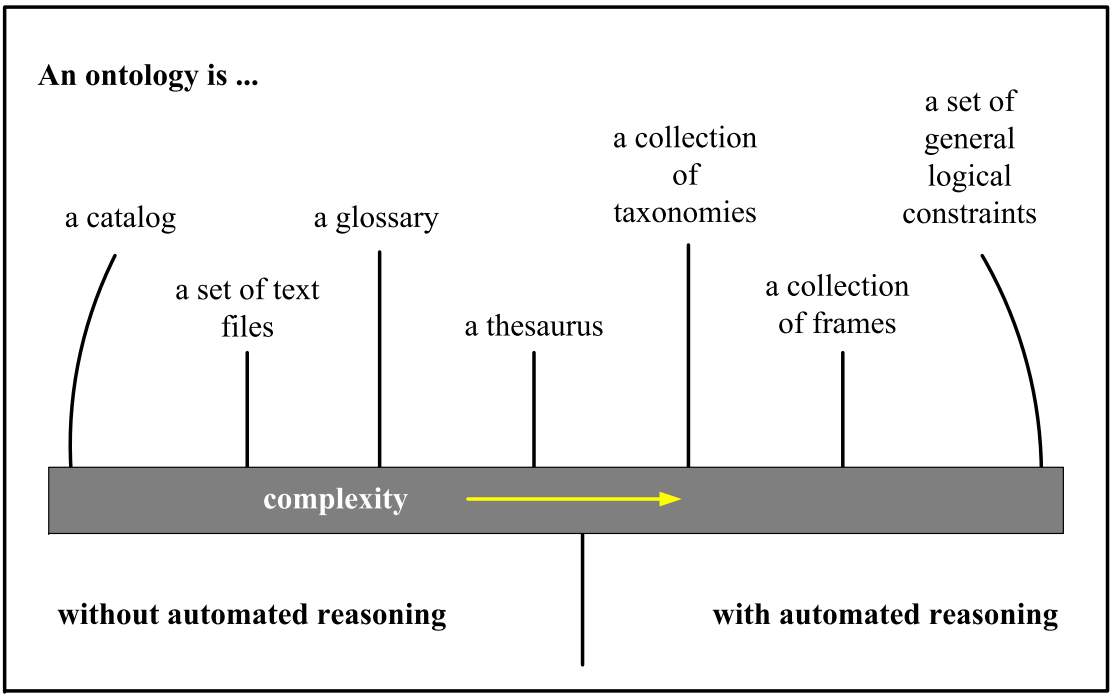


Figure 2.14: The spectrum of information artifacts.

quoted definition is "an explicit specification of a conceptualization" [10]. The main difference is an ontology in Philosophy is a systematic and exhaustive account of Existence, only driven by the task of making the ontology. When making an AI-ontology, there should always be a task in mind, which means all the related restrictions of concepts need to be included in the ontology.

According to [11] a task neutral ontology intended to represent the common sense needed by any application does not exist, nor is it realistic it ever will exist. [12] States that Gruber's definition - "an ontology is a specification of a conceptualization", left room for too many possible interpretations, and despite an attempt to clarify and formalize the definition further by Guarino [13], new meanings of the term "ontology" continued to proliferate. *Welty*, *Lehmann*, *Gruninger*, and *Uschold* reported in 1999 on a wide spectrum of information artifacts that had been at some time classified as ontologies. The results of their work are shown in Figure 2.14 [14].

- A catalog is nothing more than a numbered list of all relevant classes/concepts.
- A slightly more complex information system may provide simple natural language

texts and allow string matching.

- Glossaries are indexed catalogs with a natural language description of each concept/class.

- Thesauri are glossaries with a hierarchical structure.

- Taxonomies are thesauri where the properties of more general classes/concepts are inherited by the more specific classes/concepts.

- Frame-based systems are taxonomies, which provide relations between objects and restrictions on what and how classes/concepts of objects can be related to each other.

- The most expressive and complex information system ontologies use the axioms of full first order logic, higher order, or modal logic.

There are basically four building blocks in an ontology:

- *Classes (or sometimes called concepts)*: Classes provide an abstraction mechanism for grouping resources with similar characteristics.

- *Properties*: Properties describe the features and attributes of the classes and consist of two main categories, that is, Object properties (link individuals to individuals) and Datatype properties (link individuals to data values).

- *Restrictions*: Restrictions describe the value a property is allowed to have. A property restriction is a special kind of class description. It describes an anonymous class, namely a class of all individuals that satisfy the restriction.

- *Individuals*: Individuals are instances of a class with certain restrictions.

In order to represent the conceptual architecture of the ontologies in this research, a framework of the ontology, including main Classes and Properties, is used in the following sections.

2.2.2 Web Ontology Language - OWL

The Web Ontology Language (OWL), is an integral component of the Semantic Web, as it can be used to write ontologies or formal vocabularies which form the basis for semantic web data markup and exchange [15].

OWL is a fairly recent language, released as a W3C (World Wide Consortium) recommendation in February 2004. As a part of the Semantic Web stack of languages, OWL is layered on top of RDF (basic assertional language) and RDFS (schema language extension for RDF) which themselves are layered over XML [16]. From its relationship with RDF comes the official OWL exchange syntax, namely RDF/XML [17]. In fact, OWL shares many features in common with RDF such as the use of Universal Resource Identifiers (URI) to unambiguously refer to web resources.

OWL consists of three increasingly expressive sub-languages or "species", that is, OWL-Lite, OWL-DL and OWL-Full, each of which corresponds to different requirements [18]:

- *OWL Full*: The entire language is called OWL Full, and uses all the OWL languages primitives. It also allows to combine these primitives in arbitrary ways with RDF and RDF Schema. This includes the possibility (also present in RDF) to change the meaning of the pre-defined (RDF or OWL) primitives, by applying the language primitives to each other. For example, in OWL Full we could impose a cardinality constraint on the classification of all classes, essentially limiting the number of classes that can be described in any ontology. The advantage of OWL Full is that it is fully upward compatible with RDF, both syntactically and semantically: any legal RDF document is also a legal OWL Full document, and any valid RDF/RDF Schema conclusion is also a valid OWL Full conclusion. The disadvantage of OWL Full is the language has become so powerful as to be undecidable, dashing any hope of complete (let alone efficient) reasoning support.

- *OWL DL*: In order to regain computational efficiency, OWL DL (short for: Description Logic) is a sub-language of OWL Full which restricts the way in which the constructors from OWL and RDF can be used. The amounts to disallowing application of OWL's constructor's to each other ensure that the language corresponds to a well studied description logic. The advantage of this sub-language is that it permits efficient reasoning support and the disadvantage

is that we loose full compatibility with RDF: an RDF document will in general have to be extended in some ways and restricted in others before it is a legal OWL DL document. Conversely, every legal OWL DL document is still a legal RDF document.

- *OWL Lite*: An ever further restriction limits OWL DL to a subset of the language constructors. For example, OWL Lite excludes enumerated classes, disjointness statements and arbitrary cardinality (among others). The advantage of this sub-language is a language that is both easier to grasp (for users) and easier to implement (for tool builders) while obviously, the disadvantage is a restricted expressivity.

During the process of ontology modeling, which sub-language is used depends on many factors. However, there are some regular rules as following[1]:

- The choice between OWL Lite and OWL DL depends on the extent to which users require the more-expressive constructs provided by OWL Lite and OWL DL.
- The choice between OWL DL and OWL Full mainly depends on the extent to which users require the meta-modeling facilities of RDF Schema (e.g. defining classes of classes, or attaching properties to classes).

There are strict notions of upward compatibility between these three sub-languages [18]:

- Every legal OWL Lite ontology is a legal OWL DL ontology.
- Every legal OWL DL ontology is a legal OWL Full ontology.
- Every valid OWL Lite conclusion is a valid OWL DL conclusion.
- Every valid OWL DL conclusion is a valid OWL Full conclusion.

In this research, OWL-DL is chosen to formalize the knowledge for the following reasons:

- OWL-DL is a web-oriented ontology language and adapts to the situation of sharing domain knowledge through web-based applications and systems.

[1]For more detailed information, see the OWL Web Ontology Language Overview: http://www.w3.org/TR/owl-features.

- OWL-DL is a sophisticated language, that is, on the one hand, it shares useful language constructs and design features with its predecessors, such as RDF(S); on the other hand, it adds desirable features through appropriate extensions to satisfy conflicting requirements.

- OWL-DL uses the Description Logic (DL) model, which specifies the meaning of ontology in rigorously well-defined logic semantics and in a format that can be further processed by programming. This also enables the automatic reasoning about knowledge sources, avoiding the terminological ambiguity.

2.2.3 Ontology-based Knowledge Modeling for Inventive Design

Several works have been carried out regarding the formalization of TRIZ knowledge based on ontology.

[19] intended a formalization of the TRIZ concepts and their relationships with the operational goal of having a computer-aided TRIZ problem formulation tool. This tool progressively builds a model of the problem according to different TRIZ points of view. The given problem is formulated according to different axes: contradictions, and functional and substance-field modeling. This is assured by the developed ontology which links all the TRIZ concepts together. A new ontological description of the TRIZ domain was also engaged with the operational goal of having a TRIZ problem-solving software tool (and not a tool for pure problem formulation as in [19]). With this aim, problem-solving knowledge has to be incorporated.

[20] stated that substance-field modeling provided this kind of knowledge and proposed the enrichment of the former TRIZ ontology developed by [19]. This new development has given birth to three ontologies: a general TRIZ ontology, a substance-field ontology and a scientific-engineering effects ontology. Also, the originality of these works is the modeling of substances in terms of matter and bonding fields, which makes the relationship with the physical effects ontology straightforward.

[3] proposed the logical formalization of the IDM ontology, based on which the software tools were developed to assist the TRIZ experts in the conduction of the inventive design study, from problem formulation to the proposal of solution concepts.

There are also other attempts of formalization in the literature, but these propositions mainly rely on functional approaches, or the goal of the formalizations is not to build

a knowledge-based system to assist in the formulation and solving of inventive design problems.

[21] proposed an information infrastructure for innovative product design, and the ontology developed did not propose concepts for the analysis of complex initial situations.

[22] presented an integrated architecture for innovation knowledge acquisition and TRIZ knowledge network collaborative construction based on meta-ontologies and Web 2.0.

[23] presented the use of first-order logic to improve upon currently employed engineering design knowledge management techniques. Specifically, this work used description logics with Horn logics, not only to guide the knowledge acquisition process, but also to offer much needed support in decision-making during the engineering design process in a distributed environment.

[24] proposed an ontology used to capture, retrieve and reuse novel bio-inspired design solutions and their associated physical architectures, behaviours, functions and strategies. These works intended to facilitate the use of the pointers to scientific-engineering effects.

Finally, [25] proposed the use of functional analysis with TRIZ-based tools to improve the efficiency of Altshuller's laws of engineering system evolution as a means to produce technology forecasts.

However, their research is concerned with one or more parts of the TRIZ knowledge souces, not yielding a comprehensive formalization for the TRIZ knowledge sources and their relations. Consequently, this thesis focuses on the formalization of the TRIZ knowledge sources, to complete the model and make it wholly consistent by the definition of the missing semantic links. This formalization should allow the development of an intelligent management system of these knowledge sources, with the aim of assisting the TRIZ expert during his activity.

2.3 Ontology Reasoning

2.3.1 Semantic Web Rule Language - SWRL

The SWRL [26], proposed by W3C, is an expressive OWL-based rule language. It is based on a combination of OWL DL and OWL Lite sub-languages of OWL with the

Unary/Binary Datalog RuleML sub-languages of the Rule Markup Language[1]. The SWRL proposal extends the OWL abstract syntax to include the syntax of these rules and the OWL model-theoretic semantics to provide a formal meaning for ontologies that include rules written in this syntax. The extension is strictly a syntactic and semantic extension, hence has a tight integration to OWL.

2.3.1.1 SWRL Syntax

A SWRL rule consists of an antecedent (body) and a consequent (head), both consisting of one or more atoms as following:

$$Antecedent(Atom \wedge Atom...) \Rightarrow Consequent(Atom \wedge Atom...) \tag{2.1}$$

In a rule with multiple atoms in the body, the body is treated as a conjunction of its atoms. In a rule with multiple atoms in the head, the head is also treated as a conjunction. But such a rule can be easily transformed into multiple rules each with a single-atom head [26]. Rule atoms can be of the form[2]:

- *A(x)* - where A is an OWL Class (a simple named class or a class description) or a data range[3] and x is either a variable, OWL individual or OWL data value.
- *P(x,y)* - where P is an OWL Property (an object property or a datatype property), x is either a variable or an OWL individual and y is either a variable, an OWL individual or an OWL data value.
- *sameAs(x,y)*, *differentFrom(x,y)* - where x, y are variables or OWL individuals.
- *builtIn(r,x,...)* - where r is a built-in relation and x,... are OWL data values.

A(x) can be referred to as a concept atom/class atom, *P(x,y)* can be referred to as a role atom/property atom. According to [26], variables in a rule are treated as universally quantified, limited to the scope of that rule. Rules must be safe, i.e., only variables that

[1] http://www.ruleml.org.

[2] The abstract syntax of SWRL rule axioms, in a version of extended BNF notation, is given in http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/#2.

[3] http://www.w3.org/TR/owl-ref/#DataRange.

occur in the rule body may occur in the rule head[1]. The syntax of SWRL does not allow disjunction of atoms, negation of atoms or any non-monotonic features like negation as failure or defaults.

An informal human-readable syntax for these rules is also specified for ease of readability and a typical rule in this syntax would look like:

$$hasParent(?x, ?y) \wedge hasBrother(?y, ?z) \Rightarrow hasUncle(?x, ?z) \tag{2.2}$$

It states that an individual x having a parent y who has a brother z, has an uncle z.

2.3.1.2 SWRL Semantics

The SWRL semantics, a straightforward extension of the OWL semantics, is considered as a logical implication between its body and head. The intended meaning is that: whenever the conditions in the rule body hold, then the conditions in the rule head must also hold.

According to the SWRL proposal [26], the formal semantics for the rules are given by an extension of the OWL interpretation by defining bindings. These bindings map variables in rules to elements in the ontology domain, and an interpretation satisfies an ontology if and only if (iff) it satisfies every axiom (OWL axioms and rule axioms) and fact in the ontology:

- A rule is satisfied by an interpretation iff every binding that satisfies its body also satisfies its head.
- A rule body or rule head is satisfied if the conjunction of its atoms is satisfied.
- A concept atom *A(x)* is said to be satisfied if x is an instance of the class description or data range A.
- A role atom *P(x,y)* is said to be satisfied if x is related to y by role P.
- A $sameAs(x, y)$ atom is said to be satisfied if x is interpreted as the same object as y.

[1]Unsafe rules allow variables in the head that do not occur anywhere in the body. If this is allowed, the definition of bindings for variables cannot be defined correctly in this context. Also, termination of reasoning cannot be guaranteed as all the individuals in the KB could be used to bind the variable in the head. E.g., the rule $Child(?x) \Rightarrow isInnocent(?y)$ is unsafe.

- A $differentFrom(x, y)$ atom is said to be satisfied if x and y are interpreted as different objects.

- A $builtIn(r, x, ...)$ atom is said to be satisfied if the built-in relation r holds on the interpretations of the arguments.

Since a rule is treated as a logical implication, the implied meaning is that the contraposition law holds on the implication (rule). For example, whenever the rule:

$$hasParent(?x, ?y) \wedge hasBrother(?y, ?z) \Rightarrow hasUncle(?x, ?z) \tag{2.3}$$

holds, its contrapositive implication:

$$\neg hasUncle(?x, ?z) \Rightarrow \neg(hasParent(?x, ?y) \wedge hasBrother(?y, ?z)) \tag{2.4}$$

also holds.

In this application, there are two reasons for choosing SWRL to represent the reasoning rules:

- SWRL has the advantages of formal model-theoretic semantics and the close association with OWL, and it can well integrate the OWL-based TRIZ knowledge sources ontologies and physical effects ontology into the rules.

- SWRL is a descriptive language that is independent of concrete rule implementation languages within inference engines, which makes it free to select the inference platforms.

2.3.2 A Rule Engine - Jess

Generally, there are two motivations for choosing an inference engine:

- There is a need for a data representation to enable software products (agents) to provide intelligent access to heterogeneous and distributed information.

- To provide tools that drive down the cost of establishing inter-operability between different data providers.

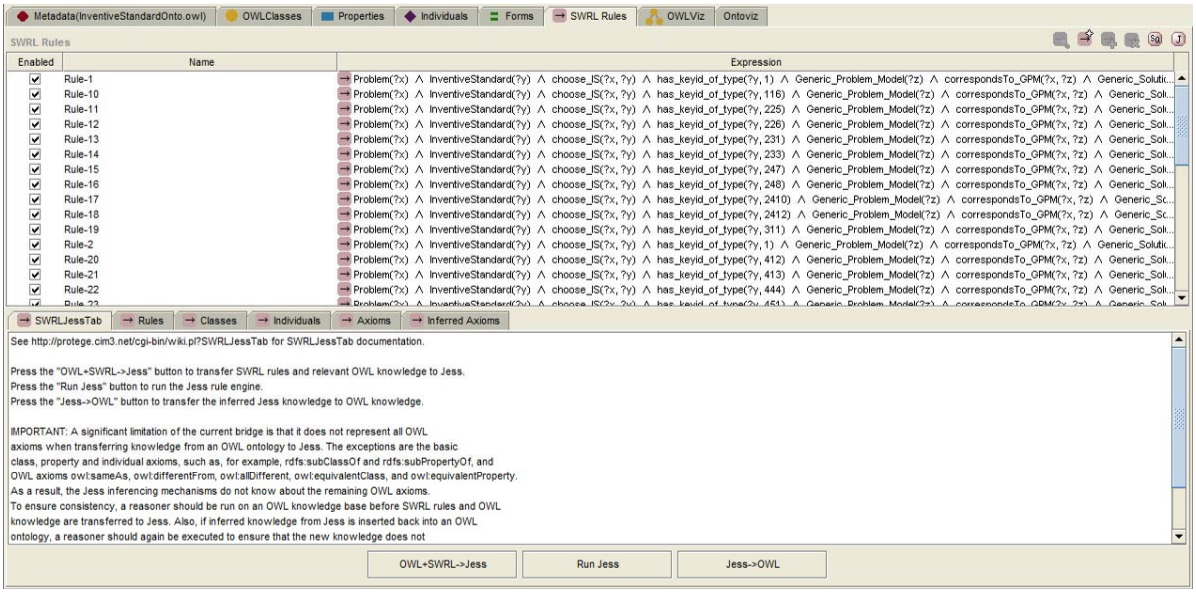


Figure 2.15: The Protégé SWRLJESSTab visualization.

The existing inference engines can be categorized into three classes according to the logic based on which they are implemented[27]:

- Description Logic based Inference Engine: This kind of DL reasoners are used to perform basic reasoning tasks like consistency checking and subsumption concepts. There are many DL reasoners, such as Racer[28] and Pellet[29].

- First Order Logic (FOL) Theorem Prover based Inference Engine: These reasoners work on First Order Logic. The reasoner takes OWL file as input and first translated into FOL, and then inference is processed by using any one of existing automated theorem prover. Semantic reasoners like Hoolet[1] uses Vampire theorem prover, Surnia[2] uses Otter theorem prover are FOL based reasoner.

- Combination of FOL and General Logic based Inference Engine: This type of reasoners are designed based on FOL and general logic. Horn Logic[3] is most widely used due to its simplicity and availability. The reasoners Java Expert System

[1]http://owl.man.ac.uk/hoolet.
[2]http://www.w3.org/2003/08/surnia/.
[3]http://www.w3.org/2005/rules/wg/wiki/Horn_Logic.

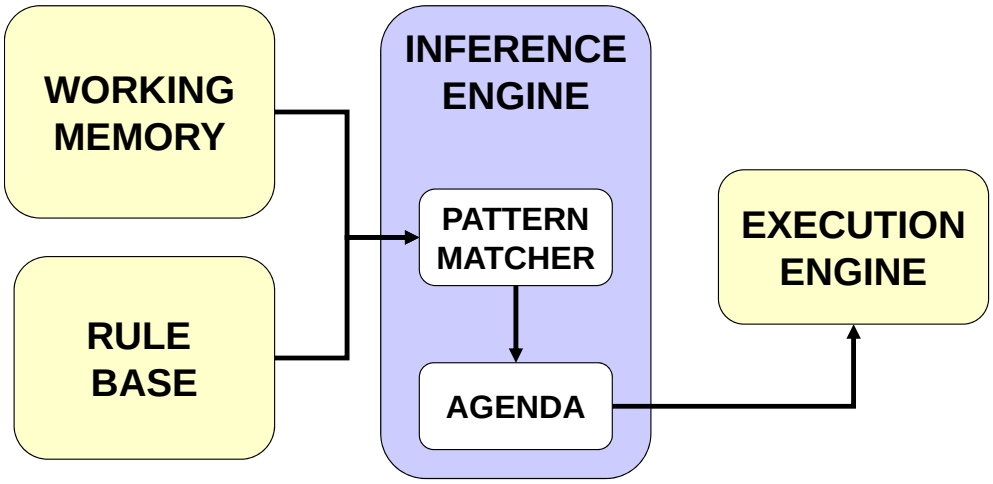


Figure 2.16: The architecture of Jess.

Shell(Jess)[30] and Jena[1] belong to this kind.

In our application, Jess is selected as the inference engine for the following four reasons:

- Jess works seamlessly with Java and is well documented, which makes easy to use and configure.
- Jess supports mainly forward chaining and, to some extent, backward chaining.
- For user-driven application, there exists *SWRLJESSTab*[2] integrated with the software Protégé as shown in Figure 2.15. The interaction between OWL/SWRL and the JESS rule engine is user-driven with the *SWRLJESSTab*.
- For Java application, there exists *SWRLRuleEngineBridge*, a part of Protégé-OWL. This bridge is used to import SWRL rules and relevant OWL classes, individuals and properties from an OWL model and write that knowledge to a rule engine, allow the rule engine to perform inference and to assert its new knowledge back, and insert that asserted knowledge into an OWL model.

The typical rule-based engine keeps a list of rules and continuously cycles through the list, checking each rule's left-hand-side (LHS) against the working memory and executing

[1]http://jena.apache.org/.
[2]http://protege.cim3.net/cgi-bin/wiki.pl?SWRLJessTab.

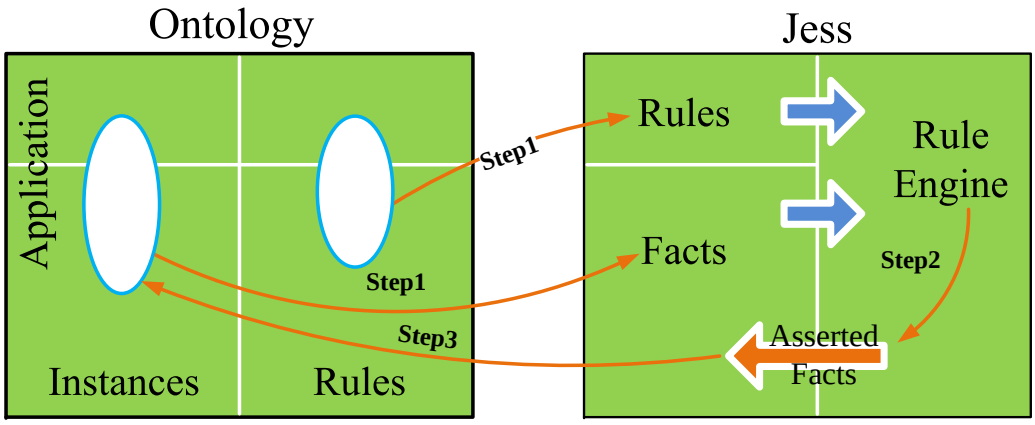


Figure 2.17: The inference process in Jess.

the right-hand-side (RHS) of any rules that apply. As a rule engine, Jess uses a more efficient method known as the Rete algorithm than the above way [31], that is, by remembering past testing results across iterations of the rule loop, only new facts are tested against any rule LHSs to which they are most likely to be relevant.

As shown in Figure 2.16, Jess contains a rule base, a working memory, an inference engine and an execution engine. The inferences are carried out in Jess inference engine by matching facts in working memories against the rules in the rule base. By transforming structural knowledge in OWL into Jess facts, and constraint knowledge in SWRL into Jess rules, the inferred results are generated by the Jess inference engine and execution engine.

As shown in Figure 2.17, the inference process of Jess includes three steps:

- Step 1: OWL + SWRL → Jess will transfer the appropriate OWL knowledge and the rule knowledge in SWRL to Jess. This transformation are relatively by straightforward using the "OWL + SWRL → Jess" button in *SWRLJESSTab*, or using the command *importSWRLRulesAndOWLKnowledge()* in the interface *SWRLRuleEngine*[1].

- Step 2: After importing Jess files containing both Jess facts and Jess rules, the inference process is implemented by executing Jess. Pressing the "RUN Jess" button in *SWRLJESSTab* or use the command *run()* can initiate this process.

[1]http://protege.cim3.net/cgi-bin/wiki.pl?SWRLRuleEngineAPI.

- Step 3: The Jess facts, including asserted facts and inferred facts, are back to OWL knowledge by using the "Jess → OWL" button in *SWRLJESSTab* or use the command *writeInferredKnowledge2OWL()*.

Generally, Step 2 is executed automatically in Java applications, so only Step 1 and Step 3 need to be considered, which can be divided into the following three phases:

- The Function of OWL → Jess.
- The Function of SWRL → Jess.
- The Execution of the Jess Inference Engine.

2.3.3 Ontology Reasoning for Inventive Design

Due to the characteristics of TRIZ knowledge, described by short texts in natural language rather than ontology language, it is necessary to firstly formalize the corresponding TRIZ knowledge sources based on ontologies, and then facilitate their use based on ontology reasoning.

As an emerging technology, ontology reasoning has not been widely used in the domain of inventive design. In order to support and simulate the process of Su-Field analysis, Bultey et al firstly established several ontologies to formalize the concepts and relations in Su-Field analysis, and then built an ideal TRIZ reasoning environment based on ontology inference[32]. However, this research focuses on the Su-Field analysis and the use of inventive standards, without considering other TRIZ knowledge sources and their relations.

Accordingly, this thesis firstly formalizes the TRIZ knowledge sources, and then develops an intelligent management system based on ontology reasoning. Indeed, during the processing of a new case, experts are brought to work with various models at different levels of abstraction. The knowledge management system should suggest the experts the use of the relevant knowledge sources, consistent with the level of abstraction of the model they are building. It will also be able to complete "automatically" the rest of the models, by exploiting the semantic links obtained among the different knowledge sources.

2.4 Summary

As the background of our research, the detailed information about TRIZ - the definition of TRIZ, the application steps of TRIZ solving inventive problems, and the basic tools of TRIZ are firstly presented. Then in order to exploit an automatic ontology-based mechanism for solving inventive problems, the knowledge about ontology modeling and the mechanism of ontology reasoning, are introduced, including their basic methods, concepts and the current situation of their use, especially for the applications in the field of inventive design.

In the next chapter, the methods of calculating semantic similarity, which are foundations of the following work in the automatic ontology-based mechanism, will be introduced.

Bibliography

[1] S. Savrancky, *Introduction to TRIZ Methodology of Inventive Problem Solving*, C. Press, Ed. CRC Press, 2000.

[2] C. Zanni-Merk, D. Cavallucci, and F. Rousselot, "An ontological basis for computer aided innovation," *Computers in Industry*, vol. 60, pp. 563–574, Oct. 2009.

[3] ——, "Use of formal ontologies as a foundation for inventive design studies," *Computers in Industry*, vol. 62, pp. 323–336, Apr. 2011.

[4] G. Cascini and D. Russo, "Computer-aided analysis of patents and search for triz contradictions," *International Journal of Product Development*, vol. 4, pp. 52–67, 2007.

[5] G. Altshuller, *Creativity as An Exact Science.* New York: Gordon and Breach Scientific Publishers, 1984.

[6] K. Gadd, *TRIZ for Engineers: Enabling Inventive Problem Solving.* John Wiley and Sons, Ltd, 2011.

[7] H. Cong and L. Tong, "Grouping of triz inventive principles to facilitate automatic patent classification," *Expert Systems with Applications*, vol. 34, no. 1, pp. 788–795, 2008.

[8] X. Mao, X. Zhang, and S. AbouRizk, "Generalized solutions for su-field analysis," *The TRIZ Journal*, 2007.

[9] V. Sushkov, L. Alberts, and N. Mars, "Innovative engineering design based on sharable physical knowledge," in *In: Proceedings of the International Conference Artificial Intelligence in Design (96).* Kluwer Academic Publishers, 1996, pp. 723–742.

[10] T. Gruber, "A translation approach to portable ontology specifications," *Knowledge Acquisition*, vol. 5, pp. 199–220, 1993.

[11] T. Bench-Capon, "Task neutral ontologies, common sense ontologies and legal information systems," in *Second International Workshop on Legal Ontologies*, 2001.

[12] B. Smith and C. Welty, "Ontology: Towards a new synthesis," in *In: Proceedings of the International Conference On Formal Ontology In Information Systems (FOIS'01)*. New York: ACM Press, 2001, pp. 3–4.

[13] N. Guarino, "Formal ontology and information systems," in *In: Proceedings of the International Conference On Formal Ontology In Information Systems (FOIS'98)*. Amsterdam: IOS Press, 2001, pp. 3–15.

[14] C. Welty, F. Lehmann, G. Gruninger, and M. Uschold, "Ontology: Expert systems all over again?" in *Invited panel at AAAI-99: The National Conference on Artificial Intelligence*, 1999.

[15] M. Dean and G. Schreiber, "Owl web ontology language reference w3c recommendation," 2004. [Online]. Available: http://www.w3.org/tr/owl-ref/.

[16] T. Bray, J. Paoli, and C. Sperberg-McQueen, "Extensible markup language (xml)," *World Wide Web Journal*, vol. 2, pp. 27–66, 1997.

[17] D. Beckett and B. Mcbride, "Rdf/xml syntax specification," *W3C Recommendation*, 2004.

[18] G. Antoniou and F. van Harmelen, "Web Ontology Language: OWL," *Handbook on Ontologies*, pp. 67–92, 2004. [Online]. Available: http://jeogoodjob.250free.com/data/OntoHandbook03OWL.pdf.

[19] S. Dubois, P. Lutz, F. Rousselot, and G. Vieux, "A model for problems' representation at various generic levels to assist inventive design," *International Journal of Computer Applications in Technology*, vol. 30, pp. 105–112, 2007.

[20] A. Bultey, F. D. Beuvron, C. Zanni-Merk, and F. Rousselot, "A hybrid system combining description logics and rules for inventive design," in *In: Proceedings of the 13th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems (KES 2009)*, vol. 5711 LNAI, 2009.

[21] I. Paik, K. Kim, W. Park, and R. Komiya, "Information infrastructure for innovative product design in scm using semantic triz," in *In: Proceedings of the 3rd International Conference on Awareness Science and Technology (iCAST 2011)*, 2011.

[22] S. Huang, F. Xu, F. Dai, Y. Zhang, X. Gu, and G. Qi, "Integrated architecture for triz based on knowledge network and its key technologies," *Journal of Zhejiang University (Engineering Science)*, vol. 45, no. 8, pp. 1337–1345, 2011.

[23] P. Witherell, S. Krishnamurty, I. Grosse, and J. Wileden, "Improved knowledge management through first-order logic in engineering design ontologies," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 24, pp. 245–257, 2010.

[24] S. Yim, J. Wilson, and D. Rosen, "Development of an ontology for bio-inspired design using description logics," in *In: Proceedings of the International Conference on Product Lifecycle Management*, 2008.

[25] G. Cascini and F. Rotini, "Functional modelling for triz-based evolutionary analyses," in *In: Proceedings of the 17th International Conference on Engineering Design (ICED)*, 2009, pp. 1–14.

[26] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean, "Swrl: A semantic web rule language combining owl and ruleml," in *W3C Member Submission*, 2004.

[27] N. Dalwadi, B. Nagar, and A. Makwana, "Semantic web and comparative analysis of inference engines," *International Journal of Computer Science and Information Technologies*, vol. 3, no. 3, pp. 3843–3847, 2012.

[28] V. Haarslev and R.Möller, "Racer: A core inference engine for the semantic web," 2003.

[29] B. Parsia and E. Sirin, "In: Proceedings of the international semantic web conference," 2005.

[30] E. Friedman-Hill, "Jess, the rule engine for the java platform." [Online]. Available: http://www.jessrules.com/jess/docs/Jess71p2.pdf.

[31] C. Forgy, "Rete: A fast algorithm for the many pattern/many object pattern match problem," *Artificial Intelligence*, vol. 19, pp. 17–37, 1982.

[32] A. Bultey, F. D. Beuvron, and F. Rousselot, "A substance-field ontology to support the triz thinking approach," *International Journal of Computer Applications in Technology*, vol. 30, no. 1-2, pp. 113–124, 2007.

Chapter 3

Semantic Similarity

This chapter presents the methods for calculating semantic similarity. Firstly, a literature review about semantic similarity is presented, and then the related methods for calculating semantic similarity in the domain of inventive design are given. Finally, two proposed methods including a basic method and an improved method are introduced in detail, to be used in the following sections.

3.1 Literature Review

3.1.1 Basic Methods

Semantic similarity is a measure of the likeness of meaning or semantic content assigned to a set of documents or to terms within term lists.

Semantic similarity is a special case of relatedness, which is applicable to taxonomic (IS-A) links [1]. As OWL ontologies are based on class structures which are comprised of IS-A links, semantic similarity measures are directly applicable to OWL ontologies. The methods developed to measure semantic similarity based on thesaurus can be categorized in three groups: Path-based Similarity, Information Content (IC) Similarity and Extended Gloss Overlaps.

Path-based Similarity is calculated according to the intuition: two concepts (senses) are similar if they are near each other in the thesaurus hierarchy. $Pathlen(c1,c2)$ is defined as $1 + number$ of edges in the shortest path in the hypernym graph between two concepts

$c1$ and $c2$, and the similarity between $c1$ and $c2$ is:

$$sim_{path}(c_1, c_2) = \frac{1}{pathlen(c_1, c_2)} \tag{3.1}$$

In IC Similarity [1], *P(c)* - the probability that a randomly selected word in a corpus is an instance of concept c, is firstly proposed to represent IC:

$$IC(c) = -logP(c) \tag{3.2}$$

Resnik method [1] asserts that the more specific (least probable) the nearest common concept under which two concepts can be categorized (lowest common subsume, LCS), the more similar the concepts are, that is,

$$sim_{Resnik}(c_1, c_2) = -logP(LCS(c_1, c_2)) \tag{3.3}$$

On the contrary, *Lin* proposed that the more differences between two concepts, the less similar they are, and defined in [2]:

$$sim_{Lin}(c_1, c_2) = \frac{2logP(LCS(c_1, c_2))}{logP(c_1) + logP(c_2)} \tag{3.4}$$

In [3], *Jiang* and *Conrath* used the form of the conditional probability of encountering an instance of a child set of synonyms given an instance of a parent set of synonyms, and the similarity is depicted as following:

$$sim_{Jiang\,and\,Conrath}(c_1, c_2) = \frac{1}{logP(c_1) + logP(c_2) - 2logP(LCS(c_1, c_2))} \tag{3.5}$$

The Extended Gloss Overlaps measure [4] words by finding overlaps in the glosses of the two synsets (Section 3.3.1), and the relatedness score is the sum of the squares of the overlap lengths.

3.1.2 Methods of Calculating Semantic Similarity between Short Texts

In recent years, several different approaches were proposed to measure the similarity between short texts, and the available work can be classified into three major categories: word co-occurrence/vector-based document model methods, corpus-based methods, and descriptive feature-based methods.

The vector-based document model methods are commonly used in Information Retrieval (IR) systems [5], where the document most relevant to an input query is determined by representing a document as a word vector and matching the query to the similar documents in the document database via a similarity metric [6]. This technique relies on the assumption that more similar documents have more words in common. But it is not always the case that texts with similar meaning necessarily share many words. The sentence representation is not efficient enough since the vector dimension is much larger than the number of words in a short-text, thus, resulting in many null vector components.

The corpus-based similarity includes two well-known methods, the Latent Semantic Analysis (LSA) [7, 8] and the Hyperspace Analogues to Language (HAL) model [9]. LSA uses Singular Value Decomposition (SVD) to find the semantic representations of words by analyzing the statistical relationships among words in a large corpus of text. It does not take into account any syntactic information and is thus more appropriate for longer texts. The HAL method uses lexical co-occurrence to produce a high-dimensional semantic space, where words are represented as points, the position of each word along the axes is related to the word's meaning, and the distance between two words measure their relationships. However, the word-by-word matrix does not capture sentence meaning well and the sentence vector becomes diluted as a large number of words are added to it [10].

The feature-based methods try to represent a sentence using a set of predefined features. Similarity between two texts is obtained through a trained classifier. Searching of effective features and obtaining values for these features from sentences make this kind of methods more impractical.

3.2 Semantic Similarity for Inventive Design

To facilitate the automatic classification of patent documents according to inventive principles, Cong and Tong [11] grouped forty inventive principles according to text similarity and meaning similarity. Keeping the classical Matrix unchanged, for all previous patent documents, they obtained the results of classification automatically and for each new problem, they easily sorted it and reused the solutions of patents of the same kind.

As stated in Section 2.1.6, Invention Machine's Goldfire is a software that links a given function search to a compliant list of sentences extracted in both selected websites and patents that seemingly fit the required search. Its key technology is "Semantic Answering System and Method" [12], that is, responding the user query using semantic methods. On the one hand, the solutions are stored in the form Subject-Action-Object in the database; On the other hand, the problem statement for each user query is in the form Action-Object. So if the terms in Action-Object problem statement have similarity with at least one term of the Subject-Action-Object solution, then this solution will be returned.

In order to enhance the capability of developing innovative products of modern design tools, a knowledge portal based on semantic capabilities of text mining technology was proposed in [13] to supply the design with all the required information. The case of "designing a polymeric wheel" in the field of plastic design is considered as an example, and the integrated adoption of the knowledge based tools both for the conceptual design phase (TRIZ-based application) and for the detailed design phase (semantic knowledge portal) has provided satisfactory results in terms of augmented creativity and design efficiency.

Compared with the research stated above, this thesis aims at calculating the semantic similarity among all the TRIZ knowledge sources instead of a single knowledge source. At the same time, different preprocesses are implemented for different knowledge sources to ease their matching. Furthermore, in order to improve the performance, the match with word weight in double direction is proposed (Section 3.3.3).

3.3 Our Proposal

3.3.1 The Dictionary-WordNet

WordNet, proposed by George A. Miller in 1980s, is a large lexical database of English [14]. It groups English words into sets of synonyms, namely, "synset", and the sense of each concept is given by a word or a synset in the same contexts. A term can have several senses (that is, a term can belong to several different concepts), according to the context.

WordNet is organized in a hierarchy among concepts, based on an "is-a" relationship. Sub-concepts of a certain concept are called "hyponyms" (as they are more specific than it) and super-concepts are called "hypernyms". Several other relationships among concepts appear in WordNet, such as "synonym" or "antonym".

Compared with other lexical dictionaries, such as Framenet[1], WordNet is chosen as the dictionary for the calculation of semantic similarity owing to the following reasons:

- WordNet interlinks not only word forms, for example, strings of letters, but also specific senses of words. As a result, words that are found in close proximity to one another in the network are semantically disambiguated.
- WordNet labels the semantic relations among words, whereas the groupings of words in a thesaurus does not follow any explicit pattern other than meaning similarity.

3.3.2 Basic Method

The basic matching process mainly includes the following five steps:

1. *Short Text Segmentation*: There are three methods, that is, sentence segmentation (e.g., use Decision Tree [15] to determine a word is end of sentence or not), word tokenization (e.g., substance appearance - disappearance → <substance, appearance, disappearance>), and word normalization and stemming (e.g., automates, automatic, automation → automat).
2. *Sense Search*: For each word obtained, WordNet is used to look for their corresponding senses, including nouns, verbs, adjectives and adverbs. For example, the noun form of the word 'way' has twelve different senses.

[1] https://framenet.icsi.berkeley.edu.

3. *Sense Similarity*: *Lin*'s measure is used to calculate the semantic similarity of the senses of two words. With this measure, it is obvious that in WordNet, the higher the rate of sharing information, the more similar two concepts.

4. *Word Similarity*: The maximum sense similarity of the two words is chosen as their word similarity.

5. *Short Text Similarity*: Short text similarity is calculated based on word similarity. Assumed that two sentences: A_1, including words sequence $A_{11}, A_{12} \cdots A_{1n}$ and A_2, including $A_{21}, A_{22} \cdots A_{2m}$. $s(A_{1i}, A_{2j})$ represents word similarity of A_{1i} and A_{2j},$1 \leqslant i \leqslant n$, $1 \leqslant j \leqslant m$. The matrix $M(A_1, A_2)$ can be built as following:

$$\begin{bmatrix} s(A_{11}, A_{21}) & s(A_{11}, A_{22}) & \cdots & s(A_{11}, A_{2m}) \\ \cdots & & & \\ s(A_{1i}, A_{21}) & s(A_{1i}, A_{22}) & \cdots & s(A_{1i}, A_{2m}) \\ \cdots & & & \\ s(A_{1n}, A_{21}) & s(A_{1n}, A_{22}) & \cdots & s(A_{1n}, A_{2m}) \end{bmatrix} \tag{3.6}$$

Generally, the most similar words in A_2 for each word in A_1 are selected and then the average value is calculated as following:

$$s(A_1, A_2) = \frac{\sum_{i=1}^{n} Max_{j=1}^{m}(s(A_{1i}, A_{2j}))}{n} \tag{3.7}$$

3.3.3 Improved Method with Word Weight

The basic matching approach asserts that all the words in the short-text play an equal role in calculating the semantic similarity. According to the analysis of the documents of TRIZ knowledge sources, words act differently on the short text similarity, for example, the word "transition" can be used to separate several short texts from others, while the words "a" or "an" are useless. As a result, word weight (word frequency), is proposed here to improve the results of the calculation. Given a word in a short text, its word weight is the ratio of the number of times it appears in the short text to the total number of times all words appear in this short text.

There is an intuition that the more frequently the word appears, the higher weight it has, and obviously it plays a more important part in the short text comparison. Nevertheless, some words appear too frequently, but they are not good keywords to distinguish relevant and non-relevant short texts, such as the word "the". These words are given high weight without giving enough weight to the more meaningful words, for instance, the word "atmosphere".

Hence an improved method is proposed with the word weight calculated by tf * idf, which were proposed by Gerard Salton [6] and Karen Sparck Jones [16] respectively.

tf, the word frequency, also called *Local Term Weight*, is defined as the number of times a word occurs in a document (short text in our application). For the word w in the particular short text s, the word frequency $tf(w, s)$ can be obtained.

idf, the inverse document frequency, also called *Global Term Weight*, is based on counting the number of documents (short texts in our application) in the collection being searched that are indexed by the term. According to this research, $idf(w, S)$ is defined as following:

$$idf(w, S) = log\frac{|S|}{|s \in S : w \in s|} \tag{3.8}$$

with $|S|$: cardinality of S (the total number of short texts in the corpus) and $s \in S : w \in s$: number of short texts where the word w appears (i.e., $tf(w, s) \neq 0$).

Finally, the $tf * idf$ is calculated as

$$tf * idf(w, s, S) = tf(w, s) \times idf(w, S) \tag{3.9}$$

Given two short texts A_1 and A_2, the most similar words in A_2 for each word in A_1 are selected. However, it is not enough for us to estimate the similarity between A_1 and A_2 without considering the inverse situation, that is, obtain the most similar words in A_1 for each word in A_2. Taking this into account, the semantic matching with word weight is proposed:

$$s(A_1, A_2) = \frac{\sum_{i=1}^{n} ww_{1i} \times Max_{j=1}^{m}(s(A_{1i}, A_{2j}))}{n} + \frac{\sum_{i=1}^{m} ww_{2i} \times Max_{j=1}^{n}(s(A_{2i}, A_{1j}))}{m} \tag{3.10}$$

where $s(A_{1i}, A_{2j})$ represents word similarity of A_{1i} and A_{2j},$1 \leqslant i \leqslant n$, $1 \leqslant j \leqslant m$, and $s(A_{2i}, A_{1j})$ represents word similarity of A_{2i} and A_{1j},$1 \leqslant i \leqslant m$, $1 \leqslant j \leqslant n$, calculated

as stated in basic matching. ww_{1i}: the word weight of the ith word in A_1, and ww_{2i}: the word weight of the ith word in A_2. Both of them could be calculated by using the methods $tf * idf$.

3.4 Summary

This chapter introduces the methods for calculating semantic similarity. Firstly, a literature review about existing methods is presented, and then the applications of semantic similarity in the domain of inventive design are given. Finally, two proposed methods are explained in detail. In Chapter 4, the basic method (Section 3.3.2) will be used to match the parameters in the process of using inventive principles, and in chapter 5, the improved method (Section 3.3.3) will be used to define the missing links among the TRIZ knowledge sources.

Bibliography

[1] P. Resnik, "Using information content to evaluate semantic similarity in a taxonomy," *In: Proceedings of the 14th International Joint Conference on Artificial Intelligence*, vol. 1, pp. 448–453, 1995.

[2] D. Lin, "An information-theoretic definition of similarity," *In: Proceedings of the 15th International Conference on Machine Learning, ICML '98*, pp. 296–304, 1998.

[3] J. Jiang and D. Conrath, "Semantic similarity based on corpus statistics and lexical taxonomy," *In: Proceedings of the International Conference on Research in Computational Linguistics*, pp. 19–33, 1997.

[4] M. Lesk, "Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone," *In: Proceedings of the SIGDOC Conference*, 1986.

[5] C. Meadow, B. Boyce, and D. Kraft, *Text Information Retrieval Systems*, S. edition, Ed. Academic Press, 2000.

[6] G. Salton and M. Lesk, *Computer Evaluation of Indexing and Text Processing.* Prentice, Hall, Inc. Englewood Cliffs, NJ, 1971.

[7] T. Landauer and S. Dumais, "A solution to platos problem: The latent semantic analysis theory of the acquisition, induction, and representation of knowledge," *Psych. Rev*, vol. 2, pp. 211–240, 1997.

[8] T. Landauer, P. Foltz, and D. Laham, "Introduction to latent semantic analysis," *Dis. Proc*, vol. 2-3, pp. 259–284, 1997.

[9] C. Burgess, K. Livesay, and K. Lund, "Explorations in context space: Words, sentences, discourse," *Dis. Proc*, vol. 2-3, pp. 211–257, 1998.

[10] Y. Li, D. Mclean, Z. Bandar, J. O'shea, and K. Crockett, "Sentence similarity based on semantic nets and corpus statistics," *IEEE Trans. Knowl. Data Eng*, vol. 8, pp. 1138–1149, 2006.

[11] H. Cong and L. Tong, "Grouping of triz inventive principles to facilitate automatic patent classification," *Expert Systems with Applications*, vol. 34, no. 1, pp. 788–795, 2008.

[12] V. Tsourikov, L. Batchilo, I. Sovpel, and A. Korzun, "Semantic answering system and method. us patent," Patent 7 962 326B2, 2011.

[13] G. Cascini and P. Rissone, "Plastics design: Integrating triz creativity and semantic knowledge portals," *Journal of Engineering Design*, vol. 15, no. 4, pp. 405–424, 2004.

[14] C. Fellbaum, *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge: MA, 1998.

[15] J. R. Quinlan, "Induction of decision trees," *Mach. Learn*, pp. 81–106, 1986.

[16] K. Sparck-Jones, "A statistical interpretation of term specificity and its application in retrieval," *Jounal of Documentation*, pp. 11–21, 1972.

Chapter 4

Automatic Match between Specific Parameters and Generic Engineering Parameters

While TRIZ is increasingly developing both in research and education, new users always encounter difficulties in their first attempts to practice it. In such situation, Altshuller's original matrix (Section 2.1.5.1) often appears as an "easy-to-begin-with" tool. While not being representative of what TRIZ really is, it continues to seduce new users, teachers and trainers.

Several approaches to automate the use of the contradiction matrix have been proposed in literature, and research on the automatic match between specific parameters of the artefact being considered and Altshuller's generic engineering parameters remains a topic of interest. As a result, the matching of specific parameters and generic engineering parameters (Section 2.1.5.1) is chosen as the first step of the automatic ontology-based mechanism.

In this chapter, firstly, the existing problems of using the contradiction matrix are given, and the inventive principles ontology, which formalizes the process of using it, is presented. Then the proposed methods of matching specific parameters and generic engineering parameters based on semantic similarity and case-based reasoning are introduced in detail. Finally, the summary and conclusion about this chapter are given.

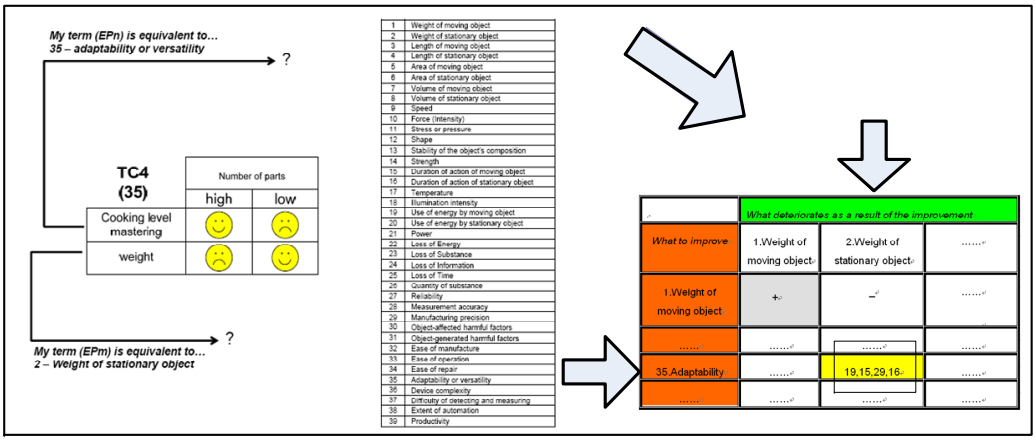


Figure 4.1: The process of using the contradiction matrix.

4.1 The Problem

The first TRIZ problem solving technique was a collection of inventive principles aimed at eliminating technical contradictions. To make the inventive principles applicable in a systematic way, the creator of TRIZ, Altshuller formulated thirty-nine generic engineering parameters (Appendix A), such as "the weight of a moving object" or "speed" which are generalizations of specific technical parameters [1].

As shown in Figure 4.1, TRIZ provides a tool called contradiction matrix (Appendix B), which presents all parameters in the form of a matrix. It consists in thirty-nine generic engineering parameters on the first line and column of it. Each cell of the matrix shows the inventive principles that can be used to solve that particular contradiction. Along the vertical axis of this matrix, the generic engineering parameters that have to be improved are specified. Along the horizontal axis, the parameters that deteriorate because of the improvement are specified. These parameters can be looked up along the vertical and horizontal axes and the matrix suggests up to four sorted principles that can be used to solve the contradiction.

In real-world problems, most of the times, contradictions are established in terms of parameters that are inherent to the artefact that is being developed, and there is a semantic gap to fill between those parameters and the generalized ones. An abstraction effort needs to be provided to choose the best generic engineering parameter, and in this way, be able to use the contradiction matrix.

Example: In the framework of an inventive design project proposed to a team of engineering students at INSA de Strasbourg, there was the study of the improvement of a barbecue grill. The students have retained the following contradiction to solve: if the number of parts in the wire mesh is high, the mastery of the beef doneness is satisfying but the weight of the grill is unsatisfying. On the other hand, if the number of parts is low, the mastery of the beef doneness is unsatisfying but the weight is satisfying. So there are two parameters: $SP1$, the weight; $SP2$, the mastery of the beef doneness.

$SP1$ is directly associated with the 2nd generic engineering parameter "weight of a stationary object"; but for $SP2$, the association with the 35th generic engineering parameter "adaptability or versatility" is not intuitive.

As explained above, the generic engineering parameters in the contradiction matrix are abstract and built independently of specific applications. In real applications, it is difficult for users to match specific parameters to the thirty-nine generic engineering parameters because it requires an extensive knowledge of different engineering domains.

4.2 The Inventive Principles Ontology

In order to facilitate the process of using the contradiction matrix and the inventive principles, the inventive principles ontology is built to formalize the concepts and their relationships.

The framework of the Inventive Principles Ontology is shown in Figure 4.2. *Feature* is defined to represent the parent concept of *PrimaryFeature* (or *Generic Engineering Parameter*) in the contradiction matrix, such as "power", and *AppliedFeature* (or *Specific Parameter*) in the specific application, such as "electrical energy".

The contradiction matrix consists of thirty-nine $PrimaryFeature$s on the vertical axis and horizontal axis and 39∗39 *Item*, which shows all the potential $InventivePrinciple$s used to solve a particular contradiction. Every two $PrimaryFeature$s correspond to an $Item$: one acts as positive $Feature$, the other as negative $Feature$. Each $Item$ can have i $InventivePrinciple$s ($i = 0 \cdots n$) while each *InventivePrinciple* can have j *SubInventivePrinciple*s ($j = 1 \cdots m$).

Each $SubInventivePrinciple$ also refers to two concepts: *PrimarySubIP* (1:1) - the initial description of the *SubInventivePrinciple*, such as "*IP38 (b)* - replace enriched air with pure oxygen"; and *AppliedSubIP* ($1 : i, i = 0 \cdots n$) - the detailed solution of the

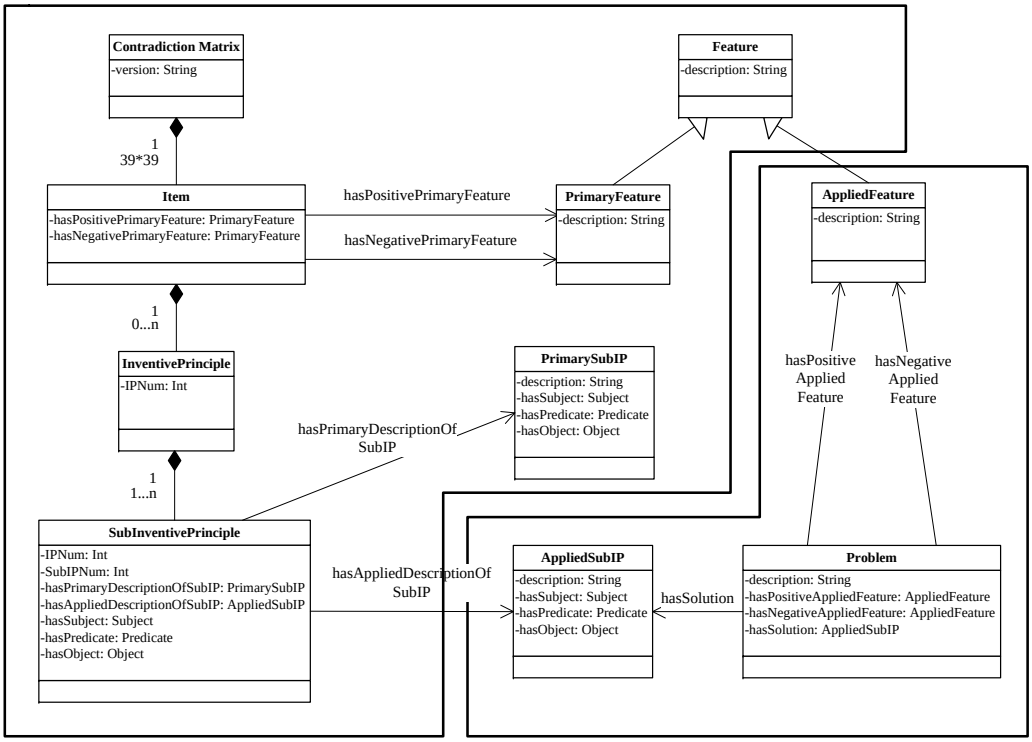


Figure 4.2: The inventive principles ontology.

application, such as "Treat wounds in a high pressure oxygen environment to kill anaerobic bacteria and aid healing".

