STUDIA

UNIVERSITATIS BABEŞ-BOLYAI

Informatica

CLUJ-NAPOCA 2006

Cluj University Press

2

EDITORIAL BOARD OF

STUDIA UNIVERSITATIS BABEŞ-BOLYAI INFORMATICA

EDITORIAL OFFICE OF INFORMATICA: M. Kogălniceanu no. 1, 400084 Cluj-Napoca ♦
Phone: 0264-40.53.00

S T U D I A
UNIVERSITATIS BABEŞ-BOLYAI

INFORMATICA

2

Redacția: 3400 Cluj-Napoca, str. M. Kogălniceanu nr. 1 Telefon 405300

SUMAR – CONTENTS – SOMMAIRE

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LI**, Number 2, 2006

A GRAPH ALGORITHM FOR IDENTIFICATION OF CROSSCUTTING CONCERNS

GABRIELA ŞERBAN AND GRIGORETA SOFIA MOLDOVAN

Abstract. The purpose of this paper is to present a new graph-based approach in aspect mining. We define the problem of identifying the crosscutting concerns as a search problem in a *graph* and we introduce *GAAM* algorithm (*Graph Algorithm in Aspect Mining*) for solving this problem. We evaluate the results obtained by applying *GAAM* algorithm from the aspect mining point of view, based on a set of quality measure that we have previously defined in [3]. The proposed approach is compared with a clustering approach in aspect mining ([4]) and a case study is also reported.

Keywords: graph, algorithm, aspect mining.

1. Introduction

1.1. **Aspect Mining.** The Aspect Oriented Programming (AOP) is a new paradigm that is used to design and implement *crosscutting concerns* [2]. A *crosscutting concern* is a feature of a software system that is spread all over it, and whose implementation is tangled with other features' implementation. A well known example of crosscutting concern is logging. In order to design and implement a crosscutting concern, AOP introduces a new modularization unit called *aspect*. Better modularization, and higher productivity are some of the advantages that AOP brings to software engineering.

Aspect mining is a relatively new research direction that tries to identify crosscutting concerns in already developed software systems, without using AOP. The goal is to identify them and then to refactor them to aspects, to achieve a system that can be easily understood, maintained and modified.

1.2. **Related Work.** Several approaches have been considered for aspect mining until now. One approach was to develop tools that would help the user to navigate and to analyze the source code in order to find crosscutting concerns. Some of them

Received by the editors: September, 30, 2006.

2000 *Mathematics Subject Classification.* 68N99, 68R10.

1998 *CR Categories and Descriptors.* D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement –*Restructuring, reverse engineering, and reengineering*; G.2.2 [**Mathematics of Computing**]: Discrete Mathematics – *Graph Theory*.

rely on lexical analysis, and some also include a type-based search ([1], [15], [16], [17]). Other approach uses clone detection techniques to identify duplicate code, that might indicate the presence of crosscutting concerns ([13], [18], [19]). These are all static approaches that analyze the source code for crosscutting concerns. There are also two dynamic approaches: one that analyzes the event traces ([12]), and one that uses formal concept analysis to analyze the execution traces ([21]). In [20] formal concept analysis is used again, but in a static manner. A comparison of three different approaches can be found in [14].

There is also a clustering approach that constructs the clusters based on the methods' names ([7]). The user can then navigate among the clusters, visualize the source code of the methods and identify the crosscutting concerns.

There are just a few aspect mining techniques proposed in the literature that use *clustering* in order to identify crosscutting concerns ([4], [5], [6], [7]).

In [5] a vector space model based clustering approach in aspect mining is proposed. This approach is improved in [4], by defining a new *k-means* based clustering algorithm in aspect mining (kAM).

In [3], a part of a formal model for clustering in aspect mining is introduced and a set of quality measures for evaluating the results of clustering based aspect mining techniques is presented.

In this paper we propose a new graph-based approach, as an alternative to the clustering approach in aspect mining. Such an approach has not been reported, in the literature, so far.

The paper is structured as follows. A theoretical model on which we base our approach is introduced in Section 2. Section 3 presents our approach. An experimental evaluation of our approach, based on some quality measures, is presented in Section 4. The obtained results are compared with the ones obtained by applying kAM algorithm ([4]). Some conclusions and further work are outlined in Section 5.

2. Theoretical Model

In this section we present the problem of identifying *crosscutting concerns* as a problem of identifying a partition of a software system.

Let $M = \{m_1, m_2, ..., m_n\}$ be the software system, where $m_i, 1 \leq i \leq n$ is a method of the system.

We consider a crosscutting concern as a set $C = \{c_1, c_2, ..., c_{cn}\}$ with $C \subset M$, of methods that implement this concern. The number of methods in the crosscutting concern C is $cn = |C|$. Let $CCC = \{C_1, C_2, ..., C_q\}$ be the set of all crosscutting concerns that exist in the system M.

Definition 1. ***Partition of a software system*** M**.**
The set $\mathcal{K} = \{K_1, K_2, ..., K_p\}$ *is called a* ***partition*** *of the system* $M = \{m_1, m_2, \ldots$

$, m_n\}$ *iff* $1 \leq p \leq n$, $K_i \subseteq M, K_i \neq \emptyset, \forall 1 \leq i \leq p$, $M = \bigcup_{i=1}^{p} K_i$ *and* $K_i \cap K_j = \emptyset$, $\forall i, j, 1 \leq i, j \leq p, i \neq j$.