The interpretations from *AppliedFeature* to *PrimaryFeature*, and from *PrimarySubIP* to *AppliedSubIP* need to be implemented manually; and this fact requires a large amount of TRIZ experience and knowledge covering a wide spectrum of domains. As a result, this research focuses on automating, as much as possible, this process in order to provide assistance to TRIZ users [2, 3].

Table 4.1 shows the definition of several classes and properties in the inventive standards ontology. *subClassOf* is the relation existing between super-classes and sub-classes, for example, *PrimaryFeature* and *AppliedFeature* are subclasses of *Feature*. Moreover, the properties can also be encoded in OWL syntax, in which the owl:DatatypeProperty means that the range of the property is data, such as, the range of *description* is string, and the owl:ObjectProperty means that the range is object, for example, *hasPositiveAppliedFeature* has the domain of *Problem* and the range of *AppliedFeature*.

Table 4.1: The OWL encoding classes and properties in the inventive principles ontology.

Classes	Properties
<owl:Class rdf:ID="Feature"> <rdfs:subClassOf> <owl:Class rdf:ID="owl:Thing"> </rdfs:subClassOf> </owl:Class>	<owl:DatatypeProperty rdf:ID="description"> <rdfs:domain rdf:resource="#Feature"/> <rdfs:range rdf:resource= "http://www.w3.org/2001/XMLSchema #string"/> </owl:DatatypeProperty>
<owl:Class rdf:ID="PrimaryFeature"> <rdfs:subClassOf> <owl:Class rdf:ID="Feature"/> </rdfs:subClassOf> </owl:Class>	<owl:ObjectProperty rdf:ID="hasPositiveAppliedFeature"> <rdfs:domain rdf:resource="#Problem"/> <rdfs:range rdf:resource= "#AppliedFeature"/> </owl:ObjectProperty>
<owl:Class rdf:ID="AppliedFeature"> <rdfs:subClassOf rdf:resource="#Feature"/> <owl:disjointWith> <owl:Class rdf:ID="PrimaryFeature"/> </owl:disjointWith> </owl:Class>	

4.3 The Basic Method based on Semantic Similarity

A method is explored to calculate the semantic distance between short texts and use it to fill the semantic gap between specific parameters and generic engineering parameters, and in this way, facilitate the process of using the contradiction matrix. The matching process is described in Section 3.3.2, and so in this section, only the preprocess and experiment for matching specific parameters and generic engineering parameters are considered.

4.3.1 Preprocess

During the matching between specific parameters and generic engineering parameters, the preprocess mainly includes two steps:

- *Subjective Words Deletion*: The specific parameters are usually obtained from the specific application, and so compared with abstract generic engineering parameters, there exist many useless subjective words (often noun, and are used to represent specific objects), such as the specific object "pipe" in the specific parameter "change the size of pipe". In order to match to the generic engineering parameters, these subjective words need to be eliminated.

- *Semantic Extension*: After the first step, the parameters to be matched generally consist of two or several words, based on which it is difficult to identify the groups of short texts with different semantic meaning. Consequently, it is necessary to extend these words with their similar words based on WordNet. For example, for the words "change size", the word "height" can be added according to the requirement of the specific application.

4.3.2 The Matching Process

The matching between specific parameters and generic engineering parameters can be divided into five phases:

1. Short Text Segmentation.
2. Sense Search.
3. Sense Similarity.
4. Word Similarity.
5. Short Text Similarity.

The detailed information for each phase is described in Section 3.3.2.

4.3.3 Experiments

To test the accuracy of the proposed semantic similarity measure between two short texts, the projects that were solved by engineering students at INSA de Strasbourg are used. In this framework, one hundred and fifty-six specific parameters of these projects and the thirty-nine Altshuller's generic engineering parameters have been matched.

In the following, $result1$ represents the set of matches done by the students and $result2$, the set of results coming from the automatic semantic similarity calculation.

For example, in the project about the improvement of a "pan", the students manually matched the specific parameter "several different foods to cook" to the generic engineering parameters "26. Quantity of substance/matter" and "35. Adaptability or versatility". With this method, this specific parameter is matched with "23. Loss of substance" and "26. Quantity of substance/matter" automatically.

The results have been identified as six different situations:

a. $result1 = result2$: $result1$ equals to $result2$ exactly.

b. $result1 \subset result2$: $result2$ contains $result1$.

c. $result1 \cap result2 \neq \emptyset$: There is an intersection between $result1$ and $result2$.

d. $result1 \supset result2$: $result1$ contains $result2$.

e. $result1 \cap result2 = \emptyset$: There is no intersection between $result1$ and $result2$.

f. $result2 = \emptyset$: $result2$ is empty.

The ratio of each kind of comparison results to the total number in our experiments is shown in Table 8.1.

Table 4.2: The ratio of each kind of comparison result to the total number.

Type of comparison	Ration
a. $result1 = result2$	19%
b. $result1 \subset result2$	21%
c. $result1 \cap result2 \neq \emptyset$	3%
d. $result1 \supset result2$	4%
e. $result1 \cap result2 = \emptyset$	45%
f. $result2 = \emptyset$	8%

Intuitively, the set a) should be as big as possible, while the sets e) and f) being as small as possible. Results of sets b), c) and f) might be acceptable to some extent, whose ratios might be decreased to improve the ratio of set a).

The results in Table 8.1 are not concluding: set a) represents only 19% of the whole, while sets e) and f) represent 45% and 8% respectively. It means that for 53% projects,

the results of the automatic approach do not correspond to the matches done by the students. Only for 19% projects, all the exact matches are obtained. For the rest, more or less correct solutions are obtained.

The analysis of this situation leads to three reasons:

- The inaccurate description of the projects: In these experiments, it is assumed that all elements in result1 are correct; that is, that the students have done the matching correctly, in the same way as a real TRIZ expert would have done. In fact, there are differences among the views of different people about the same project, and not all the students describe the projects accurately. As a result, there are some obvious mistakes in the matches retained by the students. This is also the main reason why the size of set e) is so big.

- The restriction of the dictionary: WordNet is used as our dictionary. Even though compared with other dictionaries, WordNet has the advantage of the size of its terms and the variety of its semantic relationships, it only depicts concepts and their relationships in general language, to be opposed to technical or specialized language. For example, the general concepts "water", "liquid" and their hyponyms and hypernyms can be found in WordNet while a specialized notion, such as the "Reynold's number", does not appear in the dictionary. That is the reason for the appearance of the set of type f).

- Problems with our method: The method for calculating semantic similarity between short texts is explored. This calculation is done without considering contexts. By "context", it means the area or field of the project. For instance, in the mechanical case, the specific parameter "stress" should be matched with "10. Force (Intensity)", while in other areas, the same parameter "stress" is usually matched with "11. Stress or pressure". Without considering contexts, our method does not have a good performance.

To cope with these disadvantages, an improvement of this method is proposed in next section.

4.4 The Improved Method: Semantic Similarity + Case-based Reasoning

Case-based reasoning (CBR) [4] is a well-known methodology in Artificial Intelligence (AI). The main idea of CBR is to adapt solutions that were used to solve old problems and use them for solving new problems or cases, that is, people reuse past problem-solving experiences to deal with a new case.

When CBR is used to solve a new problem, firstly the new problem is standardized according to the case representation. Then similar cases are retrieved from the case library, and the solution of the most similar problem is suggested for the new problem. If necessary, this suggested solution is revised according to previous experience, domain knowledge and the actual situation of the new problem. After obtaining the confirmed solution, the new case, composed of the new problem and its confirmed solution, is stored into the case library.

The cycle of CBR comprises five activities:

- Characterize the new problem and assign an index.
- Retrieve the most similar case(s) to the new problem from the case library.
- Reuse the case(s) to attempt to solve the new problem.
- Revise or adapt the suggested solution to satisfy the new problem if necessary.
- Retain the new problem and its confirmed solution as a new case.

The cyclic process of CBR is depicted in Figure 4.3.

According to the above five activities, a specific case-based reasoning framework is constructed to improve the method presented in Section 4.3. Specifically, on the one hand, the semantic similarity between short texts is calculated to return the matched generic engineering parameters. On the other hand, the solution to the most similar previous problem is retrieved and returned to users as a suggested solution based on case-based reasoning simultaneously. The whole flowchart of the improved method is described in Figure 4.4.

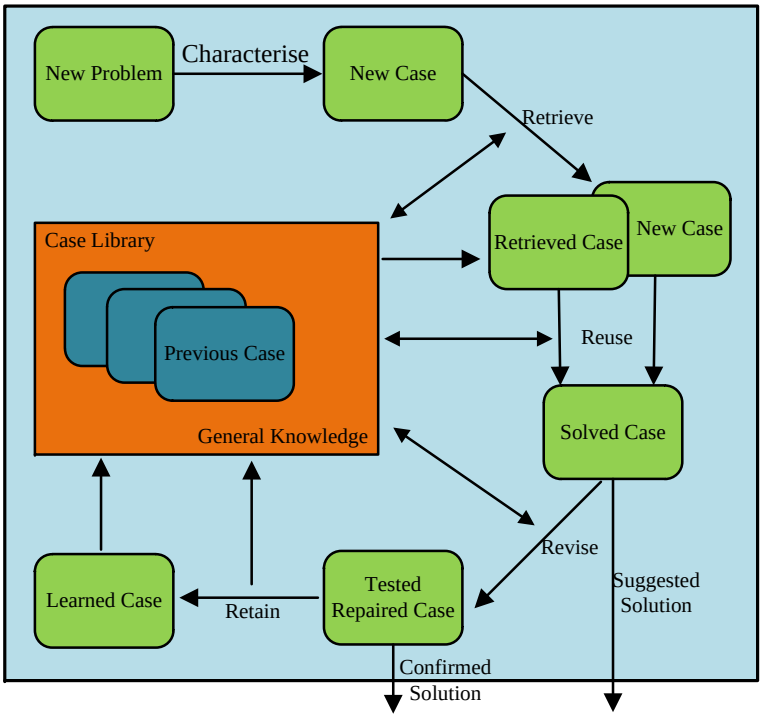


Figure 4.3: The CBR cycle.

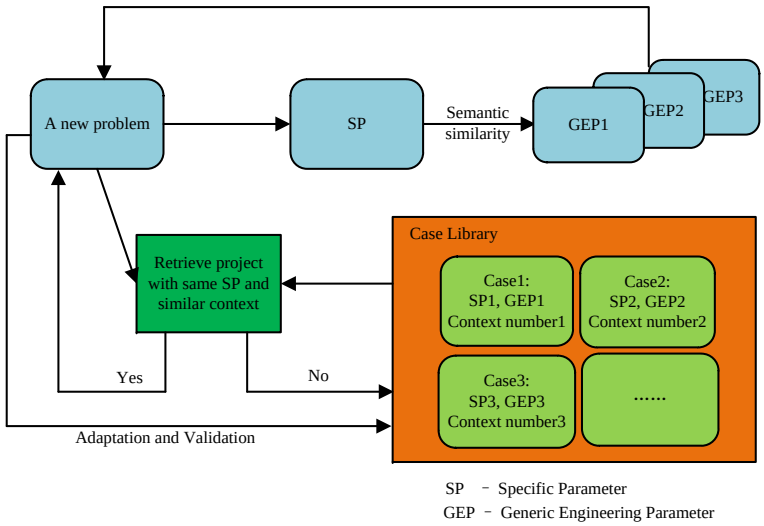


Figure 4.4: The whole flowchart of the improved method.

Case Name
-Context Number -Specific Parameter->Generic Engineering Parameter

Figure 4.5: The case representation.

4.4.1 Case Representation

Case representation is an important aspect in designing efficient CBR systems. In this application, a case is represented with the following three attributes:

- The context number.
- The match from the specific parameter (that is, one of the parameters of the artefact involved in the contradictions) to a generic engineering parameter (these matches are chosen and reused as the previous experience because they have been used under the same specific context), as shown in Figure 4.5.

In order to make a comprehensive classification of contexts, the International Patent Classification (IPC)[1], established by the Strasbourg Agreement 1971, that provides a hierarchical system of language independent symbols for the classification of patents and utility models according to the different areas of technology to which they pertain, is used. All the contexts are divided into eight groups:

A. Human Necessities

B. Performing Operations; Transporting

C. Chemistry; Metallurgy

D. Textiles; Paper

E. Fixed Constructions

F. Mechanical Engineering; Lighting; Heating; Weapons; Blasting

G. Physics

[1]http://www.wipo.int/classifications/ipc/en/.

H. Electricity

Each group is composed of several classes, and these classes are made up of numerous subclasses, organized in a hierarchy of four levels. In this method, the first three levels are chosen as the classifications of contexts. For example, there is, "H. Electricity", including all the materials and phenomena concerning electricity, and its corresponding Subclasses, such as "H01. Basic Electric Elements" and this subclass also has several Nested-Subclasses, for example, 'H01C. Resistors'.

When the user builds the representation of a case, he needs to choose a context index number for it, but sometimes, he may not find an exact match. If it is the case, the user should look for terms of approximately the same meaning, for terms of either broader or narrower scope, or for terms that represent a different approach to the subject; i.e., the essential function or effect of the device or the use or application to which the device is put.

All past cases are represented according to the above framework, and then index them by the unique identifier (the context number) to facilitate the retrieval.

It is assumed that a specific parameter with similar context can correspond to several generic engineering parameters, that is, for a new problem, several different suggested results might be obtained based on case-based reasoning.

4.4.2 Case Retrieval

It includes the retrieval of past similar cases and the selection of the best case. Firstly, a context number is chosen for the new case. Then we query the case library by comparing the specific parameter of the new case and the previous case. For cases whose specific parameter is the same as the new case, we next compare their context numbers, and decide whether they have a similar context. Finally, the solutions of cases with the similar context, that is their matches between the specific parameter and the generic engineering parameter, are returned to users as suggested results.

The process of searching the most similar context plays an important role in the improved method. Firstly, we construct a group-class-subclass tree to represent the classification of contexts with different granularities, larger for groups and smaller for subclasses. For a given case, we choose the index of the subclass - the minimal granularity to represent it. Then we apply a bottom-up approach of the retrieval of the group-class-

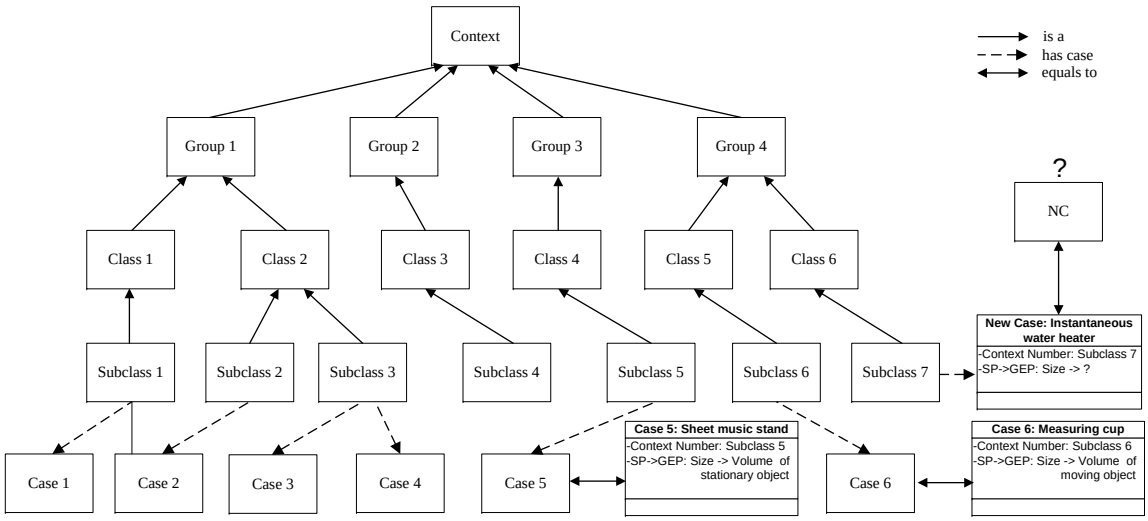


Figure 4.6: The bottom-up method of the retrieval of the group-class-subclass tree.

subclass tree, as shown in Figure 4.6[1] and search for the similar context according to the following steps:

1. Retrieve cases belonging to the same subclass, if they exist, go to step 4, else go to step 2.

2. Use Depth First Search (DFS) [5] to retrieve the cases belonging to the subclasses who have the same parent - classes with the given index of the subclass of the context, if they exist, go to step 4, else go to step 3.

3. Use DFS to retrieve cases belonging to the classes who have the same parent - group with the given index of the subclass of the context, if they exist, go to step 4, else return "no case with similar context".

4. Return the retrieved cases.

According to the above four steps, we retrieve at least the first minimal level of the group-class-subclass tree. Generally, this program will be finished as soon as the first similar case is found because the bottom-up retrieval along the group-class-subclass tree ensures that the first obtained case is the most similar case.

[1]Although the hierarchy has numerous levels, we have chosen here, for the sake of simplicity, to represent only three of them.

Let us consider a new case "NC" in Figure 4.6. If the user chooses "Subclass 3" as the context, step 1 of the algorithm just returns "Case 3" and "Case 4" as the most similar cases. On the contrary, if the user chooses "Subclass 4" as the context, steps 1 and 2 do not find any similar cases; our bottom-up search does not return any similar case. Another situation would be if the user chooses "Subclass 7" as the context. In this situation, we need to generalize to the upper levels ("Class 6" and "Group 4") to retrieve "Case 6" as the most similar one to the NC.

We take the matching of the specific parameter "size" in the new case "instantaneous water heater" as an example. Suppose that the context of this case is "Subclass 7", and according to what is stated above, we can obtain a similar case "Case 6" with the matching between "size" and "volume of moving object" because it belongs to "Group 4" as the new case does. The matching between "size" and "volume of stationary object" in "Case 5" cannot be returned because it has a different parent even in the group-level.

4.4.3 Case Adaptation and Verification

The goal of this step is to analyze items requesting adaptation and implement adaptation. The obtained similar matching between specific parameters and generic engineering parameters is considered as suggested result, which will be returned to the TRIZ user together with the semantic matching results.

The goal of this part is to retain the new case. As the new problem has a new solution, it can be described according to the case representation and added into the case library. In addition, with the rapid increase of the number of cases, the case with repeated matches under a similar context should be deleted because it may result in redundant cases. The purpose is to make sure that the size of the case library would not increase continuously to affect the retrieval speed and make each problem more correct with higher accuracy.

4.4.4 Case Library

Each case keeps track, in fact, of a "translation" such as specific parameter -> generic engineering parameter, valid in a certain context. We would like to have this context as general as possible. In other words, the idea is to find the more general context where the two short texts (that represent the specific and generic engineering parameters) have the same sense. This is why we launch periodically this maintenance task of the case library.

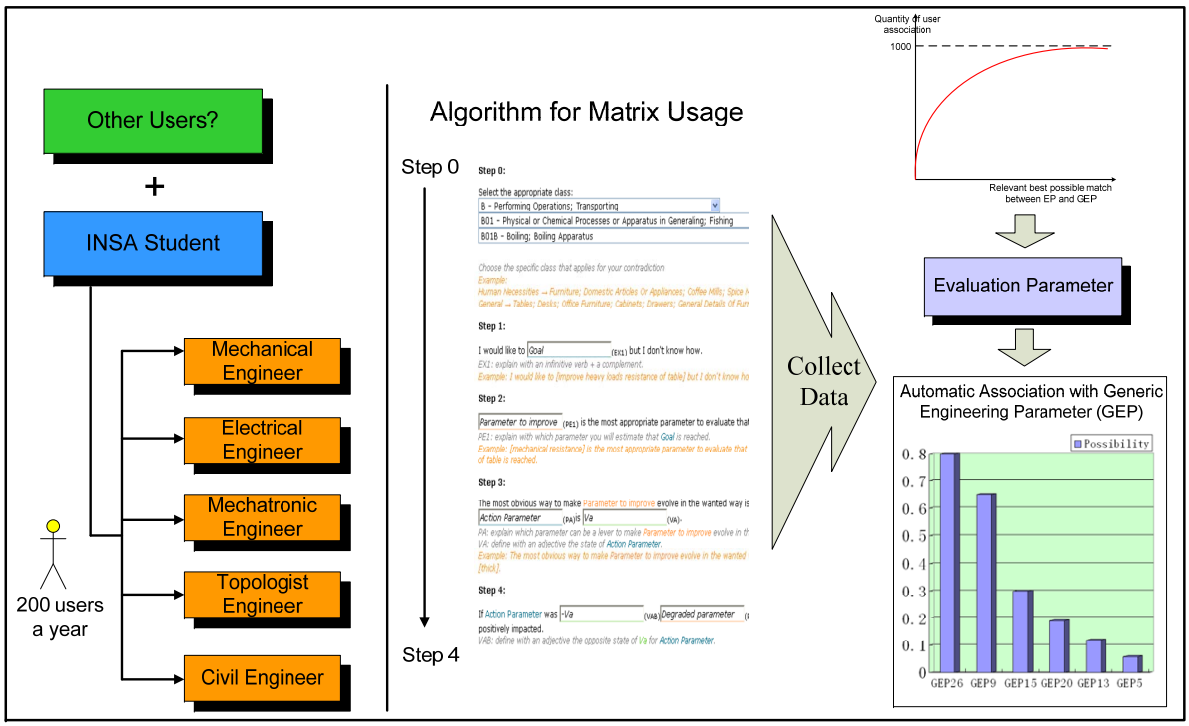


Figure 4.7: The Evaluation of the improved method.

If all the leaves of the sub-tree of contexts have cases with the same translation associated to them, we can assume that those translations are valid in the more general common context.

For example, let us consider the following leaves in the taxonomy of Figure 4.6; "Subclass 1", "Subclass 2", "Subclass 3". These three contexts have cases associated with them. Let us suppose that "Case 1", "Case 2" and "Case 3" have the same translation "SP1 -> GEP1". In this case, we can propose the user to validate these three translations in the context "Group 1" (the more general common context for the leaves we have considered).

4.4.5 Evaluation

As stated above, the performance of the improved method greatly depends on the experience and results of the previous cases, that is, a large number of cases, in which the inventive problems are solved by using the contradiction matrix and inventive principles, are necessary.

Consequently, in order to construct the case library, the prototype of our methods (shown in Chapter 7), will be firstly integrated into the software developed by the laboratory LGECO (that will be discussed in Section 7.1). On one hand, this software will provide valuable assistance to the users all over the world trained in the methodology and working on a design project in different domains, and on the other hand, this user experience makes it possible to accumulate a large amount of users' cases for our case library. As shown in Figure 4.7, with the increasing number of users' cases and feedback, the performance of our methods will be improved significantly.

4.5 Summary

In this chapter, in order to facilitate the use of Altshuller's contradiction matrix in an inventive design study, methods of matching specific parameters and generic engineering parameter based on semantic similarity and case-based reasoning are introduced in detail. The inventive principles ontology is firstly built to formalize the process of using the contradiction matrix, and then according to the characteristics of our specific application (filling the semantic gap between the specific parameters and the generic engineering parameters), the method of calculating semantic similarity presented in Section 3.3.2 is used.

In order to improve the performance of this method, the results from previous projects are returned to users as suggested solutions according to a case-based reasoning approach. As we can observe from our first attempt to test the developed methodology, some relevant potential associations between a specific term and one or several of the thirty-nine generic engineering parameters are proposed to users. The improved method also benefits from previous experiences and therefore offers a constant improvement due to CBR techniques.

However, the resolution of inventive problems only with forty inventive principles can not always provide comprehensive and efficient recommendations, and it is necessary to apply other TRIZ knowledge sources, such as seventy-six inventive standards. Therefore, next chapter introduces a method of searching heuristic abstract solution from different knowledge sources based on semantic similarity and ontology reasoning.

Bibliography

[1] G. Altshuller, *And Suddenly the Inventor Appeared, translated by Lev S.* Technical Innovation Center Inc., Worcester, MA, 1994.

[2] W. Yan, C. Zanni-Merk, and F. Rousselot, "An application of semantic distance between short texts to inventive design," *In: Proceedings of the International Conference on Knowledge Engineering and Ontology Development, KEOD2011*, pp. 261–266, 2011.

[3] W. Yan, C. Zanni-Merk, F. Rousselot, and D. Cavallucci, "A method of facilitating inventive design based on semantic similarity and case-based reasoning," *In: Proceedings of the TRIZ Future 2011*, 2011.

[4] A. Agnar and P. Enric, "Case-based reasoning: Foundational issues, methodological variations, and system approaches," *Artificial Intelligence Communications*, vol. 1, pp. 39–52, 1994.

[5] A. Aho, J. Hopcroft, and J. Ullman, *Data Structures and Algorithms.* Addison-Wesley, 1982.

Chapter 5

Ontology-based Search for Heuristic Abstract Solutions

During the development of TRIZ, different knowledge sources were established in order to solve different types of inventive problems, such as the forty inventive principles for eliminating the technical contradictions. These knowledge sources, that have different levels of abstraction, are all built independently of the specific application field, making their use difficult, as an extensive knowledge about different engineering domains is required.

In order to facilitate the use of the TRIZ knowledge sources, this chapter presents a new inventive problem solving approach based on ontologies. In this approach, the TRIZ users start solving an inventive problem with the TRIZ knowledge source of their choice to obtain an abstract solution. According to the selected items of that first knowledge source, the similar items of other knowledge sources are obtained based on the semantic similarity calculated in advance. Considering that all the TRIZ knowledge sources are described as short-texts, the missing links among the TRIZ knowledge sources are defined based on short-text semantic similarity. At the same time, an ontology reasoning mechanism deployed on Protégé and Jess, is used to provide heuristic solutions dynamically for TRIZ users.

5.1 The Problem: Missing Links among the TRIZ Knowledge Sources

In order to present objects, ideas, situations and relationships in the inventive problem solving process, as stated in [1], the TRIZ experts build models for specific applications from their own point of view. The knowledge sources for resolution are situated at different levels of abstraction and at different levels of "closeness to reality". In fact, inventive principles, for example, even if they seem to refer to concrete reality (for instance, "inert atmosphere") are conceptually more abstract than inventive standards, which refer to concrete substances or fields. The imagination effort by the user to apply an inventive principle is much more important than the one to be done if he wants to use an inventive standard.

More than that, even in the same knowledge source the description may be different or incompatible, as stated in [2]. For example, even though both *Inventive Standard 1.1.5*-"Transition to SFM by using external environment with additives" and *Inventive Standard 1.1.3*-"Transition to external complex SFM" indicate that the external substances or fields can be introduced to improve the existing model, their descriptions are different.

In order to formalize the set of knowledge bases of TRIZ and make it coherent and complete, [3] analyzes the similarity among inventive principles based on the text descriptions of examples using them. Six out of forty principles are defined as "Obscure Principles", which are hard to be analyzed by automatic classification systems. In addition, two kinds of similarities between principles are defined in this paper: text similarity and meaning similarity.

In this research, the approaches of measuring short-text semantic similarity are used to compare, analyze, and match the items in the different knowledge sources ontologies. Figure 5.1 shows an example of correspondences of items in the knowledge sources presented above. The higher the value of similarity, the more similar the two items. For example, *Inventive Standard 5.3.1*-"Changing of phase state" is more similar to *Inventive Standard 1.1.5*-"Transition to SFM by using external environment with additives" (Sim: 0.75) than *Inventive Principle 39*-"Inert atmosphere" (Sim: 0.15).

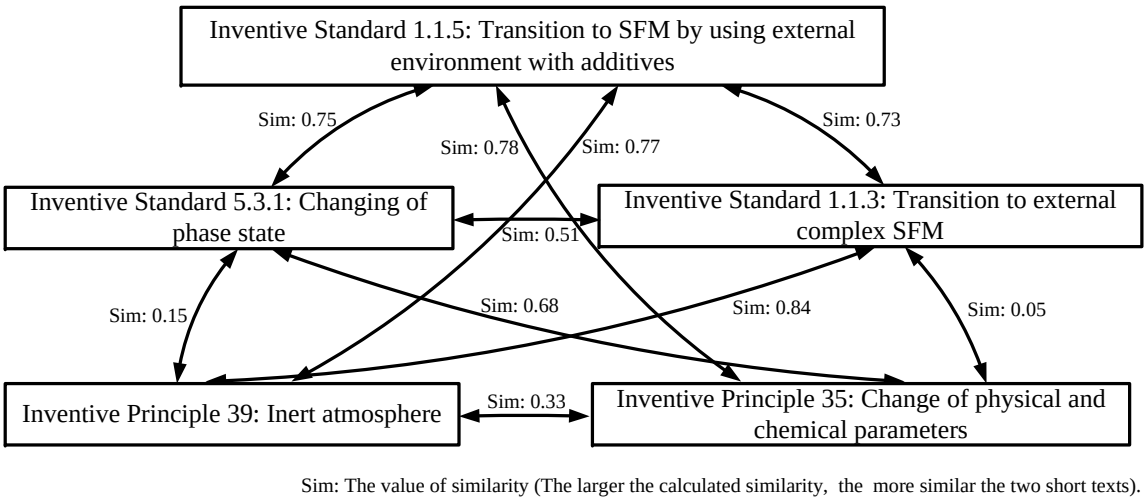


Figure 5.1: An example of correspondences of items of different knowledge sources.

5.2 The Framework

As shown in Figure 5.2, the proposed method is made up of a preprocess and a main process.

The preprocess includes two steps:

- Step $p1$: Use semantic relatedness methods to find the missing links among the TRIZ knowledge sources, and store them.

- Step $p2$: Establish the TRIZ knowledge sources ontologies and design ontology reasoning rules. Ontologies are used to formalize the main concepts in the TRIZ knowledge sources, and Semantic Web Rule Language (SWRL) [4] is used to describe the reasoning rules.

After the preprocess, the main process consists of three steps:

- Step 1: Select items from one of the TRIZ knowledge sources for an abstract solution, for example, using inventive principles to eliminate a technical contradiction.

- Step 2: For the selected items in Step 1, search the similar items of other knowledge sources on the basis of the similarities stored in the database.

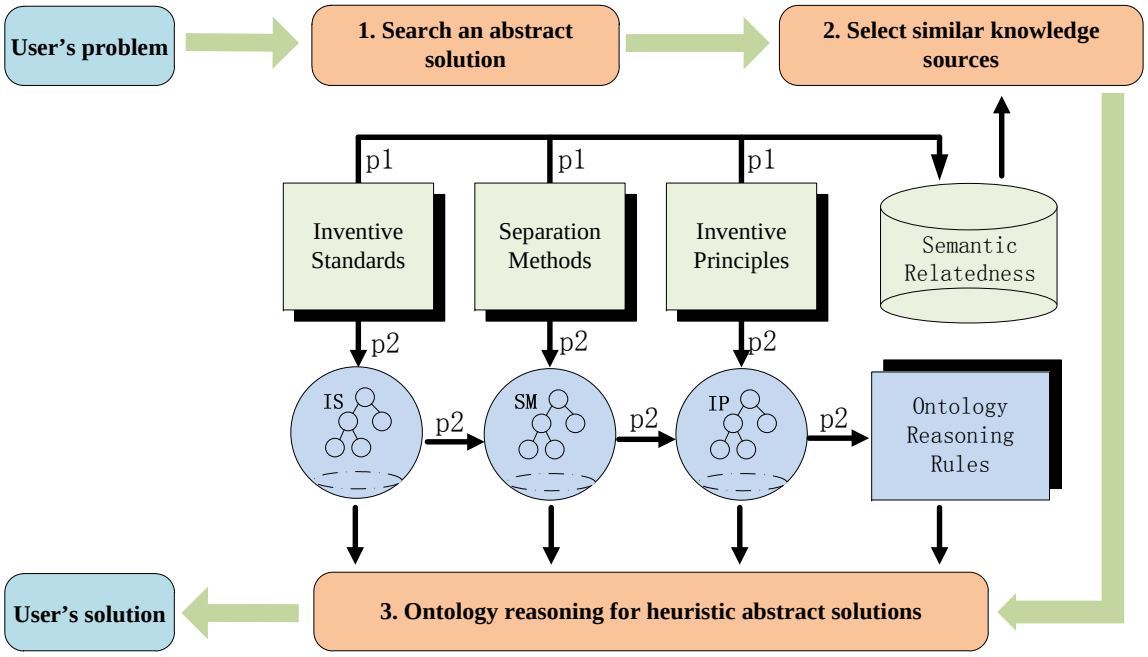


Figure 5.2: The framework of the ontology-based approach.

- Step 3: According to the TRIZ knowledge sources ontologies, choose the appropriate reasoning rules from the rule set, and implement ontology reasoning to provide the heuristic abstract solutions.

5.3 Semantic Relatedness Methods for Defining the Missing Links among the TRIZ Knowledge Sources

In this section, the process of using the method proposed in Section 3.3.3 to define the missing links among the TRIZ knowledge sources is introduced. Taking into account that the detailed matching process has been described in Section 3.3.3, only the preprocess and the obtained semantic similarity among the TRIZ knowledge sources are considered.

5.3.1 Preprocess

Before the semantic matching, data preprocess is carried out to reduce redundancy and minimize imprecision. Different kinds of knowledge sources need different preprocesses:

Inventive Principles: Compared with the distinct principles, additional information, such as the similar words, need to be added to the obscure principles to facilitate their matches with inventive standards.

Inventive Standards: As analyzed in Section 2.1.5.2, the inventive standards of C-class and D-class are identified as new inventive principles and patterns of evolution respectively, so only the seventy-one inventive standards of A-class and B-class (Appendix D.2) are matched with the given inventive principles according to the semantic similarity methods proposed in Section 3.3.3, and correspondingly, in Section 5.4, the reasoning rules only about these inventive standards are built.

Separation Methods: The eleven separation methods, separation in space, time, upon condition, and between parts and the whole, are described abstractly, and the output of using these methods might not yield to a design concept or physical function. For example, *Separation Method 1*: "Separation of conflicting properties in space" suggests the users to separate the conflicting properties in space dimension, but which properties to be divided are not indicated. As a result, the complementary information about using these methods should be added to enrich the separation methods, such as the users' cases.

5.3.2 The Matching Process

Taking into account the characteristics of the TRIZ knowledge sources to be matched, such as, high appearance of words with similar meaning, the improved method with word weight is used to calculate the semantic similarity among the TRIZ knowledge sources. The detailed process is described in Section 3.3.3.

5.3.3 The Semantic Similarity Among the TRIZ Knowledge Sources

Considering that forty inventive principles are the most-used TRIZ knowledge source, the match starts with inventive principles and the semantic similarity is divided into two kinds: the similarity between inventive principles and inventive standards, and the similarity between inventive principles and separation methods.

Semantic Similarity Between Inventive Principles and Inventive Standards There are three possibilities for the obtained results, that is, A-class, B-class and A and B-class. For a given inventive principle, we firstly determine if it corresponds to the inventive standards

Table 5.1: The similar inventive standards for *InventivePrinciple* 1, *Inventive Principle 2* and *InventivePrinciple* 36

Inventive Principles	Similar Inventive Standards	
	A-class	B-class
InventivePrinciple 1		IS59
Inventive Principle 2	IS36, IS44, IS41, IS62	IS59
InventivePrinciple 36	IS29 IS64 IS65 IS66 IS67	

of B-class, if yes, return the corresponding ones; or else, select five inventive standards of A-class most similar with the given inventive principle based on the proposed method. In addition, some of the obtained five A-class similar inventive standards may belong to B-class because the semantic similarities are calculated between the forty inventive principles and all the inventive standards of A-class and B-class.

For example, Table 5.1 shows the similar inventive standards for *InventivePrinciple* 1, *InventivePrinciple* 2 and *InventivePrinciple* 36:

- *InventivePrinciple* 1: Segmentation
  (a) Divide an object into independent parts.
  (b) Divide an object into parts so that some its part can be easily taken away.
  (c) Increase the degree of object segmentation.
- *InventivePrinciple* 2: Taking Away
  (a) Take away an interfering part of the object. If some property of the object interferes, find what part of the object is a carrier of the property and separate it from the object.
- *InventivePrinciple* 36: Phase Transitions
  (a) Use physical phenomena accompanying phase transitions: change of volume, emission or absorption of heat, etc.

InventivePrinciple 1 is matched with *Inventive Standard 59* (B-class), *Inventive Principle 36* with *Inventive Standard 29, 64, 65, 66, 67* (A-class), and *Inventive Principle 2* with *Inventive Standard 36, 44, 41, 62* (A-class) and *Inventive Standard 59* (B-class).

- *InventiveStandard* 59: 5.1.2. Dividing of product.
- *InventiveStandard* 29: 2.4.7. Using physical effects.
- *InventiveStandard* 64: 5.3.2. "Dual" phase state of substance (Phase transition 2).
- *InventiveStandard* 65: 5.3.3. Using phenomena accompanying a phase transition (Phase transition 3).
- *InventiveStandard* 66: 5.3.4. Transition to dual-phase state (Phase transition 4).
- *InventiveStandard* 67: 5.3.5. Using interaction between phases of the system.
- *InventiveStandard* 36: 4.1.2. Using of copies.
- *InventiveStandard* 44: 4.3.3. Using resonance oscillation of attached object.
- *InventiveStandard* 41: 4.2.4. Transition to measurement SFM by using external environment properties.
- *InventiveStandard* 62: 5.2.1. Using present fields (pluralistically).

Semantic Similarity Between Inventive Principles and Separation Methods Analogously, two most similar separation methods are obtained for each inventive principle. For example, *InventivePrinciple* 2 is matched with:

- *SeparationMethod* 7: Substitution of the phase state of a system's part or external environment.
- *SeparationMethod* 1: Separation of conflicting properties in space.

As stated in Section 5.3.1, inventive principles and separation methods are established in different levels of abstraction, and the accuracy of the obtained similarity depends partly on the amount of the information provided by the extensive users' cases. Consequently, only the matches among inventive principles and inventive standards are considered in the prototype presented in Chapter 7.

The semantic similarity measures among the TRIZ knowledge sources are given in Appendix F.

5.4 Ontology Reasoning for Searching Heuristic Abstract Solutions

According the semantic similarity among the TRIZ knowledge sources, when the user works with a knowledge source, the similar items of other knowledge sources can be obtained. Furthermore, from these similar items, the heuristic abstract solutions, which provide the complementary information for solving the problem, can be generated based on ontology inference. Considering that inventive standards, the most difficult TRIZ knowledge source, can provide most efficient recommendations, the process of using inventive standards to generate heuristic abstract solutions based on ontology inference is considered in this research.

This section describes this process in detail. Firstly, the TRIZ knowledge sources ontologies are presented, including the inventive principles ontology (Section 4.2) and the inventive standards ontology. Then reasoning rules are set to describe the process of searching heuristic abstract solutions. Finally, the Jess rule engine[5] is used to execute the ontology inference.

5.4.1 The Inventive Standards Ontology

As shown in Section 1.1, the specific process of using Su-Field models to solve an inventive problem includes: building a Problem Model, mapping to a Generic Problem Model, finding a Generic Solution Model based on the corresponding inventive standard, and finally establishing and instantiating a Solution Model.

As shown in Figure 5.3, *Problem* is used to represent the concept of Problem Model. For a specific case, its *Problem* corresponds to a *Generic_Problem_Model* and a *Generic_Solution_Model* through the two properties *correspondsTo_GPM* and *correspondsTo_GSM* respectively. *Generic_Problem_Model* has one or two *Substance*s and one *Field*, depicted by *hasSubstance1_GPM*, *hasSubstance2_GPM* and *hasField_GPM*, while *Generic_Solution_Model* uses *hasSubstance1_GSM*, *hasSubstance2_GSM* and *hasField_GSM* to describe these relations.

The transformation from *Generic_Problem_Model* to *Generic_Solution_Model* is implemented through the use of the appropriate *InventiveStandard*, which is described by *chooses_IS*.

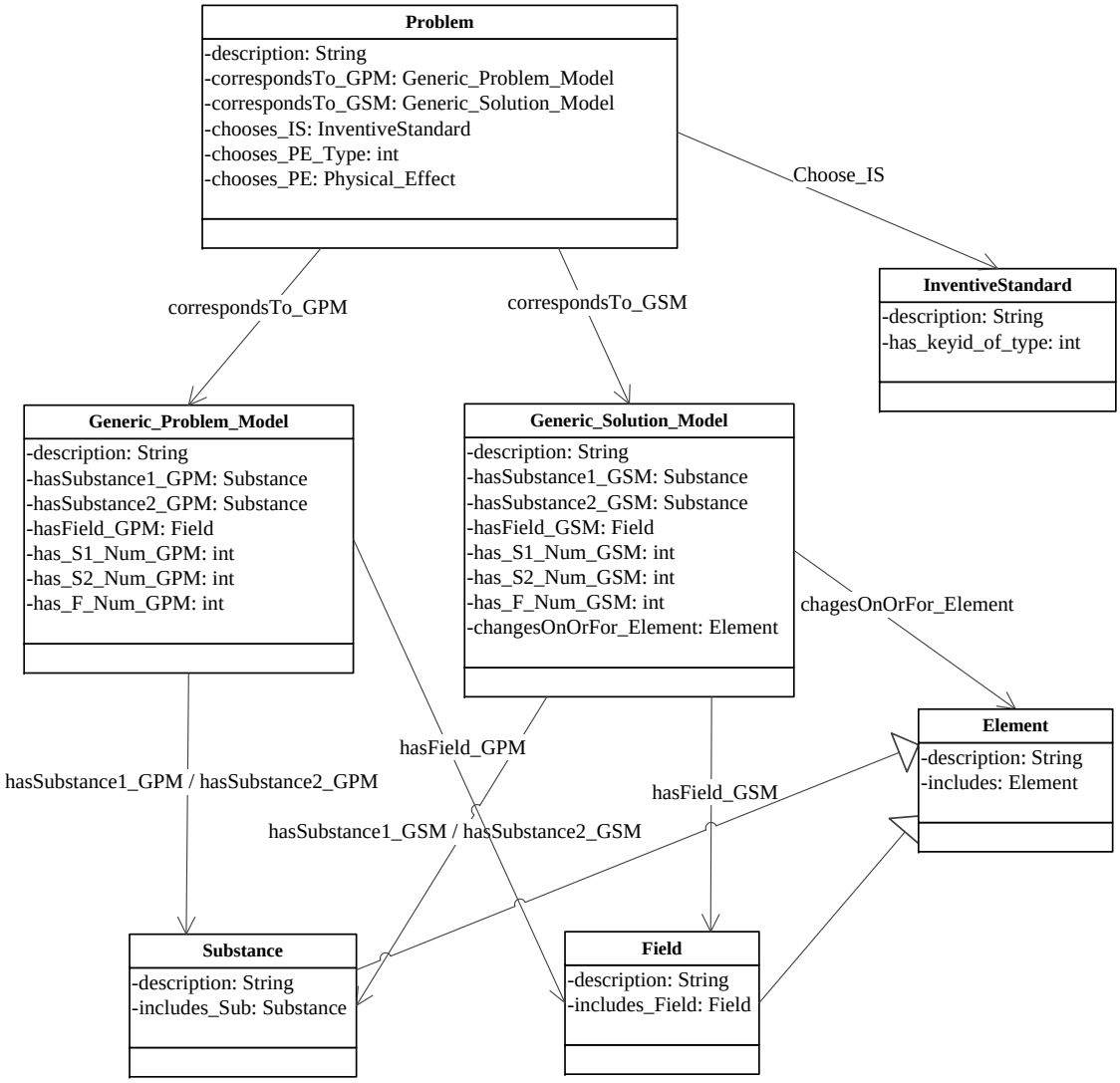


Figure 5.3: The inventive standards ontology.

In order to detect the change of states before and after using *InventiveStandard*, three properties *has_S1_Num_GPM*, *has_S2_Num_GPM* and *has_F_Num_GPM* are defined to obtain the number of *Substance*s and *Field* for *Generic_Problem_Model*, while three properties *has_S1_Num_GSM*, *has_S2_Num_GSM* and *has_F_Num_GSM* for *Generic_Solution_Model*.

Chooses_PE is defined to represent the relation between *Problem* and the chosen *Physical_Effect* (see next chapter). The properties *has_keyid_of_type* and *chooses_PE_Type*

Table 5.2: The OWL encoding classes and properties in the inventive standards ontology.

Classes and Properties
<owl:Class rdf:ID="Element">
<rdfs:subClassOf>
<owl:Class rdf:ID="owl:Thing">
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Substance">
<rdfs:subClassOf>
<owl:Class rdf:ID="Element"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Field">
<rdfs:subClassOf rdf:resource="#Element"/>
<owl:disjointWith>
<owl:Class rdf:ID="Substance"/>
</owl:disjointWith>
</owl:Class>
<owl:DatatypeProperty rdf:ID="description">
<rdfs:domain rdf:resource="#Element"/>
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:ObjectProperty rdf:ID="includes">
<rdfs:domain rdf:resource="#Element"/>
<rdfs:range rdf:resource="#Element"/>
</owl:ObjectProperty>

are used to mark the type of the chosen inventive standard and the physical effects to be chosen respectively.

The property *changesOnOrFor_Element* is used to record the concrete object, including *Substance* and *Field*, chosen by the user in the specific case.

Table 5.2 shows several OWL encoding classes and properties in the inventive standards ontology. The relations between super-classes and sub-classes are defined by *SubClassOf*, such as *Substance* and *Field* are two subclasses of *Element*. The properties are used to represent the relations between one object and another object or data, for example, *description* maps *Element* with its description in string type, and *correspondsTo_GPM*

matches *Problem* and its corresponding *Generic_Problem_Model*.

5.4.2 Ontology Reasoning Rules

As stated in Section 5.3, only five inventive standards of A-class and B-class are obtained as the final similar results, based on which the heuristic abstract solutions will be generated in Section 5.4.3.

Correspondingly, the reasoning rules are built according to these two classes:

- The rules for A-class inventive standards: The forty inventive standards are condensed and generalized into seven generalized standard solutions, and each kind of standards have similar modifications or evolutions on the Su-Field model. Consequently, it is possible to use the reasoning rules for seven generalized standard solutions to represent the modification of all the forty inventive standards.

- The rules for B-class inventive standards: The thirty-one inventive standards are identified as the implementation of existing inventive principles, and each inventive standard yields to a distinctive modification. As a result, for each inventive standard, the reasoning rules to describe its modification are set.

According to the analysis of these two kinds of inventive standards, the number of substances and fields can be used to identify the modification of inventive standards. So the properties *has_S1_Num_GPM*, *has_S2_Num_GPM*, *has_F_Num_GPM* are used to indicate the number of substances and field before the modification while *has_S1_Num_GSM*, *has_S2_Num_GSM* and *has_F_Num_GSM* are used to indicate the number of substances and field after the modification. The regular patterns about the number of substance and field are described as following:

- For *Substance 1*:
  - *has_S1_Num_GPM*$\neq$0:
    - *has_S1_Num_GSM*=1: The additive *Substance 1* is used instead of the *Substance 1* before the modification.
    - *has_S1_Num_GSM*$\neq$1: The additive *Substance 1* is added to the *Substance 1* before the modification.

  - *has_S1_Num_GPM*=0:
    - The additive *Substance 1* is added to complete the incomplete problem model.

- For *Substance 2*:
  - *has_S2_Num_GPM*≠0:
    - *has_S2_Num_GSM*=1: The additive *Substance 2* is used instead of the *Substance 2* before the modification.
    - *has_S2_Num_GSM*≠1: The additive *Substance 2* is added to the *Substance 2* before the modification.
  - *has_S2_Num_GPM*=0:
    - The additive *Substance 2* is added to complete the incomplete problem model.

- For *Field*:
  - *has_F_Num_GPM*≠0:
    - *has_F_Num_GSM*=1: The additive *Field* is used instead of the *Field* before the modification.
    - *has_F_Num_GSM*≠1: The additive *Field* is added to the *Field* before the modification.
  - *has_F_Num_GPM*=0:
    - The additive *Field* is added to complete the incomplete problem model.

Taking *General Solution 1*- “Complete an incomplete Su-Field model"as an example, its reasoning rules need to describe the corresponding modification, that is, firstly the missing components are detected, and then according to the missing component, several additive elements, substances or filed, are added into the Su-Field model after the ontology inference.

As shown in Table 5.3, there are three SWRL rules for *General Solution 1*. *Rule 1* and *Rule 2* describe the modification when the substances are missing (*has_S1_Num_GPM*=0 or *has_S2_Num_GPM*=0), based on which an additive substance is added to the Su-Field model after the ontology inference. Similarly, *Rule 3* expresses the modification

Table 5.3: The SWRL rules and explanation for *General Solution 1.*

Name	SWRL rules and explanation
Rule 1	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 1) \wedge Generic\_Problem\_Model(?z) \wedge$ $correspondsTo\_GPM(?x, ?z) \wedge Generic\_Solution\_Model(?a) \wedge$ $correspondsTo\_GSM(?x, ?a) \wedge Field(?b) \wedge Substance(?c) \wedge$ $hasField\_GPM(?z, ?b) \wedge hasSubstance2\_GPM(?z, ?c) \wedge$ $has\_S1\_Num\_GPM(?z, 0) \wedge has\_S2\_Num\_GPM(?z, 1) \wedge$ $has\_F\_Num\_GPM(?z, 1) \rightarrow hasField\_GSM(?a, ?b) \wedge$ $hasSubstance1\_GSM(?a, added\_S1\_x2) \wedge$ $hasSubstance2\_GSM(?a, ?c) \wedge has\_S1\_Num\_GSM(?a, 1) \wedge$ $has\_S2\_Num\_GSM(?a, 1) \wedge has\_F\_Num\_GSM(?a, 1)$ If the component Substance1 is missing, add one to complete the Su-Field Model.
Rule 2	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 1) \wedge Generic\_Problem\_Model(?z) \wedge$ $correspondsTo\_GPM(?x, ?z) \wedge Generic\_Solution\_Model(?a) \wedge$ $correspondsTo\_GSM(?x, ?a) \wedge Field(?b) \wedge Substance(?c) \wedge$ $hasField\_GPM(?z, ?b) \wedge hasSubstance1\_GPM(?z, ?c) \wedge$ $has\_S1\_Num\_GPM(?z, 1) \wedge has\_S2\_Num\_GPM(?z, 0) \wedge$ $has\_F\_Num\_GPM(?z, 1) \rightarrow hasField\_GSM(?a, ?b) \wedge$ $hasSubstance2\_GSM(?a, added\_S2\_y2) \wedge$ $hasSubstance1\_GSM(?a, ?c) \wedge has\_S1\_Num\_GSM(?a, 1) \wedge$ $has\_S2\_Num\_GSM(?a, 1) \wedge has\_F\_Num\_GSM(?a, 1)$ If the component Substance2 is missing, add one to complete the Su-Field Model.
Rule 3	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 1) \wedge Generic\_Problem\_Model(?z) \wedge$ $correspondsTo\_GPM(?x, ?z) \wedge Generic\_Solution\_Model(?a) \wedge$ $correspondsTo\_GSM(?x, ?a) \wedge Substance(?c) \wedge Substance(?d) \wedge$ $hasSubstance1\_GPM(?z, ?c) \wedge hasSubstance2\_GPM(?z, ?d) \wedge$ $has\_S1\_Num\_GPM(?z, 1) \wedge has\_S2\_Num\_GPM(?z, 1) \wedge$ $has\_F\_Num\_GPM(?z, 0) \rightarrow hasField\_GSM(?a, added\_F\_z2) \wedge$ $hasSubstance1\_GSM(?a, ?c) \wedge hasSubstance2\_GSM(?a, ?d) \wedge$ $has\_S1\_Num\_GSM(?a, 1) \wedge has\_S2\_Num\_GSM(?a, 1) \wedge$ $has\_F\_Num\_GSM(?a, 1)$ If the component Field is missing, add one to complete the Su-Field Model.

Table 5.4: Examples of Jess facts.

Jess facts
Jess Classes
(deftemplate owl:Thing(slot name))
(deftemplate Element extends owl:Thing)
(deftemplate Substance extends Element)
(deftemplate Field extends Element)
Jess Individuals
(assert(Substance(name Water)))
(assert(Substance(name Liquid)))
Jess Properties
(assert(description Water "H_2O"))
(assert(description Liquid "A substance with a definite volume but no fixed shape"))
(assert(includes Liquid Water))

when the field is missing (*has_F_Num_GPM*=0), based on which an additive field is added to the Su-Field model after the ontology reasoning.

In the process of searching heuristic abstract solutions, there are forty SWRL rules (Appendix G) in total, including nine rules for the inventive standards of A-class and thirty-one rules for the standards of B-class.

5.4.3 Implementing the Search of Heuristic Abstract Solutions based on the Rule Engine

5.4.3.1 The Function of OWL -> Jess

During the process of OWL → Jess, the OWL knowledge is transferred to the Jess rule engine.

The classes in OWL knowledge base are mapped onto the Jess templates that are used to define the classes and the hierarchy of classes, similar to object-oriented way. For example, in the TRIZ knowledge sources ontologies, the classes *Element*, *Substance* and *Field* are transformed into Jess classes, as shown in Table 5.4. The *deftemplate* construct in Jess is employed to define the type of slots in a fact. The *extends* construct indicates

the inheritance relationship between two templates defining facts. *OWL:Thing* ia a most top level class in OWL from which all other classes are directly or indirectly inherited.

In order to perform inference by means of Jess, these facts in TRIZ knowledge sources ontologies should be transformed into Jess facts. For example, as shown in Table 5.4, the fact that *Water* and *Liquid* are two instances of the class *Substance*, is transformed into the Jess facts by using the *assert* construct. Meanwhile, it is also necessary to transform the relationships facts between instances or between instances and data values into Jess facts. For example, two relations, that is, *Water* (an instance of *Substance*) has the description of "H_2O" and *Liquid* (an instance of *Substance*) includes *Water*, are transformed into the Jess facts as shown in Table 5.4.

5.4.3.2 The Function of SWRL -> Jess

In this phase, the SWRL-based reasoning rules are transformed to Jess rules. The transformation is relatively straightforward. For example, the rules in Table 5.3 is transformed to Jess rules, as depicted in Table 5.5.