In the following we will refer to K_i as the i-th *cluster* of $\mathcal{K}$.

In fact, the problem of aspect mining can be viewed as the problem of finding a partition $\mathcal{K}$ of the system M such that $CCC \subset \mathcal{K}$. So, in Definition 2 we introduce the notion of **partitioning aspect mining technique**, that will be used in our approach.

Definition 2. ***Partitioning aspect mining technique.***
Let $\mathcal{T}$ be an aspect mining technique and M a software system to be mined. We say that $\mathcal{T}$ is a ***partitioning aspect mining technique*** *if the result obtained by $\mathcal{T}$ is a* ***partition*** (Definition 1) *$\mathcal{K}$ of M.*

For a software system, we propose the following steps for identifying the crosscutting concerns that have the scattered code symptom:

- **Computation** - Computation of the set of methods in the selected source code, and computation of the attribute set values, for each method in the set.
- **Filtering** - Methods belonging to some data structures classes like *ArrayList, Vector* are eliminated. We also eliminate the methods belonging to some built-in classes like *String, StringBuffer, StringBuilder, in a Java program etc.*
- **Grouping** - The remaining set of methods is grouped in order to obtain a partition of the software system M (in our approach $GAAM$).
- **Analysis** - A part of the obtained clusters are analyzed in order to discover which clusters contain methods belonging to crosscutting concerns.

We mention that at the **Grouping** step, a partition of the software system can be obtained using a clustering algorithm ([4]) in aspect mining, or using $GAAM$ algorithm, that will be introduced in the next section.

3. Our Approach

In this section we present the problem of obtaining a *partition* (Definition 1) of a software system as a search problem in a *graph*.

This graph based approach is, in fact, a method to identify the clusters in the system and can be viewed as an alternative to a *clustering* algorithm in aspect mining ([4]).

In our approach, the objects to be grouped (clustered) are the methods from the software system: $m_1, m_2, \ldots, m_n$. The methods belong to the application classes or are called from the application classes.

Based on the vector space model, we will consider each method as a l-dimensional vector: $m_i = (m_{i1}, \ldots, m_{il})$.

Crosscutting concerns in non AO systems have two symptoms: *code scattering* and *code tangling*. *Code scattering* means that the code that implements a crosscutting concern is spread across the system, and *code tangling* means that the code that implements some concern is mixed with code from other (crosscutting) concerns.

We have considered two vector-space models that illustrate only the *scattered code* symptom. Future development will also consider the *code tangling* symptom.

- The vector associated with the method m is $\{FIV, CC\}$, where FIV is the fan-in value ([8]) of m (the number of methods that call m) and CC is the number of calling classes for m. We denote this model by $\mathcal{M}_1$.
- The vector associated with the method m is $\{FIV, B_1,\ B_2, ...B_{l-1}\}$, where FIV is the fan-in value of m and B_i $(1 \leq i \leq l-1)$ is 1, if the method m is called from a method belonging to the application class C_i, and 0, otherwise. We denote this model by $\mathcal{M}_2$.

As in a vector space model based clustering ([22]), we consider the *distance* between methods as a measure of dissimilarity between them.

In our approach we will consider that the distance between two methods m_i and m_j is expressed using the *Euclidian distance*, as:

$$d_E(m_i, m_j) = \sqrt{\sum_{k=1}^{l}(m_{ik} - m_{jk})^2}. \tag{1}$$

After a partition of the software system is determined using a **partitioning aspect mining technique**, the clusters are sorted by the average distance from the point 0_l in descending order, where 0_l is the l dimensional vector with each component 0 (l is the dimension of the vector space model). Then, we analyze the clusters whose distance from 0_l point is greater than a given threshold.

3.1. **The Methods Graph.** In this section we introduce the concept of *methods graph* and auxiliary definitions needed to define our search problem.

We mention that the idea of constructing the *methods graph* is specific to aspect mining and will be explained later.

Definition 3. *Let $M = \{m_1, m_2, \ldots, m_n\}$ be a software system and d_E* (Equation 1) *the metric between methods in a multidimensional space. The **methods graph** corresponding to the software system M, $\mathcal{MG}_M$, is an undirected graph defined as follows: $\mathcal{MG}_M = (\mathcal{V}, \mathcal{E})$, where:*

- *The set $\mathcal{V}$ of vertices is the set of methods from the software system, i.e., $\mathcal{V}\{m_1, m_2, \ldots, m_n\}$.*

- *The set* $\mathcal{E}$ *of edges is* $\mathcal{E} = \bigcup_{i=1}^{n} \{(m_i, m_j) \mid 1 \leq j \leq n,\ j \neq i,\ d_E(m_i, m_j) = min\{d_E(m_i, m_k),\ 1 \leq k \leq n,\ k \neq i,\ (m_i, m_k) \notin \mathcal{E}\} \wedge d_E(m_i, m_k) \leq distMin\}$*, where* ***distMin*** *is a given threshold.*

We have chosen the value 1 for the threshold *distMin*. The reason for choosing this value is the following: if the distance between two methods m_i and m_j is less or equal to 1, we consider that they are similar enough to be placed in the same (crosscutting) concern. We mention that, from the aspect mining point of view, using *Euclidian distance* as metric and the vector space models proposed above, the value 1 for $distMin$ makes the difference between a crosscutting concern and a non-crosscutting one.