Table 5.5: The Jess rules for *General Solution 1.*

Name	Jess rules
Rule 1	$(defrule\ Rule\ 1\ (Problem(name\ ?x))(InventiveStandard(name\ ?y))$
	$(chooses\_IS\ ?x\ ?y)(has\_keyid\_of\_type\ ?y\ 1)$
	$(Generic\_Problem\_Model(name\ ?z))(correspondsTo\_GPM\ ?x\ ?z)$
	$(Generic\_Solution\_Model(name\ ?a))(correspondsTo\_GSM\ ?x\ ?a)$
	$(Field(name\ ?b))(Substance(name\ ?c))(hasField\_GPM\ ?z\ ?b)$
	$(hasSubstance2\_GPM\ ?z\ ?c)(has\_S1\_Num\_GPM\ ?z\ 0)$
	$(has\_S2\_Num\_GPM\ ?z\ 1)(has\_F\_Num\_GPM\ ?z\ 1)$
	$\Rightarrow\ (assert(hasField\_GSM\ ?a\ ?b))$
	$(assertOWLProperty\ hasField\_GSM\ ?a\ ?b)$
	$(assert(hasSubstance1\_GSM\ ?a\ added\_S1\_x2))$
	$(assertOWLProperty\ hasSubstance1\_GSM\ ?a\ added\_S1\_x2)$
	$(assert(hasSubstance2\_GSM\ ?a\ ?c))$
	$(assertOWLProperty\ hasSubstance2\_GSM\ ?a\ ?c)$
	$(assert(has\_S1\_Num\_GSM\ ?a\ 1))$

Name	Jess rules
	$(assertOWLProperty\ has\_S1\_Num\_GSM\ ?a\ 1)$
	$(assert(has\_S2\_Num\_GSM\ ?a\ 1))$
	$(assertOWLProperty\ has\_S2\_Num\_GSM\ ?a\ 1)$
	$(assert(has\_F\_Num\_GSM\ ?a\ 1))$
	$(assertOWLProperty\ has\_F\_Num\_GSM\ ?a\ 1))$
Rule 2	$(defrule\ Rule\ 2\ (Problem(name\ ?x))(InventiveStandard(name\ ?y))$
	$(chooses\_IS\ ?x\ ?y)(has\_keyid\_of\_type\ ?y\ 1)$
	$(Generic\_Problem\_Model(name\ ?z))(correspondsTo\_GPM\ ?x\ ?z)$
	$(Generic\_Solution\_Model(name\ ?a))(correspondsTo\_GSM\ ?x\ ?a)$
	$(Field(name\ ?b))(Substance(name\ ?c))(hasField\_GPM\ ?z\ ?b)$
	$(hasSubstance1\_GPM\ ?z\ ?c)(has\_S1\_Num\_GPM\ ?z\ 1)$
	$(has\_S2\_Num\_GPM\ ?z\ 0)(has\_F\_Num\_GPM\ ?z\ 1)$
	$\Rightarrow\ (assert(hasField\_GSM\ ?a\ ?b))$
	$(assertOWLProperty\ hasField\_GSM\ ?a\ ?b)$
	$(assert(hasSubstance1\_GSM\ ?a\ ?c))$
	$(assertOWLProperty\ hasSubstance1\_GSM\ ?a\ ?c)$
	$(assert(hasSubstance2\_GSM\ ?a\ added\_S2\_y2))$
	$(assertOWLProperty\ hasSubstance2\_GSM\ ?a\ added\_S2\_y2)$
	$(assert(has\_S1\_Num\_GSM\ ?a\ 1))$
	$(assertOWLProperty\ has\_S1\_Num\_GSM\ ?a\ 1)$
	$(assert(has\_S2\_Num\_GSM\ ?a\ 1))$
	$(assertOWLProperty\ has\_S2\_Num\_GSM\ ?a\ 1)$
	$(assert(has\_F\_Num\_GSM\ ?a\ 1))$
	$(assertOWLProperty\ has\_F\_Num\_GSM\ ?a\ 1))$
Rule 3	$(defrule\ Rule\ 3\ (Problem(name\ ?x))(InventiveStandard(name\ ?y))$
	$(chooses\_IS\ ?x\ ?y)(has\_keyid\_of\_type\ ?y\ 1)$
	$(Generic\_Problem\_Model(name\ ?z))(correspondsTo\_GPM\ ?x\ ?z)$
	$(Generic\_Solution\_Model(name\ ?a))(correspondsTo\_GSM\ ?x\ ?a)$
	$(Substance(name\ ?c))(Substance(name\ ?d)(hasSubstance1\_GPM\ ?z\ ?c)$
	$(hasSubstance2\_GPM\ ?z\ ?d)(has\_S1\_Num\_GPM\ ?z\ 1)$
	$(has\_S2\_Num\_GPM\ ?z\ 1)(has\_F\_Num\_GPM\ ?z\ 0)$
	$\Rightarrow\ (assert(hasField\_GSM\ ?a\ added\_F\_z2))$

Name	Jess rules
	$(assertOWLProperty\ hasField\_GSM\ ?a\ added\_F\_z2)$
	$(assert(hasSubstance1\_GSM\ ?a\ ?c))$
	$(assertOWLProperty\ hasSubstance1\_GSM\ ?a\ ?c)$
	$(assert(hasSubstance2\_GSM\ ?a\ ?d))$
	$(assertOWLProperty\ hasSubstance2\_GSM\ ?a\ ?d)$
	$(assert(has\_S1\_Num\_GSM\ ?a\ 1))$
	$(assertOWLProperty\ has\_S1\_Num\_GSM\ ?a\ 1)$
	$(assert(has\_S2\_Num\_GSM\ ?a\ 1))$
	$(assertOWLProperty\ has\_S2\_Num\_GSM\ ?a\ 1)$
	$(assert(has\_F\_Num\_GSM\ ?a\ 1))$
	$(assertOWLProperty\ has\_F\_Num\_GSM\ ?a\ 1))$

5.4.3.3 The Execution of the Jess Inference Engine

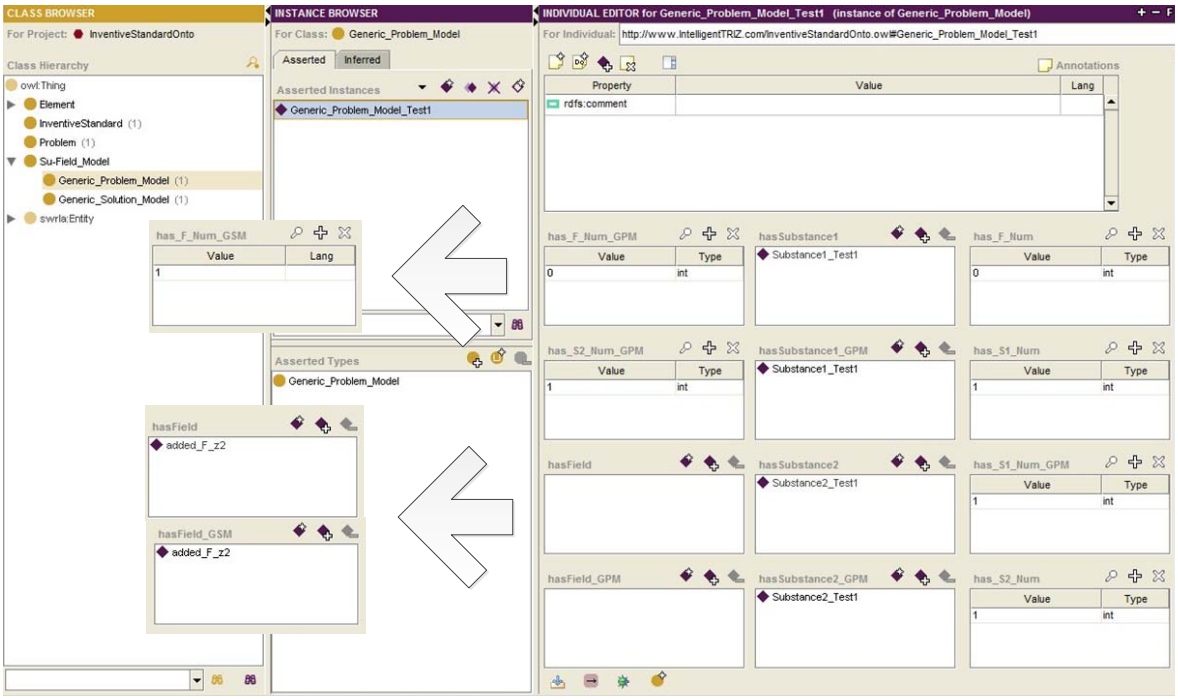


Figure 5.4: An example of inference result.

Importing Jess files including Jess facts and Jess rules, the process of searching

heuristic abstract solutions are implemented by the inference engine. The engine applies the rules to facts in the fact base, and activates the rules whose premises are satisfied, fires the activated rules and executes the actions specified in the head of these activated rules. The process is terminated until no rules are in the rule base, and then the heuristic abstract solutions are obtained. Figure 5.4 depicts the inference results for an incomplete problem model without *Field*, in which a heuristic abstract solution- "an additive *Field* is added" is obtained. The detailed process of using this method to solve a specific problem is depicted in Section 7.4.3.

5.5 Summary

This chapter introduces a method for searching heuristic abstract solutions to inventive problems based on semantic similarity and ontology inference.

Considering that all the TRIZ knowledge sources are described as short texts, the missing links among the TRIZ knowledge sources are defined based on short-text semantic similarity. Firstly, TRIZ users start solving inventive problems with the TRIZ knowledge source of their choice to obtain an abstract solution. According to the selected items of that first knowledge source, the similar items of other knowledge sources are obtained based on the semantic similarity calculated in advance. Then, with the help of these similar items, the ontology reasoning mechanism deployed on Protégé and Jess, is used to provide heuristic solutions dynamically for TRIZ users.

However, there is still a problem, that is, the interpretation of Solution Model in real life, which is the last step in Su-Field analysis. As stated in this chapter, an abstract solution is obtained finally according to this approach. Then, in order to obtain the concept solution, the interpretation of the abstract solution is generally implemented manually with the help of the physical effects knowledge base, which link generic technical functions with specific applications and systems. The physical effects compatible with the context of the specific problem should be chosen to assist the users to instantiate the Solution Model. So in order to facilitate this process, an ontology-based approach of using physical effects is proposed in next chapter.

Bibliography

[1] E. Bonjour and J. Renaud, "Pilotage des systeme de connaissances de competences: Comment definir les concepts principaux," in *In: Proceedings of Colloque Interrnational de Génie Industriel (CIGI)*, 2005.

[2] D. Cavallucci and T. Eltzer, "Parameter network as a means for driving problem solving process," *International Journal of Computer Applications in Technology*, vol. 30, no. 12, pp. 125–136, 2007.

[3] H. Cong and L. Tong, "Similarities between triz principles," *The TRIZ Journal*, 2005.

[4] I. Horrocks, P. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean, "Swrl: A semantic web rule language combining owl and ruleml," in *W3C Member Submission*, 2004.

[5] E. Friedman-Hill, "Jess, the rule engine for the java platform," http://www.jessrules.com/jess/docs/Jess71p2.pdf.

Chapter 6

Ontology-based Approach for Using the Physical-Chemical-Geometrical Effects

Su-Field analysis, one of the inventive problem solving tools, can be used to analyze and improve the efficiency of a technical system. As one of the most important phases of Su-Field analysis, the last step is normally implemented manually with the help of the physical effects knowledge base, which links generic technical functions with specific applications and systems. The physical effects compatible with the context of the specific problem should be chosen to assist the users to instantiate the Solution Model. However, the physical effects and the specific problems are built at different levels of abstraction, and it is difficult for the users to choose one. In fact, given a certain function, too many physical effects are eligible while with the detailed context of the problem, no physical effect could be returned.

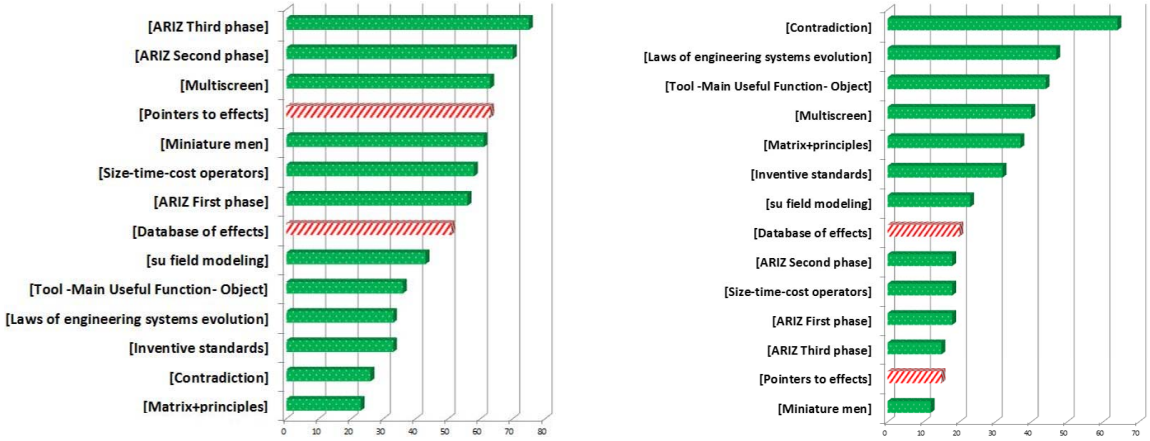
In order to facilitate the use of physical effects, this chapter firstly proposes a new way of representing physical effects by making explicit a state change, that is, the couple of two states before and after applying them. Then, knowledge about using physical effects is formalized based on ontologies. Finally, the reasoning process of using physical effects is performed with the support of the inference engine.

6.1 The Problem

In the survey of "Worldwide status of TRIZ perceptions and uses" implemented by Cavallucci in 2009 [1], two frequencies were obtained, that is, the frequency of TRIZ's main components (most unknown and most often used), as shown in Figure 6.1. According to these figures, we can observe that the pointers and the database of physical effects ranks highly in the list of the most unknown TRIZ components, and ranks lowly in the list of the most often used components. Compared with other TRIZ tools, most users do not know the pointers and effects and only use the pointers and effects occasionally when they deem it necessary.

There are many reasons for this situation, such as, the large number of physical effects and the description of the pointers at high level of abstraction. With the traditional method, the search of physical effects is normally implemented manually with the help of the pointers to physical effects - consists in linking a generic technical functions with specific applications and systems, the pointers to physical effects that are compatible with the context of the specific problem should be chosen to complete the Su-Field model and assist the users to interpret the Solution Model in the real world. However, the pointers to physical effects and the specific problems are built at different levels of abstraction. It is therefore difficult for users to choose among too many eligible pointers to physical effects given a certain function while with a detailed context of the problem, it is possible that no pointer to a physical effect be returned [2].

Accordingly, there is a need for a new manner of modeling physical effects and a method that does not rely on the pre-stored pointers and that can process the user's retrieval more dynamically. In this chapter, a new way of representing physical effects, that is, as a couple of two states before and after applying them is proposed. Then, based on ontology, the knowledge of using physical effects is formalized in OWL and the rules for retrieving physical effects are interpreted and represented in SWRL. After mapping the knowledge and constraints of using physical effects onto Jess facts and rules, the reasoning processes are performed by Jess rule engine to return the heuristic physical effects to the users.



(a) most unknown (b) most often used

Figure 6.1: Frequency of TRIZ's main components.

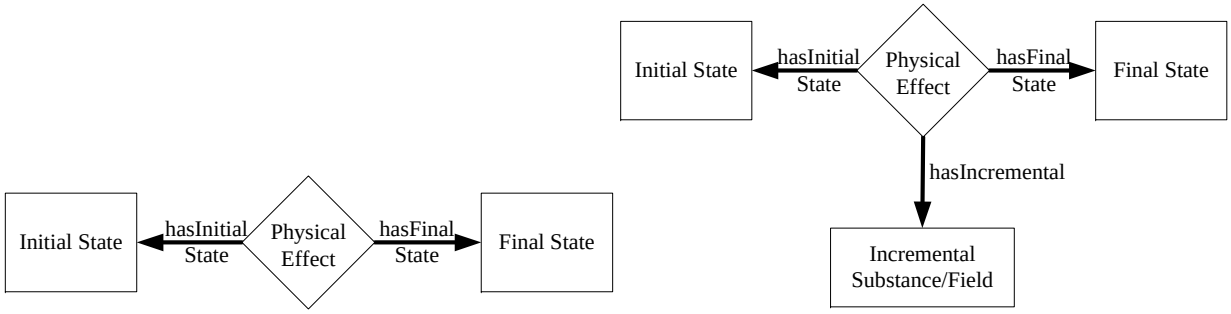
6.2 The Formalization of Physical Effects

During the process of Su-Field analysis, while the inventive standards do not produce recommendations in terms of what physical substances or fields should be used, the collection of physical effects provides the mapping between technical functions and available natural laws.

In the standard method, the physical effects are searched with the help of pointers, that is, each physical effect should be mapped to its corresponding pointers. However, the pointers to physical effects and the specific problems are built at different levels of abstraction. It is therefore difficult for users to choose among too many eligible pointers to physical effects given a certain function, while with a detailed context of the problem, it is possible that no pointer to a physical effect be returned. Accordingly, the physical effects need to be re-formalized to be at the same level of abstraction with the specific problems. The intuition is that if the physical effects can be formalized with two states, which is similar with Su-Field model, then the match between the solution model of the specific problem and the physical effects should be much easier.

As stated above, the basic idea is that each physical effect is formalized as a couple of two states, before and after applying it. Firstly, the physical effects (PE) are divided into three types, that is, PE about substances, PE about fields and PE about parameters[1]

[1]http://www.triz.co.uk/cp12.php.



(a) PE type: to modify substances/fields.

(b) PE type: to add substances/fields.

Figure 6.2: The formalization of physical effects(PE).

according to the object they are related to, for example, a physical effect used to absorb gas is considered as PE about substances. Considering that the classifications of substance and field are much more clear and well-defined than the hierarchy of parameters of system, only the physical effects about substances and fields are taken into account in this research. Then, according to the different ways to change, two sub-types are defined respectively for each kind of physical effects:

- PE about substances
  - PE to add substances
  - PE to modify substances
- PE about fields
  - PE to add fields
  - PE to modify fields

The physical effects of the types “PE to modify substances/fields" are modeled as shown in Figure 6.2(a). For the physical effects of the types “PE to add substances/fields", two additional states “Incremental substance" and “Incremental field" are added respectively to describe the incremental part after applying the physical effects, as depicted in Figure 6.2(b).

Generally, each physical effect may correspond to several kinds of change of states, and so this formalization of physical effects is implemented based on the accurate analysis of physical effects with the help of domain experts. For example, "PE81- Evaporation: by

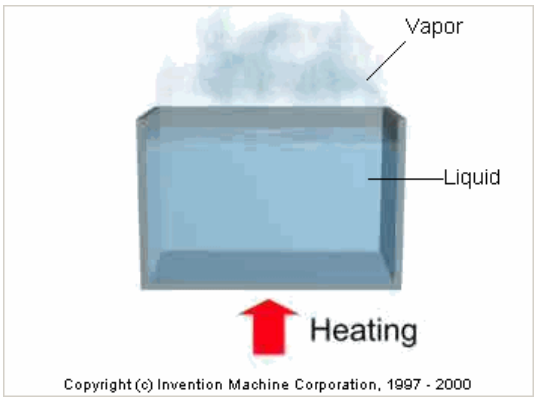


Figure 6.3: Evaporation: Liquid -> Vapor.

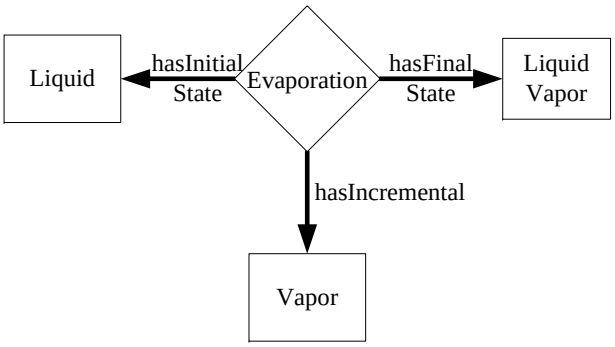


Figure 6.4: The formalization of Evaporation.

using the physical effect Evaporation, vapor can be generated from liquid", described as Figure 6.3, and its corresponding formalization is shown in Figure 6.4.

6.3 Ontology Reasoning for Searching Heuristic Physical Effects

This section describes the process of searching heuristic physical effects in detail. Firstly, in order to formalize the knowledge of using physical effects, two ontologies - the inventive standards ontology (Section 5.4.1) and the physical effects ontology are built. Then, the reasoning rules are set to describe this heuristic process. Finally, the ontology inference is carried out to provide the heuristic physical effects.

6.3.1 The Physical Effects Ontology

Figure 6.5 shows the framework of the physical effects ontology. The class *Physical_Effect* is built with the property *keyword*, which makes it possible to obtain the heuristic physical effects through the search of keyword [3]. For each *Physical_Effect*, two *state*s before and after the use of physical effects, are defined through the properties *hasInitialState* and *hasFinalState*.

All the *Physical_Effect*s are divided into two main kinds: *Sub_PE* - the *Physical_Effect*s which change *Substance* and *Field_PE* - the *Physical_Effect*s which change

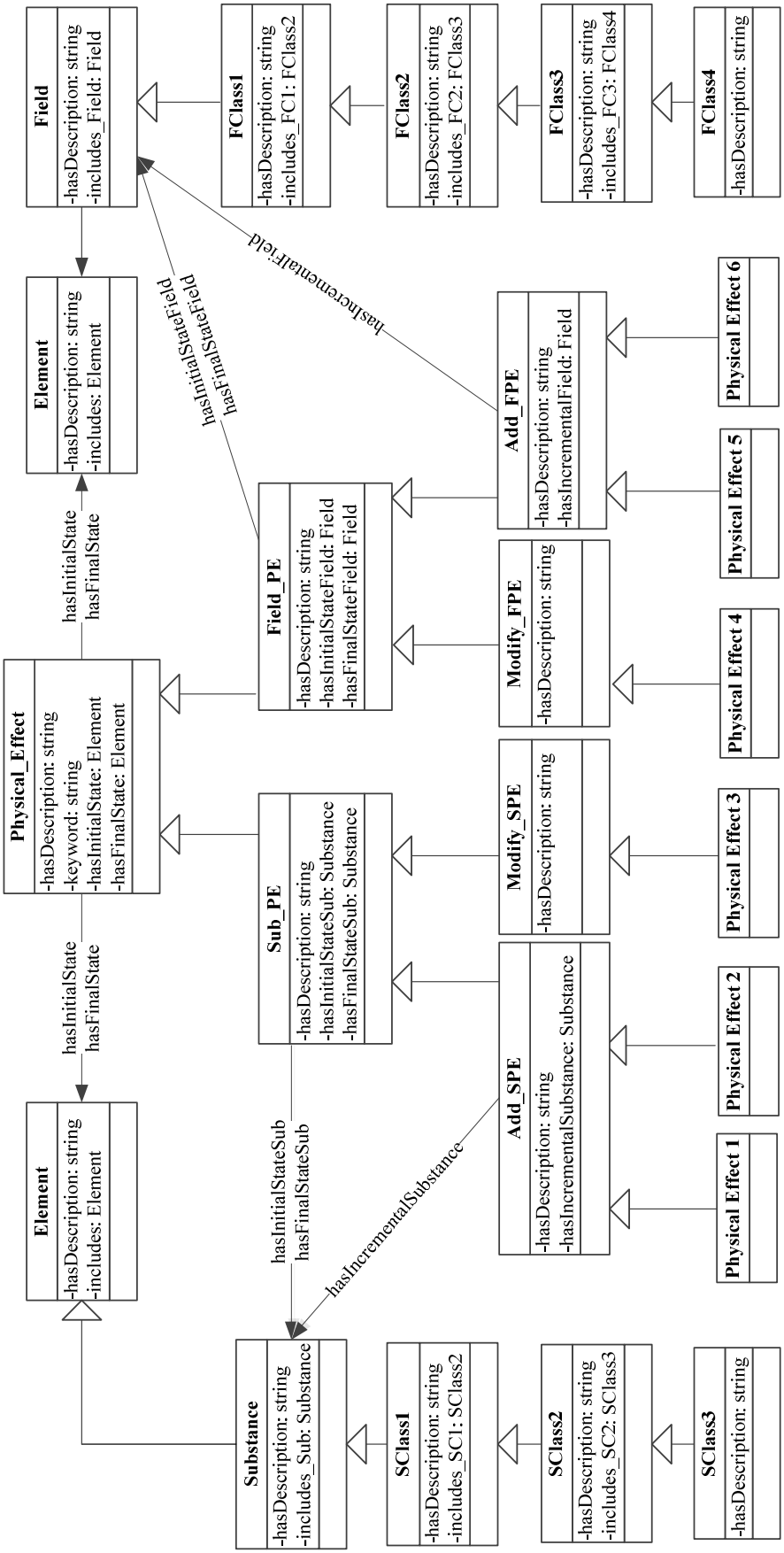


Figure 6.5: The physical effects ontology.

Field. Two inherited properties *hasInitialStateSub* and *hasFinalStateSub* of *Sub_PE* represent the change of *Substance*, while the two properties *hasInitialStateField* and *hasFinalStateField* of *Field_PE* represent the change of *Field*.

According to the four behavior concepts presented in Section 6.2, *Sub_PE* corresponds to *Add_SPE* and *Modify_SPE* and *Field_PE* to *Add_FPE* and *Modify_FPE*. For *Add_SPE* and *Add_FPE*, there are two additional properties *hasIncrementalStateSub* and *hasIncrementalStateField* to describe the incremental *Substance* or *Field*.

As shown in Figure 6.5, the hierarchy of *Substance* with three-level (*SClass1*, *SClass2* and *SClass3*) and *Field* with four-level (*FClass1*, *FClass2*, *FClass3* and *FClass4*) are defined respectively. The property *includes* is used to define the relations "parent-child" (hypernym-hyponym) between two objects, such as, "Liquid" and "Water", and it consists of two sub-properties: *includes_Sub* for *Substance* and *includes_Field* for *Field*. For each sub-property, several sub-subsidiary properties are defined to represent the relations in different levels, for example, two sub-subsidiary properties *includes_SC1* and *includes_SC2* represent the relations of *SClass1* and *SClass2*, *SClass2* and *SClass3*. According to this hierarchy, *Substance* and *Field* are instantiated, and the detailed classifications are shown in Appendix H.

We take the physical effect "PE81- Evaporation" as an example, its corresponding concepts and relationships are built in the physical effects ontology. There are several types of change during the process of the evaporation, such as, add "Vapor" or change "Pressure". Supposed that its initial state includes Substance "Liquid" and the final state is comprised of "Liquid" and "Vapor". As a result, the physical effect "PE81- Evaporation" is considered as an *Add_SPE*, and its values of the properties *hasInitialStateSub*, *hasFinalStateSub* and *hasIncrementalStateSub* are "Liquid", "Liquid and Vapor" and "Vapor".

Table 6.1 shows several OWL encoding classes and properties in the physical effects ontology.

6.3.2 Ontology Reasoning Rules

The rules for searching heuristic physical effects are divided into two classes: forty-two IS (Inventive Standard) rules and eight PE (Physical Effect) rules. The inference with the IS rules yields to several abstract types of physical effects, and the PE rules are used to

Table 6.1: The OWL encoding classes and properties in the physical effects ontology.

Classes and Properties
<owl:Class rdf:ID=" Physical_Effect ">
<rdfs:subClassOf>
<owl:Class rdf:ID="owl:Thing"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Field_PE">
<rdfs:subClassOf>
<owl:Class rdf:ID="Physical_Effect"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Add_FPE">
<rdfs:subClassOf rdf:resource="#Field_PE"/>
</owl:Class>
<owl:Class rdf:ID="Modify_FPE">
<rdfs:subClassOf rdf:resource="#Field_PE"/>
<owl:disjointWith>
<owl:Class rdf:ID="Add_FPE"/>
</owl:disjointWith>
</owl:Class>
<owl:DatatypeProperty rdf:ID="hasKeyword_PE">
<rdfs:domain rdf:resource="#Physical_Effect"/>
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:ObjectProperty rdf:ID="hasInitialStateField">
<rdfs:domain rdf:resource="#Field_PE"/>
<rdfs:range rdf:resource="#Field"/>
<rdfs:subPropertyOf>
<owl:ObjectProperty rdf:ID="hasInitialState"/>
</rdfs:subPropertyOf>
</owl:ObjectProperty>

find the concrete physical effects to instantiate the Solution Model.

Table 6.2: The IS rules and explanation for *General Solution 1*.

Name	IS rules and explanation
Rule 1	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 1) \rightarrow chooses\_PE\_Type(?x, 0)$ If the chosen Inventive Standard belongs to the type 1, the PE of type 0 will be chosen.
Rule 2	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 1) \rightarrow chooses\_PE\_Type(?x, 2)$ If the chosen Inventive Standard belongs to the type 1, the PE of type 2 will be chosen.

6.3.2.1 IS Rules

As stated Section 5.3.3, the user needs to choose one from five similar inventive standards of A-class and B-class obtained based on the method of calculating semantic similarity. According to the type of chosen inventive standard, its related types of physical effects are generated based on the inference with the IS rules in this section.

Accordingly, forty-two IS rules (Appendix I.1) are divided into two classes:

- Eight IS rules for A-class inventive standards: Forty inventive standards are condensed and generalized into seven generalized standard solutions, and each kind of standards correspond to the same types of physical effects. As a result, the inference rules are set according to seven generalized standard solutions instead of all the forty inventive standards. There are at least one IS rule for each generalized standard solution.

- Thirty-four IS rules for B-class inventive standards: There are thirty-one inventive standards identified as the implementation of existing inventive principles, and each standard corresponds to distinctive types of physical effects. So the IS rules are set for each inventive standard.

For example, there are two IS rules for *General Solution 1*, as shown in Table 6.2. The obtained values (0,1,2,3) of the property *chooses_PE_Type* are: 0 - add substance, 1 - modify substance, 2 - add field, 3 - modify field.

Generally, several kinds of physical effects are obtained in this step, and the user needs to provide more concrete information in order to find the appropriate physical effects in

Table 6.3: The PE rules and explanation for the physical effects *Add_SPE*s.

Name	PE rules and explanation
Rule 1	$Problem(?x) \wedge Generic\_Problem\_Model(?y) \wedge$ $Generic\_Solution\_Model(?z) \wedge correspondsTo\_GPM(?x, ?y) \wedge$ $correspondsTo\_GSM(?x, ?z) \wedge chooses\_PE\_Type(?x, 0) \wedge$ $Substance(?a) \wedge changesOnOrFor\_Element(?z, ?a) \wedge Add\_SPE(?b) \wedge$ $hasIncrementalSubstance(?b, ?a) \rightarrow chooses\_PE(?x, ?b)$ If a certain substance needs to be added, all the PEs which can add this substance, are obtained.
Rule 2	$Problem(?x) \wedge Generic\_Problem\_Model(?y) \wedge$ $Generic\_Solution\_Model(?z) \wedge correspondsTo\_GPM(?x, ?y) \wedge$ $correspondsTo\_GSM(?x, ?z) \wedge chooses\_PE\_Type(?x, 0) \wedge$ $Substance(?a) \wedge changesOnOrFor\_Element(?z, ?a) \wedge Substance(?c) \wedge$ $includes\_Sub(?a, ?c) \wedge Add\_SPE(?b) \wedge$ $hasIncrementalSubstance(?b, ?c) \rightarrow chooses\_PE(?x, ?b)$ If a certain substance needs to be added, all the PEs which can add this substance or its children substances, such as, liquid and water, are obtained.

the next step. Two types of information need to be provided: on the one hand, only one type of physical effect needs to be chosen from the obtained set, and on the other hand, the level of abstraction for the problem description needs to be set according to the classifications of *Substance* and *Field* (Appendix H) in the Su-Field Model Ontology. For example, the user wants to "add pure gas" rather than "add mixed gas".

6.3.2.2 PE Rules

According to the physical effects Ontology, the PE rules are set to search the most appropriate physical effects for the special case. As presented in Section 6.3.1, there are four classes of physical effects, that is, *Add_SPE*, *Modify_SPE*, *Add_FPE* and *Modify_FPE*, and each class corresponds to two PE rules, that is, one for searching the physical effects related to the chosen object (*Substance* or *Field*) and the other for searching the physical effects related to the children objects of the chosen one. Eight PE rules (Appendix I.2) are designed for searching heuristic physical effects, for example, for the physical effects *Add_SPE*s, the PE rules and explanation are shown in Table 6.3.

Table 6.4: Examples of Jess facts.

Jess facts
Jess Classes
(deftemplate owl:Thing (slot name))
(deftemplate Physical_Effect extends owl:Thing)
(deftemplate Field_PE extends Physical_Effect)
(deftemplate Add_FPE extends Field_PE)
(deftemplate Modify_FPE extends Field_PE)
Jess Individuals
(assert (Modify_FPE (name Mosbauer_Absorption)))
(assert (Field (name gammaRayField)))
Jess Properties
(assert (hasKeyword_PE Mosbauer_Absorption "absorb gamma ray"))
(assert(hasInitialStateField Mosbauer_Absorption gammaRayField))

6.3.3 Implementing the Query of Physical Effects based on the Rule Engine

The process of searching heuristic physical effects is carried out through the use of Jess, which includes three tasks:

- The knowledge about the use of physical effects, including the inventive standards ontology and the physical effects ontology in OWL is transferred to Jess facts.
- The constraints about searching for heuristic physical effects in SWRL are transferred to Jess rules.
- Jess rule engine executes to generate the heuristic physical effects.

6.3.3.1 The Function of OWL -> Jess

This phase consists of two phases: on the one hand, the classes and their hierarchy in the inventive standards ontology and the physical effects ontology are mapped onto the Jess templates; on the other hand, the individuals and their relationships in the inventive standards ontology and the physical effects ontology are transformed into Jess facts.

For example, as shown in Table 6.4, the class *Physical_Effect* and its classification are transformed into Jess classes, and an instance of *Modify_FPE*, that is, *Mosbauer_Absorption*, and its relationship with *gammaRayField* are transferred to Jess facts.

6.3.3.2 The Function of SWRL -> Jess

There are fifty SWRL rules, including forty-two IS rules and eight PE rules, transferred to Jess rules. Taking the IS rules in Table 6.2 and the PE rules in Table 6.3 as examples, the corresponding Jess rules are shown in Table 6.5.

Table 6.5: Examples of Jess rules.

Name	Jess rules
ISRule 1	$(defrule\ ISRule1\ (Problem(name\ ?x))(InventiveStandard(name\ ?y))$
	$(chooses\_IS\ ?x\ ?y)(has\_keyid\_of\_type\ ?y\ 1)$
	$\Rightarrow\ (assert(chooses\_PE\_Type\ ?x\ 0))$
	$(assertOWLProperty\ chooses\_PE\_Type\ ?x\ 0))$
ISRule 2	$(defrule\ ISRule2\ (Problem(name\ ?x))(InventiveStandard(name\ ?y))$
	$(chooses\_IS\ ?x\ ?y)(has\_keyid\_of\_type\ ?y\ 1)$
	$\Rightarrow\ (assert(chooses\_PE\_Type\ ?x\ 2))$
	$(assertOWLProperty\ chooses\_PE\_Type\ ?x\ 2))$
PERule 1	$(defrule\ PERule1\ (Problem(name\ ?x))$
	$(Generic\_Problem\_Model(name\ ?y))$
	$(Generic\_Solution\_Model(name\ ?z))$
	$(correspondsTo\_GPM\ ?x\ ?y)(correspondsTo\_GSM\ ?x\ ?z)$
	$(chooses\_PE\_Type\ ?x\ 0)(Substance(name\ ?a))$
	$(changesOnOrFor\_Element\ ?z\ ?a)(Add\_SPE(name\ ?b))$
	$(hasIncrementalSubstance\ ?b\ ?a)$
	$\Rightarrow\ (assert(chooses\_PE\ ?x\ ?b))$
	$(assertOWLProperty\ chooses\_PE\_Type\ ?x\ ?b))$
PERule 2	$(defrule\ PERule2\ (Problem(name\ ?x))$
	$(Generic\_Problem\_Model(name\ ?y))$
	$(Generic\_Solution\_Model(name\ ?z))$
	$(correspondsTo\_GPM\ ?x\ ?y)(correspondsTo\_GSM\ ?x\ ?z)$

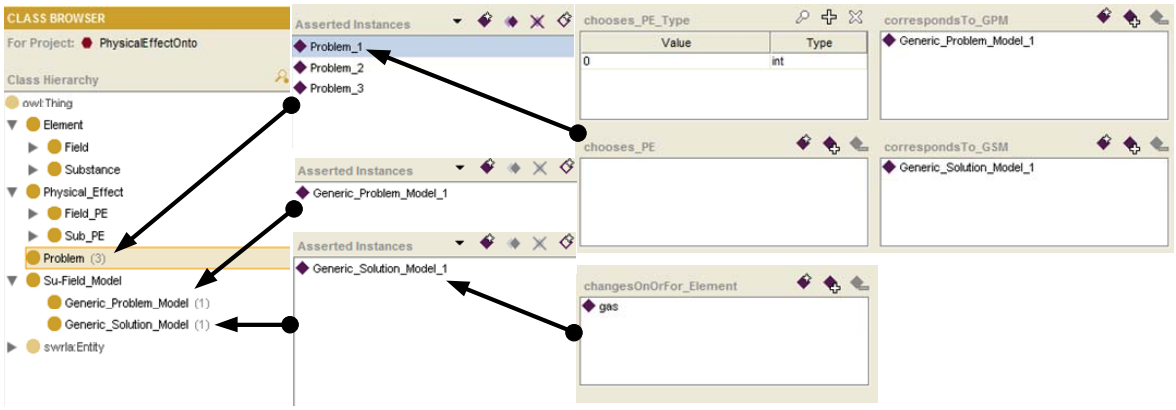


Figure 6.6: Before inference.

Name	Jess rules
	$(chooses\_PE\_Type\ ?x\ 0)(Substance(name\ ?a))$
	$(changesOnOrFor\_Element\ ?z\ ?a)(Substance(name\ ?c))$
	$(includes\_Sub\ ?a\ ?c)(Add\_SPE(name\ ?b))$
	$(hasIncrementalSubstance\ ?b\ ?c)$
	$\Rightarrow\ (assert(chooses\_PE\ ?x\ ?b))$
	$(assertOWLProperty\ chooses\_PE\_Type\ ?x\ ?b))$

6.3.3.3 The Execution of the Jess Inference Engine

In this phase, the Jess rule engine is executed to generate the heuristic physical effects. As shown in Figure 6.6, several necessary individuals are built through Protégé or Protégé-OWL API[1] in Java applications, for example, "Problem_1", "Generic_Problem_Model_1" and "Generic_Solution_Model_1". Then Jess will run its inference engine and generate some new knowledge, which is represented as Jess facts. After transferring to OWL format, the final results are shown in Figure 6.7, in which a heuristic physical effect "Evaporation" is related to "Problem_1" for generating gas. The detailed process of using this method to solve a specific problem is depicted in Section 7.4.4.

[1]http://protege.stanford.edu/plugins/owl/api/.

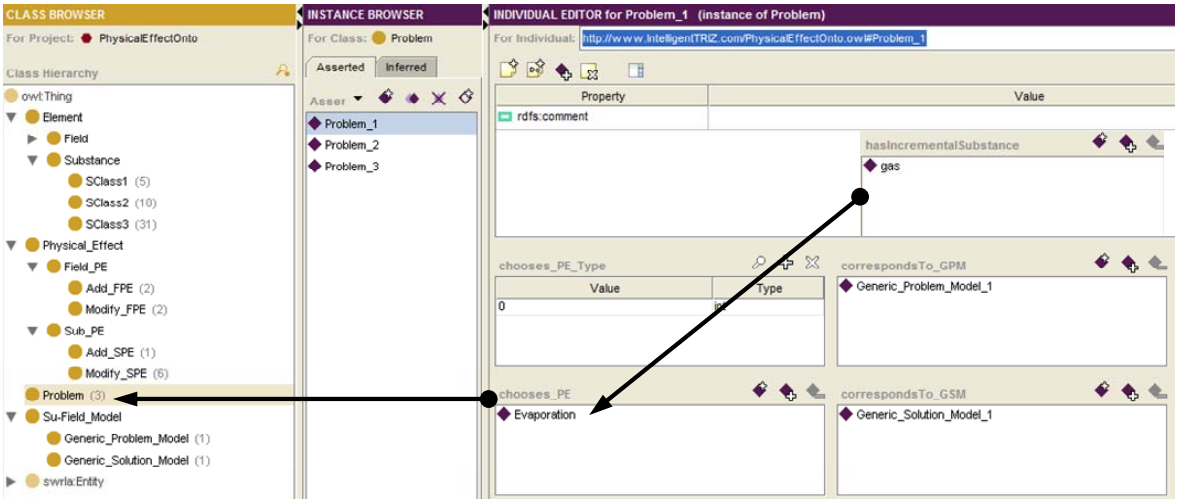


Figure 6.7: After inference.

6.4 Summary

This chapter firstly proposes a new way of representing physical effects using the change of two states, that is, the couple of two states before and after applying physical effects. Then, the physical effects ontology is built in OWL (Ontology Web Language), and the constraint knowledge of using physical effects is formalized in SWRL (Semantic Web Rule Language). Finally, the reasoning process of using physical effects is performed with the support of Jess (Java Expert System Shell) rule engine.

This research facilitates the use of physical effects and can improve the available status of using effects to a certain extent. Compared with the classical "indexing" way used in most commercial software, the formalization based on ontologies provides conceptual resources for knowledge based system (KBS) and makes it possible to automate the process of the resolution of inventive problems. It also permits the tracking of the different applications to study and compare them, and, in this way, the improvement of the whole methodology.

Bibliography

[1] D. Cavallucci, "Word wide status of TRIZ perceptions and user: a survey of results," in *Report at TRIZ Future 2009*, 2009.

[2] A. Bultey, F. D. B. de Beuvron, and F. Rousselot, "A substance-field ontology to support the TRIZ thinking approach," *International Journal of Computer Applications in Technology*, vol. 30, pp. 113–124, 2007.

[3] W. Yan, C. Zanni-Merk, F. Rousselot, D. Cavallucci, and P. Collet, "A heuristic method of using the pointers to physical effects in su-field analysis," in *TRIZ Future 2012*, 2012.

Chapter 7

The Prototype: IngeniousTRIZ

From Chapter 3 to Chapter 6, we have introduced the automatic ontology-based mechanism for solving inventive problems in detail.

Chapter 3 introduces methods of calculating semantic similarity, which are the foundations of the research presented in Chapters 4 and 5. In order to facilitate the use of the contradiction matrix and inventive principles, Chapter 4 presents the automatic match between specific parameters and generic engineering parameters based on semantic similarity and case-based reasoning. In Chapter 5, the TRIZ knowledge sources are firstly formalized based on ontology, then the missing links among them are defined through the calculation of semantic similarity, and finally based on the TRIZ knowledge sources ontologies and their links, the heuristic abstract solutions are obtained through the execution of ontology inference. Chapter 6 deals with the use of physical effects to interpret the Solution Model in real life. In order to verify the proposed approaches with the users' cases, a prototype called IngeniousTRIZ, which combines the function of the approaches stated above, is developed.

This chapter introduces this prototype in terms of the following three parts: state of the art, the development, and the detailed process of using this prototype to solve a specific problem.

7.1 State of the Art

Over the last two decades, many techniques in computer science, especially in Artificial Intelligence, are brought in the field of inventive design to facilitate the process of using

TRIZ to solve inventive problems.

The CREAX[1] Function database organizes a database of physical effects by function, and uses a web-based application to support the search of physical effects. It responds to user-entered query by processing the query to a combination of two pre-stored key words, which is quite limited in the specific applications. The Invention Machine's GoldFire[2] implements the search of scientific physical effects based on semantic methods. With the good performance of the proposed semantic methods and the standard representative way of problem and solution, the results are almost satisfied. However, the hierarchy of the effects is stable and any modification of the classification need to be maintained manually.

Our research group, including an expert practitioner of IDM[3]/TRIZ, has been working for several years on the formalization of basic concepts and the knowledge bases of the methodology, as well as on the extension of it to complex problems. The work within a consortium with Arcelor-Mittal, Alstom and EADS has permitted the funding of the development of a software tool based on these research works. The industrial partners provided a number of examples and cases to test the tool, refine it and finally evaluate it.

The software developed by LGECO, called STEPS, is able to effectively accompany the user from the formulation phase of the problem to its resolution, taking into account the specific context of its application. However, using STEPS to solve a specific problem requires extensive knowledge of different engineering domains and is not currently supported by it. Consequently, the resolution mainly depends on the experience and knowledge of the user.

STEPS exists in two versions. A commercial one, managed by a start-up company, Time To Innovate[4]; and a research version, property of the consortium and INSA de Strasbourg, which implements the ontology of the TRIZ/IDM concepts we have created [1, 2]. Both versions of STEPS can provide valuable assistance to the user through the intelligent analysis based on the statistic data.

In order to facilitate the inventive problem solving process, a prototype called IngeniousTRIZ was developed based on semantic similarity and ontology reasoning during these research works. On the one hand, according to the TRIZ knowledge sources

[1]http://function.creax.com/.

[2]http://inventionmachine.com/products-and-services/innovation-software/.

[3]IDM (Inventive Design Methodology) [1, 2] is an extension to TRIZ, that is intended to process problems with hundreds of parameters, contradictions and problems not necessarily linked to inventive product design (for example, in software engineering or in the organization field).

[4]http:// www.time-to-innovate.com.

Table 7.1: The environment and tools for the development of IngeniousTRIZ.

Name	Version	Library
Java Platform(Java SE)	Version 1.7.0_02	
NetBeans IDE	Version 7.2	
MySQL	Version 5.1.22	JDBC driver: mysql-connector-java-5.1.22-bin.jar
WordNet	Version 2.0	jwnl14-rc2.zip
Protégé	Version 3.4.3	OWL-API: owlapi_bin.jar owlapi_src.jar Protege-OWL API: edu.standard.smi.protegex.owl*.jar
Jess	Version 7.1p2	jess.jar

ontologies, the prototype offers to the users the relevant knowledge sources of the model they are building. On the other hand, the prototype has the ability to fill the models of the other knowledge sources automatically.

Through IngeniousTRIZ, the users start solving inventive problems with forty inventive principles, which are simple to use, to obtain an abstract solution. Then according to the selected inventive principle, the similar inventive standards, which are complicated to use, are obtained based on the semantic similarity calculated in advance. With the help of these similar inventive standards, the Su-Field models are built to generate the abstract solutions based on ontology inference. Finally, in order to interpret the abstract solution in real life, the heuristic physical effects are obtained based on ontology inference to generate the appropriate concept solutions.

7.2 The Environment and Tools

The prototype IngeniousTRIZ is developed in the Windows operating system, and the used software and libraries are shown in Table 7.1.

- **Java Platform(Java SE)** Java[1], created by by James Gosling from Sun Microsystems in 1991, is an object-oriented programming language with a built-in application

[1]http://www.oracle.com/us/technologies/java/overview/index.html.

programming interface (API)[1] that can handle graphics and user interfaces and that can be used to create applications or applets. The first publicly available version of Java (Java 1.0) was released in 1995, and the current version is Java 1.7 which is also known as Java 7. From the Java programming language the Java platform evolved.

The Java platform is not specific to any one processor or operating system, but rather an execution engine (called a virtual machine) and a compiler with a set of libraries that are implemented for various hardware and operating systems so that Java programs can run identically on all of them. The Java platform is divided into four types in terms of the purpose of use:

- Java ME[2]: Java Platform, Micro Edition, is a Java platform designed for embedded systems (mobile devices are one kind of such systems). Target devices range from industrial controls to mobile phones (especially feature phones) and set-top boxes.
- Java SE[3]: Java Platform, Standard Edition, is a widely used platform for development and deployment of portable applications for desktop and server environments.
- Java EE[4]: Java Platform, Enterprise Edition, is Oracle's enterprise Java computing platform. The platform provides an API and runtime environment for developing and running enterprise software, including network and web services, and other large-scale, multi-tiered, scalable, reliable, and secure network applications.
- JavaFX[5]: JavaFX is the next step in the evolution of Java as a rich client platform. It is designed to provide a lightweight, hardware-accelerated Java UI platform for enterprise business applications.

Taking into account the development with portable application in this research, Java Platform(Java SE) 1.7 is used.

[1] http://docs.oracle.com/javase/7/docs/api/.
[2] http://www.oracle.com/technetwork/java/javame/index.html.
[3] http://www.oracle.com/technetwork/java/javase/overview/index.html.
[4] http://www.oracle.com/technetwork/java/javaee/overview/index.html.
[5] http://www.oracle.com/technetwork/java/javafx/overview/index.html.

- **NetBeans**[1] NetBeans is an integrated development environment (IDE) for developing with Java. NetBeans IDE provides first-class comprehensive support for the newest Java technologies and latest Java enhancements before other IDEs, and it is the first IDE providing support for JDK 7. Furthermore, the design of the graphical user interface (GUI) for Java EE, Java SE and Java ME applications can be implemented quickly and smoothly by dragging and positioning GUI components from a palette into the NetBeans Editor. Especially for Java SE applications, NetBeans GUI Builder automatically takes care of correct spacing and alignment, while supporting in-place editing, as well. Taking into account the characteristics of the prototype IngeniousTRIZ, that is, a frame-based Java SE application, NetBeans 7.2 is chosen as the environment for its development.

- **MySQL**[2] The history[3] of the MySQL database goes back to 1979 when TcX, the company that developed MySQL, started working with database programs. In May 1996, MySQL version 1.0 was released to a limited group of four people, and in October 1996, MySQL 3.11.1 was released to the public as a binary distribution for Solaris. A month later, a Linux binary and the source distribution were released. The MySQL release included an ODBC driver in source form, which also included many free MySQL clients ported to MySQL. Nowadays, the MySQL database has become the world's most popular open source database because of its consistent fast performance, high reliability and ease of use.

  In order to connect to the MySQL database from the Java program, JDBC (Java Database Connectivity)[4], a Java API that provides a set of interfaces, is used. JDBC programming consists of the following six steps:

  – Register the driver. When the driver class is initialized, it is automatically registered with JDBC drivers.

  – Make a connection to the MySQL database using Connection object. *DriverManager* class or *DataSource* object can be used to create Connection object.

  – Create *Statement* object. This act as a channel to send SQL queries.

[1] https://netbeans.org/.
[2] http://dev.mysql.com/.
[3] http://www.linuxjournal.com/article/3609.
[4] http://dev.mysql.com/downloads/connector/j/3.1.html.

Table 7.2: The tables in the database "IngeniousTRIZ".

Name	Description
allinformation	Store all the information for each use, such as the specific parameters for the problem to be solved.
contradictionmatrix	Store the information for the contradiction matrix.
ip	Store the detailed information for 40 inventive principles.
is	Store the detailed information for 76 inventive standards.
isbehavior	Store the types of modification for 76 inventive standard.
sipis	Store the matching results between inventive principles and inventive standards.
sipsm	Store the matching results between inventive principles and separation methods.
sm	Store the detailed information for 11 separation methods.
transformation	Store the detailed modifications of 76 inventive standards.

Alternatively *PreparedStatement*, the pre-compiled SQL statement, can be used.

- Send SQL statements to the MySQL database.
- Process the result returned from SQL statement. For example, *SELECT* statement returns number of records, and *ResultSet* object is used to store the obtained records.
- Close all resources like *ResultSet*, *Statement* and Connection objects.

In this application, a database called "IngeniousTRIZ", which consists of nine tables, is built as shown in Table 7.2.

- **WordNet**[1] WordNet, proposed by George A. Miller in 1980s, is a large lexical database of English. It groups English words into sets of synonyms called synsets, provides short, general definitions, and records the various semantic relations between these synonym sets. The purpose is twofold: to produce a combination of dictionary and thesaurus that is more intuitively usable, and to support automatic text analysis and artificial intelligence applications.

  WordNet distinguishes between nouns, verbs, adjectives and adverbs because they follow different grammatical rules. Every synset contains a group of synonymous

[1]http://wordnet.princeton.edu/.

words or collocations (a collocation is a sequence of words that go together to form a specific meaning, such as "car pool"); different senses of a word are in different synsets. The meaning of the synsets is further clarified with short defining glosses (Definitions and/or example sentences). A typical example synset with gloss is: good, right, ripe – (most suitable or right for a particular purpose; "a good time to plant tomatoes"; "the right time to act"; "the time is ripe for great sociological changes").

In order to use WordNet in Java program, JWNL (Java WordNet Library)[1], a pure java implementation of WordNet API, provides functionality not only for data access, but also for relationship discovery and morphological processing.

- **Protégé**[2] Protégé, developed by Stanford University, is a free, open-source platform that provides a growing user community with a suite of tools to construct domain models and knowledge-based applications with ontologies. Protégé can be extended by the way of a plug-in architecture and a Java-based Application Programming Interface (API) for building knowledge-based tools and applications.

  The Protégé-OWL editor[3] is an extension of Protégé that supports OWL, which is the most recent development in standard ontology languages, endorsed by W3C to promote the Semantic Web vision. The Protégé-OWL editor enables users to[4]:

  – Load and save OWL and RDF ontologies.

  – Edit and visualize classes, properties, and SWRL rules.

  – Define logical class characteristics as OWL expressions.

  – Execute reasoners such as description logic classifiers.

  – Edit OWL individuals for Semantic Web markup.

  In the Protégé-OWL editor, there exists various plug-ins offering a variety of additional features, such as extra ontology management tools, multimedia support, querying and reasoning engines, problem solving methods, etc. The following are the plug-ins that we use in this research:

[1]http://www.stanford.edu/class/archive/cs/cs276a/cs276a.1032/projects/docs/jwnl/javadoc/.
[2]http://protege.stanford.edu/.
[3]http://protege.stanford.edu/plugins/owl/download.html.
[4]http://protege.stanford.edu/overview/protege-owl.html.

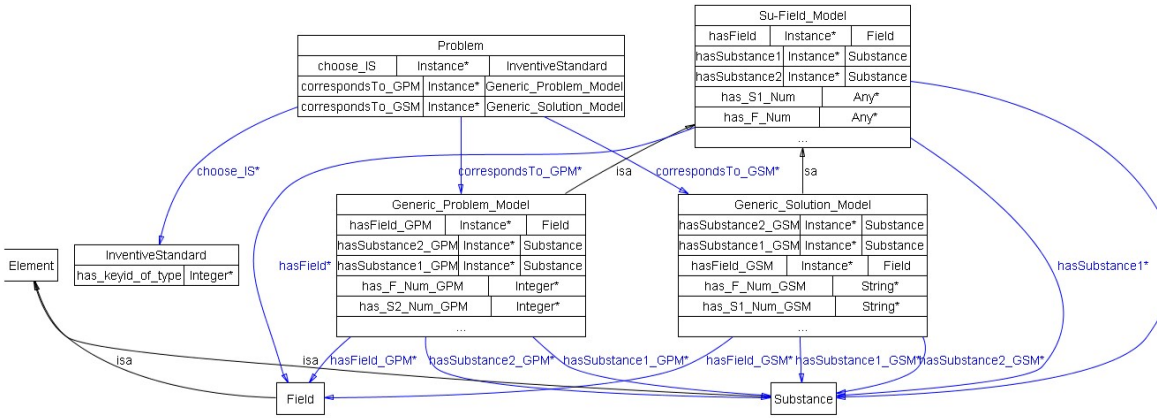


Figure 7.1: The inventive standards ontology in Protégé.

- *OntoViz*[1]: The OntoViz Tab allows you to visualize Protégé ontologies with the help of a highly sophisticated graph visualization software called Graphviz from AT&T. The types of visualizations are highly configurable and include picking a set of classes or instances to visualize parts of an ontology, displaying slots and slot edges, and specifying colors for nodes and edges.
- *OWLViz*[2]: The OWLViz Tab enables the class hierarchies in an OWL ontology to be viewed and incrementally navigated, allowing comparison of the asserted class hierarchy and the inferred class hierarchy.
- *SWRLTab*[3]: The SWRLTab is a development environment for working with SWRL rules in Protégé-OWL. It supports the editing and execution of SWRL rules and includes a set of libraries that can be used in rules, mainly including SWRL Editor, SWRL APIs, SWRL Built-in Libraries, SWRL Jess Bridge, etc.

As presented in Chapter 4, 5 and 6, three ontologies, that is, the inventive principles ontology, the inventive standards ontology, and the physical effects ontology are built in Protégé. For example, Figure 7.1 shows the framework of the inventive standards ontology in Protégé.

In order to manipulate the OWL ontologies in this Java application, the following two interfaces are used:

[1]http://protegewiki.stanford.edu/wiki/OntoViz.
[2]http://protegewiki.stanford.edu/wiki/OWLViz.
[3]http://protege.cim3.net/cgi-bin/wiki.pl?SWRLTab.

  - **Protege-OWL API**[1] The Protege-OWL API is an open-source Java library for the Web Ontology Language (OWL). This API provides classes and methods to load and save OWL files, to query and manipulate OWL data models, and to perform reasoning based on Description Logic Engines. Furthermore, the API is optimized for the implementation of graphical user interfaces. The API is designed to be used in two contexts:
    * For the development of components that are executed inside of the Protege-OWL editor's user interface
    * For the development of stand-alone applications (e.g., Java applications or Eclipse plug-ins)
  - **OWL API**[2] The OWL API is a Java API and reference implementation for creating, manipulating and serializing OWL Ontologies. The latest version of the API is focused towards OWL 2.