In Definition 4 below we will define the problem of computing a partition of the software system M.

Definition 4. *Let* $M = \{m_1, m_2, \ldots, m_n\}$ *be a software system,* d_E (Equation 1) *the metric between methods in a multidimensional space and* $\mathcal{MG}_M$ *the corresponding* ***methods graph*** (Definition 3)*. We define the problem of computing a partition* $\mathcal{K} = \{K_1, K_2, ..., K_p\}$ *of* M *as the problem of computing the connex components of* $\mathcal{MG}_M$*.*

3.2. ***GAAM* Algorithm.** In this subsection we briefly describe *GAAM* algorithm for determining a *partition* $\mathcal{K}$ of a software system M. This algorithm will be used in the **Grouping** step (Section 2) for identification of crosscutting concerns.

Let us consider a software system $M = \{m_1, m_2, \ldots, m_n\}$ and the metric d_E (Equation 1) between methods in a multidimensional space.

The main steps of *GAAM* algorithm are:

(i) Create the *methods graph*, $\mathcal{MG}_M$, as shown in Definition 3. We mention that the threshold $distMin$ used for creating the edges in the graph is chosen to be 1. The reason for this choice was explained above.

(ii) Determine the connex components of $\mathcal{MG}_M$. These components give a partition $\mathcal{K}$ of the software system M.

4. Experimental Evaluation

In order to evaluate the results of *GAAM* algorithm from the aspect mining point of view, we use three quality measures defined in [3]: *DIV*, *PAM* and *PREC*.

These measures will be applied on a case study (Subsection 4.1). The obtained results will be reported in Subsection 4.1. Based on the obtained results, *GAAM* algorithm will be compared with *kAM* algorithm proposed in [4].

In order to compare two partitions of a software system M from the aspect mining point of view, we introduce the Definition 5. The definition is based on the properties of the quality measures defined above ([3]).

Definition 5. *If $\mathcal{K}_1$ and $\mathcal{K}_2$ are two partitions of the software system M, CCC is the set of crosscutting concerns in M and $\mathcal{T}$ is a partitioning aspect mining technique, then $\mathcal{K}_1$ is **better** than $\mathcal{K}_2$ **from the aspect mining point of view** iff the following inequalities hold:*

$$DIV(CCC, \mathcal{K}_1) \geq DIV(CCC, \mathcal{K}_2), \;\; PREC(CCC, \mathcal{K}_1, \mathcal{T}) \geq PREC(CCC, \mathcal{K}_2, \mathcal{T}),$$
$$PAM(CCC, \mathcal{K}_1) \leq PAM(CCC, \mathcal{K}_2).$$

Remark 1. *If at least one of the inequalities from Definition 5 are not satisfied, we cannot decide which of the partitions $\mathcal{K}_1$ or $\mathcal{K}_2$ is better.*

4.1. **Results.** In order to evaluate the results of *GAAM* algorithm, we consider as case study JHotDraw, version 5.2 ([9]).

This case study is a Java GUI framework for technical and structured graphics, developed by Erich Gamma and Thomas Eggenschwiler, as a design exercise for using design patterns. It consists in **190** classes and **1963** methods.

In this subsection we present the results obtained after applying *GAAM* algorithm described in Subsection 3.2, for the vector space models presented in Section 3, with respect to the quality measures, for the case study presented above.

The results obtained by *GAAM* are compared with the results obtained by *kAM* algorithm proposed in ([4]).

In Table 1 we present the comparative results.

Algorithm	Model	DIV	PREC	PAM
GAAM	$\mathcal{M}_1$	0.844	0.875	0.073
kAM	$\mathcal{M}_1$	0.842	0.875	0.073
GAAM	$\mathcal{M}_2$	0.993	0.875	0.073
kAM	$\mathcal{M}_2$	0.993	0.875	0.081

TABLE 1. The values of the quality measures for JHotDraw case study.

From Table 1 we observe, based on Definition 5, that *GAAM* algorithm provides **better** results from the aspect mining point of view, than *kAM* algorithm, for both vector space models $\mathcal{M}_1$ and $\mathcal{M}_2$.

Moreover, *GAAM* with vector space model $\mathcal{M}_2$ provides the best results.

We can conclude that vector space model $\mathcal{M}_2$ is more appropriate, from the aspect mining point of view.

5. Conclusions and Future Work

We have presented in this paper a new *graph-based approach* in aspect mining. For this purpose we have proposed *GAAM* algorithm, that identifies a partition of a software system. This partition will be analyzed in order to identify the crosscutting concerns from the system.

In order to evaluate the obtained results from the aspect mining point of view, we have used a set of quality measures.

We have given a definition in order to compare two partitions from the aspect mining point of view. Based on this definition, we showed that $GAAM$ algorithm provides **better** partitions than kAM algorithm (previously introduced in [4]).