  As shown in Figure 7.2, OWL-API is used to build the instances and their properties, SWRL Rule Engine API[3] to execute the ontology reasoning, and Protege-OWL API to read the inferred results, which are stored in new ontologies after the inference.

- **Jess**[4] Jess is a rule engine and scripting environment written entirely in Sun's Java language by Ernest Friedman-Hill. Because its powerful scripting language gives access to all of Java's APIs, Jess is selected as the reasoning engine in this application. Figure 7.3 shows the process of ontology inference with Jess.

7.3 The Framework of IngeniousTRIZ

As shown in Figure 7.4, the window-based IngeniousTRIZ consists of eleven continuous windows to solve the problem and three windows to store the results. This prototype integrates the approaches presented in Chapter 3, 4, 5 and 6 and therefore, each approach corresponds to several steps as follows:

[1]http://protegewiki.stanford.edu/wiki/ProtegeOWL_API_Programmers_Guide.
[2]http://owlapi.sourceforge.net/index.html.
[3]SWRL Rule Engine API is packaged with Protege-OWL API before Protégé 3.5 (included).
[4]http://herzberg.ca.sandia.gov/.

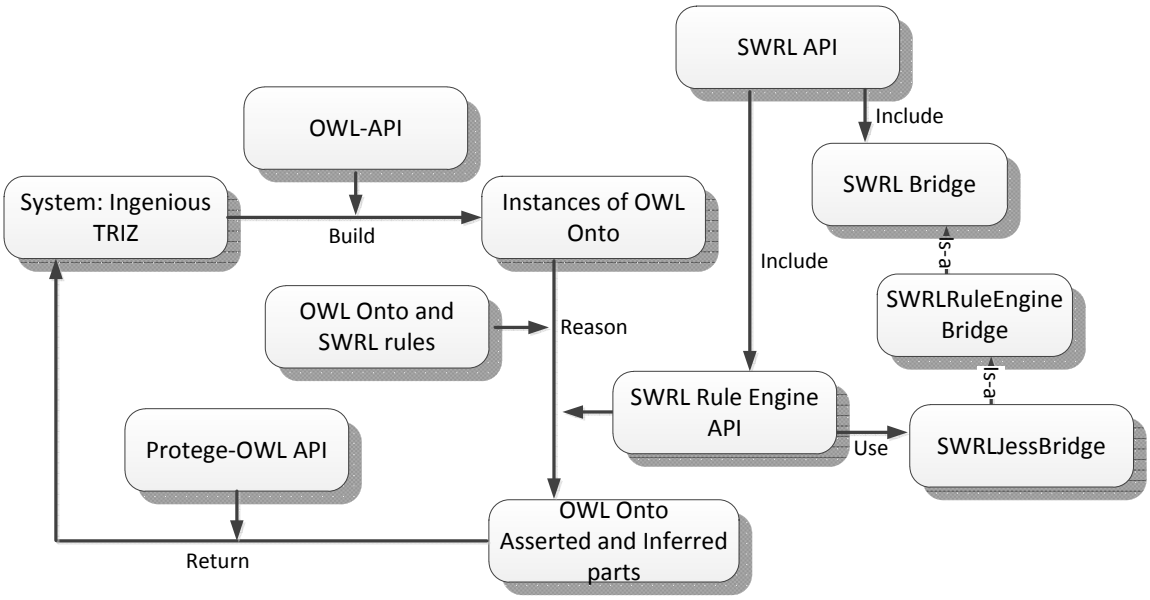


Figure 7.2: The Interaction Between IngeniousTRIZ and Ontologies.

```
run:
Loading triples for: null
   Completed triple loading after 844 ms
Importing http://sqwrl.stanford.edu/ontologies/built-ins/3.4/sqwrl.owl from location: \plugins\edu.stanford.smi.protegex.owl\sqwrl.owl
edu.stanford.smi.protegex.owl.model.impl.AbstractOWLModel loadImportedAssertionsImporting http://sqwrl.stanford.edu/ontologies/built-ins/3.4/sqwrl.owl from location:
\plugins\edu.stanford.smi.protegex.owl\sqwrl.owl
Loading triples for: http://sqwrl.stanford.edu/ontologies/built-ins/3.4/sqwrl.owl
   Completed triple loading after 14 ms
Importing http://swrl.stanford.edu/ontologies/3.3/swrla.owl from location: \plugins\edu.stanford.smi.protegex.owl\swrla.owl
edu.stanford.smi.protegex.owl.model.impl.AbstractOWLModel loadImportedAssertions
Importing http://swrl.stanford.edu/ontologies/3.3/swrla.owl from location: \plugins\edu.stanford.smi.protegex.owl\swrla.owl
Loading triples for: http://swrl.stanford.edu/ontologies/3.3/swrla.owl
   Completed triple loading after 13 ms
Postprocess: Process entities with incorrect Java type (0 entities) ... 0 ms
Postprocess: Process metaclasses (3 metaclasses) ... 1 ms
Postprocess: Process subclasses of rdf:List (1 classes) ... 0 ms
Postprocess: Instances with multiple types (12 instances) ... 3 ms
Postprocess: Add inferred superclasses ... 0 ms
Postprocess: Process orphan classes (10 classes) ... 2 ms
Postprocess: Generalized Concept Inclusion (0 axioms) ... 0 ms
Postprocess: Abstract classes... 0 ms
Postprocess: Domain and range of properties... 6 ms
Postprocess: Possibly typed entities (0 resources) ... 0 ms
Updating underlying frames model in 0 ms
 edu.stanford.smi.protegex.owl.jena.parser.ProtegeOWLParser doFinalPostProcessing
Updating underlying frames model in 0 ms
Rule engine 'SWRLJessBridge' registered with the SWRLTab bridge.
edu.stanford.smi.protegex.owl.swrl.bridge.BridgeFactory registerBridge
Rule engine 'SWRLJessBridge' registered with the SWRLTab bridge.
ruleEngine.tostring=edu.stanford.smi.protegex.owl.swrl.bridge.jess.SWRLJessBridge@33a2831e
Infered axioms size=6
Infered axioms =[has_S2_Num_GSM(Generic_Solution_Model_Test1, 1), hasField_GSM(Generic_Solution_Model_Test1, added_F_z2), has_S1_Num_GSM(Generic_Solution_Model_Test1, 1),
hasSubstance2_GSM(Generic_Solution_Model_Test1, Substance2_Test1), hasSubstance1_GSM(Generic_Solution_Model_Test1, Substance1_Test1), has_F_Num_GSM(Generic_Solution_Model_Test1, 1)]
Infered Individuals =0
 Infered Individuals =[]
File saved with 0 errors.
```

Figure 7.3: The process of ontology inference with Jess.

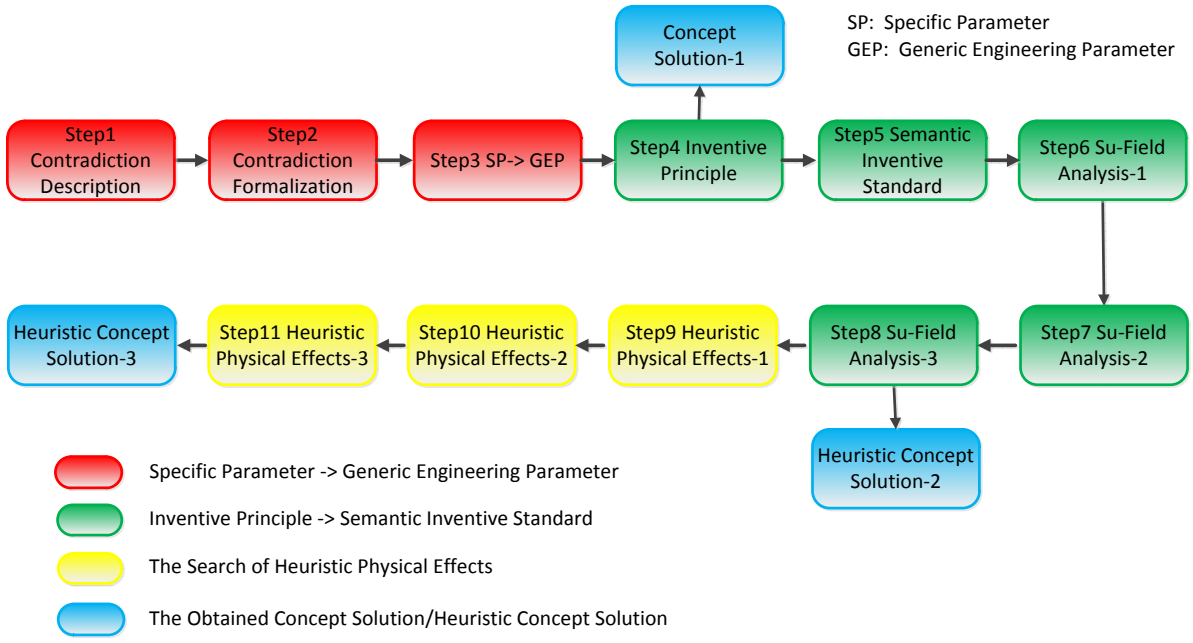


Figure 7.4: The framework of IngeniousTRIZ.

- Step1->Step4: The resolution with contradiction matrix and inventive principles. The sequence diagram for Step1->Step4 is shown in Figure 7.5.

  - **Step1: Contradiction Description** In order to solve the contradiction in the specific problem, the user is guided to input the related parameters, such as, two specific parameters.
  - **Step2: Contradiction Formalization** Based on the information obtained in Step1, the contradiction is displayed in the table form, which makes easier for the user to implement the following analysis.
  - **Step3: SP->GEP** In order to search inventive principles through the contradiction matrix, two specific parameters need to be matched with thirty-nine generic engineering parameters respectively. The user can choose them manually, or with the help of semantic search. According to the proposed methods in Chapter 4, five most similar generic engineering parameters are obtained for each specific parameter.
  - **Step4: Inventive Principle** With the help of the contradiction matrix, a

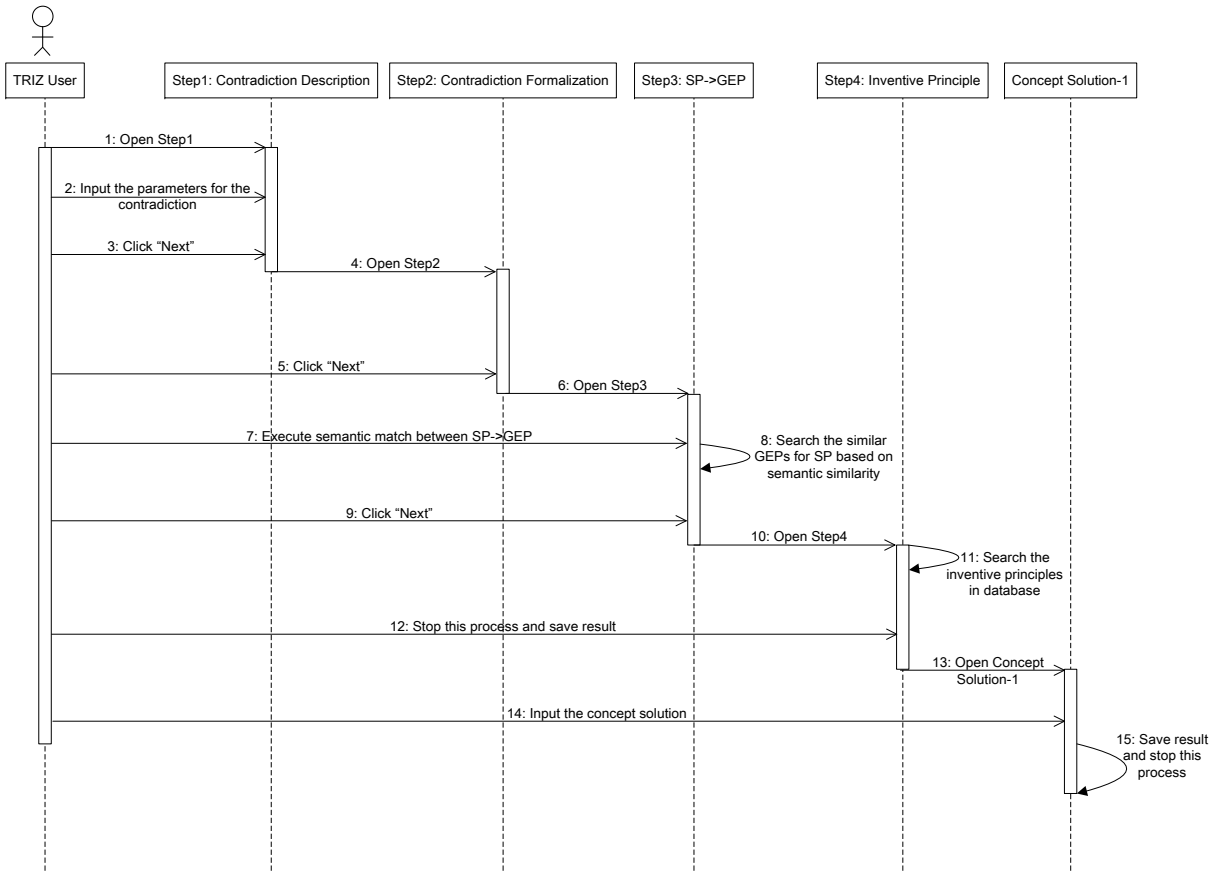


Figure 7.5: The sequence diagram for Step1->Step4.

series of inventive principles are obtained. The user can check the detailed information of each inventive principle.

– **Concept Solution** After obtaining the appropriate inventive principle, the user can stop this process, design concept solution, and store the related information from Step1 to Step4.

- Step4->Step8: The search of heuristic abstract solutions. The sequence diagram for Step4->Step8 is shown in Figure 7.6.

  – **Step5: Semantic Inventive Standard** According to the specific problem, the user needs to select an appropriate inventive standard, which is the most similar one with the obtained inventive principle. The user can choose them

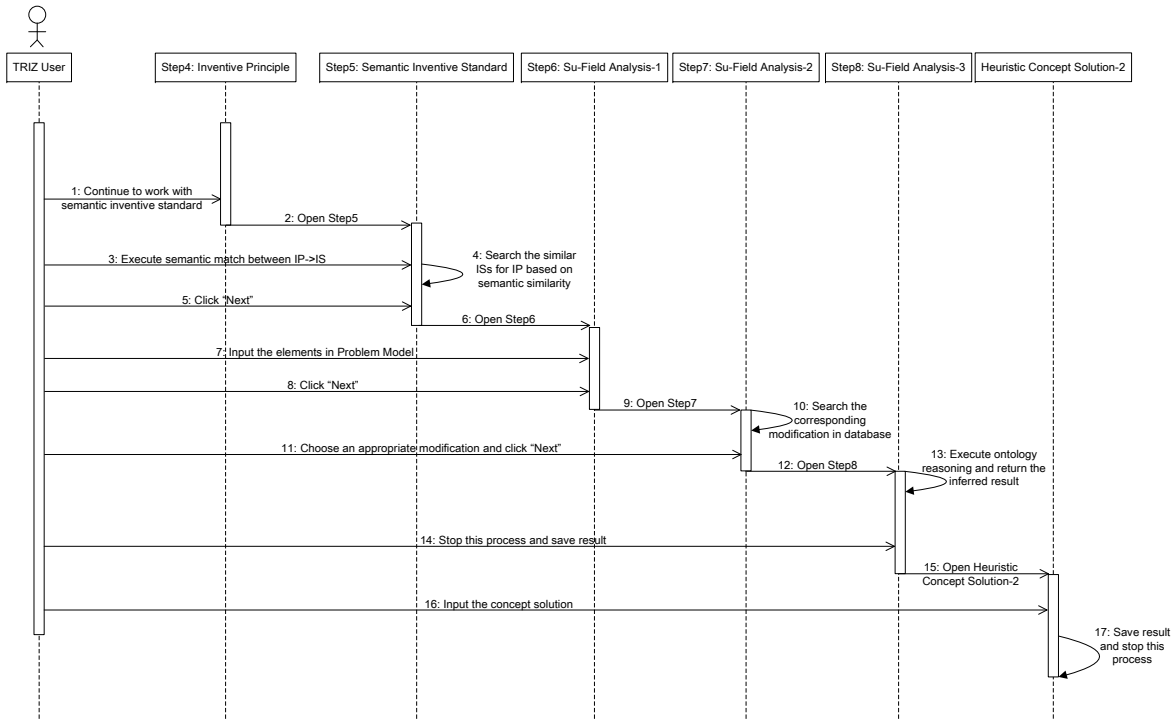


Figure 7.6: The sequence diagram for Step4->Step8.

manually, or with the help of semantic search, through which, the five most similar inventive standards can be obtained automatically.

- **Step6: Su-Field Analysis-1** As the first step of Su-Field analysis, the user is guided to provide the elements (substances and field) of the Problem Model.
- **Step7: Su-Field Analysis-2** According to the chosen inventive standard, the corresponding types of modification are returned. The user needs to select one from them.
- **Step8: Su-Field Analysis-3** Based on the information provided above, the ontology inference is executed on the inventive standards ontology to generate heuristic abstract solutions, which is displayed in the form of the Solution Model.
- **Heuristic Concept Solution** After obtaining the heuristic abstract solutions, the user can stop this process, design heuristic concept solution, and store the related information from Step1 to Step8.

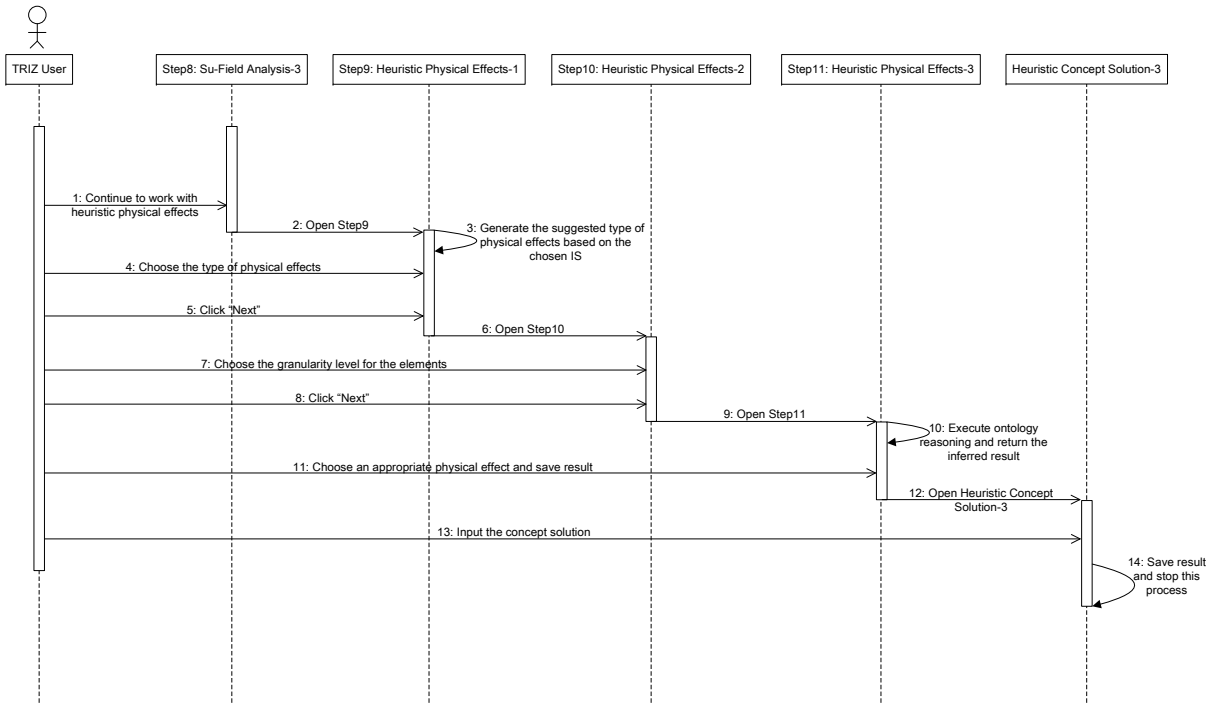


Figure 7.7: The sequence diagram for Step8->Step11.

- Step8->Step11: The use of heuristic physical effects. The sequence diagram for Step8->Step11 is shown in Figure 7.7.

  - **Step9: Heuristic Physical Effects-1** In this step, the type of physical effects is selected. Several suggested types of physical effects are returned based on the chosen inventive standard, and the user can select one from them.
  - **Step10: Heuristic Physical Effects-2** The user needs to provide the level of granularity for the element to be changed or added in the Solution Model.
  - **Step11: Heuristic Physical Effects-3** The ontology reasoning is executed on the inventive standards ontology and the physical effects ontology to generate the heuristic physical effects.
  - **Heuristic Concept Solution** After obtaining the heuristic abstract solutions and physical effects, the user finishes the whole process, designs heuristic concept solution, and stores the related information from Step1 to Step11.

7.4 IngeniousTRIZ: The Case of the "Diving Fin"

In order to verify the functions of the proposed prototype, the detailed process of using IngeniousTRIZ to solve the specific problem of the "Diving Fin" is introduced in this section.

7.4.1 The Problem

Even if the use of diving fins becomes popular, there are still some problems with them. The divers need to make a great effort to push water, which often makes them tired. Generally, the surface of the diving fin should be small for offering minimal resistance to water in order to minimize the effort of the diver and, at the same time, it should also be big in order to push water more efficiently.

In the following steps, the user is guided to solve this problem step by step in the prototype IngeniousTRIZ.

7.4.2 Abstract Solution from Inventive Principles (Step1 -> Step4)

In the phase from Step1 to Step4, the user works with the contradiction matrix and inventive principles to obtain the abstract solution.

Firstly, in order to launch the prototype, the user can easily double click the file IngeniousTRIZ.jar or use the command *Java -jar IngeniousTRIZ.jar* in the Command Prompt Window.

- **Step1 Contradiction Description**: As shown in Figure 7.8, the user needs to provide the parameters and values for the technical contradiction[1] in the case of the "Diving Fin".

- **Step2 Contradiction Formalization**: According to the information in Step1, the contradiction in the case of the "Diving Fin" is represented in the table form, which is much easier for the user to analyze the contradiction. As presented in Figure 7.9, there is a contradiction between "Kicking efficiency" and "Ease of use", that is, if

[1]The process of defining the technical contradiction for a specific case is introduced in the website: http://www.time-to-innovate.com/steps_matrix.

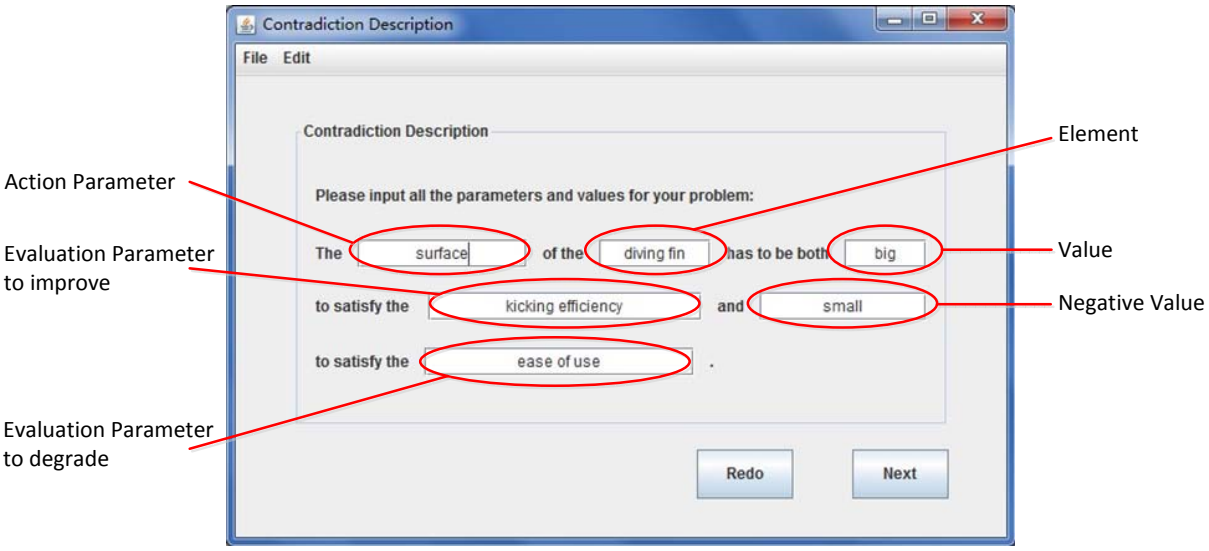


Figure 7.8: Step1: Contradiction description for the case of the "Diving Fin".

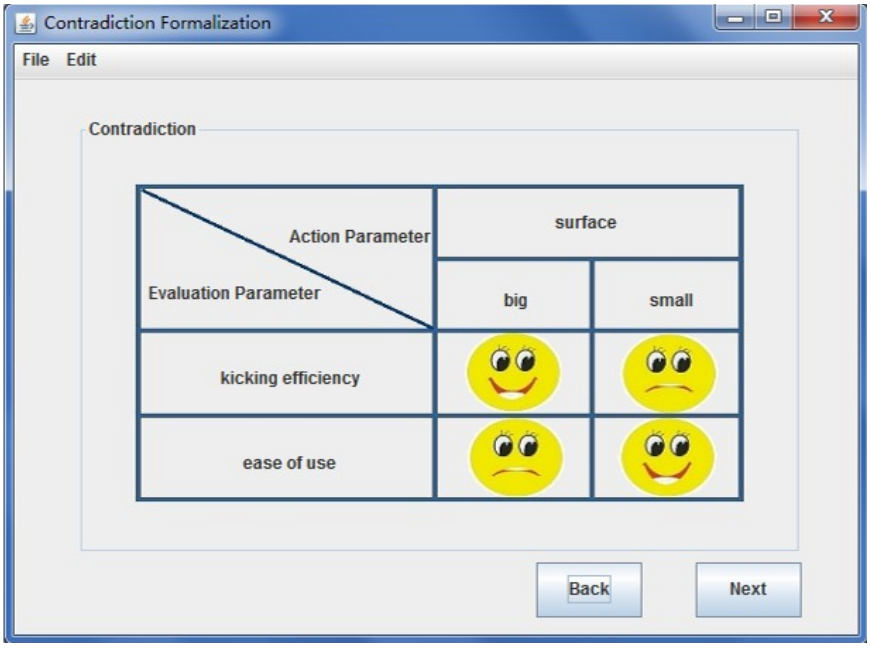


Figure 7.9: Step2: Contradiction formalization for the case of the "Diving Fin".

the surface of the diving fin is big, it is better for the kicking efficiency but worse for the ease of use, and vice versa.

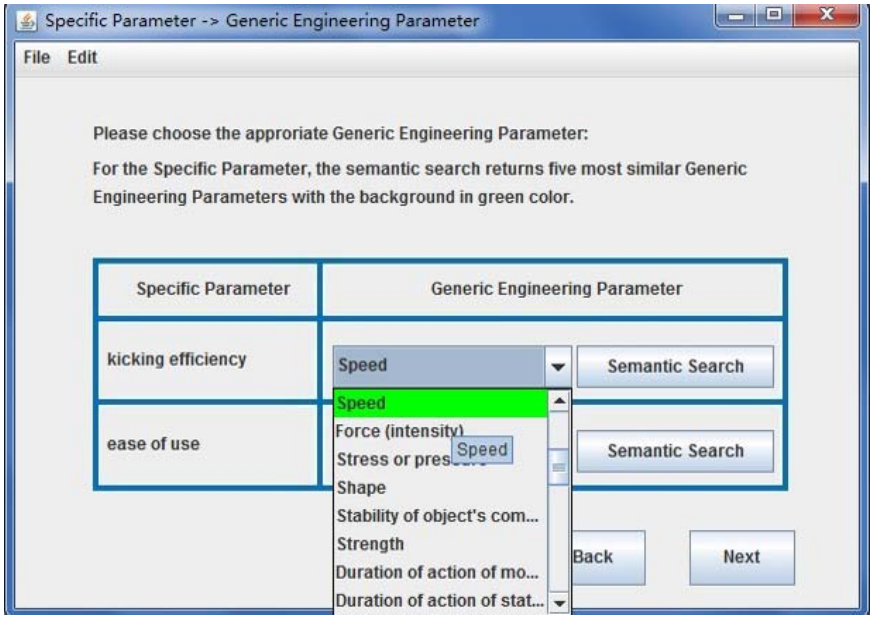
- **Step3 SP->GEP**: In order to use contradiction matrix, the specific parameters need to be matched with thirty-nine generic engineering parameters. Based on the methods proposed in Chapter 4, five most similar generic engineering parameters are obtained through the semantic search for each specific parameter. As shown in Figure 7.10 (a) and (b), the two specific parameters in the case of the "Diving Fin" correspond to five similar generic engineering parameters respectively:

  - SP1: kicking efficiency
    - GEP9: speed
    - GEP19: use of energy by moving object
    - GEP33: ease of operation
    - GEP34: ease of repair
    - GEP39: productivity
  - SP2: ease of use
    - GEP19: use of energy by moving object
    - GEP20: use of energy by stationary object
    - GEP32: ease of manufacture
    - GEP33: ease of operation
    - GEP34: ease of repair

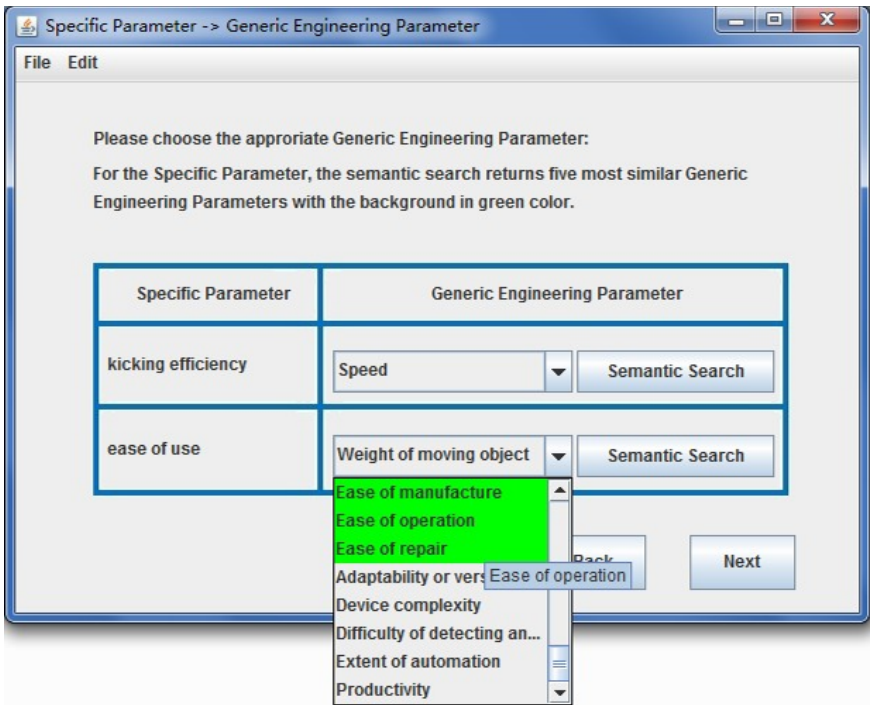
  Taking into account the characteristics of this case, "SP1: kicking efficiency" is matched with "GEP19: use of energy by moving object" while "SP2: ease of use" is matched with "GEP33: ease of operation".

- **Step4 Inventive Principle**: In this step, a series of inventive principles are obtained according to the two selected generic engineering parameters in Step3. For this case, 4 inventive principles are displayed as shown in Figure 7.11:

  - *Inventive Principle 32*: Color changes
    (a) Change the color of an object or its external environment.
    (b) Change the transparency of an object or its external environment.
  - *Inventive Principle 28*: Mechanics substitution



(a) SP1 (kicking efficiency)



(b) SP2 (ease of use)

Figure 7.10: Step3: The semantic match between SPs and GEPs.

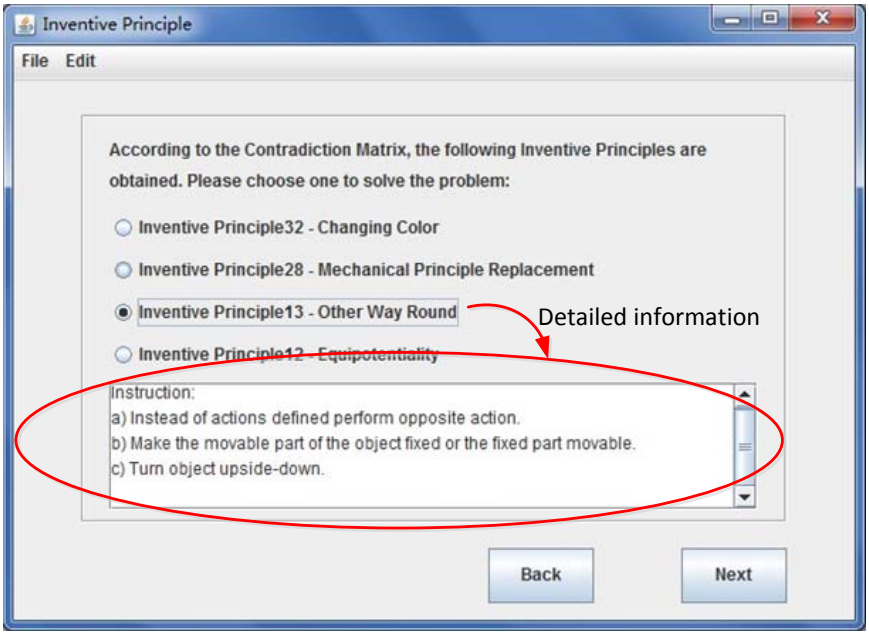


Figure 7.11: Step4: The obtained inventive principles for the case of the "Diving Fin".

(a) Replace a mechanical means with a sensory (optical, acoustic, taste or smell) means.

(b) Use electric, magnetic and electromagnetic fields to interact with the object.

(c) Change from static to movable fields, from unstructured fields to those having structure.

(d) Use fields in conjunction with field-activated (e.g. ferromagnetic) particles.

– *Inventive Principle 13*: "The other way round"

(a) Invert the action(s) used to solve the problem (e.g. instead of cooling an object, heat it).

(b) Make movable parts (or the external environment) fixed, and fixed parts movable).

(c) Turn the object (or process) "upside down".

– *Inventive Principle 12*: Equipotentiality

(a) In a potential field, limit position changes (e.g. change operating conditions to eliminate the need to raise or lower objects in a gravity field).

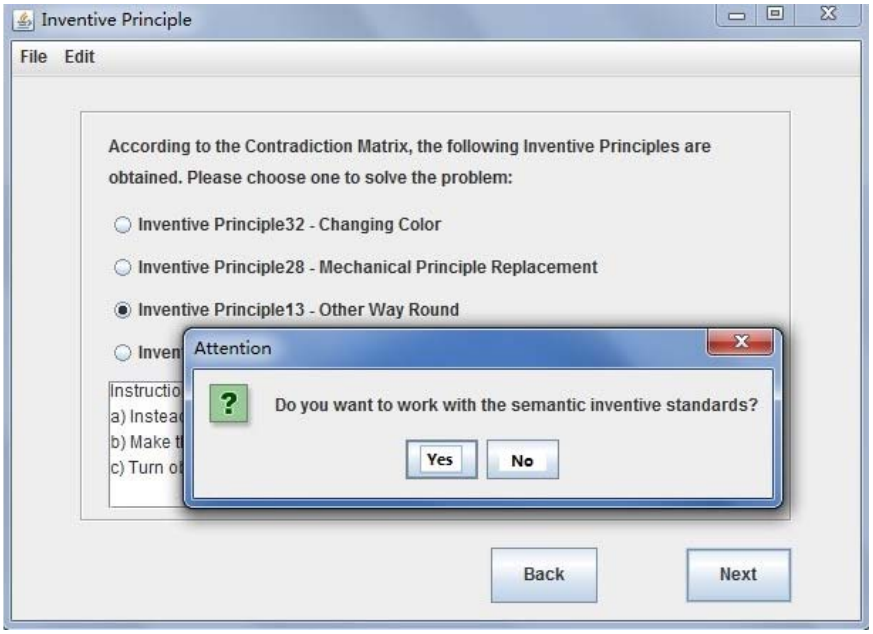


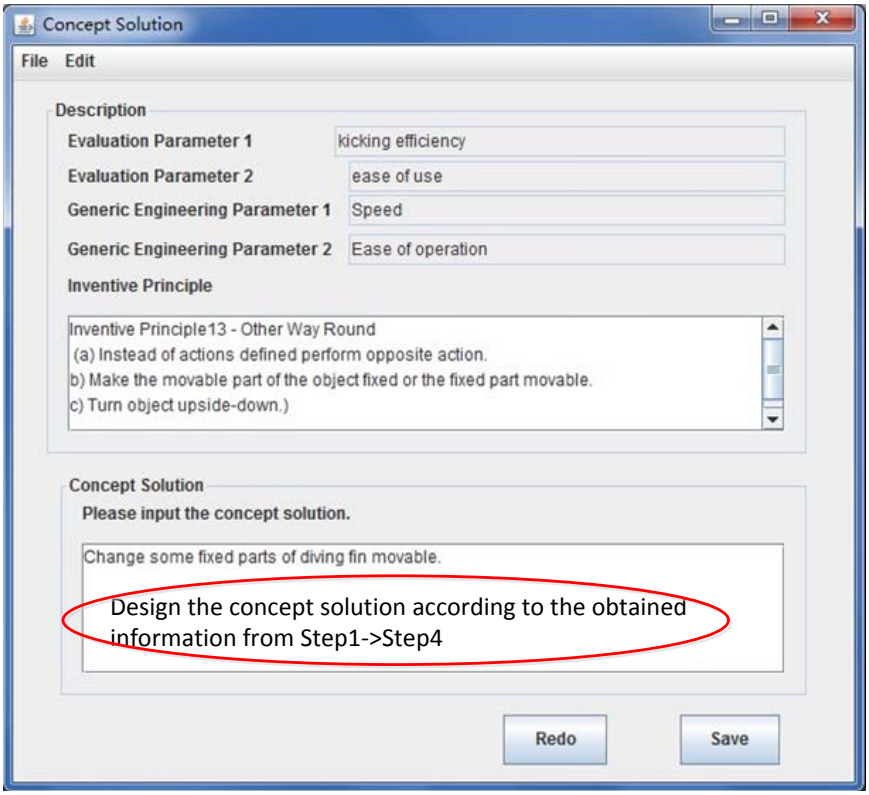
Figure 7.12: Step4: The message box.

Then the user needs to choose one from them, and in this case, for example, *Inventive Principle 13* is selected, and the detailed description is displayed in the text box. Finally, when click the button "Next", a message box is launched to ask whether the user continues to work with semantic inventive standards or not, as shown in Figure 7.12. If the user choose "Yes", Step5 is launched, and or else, the window "Concept Solution" to save the results from Step1 to Step4 is launched.

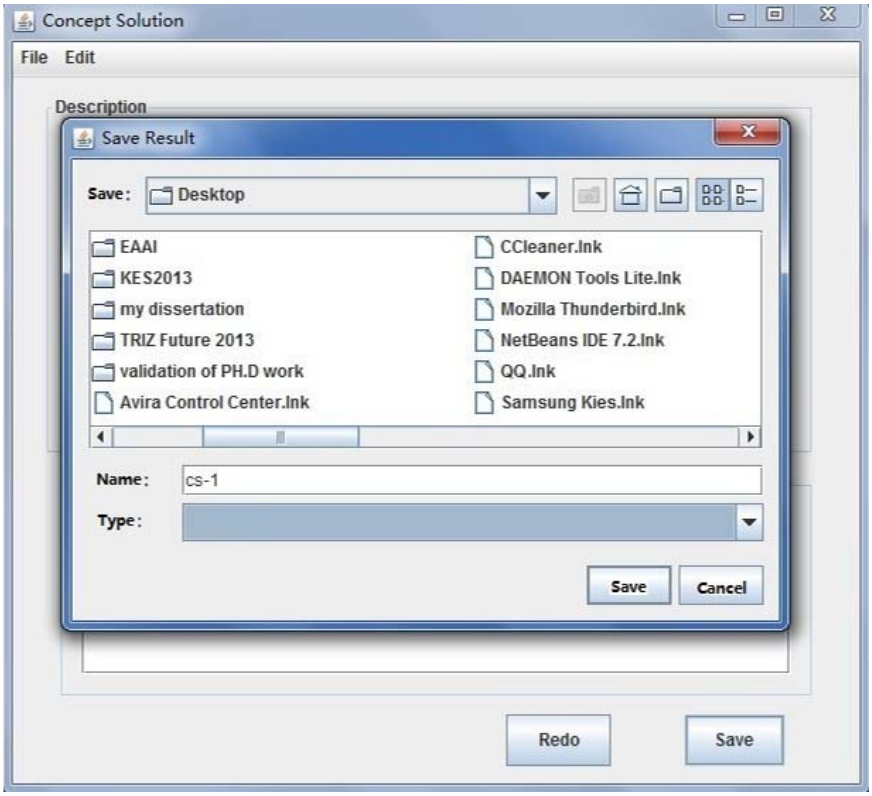
- **Concept Solution**: Based on the information obtained above, the user can design concept solution, for example, "change some fixed parts of diving fin movable", as shown in Figure 7.13 (a), and then save all the results in a document, as described in Figure 7.13 (b).

7.4.3 Heuristic Abstract Solution from Inventive Standards (Step4 -> Step8)

In the phase from Step4 to Step8, the user is guided to work with semantic inventive standards based on the chosen inventive principles.



(a) To design



(b) To save

Figure 7.13: Concept Solution for the case of the "Diving Fin".

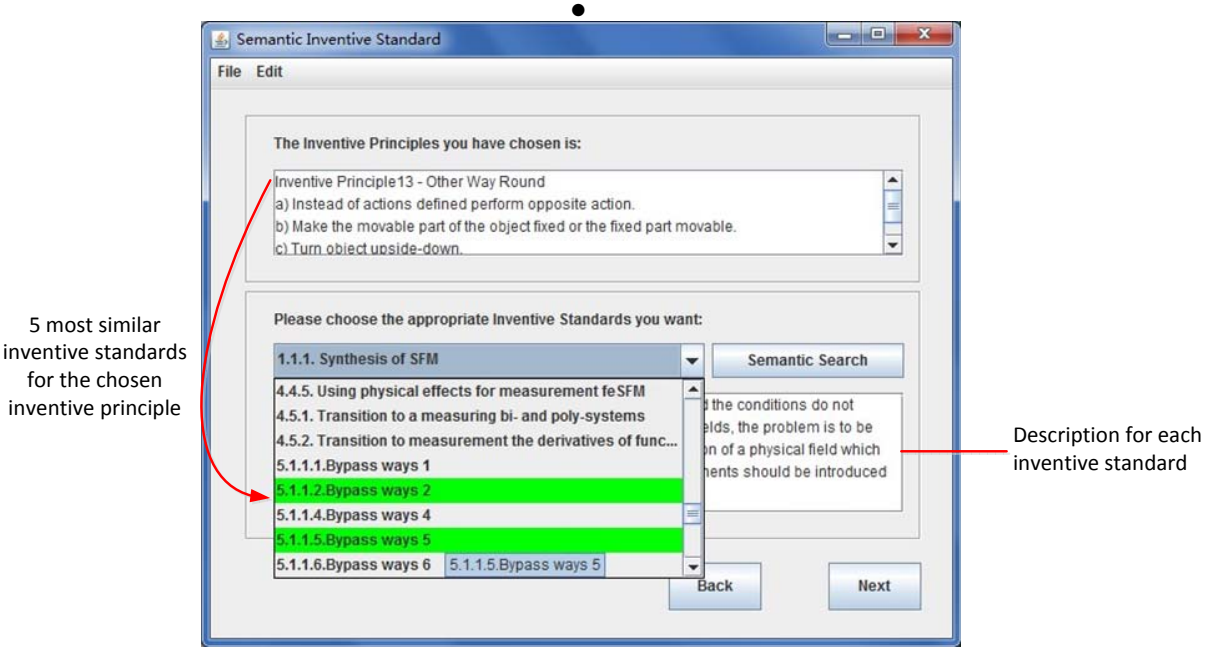


Figure 7.14: Step5: Semantic inventive standards for the case of the "Diving Fin".

Step5 Semantic Inventive Standard: As shown in Figure 7.14, the chosen inventive principle is firstly displayed, and the user needs to choose its most similar inventive standard for the case of the "Diving Fin". The methods proposed in Chapter 5 can facilitate this process, based on which five most similar inventive standards are returned. In this case, the five most similar inventive standards for *Inventive Principle 13* are:

- *Inventive Standard 1.1.6*: Minimum mode of action
- *Inventive Standard 1.1.7*: Maximum mode of action
- *Inventive Standard 4.3.3*: Using resonance oscillation of attached object
- *Inventive Standard 5.1.1.2*: Bypass ways 2
- *Inventive Standard 5.1.1.5*: Bypass ways 5

In order to implement Su-Field analysis, the user needs to choose one from the obtained five similar inventive standards. In the case of the "Diving Fin", *Inventive Standard 5.1.1.5* is selected.

- **Step6 Su-Field Analysis-1**: With the help of the chosen inventive standard, the

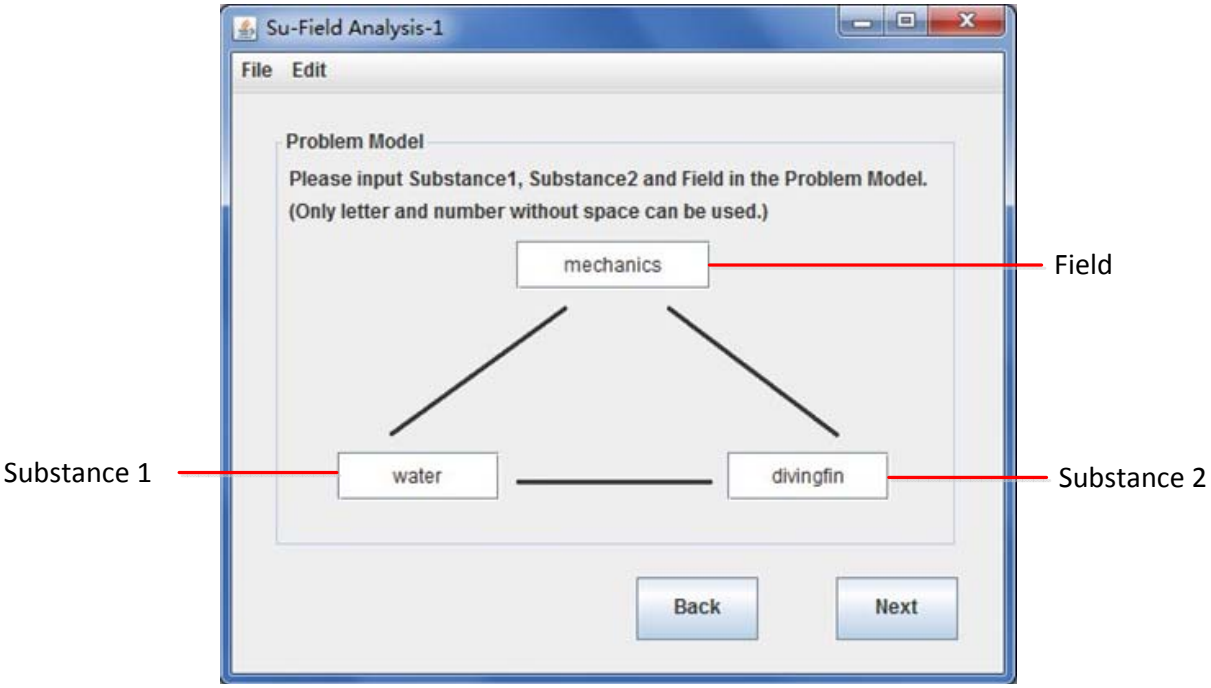
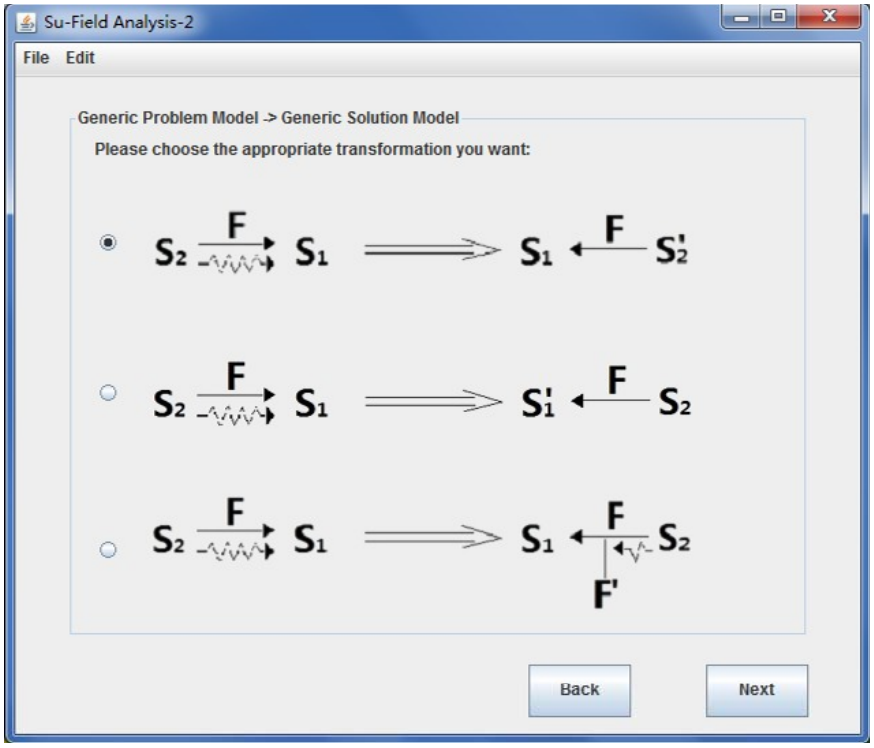


Figure 7.15: Step6: Su-Field Analysis-1 for the case of the "Diving Fin".

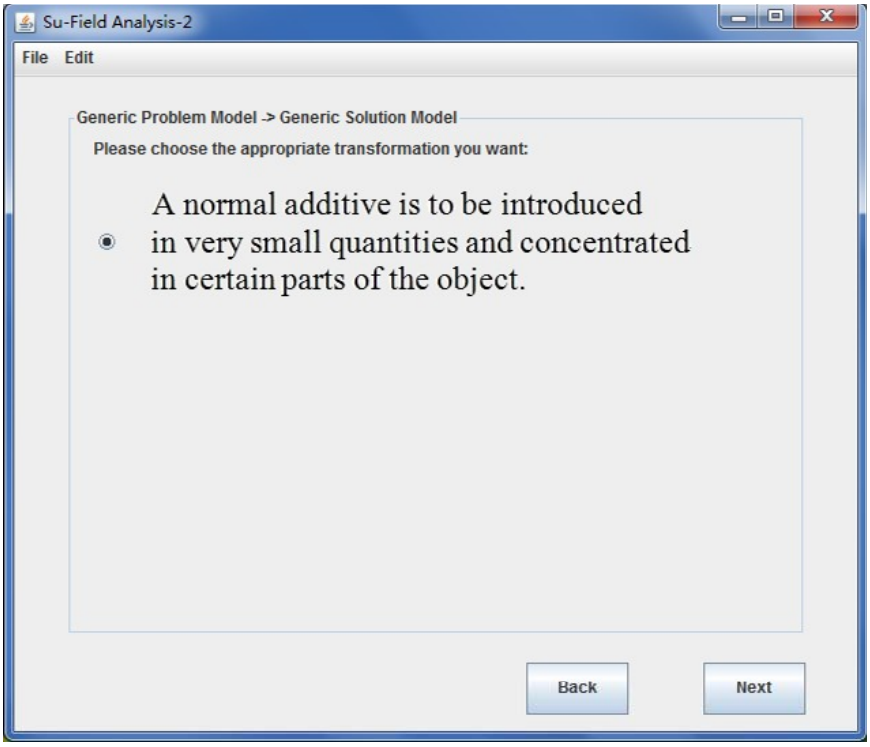
user is guided to implement Su-Field analysis from this step. As shown in Figure 7.15, the user needs to provide the basic information about the Problem Model, including two substances and a field. This information will be used to instantiate the inventive standards ontology in the process of ontology reasoning.

- **Step7 Su-Field Analysis-2**: In this step, several related types of transformation are obtained based on the chosen inventive standards, from which the user needs to choose appropriate one for the specific case. There are two ways of representing the transformation, that is, the transformation of Su-Field models and the text description, as shown in Figure 7.16 (a) and (b). In the case of the "Diving Fin", only one type of transformation - "A normal additive is to be introduced in very small quantities and concentrated in certain parts of the object." is obtained and used for the following ontology inference, as shown in Figure 7.16 (b).

  Step8 Su-Field Analysis-3: Based on the obtained information from Step5 to Step7, the ontology inference, as presented in Figure 7.17 (a) and (b), is executed to generate heuristic abstract solutions, which are stored in a new-built ontology.

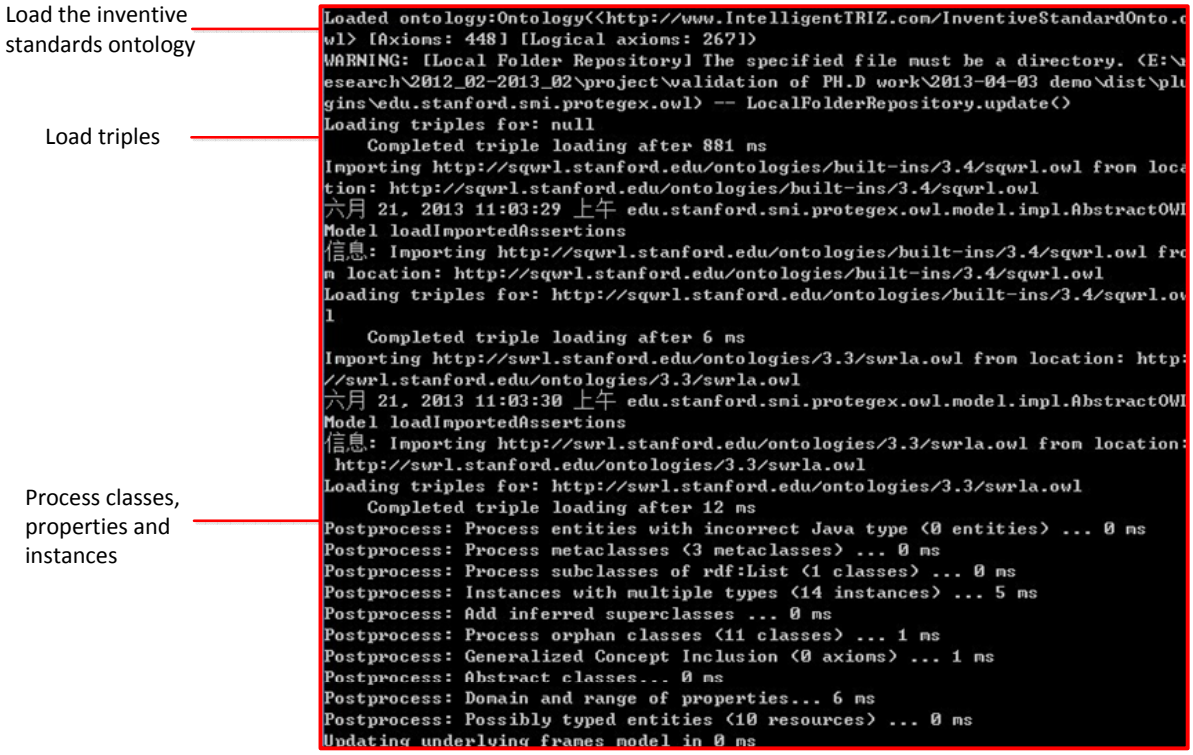


(a) The transformation: Su-Field models

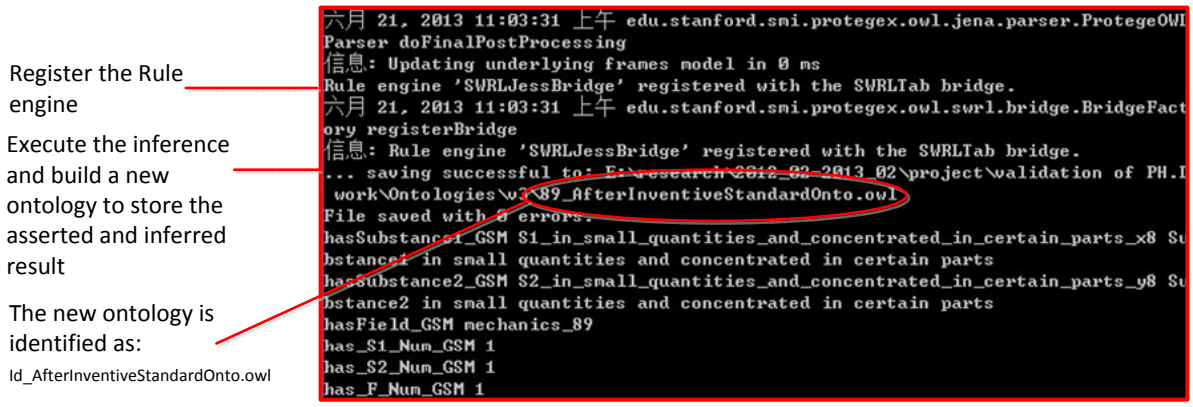


(b) The transformation: Text description (The case of the "Diving Fin")

Figure 7.16: Step7: Su-Field Analysis-2 for the case of the "Diving Fin".