Further work can be done in the following directions:

- To apply this approach for other case studies like JEdit ([11]).
- To compare the results provided by $GAAM$ with the results of other approaches in aspect mining.
- To identify a choice for the threshold $distMin$ that will lead to better results.
- To improve the results obtained by $GAAM$, by improving the vector space model used.
- To determine the distance metric used that will provide better results from the aspect mining point of view (Minkowski, Manhattan, Hamming, etc).
- To identify the heuristics for constructing the *methods graph* that will lead to better results from the aspect mining point of view.

References

[1] Robillard, M.P., Murphy, G.C., "Concern graphs: finding and describing concerns using structural program dependencies", In: Proceedings of the 24th International Conference on Software Engineering . Orlando, Florida, 2002, pp. 406–416.

[2] Kiczales, G., Lamping, J., Menhdhekar, A., Maeda, C., Lopes, C., Loingtier, J.M., Irwin, J., "Aspect-Oriented Programming", In: Proceedings European Conference on Object-Oriented Programming. Volume 1241. Springer-Verlag, 1997, pp. 220–242.

[3] Moldovan, G.S., Serban, G., "Quality Measures for Evaluating the Results of Clustering Based Aspect Mining Techniques", In: Proceedings of Towards Evaluation of Aspect Mining(TEAM), ECOOP, 2006, pp. 13–16.

[4] Serban, G., Moldovan, G.S., "A new k-means based clustering algorithm in aspect mining", In: 8th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC'06), 2006, pp. 60–64.

[5] Moldovan, G.S., Serban, G., "Aspect Mining using a Vector-Space Model Based Clustering Approach", In: Proceedings of Linking Aspect Technology and Evolution (LATE) Workshop, 2006, to be published.

[6] He, L., Bai, H., "Aspect Mining using Clustering and Association Rule Method" International Journal of Computer Science and Network Security **6**, 2006, pp. 247–251.

[7] Shepherd, D., Pollock, L., "Interfaces, Aspects, and Views" In: Proceedings of Linking Aspect Technology and Evolution (LATE) Workshop, 2005.

[8] Marin, M., van, A., Deursen, Moonen, L., "Identifying Aspects Using Fan-in Analysis" In: Proceedings of the 11th Working Conference on Reverse Engineering (WCRE2004), IEEE Computer Society, 2004, pp. 132–141.

[9] JHotDraw Project, http://sourceforge.net/projects/jhotdraw, 1997.

[10] Mancoridis, S., Mitchell, B.S., Chen, Y., Gansner, E.R., "Bunch: A Clustering Tool for the Recovery and Maintenance of Software System Structures", In: ICSM '99: Proceedings

of the IEEE International Conference on Software Maintenance, IEEE Computer Society, 1999, pp. 50–59.
[11] jEdit Programmer's Text Editor: http://www.jedit.org, 2002.
[12] Breu, S., Krinke, J., "Aspect Mining using Event Traces", In: *Proceedings of International Conference on Automated Software Engineering*, 2004, pp. 310–315.
[13] Bruntink, M., van Deursen, A., van Engelen, R., Tourwé, T., "An Evaluation of Clone Detection Techniques for Identifying Crosscutting Concerns", In: *Proceedings International Conference on Software Maintenance(ICSM 2004)*, IEEE Computer Society, 2004.
[14] Ceccato, M., Marin, M., Mens, K., Moonen, L., Tonella, P., Tourwé, T., "A Qualitative Comparison of Three Aspect Mining Techniques", In: *IWPC '05, Proceedings of the 13th International Workshop on Program Comprehension*, IEEE Computer Society, 2005, pp. 13–22.
[15] Griswold, W.G., Kato, Y., Yuan, J.J., "AspectBrowser: Tool Support for Managing Dispersed Aspects", Technical Report CS1999-0640, UCSD, 3, 2000.
[16] Hannemann, J., Kiczales, G., "Overcoming the Prevalent Decomposition of Legacy Code", In: *Advanced Separation of Concerns Workshop,at the International Conference on Software Engineering. (ICSE)*, 2001.
[17] Zhang, C., Gao, G., Jacobsen, H., "Multi Visualizer", http://www.eecg.utoronto.ca/ czhang/amtex/.
[18] Sheperd, D., Gibson, E., Pollock, L., "Design and Evaluation of an Automated Apect Mining Tool", In: *Proceedings of Mid-Atlantic Student Workshop on Programming Languages and Systems*, 2004.
[19] Morales, O.A.M., "Aspect Mining Using Clone Detection", Master's thesis, Delft University of Technology, The Netherlands, 2004.
[20] Tourwé, T., Mens, K., "Mining Aspectual Views using Formal Concept Analysis", In: *Proc. IEEE International Workshop on Source Code Analysis and Manipulation*, 2004.
[21] Tonella, P., Ceccato, M., "Aspect Mining through the Formal Concept Analysis of Execution Traces", In: *Proceedings of the IEEE Eleventh Working Conference on Reverse Engineering (WCRE 2004)*, 2004, pp. 112–121.
[22] Jain, A., Dubes, R., "Algorithms for Clustering Data", Prentice Hall, Englewood Cliffs, New Jersey, 1998.

Babeş-Bolyai University, Faculty of Mathematics and Computer Science, Cluj-Napoca, Romania
E-mail address: gabis@cs.ubbcluj.ro

Babeş-Bolyai University, Faculty of Mathematics and Computer Science, Cluj-Napoca, Romania
E-mail address: grigo@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LI**, Number 2, 2006

NOTES ON THE ROLE OF THE INCREMENTALITY IN SOFTWARE ENGINEERING

LADISLAV SAMUELIS AND CSABA SZABÓ

Abstract. The incrementality principle appears in various contexts in the relatively short history of software engineering. This fact seems natural because the processes like software comprehension, design, refinement and implementation are done incrementally in practice. Due to this common fact the incrementality principle is interpreted often superficially in the software engineering literature. The aim of this paper is to highlight the ubiquity of the incrementality utility in software engineering literature and to clarify the concepts behind its role in various contexts.

1. Background and motivation

The notion of incrementality is being applied simultaneously both in the field of artificial intelligence (e.g. in the field of machine learning) and software engineering.

While reading the software engineering literature we observe that the incrementality principle is mentioned and utilized frequently in various contexts in its relatively short history (see in the next paragraph). This ubiquitous presence of the incrementality principle, almost in every software development paradigm, motivated the investigation towards unrevealing the similarities and differences between its interpretations in various contexts. Another source of the motivation is the challenge to condense the knowledge about the incrementality principle used in contemporary experimental software engineering practices. The aim of this paper is to pave the route towards the understanding and reuse of this notion in the future contexts. In order to achieve this goal, we will investigate the utilization of the incrementality utility in several selected software development approaches.

Received by the editors: September 13, 2006.

2000 *Mathematics Subject Classification.* 68N19, 68N30.

1998 *CR Categories and Descriptors.* D.1.2 [**Software**]: Programming Techniques – *Automatic Programming*; D.2.2 [**Software**]: Software Engineering – *Design Tools and Techniques*.

2. The ubiquity of incremental tasks

The following selective sample of works (not exhaustive) shows the wide range of the usage of the incrementality utility. A brief historical overview of the *"incremental and iterative development"* is presented in the work of C. Larman and V. Basili [11]. This work summarizes the role of the iterative and incremental software development through significant software projects since the mid-1950s. It focuses on the incrementality utility, applied in the software engineering processes, from the managerial point of view. Describes the driving thoughts and misbeliefs, which were behind the practices applied in the past decades in the field of software engineering.

The incrementality concept is stressed and dealt also in the field of program comprehension, which is already a matured topic with standalone conferences [8]. It is not a trivial task to understand the architecture of object-oriented programs [22]. It is obvious, that to enhance the functionality or to add new features requires to comprehend the task in more detail. As software engineering topics evolved, the incremental comprehension of programs came into focus. In other words it means that without the thorough analysis it is impossible to make effective reform or re-engineering. Remarkable is the statement of K. Nygaard [5] who said: "to program is to understand". We accept in general that comprehension is also a continuous iterative and incremental process. The fact that problem solving does not progress in a linear manner from one activity to the next is highlighted as the conjecture: "empirically based models mature from understanding to explaining and predicting capability". This conjecture is explained in the handbook of authors A. Endres and D. Rombach [6, chap. 12], which is devoted to the empirical aspects of software engineering.

The recent work of C. Larman [10] stresses the role of the incrementality against the waterfall model in the software development. In addition, the work stresses its crucial role in the history of the software engineering and considers it as a fundamental revolutionary change against the waterfall model of the software development. Authors R. E. Fairley and M. J. Wilshire [7] exhaustively identify the nature of the iterative rework. They point to the fuzziness between the avoidable and unavoidable rework and the incrementality issue is dealt from this point of view.

The field of software design also uses the notion of iteration and incrementality, e.g. Arlow and Neustadt characterize iterations, in association with the Unified Process [3], as "mini projects, which are easier to manage and complete than the original large SW development project".

Machine learning differentiates between nonincremental and incremental learning (e.g. in [14]). The same terms are marked in [17] as revolutionary and evolutionary learning strategies. In essence, the difference between these two definitions

lies in the fact that the non-incremental (revolutionary) approach is based on one-shot experience and the incremental (evolutionary) learning allows the learning process to take place over time in a continuous and progressive way, taking into consideration also the history of the training sets at building the inferred rules.

Incremental change plays important role in practical software engineering. At the present time the incremental change in object-oriented programs are in the focus (see for instance [20]). These activities investigate the impact of adding new functionalities into the code and finding the relevant program dependencies. Incrementality is of importance to software visualization too [9]. The aim is to get a better comprehension of the software behavior by representing complex structures graphically.

We may conclude the above mentioned remarks by the statement of E. W. Dijkstra: "the only available technique for effective ordering of one's thoughts is by separation of concerns". This approach leads to the observation of new facts and in ths way to improve incrementally the previous knowledge. In summary, the incrementality utility is a broad term and is in the focus of software engineers from various aspects. In the next sections we narrow the focus towards revealing the principles behind these experiences and endeavours.

3. Iteration and incrementality

The objective of the software development is to model a certain aspect or abstraction of the reality [16]. Software engineering, as every engineering discipline, is characterized by trials and errors, which are necessary steps for clarifying the comprehension of the requirements, design and implementation. It consists of many small steps and it is necessary to take into account plenty of details. Software systems are becoming more and more complex over time. They are changing and modified; the complexity arises and we need more time to comprehend them. The maintenance of the final code and other software artifacts consumes more time too.