(a) Loading the inventive standards ontology



(b) Ontology reasoning

Figure 7.17: The ontology inference for the heuristic abstract solution in the case of the "Diving Fin".

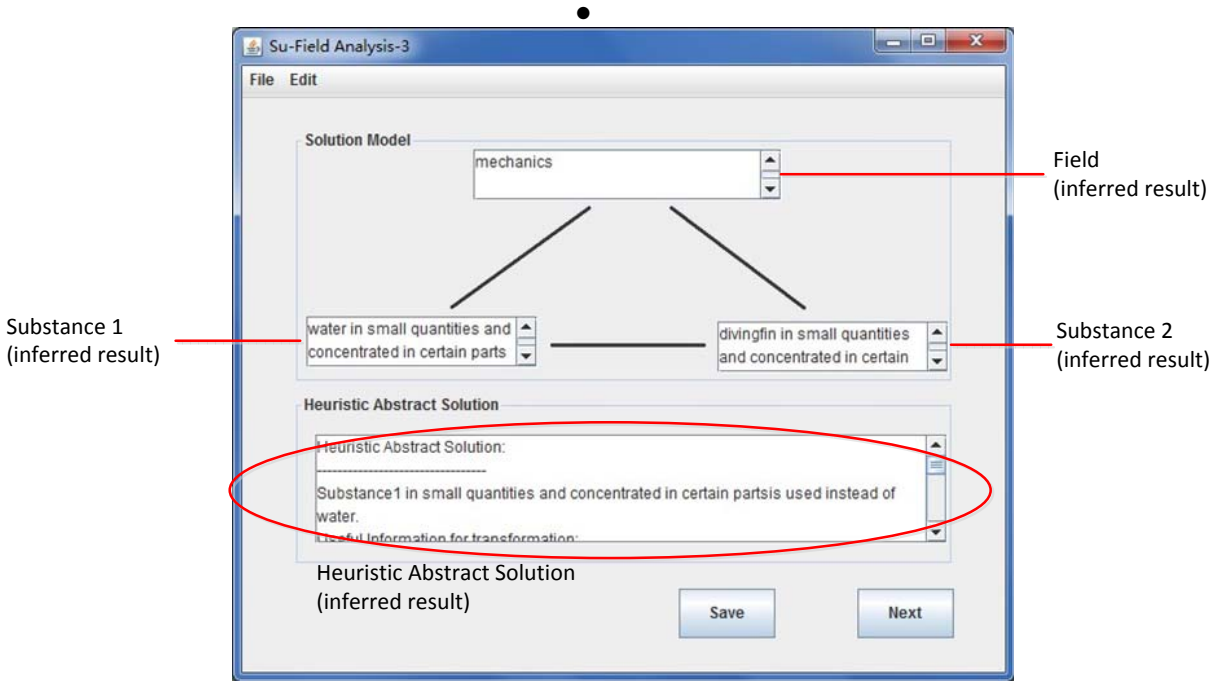


Figure 7.18: Step8: Su-Field Analysis-3 for the case of the "Diving Fin".

As shown in Figure 7.18, for the case of the "Diving Fin", the elements (substances and field) in the Solution Model are generated after ontology inference. At the same time, the heuristic abstract solutions are created automatically in terms of two aspects:

- The modification in Substance1-"Water": A normal additive in very small quantities and concentrated in certain parts of "Water" is to be introduced.
- The modification in Substance2-"Diving Fin": A normal additive in very small quantities and concentrated in certain parts of "Diving Fin" is to be introduced.

Assuming that water cannot be modified in this case, we need to modify the diving fin by bringing in some additives. There are several kinds of modifications of the diving fin, for example, the change of its structure and its shape by using liquid or gas. In order to modify the diving fin, the user can choose to do it manually by closing this prototype, and also can continue to work with the heuristic physical effects, which can help the user to instantiate the Solution Model in real life. As a result, when the user clicks the button "Next", a message box is launched to ask

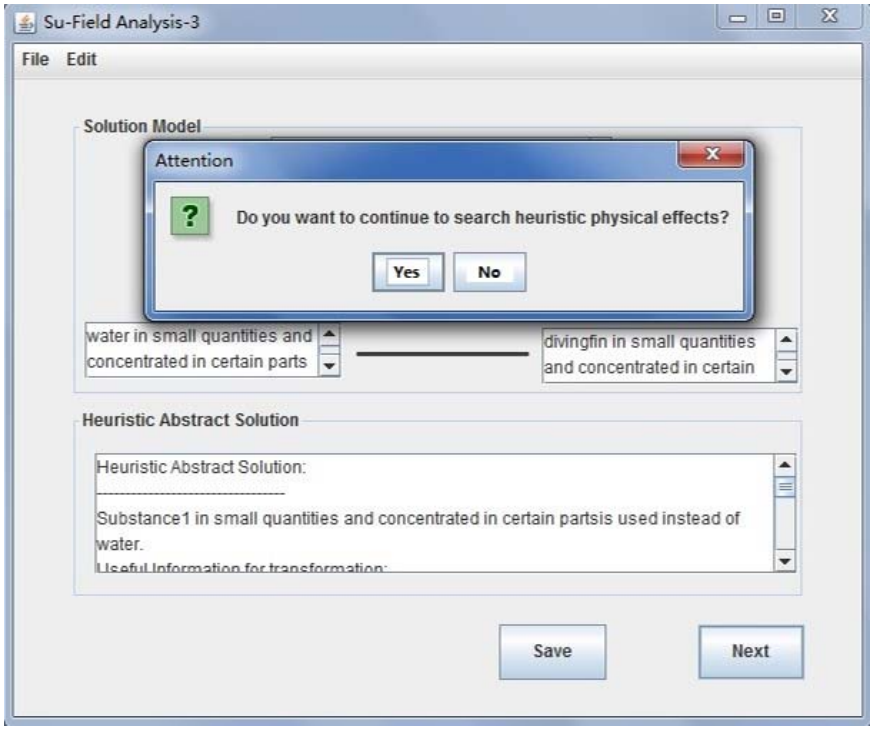


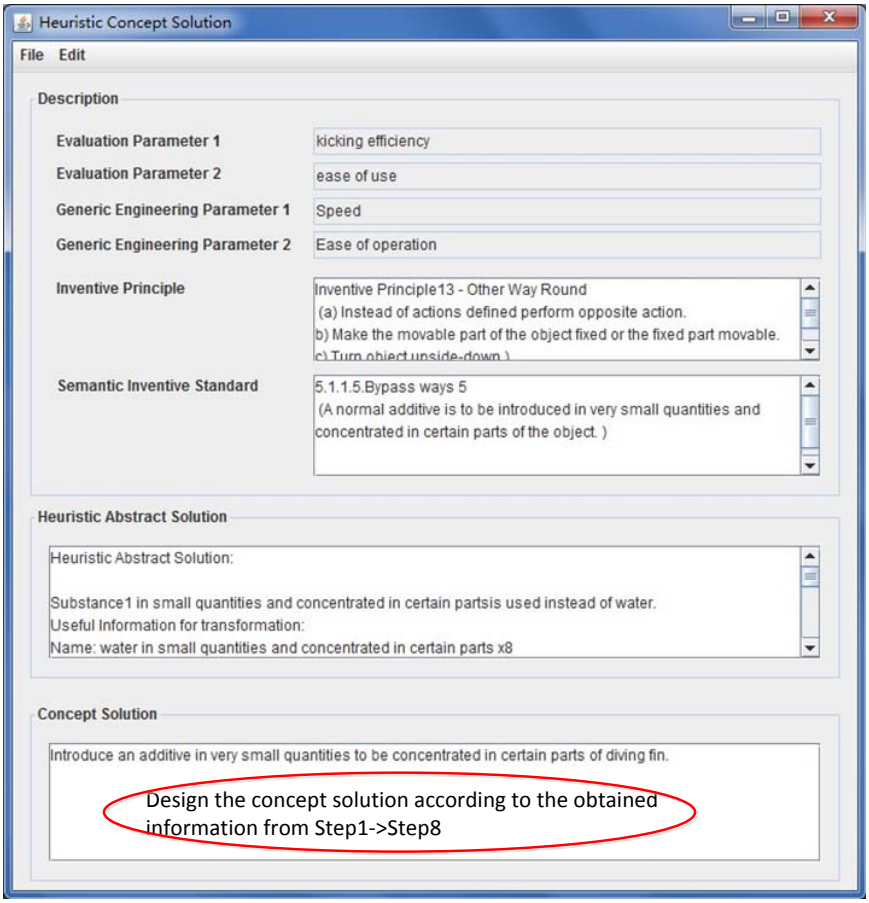
Figure 7.19: Step8: The message box.

whether the user continues to work with heuristic physical effects or not, as shown in Figure 7.19. If the user choose "Yes", Step9 is launched, and or else, the window "Heuristic Concept Solution" to save the results from Step1->Step8 is launched.

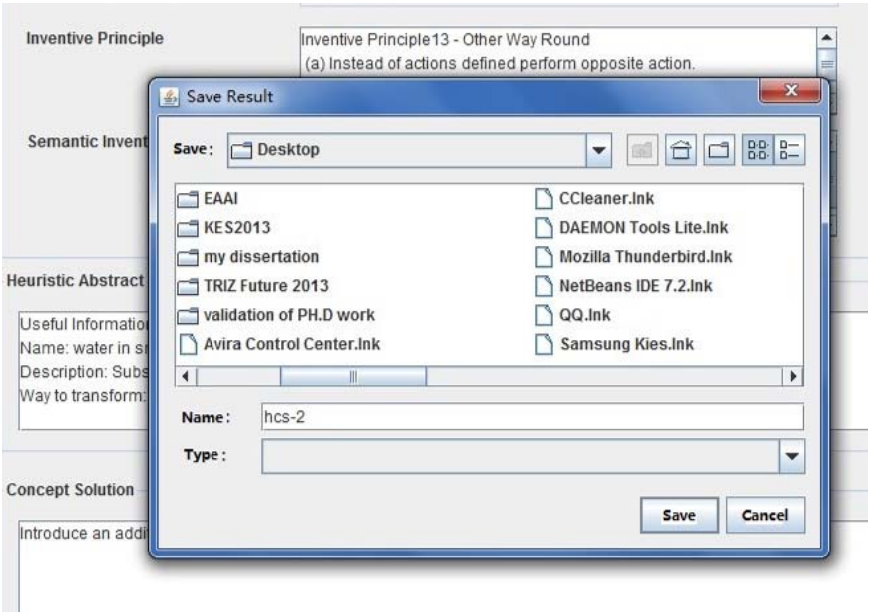
- **Heuristic Concept Solution**: According to the obtained heuristic abstract solutions, the user can design heuristic concept solution, such as "Introduce an additive in very small quantities to be concentrated in certain parts of the diving fins", as shown in Figure 7.20 (a), and then save all the results in a document, as depicted in Figure 7.20 (b).

7.4.4 The Search of Heuristic Physical Effects (Step8-> Step11)

In the phase from Step8 to Step11, the user works with the heuristic physical effects to instantiate the obtained heuristic abstract solution in real life.



(a) To design



(b) To save

Figure 7.20: Heuristic Concept Solution for the case of the "Diving Fin".

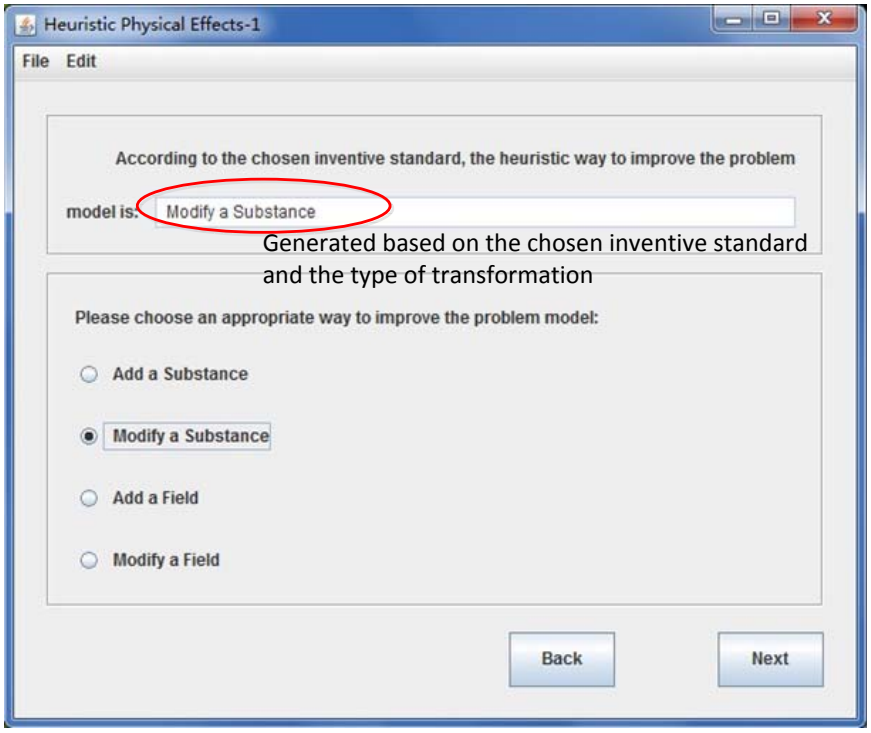
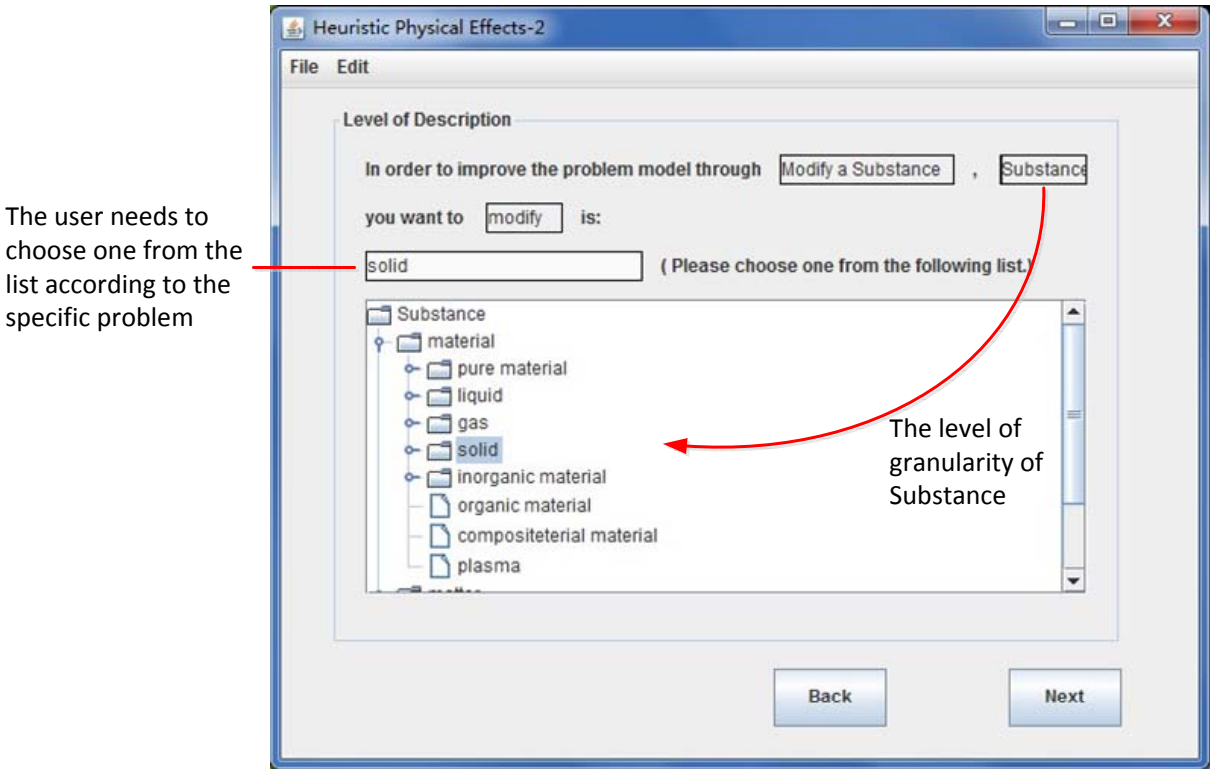


Figure 7.21: Step9: Heuristic Physical Effects-1 for the case of the "Diving Fin".

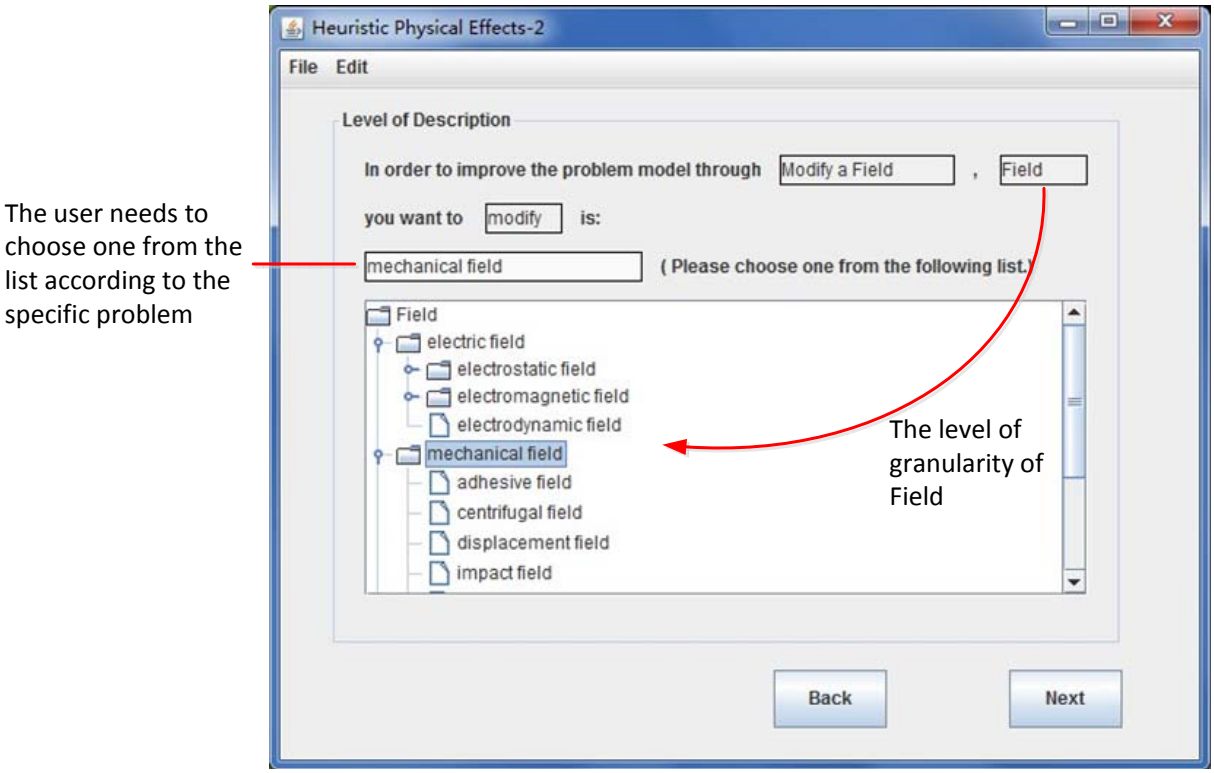
- **Step9 Heuristic Physical Effects-1**: In this step, four ways to improve the Problem Model are given, that is, "Add a Substance", "Modify a Substance", "Add a Field" and "Modify a Field", and the user needs to select appropriate one from them. According to the inventive standards chosen above, a heuristic ways is indicated to facilitate this process. For the case of the "Diving Fin", as presented in Figure 7.21, "Modify a Substance" is suggested to modify the Problem Model.

- **Step10 Heuristic Physical Effects-2**: In order to implement the ontology reasoning, the user also needs to provide the level of granularity for the element (substance or field) to be modified. Figure 7.22 (a) and (b) show the level of granularity for substance and field respectively, and this case corresponds to the first one.

- **Step11 Heuristic Physical Effects-3**: Based on the obtained information from Step9 and Step10, the ontology inference, presented in Figure 7.23 (a), is executed to generate the heuristic physical effects, which are stored in a new-built ontology.



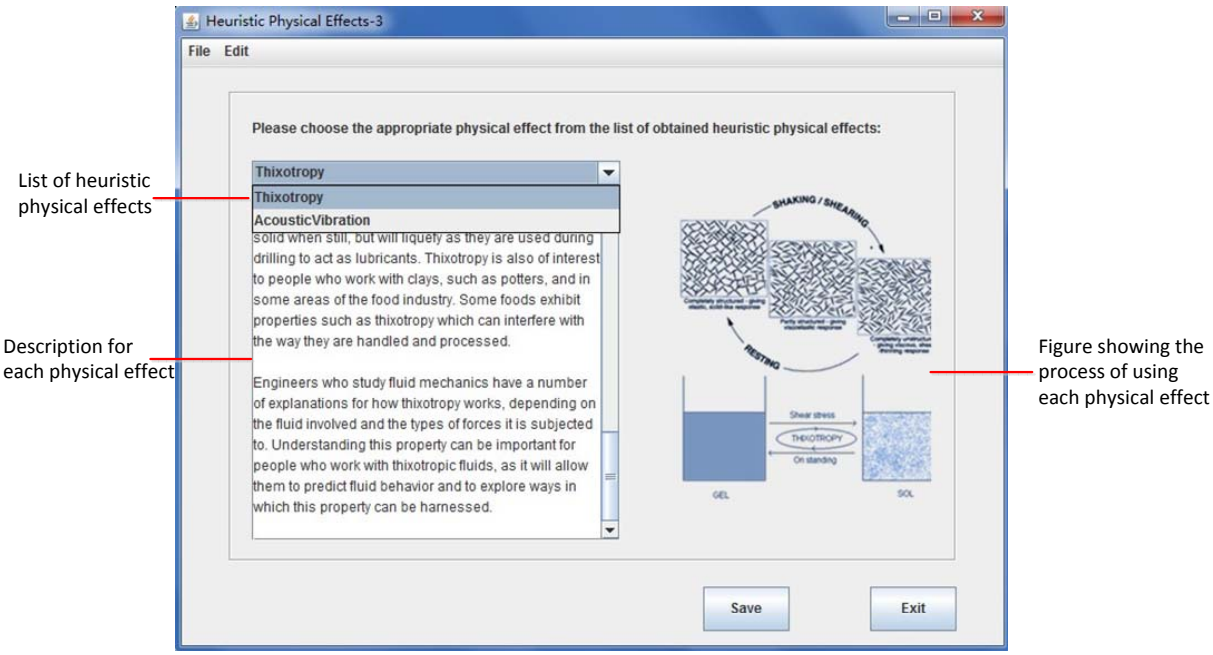
(a) The level of granularity for substance



(b) The level of granularity for field

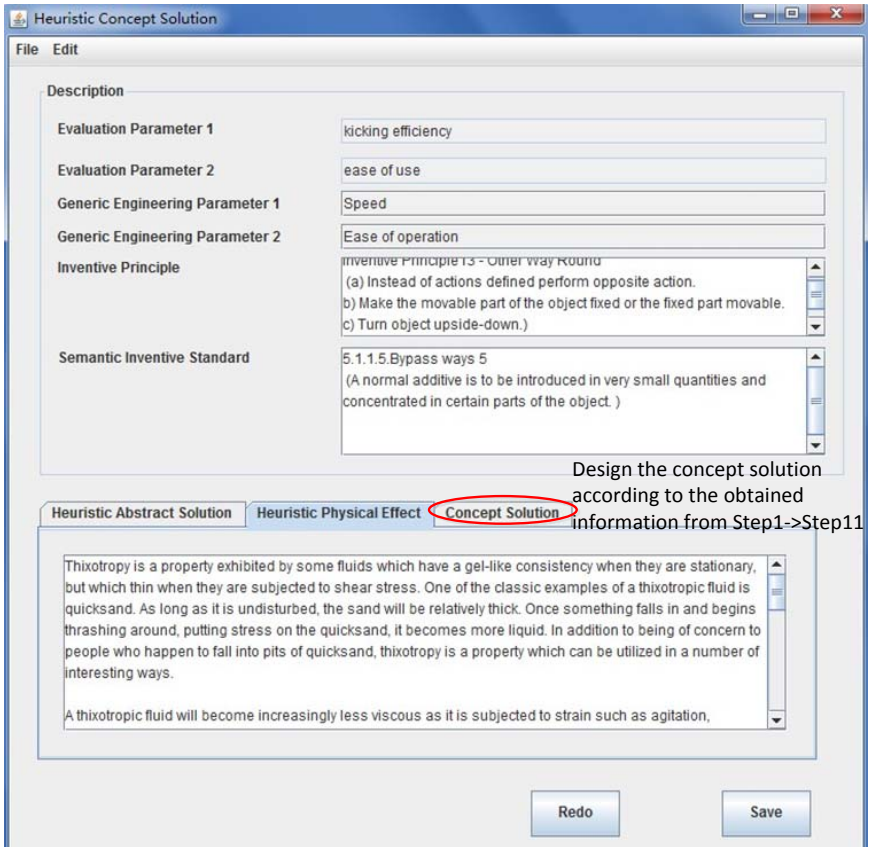
Figure 7.22: Step10: Heuristic Physical Effects-2 for the case of the "Diving Fin".

(a) The ontology inference based on the physical effects ontology

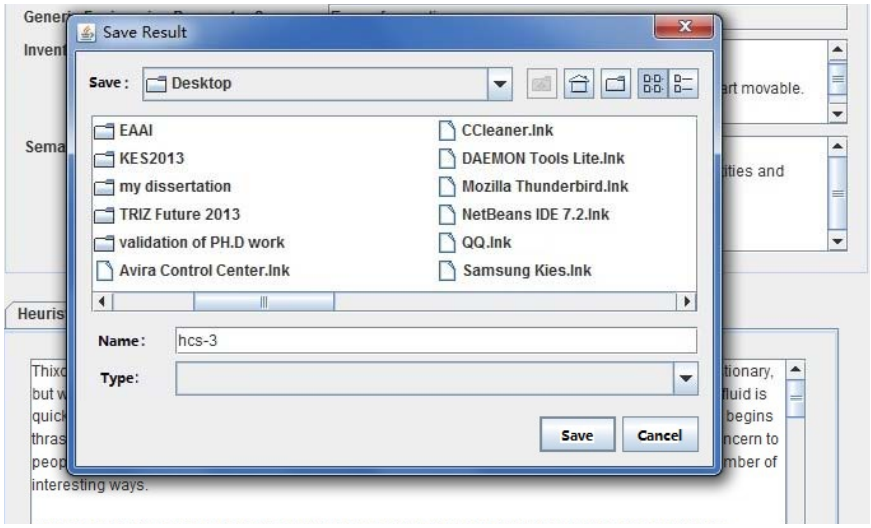


(b) The detailed information for the heuristic physical effects

Figure 7.23: Step11: Heuristic Physical Effects-3 for the case of the "Diving Fin".



(a) To design



(b) To save

Figure 7.24: Heuristic Concept Solution for the case of the "Diving Fin".

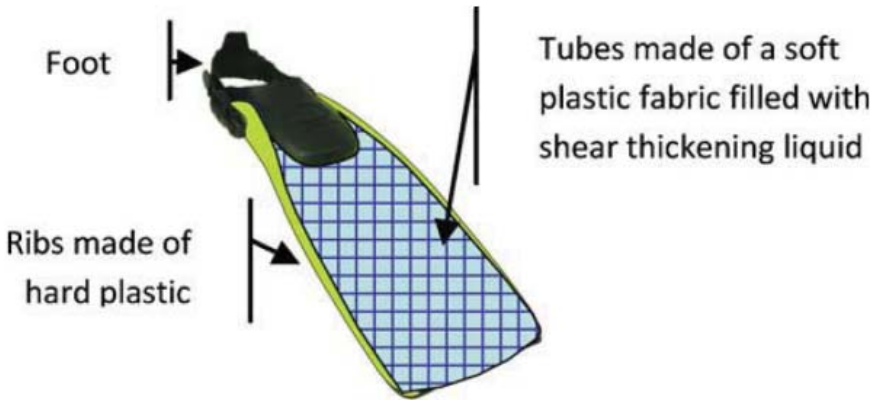


Figure 7.25: The concept of tubular shear thickening fins.

The user can select an appropriate physical effect for solving the specific problem. As shown in Figure 7.23 (b), for the case of the "Diving Fin", two physical effects- "Thixotropy" and "Acoustic Vibrations" are obtained[1], and their detailed description is displayed in the text box with images to show the functional process.

Assuming that the physical effect "Thixotropy" is chosen to solve this problem, and the user can easily save the results by clicking the button "Save", and or else, he/she can exit the prototype with out saving the results by clicking the button "Exit".

- **Heuristic Concept Solution**: As presented in Figure 7.24 (a), the user can design and explore the heuristic concept solution according to the obtained heuristic abstract solution and physical effect as following:

  Heuristic Concept Solution: The effect "Thixotropy" is the property of certain gels or fluids that are thick (viscous) under normal conditions, but flow (become thin, less viscous) over time when shaken, agitated, or otherwise stressed. Taking this into account, we use a shear thickening liquid to modify the diving fin, and propose the concept of tubular shear thickening fins as shown in Figure 7.25. This concept is a

[1]In this application, 20 physical effects are used to test the related functions of the ontology inference.

truly inventive concept since there are no-existing widely distributed and produced products in industry based on this principle (only prototypes, patented fabrics, dampers in the truck industry, etc.).

Finally, all the obtained results are saved in a document, as depicted in Figure 7.24 (b).

7.5 Summary

In order to verify the methods proposed in Chapters 3, 4, 5 and 6, the prototype IngeniousTRIZ was developed. In this chapter, firstly a general introduction about the development of this prototype is given. Then the environment and tools for its development are illustrated in detail. Finally, the framework of the prototype is introduced and taking the case of the "Diving Fin" as an example, the detailed process of using IngeniousTRIZ to solve problem is explained step by step.

The implementation of the prototype IngeniousTRIZ, not only verifies the proposed methods in this research, but also stimulates the development of a new research area in the field of inventive design, that is, the use of an automatic ontology-based mechanism for solving inventive problems.

Bibliography

[1] C. Zanni-Merk, D. Cavallucci, and F. Rousselot, "An ontological basis for computer aided innovation," *Computers in Industry*, vol. 60, pp. 563–574, Oct. 2009.

[2] ——, "Use of formal ontologies as a foundation for inventive design studies," *Computers in Industry*, vol. 62, pp. 323–336, Apr. 2011.

Chapter 8

Conclusions and Perspectives

In recent decades, modern innovation theories and methods proposed various knowledge sources to solve different types of inventive problems. The TRIZ knowledge sources are all built independently of the specific application field, and their levels of abstraction are very different, resulting in an additional difficulty of interpretation.

These research works aim at facilitating and automating the process of solving inventive problems based on semantic similarity and ontology techniques. At first, the TRIZ knowledge sources are formalized based on ontologies, such that heuristic inference can be executed to search for specific solutions. Then, methods for calculating semantic similarity are explored to search and define the missing links among the TRIZ knowledge sources.

In order to solve inventive problems, the TRIZ user firstly chooses a TRIZ knowledge source to work for an abstract solution. Then, the items of other knowledge sources, which are similar with the selected items of the first knowledge source, are obtained based on semantic similarity calculated in advance. With the help of these similar items and the heuristic physical effects, other specific solutions are returned through ontology inference.

The prototype "IngeniousTRIZ" is developed based on semantic similarity and ontology inference to support this automatic process of solving inventive problems.

In the following sections, the research works of each chapter is firstly summarized, and then, the major contributions of our research are presented. Finally, in order to improve this research in the future, several prospective points are proposed.

8.1 Conclusions

In Chapter 1, the background of this research has been firstly introduced, that is, the proposed TRIZ knowledge sources and models were built at different levels of abstraction and their use required extensive knowledge about different engineering domains. In order to facilitate the use of TRIZ, an Automatic Ontology-based Mechanism for Solving Inventive Problems has been proposed. Then, the motivations and research scope of this thesis have been explained in detail. Finally, the structure of this thesis has been presented.

Chapter 2 has presented a literature review on TRIZ and the comprehensive knowledge about Ontology-based Knowledge Modeling and Ontology Reasoning. In the TRIZ part, we have presented the definition of TRIZ, illustrated the application steps of TRIZ solving inventive problem and introduced the basic tools of TRIZ, including the contradiction, the TRIZ knowledge sources and the physical-chemical-geometrical effects. In the parts of Ontology Modeling and Reasoning, an overview of their basic methods and concepts has been provided, and then the current situation of their applications in the field of inventive design has been given.

In Chapter 3, two methods have been presented, which were used to match the parameters in the use of inventive principles and to define the missing links among the TRIZ knowledge sources respectively. Firstly, a literature review about semantic similarity has been presented, and then the related methods of calculating semantic similarity in the domain of inventive design have been given. Finally, the proposed methods in this thesis have been introduced in detail.

In Chapter 4, in order to facilitate the use of Altshuller's contradiction matrix in an inventive design study, the methods of matching specific parameters and generic engineering parameters based on semantic similarity and case-based reasoning have been introduced in detail. The inventive principles ontology has been firstly built to formalize the process of using the contradiction matrix, and then according to Chapter 3, the methods of calculating semantic similarity have been used to fill the gap between specific parameters and generic engineering parameters.

In order to improve the performance of this method, the results from previous projects have been returned to users as suggested solutions according to a case-based reasoning approach. As we can observe from our first attempt to test the developed methodology,

some relevant potential associations between a specific term and one or several of the thirty-nine generic engineering parameters have been proposed to users. The improved method has also benefited from previous experiences and therefore offered a constant improvement due to CBR techniques.

However, the resolution of inventive problems only with forty inventive principles can not always provide comprehensive and efficient recommendations, and it is necessary to apply other TRIZ knowledge sources, such as seventy-six inventive standards.

As a result, Chapter 5 has introduced a method of searching heuristic abstract solutions from different knowledge sources based on semantic similarity and ontology reasoning. Considering that all the TRIZ knowledge sources are described in short-text, the missing links among the TRIZ knowledge sources have been defined based on the methods of calculating short-text semantic similarity, which have been introduced in Chapter 3. Meanwhile, the ontology reasoning mechanism deployed on Protégé and Jess, has been used to provide heuristic solutions dynamically for TRIZ users. Firstly, TRIZ users started solving inventive problems with the TRIZ knowledge source of their choice to obtain an abstract solution, and then, according to the selected items of that first knowledge source, the similar items of other knowledge sources were obtained based on the semantic similarity calculated in advance. With the help of these similar items, heuristic abstract solutions have been returned through the ontology inference on Jess.

As the last step of solving inventive problems, especially for Su-Field analysis, the obtained abstract solutions need to be instantiated in real life with the help of physical effects, that is, the physical effects compatible with the context of the specific problem should be chosen to assist the users to instantiate the abstract solution. However, the physical effects and the specific problems are built at different levels of abstraction, and it is difficult for the users to choose, that is, given a certain function, too many physical effects are chosen while with the detailed context of the problem, no physical effect is returned.

In order to facilitate the use of physical effects, Chapter 6 has firstly proposed a new way of representing physical effects using the change of two states, that is, the couple of two states before and after applying them. Then, knowledge about using physical effects has been formalized based on ontology, and constraint knowledge, such as the conditions to use each kind of physical effects, has been formalized by using the rule language. Finally, the reasoning process of using physical effects has been performed with the support of the

Jess rule engine.

In order to verify the methods proposed in Chapter 3, 4, 5 and 6, the prototype IngeniousTRIZ has been developed. In Chapter 7, firstly a general introduction about the development of this prototype has been given. Then the environment and tools for the development and the framework of the prototype have been introduced in detail. At last, taking the case of the "Diving Fin" as an example, the detailed process of using IngeniousTRIZ to solve the inventive problem has been explained step by step.

8.2 Contributions

By solving the research problems described in Section 1.2, an automatic ontology-based mechanism for solving inventive problems is proposed based on semantic similarity and ontology techniques in this thesis. The contributions of this thesis can be summarized into three aspects as following:

Firstly, the TRIZ knowledge sources are formalized based on ontology modeling. Through the detailed analysis for the process of using each knowledge source, the related concepts and their relations are defined in the ontologies. In comparison to traditional commercial software, the formalization based on ontologies provides conceptual resources for knowledge based systems (KBS) and makes it possible to automate the process of solving inventive problems by using ontology reasoning. It also permits the tracking of different applications to study and compare them based on ontologies, and, in this way, the improvement of the whole methodology.

Secondly, different methods for calculating semantic similarity are proposed to facilitate the process of solving inventive problems. On the one hand, in order to search appropriate inventive principles in the contradiction matrix, the match between specific parameters and generic engineering parameters is implemented automatically based on semantic similarity, through which the efficiency and accuracy of choosing inventive principles to solve problems have been improved greatly. On the other hand, the missing links among the TRIZ knowledge sources are defined based on semantic similarity. Compared with the manual work, it uses semantic methods to compare the different abstract models or solutions and gives an additional analysis to eliminate the various interpretations of different TRIZ users.

Thirdly, in order to facilitate the resolution of inventive problems with TRIZ, an automatic mechanism of solving inventive problems is proposed based on semantic similarity, ontology modeling and ontology inference. In this mechanism, the TRIZ users start solving an inventive problem with the TRIZ knowledge source of their choice to obtain an abstract solution. According to the selected items of that first knowledge source, the similar items of other knowledge sources are obtained based on the semantic similarity calculated in advance. With the help of these similar items and the heuristic physical effects, other specific solutions are returned through ontology inference.

For the industries today, this thesis presents a proposal for solving inventive problems with TRIZ in a different way from the existing approaches widely used in the domain of inventive design, since our research makes it possible to transform the solution obtained with one knowledge source to other that is obtained with another one that would not have been used by the TRIZ user, and therefore provides new options to solve inventive problems with diversified knowledge sources. By using this mechanism, on the one hand, more solutions for specific cases from different TRIZ knowledge sources can be obtained automatically, and so the efficiency of operating inventive practice in design and also in the Research and Development (R&D) department can be improved. On the other hand, the resolution with more than one TRIZ knowledge sources instead of a single source can yield to more stable and appropriate specific solutions, and so the risk of generating useless solutions could be reduced greatly.

Furthermore, with the help of this mechanism, it is not necessary for the designers in the industries to be trained for all the TRIZ knowledge sources. They can be trained to use a simple TRIZ knowledge source to solve inventive problems, and the heuristic solutions from other complicated knowledge sources can be obtained automatically through this mechanism. Consequently, it will increase the attractiveness for the companies to train and use TRIZ in the process of inventive design.

For the educational viewpoint, this research provides an intelligent support for solving inventive problems. An educator would argue our proposals by saying that easing the job of an engineer by automating part of his thinking process may result in a diminution of his potential skills acquisition. It is partly true, this is why we would not propose the systematic use of these methods if the goals were to increase engineering knowledge or improve analogy-reasoning skills of a population of students. Nevertheless, even in such circumstances, our automated procedure can be a "backup solution" to a lack of

understanding of a given situation.

8.3 Perspectives

8.3.1 Evaluation

In order to demonstrate the applicability and usefulness of our approaches and the prototype, more applications in real industrial cases are necessary and important. As stated in this thesis, the process of using TRIZ to solve inventive problems is complex and several simple cases studies are not enough to completely evaluate and validate the applicability and efficiency of the automatic ontology-based mechanism for solving inventive problems.

Through the case study presented in Chapter 7, the progressiveness and appropriateness of the prototype for providing more relevant and heuristic knowledge for helping TRIZ users to innovate have been illustrated . But, the actual results show that we are far from to say that a really new innovation has been achieved with the help of our prototype (lack of industrial data).

As a result, all the proposed methods in this thesis will be integrated in the research version of the software STEPS for the industrial evaluation[1], and the detailed process of evaluation will consist of four steps:

- **Step1. Define the parameters of the evaluation.**

  a. The purpose of the evaluation is to test the performance of the prototype IngeniousTRIZ, which will be integrated as a part of software STEPS.

  b. The evaluator consist of direct evaluator-software users and indirect evaluator-software developer.

  c. The users will use IngeniousTRIZ to search concept solutions for specific problems.

  d. The process of using IngeniousTRIZ is made up of three phases, and at the end of each phase, the corresponding results will be obtained:

[1]Up to now, only the proposed methods in Chapter 3 and 4 have been integrated in the research version of STEPS.

- Abstract solution from inventive principles -> Obtained: Appropriate inventive principles.
- Heuristic abstract solution from inventive standards -> Obtained: Heuristic abstract solution.
- The search of heuristic physical effects -> Obtained: Heuristic physical effects.

The users will be guided to evaluate the results obtained in each phase according to the methods described in Step2.

e. The software developer will receive all information from the users, based on which the performance of using IngeniousTRIZ to solve specific problems can be calculated according to Step3.

- **Step2. Design the methods used for the evaluation.**

a. The criterion is the ability to help users to find accurate and complete solutions for specific cases efficiently.

b. Three parameters are defined to measure the performance of IngeniousTRIZ:

- Accuracy: to estimate whether the obtained results are accurate for solving the specific case. The score value ranges from 0.0 to 1.0, and the higher the value, the more accurate the results.
- Completeness: to estimate whether all potential solutions for specific case are obtained. The value is from 0.0 to 1.0, and the higher the value, the more complete the results.
- Efficiency: to estimate whether the time is well used during the resolution of specific problem. The value is from 0.0 to 1.0, and the higher the value, the more efficient the process of this prototype.

c. At the end of each phase, the users need to give a value for each parameter according to the following guidelines:

1. Accuracy:
   - It is exact result I want in this phase, and I am sure it can be used to solve this problem. (value=1.0)

- It is exact result I want in this phase, but I am not sure it can be used to solve this problem. (value=0.8)
- The result is acceptable to some extent in this phase. (value=0.4)
- It is not result I want in this phase. (value=0.0)

2. Completeness:
   - All potential solutions are obtained in this phase. (value=1.0)
   - Most solutions are obtained in this phase. (value=0.8)
   - A few of solutions are obtained in this phase. (value=0.4)
   - No useful solution is obtained in this phase. (value=0.0)
3. Efficiency:
   - High. (value=1.0)
   - Moderate. (value=0.8)
   - Acceptable. (value=0.4)
   - Inacceptable. (value=0.0)

d. The software developer collect the data when the users finish the whole process of solving problems.

- **Step3. Set standards and collect evidence.**

a. The weight of three parameters W_a(weight of accuracy), W_c(weight of completeness) and W_e(weight of efficiency) ($W_a + W_c + W_e = 1$) can be set manually. For example, the similar generic engineering parameters need to be obtained as many as possible to match the specific parameters in different domains, so the weights are: $W_a = 0.3$, $W_c = 0.5$ and $W_e = 0.2$.

b. The performance for each phase can be quantified according to the following formula:

$$V_{performance-phase} = \frac{\sum_{i=1}^{n}(W_a S_{ai} + W_c S_{ci} + W_e S_{ei})}{n} \tag{8.1}$$

where n is the number of users, and S_{ai}, S_{ci} and S_{ei} are the score values of parameter accuracy, completeness and efficiency given be the ith user ($1 \leq i \leq n$).

c. The performance for the whole prototype can be quantified as following:

$$V_{performance} = \frac{\sum_{i=1}^{n} \frac{1}{3} \sum_{j=1}^{3} (W_a^j S_{ai}^j + W_c^j S_{ci}^j + W_e^j S_{ei}^j)}{n} \tag{8.2}$$

where W_a^j, W_c^j and W_e^j are weights of three parameter in the jth phase ($1 \leq j \leq 3$), and S_{ai}^j, S_{ci}^j and S_{ei}^j are the score values of three parameters for the jth phase given by the ith user ($1 \leq i \leq n$).

- **Step4. Report and make decisions.**

a. The software developer calculates the value of performance for each phase and also for the whole process.

b. Based on the obtained values, the performance of each phase and the whole process of the prototype IngeniousTRIZ can be identified as four types:

Table 8.1: Four types of the performance of IngeniousTRIZ

Value	Type
$0.8 - 1.0$	Excellent
$0.5 - 0.8$	Good
$0.3 - 0.5$	Acceptable
$0.0 - 0.3$	Poor

According to this evaluation process, the human-made "translations" will be tested and compared with the results obtained with the proposed methods at a wider scale, and in this way, the relative importance of them with regards to the pertinence of the solutions obtained can also be measured.

8.3.2 Improvement

Although the automatic ontology-based mechanism for solving inventive problems proposed in this thesis can facilitate the process of using TRIZ effectively, there are still some improvements that can be made in the further research:

The first problematic factor is WordNet, which is the semantic dictionary used in the calculation of semantic similarity of short texts. The proposed methods will not always

be able to satisfy the requirements of some specialized problem because of the limits of the WordNet dictionary. The interaction with users will solve this problem to a certain extent. For the moment, there is no strategy to lead the use of the SWRL rules to take profit of the knowledge about the specific case being solved. We intend, therefore, to provide a communication platform with users thanks to the development of SWRL rules describing the process of the interaction.

Secondly, during the process of searching heuristic abstract solutions, TRIZ users start solving inventive problems with forty inventive principles to obtain an abstract solution, and then, according to the selected inventive principles, the similar items of seventy-six inventive standards are obtained based on the semantic similarity calculated in advance. With the help of these similar items, the useful heuristic abstract solutions are obtained through the ontology inference. However, the direct transformation from solving technical contradictions to implementing Su-Field analysis, from the resolution with forty inventive principles to that with seventy-six inventive standards, results in a certain amount of missing information in the heuristic abstract solutions. For example, in the case of the "Diving Fin", only the specific parameters "kicking efficiency" and "ease of use" are considered to solve the technical contradiction, while the physical parameters of diving fins or using diving fins, such as the circumstance of its use and its components, are not considered, which often makes the user confused to modify the diving fin according to the obtained concept solution. One way to solve this problem is to bring in the analysis with eleven separation methods and physical contradictions to provide complementary information, that is, for the chosen inventive principles, not only the similar inventive standards are used to generate the heuristic abstract solutions, but also the similar separation methods are used to provide the information in physical level.

Last but not least, in order to solve various specific problems, the number and the content of the physical effects need to change dynamically according to the development of different kinds of fields, for example, Visual Effects recently becoming accessible owing to the appearance of the affordable animation and compositing software. The list of physical effects used in our research has been proposed several years ago, and in order to keep its dynamicity, we intend to use text mining techniques to extract the useful information for the physical effects from online resources - Wikipedia, such as, their initial and final states, and then instantiate them automatically through Protégé-OWL API.

Appendix A

The 39 Generic Engineering Parameters

1. Weight of moving object
2. Weight of stationary object
3. Length of moving object
4. Length of stationary object
5. Area of moving object
6. Area of stationary object
7. Volume of moving object
8. Volume of stationary object
9. Speed
10. Force (intensity)
11. Stress or pressure
12. Shape
13. Stability of object's composition
14. Strength
15. Duration of action of moving object
16. Duration of action of stationary object
17. Temperature
18. Illumination intensity
19. Use of energy by moving object
20. Use of energy by stationary object
21. Power
22. Loss of energy
23. Loss of substance
24. Loss of information
25. Loss of time
26. Quantity of substance or the matter
27. Reliability
28. Measurement accuracy
29. Manufacturing precision
30. Object-affected harmful factors
31. Object-generated harmful factors
32. Ease of manufacture
33. Ease of operation
34. Ease of repair
35. Adaptability or versatility
36. Device complexity
37. Difficulty of detecting and measuring
38. Extent of automation
39. Productivity

Appendix B

The Contradiction Matrix

The "Y-axis" of the contradiction matrix stands for the parameters to be improved and the "X-axis" shows the "undesired results", that is, the parameters to be deteriorated ($http://www.inventive-design.net$).