As noted in Section 2, there are many approaches that present some aspects of the incrementality utility and are used under names like incremental learning, evolutionary and revolutionary rework, program synthesis and incremental building. Therefore, a clear definition of the *"iteration"* and the *"incrementality"* turns out to be vital. Here are the definitions:

- We define that *"iteration"* refers to repeating an activity, e.g. phases, in the software development process. Iteration is applied e.g. in refactoring when developers perform semantics-preserving structural transformations usually in small steps. Motivation for the improvement may be focused towards the enhancement of the efficiency of the code with respect to the time or space complexity or towards the improvement the structure so that developers can more easily understand, modify, evolve

and test it. The research domain that addresses this problem is referred also as restructuring.

- On the other hand *"incrementality"* refers to the process of adding new functionalities through successive implementations. This is a significant and essential difference to the iteration and deserves much more attention. First of all the incrementality principle has its mathematical roots and is explained in the theory of inductive inference [2]. This approach to problem solving is also called generalization and will be explained in more detail in Section 4. Incremental software development is sometimes called build a little, test a little. We may observe the similarity between building concepts and models in software engineering and building hypotheses in mathematics. This process is very clearly highlighted in Polya's classic work, "How to Solve It" [19].

The empirical evidence from the real-world software suggests that learning or incremental program development is possible only when the data are presented incrementally. For instance programming languages dispose with constructs, which help to postpone solving some issues. As an example is the exception mechanism in the object-oriented programming. This process makes, of course, the software more complex and drifts away from the original design. These facts may lower the quality of the software but it is the task of the validation and verification to ensure the formal quality software.

To sum up, the incrementality principle is ubiquitous in the literature devoted to the software engineering. After every step we discover new requirements, analyze them, plan, implement and test. Every iteration adds new insights and the system grows in this way, or logically clarifies. In other words, software programs are too complex to try to get the details of any one artifact entirely correct without some amount of experimentation. Software developers' ideas are evolving as they work and the steps are associated with the progress, this is evidence.

4. Incrementality in the software development

In the next sections we will focus on the application of the incrementality principle in software engineering paradigms, which are aimed at program synthesis. The selected paradigms are as follows:

(1) Programming by examples,
(2) Automatic program synthesis from specifications,
(3) Test driven programming,
(4) Programming by sketching.

4.1. **Programming by examples.** Programming by examples (positive or negative) seems popular and recurrent topic to the software research community, but often is neglected the fact that this approach has very limited usage actually and it is not generally applicable [21]. Let us analyze it in more detail. The principle of

Programming By Example (PBE), or Programming By Demonstration (PBD) or doing by watching, was investigated intensively around the eighties and a survey is available in the work of H. Lieberman [12].

We have to be aware that the paradigm of programming by example has theoretical background in the theory of inductive inference, as we mentioned in Section 3. Thought provoking is again the work of G. Polya [19], who declares the fundamental role of the mathematical induction in the problem-solving domain.

First attempts to synthesize programs by examples were done in the field of the automata theory. The task was to synthesize finite automata from a set of examples. The examples were defined by set of pairs (state and transition). The work of A. W. Biermann [4], e.g., describes practical results from experiences with implemented incremental algorithms.

Programming by example is recently also mentioned as an exciting new technology in the work of H. Lieberman and C. Fry: Will Software Ever Work? [13]. But in reality this approach is old and characterized in the work of A. Endres and D. Rombach: A Handbook of Software and Systems Engineering. They state that in fact the code generated from test cases will satisfy all test cases but we will always need one more test case because the generalization delivers a model which need not cover all test cases [6, p. 89].

What is the definition of the incremental algorithm? An algorithm is incremental if, for any given training example $e_1 \dots e_n$, it produces a sequence of hypotheses $h_0, h_1, \dots, h_n$, such that h_{i+1} depends only on h_i and the current example e_i. Now we may substitute various software artifacts, e.g. components, for h_i, which may appear in various contexts of the software development cycle (see Figure 1).

In other words it means that the hypothesis built by inductive inference will be compatible only with the current set of the proposed examples and nothing more. E.g., when we construct a cycle, we widen the scope of the algorithm in inductive way. This is the inherent feature of the inductive inference that newly generated hypothesis need not follow the intended functionality.

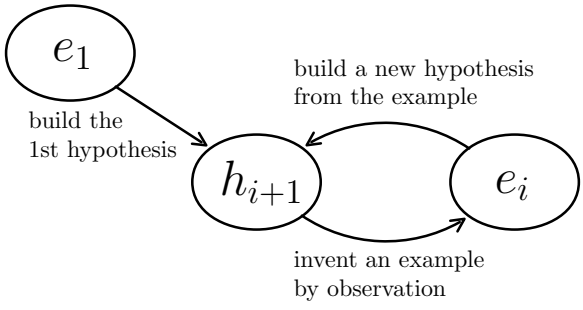


FIGURE 1. The principle of the incremental algorithm.

4.2. **Automatic program synthesis from specifications.** Synthesis of programs from specifications is a methodology, which allows to construct the program code automatically from the specification. The essence of these methods lies in the transformations, which could be used for the modification of the specifications and in order to reach the final code. The final code is constructed by the application of transformation rules to the specifications. Such a program is verified against the specification and that is why it is not necessary to prove the correctness of the program additionally. Recent review can be found in [18]. The problem with this approach is that it is an illusion to expect that perfect requirements can be formulated ahead of time. Both developers and users may need some feedback. They require a *(learning cycle)*. This is cited again from the Handbook of Software and Systems Engineering [6, p. 15]. The role of the incrementality is hidden behind the phrase learning cycle. In other words, we cannot predict the correctness of our abstraction with the reality. This approach is suitable only for trivial tasks.

4.3. **Test driven programming.** Incrementality principle may be observed both in the specification and in the implementation phases of the software development. The following simple figure illustrates the idea.

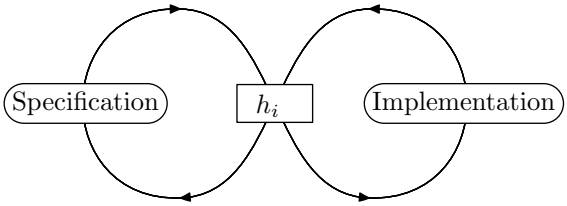


FIGURE 2. Location of the model h_i.

On the specification (left hand) side we may consider the examples e_i as test cases t_i against which the requirements are clarified and in this way h_i is continuously modified. If we consider h_i as a fixed artifact (e.g. program) and we intend to test the program (approaching the picture from the right hand side) then we may consider the tests t_i as examples e_i, which have to be accepted or ejected by the h_i. Of course, testing shows only the presence of bugs and not the absence of errors (Dijkstra's law [5]). This simple picture shows the relativity of the concepts as *"example"* and *"test"*. Both concepts' interpretations depends on the context. To sum up, the interpretation of the examples and the tests depends only on the actual agreement whether h_i is fixed or not. We stress again, that the principle behind building h_i is the incremental principle as mentioned in Section 4.1. If a new example has to be embedded into the actual model then the incremental algorithm has to modify (including the regression) the already prepared model. In each case the final model has to comply with the new set of examples.

4.4. **Programming by sketching.** Programming by sketching is motivated by the fact that programmers may acquire powerful help during the software development. In fact this approach helps in the software development with the visualization of available components. The group led by R. Bodik [15] recently experiments with this approach in practice. This approach argues that the final work of coding has to be done by computer and also the computer has to ease developers' work by provision of efficient support, or data retrieval, in order to create the final code more efficiently. Programming by sketching relies on the fact that programmers are sketching actually the skeletons (e.g. applying patterns or components off the shelf) of programs and the coding is done automatically. This approach is also part of the agile programming [1]. Following our definition of the incremental algorithm in Section 4.1, we may observe that in this case the artifact of the incrementality paradigm is the code segment or component.

5. Conclusion

The aim of this paper was to show and discuss the role of the incrementality principle in selected software development paradigms, which do not fit together at the first sight and in this way to provoke thinking. We have tried to synthesize the scattered fragments of the applied incrementality principle in the software engineering literature in order to create a more condensed foresight. We know that we did not invent a new solution to an existing problem, we dug out rather old ideas and observed them in new contexts. There is no doubt that new paradigms will emerge in the future. It is the task of the informatics to show the role of the principles in the emerging fashionable ideas.

References

[1] Scott W. Ambler. *The Object Primer 3rd Edition, Agile Model Driven Development with UML 2*. Cambridge University Press, 2002. ISBN: 0-521-54018-6.

[2] D. Angluin and C. H. Smith. Inductive Inference: Theory and Methods. *Computing Surveys*, 15(3):238–269, September 1983.

[3] Jim Arlow and Ila Neustadt. *UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design*. Addison-Wesley, second edition, June 2005. ISBN: 0-321-32127-8.

[4] Alan W. Biermann. Automatic programming. In Stuart C. Shapiro, editor, *Encyclopedia of Artificial Intelligence*. John Wiley and Sons, January 1992.

[5] L. Böszörményi, S. Podlipnig (with contributions of Manfred Broy, Tove Dahl, and Marius Nygaard). *People behind Informatics (In memory of Ole-Johann Dahl, Edsger W. Dijkstra, Kristen Nygaard)*. Institute of Information Technology, University of Klagenfurt, 2003.

[6] A. Endres and S. Rombach. *A Handbook of Software and Systems Engineering; Empirical observations, laws and theories*. Pearson, Addison Wesley, May 2003.

[7] Richard E. Fairley and Mary Jane Wilshire. Iterative rework: The good, the bad and the ugly. *IEEE Computer*, 38(9):34–41, September 2005.

[8] `http://www.ieee-iwpc.org`, International Conferences on Program Comprehension.