	Improving parameter \ Worsening parameter	1 Weight of moving object	2 Weight of stationary object	3 Length of moving object	4 Length of stationary object	5 Area of moving object	6 Area of stationary object	7 Volume of moving object	8 Volume of stationary object	9 Speed	10 Force (Intensity)	11 Stress or pressure	12 Shape	13 Stability of the object's composition	14 Strength	15 Duration of action of moving object	16 Duration of action of stationary object	17 Temperature	18 Illumination intensity	19 Use of energy by moving object	20 Use of energy by stationary object	21 Power	22 Loss of Energy	23 Loss of Substance	24 Loss of Information	25 Loss of Time	26 Quantity of substance	27 Reliability	28 Measurement accuracy	29 Manufacturing precision	30 Object-affected harmful factors	31 Object-generated harmful factors	32 Ease of manufacture	33 Ease of operation	34 Ease of repair	35 Adaptability or versatility	36 Device complexity	37 Difficulty of detecting and measuring	38 Extent of automation	39 Productivity
1	Weight of moving object	+	-	15, 8, 29,34	-	29, 17, 38, 34	-	29, 2, 40, 28	-	2, 8, 15, 38	8, 10, 18, 37	10, 36, 37, 40	10, 14, 35, 40	1, 35, 19, 39	28, 27, 18, 40	5, 34, 31, 35	-	6, 29, 4, 38	19, 1, 32	35, 12, 34, 31	-	12, 36, 18, 31	6, 2, 34, 19	5, 35, 3, 31	10, 24, 35	10, 35, 20, 28	3, 26, 18, 31	1, 3, 11, 27	28, 27, 35, 26	28, 35, 26, 18	22, 21, 18, 27	22, 35, 31, 39	27, 28, 1, 36	35, 3, 2, 24	2, 27, 28, 11	29, 5, 15, 8	26, 30, 36, 34	28, 29, 26, 32	26, 35 18, 19	35, 3, 24, 37
2	Weight of stationary object	-	+	-	10, 1, 29, 35	-	35, 30, 13, 2	-	5, 35, 14, 2	-	8, 10, 19, 35	13, 29, 10, 18	13, 10, 29, 14	26, 39, 1, 40	28, 2, 10, 27	-	2, 27, 19, 6	28, 19, 32, 22	19, 32, 35	-	18, 19, 28, 1	15, 19, 18, 22	18, 19, 28, 15	5, 8, 13, 30	10, 15, 35	10, 20, 35, 26	19, 6, 18, 26	10, 28, 8, 3	18, 26, 28	10, 1, 35, 17	2, 19, 22, 37	35, 22, 1, 39	28, 1, 9	6, 13, 1, 32	2, 27, 28, 11	19, 15, 29	1, 10, 26, 39	25, 28, 17, 15	2, 26, 35	1, 28, 15, 35
3	Length of moving object	8, 15, 29, 34	-	+	-	15, 17, 4	-	7, 17, 4, 35	-	13, 4, 8	17, 10, 4	1, 8, 35	1, 8, 10, 29	1, 8, 15, 34	8, 35, 29, 34	19	-	10, 15, 19	32	8, 35, 24	-	1, 35	7, 2, 35, 39	4, 29, 23, 10	1, 24	15, 2, 29	29, 35	10, 14, 29, 40	28, 32, 4	10, 28, 29, 37	1, 15, 17, 24	17, 15	1, 29, 17	15, 29, 35, 4	1, 28, 10	14, 15, 1, 16	1, 19, 26, 24	35, 1, 26, 24	17, 24, 26, 16	14, 4, 28, 29
4	Length of stationary object		35, 28, 40, 29	-	+	-	17, 7, 10, 40	-	35, 8, 2,14	-	28, 10	1, 14, 35	13, 14, 15, 7	39, 37, 35	15, 14, 28, 26	-	1, 40, 35	3, 35, 38, 18	3, 25	-		12, 8	6, 28	10, 28, 24, 35	24, 26,	30, 29, 14		15, 29, 28	32, 28, 3	2, 32, 10	1, 18		15, 17, 27	2, 25	3	1, 35	1, 26	26		30, 14, 7, 26
5	Area of moving object	2, 17, 29, 4	-	14, 15, 18, 4	-	+	-	7, 14, 17, 4		29, 30, 4, 34	19, 30, 35, 2	10, 15, 36, 28	5, 34, 29, 4	11, 2, 13, 39	3, 15, 40, 14	6, 3	-	2, 15, 16	15, 32, 19, 13	19, 32	-	19, 10, 32, 18	15, 17, 30, 26	10, 35, 2, 39	30, 26	26, 4	29, 30, 6, 13	29, 9	26, 28, 32, 3	2, 32	22, 33, 28, 1	17, 2, 18, 39	13, 1, 26, 24	15, 17, 13, 16	15, 13, 10, 1	15, 30	14, 1, 13	2, 36, 26, 18	14, 30, 28, 23	10, 26, 34, 2
6	Area of stationary object	-	30, 2, 14, 18	-	26, 7, 9, 39	-	+	-		-	1, 18, 35, 36	10, 15, 36, 37		2, 38	40	-	2, 10, 19, 30	35, 39, 38		-		17, 32	17, 7, 30	10, 14, 18, 39	30, 16	10, 35, 4, 18	2, 18, 40, 4	32, 35, 40, 4	26, 28, 32, 3	2, 29, 18, 36	27, 2, 39, 35	22, 1, 40	40, 16	16, 4	16	15, 16	1, 18, 36	2, 35, 30, 18	23	10, 15, 17, 7
7	Volume of moving object	2, 26, 29, 40	-	1, 7, 4, 35	-	1, 7, 4, 17	-	+	-	29, 4, 38, 34	15, 35, 36, 37	6, 35, 36, 37	1, 15, 29, 4	28, 10, 1, 39	9, 14, 15, 7	6, 35, 4	-	34, 39, 10, 18	2, 13, 10	35	-	35, 6, 13, 18	7, 15, 13, 16	36, 39, 34, 10	2, 22	2, 6, 34, 10	29, 30, 7	14, 1, 40, 11	25, 26, 28	25, 28, 2, 16	22, 21, 27, 35	17, 2, 40, 1	29, 1, 40	15, 13, 30, 12	10	15, 29	26, 1	29, 26, 4	35, 34, 16, 24	10, 6, 2, 34
8	Volume of stationary object	-	35, 10, 19, 14	19, 14	35, 8, 2, 14	-		-	+	-	2, 18, 37	24, 35	7, 2, 35	34, 28, 35, 40	9, 14, 17, 15	-	35, 34, 38	35, 6, 4		-		30, 6		10, 39, 35, 34		35, 16, 32 18	35, 3	2, 35, 16		35, 10, 25	34, 39, 19, 27	30, 18, 35, 4	35		1		1, 31	2, 17, 26		35, 37, 10, 2
9	Speed	2, 28, 13, 38	-	13, 14, 8	-	29, 30, 34	-	7, 29, 34	-	+	13, 28, 15, 19	6, 18, 38, 40	35, 15, 18, 34	28, 33, 1, 18	8, 3, 26, 14	3, 19, 35, 5	-	28, 30, 36, 2	10, 13, 19	8, 15, 35, 38	-	19, 35, 38, 2	14, 20, 19, 35	10, 13, 28, 38	13, 26		10, 19, 29, 38	11, 35, 27, 28	28, 32, 1, 24	10, 28, 32, 25	1, 28, 35, 23	2, 24, 35, 21	35, 13, 8, 1	32, 28, 13, 12	34, 2, 28, 27	15, 10, 26	10, 28, 4, 34	3, 34, 27, 16	10, 18	
10	Force (Intensity)	8, 1, 37, 18	18, 13, 1, 28	17, 19, 9, 36	28, 10	19, 10, 15	1, 18, 36, 37	15, 9, 12, 37	2, 36, 18, 37	13, 28, 15, 12	+	18, 21, 11	10, 35, 40, 34	35, 10, 21	35, 10, 14, 27	19, 2		35, 10, 21	-	19, 17, 10	1, 16, 36, 37	19, 35, 18, 37	14, 15	8, 35, 40, 5		10, 37, 36	14, 29, 18, 36	3, 35, 13, 21	35, 10, 23, 24	28, 29, 37, 36	1, 35, 40, 18	13, 3, 36, 24	15, 37, 18, 1	1, 28, 3, 25	15, 1, 11	15, 17, 18, 20	26, 35, 10, 18	36, 37, 10, 19	2, 35	3, 28, 35, 37
11	Stress or pressure	10, 36, 37, 40	13, 29, 10, 18	35, 10, 36	35, 1, 14, 16	10, 15, 36, 28	10, 15, 36, 37	6, 35, 10	35, 24	6, 35, 36	36, 35, 21	+	35, 4, 15, 10	35, 33, 2, 40	9, 18, 3, 40	19, 3, 27		35, 39, 19, 2	-	14, 24, 10, 37		10, 35, 14	2, 36, 25	10, 36, 3, 37		37, 36, 4	10, 14, 36	10, 13, 19, 35	6, 28, 25	3, 35	22, 2, 37	2, 33, 27, 18	1, 35, 16	11	2	35	19, 1, 35	2, 36, 37	35, 24	10, 14, 35, 37
12	Shape	8, 10, 29, 40	15, 10, 26, 3	29, 34, 5, 4	13, 14, 10, 7	5, 34, 4, 10		14, 4, 15, 22	7, 2, 35	35, 15, 34, 18	35, 10, 37, 40	34, 15, 10, 14	+	33, 1, 18, 4	30, 14, 10, 40	14, 26, 9, 25		22, 14, 19, 32	13, 15, 32	2, 6, 34, 14		4, 6, 2	14	35, 29, 3, 5		14, 10, 34, 17	36, 22	10, 40, 16	28, 32, 1	32, 30, 40	22, 1, 2, 35	35, 1	1, 32, 17, 28	32, 15, 26	2, 13, 1	1, 15, 29	16, 29, 1, 28	15, 13, 39	15, 1, 32	17, 26, 34, 10
13	Stability of the object's composition	21, 35, 2, 39	26, 39, 1, 40	13, 15, 1, 28	37	2, 11, 13	39	28, 10, 19, 39	34, 28, 35, 40	33, 15, 28, 18	10, 35, 21, 16	2, 35, 40	22, 1, 18, 4	+	17, 9, 15	13, 27, 10, 35	39, 3, 35, 23	35, 1, 32	32, 3, 27, 15	13, 19	27, 4, 29, 18	32, 35, 27, 31	14, 2, 39, 6	2, 14, 30, 40		35, 27	15, 32, 35		13	18	35, 24, 30, 18	35, 40, 27, 39	35, 19	32, 35, 30	2, 35, 10, 16	35, 30, 34, 2	2, 35, 22, 26	35, 22, 39, 23	1, 8, 35	23, 35, 40, 3
14	Strength	1, 8, 40, 15	40, 26, 27, 1	1, 15, 8, 35	15, 14, 28, 26	3, 34, 40, 29	9, 40, 28	10, 15, 14, 7	9, 14, 17, 15	8, 13, 26, 14	10, 18, 3, 14	10, 3, 18, 40	10, 30, 35, 40	13, 17, 35	+	27, 3, 26		30, 10, 40	35, 19	19, 35, 10	35	10, 26, 35, 28	35	35, 28, 31, 40		29, 3, 28, 10	29, 10, 27	11, 3	3, 27, 16	3, 27	18, 35, 37, 1	15, 35, 22, 2	11, 3, 10, 32	32, 40, 28, 2	27, 11, 3	15, 3, 32	2, 13, 28	27, 3, 15, 40	15	29, 35, 10, 14
15	Duration of action of moving object	19, 5, 34, 31	-	2, 19, 9	-	3, 17, 19	-	10, 2, 19, 30	-	3, 35, 5	19, 2, 16	19, 3, 27	14, 26, 28, 25	13, 3, 35	27, 3, 10	+	-	19, 35, 39	2, 19, 4, 35	28, 6, 35, 18		19, 10, 35, 38		28, 27, 3, 18	10	20, 10, 28, 18	3, 35, 10, 40	11, 2, 13	3	3, 27, 16, 40	22, 15, 33, 28	21, 39, 16, 22	27, 1, 4	12, 27	29, 10, 27	1, 35, 13	10, 4, 29, 15	19, 29, 39, 35	6, 10	35, 17, 14, 19
16	Duration of action by stationary object	-	6, 27, 19, 16	-	1, 40, 35	-		-	35, 34, 38	-				39, 3, 35, 23		-	+	19, 18, 36, 40		-		16		27, 16, 18, 38	10	28, 20, 10, 16	3, 35, 31	34, 27, 6, 40	10, 26, 24		17, 1, 40, 33	22	35, 10	1	1	2		25, 34, 6, 35	1	20, 10, 16, 38
17	Temperature	36,22, 6, 38	22, 35, 32	15, 19, 9	15, 19, 9	3, 35, 39, 18	35, 38	34, 39, 40, 18	35, 6, 4	2, 28, 36, 30	35, 10, 3, 21	35, 39, 19, 2	14, 22, 19, 32	1, 35, 32	10, 30, 22, 40	19, 13, 39	19, 18, 36, 40	+	32, 30, 21, 16	19, 15, 3, 17		2, 14, 17, 25	21, 17, 35, 38	21, 36, 29, 31		35, 28, 21, 18	3, 17, 30, 39	19, 35, 3, 10	32, 19, 24	24	22, 33, 35, 2	22, 35, 2, 24	26, 27	26, 27	4, 10, 16	2, 18, 27	2, 17, 16	3, 27, 35, 31	26, 2, 19, 16	15, 28, 35
18	Illumination intensity	19, 1, 32	2, 35, 32	19, 32, 16		19, 32, 26		2, 13, 10		10, 13, 19	26, 19, 6		32, 30	32, 3, 27	35, 19	2, 19, 6		32, 35, 19	+	32, 1, 19	32, 35, 1, 15	32	19, 16, 1, 6	13, 1	1, 6	19, 1, 26, 17	1, 19		11, 15, 32	3, 32	15, 19	35, 19, 32, 39	19, 35, 28, 26	28, 26, 19	15, 17, 13, 16	15, 1, 19	6, 32, 13	32, 15	2, 26, 10	2, 25, 16
19	Use of energy by moving object	12,18,28, 31	-	12, 28	-	15, 19, 25	-	35, 13, 18	-	8, 15, 35	16, 26, 21, 2	23, 14, 25	12, 2, 29	19, 13, 17, 24	5, 19, 9, 35	28, 35, 6, 18	-	19, 24, 3, 14	2, 15, 19	+	-	6, 19, 37, 18	12, 22, 15, 24	35, 24, 18, 5		35, 38, 19, 18	34, 23, 16, 18	19, 21, 11, 27	3, 1, 32		1, 35, 6, 27	2, 35, 6	28, 26, 30	19, 35	1, 15, 17, 28	15, 17, 13, 16	2, 29, 27, 28	35, 38	32, 2	12, 28, 35
20	Use of energy by stationary object	-	19, 9, 6, 27	-		-		-		-	36, 37			27, 4, 29, 18	35				19, 2, 35, 32	-	+			28, 27, 18, 31			3, 35, 31	10, 36, 23			10, 2, 22, 37	19, 22, 18	1, 4					19, 35, 16, 25		1, 6
21	Power	8, 36, 38, 31	19, 26, 17, 27	1, 10, 35, 37		19, 38	17, 32, 13, 38	35, 6, 38	30, 6, 25	15, 35, 2	26, 2, 36, 35	22, 10, 35	29, 14, 2, 40	35, 32, 15, 31	26, 10, 28	19, 35, 10, 38	16	2, 14, 17, 25	16, 6, 19	16, 6, 19, 37		+	10, 35, 38	28, 27, 18, 38	10, 19	35, 20, 10, 6	4, 34, 19	19, 24, 26, 31	32, 15, 2	32, 2	19, 22, 31, 2	2, 35, 18	26, 10, 34	26, 35, 10	35, 2, 10, 34	19, 17, 34	20, 19, 30, 34	19, 35, 16	28, 2, 17	28, 35, 34
22	Loss of Energy	15, 6, 19, 28	19, 6, 18, 9	7, 2, 6, 13	6, 38, 7	15, 26, 17, 30	17, 7, 30, 18	7, 18, 23	7	16, 35, 38	36, 38			14, 2, 39, 6	26			19, 38, 7	1, 13, 32, 15			3, 38	+	35, 27, 2, 37	19, 10	10, 18, 32, 7	7, 18, 25	11, 10, 35	32		21, 22, 35, 2	21, 35, 2, 22		35, 32, 1	2, 19		7, 23	35, 3, 15, 23	2	28, 10, 29, 35
23	Loss of substance	35, 6, 23, 40	35, 6, 22, 32	14, 29, 10, 39	10, 28,24	35, 2, 10, 31	10, 18, 39, 31	1, 29, 30, 36	3, 39, 18, 31	10, 13, 28, 38	14, 15, 18, 40	3, 36, 37, 10	29, 35, 3, 5	2, 14, 30, 40	35, 28, 31, 40	28, 27, 3, 18	27, 16, 18, 38	21, 36, 39, 31	1, 6, 13	35, 18, 24, 5	28, 27, 12, 31	28, 27, 18, 38	35, 27, 2, 31	+		15, 18, 35, 10	6, 3, 10, 24	10, 29, 39, 35	16, 34, 31, 28	35, 10, 24, 31	33, 22, 30, 40	10, 1, 34, 29	15, 34, 33	32, 28, 2, 24	2, 35, 34, 27	15, 10, 2	35, 10, 28, 24	35, 18, 10, 13	35, 10, 18	28, 35, 10, 23
24	Loss of Information	10, 24, 35	10, 35, 5	1, 26	26	30, 26	30, 16		2, 22	26, 32						10	10		19			10, 19	19, 10		+	24, 26, 28, 32	24, 28, 35	10, 28, 23			22, 10, 1	10, 21, 22	32	27, 22				35, 33	35	13, 23, 15
25	Loss of Time	10, 20, 37, 35	10, 20, 26, 5	15, 2, 29	30, 24, 14, 5	26, 4, 5, 16	10, 35, 17, 4	2, 5, 34, 10	35, 16, 32, 18		10, 37, 36,5	37, 36,4	4, 10, 34, 17	35, 3, 22, 5	29, 3, 28, 18	20, 10, 28, 18	28, 20, 10, 16	35, 29, 21, 18	1, 19, 26, 17	35, 38, 19, 18	1	35, 20, 10, 6	10, 5, 18, 32	35, 18, 10, 39	24, 26, 28, 32	+	35, 38, 18, 16	10, 30, 4	24, 34, 28, 32	24, 26, 28, 18	35, 18, 34	35, 22, 18, 39	35, 28, 34, 4	4, 28, 10, 34	32, 1, 10	35, 28	6, 29	18, 28, 32, 10	24, 28, 35, 30	
26	Quantity of substance/the matter	35, 6, 18, 31	27, 26, 18, 35	29, 14, 35, 18		15, 14, 29	2, 18, 40, 4	15, 20, 29		35, 29, 34, 28	35, 14, 3	10, 36, 14, 3	35, 14	15, 2, 17, 40	14, 35, 34, 10	3, 35, 10, 40	3, 35, 31	3, 17, 39		34, 29, 16, 18	3, 35, 31	35	7, 18, 25	6, 3, 10, 24	24, 28, 35	35, 38, 18, 16	+	18, 3, 28, 40	3, 2, 28	33, 30	35, 33, 29, 31	3, 35, 40, 39	29, 1, 35, 27	35, 29, 25, 10	2, 32, 10, 25	15, 3, 29	3, 13, 27, 10	3, 27, 29, 18	8, 35	13, 29, 3, 27
27	Reliability	3, 8, 10, 40	3, 10, 8, 28	15, 9, 14, 4	15, 29, 28, 11	17, 10, 14, 16	32, 35, 40, 4	3, 10, 14, 24	2, 35, 24	21, 35, 11, 28	8, 28, 10, 3	10, 24, 35, 19	35, 1, 16, 11		11, 28	2, 35, 3, 25	34, 27, 6, 40	3, 35, 10	11, 32, 13	21, 11, 27, 19	36, 23	21, 11, 26, 31	10, 11, 35	10, 35, 29, 39	10, 28	10, 30, 4	21, 28, 40, 3	+	32, 3, 11, 23	11, 32, 1	27, 35, 2, 40	35, 2, 40, 26		27, 17, 40	1, 11	13, 35, 8, 24	13, 35, 1	27, 40, 28	11, 13, 27	1, 35, 29, 38
28	Measurement accuracy	32, 35, 26, 28	28, 35, 25, 26	28, 26, 5, 16	32, 28, 3, 16	26, 28, 32, 3	26, 28, 32, 3	32, 13, 6		28, 13, 32, 24	32, 2	6, 28, 32	6, 28, 32	32, 35, 13	28, 6, 32	28, 6, 32	10, 26, 24	6, 19, 28, 24	6, 1, 32	3, 6, 32		3, 6, 32	26, 32, 27	10, 16, 31, 28		24, 34, 28, 32	2, 6, 32	5, 11, 1, 23	+		28, 24, 22, 26	3, 33, 39, 10	6, 35, 25, 18	1, 13, 17, 34	1, 32, 13, 11	13, 35, 2	27, 35, 10, 34	26, 24, 32, 28	28, 2, 10, 34	10, 34, 28, 32
29	Manufacturing precision	28, 32, 13, 18	28, 35, 27, 9	10, 28, 29, 37	2, 32, 10	28, 33, 29, 32	2, 29, 18, 36	32, 28, 2	25, 10, 35	10, 28, 32	28, 19, 34, 36	3, 35	32, 30, 40	30, 18	3, 27	3, 27, 40		19, 26	3, 32	32, 2		32, 2	13, 32, 2	35, 31, 10, 24		32, 26, 28, 18	32, 30	11, 32, 1		+	26, 28, 10, 36	4, 17, 34, 26		1, 32, 35, 23	25, 10		26, 2, 18		26, 28, 18, 23	10, 18, 32, 39
30	Object-affected harmful factors	22, 21, 27, 39	2, 22, 13, 24	17, 1, 39, 4	1, 18	22, 1, 33, 28	27, 2, 39, 35	22, 23, 37, 35	34, 39, 19, 27	21, 22, 35, 28	13, 35, 39, 18	22, 2, 37	22, 1, 3, 35	35, 24, 30, 18	18, 35, 37, 1	22, 15, 33, 28	17, 1, 40, 33	22, 33, 35, 2	1, 19, 32, 13	1, 24, 6, 27	10, 2, 22, 37	19, 22, 31, 2	21, 22, 35, 2	33, 22, 19, 40	22, 10, 2	35, 18, 34	35, 33, 29, 31	27, 24, 2, 40	28, 33, 23, 26	26, 28, 10, 18	+		24, 35, 2	2, 25, 28, 39	35, 10, 2	35, 11, 22, 31	22, 19, 29, 40	22, 19, 29, 40	33, 3, 34	22, 35, 13, 24
31	Object-generated harmful factors	19, 22, 15, 39	35, 22, 1, 39	17, 15, 16, 22		17, 2, 18, 39	22, 1, 40	17, 2, 40	30, 18, 35, 4	35, 28, 3, 23	35, 28, 1, 40	2, 33, 27, 18	35, 1	35, 40, 27, 39	15, 35, 22, 2	15, 22, 33, 31	21, 39, 16, 22	22, 35, 2, 24	19, 24, 39, 32	2, 35, 6	19, 22, 18	2, 35, 18	21, 35, 2, 22	10, 1, 34	10, 21, 29	1, 22	3, 24, 39, 1	24, 2, 40, 39	3, 33, 26	4, 17, 34, 26		+					19, 1, 31	2, 21, 27, 1	2	22, 35, 18, 39
32	Ease of manufacture	28, 29, 15, 16	1, 27, 36, 13	1, 29, 13, 17	15, 17, 27	13, 1, 26, 12	16, 40	13, 29, 1, 40	35	35, 13, 8, 1	35, 12	35, 19, 1, 37	1, 28, 13, 27	11, 13, 1	1, 3, 10, 32	27, 1, 4	35, 16	27, 26, 18	28, 24, 27, 1	28, 26, 27, 1	1, 4	27, 1, 12, 24	19, 35	15, 34, 33	32, 24, 18, 16	35, 28, 34, 4	35, 23, 1, 24		1, 35, 12, 18		24, 2		+	2, 5, 13, 16	35, 1, 11, 9	2, 13, 15	27, 26, 1	6, 28, 11, 1	8, 28, 1	35, 1, 10, 28
33	Ease of operation	25, 2, 13, 15	6, 13, 1, 25	1, 17, 13, 12		1, 17, 13, 16	18, 16, 15, 39	1, 16, 35, 15	4, 18, 39, 31	18, 13, 34	28, 13 35	2, 32, 12	15, 34, 29, 28	32, 35, 30	32, 40, 3, 28	29, 3, 8, 25	1, 16, 25	26, 27, 13	13, 17, 1, 24	1, 13, 24		35, 34, 2, 10	2, 19, 13	28, 32, 2, 24	4, 10, 27, 22	4, 28, 10, 34	12, 35	17, 27, 8, 40	25, 13, 2, 34	1, 32, 35, 23	2, 25, 28, 39		2, 5, 12	+	12, 26, 1, 32	15, 34, 1, 16	32, 25, 12, 17		1, 34, 12, 3	15, 1, 28
34	Ease of repair	2, 27 35, 11	2, 27, 35, 11	1, 28, 10, 25	3, 18, 31	15, 13, 32	16, 25	25, 2, 35, 11	1	34, 9	1, 11, 10	13	1, 13, 2, 4	2, 35	11, 1, 2, 9	11, 29, 28, 27	1	4, 10	15, 1, 13	15, 1, 28, 16		15, 10, 32, 2	15, 1, 32, 19	2, 35, 34, 27		32, 1, 10, 25	2, 28, 10, 25	11, 10, 1, 16	10, 2, 13	25, 10	35, 10, 2, 16		1, 35, 11, 10	1, 12, 26, 15	+	7, 1, 4, 16	35, 1, 13, 11		34, 35, 7, 13	1, 32, 10
35	Adaptability or versatility	1, 6, 15, 8	19, 15, 29, 16	35, 1, 29, 2	1, 35, 16	35, 30, 29, 7	15, 16	15, 35, 29		35, 10, 14	15, 17, 20	35, 16	15, 37, 1, 8	35, 30, 14	35, 3, 32, 6	13, 1, 35	2, 16	27, 2, 3, 35	6, 22, 26, 1	19, 35, 29, 13		19, 1, 29	18, 15, 1	15, 10, 2, 13		35, 28	3, 35, 15	35, 13, 8, 24	35, 5, 1, 10		35, 11, 32, 31		1, 13, 31	15, 34, 1, 16	1, 16, 7, 4	+	15, 29, 37, 28	1	27, 34, 35	35, 28, 6, 37
36	Device complexity	26, 30, 34, 36	2, 26, 35, 39	1, 19, 26, 24	26	14, 1, 13, 16	6, 36	34, 26, 6	1, 16	34, 10, 28	26, 16	19, 1, 35	29, 13, 28, 15	2, 22, 17, 19	2, 13, 28	10, 4, 28, 15		2, 17, 13	24, 17, 13	27, 2, 29, 28		20, 19, 30, 34	10, 35, 13, 2	35, 10, 28, 29		6, 29	13, 3, 27, 10	13, 35, 1	2, 26, 10, 34	26, 24, 32	22, 19, 29, 40	19, 1	27, 26, 1, 13	27, 9, 26, 24	1, 13	29, 15, 28, 37	+	15, 10, 37, 28	15, 1, 24	12, 17, 28
37	Difficulty of detecting and measuring	27, 26, 28, 13	6, 13, 28, 1	16, 17, 26, 24	26	2, 13, 18, 17	2, 39, 30, 16	29, 1, 4, 16	2, 18, 26, 31	3, 4, 16, 35	36, 28, 40, 19	35, 36, 37, 32	27, 13, 1, 39	11, 22, 39, 30	27, 3, 15, 28	19, 29, 39, 25	25, 34, 6, 35	3, 27, 35, 16	2, 24, 26	35, 38	19, 35, 16	19, 1, 16, 10	35, 3, 15, 19	1, 18, 10, 24	35, 33, 27, 22	18, 28, 32, 9	3, 27, 29, 18	27, 40, 28, 8	26, 24, 32, 28		22, 19, 29, 28	2, 21	5, 28, 11, 29	2, 5	12, 26	1, 15	15, 10, 37, 28	+	34, 21	35, 18
38	Extent of automation	28, 26, 18, 35	28, 26, 35, 10	14, 13, 17, 28	23	17, 14, 13		35, 13, 16		28, 10	2, 35	13, 35	15, 32, 1, 13	18, 1	25, 13	6, 9		26, 2, 19	8, 32, 19	2, 32, 13		28, 2, 27	23, 28	35, 10, 18, 5	35, 33	24, 28, 35, 30	35, 13	11, 27, 32	28, 26, 10, 34	28, 26, 18, 23	2, 33	2	1, 26, 13	1, 12, 34, 3	1, 35, 13	27, 4, 1, 35	15, 24, 10	34, 27, 25	+	5, 12, 35, 26
39	Productivity	35, 26, 24, 37	28, 27, 15, 3	18, 4, 28, 38	30, 7, 14, 26	10, 26, 34, 31	10, 35, 17, 7	2, 6, 34, 10	35, 37, 10, 2		28, 15, 10, 36	10, 37, 14	14, 10, 34, 40	35, 3, 22, 39	29, 28, 10, 18	35, 10, 2, 18	20, 10, 16, 38	35, 21, 28, 10	26, 17, 19, 1	35, 10, 38, 19	1	35, 20, 10	28, 10, 29, 35	28, 10, 35, 23	13, 15, 23		35, 38	1, 35, 10, 38	1, 10, 34, 28	18, 10, 32, 1	22, 35, 13, 24	35, 22, 18, 39	35, 28, 2, 24	1, 28, 7, 19	1, 32, 10, 25	1, 35, 28, 37	12, 17, 28, 24	35, 18, 27, 2	5, 12, 35, 26	+

Appendix C

The 40 Inventive Principles

The 40 Inventive Principles (IP) are used with the contradiction matrix to solve technical contradictions. This list with the so-called "sub-principles" that intend to help clarify the meaning of the principles was taken from the TRIZ Journal ($http://www.inventive-design.net$).

- *Inventive Principle 1*: Segmentation
  (a) Divide an object into independent parts.
  (b) Make an object easy to disassemble.
  (c) Increase the degree of fragmentation or segmentation.
- *Inventive Principle 2*: Taking out
  (a) Separate an interfering part or property from an object, or single out the only necessary part (or property) of an object.
- *Inventive Principle 3*: Local quality
  (a) Change an object's structure from uniform to non-uniform, change an external environment (or external influence) from uniform to non-uniform.
  (b) Make each part of an object function in conditions most suitable for its operation.

(c) Make each part of an object fulfill a different and useful function.

- *Inventive Principle 4*: Asymmetry

  (a) Change the shape of an object from symmetrical to asymmetrical.

  (b) If an object is asymmetrical, increase its degree of asymmetry.

- *Inventive Principle 5*: Merging

  (a) Bring closer together (or merge) identical or similar objects, assemble identical or similar parts to perform parallel operations.

  (b) Make operations contiguous or parallel; bring them together in time.

- *Inventive Principle 6*: Universality

  (a) Make a part or object perform multiple functions; eliminate the need for other parts.

- *Inventive Principle 7*: “Nested doll"

  (a) Place one object inside another; place each object, in turn, inside the other.

  (b) Make one part pass through a cavity in the other.

- *Inventive Principle 8*: Anti-weight

  (a) To compensate for the weight of an object, merge it with other objects that provide lift.

  (b) To compensate for the weight of an object, make it interact with the environment (e.g. use aerodynamic, hydrodynamic, buoyancy and other forces).

- *Inventive Principle 9*: Preliminary anti-action

  (a) If it will be necessary to do an action with both harmful and useful effects, this action should be replaced with anti-actions to control harmful effects.

  (b) Create beforehand stresses in an object that will oppose known undesirable working stresses later on.

- *Inventive Principle 10*: Preliminary action

(a) Perform, before it is needed, the required change of an object (either fully or partially).

(b) Pre-arrange objects such that they can come into action from the most convenient place and without losing time for their delivery.

- *Inventive Principle 11*: Beforehand cushioning

  (a) Prepare emergency means beforehand to compensate for the relatively low reliability of an object.

- *Inventive Principle 12*: Equipotentiality

  (a) In a potential field, limit position changes (e.g. change operating conditions to eliminate the need to raise or lower objects in a gravity field).

- *Inventive Principle 13*: “The other way round"

  (a) Invert the action(s) used to solve the problem (e.g. instead of cooling an object, heat it).

  (b) Make movable parts (or the external environment) fixed, and fixed parts movable).

  (c) Turn the object (or process) ‘upside down’.

- *Inventive Principle 14*: Spheroidality - Curvature

  (a) Instead of using rectilinear parts, surfaces, or forms, use curvilinear ones; move from flat surfaces to spherical ones; from parts shaped as a cube (parallelepiped) to ball-shaped structures.

  (b) Use rollers, balls, spirals, domes.

  (c) Go from linear to rotary motion, use centrifugal forces.

- *Inventive Principle 15*: Dynamics

  (a) Allow (or design) the characteristics of an object, external environment, or process to change to be optimal or to find an optimal operating condition.

  (b) Divide an object into parts capable of movement relative to each other.

(c) If an object (or process) is rigid or inflexible, make it movable or adaptive.

- *Inventive Principle 16*: Partial or excessive actions

  (a) If 100 percent of an object is hard to achieve using a given solution method then, by using 'slightly less' or 'slightly more' of the same method, the problem may be considerably easier to solve.

- *Inventive Principle 17*: Another dimension

  (a) To move an object in two- or three-dimensional space.

  (b) Use a multi-story arrangement of objects instead of a single-story arrangement.

  (c) Tilt or re-orient the object, lay it on its side.

  (d) Use 'another side' of a given area.

- *Inventive Principle 18*: Mechanical vibration

  (a) Cause an object to oscillate or vibrate.

  (b) Increase its frequency (even up to the ultrasonic).

  (c) Use an object's resonant frequency.

  (d) Use piezoelectric vibrators instead of mechanical ones.

  (e) Use combined ultrasonic and electromagnetic field oscillations.

- *Inventive Principle 19*: Periodic action

  (a) Instead of continuous action, use periodic or pulsating actions.

  (b) If an action is already periodic, change the periodic magnitude or frequency.

  (c) Use pauses between impulses to perform a different action.

- *Inventive Principle 20*: Continuity of useful action

  (a) Carry on work continuously; make all parts of an object work at full load, all the time.

  (b) Eliminate all idle or intermittent actions or work.

- *Inventive Principle 21*: Skipping

(a) Conduct a process , or certain stages (e.g. destructible, harmful or hazardous operations) at high speed.

- *Inventive Principle 22*: “Blessing in disguise" or “Turn Lemons into Lemonade"

  (a) Use harmful factors (particularly, harmful effects of the environment or surroundings) to achieve a positive effect.

  (b) Eliminate the primary harmful action by adding it to another harmful action to resolve the problem.

  (c) Amplify a harmful factor to such a degree that it is no longer harmful.

- *Inventive Principle 23*: Feedback

  (a) Introduce feedback (referring back, cross-checking) to improve a process or action.

  (b) If feedback is already used, change its magnitude or influence.

- *Inventive Principle 24*: ‘Intermediary’

  (a) Use an intermediary carrier article or intermediary process.

  (b) Merge one object temporarily with another (which can be easily removed).

- *Inventive Principle 25*: Self-service

  (a) Make an object serve itself by performing auxiliary helpful functions.

  (b) Use waste resources, energy, or substances.

- *Inventive Principle 26*: Copying

  (a) Instead of an unavailable, expensive, fragile object, use simpler and inexpensive copies.

  (b) Replace an object, or process with optical copies.

  (c) If visible optical copies are already used, move to infrared or ultraviolet copies.

- *Inventive Principle 27*: Cheap short-living objects

(a) Replace an inexpensive object with a multiple of inexpensive objects, comprising certain qualities (such as service life, for instance).

- *Inventive Principle 28*: Mechanics substitution

  (a) Replace a mechanical means with a sensory (optical, acoustic, taste or smell) means.

  (b) Use electric, magnetic and electromagnetic fields to interact with the object.

  (c) Change from static to movable fields, from unstructured fields to those having structure.

  (d) Use fields in conjunction with field-activated (e.g. ferromagnetic) particles.

- *Inventive Principle 29*: Pneumatics and hydraulics

  (a) Use gas and liquid parts of an object instead of solid parts (e.g. inflatable, filled with liquids, air cushion, hydrostatic, hydro-reactive).

- *Inventive Principle 30*: Flexible shells and thin films

  (a) Use flexible shells and thin films instead of three dimensional structures.

  (b) Isolate the object from the external environment using flexible shells and thin films.

- *Inventive Principle 31*: Porous materials

  (a) Make an object porous or add porous elements (inserts, coatings, etc.).

  (b) If an object is already porous, use the pores to introduce a useful substance or function.

- *Inventive Principle 32*: Color changes

  (a) Change the color of an object or its external environment.

  (b) Change the transparency of an object or its external environment.

- *Inventive Principle 33*: Homogeneity

  (a) Make objects interacting with a given object of the same material (or material with identical properties).

- *Inventive Principle 34*: Discarding and recovering
  (a) Make portions of an object that have fulfilled their functions go away (discard by dissolving, evaporating, etc.) or modify these directly during operation.
  (b) Conversely, restore consumable parts of an object directly in operation.
- *Inventive Principle 35*: Change of physical and chemical parameters
  (a) Change the object's aggregate state.
  (b) Change concentration or consistency of the object.
  (c) Change the degree of flexibility of the object.
  (d) Change the temperature of the object or environment.
- *Inventive Principle 36*: Phase transitions
  (a) Use phenomena occurring during phase transitions (e.g. volume changes, loss or absorption of heat, etc.).
- *Inventive Principle 37*: Thermal expansion
  (a) Use thermal expansion (or contraction) of materials.
  (b) If thermal expansion is being used, use multiple materials with different coefficients of thermal expansion.
- *Inventive Principle 38*: Strong oxidants
  (a) Replace common air with oxygen-enriched air.
  (b) Replace enriched air with pure oxygen.
  (c) Expose air or oxygen to ionizing radiation.
  (d) Use ionized oxygen.
  (e) Replace ozonized (or ionized) oxygen with ozone.
- *Inventive Principle 39*: Inert atmosphere
  (a) Use inert gases instead of usual ones.
  (b) Add neutral parts or additives to the object.

- *Inventive Principle 40*: Composite materials

  (a) Change from uniform to composite (multiple) materials.

Appendix D

The 76 Inventive Standards

D.1 Some Extra Precisions on Inventive Standards

The 76 Inventive Standard Solutions (IS) are in five classes, with various sub-classes, which are used according to the type of engineering problems they solve. The five classes are:

- Class 1: Building and Destruction of Substance-Field Models (13 IS)
- Class 2: Development of Substance-Field Models (23 IS)
- Class 3 Transition to Super-system and Micro level (6 IS)
- Class 4: Standards for Detection and Measuring (17 IS)
- Class 5: Standards on Application of Standards (17 IS).

This appendix describes the classes and the sub-classes.

Class 1: Building and Destruction of Substance-Field Models

Class 1 helps solve problems by building or destroying the Su-Field Models if they are incomplete or have harmful functions. Class 1 has two sub-classes containing 13 IS:

Sub-class 1.1 Building of Su-Fields (if incomplete) (8 IS)

The major recommendations from this sub-class are:

- Make the Su-Field complete.
- Make it minimally workable by introducing an internal additive.
- Make it minimally workable by introducing an external additive.
- Use minimal - maximal mode (add more and remove the extras; add less and enhance locally).

Sub-class 1.2 Destruction of Su-Field (harms) (5 IS)

The major recommendations from this sub-class are:

- Introduce a third substance between the given two substances.
- Introduce a third substance from the super-system.
- Introduce a third substance that is a modification of one of the given two substances.
- Introduce a sacrificial substance.
- Introduce a field that counteracts the harmful field.

Class 2: Development of Substance-Field Models

This class is used for improving the efficiency of engineering systems by introducing minor modifications.

It offers concept solutions of how to improve and evolve systems. The major recommendations from this class are:

- Use of chain Su - Fields
- Use of double Su-Fields
- Segmentation (including porosity increase)

- Dynamisation
- Rhythm coordination
- Use of magnetic substances.

Class 2 contains 4 subclasses and 23 IS:

Sub-class 2.1 Transition to complex Su-Field Models (2 IS)

Sub-class 2.2 Evolution of Su-Fields Models (6 IS)

Sub-class 2.3 Evolution of rhythms (3 IS)

Sub-class 2.4 Complex forced Su-Field Models (12 IS)

Class 3: System Transitions and Evolution-Transition to Super-system and Sub-system

This class is used for solving problems by developing solutions at different levels in the system (super-system or sub-system). The major recommendations from this class are how to improve systems by combining elements or combining with other systems.

Class 3 contains 2 sub-classes containing 6 IS:

Sub-class 3.1 Simplicity-complexity-simplicity (mono-bi-poly) and increasing flexibility and dynamisation (Transition to super-system and to bi and poly systems; use no links, rigid links, flexible links, "field" links) (5 IS)

Sub-class 3.2 Transition to micro-level (examine the sub-system, use smart substances) (1 IS)

Class 4: Solutions for Detection and Measurement

This class is used for solving measuring or detection problems in engineering systems. These solutions have many distinguishing features, especially the use of indirect methods and the use of copies.

Detection and measurement are typically for control. Detection is binary (something either happens or doesn't happen) and measurement has some level of quantification and precision. For example, a length measurement might be 2.15 m $\pm$ 0.01 m. Often the most innovative solution is automatic control which removes formal detection/measurement by taking advantage of physical, chemical or geometrical effects.

The major recommendations of this class are:

- Try to change the system so that there is no need to measure/detect.
- Measure a copy.
- Introduce a substance that generates a field (introduce a mark internally or externally).

Class 4 contains 5 sub-classes and 17 IS:

Sub-class 4.1 Indirect Methods (3 IS)
Sub-class 4.2 Create or Build a Measurement System (4 IS)
Sub-class 4.3 Enhancing the Measurement System (3 IS)
Sub-class 4.4 Measure Ferromagnetic-field (5 IS)
Sub-class 4.5 Direction of Evolution of the Measuring Systems (2 IS)

Class 5: Standards on Application of Standards

After using the other four classes of the Standard Solutions, Class 5 is additionally helpful for further general improvements and simplification of systems. These Standard Solutions give recommendations of how to introduce new substances or fields or use scientific effects more effectively after applying the relevant Standard Solutions in the four previous classes.

Class 5 solutions help when simplifying or trimming the system to remove components or to reduce the strength of the relevant interaction. The first four classes of Standard Solutions above often lead to solutions which increase complexity because something

if often added to the system to solve the problem. This fifth class shows how to get something extra through simplification but without introducing anything new.

The useful recommendations from this class are:

- Instead of a substance, introduce a field.
- Instead of a substance, introduce a void.
- Introduce a substance for a limited time.
- Introduce a little bit of a substance, but in a very concentrated way.
- Use phase changes.
- Get the substance or environment to change themselves to solve the problem.
- Use segmentation.

Class 5 contains 5 sub-classes and 17 IS:

Sub-class 5.1 Indirect methods for introducing substances under restricted conditions (4 IS)
Sub-class 5.2 Introducing fields under restricted conditions (3 IS)
Sub-class 5.3 Phase transitions (5 IS)
Sub-class 5.4 Clever use of natural phenomena (2 IS)
Sub-class 5.5 Generating higher or lower forms of substances (3 IS)

D.2 The 71 Inventive Standards of A-class and B-class

The 71 inventive standards of A-class and B-class, which are used in our research, are shown as following:

1 1.1.1. Synthesis of SFM
2 1.1.2. Transition to Internal complex SFM
3 1.1.3. Transition to External Complex SFM
4 1.1.4. Transition to SFM by using external environment
5 1.1.5. Transition to SFM by using external environment with additives
6 1.1.6. Minimum mode of action
7 1.1.7. Maximum mode of action
8 1.1.8. Selective-maximum mode
9 1.2.1. Elimination of harmful interaction by introducing foreign substance
10 1.2.2. Elimination of harmful interaction by modification existing substances
11 1.2.3. Drawing of harmful action of the field
12 1.2.4. Counteraction for harmful actions through the field
13 1.2.5. "Disconnection" of magnetic interactions
14 2.1.1. Transition to chain SFM
15 2.1.2. Transition to dual SFM
16 2.2.1. Transition to more controlled fields
17 2.2.3. Transition to capillary porous substance
18 2.2.5. Structuring of field
19 2.2.6. Structuring of substance
20 2.3.1. Matching the rhythm of field and product (or tool)
21 2.3.2. Matching of rhythm of fields
22 2.3.3. Coordination of the incompatible or independent actions
23 2.4.1. Transition to a ferromagnetic substance and a magnetic field
24 2.4.2. Transition to a ferromagnetic substance
25 2.4.3. Using of magnetic fluids
26 2.4.4. Capillary porous structure of ferromagnetic SFM
27 2.4.5. Transition to complex ferromagnetic SFM
28 2.4.6. Transition to ferromagnetic SFM in the external environment
29 2.4.7. Using physical effects
30 2.4.8. Increasing the degree of dynamism of feSFM
31 2.4.9. Structuring of feSFM
32 2.4.10.Matching the rhythms into feSFM
33 2.4.11.Transition to electrical SFM
34 2.4.12.Applying electrorheologic fluid
35 3.1.1. Transition to bi- and poly-systems
36 4.1.2. Using of copies
37 4.1.3. Sequentially detection of changes
38 4.2.1. Synthesis of measurement SFM
39 4.2.2. Transition to complex measuring SFM
40 4.2.3. Transition to external complex measuring SFM
41 4.2.4. Transition to measurement SFM by using external environment properties
42 4.3.1. Using physical effects
43 4.3.2. Using resonance oscillation of object
44 4.3.3. Using resonance oscillation of attached object
45 4.4.1. Transition to ferromagnetic substance and magnetic fields
46 4.4.2. Transition to measurement feSFM
47 4.4.3. Transition to complex ferromagnetic SFM
48 4.4.4. Transition to measurement feSFM by using external environment properties
49 4.4.5. Using physical effects for measurement feSFM
50 4.5.1. Transition to a measuring bi- and poly-systems
51 4.5.2. Transition to measurement the derivatives of function
52 5.1.1.1.Bypass ways 1
53 5.1.1.2.Bypass ways 2
54 5.1.1.4.Bypass ways 4
55 5.1.1.5.Bypass ways 5
56 5.1.1.6.Bypass ways 6
57 5.1.1.7.Bypass ways 7
58 5.1.1.8.Bypass ways 8
59 5.1.2. Dividing of product
60 5.1.3. Self-elimination of used substances
61 5.1.4. Using of inflatable structures
62 5.2.1. Using present fields (pluralistically)
63 5.3.1. Changing of phase state (Phase transition 1)
64 5.3.2. "Dual" phase state of substance (Phase transition 2)
65 5.3.3. Using phenomena accompanying a phase transition (Phase transition 3)
66 5.3.4. Transition to dual-phase state (Phase transition 4)
67 5.3.5. Using interaction between phases of the system
68 5.4.1. Using reversible physical transformation
69 5.5.1. Substance particles obtaining by decomposition
70 5.5.2. Substance particles obtaining by completing or combining
71 5.5.3. Simple methods for substance particles obtaining

Appendix E

The 11 Separation Methods

The 11 separation methods (SM) are used to solve physical contradictions. This list was taken from the TRIZ Journal ($http://www.inventive-design.net$).
Separation in Space

- SM1. Separation of conflicting properties in space.

Separation in Time

- SM2. Separation of conflicting properties in time.

Separation by system transition

- SM3. Combination of homogeneous or heterogeneous systems into a super-system.
- SM4. Transition from a system to an anti-system, or combination of system with anti-system.
- SM5. The entire system has a property X while its parts have a property opposite to X (anti-X).
- SM6. System transition 2: transition to system that works on the micro-level.

Separation by phase transition

- SM7. Substitution of the phase state of a system's part or external environment.

- SM8. Dual phase state of a system part (using substances capable of converting from one phase to another according to operating conditions).
- SM9. Using of phenomena associated with phase transitions.
- SM10. Substitution of a mono-phase substance with a dual-phase state.

Separation by physical-chemical transition

- SM11. Substance appearance-disappearance as a result of decomposition-combination, ionization-recombination.

Appendix F

The Semantic Similarity Among the TRIZ Knowledge Sources

The semantic similarity among the TRIZ knowledge sources is divided into two kinds: the similarity between inventive principles and inventive standards, and the similarity between inventive principles and separation methods.

F.1 Semantic Similarity Between Inventive Principles and Inventive Standards

There are two types of similar inventive standards for 40 inventive principles, that is, A-class inventive standards: the inventive standards generalized into seven general solutions and B-class inventive standards: the inventive standards identified as the existing inventive principles.

Inventive Principles	Similar Inventive Standards	
	A-class	B-class
InventivePrinciple 1		IS59
InventivePrinciple 2	IS44, IS41, IS62	IS59, IS36
InventivePrinciple 3		IS19, IS55
InventivePrinciple 4		IS19
InventivePrinciple 5		IS35
InventivePrinciple 6	IS51, IS67, IS66	IS63, IS59
InventivePrinciple 7	IS44, IS11, IS40, IS14, IS28	
InventivePrinciple 8	IS11, IS44, IS21	IS67, IS63
InventivePrinciple 9	IS43, IS44, IS12	IS20, IS63
InventivePrinciple 10	IS1, IS15	IS63, IS66, IS67
InventivePrinciple 11	IS16, IS11, IS28	IS36, IS57
InventivePrinciple 12	IS16, IS21, IS1	IS20, IS71
InventivePrinciple 13	IS7, IS44	IS53, IS6, IS55
InventivePrinciple 14	IS43, IS44, IS14, IS1	IS59
InventivePrinciple 15		IS30
InventivePrinciple 16		IS6, IS50
InventivePrinciple 17	IS28, IS41, IS4	IS59, IS67
InventivePrinciple 18		IS20, IS32
InventivePrinciple 19		IS18, IS32, IS37
InventivePrinciple 20		IS22
InventivePrinciple 21	IS1, IS43, IS44	IS20, IS65
InventivePrinciple 22	IS62, IS43, IS44	IS61, IS63
InventivePrinciple 23		IS30, IS68
InventivePrinciple 24	IS43, IS1, IS44	IS59, IS70
InventivePrinciple 25	IS44	IS63, IS65, IS20, IS67
InventivePrinciple 26		IS36, IS57
InventivePrinciple 27	IS1, IS44, IS43	IS36, IS63
InventivePrinciple 28		IS48, IS53
InventivePrinciple 29		IS52, IS61
InventivePrinciple 30	IS28, IS11, IS51	IS53, IS67
InventivePrinciple 31		IS36, IS57, IS67, IS63, IS61
InventivePrinciple 32	IS62, IS43, IS44	IS63, IS67
InventivePrinciple 33	IS11, IS28	IS36, IS57, IS59
InventivePrinciple 34		IS60
InventivePrinciple 35		IS34, IS63
InventivePrinciple 36	IS29 IS64 IS65 IS66 IS67	
InventivePrinciple 37	IS28, IS43	IS63, IS67, IS65
InventivePrinciple 38		IS54, IS69, IS70, IS71
InventivePrinciple 39	IS44, IS62	IS59, IS63, IS67
InventivePrinciple 40	IS3	IS36, IS57, IS63, IS61

F.2 Semantic Similarity Between Inventive Principles and Separation Methods

Inventive Principles	Similar Separation Methods
InventivePrinciple 1	SM7, SM6
InventivePrinciple 2	SM7, SM1
InventivePrinciple 3	SM7, SM6
InventivePrinciple 4	SM1, SM6
InventivePrinciple 5	SM7, SM2
InventivePrinciple 6	SM7, SM8
InventivePrinciple 7	SM6, SM1
InventivePrinciple 8	SM2, SM9
InventivePrinciple 9	SM7, SM9
InventivePrinciple 10	SM9, SM6
InventivePrinciple 11	SM2, SM6
InventivePrinciple 12	SM7, SM8
InventivePrinciple 13	SM7, SM6
InventivePrinciple 14	SM7, SM6
InventivePrinciple 15	SM8, SM7
InventivePrinciple 16	SM8, SM9
InventivePrinciple 17	SM6, SM1
InventivePrinciple 18	SM7, SM2
InventivePrinciple 19	SM2, SM6
InventivePrinciple 20	SM7, SM6
InventivePrinciple 21	SM9, SM6
InventivePrinciple 22	SM6, SM7
InventivePrinciple 23	SM8, SM7
InventivePrinciple 24	SM7, SM6
InventivePrinciple 25	SM8, SM7
InventivePrinciple 26	SM9, SM7
InventivePrinciple 27	SM2, SM8

Inventive Principles	Similar Separation Methods
InventivePrinciple 28	SM8, SM7
InventivePrinciple 29	SM6, SM7
InventivePrinciple 30	SM6, SM7
InventivePrinciple 31	SM11, SM8
InventivePrinciple 32	SM8, SM9
InventivePrinciple 33	SM2, SM6
InventivePrinciple 34	SM7, SM6
InventivePrinciple 35	SM6, SM9
InventivePrinciple 36	SM9, SM7
InventivePrinciple 37	SM9, SM7
InventivePrinciple 38	SM8, SM6
InventivePrinciple 39	SM7, SM9
InventivePrinciple 40	SM11, SM1

Appendix G

The SWRL Rules and Their Explanations for Searching Heuristic Abstract Solutions

The 40 SWRL rules for searching heuristic abstract solutions are divided into two kinds: the first 9 rules describing the modification of A-class inventive standards and the other 31 rules for the modification of B-class inventive standards.

Name	SWRL rules and explanations
Rule 1	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 1) \wedge Generic\_Problem\_Model(?z) \wedge$ $correspondsTo\_GPM(?x, ?z) \wedge Generic\_Solution\_Model(?a) \wedge$ $correspondsTo\_GSM(?x, ?a) \wedge Field(?b) \wedge Substance(?c) \wedge$ $hasField\_GPM(?z, ?b) \wedge hasSubstance2\_GPM(?z, ?c) \wedge$ $has\_S1\_Num\_GPM(?z, 0) \wedge has\_S2\_Num\_GPM(?z, 1) \wedge$ $has\_F\_Num\_GPM(?z, 1) \rightarrow hasField\_GSM(?a, ?b) \wedge$ $hasSubstance1\_GSM(?a, added\_S1\_x2) \wedge$ $hasSubstance2\_GSM(?a, ?c) \wedge has\_S1\_Num\_GSM(?a, 1) \wedge$ $has\_S2\_Num\_GSM(?a, 1) \wedge has\_F\_Num\_GSM(?a, 1)$ If the component Substance1 is missing, add one to complete the Su-Field Model.
Rule 2	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 1) \wedge Generic\_Problem\_Model(?z) \wedge$ $correspondsTo\_GPM(?x, ?z) \wedge Generic\_Solution\_Model(?a) \wedge$ $correspondsTo\_GSM(?x, ?a) \wedge Field(?b) \wedge Substance(?c) \wedge$ $hasField\_GPM(?z, ?b) \wedge hasSubstance1\_GPM(?z, ?c) \wedge$ $has\_S1\_Num\_GPM(?z, 1) \wedge has\_S2\_Num\_GPM(?z, 0) \wedge$ $has\_F\_Num\_GPM(?z, 1) \rightarrow hasField\_GSM(?a, ?b) \wedge$ $hasSubstance2\_GSM(?a, added\_S2\_y2) \wedge$ $hasSubstance1\_GSM(?a, ?c) \wedge has\_S1\_Num\_GSM(?a, 1) \wedge$ $has\_S2\_Num\_GSM(?a, 1) \wedge has\_F\_Num\_GSM(?a, 1)$ If the component Substance2 is missing, add one to complete the Su-Field Model.
Rule 3	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 1) \wedge Generic\_Problem\_Model(?z) \wedge$ $correspondsTo\_GPM(?x, ?z) \wedge Generic\_Solution\_Model(?a) \wedge$ $correspondsTo\_GSM(?x, ?a) \wedge Substance(?c) \wedge Substance(?d) \wedge$ $hasSubstance1\_GPM(?z, ?c) \wedge hasSubstance2\_GPM(?z, ?d) \wedge$ $has\_S1\_Num\_GPM(?z, 1) \wedge has\_S2\_Num\_GPM(?z, 1) \wedge$ $has\_F\_Num\_GPM(?z, 0) \rightarrow hasField\_GSM(?a, added\_F\_z2) \wedge$ $hasSubstance1\_GSM(?a, ?c) \wedge hasSubstance2\_GSM(?a, ?d) \wedge$ $has\_S1\_Num\_GSM(?a, 1) \wedge has\_S2\_Num\_GSM(?a, 1) \wedge$ $has\_F\_Num\_GSM(?a, 1)$ If the component Field is missing, add one to complete the Su-Field Model.

Name	SWRL rules and explanations
Rule 4	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 2) \wedge Generic\_Problem\_Model(?z) \wedge$ $correspondsTo\_GPM(?x, ?z) \wedge Generic\_Solution\_Model(?a) \wedge$ $correspondsTo\_GSM(?x, ?a) \wedge Field(?b) \wedge Substance(?c) \wedge$ $Substance(?d) \wedge hasField\_GPM(?z, ?b) \wedge hasSubstance1\_GPM(?z, ?c) \wedge$ $hasSubstance2\_GPM(?z, ?d) \wedge has\_S1\_Num\_GPM(?z, 1) \wedge$ $has\_S2\_Num\_GPM(?z, 1) \wedge has\_F\_Num\_GPM(?z, 1) \rightarrow$ $hasField\_GSM(?a, ?b) \wedge hasSubstance1\_GSM(?a, ?c) \wedge$ $hasSubstance2\_GSM(?a, S2\_with\_modification\_y1) \wedge$ $has\_S1\_Num\_GSM(?a, 1) \wedge has\_S2\_Num\_GSM(?a, 1) \wedge$ $has\_F\_Num\_GSM(?a, 1)$ Modify Substance2 to eliminate or reduce harmful impact.
Rule 5	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 3) \wedge Generic\_Problem\_Model(?z) \wedge$ $correspondsTo\_GPM(?x, ?z) \wedge Generic\_Solution\_Model(?a) \wedge$ $correspondsTo\_GSM(?x, ?a) \wedge Field(?b) \wedge Substance(?c) \wedge$ $Substance(?d) \wedge hasField\_GPM(?z, ?b) \wedge hasSubstance1\_GPM(?z, ?c) \wedge$ $hasSubstance2\_GPM(?z, ?d) \wedge has\_S1\_Num\_GPM(?z, 1) \wedge$ $has\_S2\_Num\_GPM(?z, 1) \wedge has\_F\_Num\_GPM(?z, 1) \rightarrow$ $hasField\_GSM(?a, ?b) \wedge hasSubstance2\_GSM(?a, ?d) \wedge$ $hasSubstance1\_GSM(?a, S1\_with\_modification\_x1) \wedge$ $has\_S1\_Num\_GSM(?a, 1) \wedge has\_S2\_Num\_GSM(?a, 1) \wedge$ $has\_F\_Num\_GSM(?a, 1)$ Modify Substance1 to be less sensitive to harmful impact.
Rule 6	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 4) \wedge Generic\_Problem\_Model(?z) \wedge$ $correspondsTo\_GPM(?x, ?z) \wedge Generic\_Solution\_Model(?a) \wedge$ $correspondsTo\_GSM(?x, ?a) \wedge Field(?b) \wedge Substance(?c) \wedge$ $Substance(?d) \wedge hasField\_GPM(?z, ?b) \wedge hasSubstance1\_GPM(?z, ?c) \wedge$ $hasSubstance2\_GPM(?z, ?d) \wedge has\_S1\_Num\_GPM(?z, 1) \wedge$ $has\_S2\_Num\_GPM(?z, 1) \wedge has\_F\_Num\_GPM(?z, 1) \rightarrow$ $hasSubstance1\_GSM(?a, ?c) \wedge hasSubstance2\_GSM(?a, ?d) \wedge$ $hasField\_GSM(?a, F\_with\_modification\_z1) \wedge$ $has\_S1\_Num\_GSM(?a, 1) \wedge has\_S2\_Num\_GSM(?a, 1) \wedge$ $has\_F\_Num\_GSM(?a, 1)$ Change Field to reduce or eliminate harmful impact.

Name	SWRL rules and explanations
Rule 7	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 5) \wedge Generic\_Problem\_Model(?z) \wedge$ $correspondsTo\_GPM(?x, ?z) \wedge Generic\_Solution\_Model(?a) \wedge$ $correspondsTo\_GSM(?x, ?a) \wedge Field(?b) \wedge Substance(?c) \wedge$ $Substance(?d) \wedge hasField\_GPM(?z, ?b) \wedge hasSubstance1\_GPM(?z, ?c) \wedge$ $hasSubstance2\_GPM(?z, ?d) \wedge has\_S1\_Num\_GPM(?z, 1) \wedge$ $has\_S2\_Num\_GPM(?z, 1) \wedge has\_F\_Num\_GPM(?z, 1) \rightarrow$ $hasSubstance1\_GSM(?a, ?c) \wedge hasSubstance2\_GSM(?a, ?d) \wedge$ $hasField\_GSM(?a, added\_F\_z2) \wedge$ $has\_S1\_Num\_GSM(?a, 1) \wedge has\_S2\_Num\_GSM(?a, 1) \wedge$ $has\_F\_Num\_GSM(?a, 2)$ A counteractive Field is added to eliminate, neutralize or isolate harmful impact.
Rule 8	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 6) \wedge Generic\_Problem\_Model(?z) \wedge$ $correspondsTo\_GPM(?x, ?z) \wedge Generic\_Solution\_Model(?a) \wedge$ $correspondsTo\_GSM(?x, ?a) \wedge Field(?b) \wedge Substance(?c) \wedge$ $Substance(?d) \wedge hasField\_GPM(?z, ?b) \wedge hasSubstance1\_GPM(?z, ?c) \wedge$ $hasSubstance2\_GPM(?z, ?d) \wedge has\_S1\_Num\_GPM(?z, 1) \wedge$ $has\_S2\_Num\_GPM(?z, 1) \wedge has\_F\_Num\_GPM(?z, 1) \rightarrow$ $hasSubstance1\_GSM(?a, ?c) \wedge hasSubstance2\_GSM(?a, ?d) \wedge$ $hasField\_GSM(?a, added\_F\_z2) \wedge$ $has\_S1\_Num\_GSM(?a, 1) \wedge has\_S2\_Num\_GSM(?a, 1) \wedge$ $has\_F\_Num\_GSM(?a, 2)$ A positive Field is introduced to increase useful effect.
Rule 9	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 7) \wedge Generic\_Problem\_Model(?z) \wedge$ $correspondsTo\_GPM(?x, ?z) \wedge Generic\_Solution\_Model(?a) \wedge$ $correspondsTo\_GSM(?x, ?a) \wedge Field(?b) \wedge Substance(?c) \wedge$ $Substance(?d) \wedge hasField\_GPM(?z, ?b) \wedge hasSubstance1\_GPM(?z, ?c) \wedge$ $hasSubstance2\_GPM(?z, ?d) \wedge has\_S1\_Num\_GPM(?z, 1) \wedge$ $has\_S2\_Num\_GPM(?z, 1) \wedge has\_F\_Num\_GPM(?z, 1) \rightarrow$ $hasSubstance1\_GSM(?a, added\_S1\_x2) \wedge$ $hasSubstance2\_GSM(?a, added\_S2\_y2) \wedge$ $hasField\_GSM(?a, added\_F\_z2) \wedge$ $has\_S1\_Num\_GSM(?a, 2) \wedge has\_S2\_Num\_GSM(?a, 2) \wedge$ $has\_F\_Num\_GSM(?a, 2)$ Expand the existing Su-Field Model by introduce new substances or Field.

Name	SWRL rules and explanations
Rule 10	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 116) \wedge Generic\_Problem\_Model(?z) \wedge$ $correspondsTo\_GPM(?x, ?z) \wedge Generic\_Solution\_Model(?a) \wedge$ $correspondsTo\_GSM(?x, ?a) \wedge Field(?b) \wedge Substance(?c) \wedge$ $Substance(?d) \wedge hasField\_GPM(?z, ?b) \wedge hasSubstance1\_GPM(?z, ?c) \wedge$ $hasSubstance2\_GPM(?z, ?d) \wedge has\_S1\_Num\_GPM(?z, 1) \wedge$ $has\_S2\_Num\_GPM(?z, 1) \wedge has\_F\_Num\_GPM(?z, 1) \rightarrow$ $hasSubstance1\_GSM(?a, ?c) \wedge hasSubstance2\_GSM(?a, ?d) \wedge$ $hasField\_GSM(?a, added\_F\_z2) \wedge$ $has\_S1\_Num\_GSM(?a, 1) \wedge has\_S2\_Num\_GSM(?a, 1) \wedge$ $has\_F\_Num\_GSM(?a, 2)$ A surplus Field is introduced to remove the surplus substance.
Rule 11	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 225) \wedge Generic\_Problem\_Model(?z) \wedge$ $correspondsTo\_GPM(?x, ?z) \wedge Generic\_Solution\_Model(?a) \wedge$ $correspondsTo\_GSM(?x, ?a) \wedge Field(?b) \wedge Substance(?c) \wedge$ $Substance(?d) \wedge hasField\_GPM(?z, ?b) \wedge hasSubstance1\_GPM(?z, ?c) \wedge$ $hasSubstance2\_GPM(?z, ?d) \wedge has\_S1\_Num\_GPM(?z, 1) \wedge$ $has\_S2\_Num\_GPM(?z, 1) \wedge has\_F\_Num\_GPM(?z, 1) \rightarrow$ $hasSubstance1\_GSM(?a, ?c) \wedge hasSubstance2\_GSM(?a, ?d) \wedge$ $hasField\_GSM(?a,$ $F\_with\_definite\_spatial-temporal\_structure\_z4) \wedge$ $has\_S1\_Num\_GSM(?a, 1) \wedge has\_S2\_Num\_GSM(?a, 1) \wedge$ $has\_F\_Num\_GSM(?a, 1)$ Transition from Field with a disordered structure to non-uniform one with definite spatial-temporal structure.
Rule 12	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 226) \wedge Generic\_Problem\_Model(?z) \wedge$ $correspondsTo\_GPM(?x, ?z) \wedge Generic\_Solution\_Model(?a) \wedge$ $correspondsTo\_GSM(?x, ?a) \wedge Field(?b) \wedge Substance(?c) \wedge$ $Substance(?d) \wedge hasField\_GPM(?z, ?b) \wedge hasSubstance1\_GPM(?z, ?c) \wedge$ $hasSubstance2\_GPM(?z, ?d) \wedge has\_S1\_Num\_GPM(?z, 1) \wedge$ $has\_S2\_Num\_GPM(?z, 1) \wedge has\_F\_Num\_GPM(?z, 1) \rightarrow$ $hasSubstance1\_GSM(?a, ?c) \wedge hasField\_GSM(?a, ?b) \wedge$ $hasSubstance2\_GSM(?a,$ $S2\_with\_predefined\_spatial-temporal\_structure\_y4) \wedge$ $has\_S1\_Num\_GSM(?a, 1) \wedge has\_S2\_Num\_GSM(?a, 1) \wedge$ $has\_F\_Num\_GSM(?a, 1)$ Transition from Substance2 with a disordered structure to non-uniform one with predefined spatial-temporal structure.

Name	SWRL rules and explanations
Rule 13	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 231) \wedge Generic\_Problem\_Model(?z) \wedge$ $correspondsTo\_GPM(?x, ?z) \wedge Generic\_Solution\_Model(?a) \wedge$ $correspondsTo\_GSM(?x, ?a) \wedge Field(?b) \wedge Substance(?c) \wedge$ $Substance(?d) \wedge hasField\_GPM(?z, ?b) \wedge hasSubstance1\_GPM(?z, ?c) \wedge$ $hasSubstance2\_GPM(?z, ?d) \wedge has\_S1\_Num\_GPM(?z, 1) \wedge$ $has\_S2\_Num\_GPM(?z, 1) \wedge has\_F\_Num\_GPM(?z, 1) \rightarrow$ $hasSubstance1\_GSM(?a, ?c) \wedge$ $hasSubstance2\_GSM(?a, S2\_matching\_rhythm\_with\_Field\_y21) \wedge$ $hasField\_GSM(?a, F\_matching\_rhythm\_with\_Substance\_z21) \wedge$ $has\_S1\_Num\_GSM(?a, 1) \wedge has\_S2\_Num\_GSM(?a, 1) \wedge$ $has\_F\_Num\_GSM(?a, 1)$ Matching the rhythm of Field and Substance2.
Rule 14	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 233) \wedge Generic\_Problem\_Model(?z) \wedge$ $correspondsTo\_GPM(?x, ?z) \wedge Generic\_Solution\_Model(?a) \wedge$ $correspondsTo\_GSM(?x, ?a) \wedge Field(?b) \wedge Substance(?c) \wedge$ $Substance(?d) \wedge hasField\_GPM(?z, ?b) \wedge hasSubstance1\_GPM(?z, ?c) \wedge$ $hasSubstance2\_GPM(?z, ?d) \wedge has\_S1\_Num\_GPM(?z, 1) \wedge$ $has\_S2\_Num\_GPM(?z, 1) \wedge has\_F\_Num\_GPM(?z, 1) \rightarrow$ $hasSubstance1\_GSM(?a, ?c) \wedge hasSubstance2\_GSM(?a, ?d) \wedge$ $hasField\_GSM(?a, F\_with\_compatible\_independent\_actions\_z22) \wedge$ $has\_S1\_Num\_GSM(?a, 1) \wedge has\_S2\_Num\_GSM(?a, 1) \wedge$ $has\_F\_Num\_GSM(?a, 1)$ Use compatible Fields instead of incompatible ones.
Rule 15	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 247) \wedge Generic\_Problem\_Model(?z) \wedge$ $correspondsTo\_GPM(?x, ?z) \wedge Generic\_Solution\_Model(?a) \wedge$ $correspondsTo\_GSM(?x, ?a) \wedge Field(?b) \wedge Substance(?c) \wedge$ $Substance(?d) \wedge hasField\_GPM(?z, ?b) \wedge hasSubstance1\_GPM(?z, ?c) \wedge$ $hasSubstance2\_GPM(?z, ?d) \wedge has\_S1\_Num\_GPM(?z, 1) \wedge$ $has\_S2\_Num\_GPM(?z, 1) \wedge has\_F\_Num\_GPM(?z, 1) \rightarrow$ $hasField\_GSM(?a, ?b) \wedge hasSubstance1\_GSM(?a,$ $S1\_with\_phase\_transition\_by\_using\_physical\_phenomena\_x14) \wedge$ $hasSubstance2\_GSM(?a,$ $S2\_with\_phase\_transition\_by\_using\_physical\_phenomena\_y14) \wedge$ $has\_S1\_Num\_GSM(?a, 1) \wedge has\_S2\_Num\_GSM(?a, 1) \wedge$ $has\_F\_Num\_GSM(?a, 1)$ In feSFM, use physical phenomena for the phase transition of Substances.

Name	SWRL rules and explanations
Rule 16	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 248) \wedge Generic\_Problem\_Model(?z) \wedge$ $correspondsTo\_GPM(?x, ?z) \wedge Generic\_Solution\_Model(?a) \wedge$ $correspondsTo\_GSM(?x, ?a) \wedge Field(?b) \wedge Substance(?c) \wedge$ $Substance(?d) \wedge hasField\_GPM(?z, ?b) \wedge hasSubstance1\_GPM(?z, ?c) \wedge$ $hasSubstance2\_GPM(?z, ?d) \wedge has\_S1\_Num\_GPM(?z, 1) \wedge$ $has\_S2\_Num\_GPM(?z, 1) \wedge has\_F\_Num\_GPM(?z, 1) \rightarrow$ $hasSubstance1\_GSM(?a, S1\_with\_high\_degree\_of\_dynamism\_x23) \wedge$ $hasSubstance2\_GSM(?a, S2\_with\_high\_degree\_of\_dynamism\_y23) \wedge$ $hasField\_GSM(?a, F\_with\_high\_degree\_of\_dynamism\_z23) \wedge$ $has\_S1\_Num\_GSM(?a, 1) \wedge has\_S2\_Num\_GSM(?a, 1) \wedge$ $has\_F\_Num\_GSM(?a, 1)$ In feSFM, increase the degree of dynamism of Substances and Field.
Rule 17	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 2410) \wedge Generic\_Problem\_Model(?z) \wedge$ $correspondsTo\_GPM(?x, ?z) \wedge Generic\_Solution\_Model(?a) \wedge$ $correspondsTo\_GSM(?x, ?a) \wedge Field(?b) \wedge Substance(?c) \wedge$ $Substance(?d) \wedge hasField\_GPM(?z, ?b) \wedge hasSubstance1\_GPM(?z, ?c) \wedge$ $hasSubstance2\_GPM(?z, ?d) \wedge has\_S1\_Num\_GPM(?z, 1) \wedge$ $has\_S2\_Num\_GPM(?z, 1) \wedge has\_F\_Num\_GPM(?z, 1) \rightarrow$ $hasSubstance1\_GSM(?a, ?c) \wedge$ $hasSubstance2\_GSM(?a, S2\_matching\_rhythm\_with\_Field\_y21) \wedge$ $hasField\_GSM(?a, F\_matching\_rhythm\_with\_Substance\_z21) \wedge$ $has\_S1\_Num\_GSM(?a, 1) \wedge has\_S2\_Num\_GSM(?a, 1) \wedge$ $has\_F\_Num\_GSM(?a, 1)$ Match the rhythms of elements of feSFM.
Rule 18	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 2412) \wedge Generic\_Problem\_Model(?z) \wedge$ $correspondsTo\_GPM(?x, ?z) \wedge Generic\_Solution\_Model(?a) \wedge$ $correspondsTo\_GSM(?x, ?a) \wedge Field(?b) \wedge Substance(?c) \wedge$ $Substance(?d) \wedge hasField\_GPM(?z, ?b) \wedge hasSubstance1\_GPM(?z, ?c) \wedge$ $hasSubstance2\_GPM(?z, ?d) \wedge has\_S1\_Num\_GPM(?z, 1) \wedge$ $has\_S2\_Num\_GPM(?z, 1) \wedge has\_F\_Num\_GPM(?z, 1) \rightarrow$ $hasField\_GSM(?a, ?b) \wedge$ $hasSubstance1\_GSM(?a, S1\_to\_be\_electrorheologic\_fluid\_x24) \wedge$ $hasSubstance2\_GSM(?a, S2\_to\_be\_electrorheologic\_fluid\_y24) \wedge$ $has\_S1\_Num\_GSM(?a, 1) \wedge has\_S2\_Num\_GSM(?a, 1) \wedge$ $has\_F\_Num\_GSM(?a, 1)$ Apply electrorheologic fluid instead of magnetic fluid in feSFM.

Name	SWRL rules and explanations
Rule 19	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 311) \wedge Generic\_Problem\_Model(?z) \wedge$ $correspondsTo\_GPM(?x, ?z) \wedge Generic\_Solution\_Model(?a) \wedge$ $correspondsTo\_GSM(?x, ?a) \wedge Field(?b) \wedge Substance(?c) \wedge$ $Substance(?d) \wedge hasField\_GPM(?z, ?b) \wedge hasSubstance1\_GPM(?z, ?c) \wedge$ $hasSubstance2\_GPM(?z, ?d) \wedge has\_S1\_Num\_GPM(?z, 1) \wedge$ $has\_S2\_Num\_GPM(?z, 1) \wedge has\_F\_Num\_GPM(?z, 1) \rightarrow$ $hasField\_GSM(?a, ?b) \wedge$ $hasSubstance1\_GSM(?a, added\_S1\_x2) \wedge$ $hasSubstance2\_GSM(?a, added\_S2\_y2) \wedge$ $has\_S1\_Num\_GSM(?a, 2) \wedge has\_S2\_Num\_GSM(?a, 2) \wedge$ $has\_F\_Num\_GSM(?a, 1)$ Combine with other systems to form a new super-system.
Rule 20	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 412) \wedge Generic\_Problem\_Model(?z) \wedge$ $correspondsTo\_GPM(?x, ?z) \wedge Generic\_Solution\_Model(?a) \wedge$ $correspondsTo\_GSM(?x, ?a) \wedge Field(?b) \wedge Substance(?c) \wedge$ $Substance(?d) \wedge hasField\_GPM(?z, ?b) \wedge hasSubstance1\_GPM(?z, ?c) \wedge$ $hasSubstance2\_GPM(?z, ?d) \wedge has\_S1\_Num\_GPM(?z, 1) \wedge$ $has\_S2\_Num\_GPM(?z, 1) \wedge has\_F\_Num\_GPM(?z, 1) \rightarrow$ $hasField\_GSM(?a, ?b) \wedge hasSubstance2\_GSM(?a, ?d) \wedge$ $hasSubstance1\_GSM(?a, copy\_of\_S1\_X5) \wedge$ $has\_S1\_Num\_GSM(?a, 1) \wedge has\_S2\_Num\_GSM(?a, 1) \wedge$ $has\_F\_Num\_GSM(?a, 1)$ Use the copy of Substance to be measured.
Rule 21	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 413) \wedge Generic\_Problem\_Model(?z) \wedge$ $correspondsTo\_GPM(?x, ?z) \wedge Generic\_Solution\_Model(?a) \wedge$ $correspondsTo\_GSM(?x, ?a) \wedge Field(?b) \wedge Substance(?c) \wedge$ $Substance(?d) \wedge hasField\_GPM(?z, ?b) \wedge hasSubstance1\_GPM(?z, ?c) \wedge$ $hasSubstance2\_GPM(?z, ?d) \wedge has\_S1\_Num\_GPM(?z, 1) \wedge$ $has\_S2\_Num\_GPM(?z, 1) \wedge has\_F\_Num\_GPM(?z, 1) \rightarrow$ $hasSubstance1\_GSM(?a, ?c) \wedge hasSubstance2\_GSM(?a, ?d) \wedge$ $hasField\_GSM(?a,$ $F\_with\_definite\_spatial-temporal\_structure\_z4) \wedge$ $has\_S1\_Num\_GSM(?a, 1) \wedge has\_S2\_Num\_GSM(?a, 1) \wedge$ $has\_F\_Num\_GSM(?a, 1)$ For measurement problem, the sequentially detection of changes is used.

Name	SWRL rules and explanations
Rule 22	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 444) \wedge Generic\_Problem\_Model(?z) \wedge$ $correspondsTo\_GPM(?x, ?z) \wedge Generic\_Solution\_Model(?a) \wedge$ $correspondsTo\_GSM(?x, ?a) \wedge Field(?b) \wedge Substance(?c) \wedge$ $Substance(?d) \wedge hasField\_GPM(?z, ?b) \wedge hasSubstance1\_GPM(?z, ?c) \wedge$ $hasSubstance2\_GPM(?z, ?d) \wedge has\_S1\_Num\_GPM(?z, 1) \wedge$ $has\_S2\_Num\_GPM(?z, 1) \wedge has\_F\_Num\_GPM(?z, 1) \rightarrow$ $hasSubstance1\_GSM(?a, ?c) \wedge hasSubstance2\_GSM(?a, ?d) \wedge$ $hasField\_GSM(?a, added\_F\_z2) \wedge$ $has\_S1\_Num\_GSM(?a, 1) \wedge has\_S2\_Num\_GSM(?a, 1) \wedge$ $has\_F\_Num\_GSM(?a, 1)$ For measurement problem of feSFM, the external environment preoperties are used.
Rule 23	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 451) \wedge Generic\_Problem\_Model(?z) \wedge$ $correspondsTo\_GPM(?x, ?z) \wedge Generic\_Solution\_Model(?a) \wedge$ $correspondsTo\_GSM(?x, ?a) \wedge Field(?b) \wedge Substance(?c) \wedge$ $Substance(?d) \wedge hasField\_GPM(?z, ?b) \wedge hasSubstance1\_GPM(?z, ?c) \wedge$ $hasSubstance2\_GPM(?z, ?d) \wedge has\_S1\_Num\_GPM(?z, 1) \wedge$ $has\_S2\_Num\_GPM(?z, 1) \wedge has\_F\_Num\_GPM(?z, 1) \rightarrow$ $hasField\_GSM(?a, ?b) \wedge hasSubstance2\_GSM(?a, ?d) \wedge$ $hasSubstance1\_GSM(?a, added\_S1\_x2) \wedge$ $has\_S1\_Num\_GSM(?a, 2) \wedge has\_S2\_Num\_GSM(?a, 1) \wedge$ $has\_F\_Num\_GSM(?a, 1)$ For measurement problem, combine another system to form a bi- or poly-system.
Rule 24	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 5111) \wedge Generic\_Problem\_Model(?z) \wedge$ $correspondsTo\_GPM(?x, ?z) \wedge Generic\_Solution\_Model(?a) \wedge$ $correspondsTo\_GSM(?x, ?a) \wedge Field(?b) \wedge Substance(?c) \wedge$ $Substance(?d) \wedge hasField\_GPM(?z, ?b) \wedge hasSubstance1\_GPM(?z, ?c) \wedge$ $hasSubstance2\_GPM(?z, ?d) \wedge has\_S1\_Num\_GPM(?z, 1) \wedge$ $has\_S2\_Num\_GPM(?z, 1) \wedge has\_F\_Num\_GPM(?z, 1) \rightarrow$ $hasField\_GSM(?a, ?b) \wedge$ $hasSubstance1\_GSM(?a, S1\_in\_void\_form\_x6) \wedge$ $hasSubstance2\_GSM(?a, S2\_in\_void\_form\_y6) \wedge$ $has\_S1\_Num\_GSM(?a, 1) \wedge has\_S2\_Num\_GSM(?a, 1) \wedge$ $has\_F\_Num\_GSM(?a, 1)$ Substance in the form of "void".

Name	SWRL rules and explanations
Rule 25	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 5112) \wedge Generic\_Problem\_Model(?z) \wedge$ $correspondsTo\_GPM(?x, ?z) \wedge Generic\_Solution\_Model(?a) \wedge$ $correspondsTo\_GSM(?x, ?a) \wedge Field(?b) \wedge Substance(?c) \wedge$ $Substance(?d) \wedge hasField\_GPM(?z, ?b) \wedge hasSubstance1\_GPM(?z, ?c) \wedge$ $hasSubstance2\_GPM(?z, ?d) \wedge has\_S1\_Num\_GPM(?z, 1) \wedge$ $has\_S2\_Num\_GPM(?z, 1) \wedge has\_F\_Num\_GPM(?z, 1) \rightarrow$ $hasSubstance1\_GSM(?a, ?c) \wedge hasSubstance2\_GSM(?a, ?d) \wedge$ $hasField\_GSM(?a, added\_F\_z2) \wedge$ $has\_S1\_Num\_GSM(?a, 1) \wedge has\_S2\_Num\_GSM(?a, 1) \wedge$ $has\_F\_Num\_GSM(?a, 1)$ A Field is used instead of a Substance.
Rule 26	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 5114) \wedge Generic\_Problem\_Model(?z) \wedge$ $correspondsTo\_GPM(?x, ?z) \wedge Generic\_Solution\_Model(?a) \wedge$ $correspondsTo\_GSM(?x, ?a) \wedge Field(?b) \wedge Substance(?c) \wedge$ $Substance(?d) \wedge hasField\_GPM(?z, ?b) \wedge hasSubstance1\_GPM(?z, ?c) \wedge$ $hasSubstance2\_GPM(?z, ?d) \wedge has\_S1\_Num\_GPM(?z, 1) \wedge$ $has\_S2\_Num\_GPM(?z, 1) \wedge has\_F\_Num\_GPM(?z, 1) \rightarrow$ $hasField\_GSM(?a, ?b) \wedge$ $hasSubstance1\_GSM(?a, S1\_in\_small\_quantities\_x7) \wedge$ $hasSubstance2\_GSM(?a, S2\_in\_small\_quantities\_y7) \wedge$ $has\_S1\_Num\_GSM(?a, 1) \wedge has\_S2\_Num\_GSM(?a, 1) \wedge$ $has\_F\_Num\_GSM(?a, 1)$ Substances in small quantities are used.
Rule 27	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 5115) \wedge Generic\_Problem\_Model(?z) \wedge$ $correspondsTo\_GPM(?x, ?z) \wedge Generic\_Solution\_Model(?a) \wedge$ $correspondsTo\_GSM(?x, ?a) \wedge Field(?b) \wedge Substance(?c) \wedge$ $Substance(?d) \wedge hasField\_GPM(?z, ?b) \wedge hasSubstance1\_GPM(?z, ?c) \wedge$ $hasSubstance2\_GPM(?z, ?d) \wedge has\_S1\_Num\_GPM(?z, 1) \wedge$ $has\_S2\_Num\_GPM(?z, 1) \wedge has\_F\_Num\_GPM(?z, 1) \rightarrow$ $hasField\_GSM(?a, ?b) \wedge$ $hasSubstance1\_GSM(?a,$ $S1\_in\_small\_quantities\_and\_concentrated\_in\_certain\_parts\_x8) \wedge$ $hasSubstance2\_GSM(?a,$ $S2\_in\_small\_quantities\_and\_concentrated\_in\_certain\_parts\_y8) \wedge$ $has\_S1\_Num\_GSM(?a, 1) \wedge has\_S2\_Num\_GSM(?a, 1) \wedge$ $has\_F\_Num\_GSM(?a, 1)$ Substances in small quantities and concentrated in certain parts are used.

Name	SWRL rules and explanations
Rule 28	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 5117) \wedge Generic\_Problem\_Model(?z) \wedge$ $correspondsTo\_GPM(?x, ?z) \wedge Generic\_Solution\_Model(?a) \wedge$ $correspondsTo\_GSM(?x, ?a) \wedge Field(?b) \wedge Substance(?c) \wedge$ $Substance(?d) \wedge hasField\_GPM(?z, ?b) \wedge hasSubstance1\_GPM(?z, ?c) \wedge$ $hasSubstance2\_GPM(?z, ?d) \wedge has\_S1\_Num\_GPM(?z, 1) \wedge$ $has\_S2\_Num\_GPM(?z, 1) \wedge has\_F\_Num\_GPM(?z, 1) \rightarrow$ $hasField\_GSM(?a, ?b) \wedge$ $hasSubstance1\_GSM(?a, copy\_of\_S1\_X5) \wedge$ $hasSubstance2\_GSM(?a, copy\_of\_S2\_y5) \wedge$ $has\_S1\_Num\_GSM(?a, 1) \wedge has\_S2\_Num\_GSM(?a, 1) \wedge$ $has\_F\_Num\_GSM(?a, 1)$ A copy of Substance is used instead of itself.
Rule 29	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 512) \wedge Generic\_Problem\_Model(?z) \wedge$ $correspondsTo\_GPM(?x, ?z) \wedge Generic\_Solution\_Model(?a) \wedge$ $correspondsTo\_GSM(?x, ?a) \wedge Field(?b) \wedge Substance(?c) \wedge$ $Substance(?d) \wedge hasField\_GPM(?z, ?b) \wedge hasSubstance1\_GPM(?z, ?c) \wedge$ $hasSubstance2\_GPM(?z, ?d) \wedge has\_S1\_Num\_GPM(?z, 1) \wedge$ $has\_S2\_Num\_GPM(?z, 1) \wedge has\_F\_Num\_GPM(?z, 1) \rightarrow$ $hasField\_GSM(?a, ?b) \wedge$ $hasSubstance1\_GSM(?a, product\_divided\_instead\_of\_S1\_x9) \wedge$ $hasSubstance2\_GSM(?a, product\_divided\_instead\_of\_S2\_y9) \wedge$ $has\_S1\_Num\_GSM(?a, 1) \wedge has\_S2\_Num\_GSM(?a, 1) \wedge$ $has\_F\_Num\_GSM(?a, 1)$ The Substance, divided into several interacting parts, is used.
Rule 30	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 512) \wedge Generic\_Problem\_Model(?z) \wedge$ $correspondsTo\_GPM(?x, ?z) \wedge Generic\_Solution\_Model(?a) \wedge$ $correspondsTo\_GSM(?x, ?a) \wedge Field(?b) \wedge Substance(?c) \wedge$ $Substance(?d) \wedge hasField\_GPM(?z, ?b) \wedge hasSubstance1\_GPM(?z, ?c) \wedge$ $hasSubstance2\_GPM(?z, ?d) \wedge has\_S1\_Num\_GPM(?z, 1) \wedge$ $has\_S2\_Num\_GPM(?z, 1) \wedge has\_F\_Num\_GPM(?z, 1) \rightarrow$ $hasField\_GSM(?a, ?b) \wedge$ $hasSubstance1\_GSM(?a, S1\_with\_ability\_of\_disappearing\_x10) \wedge$ $hasSubstance2\_GSM(?a, S2\_with\_ability\_of\_disappearing\_y10) \wedge$ $has\_S1\_Num\_GSM(?a, 1) \wedge has\_S2\_Num\_GSM(?a, 1) \wedge$ $has\_F\_Num\_GSM(?a, 1)$ Substance which can disappear after its function is used.

Name	SWRL rules and explanations
Rule 31	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 514) \wedge Generic\_Problem\_Model(?z) \wedge$ $correspondsTo\_GPM(?x, ?z) \wedge Generic\_Solution\_Model(?a) \wedge$ $correspondsTo\_GSM(?x, ?a) \wedge Field(?b) \wedge Substance(?c) \wedge$ $Substance(?d) \wedge hasField\_GPM(?z, ?b) \wedge hasSubstance1\_GPM(?z, ?c) \wedge$ $hasSubstance2\_GPM(?z, ?d) \wedge has\_S1\_Num\_GPM(?z, 1) \wedge$ $has\_S2\_Num\_GPM(?z, 1) \wedge has\_F\_Num\_GPM(?z, 1) \rightarrow$ $hasField\_GSM(?a, ?b) \wedge$ $hasSubstance1\_GSM(?a, S1\_in\_form\_of\_inflatable\_structure\_x11) \wedge$ $hasSubstance2\_GSM(?a, S2\_in\_form\_of\_inflatable\_structure\_y11) \wedge$ $has\_S1\_Num\_GSM(?a, 1) \wedge has\_S2\_Num\_GSM(?a, 1) \wedge$ $has\_F\_Num\_GSM(?a, 1)$ Using Substance with inflatable structure.
Rule 32	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 531) \wedge Generic\_Problem\_Model(?z) \wedge$ $correspondsTo\_GPM(?x, ?z) \wedge Generic\_Solution\_Model(?a) \wedge$ $correspondsTo\_GSM(?x, ?a) \wedge Field(?b) \wedge Substance(?c) \wedge$ $Substance(?d) \wedge hasField\_GPM(?z, ?b) \wedge hasSubstance1\_GPM(?z, ?c) \wedge$ $hasSubstance2\_GPM(?z, ?d) \wedge has\_S1\_Num\_GPM(?z, 1) \wedge$ $has\_S2\_Num\_GPM(?z, 1) \wedge has\_F\_Num\_GPM(?z, 1) \rightarrow$ $hasField\_GSM(?a, ?b) \wedge$ $hasSubstance1\_GSM(?a, S1\_in\_different\_phases\_x12) \wedge$ $hasSubstance2\_GSM(?a, S2\_in\_different\_phases\_y12) \wedge$ $has\_S1\_Num\_GSM(?a, 1) \wedge has\_S2\_Num\_GSM(?a, 1) \wedge$ $has\_F\_Num\_GSM(?a, 1)$ Change the phase state of Substance.
Rule 33	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 532) \wedge Generic\_Problem\_Model(?z) \wedge$ $correspondsTo\_GPM(?x, ?z) \wedge Generic\_Solution\_Model(?a) \wedge$ $correspondsTo\_GSM(?x, ?a) \wedge Field(?b) \wedge Substance(?c) \wedge$ $Substance(?d) \wedge hasField\_GPM(?z, ?b) \wedge hasSubstance1\_GPM(?z, ?c) \wedge$ $hasSubstance2\_GPM(?z, ?d) \wedge has\_S1\_Num\_GPM(?z, 1) \wedge$ $has\_S2\_Num\_GPM(?z, 1) \wedge has\_F\_Num\_GPM(?z, 1) \rightarrow$ $hasField\_GSM(?a, ?b) \wedge$ $hasSubstance1\_GSM(?a, S1\_with\_ability\_of\_changing\_phase\_x13) \wedge$ $hasSubstance2\_GSM(?a, S2\_with\_ability\_of\_changing\_phase\_y13) \wedge$ $has\_S1\_Num\_GSM(?a, 1) \wedge has\_S2\_Num\_GSM(?a, 1) \wedge$ $has\_F\_Num\_GSM(?a, 1)$ Use the dual phase state of Substance.

Name	SWRL rules and explanations
Rule 34	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 533) \wedge Generic\_Problem\_Model(?z) \wedge$ $correspondsTo\_GPM(?x, ?z) \wedge Generic\_Solution\_Model(?a) \wedge$ $correspondsTo\_GSM(?x, ?a) \wedge Field(?b) \wedge Substance(?c) \wedge$ $Substance(?d) \wedge hasField\_GPM(?z, ?b) \wedge hasSubstance1\_GPM(?z, ?c) \wedge$ $hasSubstance2\_GPM(?z, ?d) \wedge has\_S1\_Num\_GPM(?z, 1) \wedge$ $has\_S2\_Num\_GPM(?z, 1) \wedge has\_F\_Num\_GPM(?z, 1) \rightarrow$ $hasField\_GSM(?a, ?b) \wedge$ $hasSubstance1\_GSM(?a,$ $S1\_with\_phase\_transition\_by\_using\_physical\_phenomena_x14) \wedge$ $hasSubstance2\_GSM(?a,$ $S2\_with\_phase\_transition\_by\_using\_physical\_phenomena\_y14) \wedge$ $has\_S1\_Num\_GSM(?a, 1) \wedge has\_S2\_Num\_GSM(?a, 1) \wedge$ $has\_F\_Num\_GSM(?a, 1)$ Use phenomena accompanying a phase transition.
Rule 35	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 534) \wedge Generic\_Problem\_Model(?z) \wedge$ $correspondsTo\_GPM(?x, ?z) \wedge Generic\_Solution\_Model(?a) \wedge$ $correspondsTo\_GSM(?x, ?a) \wedge Field(?b) \wedge Substance(?c) \wedge$ $Substance(?d) \wedge hasField\_GPM(?z, ?b) \wedge hasSubstance1\_GPM(?z, ?c) \wedge$ $hasSubstance2\_GPM(?z, ?d) \wedge has\_S1\_Num\_GPM(?z, 1) \wedge$ $has\_S2\_Num\_GPM(?z, 1) \wedge has\_F\_Num\_GPM(?z, 1) \rightarrow$ $hasField\_GSM(?a, ?b) \wedge$ $hasSubstance1\_GSM(?a, S1\_with\_dual-phase\_state\_x15) \wedge$ $hasSubstance2\_GSM(?a, S2\_with\_dual-phase\_state\_y15) \wedge$ $has\_S1\_Num\_GSM(?a, 1) \wedge has\_S2\_Num\_GSM(?a, 1) \wedge$ $has\_F\_Num\_GSM(?a, 1)$ Use Substance with dual-phase state.
Rule 36	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 535) \wedge Generic\_Problem\_Model(?z) \wedge$ $correspondsTo\_GPM(?x, ?z) \wedge Generic\_Solution\_Model(?a) \wedge$ $correspondsTo\_GSM(?x, ?a) \wedge Field(?b) \wedge Substance(?c) \wedge$ $Substance(?d) \wedge hasField\_GPM(?z, ?b) \wedge hasSubstance1\_GPM(?z, ?c) \wedge$ $hasSubstance2\_GPM(?z, ?d) \wedge has\_S1\_Num\_GPM(?z, 1) \wedge$ $has\_S2\_Num\_GPM(?z, 1) \wedge has\_F\_Num\_GPM(?z, 1) \rightarrow$ $hasField\_GSM(?a, ?b) \wedge$ $hasSubstance1\_GSM(?a,$ $S1\_with\_dual-phase\_state\_and\_interaction\_x16) \wedge$ $hasSubstance2\_GSM(?a,$ $S2\_with\_dual-phase\_state\_and\_interaction\_y16) \wedge$ $has\_S1\_Num\_GSM(?a, 1) \wedge has\_S2\_Num\_GSM(?a, 1) \wedge$ $has\_F\_Num\_GSM(?a, 1)$ Use Substance with interacting dual-phase state.

Name	SWRL rules and explanations
Rule 37	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 541) \wedge Generic\_Problem\_Model(?z) \wedge$ $correspondsTo\_GPM(?x, ?z) \wedge Generic\_Solution\_Model(?a) \wedge$ $correspondsTo\_GSM(?x, ?a) \wedge Field(?b) \wedge Substance(?c) \wedge$ $Substance(?d) \wedge hasField\_GPM(?z, ?b) \wedge hasSubstance1\_GPM(?z, ?c) \wedge$ $hasSubstance2\_GPM(?z, ?d) \wedge has\_S1\_Num\_GPM(?z, 1) \wedge$ $has\_S2\_Num\_GPM(?z, 1) \wedge has\_F\_Num\_GPM(?z, 1) \rightarrow$ $hasField\_GSM(?a, ?b) \wedge$ $hasSubstance1\_GSM(?a, S1\_with\_different\_physical\_state\_x17) \wedge$ $hasSubstance2\_GSM(?a, S2\_with\_different\_physical\_states\_y17) \wedge$ $has\_S1\_Num\_GSM(?a, 1) \wedge has\_S2\_Num\_GSM(?a, 1) \wedge$ $has\_F\_Num\_GSM(?a, 1)$ Use the reversible physical transformation.
Rule 38	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 551) \wedge Generic\_Problem\_Model(?z) \wedge$ $correspondsTo\_GPM(?x, ?z) \wedge Generic\_Solution\_Model(?a) \wedge$ $correspondsTo\_GSM(?x, ?a) \wedge Field(?b) \wedge Substance(?c) \wedge$ $Substance(?d) \wedge hasField\_GPM(?z, ?b) \wedge hasSubstance1\_GPM(?z, ?c) \wedge$ $hasSubstance2\_GPM(?z, ?d) \wedge has\_S1\_Num\_GPM(?z, 1) \wedge$ $has\_S2\_Num\_GPM(?z, 1) \wedge has\_F\_Num\_GPM(?z, 1) \rightarrow$ $hasField\_GSM(?a, ?b) \wedge$ $hasSubstance1\_GSM(?a, S1\_with\_particles\_x18) \wedge$ $hasSubstance2\_GSM(?a, S2\_with\_particles\_y18) \wedge$ $has\_S1\_Num\_GSM(?a, 1) \wedge has\_S2\_Num\_GSM(?a, 1) \wedge$ $has\_F\_Num\_GSM(?a, 1)$ Substance particles are obtained by decomposition of a higher structural level.
Rule 39	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 552) \wedge Generic\_Problem\_Model(?z) \wedge$ $correspondsTo\_GPM(?x, ?z) \wedge Generic\_Solution\_Model(?a) \wedge$ $correspondsTo\_GSM(?x, ?a) \wedge Field(?b) \wedge Substance(?c) \wedge$ $Substance(?d) \wedge hasField\_GPM(?z, ?b) \wedge hasSubstance1\_GPM(?z, ?c) \wedge$

Name	SWRL rules and explanations
	$hasSubstance2\_GPM(?z, ?d) \wedge has\_S1\_Num\_GPM(?z, 1) \wedge$
	$has\_S2\_Num\_GPM(?z, 1) \wedge has\_F\_Num\_GPM(?z, 1) \rightarrow$
	$hasField\_GSM(?a, ?b) \wedge$
	$hasSubstance1\_GSM(?a, S1\_with\_particles\_x19) \wedge$
	$hasSubstance2\_GSM(?a, S2\_with\_particles\_y19) \wedge$
	$has\_S1\_Num\_GSM(?a, 1) \wedge has\_S2\_Num\_GSM(?a, 1) \wedge$
	$has\_F\_Num\_GSM(?a, 1)$
	Substance particles are obtained by combining a lower structural level.
	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$
	$has\_keyid\_of\_type(?y, 553) \wedge Generic\_Problem\_Model(?z) \wedge$
	$correspondsTo\_GPM(?x, ?z) \wedge Generic\_Solution\_Model(?a) \wedge$
	$correspondsTo\_GSM(?x, ?a) \wedge Field(?b) \wedge Substance(?c) \wedge$
	$Substance(?d) \wedge hasField\_GPM(?z, ?b) \wedge hasSubstance1\_GPM(?z, ?c) \wedge$
	$hasSubstance2\_GPM(?z, ?d) \wedge has\_S1\_Num\_GPM(?z, 1) \wedge$
Rule 40	$has\_S2\_Num\_GPM(?z, 1) \wedge has\_F\_Num\_GPM(?z, 1) \rightarrow$
	$hasField\_GSM(?a, ?b) \wedge$
	$hasSubstance1\_GSM(?a, S1\_nearest\_higher\_or\_lower\_element\_x20) \wedge$
	$hasSubstance2\_GSM(?a, S2\_nearest\_higher\_or\_lower\_element\_y20) \wedge$
	$has\_S1\_Num\_GSM(?a, 1) \wedge has\_S2\_Num\_GSM(?a, 1) \wedge$
	$has\_F\_Num\_GSM(?a, 1)$
	Simple method to obtain Substance particle, the nearest higher or lower element is chosen.

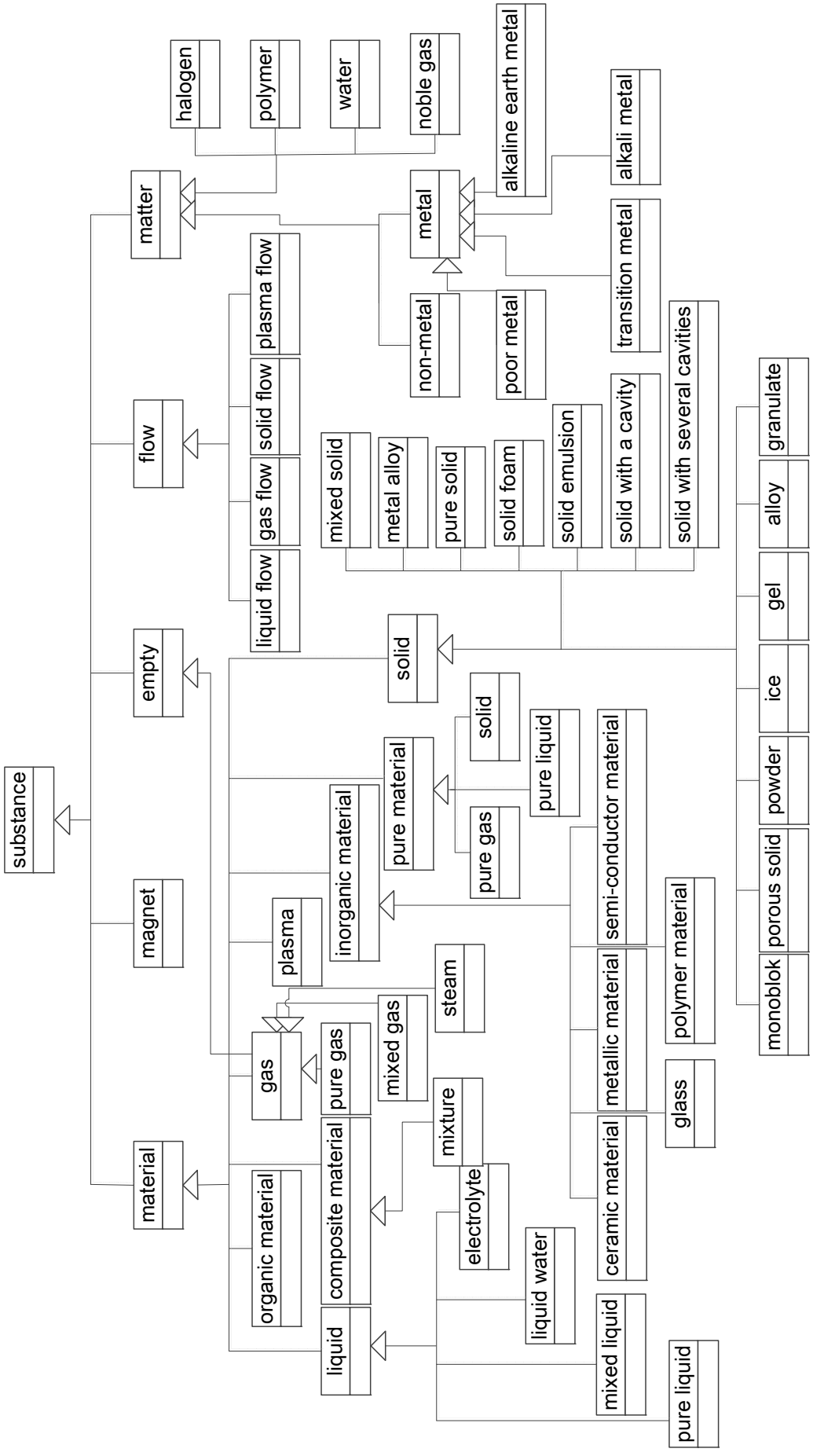
Appendix H

The Classification of Substance and Field

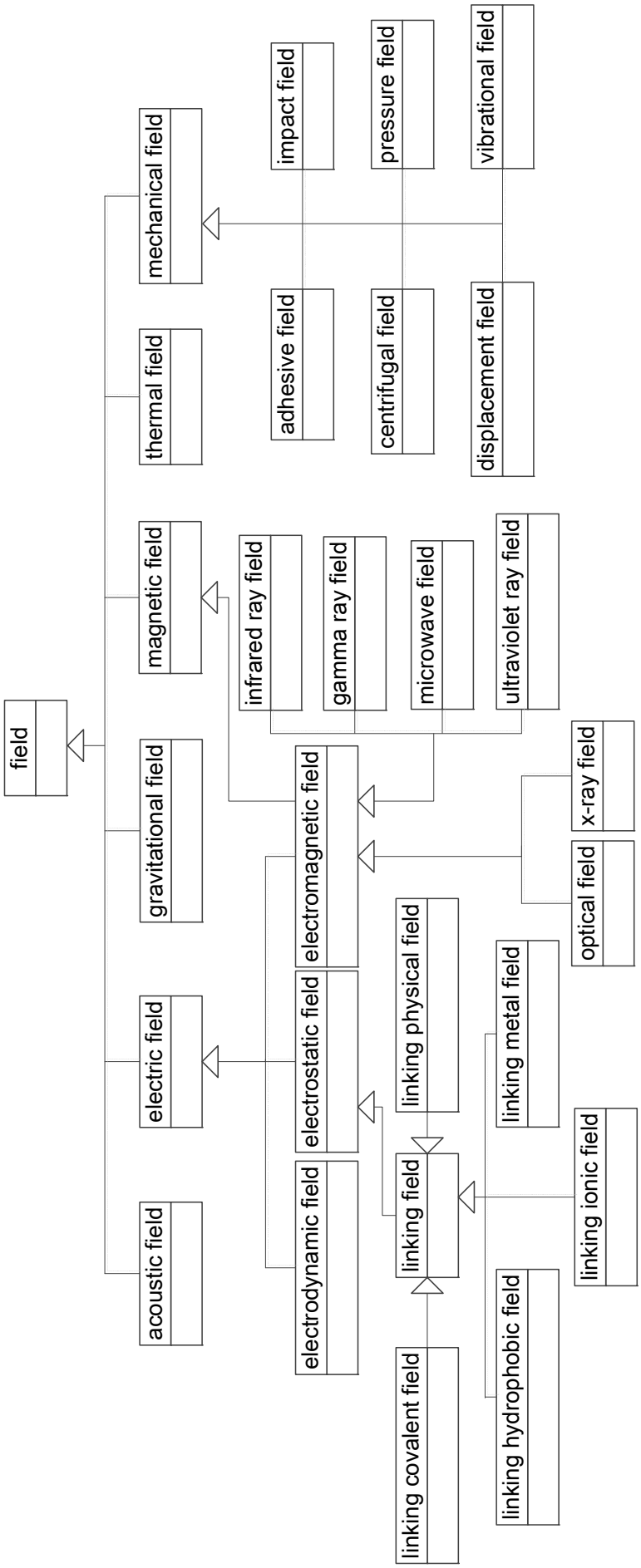
The substance and field are classified based on the ontology built by Bultey[1].

H.1 The Classification of Substance

[1]A. Bultey, F. D. B. de Beuvron, and F. Rousselot, "A substance-field ontology to support the TRIZ thinking approach," International Journal of Computer Applications in Technology, vol. 30, pp. 113-124, 2007.


substance
material
magnet
empty
flow
matter
halogen
polymer
water
noble gas
alkaline earth metal
alkali metal
metal
transition metal
non-metal
poor metal
plasma flow
solid flow
gas flow
liquid flow
mixed solid
metal alloy
pure solid
solid foam
solid emulsion
solid with a cavity
solid with several cavities
granulate
alloy
gel
ice
powder
porous solid
monoblok
solid
pure material
inorganic material
solid
pure liquid
pure gas
plasma
gas
steam
mixed gas
pure gas
mixture
electrolyte
liquid water
liquid
composite material
mixed liquid
pure liquid
organic material
semi-conductor material
metallic material
ceramic material
polymer material
glass

H.2 The Classification of Field


field
acoustic field
electric field
gravitational field
magnetic field
thermal field
mechanical field
impact field
adhesive field
pressure field
centrifugal field
vibrational field
displacement field
infrared ray field
gamma ray field
microwave field
ultraviolet ray field
electrodynamic field
electrostatic field
electromagnetic field
x-ray field
optical field
linking covalent field
linking hydrophobic field
linking ionic field
linking metal field
linking physical field
linking field

Appendix I

The SWRL Rules and Their Explanations for Searching Heuristic Physical Effects

The 50 SWRL rules for searching heuristic physical effects are divided into two kinds: 42 IS (Inventive Standard) rules and 8 PE (Physical Effect) rules.

I.1 IS Rules

Name	IS rules and explanations
Rule 1	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 1) \rightarrow chooses\_PE\_Type(?x, 0)$ If the chosen Inventive Standard belongs to the type 1, the PE of type 0 will be chosen.
Rule 2	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 1) \rightarrow chooses\_PE\_Type(?x, 2)$ If the chosen Inventive Standard belongs to the type 1, the PE of type 2 will be chosen.

Name	IS rules and explanations
Rule 3	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 2) \rightarrow chooses\_PE\_Type(?x, 1)$ If the chosen Inventive Standard belongs to the type 2, the PE of type 1 will be chosen.
Rule 4	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 3) \rightarrow chooses\_PE\_Type(?x, 1)$ If the chosen Inventive Standard belongs to the type 3, the PE of type 1 will be chosen.
Rule 5	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 4) \rightarrow chooses\_PE\_Type(?x, 3)$ If the chosen Inventive Standard belongs to the type 4, the PE of type 3 will be chosen.
Rule 6	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 5) \rightarrow chooses\_PE\_Type(?x, 2)$ If the chosen Inventive Standard belongs to the type 5, the PE of type 2 will be chosen.
Rule 7	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 6) \rightarrow chooses\_PE\_Type(?x, 2)$ If the chosen Inventive Standard belongs to the type 6, the PE of type 2 will be chosen.
Rule 8	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 7) \rightarrow chooses\_PE\_Type(?x, 0)$ If the chosen Inventive Standard belongs to the type 7, the PE of type 0 will be chosen.
Rule 9	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 116) \rightarrow chooses\_PE\_Type(?x, 0)$ If the chosen Inventive Standard belongs to the type 116, the PE of type 0 will be chosen.
Rule 10	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 225) \rightarrow chooses\_PE\_Type(?x, 3)$ If the chosen Inventive Standard belongs to the type 225,

Name	IS rules and explanations
	the PE of type 3 will be chosen.
Rule 11	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 226) \rightarrow chooses\_PE\_Type(?x, 1)$ If the chosen Inventive Standard belongs to the type 226, the PE of type 1 will be chosen.
Rule 12	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 231) \rightarrow chooses\_PE\_Type(?x, 1)$ If the chosen Inventive Standard belongs to the type 231, the PE of type 1 will be chosen.
Rule 13	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 231) \rightarrow chooses\_PE\_Type(?x, 3)$ If the chosen Inventive Standard belongs to the type 231, the PE of type 3 will be chosen.
Rule 14	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 233) \rightarrow chooses\_PE\_Type(?x, 3)$ If the chosen Inventive Standard belongs to the type 233, the PE of type 3 will be chosen.
Rule 15	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 247) \rightarrow chooses\_PE\_Type(?x, 1)$ If the chosen Inventive Standard belongs to the type 247, the PE of type 1 will be chosen.
Rule 16	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 248) \rightarrow chooses\_PE\_Type(?x, 1)$ If the chosen Inventive Standard belongs to the type 248, the PE of type 1 will be chosen.
Rule 17	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 248) \rightarrow chooses\_PE\_Type(?x, 3)$ If the chosen Inventive Standard belongs to the type 248, the PE of type 3 will be chosen.
Rule 18	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 2410) \rightarrow chooses\_PE\_Type(?x, 1)$ If the chosen Inventive Standard belongs to the type 2410,

Name	IS rules and explanations
	the PE of type 1 will be chosen.
Rule 19	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge has\_keyid\_of\_type(?y, 2410) \rightarrow chooses\_PE\_Type(?x, 3)$ If the chosen Inventive Standard belongs to the type 2410, the PE of type 3 will be chosen.
Rule 20	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge has\_keyid\_of\_type(?y, 2412) \rightarrow chooses\_PE\_Type(?x, 1)$ If the chosen Inventive Standard belongs to the type 2412, the PE of type 1 will be chosen.
Rule 21	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge has\_keyid\_of\_type(?y, 311) \rightarrow chooses\_PE\_Type(?x, 0)$ If the chosen Inventive Standard belongs to the type 311, the PE of type 0 will be chosen.
Rule 22	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge has\_keyid\_of\_type(?y, 412) \rightarrow chooses\_PE\_Type(?x, 1)$ If the chosen Inventive Standard belongs to the type 412, the PE of type 1 will be chosen.
Rule 23	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge has\_keyid\_of\_type(?y, 413) \rightarrow chooses\_PE\_Type(?x, 3)$ If the chosen Inventive Standard belongs to the type 413, the PE of type 3 will be chosen.
Rule 24	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge has\_keyid\_of\_type(?y, 444) \rightarrow chooses\_PE\_Type(?x, 3)$ If the chosen Inventive Standard belongs to the type 444, the PE of type 3 will be chosen.
Rule 25	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge has\_keyid\_of\_type(?y, 451) \rightarrow chooses\_PE\_Type(?x, 0)$ If the chosen Inventive Standard belongs to the type 451, the PE of type 0 will be chosen.
Rule 26	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge has\_keyid\_of\_type(?y, 5111) \rightarrow chooses\_PE\_Type(?x, 1)$ If the chosen Inventive Standard belongs to the type 5111,

Name	IS rules and explanations
	the PE of type 1 will be chosen.
Rule 27	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 5112) \rightarrow chooses\_PE\_Type(?x, 3)$ If the chosen Inventive Standard belongs to the type 5112, the PE of type 3 will be chosen.
Rule 28	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 5114) \rightarrow chooses\_PE\_Type(?x, 1)$ If the chosen Inventive Standard belongs to the type 5114, the PE of type 1 will be chosen.
Rule 29	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 5115) \rightarrow chooses\_PE\_Type(?x, 1)$ If the chosen Inventive Standard belongs to the type 5115, the PE of type 1 will be chosen.
Rule 30	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 5117) \rightarrow chooses\_PE\_Type(?x, 1)$ If the chosen Inventive Standard belongs to the type 5117, the PE of type 1 will be chosen.
Rule 31	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 512) \rightarrow chooses\_PE\_Type(?x, 1)$ If the chosen Inventive Standard belongs to the type 512, the PE of type 1 will be chosen.
Rule 32	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 513) \rightarrow chooses\_PE\_Type(?x, 1)$ If the chosen Inventive Standard belongs to the type 513, the PE of type 1 will be chosen.
Rule 33	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 514) \rightarrow chooses\_PE\_Type(?x, 1)$ If the chosen Inventive Standard belongs to the type 514, the PE of type 1 will be chosen.
Rule 34	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge$ $has\_keyid\_of\_type(?y, 531) \rightarrow chooses\_PE\_Type(?x, 1)$ If the chosen Inventive Standard belongs to the type 531,

Name	IS rules and explanations
	the PE of type 1 will be chosen.
Rule 35	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge has\_keyid\_of\_type(?y, 532) \rightarrow chooses\_PE\_Type(?x, 1)$ If the chosen Inventive Standard belongs to the type 532, the PE of type 1 will be chosen.
Rule 36	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge has\_keyid\_of\_type(?y, 533) \rightarrow chooses\_PE\_Type(?x, 1)$ If the chosen Inventive Standard belongs to the type 533, the PE of type 1 will be chosen.
Rule 37	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge has\_keyid\_of\_type(?y, 534) \rightarrow chooses\_PE\_Type(?x, 1)$ If the chosen Inventive Standard belongs to the type 534, the PE of type 1 will be chosen.
Rule 38	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge has\_keyid\_of\_type(?y, 535) \rightarrow chooses\_PE\_Type(?x, 1)$ If the chosen Inventive Standard belongs to the type 535, the PE of type 1 will be chosen.
Rule 39	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge has\_keyid\_of\_type(?y, 541) \rightarrow chooses\_PE\_Type(?x, 1)$ If the chosen Inventive Standard belongs to the type 541, the PE of type 1 will be chosen.
Rule 40	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge has\_keyid\_of\_type(?y, 551) \rightarrow chooses\_PE\_Type(?x, 1)$ If the chosen Inventive Standard belongs to the type 551, the PE of type 1 will be chosen.
Rule 41	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge has\_keyid\_of\_type(?y, 552) \rightarrow chooses\_PE\_Type(?x, 1)$ If the chosen Inventive Standard belongs to the type 552, the PE of type 1 will be chosen.
Rule 42	$Problem(?x) \wedge InventiveStandard(?y) \wedge chooses\_IS(?x, ?y) \wedge has\_keyid\_of\_type(?y, 553) \rightarrow chooses\_PE\_Type(?x, 1)$ If the chosen Inventive Standard belongs to the type 553,

Name	IS rules and explanations
	the PE of type 1 will be chosen.

I.2 PE Rules

Name	PE rules and explanations
Rule 1	$Problem(?x) \wedge Generic\_Problem\_Model(?y) \wedge$ $Generic\_Solution\_Model(?z) \wedge correspondsTo\_GPM(?x, ?y) \wedge$ $correspondsTo\_GSM(?x, ?z) \wedge chooses\_PE\_Type(?x, 0) \wedge$ $Substance(?a) \wedge changesOnOrFor\_Element(?z, ?a) \wedge Add\_SPE(?b) \wedge$ $hasIncrementalSubstance(?b, ?a) \rightarrow chooses\_PE(?x, ?b)$ If a certain substance needs to be added, all the PEs which can add this substance, are obtained.
Rule 2	$Problem(?x) \wedge Generic\_Problem\_Model(?y) \wedge$ $Generic\_Solution\_Model(?z) \wedge correspondsTo\_GPM(?x, ?y) \wedge$ $correspondsTo\_GSM(?x, ?z) \wedge chooses\_PE\_Type(?x, 0) \wedge$ $Substance(?a) \wedge changesOnOrFor\_Element(?z, ?a) \wedge Substance(?c) \wedge$ $includes\_Sub(?a, ?c) \wedge Add\_SPE(?b) \wedge$ $hasIncrementalSubstance(?b, ?c) \rightarrow chooses\_PE(?x, ?b)$ If a certain substance needs to be added, all the PEs which can add this substance or its children substances, such as, liquid and water, are obtained.
Rule 3	$Problem(?x) \wedge Generic\_Problem\_Model(?y) \wedge$ $Generic\_Solution\_Model(?z) \wedge correspondsTo\_GPM(?x, ?y) \wedge$ $correspondsTo\_GSM(?x, ?z) \wedge chooses\_PE\_Type(?x, 1) \wedge$ $Substance(?a) \wedge changesOnOrFor\_Element(?z, ?a) \wedge Modify\_SPE(?c) \wedge$ $hasInitialStateSub(?c, ?a) \rightarrow chooses\_PE(?x, ?c)$ If a certain substance needs to be modified, all the PEs which can modify this substance, are obtained.
Rule 4	$Problem(?x) \wedge Generic\_Problem\_Model(?y) \wedge$ $Generic\_Solution\_Model(?z) \wedge correspondsTo\_GPM(?x, ?y) \wedge$ $correspondsTo\_GSM(?x, ?z) \wedge chooses\_PE\_Type(?x, 1) \wedge$

Name	PE rules and explanations
	$Substance(?a) \wedge changesOnOrFor\_Element(?z, ?a) \wedge Substance(?b) \wedge$ $includes\_Sub(?a, ?b) \wedge Modify\_SPE(?c) \wedge$ $hasInitialStateSub(?c, ?b) \rightarrow chooses\_PE(?x, ?c)$ If a certain substance needs to be modified, all the PEs which can modify this substance or its children substances, such as, liquid and water, are obtained.
Rule 5	$Problem(?x) \wedge Generic\_Problem\_Model(?y) \wedge$ $Generic\_Solution\_Model(?z) \wedge correspondsTo\_GPM(?x, ?y) \wedge$ $correspondsTo\_GSM(?x, ?z) \wedge chooses\_PE\_Type(?x, 2) \wedge$ $Field(?a) \wedge changesOnOrFor\_Element(?z, ?a) \wedge Add\_FPE(?b) \wedge$ $hasIncrementalField(?b, ?a) \rightarrow chooses\_PE(?x, ?b)$ If a certain field needs to be added, all the PEs which can add this field, are obtained.
Rule 6	$Problem(?x) \wedge Generic\_Problem\_Model(?y) \wedge$ $Generic\_Solution\_Model(?z) \wedge correspondsTo\_GPM(?x, ?y) \wedge$ $correspondsTo\_GSM(?x, ?z) \wedge chooses\_PE\_Type(?x, 2) \wedge$ $Field(?a) \wedge changesOnOrFor\_Element(?z, ?a) \wedge Field(?c) \wedge$ $includes\_Field(?a, ?c) \wedge Add\_FPE(?b) \wedge$ $hasIncrementalField(?b, ?c) \rightarrow chooses\_PE(?x, ?b)$ If a certain field needs to be added, all the PEs which can add this field or its children fields, such as, Electric field and Electrodynamic field, are obtained.
Rule 7	$Problem(?x) \wedge Generic\_Problem\_Model(?y) \wedge$ $Generic\_Solution\_Model(?z) \wedge correspondsTo\_GPM(?x, ?y) \wedge$ $correspondsTo\_GSM(?x, ?z) \wedge chooses\_PE\_Type(?x, 3) \wedge$ $Field(?a) \wedge changesOnOrFor\_Element(?z, ?a) \wedge Modify\_FPE(?b) \wedge$ $hasInitialStateField(?b, ?a) \rightarrow chooses\_PE(?x, ?b)$ If a certain field needs to be modified, all the PEs which can modify this field, are obtained.
Rule 8	$Problem(?x) \wedge Generic\_Problem\_Model(?y) \wedge$ $Generic\_Solution\_Model(?z) \wedge correspondsTo\_GPM(?x, ?y) \wedge$ $correspondsTo\_GSM(?x, ?z) \wedge chooses\_PE\_Type(?x, 3) \wedge$

Name	PE rules and explanations
	$Field(?a) \wedge changesOnOrFor\_Element(?z, ?a) \wedge Field(?c) \wedge$ $includes\_Field(?a, ?c) \wedge Modify\_FPE(?b) \wedge$ $hasInitialStateField(?b, ?c) \rightarrow chooses\_PE(?x, ?b)$ If a certain field needs to be modified, all the PEs which can modify this field or its children fields, such as, Electric field and Electrodynamic field, are obtained.

Publications

Journals

1. Wei Yan, Cecilia Zanni-Merk, Denis Cavallucci, and Pierre Collet, An Ontology-based Approach for Inventive Problem Solving. Engineering Applications of Artificial Intelligence, 2014, 27: 175-190.

2. Wei Yan, Cecilia Zanni-Merk, Denis Cavallucci, and Pierre Collet, An Ontology-based Approach for Using Physical Effects in Inventive Design. Engineering Applications of Artificial Intelligence, 2013. (Accepted for publication)

3. Wei Yan, Cecilia Zanni-Merk, François Rousselot, Denis Cavallucci, and Pierre Collet, Facilitating the Resolution of Inventive Problems Using Semantic Relatedness and Ontology Reasoning. International Journal of Knowledge-Based and Intelligent Engineering Systems (KES Journal), 2013, 17: 79-96.

4. Wei Yan, Cecilia Zanni-Merk, François Rousselot and Denis Cavallucci, Ontology Matching for Facilitating Inventive Design based on Semantic Similarity and Case-Based Reasoning. International Journal of Knowledge-Based and Intelligent Engineering Systems (KES Journal), 2013, 17: 243-256.

5. Wei Yan, François Rousselot and Cecilia Zanni-Merk, Component Retrieval Based on Ontology and Graph Patterns Matching. Journal of Information and Computational Science. 2010, 7(4): 893-900.

Conference Proceedings

1. Wei Yan, Cecilia Zanni-Merk, François Rousselot, Denis Cavallucci and Pierre Collet, A New Method of Using Physical Effects in Su-Field Analysis based on Ontology Reasoning. 17th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES2013), Procedia Computer Science Journal (ISSN: 1877-0509) by ELSEVIER, 2013, 22:30-39.

2. Wei Yan, Cecilia Zanni-Merk, François Rousselot, Denis Cavallucci and Pierre Collet, Ontology-based Knowledge Modeling for Using Physical Effects. TRIZ Future 2013, Paris, France.

3. Wei Yan, Cecilia Zanni-Merk, François Rousselot, Denis Cavallucci and Pierre Collet, A Heuristic TRIZ Problem Solving Approach based on Semantic Relatedness and Ontology Reasoning. 16th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES2012), Frontiers in Artificial Intelligence and Applications, 2012, 243:1563-1572.

4. Wei Yan, Cecilia Zanni-Merk, François Rousselot, Denis Cavallucci and Pierre Collet, A Heuristic Method of Using the Pointers to Physical Effects in Su-Field Analysis. TRIZ Future 2012, Lisbon, Portugal.

5. Wei Yan, Cecilia Zanni-Merk, François Rousselot, Denis Cavallucci, and Pierre Collet, Heuristic Inventive Design Problem Solving based on Semantic Relatedness. International Conference on Frontiers of Mechanical Engineering, Materials and Energy (ICFMEME2012) (The paper will be published in Advanced Materials Research).

6. Wei Yan, Cecilia Zanni-Merk and François Rousselot, Matching of Different Abstraction Level Knowledge Sources: The Case of Inventive Design. 15th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES2011), Germany, Part IV, LNAI 6884, pp. 445-454, 2011.

7. Wei Yan, Cecilia Zanni-Merk and François Rousselot, Skyline Adaptive Fuzzy Query. 15th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES2011), Germany, Part II, LNAI 6882, pp. 345-354, 2011.

8. Wei Yan, Cecilia Zanni-Merk and François Rousselot, An Application of Semantic Distance between Short Texts to Inventive Design. International Conference on Knowledge Engineering and Ontology Development (KEOD2011), France, Paris, pp. 261-266.

9. Wei Yan, Cecilia Zanni-Merk, François Rousselot and Denis Cavallucci, A Method of Facilitating Inventive Design Based on Semantic Similarity and Case-Based Reasoning. TRIZ Future 2011, Dublin, Ireland.

Submitted Papers

1. Cecilia Zanni-Merk, Françcois de Bertrand de Beuvron, François Rousselot and Wei Yan, A Formal Ontology for a Generalized Inventive Design Methodology. Journal of Applied Ontology, 2013.

Modélisation des (Méta)Connaissances pour la Conception Inventive

Doctorante : Wei Yan (LGECO & ICUBE)
Directeur de thèse : Denis Cavallucci (LGECO)
Co-Directeur de thèse: Pierre Collet (ICUBE)
Encadrante: Cecilia Zanni-Merk (ICUBE)
Laboratoire : LGECO (EA3938) et ICUBE (UMR CNRS 7357)

1. Introduction

Un nombre croissant d'industries ressentent le besoin de formaliser leurs processus d'innovation. Dans ce contexte, les outils du domaine de la qualité et les approches d'aide à la créativité provenant du "brain storming" ont déjà montré leurs limites. Afin de répondre à ces besoins, la TRIZ (Acronyme russe pour Théorie de Résolution des Problèmes Inventifs), développée par l'ingénieur russe G. S. Altshuller au milieu du 20ème siècle, propose une méthode systématique de résolution de problèmes inventifs multi-domaines. L'approche de résolution de problèmes inventifs de la TRIZ se déroule en trois phases, comme illustré dans la Fig. 1.

1. La phase "Formulation", où l'expert utilise différents outils pour modéliser le problème sous la forme d'une ou plusieurs contradictions[1] ou d'autres modèles.
2. La phase "Recherche d'une solution abstraite", où

[1] Les contradictions peuvent être techniques ou physiques.

l'accès aux différentes bases de connaissances permet d'obtenir un ou plusieurs modèles de solution. Généralement, les utilisateurs de la TRIZ doivent avoir une grande maîtrise des concepts associés à la méthodologie. Ils doivent être capables de choisir la solution abstraite en fonction du problème réel.

3. La phase "Instanciation", où la solution abstraite est instanciée à l'aide de la base de connaissances d'effets physiques, pour obtenir une ou plusieurs solutions conceptuelles dans le monde réel.

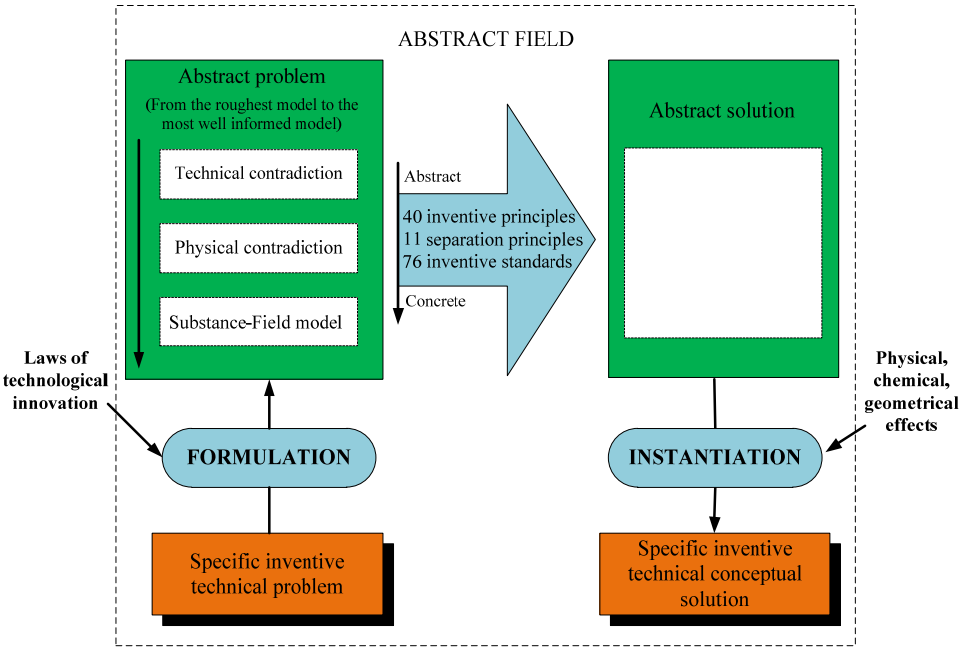


Figure 1. Le processus d'utilisation de TRIZ pour résoudre des problèmes inventifs.

Plusieurs modèles et bases de connaissances contribuent à la résolution de problèmes inventifs. Ils sont de nature différente:

1. Les Principes Inventifs: La TRIZ a formulé

trente-neuf Paramètres Génériques d'Ingénierie, comme par exemple "le poids d'un objet mobile" ou "la vitesse". Un nouveau problème peut être résolu sans compromis en utilisant un Principe Inventif après que le problème ait été formulé comme une contradiction technique[2]. Une telle contradiction met en opposition deux Paramètres Génériques: "un paramètre générique que l'on souhaite voir s'améliorer sans qu'un autre paramètre générique ne se détériore". Le Principe Inventif fournit donc un guide indiquant de quelle manière le problème pourrait être résolu sans avoir l'effet négatif identifié.

2. Les Standards Inventifs: Les Standards Inventifs sont construits sous la forme de recommandations qui généralement peuvent être assimilées à des règles "Si <Condition1> et <Condition2> puis <Recommandation>". Ces deux conditions permettent de reconnaître la typologie du problème lié au standard. Pour tout problème abstraite, il existe un certain nombre de recommandations permettant la construction de sa solution abstraite correspondante. Les Standards Inventifs sont formulés de façon très abstraite, et donc leur utilisation pratique est assez difficile.
3. Les Méthodes de Séparation: Pour résoudre des problèmes contenant une contradiction physique[3],

[2] Une contradiction technique se pose quand il est nécessaire d'améliorer certaines caractéristiques du prototype existant, mais toutes les solutions connues dans le domaine ne produit pas le résultat souhaité ou leur utilisation entraînerait un effet négatif.

[3] Une contradiction physique indique qu'une partie d'un

onze Méthodes de Séparation sont utilisées pour l'éliminer. Les Méthodes de Séparation sont très génériques, et sont donc très difficiles à transposer dans la réalité sans connaissances spécifiques.

Comme indiqué précédemment, toutes ces sources de connaissances de la TRIZ sont construites indépendamment du domaine d'application spécifique, et leurs niveaux d'abstraction sont très différents, ce qui entraîne une difficulté supplémentaire dans leurs interprétations.

Afin de faciliter le processus de résolution de problèmes inventifs, un "Système Intelligent de Gestion de Connaissances" est développé dans cette thèse. D'une part, en intégrant les ontologies des bases de connaissance de la TRIZ, le gestionnaire propose aux utilisateurs de sources de connaissance pertinentes pour le modèle qu'ils construisent, et d'autre part, le gestionnaire a la capacité de remplir "automatiquement" les modèles associés aux autres bases de connaissance.

Ces travaux de recherche visent à faciliter et automatiser le processus de résolution de problèmes inventifs. Ils sont basés sur le calcul de similarité sémantique et font usage de différentes technologies provenantes de domaine de l'Ingénierie de Connaissances (modélisation et raisonnement basés sur les ontologies, notamment). Tout d'abord, des méthodes de calcul de similarité sémantique sont proposées pour rechercher et définir les liens

---

prototype de conception doit avoir deux valeurs mutuellement exclusives de la même paramètre physique.

manquants entre les bases de connaissance de la TRIZ. Ensuite, les sources de connaissance de la TRIZ sont formalisées comme des ontologies afin de pouvoir utiliser des mécanismes d'inférence heuristique pour la recherche de solutions spécifiques.

Pour résoudre des problèmes inventifs, les utilisateurs de la TRIZ choisissent dans un premier temps une base de connaissance et obtiennent une solution abstraite. Ensuite, les éléments des autres bases de connaissance similaires aux éléments sélectionnés dans la première base sont proposés sur la base de la similarité sémantique préalablement calculée. A l'aide de ces éléments et des effets physiques heuristiques, d'autres solutions conceptuelles sont obtenues par inférence sur les ontologies.

Enfin, un prototype logiciel est développé. Il est basé sur cette similarité sémantique et les ontologies interviennent en support du processus de génération automatique de solutions conceptuelles.

2. Recherche Doctorale

◆ Théorie

Comme le montre la Fig. 2, il existe trois approches théoriques (les boites de couleur verte) utilisées dans cette thèse: Le calcul de similarité sémantique, la modélisation des connaissances basée sur les ontologies et le raisonnement basé sur les ontologies. Le mécanisme proposé pour la résolution des problèmes inventifs basé sur la connaissance est composé de trois étapes (les boites de couleur orange):

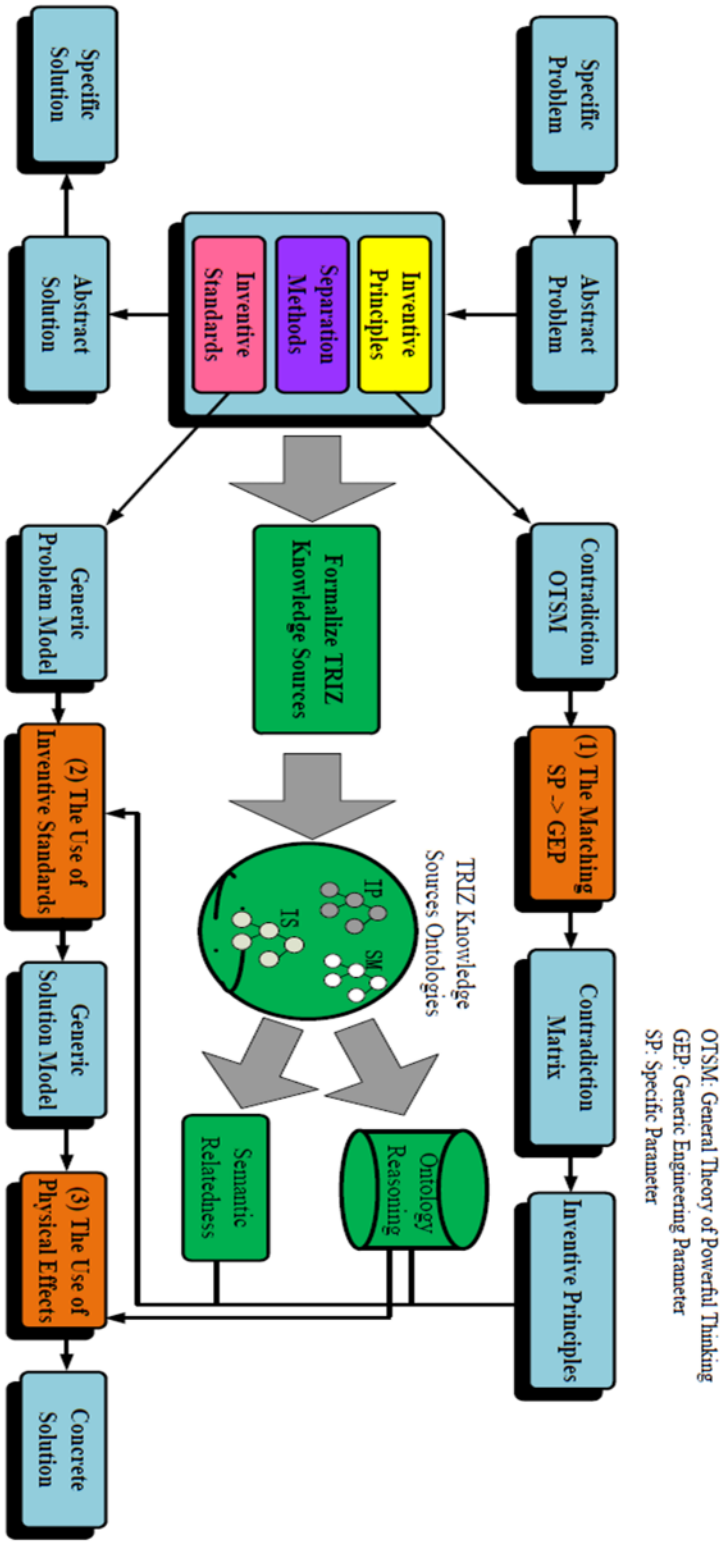


Figure 2. La structure principale.

- La correspondance entre les Paramètres Spécifiques (SP) et les Paramètres Génériques d'Ingénierie (GEP en anglais): Afin de faciliter l'utilisation des Principes Inventifs lors des études de conception inventive, des méthodes de correspondance entre SP et GEP basées sur la similarité sémantique et le raisonnement à partir de cas sont proposées comme illustré dans la Fig. 3 [3][12][13].

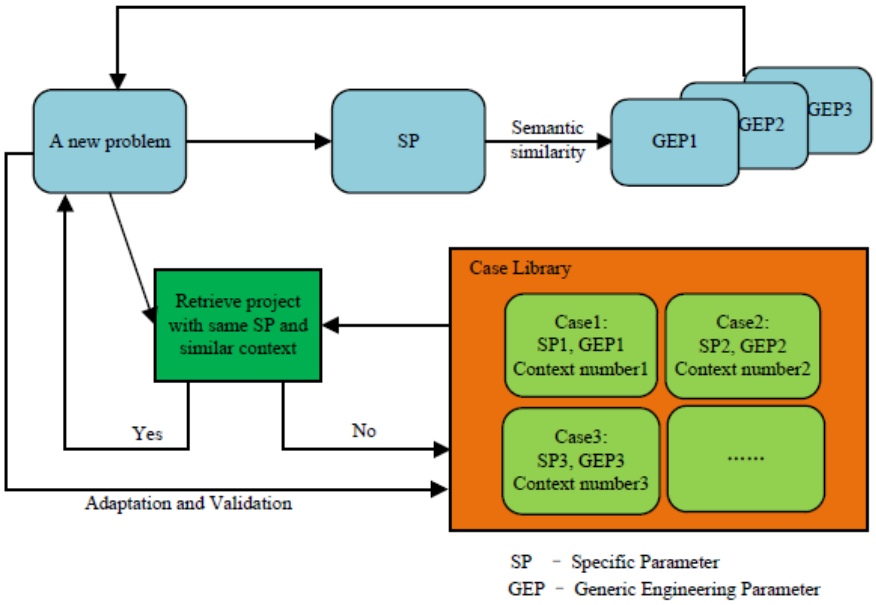


Figure 3. L'organigramme de la méthode proposée.

- L'utilisation des sources de connaissance de la TRIZ: Afin de faciliter l'utilisation des bases de connaissance de la TRIZ, une nouvelle méthode de résolution de problèmes inventifs est proposée basée sur des ontologies, comme explicité dans la Section 1. L'architecture de cette méthode est illustré dans la Fig. 4. En prenant en compte que toutes les sources de connaissance de la TRIZ sont décrites comme des textes courts, les liens

manquants entre les sources de connaissance sont définis à l'aide d'un calcul de similarité sémantique entre textes courts que nous proposons [1][2]. Au même temps, nous proposons l'utilisation du raisonnement sur les ontologies avec Protégé et Jess (Java Expert System Shell) pour générer des solutions abstraites heuristiques dynamiques [6][7].

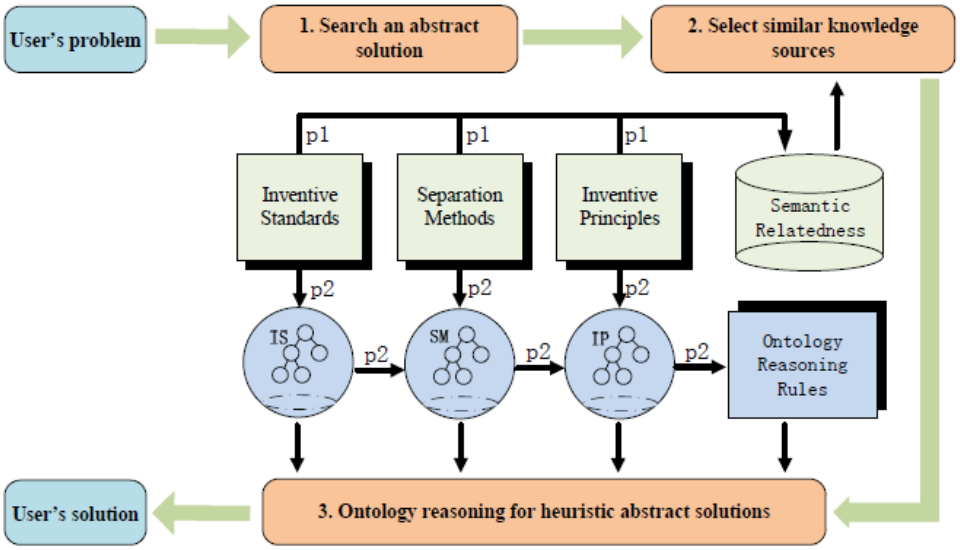


Figure 4. L'architecture de la méthode.

- L'utilisation d'effets physiques: Une nouvelle méthode de représentation des effets physiques est proposée. Un effet physique est représenté par un couple d'états du système avant et après son utilisation. Nous avons formalisé les informations quant à l'usage des effets physiques en OWL (Web Ontology Language). Les informations relatives aux contraintes liées à l'usage de chaque type d'effets physiques ont été formalisées en SWRL (Semantic Web Rule Language). Enfin, comme

illustré dans la Fig. 5, Jess est utilisé pour la recherche heuristique d'effets physiques [4][5].

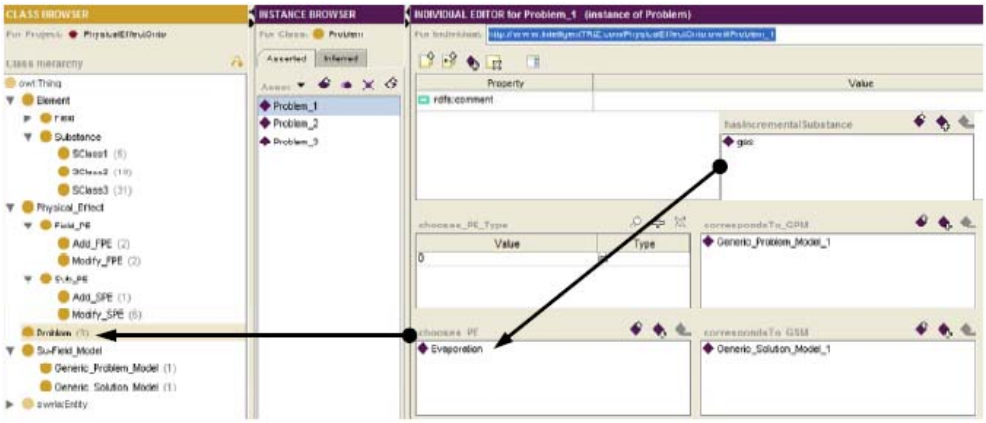


Figure 5. Le processus de raisonnement.

◆ Prototype

Afin de vérifier les approches proposées sur des cas réels de conception inventive, le prototype IngeniousTRIZ a été développé à l'aide de la plateforme Java 1.7.02, de WordNet 2.0, de Protégé 3.4.3 et de Jess 7.1p2 (sur Windows OS).

Comme illustré dans la Fig. 6, le processus d'utilisation de ce prototype est divisé en trois phases:

A. **Etape 1-> Etape 4: L'usage des Principes Inventifs.** Le diagramme de séquence pour Etape 1-> Etape 4 est illustré dans la Fig. 7.

- **Etape 1: Contradiction Description** Afin de résoudre les contradictions dans un problème spécifique, l'utilisateur est guidé à entrer des paramètres, par exemple, deux paramètres spécifiques.
- **Etape 2: Contradiction Formalisation** Basée

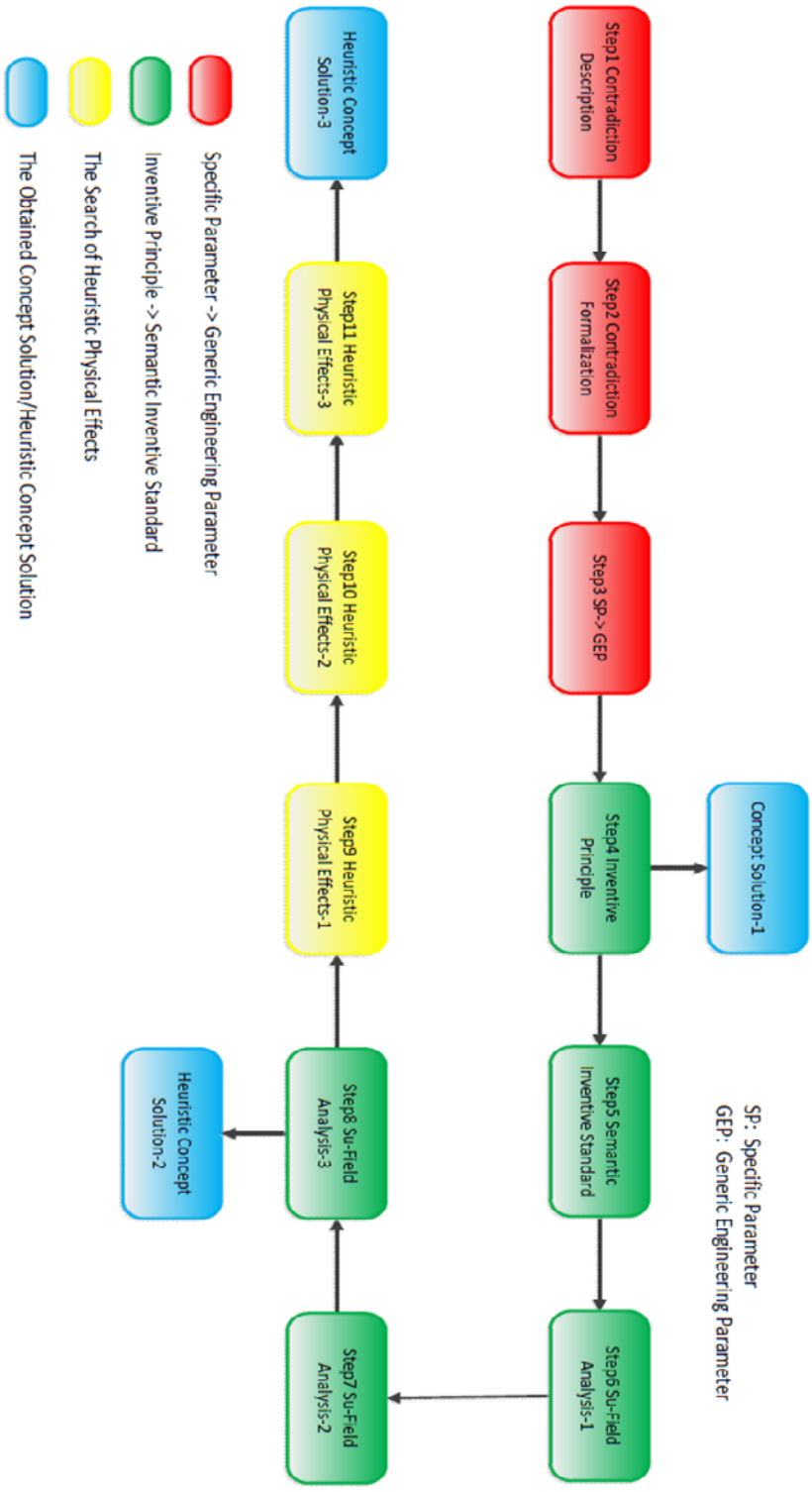


Figure 6. L'organigramme du prototype: IngeniousTRIZ.

sur l'information obtenue dans Etape 1, la contradiction est affichée sous forme d'un tableau.

- **Etape 3: SP->GEP** Afin de chercher les Principes Inventifs à l'aide de la matrice de contradiction, deux Paramètres Spécifiques (SP) doivent correspondre aux trente-neuf Paramètres Génériques d'Ingénierie (GEP). L'utilisateur peut les choisir manuellement ou avec l'aide de la recherche sémantique.
- **Etape 4: Principes Inventifs** A l'aide de la matrice de contradiction, une série de Principes Inventifs est obtenue.
- **Concept Solution** Basé sur le Principe Inventif obtenu, l'utilisateur peut arrêter ce processus, concevoir une concept solution, et stocker l'information obtenue.

B. **Etape 4-> Etape 8: La recherche heuristique de solutions abstraites.** Le diagramme de séquence pour Etape 4-> Etape 8 est illustré dans la Fig. 8.

- **Etape 5: Standards Inventifs Sémantique** Selon le problème spécifique, l'utilisateur doit choisir un Standard Inventif, qui est le plus similaire avec le Principe Inventif obtenu. L'utilisateur peut les sélectionner manuellement ou avec l'aide de la recherche sémantique.
- **Etape 6: Su-Field Analyse-1** Dans cette étape, l'utilisateur doit fournir les éléments (substances et champ) du modèle de problème.
- **Etape 7: Su-Field Analyse-2** Selon le Standard Inventif choisi, l'utilisateur doit sélectionner un type de modification correspondant.

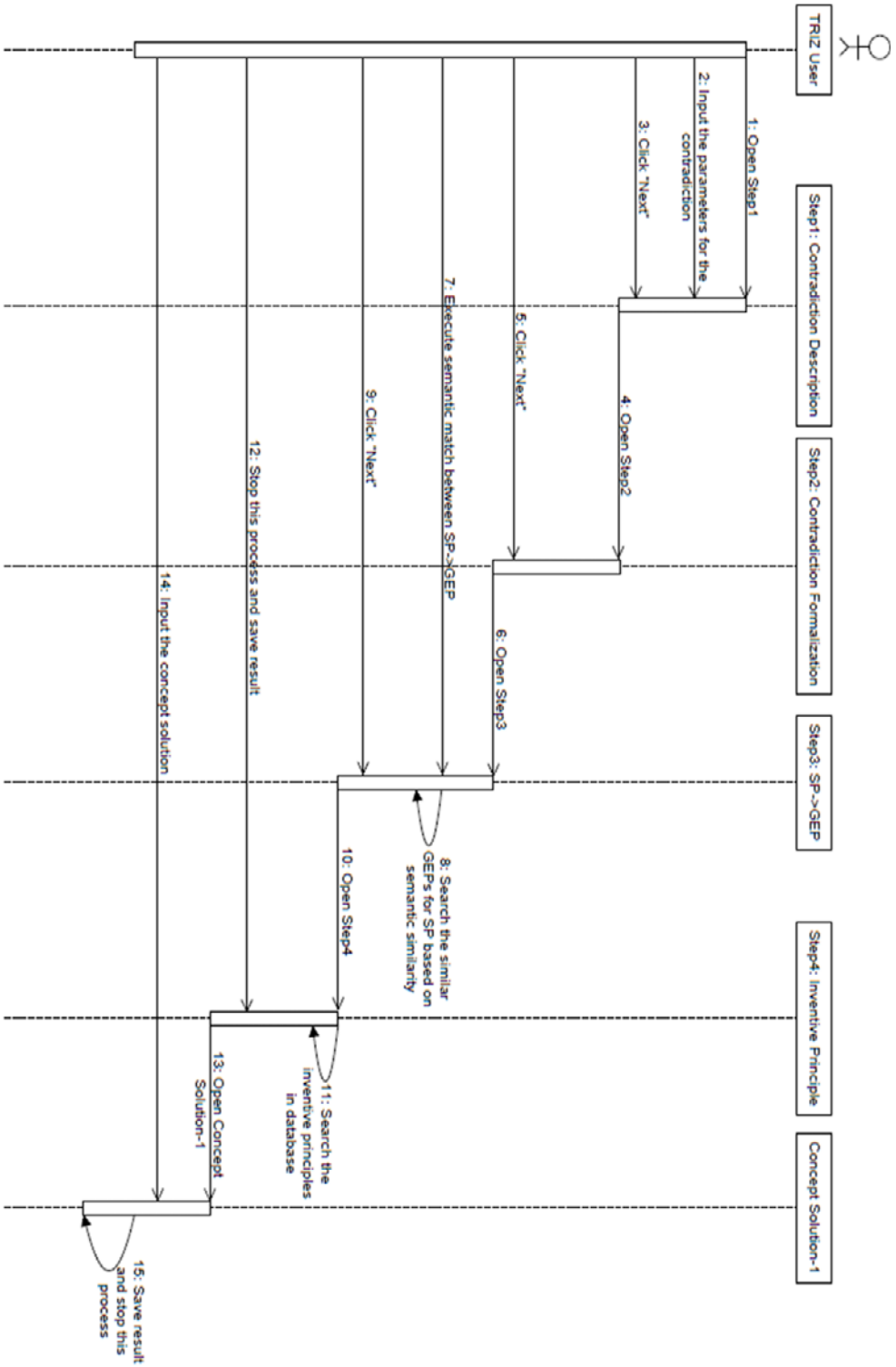


Figure 7. Diagramme de séquence pour Etape 1-> Etape 4.

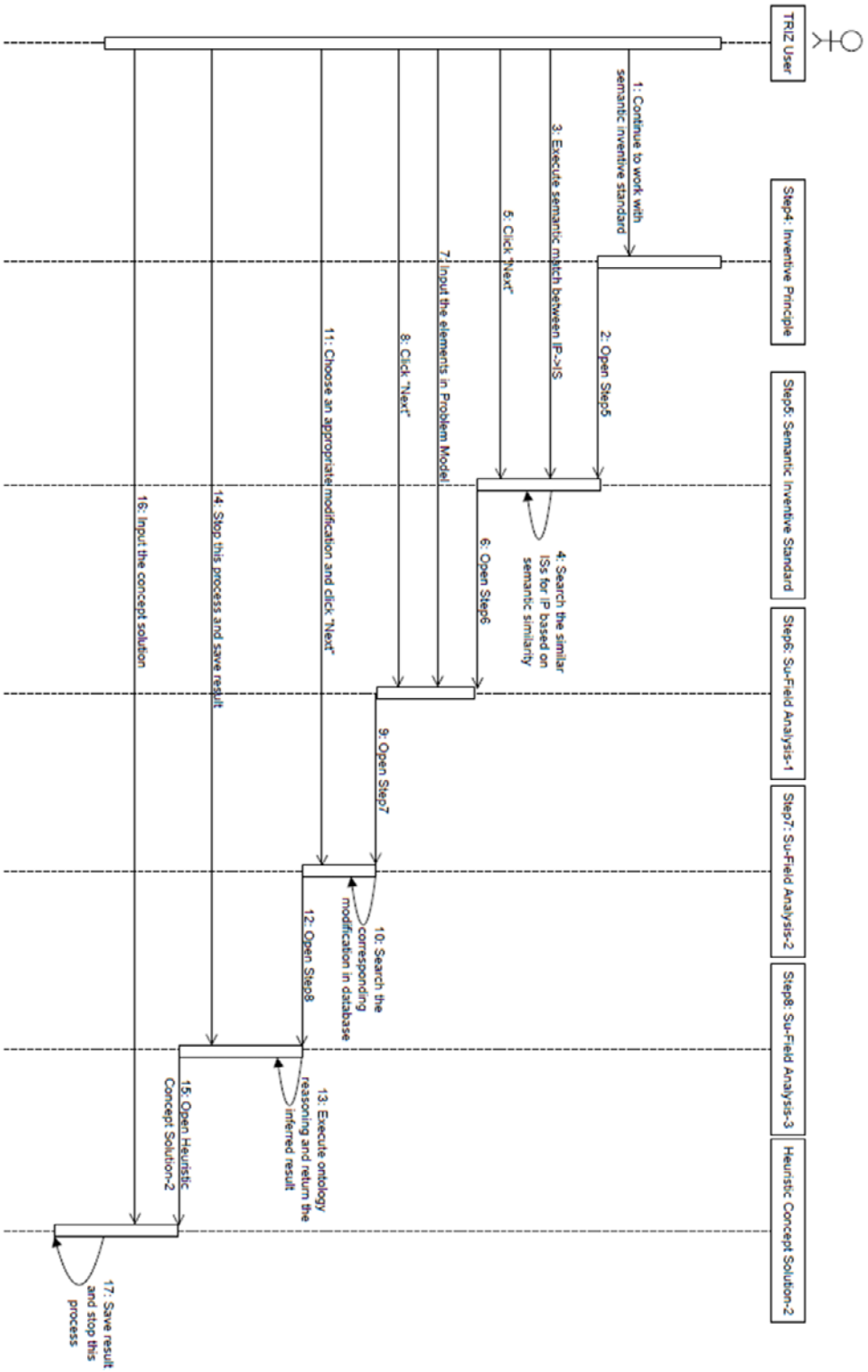


Figure 8. Diagramme de séquence pour Etape 4-> Etape 8.

- **Etape 8: Su-Field Analyse-3** Basé sur l'information obtenue ci-dessus, le raisonnement sur les ontologies est utilisé pour générer des solutions abstraites heuristiques dynamiques.
- **Concept Solution Heuristique** Basé sur les solutions abstraites heuristiques obtenu, l'utilisateur peut arrêter ce processus, concevoir une concept solution heuristique, et stocker l'information obtenue.

C. **Etape 8-> Etape 11: L'utilisation heuristique d'effets physiques pour obtenir une ou des solution(s) conceptuelle(s).** Le diagramme de séquence pour Etape 8-> Etape 11 est illustré dans la Fig. 9.

- **Etape 9: Effets Physiques Heuristique-1** Basé sur le Standard Inventif choisi, plusieurs types d'effets physiques sont proposés, et l'utilisateur doit choisir l'un d'eux.
- **Etape 10: Effets Physiques Heuristique-2** L'utilisateur doit fournir le niveau de granularité de l'élément qui est modifié ou ajouté dans le solution modèle.
- **Etape 11: Effets Physiques Heuristique-3** Basé sur l'information obtenue ci-dessus, le raisonnement sur les ontologies est utilisé pour générer des effets physiques heuristiques dynamiques.
- **Concept Solution Heuristique** Basé sur les solutions abstraites heuristiques et les effets physiques heuristiques, l'utilisateur peut arrêter ce processus, concevoir une concept solution heuristique, et stocker l'information obtenue.

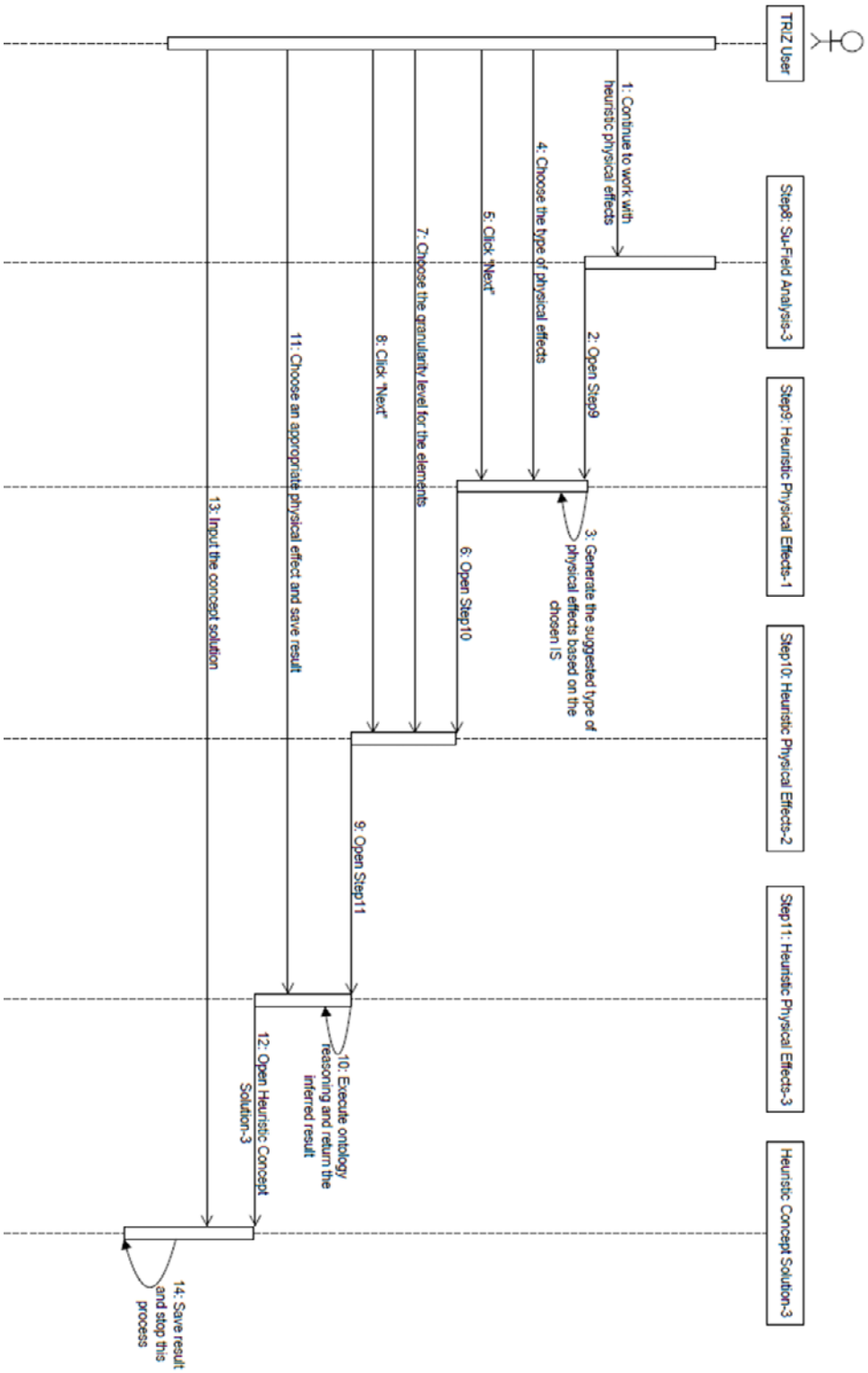


Figure 9. Diagramme de séquence pour Etape 8-> Etape 11.

3. Conclusions et Perspectives

TRIZ (Acronyme russe pour Théorie de Résolution des Problèmes Inventifs) est une méthode systématique pour la résolution de problèmes inventifs dans différents domaines. Selon TRIZ, la résolution de problèmes inventifs consiste en la construction du modèle et l'utilisation des sources de connaissance de la TRIZ. Plusieurs modèles et sources de connaissances permettent la résolution de problèmes inventifs de types différents, comme les quarante Principes Inventifs pour l'élimination des contradictions techniques. Toutes ces sources se situent à des niveaux d'abstractions relativement élevés et sont, donc, indépendantes d'un domaine particulier, ce qui nécessite des connaissances approfondies des domaines d'ingénierie différents.
Cette étude vise à faciliter et automatiser le processus de résolution de problèmes inventifs basé sur la similarité sémantique et techniques de l'ontologie. Tout d'abord, les sources de connaissance de la TRIZ sont formalisés sur des ontologies afin que l'inférence heuristique puisse être exécutée à la recherche des solutions spécifiques. Ensuite, les méthodes de calcul de similarité sémantique sont proposés pour rechercher et définir les liens manquants entre les sources de connaissance de la TRIZ.

Afin de faciliter le processus de résolution de problèmes inventifs, un "gestionnaire intelligent" est développé dans cette thèse. D'une part, selon les ontologies des sources de connaissance de la TRIZ, le gestionnaire propose aux utilisateurs des sources de

connaissance pertinentes du modèle qu'ils construisent, et d'autre part, le gestionnaire a la capacité de remplir "automatiquement" les modèles des autres sources de connaissance.

Les TRIZ utilisateurs commencent à résoudre les problèmes inventifs avec la source de connaissance de leur choix afin d'obtenir une solution abstraite. Selon les éléments sélectionnés de la première source de connaissance, les éléments similaires d'autres sources de connaissance sont obtenus sur la base de la similarité sémantique calculée à l'avance. Avec l'aide de ces éléments similaires et les pointeurs heuristiques vers les effets physiques, d'autres solutions spécifiques sont obtenues par le raisonnement de l'ontologie.

Enfin, un prototype de logiciel a été développé basé sur la similarité sémantique et les ontologies interviennent en support du processus de génération automatique de solutions conceptuelles.

En termes de perspectives, plusieurs voies peuvent être envisagées. Le premier facteur problématique est WordNet, qui est utilisé comme le dictionnaire sémantique dans le calcul de similarité sémantique de textes courts. Les méthodes proposées ne seront pas toujours efficaces en raison des limites de WordNet. L'interaction avec les utilisateurs permettra de résoudre ce problème. Par conséquent, nous avons l'intention de fournir une plate-forme de communication avec les utilisateurs grâce à l'élaboration de SWRL règles.

La deuxième perspective est qu'un certain nombre d'information est perdue pendant le processus de la

recherche des solutions abstraites heuristiques. Les TRIZ utilisateurs commencent à résoudre les problèmes inventifs avec quarante Principes Inventifs afin d'obtenir une solution abstraite. Selon les Principes Inventifs sélectionnés, les Standards Inventifs similaires sont obtenus sur la base de la similarité sémantique calculée à l'avance. Avec l'aide de ces Standards Inventifs similaires, les solutions abstraites heuristiques sont obtenues basé sur l'inférence de l'ontologies. Cependant, une grande quantité d'information est perdue pendant la transformation de la résolution avec contradictions techniques à avec Su-Field analyse. Une manière de résoudre ce problème est l'analyse avec onze Méthodes de Séparation et contradictions physiques pour fournir l'information complémentaire. Pour les Principes Inventifs choisis, ce ne sont pas seulement les Standards Inventifs similaires qui sont utilisés pour générer les solutions abstraites heuristiques, mais aussi les Méthodes de Séparation similaires, utilisées pour fournir l'information de niveau physique.

Troisièmement, afin de démontrer l'applicabilité de nos approches et du prototype, il sera nécessaire et important de les appliquer sur plus des cas industriels. Comme indiqué dans cette thèse, la résolution de problèmes inventifs avec TRIZ est complexe et plusieurs cas simples ne suffisent pas. En conséquence, toutes les méthodes proposées dans cette thèse seront intégrées dans le logiciel STEPS. Les recherches futures dans ce domaine consistèrent à comparer les traductions d'origine humaine avec les résultats obtenus par nos méthodes.

Publications

1. **Wei Yan**, Cecilia Zanni-Merk, Denis Cavallucci, and Pierre Collet, An Ontology-based Approach for Inventive Problem Solving. Engineering Applications of Artificial Intelligence, 2014, 27:175-190.
2. **Wei Yan**, Cecilia Zanni-Merk, Denis Cavallucci, and Pierre Collet, An Ontology-based Approach for Using Physical Effects in Inventive Design. Engineering Applications of Artificial Intelligence, 2013 (accept).
3. **Wei Yan**, Cecilia Zanni-Merk, Francois Rousselot, Denis Cavallucci, and Pierre Collet, Facilitating the Resolution of Inventive Problems Using Semantic Relatedness and Ontology Reasoning. International Journal of Knowledge-Based and Intelligent Engineering Systems (KES Journal), 2013, 17:79-96.
4. **Wei Yan**, Cecilia Zanni-Merk, Francois Rousselot and Denis Cavallucci, Ontology Matching for Facilitating Inventive Design based on Semantic Similarity and Case-Based Reasoning. International Journal of Knowledge-Based and Intelligent Engineering Systems (KES Journal), 2013, 17:243-256.
5. **Wei Yan**, Cecilia Zanni-Merk, François Rousselot, Denis Cavallucci and Pierre Collet, A New Method of Using Physical Effects in Su-Field Analysis based on Ontology Reasoning. Procedia Computer Science Journal (ISSN: 1877-0509) by ELSEVIER, 2013, 22:30-39.
6. **Wei Yan**, Cecilia Zanni-Merk, François Rousselot, Denis Cavallucci and Pierre Collet, A Heuristic TRIZ Problem Solving Approach based on Semantic Relatedness and Ontology Reasoning. Frontiers in Artificial Intelligence and Applications, 2012, 243:1563-1572.
7. **Wei Yan**, Francois Rousselot and Cecilia Zanni-Merk, Component Retrieval Based on Ontology and Graph Patterns Matching. Journal of Information and Computational Science. 2010, Vol 7(4): 893-900.

8. **Wei Yan**, Cecilia Zanni-Merk, François Rousselot, Denis Cavallucci and Pierre Collet, A New Method of Using Physical Effects in Su-Field Analysis based on Ontology Reasoning. *17th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES2013)*, Kitakyushu, Japan.
9. **Wei Yan**, Cecilia Zanni-Merk, François Rousselot, Denis Cavallucci and Pierre Collet, Ontology-based Knowledge Modeling for Using Physical Effects. *TRIZ Future 2013*, Paris, France.
10. **Wei Yan**, Cecilia Zanni-Merk, François Rousselot, Denis Cavallucci and Pierre Collet, A Heuristic TRIZ Problem Solving Approach based on Semantic Relatedness and Ontology Reasoning. *16th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES2012)*, San Sebastian, Spain.
11. **Wei Yan**, Cecilia Zanni-Merk, François Rousselot, Denis Cavallucci and Pierre Collet, A Heuristic Method of Using the Pointers to Physical Effects in Su-Field Analysis. *TRIZ Future 2012*, Lisbon, Portugal.
12. **Wei Yan**, Cecilia Zanni-Merk, Francois Rousselot, Denis Cavallucci, and Pierre Collet, Heuristic Inventive Design Problem Solving based on Semantic Relatedness. *International Conference on Frontiers of Mechanical Engineering, Materials and Energy (ICFMEME2012)* (The paper will be published in Advanced Materials Research).
13. **Wei Yan**, Cecilia Zanni-Merk and Francois Rousselot, Matching of Different Abstraction Level Knowledge Sources: The Case of Inventive Design. *15th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES2011)*, Germany, Part IV, LNAI 6884, pp. 445-454, 2011.
14. **Wei Yan**, Cecilia Zanni-Merk and Francois Rousselot, Skyline Adaptive Fuzzy Query. *15th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES2011)*, Germany,

Part II, LNAI 6882, pp. 345-354, 2011.
15. **Wei Yan**, Cecilia Zanni-Merk and Francois Rousselot, An Application of Semantic Distance between Short Texts to Inventive Design. *International Conference on Knowledge Engineering and Ontology Development (KEOD2011)*, France, Paris, pp. 261-266.
16. **Wei Yan**, Cecilia Zanni-Merk, François Rousselot and Denis Cavallucci, A Method of Facilitating Inventive Design Based on Semantic Similarity and Case-Based Reasoning. *TRIZ Future 2011*, Dublin, Ireland.