[9] C. Knight and M. Munro. *Visual Information: Amplifying and Foraging, Proceedings of SPIE, San Jose, USA*, volume 4032. International Society for Optical Engineering, January 2001. ISBN: 0-8194-3980-0.
[10] C. Larman. History and evidence of evolutionary versus waterfall methods. `http://it.sun.com/eventi/jc05/pdf/02_Larman.pdf`. Java Conference 05, 22–23 June 2005, Milan, Italy.
[11] C. Larman and V. R. Basili. Iterative and incremental development: A brief history. *IEEE Computer*, 36(6):46–57, June 2003.
[12] H. Liebermann, editor. *Your Wish is My Command: Programming by Example*. Morgan Kaufmann, San Francisco, February 2001.
[13] H. Liebermann and C. Fry. Will software ever work? *Communications of the ACM*, 44(3):122–124, March 2001.
[14] K. Machová. *Machine learning*. Faculty of electrical engineering and informatics, Technical University of Košice, elfa, 2002.
[15] David Mandelin, Lin Xu, and Rastislav Bodík. Jungloid Mining: Helping to Navigate the API Jungle. *ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI05)*, pages 48–61, 2005. ISSN: 0362-1340.
[16] B. Meyer. Reality: A cousin twice removed. *IEEE Computer*, 29(7):96–97, July 1996.
[17] R. Michalski. Knowledge repair mechanisms: Evolution vs. revolution. Technical report, Department of Computer Science, University of Illinois, February 1985. Reports of the Intelligent Systems Group, ISG 85-11, UIUCDCS-F-85-941.
[18] Alberto Pettorossi and Maurizio Proietti. Rules and strategies for transforming functional and logic programs. *ACM Computing Surveys*, 28(2):360–414, June 1996.
[19] G. Polya. *How to solve it: A New Aspect of Mathematical Method*. Princeton University Press, 2nd edition, 1957.
[20] V. Rajlich. Incremental change in object-oriented programming. *IEEE Software*, 21(2):62–69, July/August 2004. ISSN:.
[21] Ladislav Samuelis. Synthesis of programs by examples. Technical report, Budapest University of Technology, May 1990. PhD thesis,(in Hungarian).
[22] N. Wilde and B. Huitt. Maintenance support for object-oriented programs. *IEEE Transactions on Software Engineering*, 18(12):1038–1044, 1992.

Acknowledgements

The research was supported by the following grants:

- Mathematical Theory of Programming and its Application in the Methods of Stochastic Programming. Scientific grant agency project (VEGA) No. 1/2181/05
- Technologies for Agent-based and Component-based Distributed Systems Lifecycle Support. Scientific grant agency project (VEGA) No. 1/2176/05
- Evaluation of operational parameters in broadband communicational infrastructures: research of supporting platforms. Scientific grant agency project (VEGA) No. 1/2175/05

Department of Computers and Informatics, Technical University of Košice, Letná 9, 042 00 Košice, Slovakia

E-mail address: `Ladislav.Samuelis@tuke.sk, Csaba.Szabo@tuke.sk`

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LI**, Number 2, 2006

EXTENDED *P*-SENSITIVE *K*-ANONYMITY

ALINA CÂMPAN AND TRAIAN MARIUS TRUŢĂ

Abstract. In this paper we introduce a new privacy protection property, called *extended p-sensitive k-anonymity*, which is an extension of the *p*-sensitive *k*-anonymity property [16]. The new property is aware of confidential attributes hierarchies and of the existence of protected not ground-level confidential attributes values, situation not considered by previous work done in this direction. We describe our model and indicate an algorithm for enforcing extended *p*-sensitive *k*-anonymity to masked microdata.

Keywords: privacy protection, anonymity, generalization.

1. Introduction

To protect the privacy of individuals in the present digitized world became an increasingly difficult task. Large amounts of *microdata* (datasets where each tuple belongs to an individual entity) are collected by different agencies. Some of these microdata need to be released, for various purposes, to other parties. Obviously, direct identifying information such as *SSN*, *Name* is eliminated from the microdata before releasing it, for privacy protection. But even modified this way, the datasets could still present vulnerabilities that can be exploited by intruders, i.e. persons whose goals are to identify specific individuals and to use the confidential information they discover for malicious purposes. More elaborated techniques are needed in order to ensure a reliable and controlled privacy protection when microdata are released.

In recent years, the use and the disclosure of confidential information was subject to privacy regulations promulgated in different domains [4, 8, 7]. All these regulations, together with the necessity of collecting personal information, have fed the interest in privacy research.

Techniques to avoid the disclosure of confidential information exist in the literature [1, 17]. Among them, the *k*-anonymity property required for the released

Received by the editors: September 20, 2006.

2000 *Mathematics Subject Classification.* 68P15.

1998 *CR Categories and Descriptors.* 68P15 [**Computer science**]: Theory of data – *Database theory.*

microdata (a.k.a. masked microdata) was recently introduced [13, 14] and extensively studied [3, 5, 10, 16]. This property requires that in the released microdata every tuple will be indistinguishable from at least (k-1) other tuples with respect to a subset of attributes called quasi-identifier attributes or key attributes.

Recent results have showed that k-anonymity fails to protect the privacy of individuals in all situations [16]. Two similar models called p-sensitive k-anonymity [16] and l-diversity [11] were proposed in the literature in order to deal with the problems of the k-anonymity model. The p-sensitive k-anonymity property requires, in addition to k-anonymity, that for each group of tuples with the identical combination of quasi-identifier attributes values, the number of distinct values for each confidential attribute (attribute which values must be protected) must be at least p within the same group.

However, depending on the nature of the confidential attributes, even the p-sensitivity property still permits the information to be disclosed. We identify, in this paper, situations when p-sensitivity property is not enough for privacy protection and we propose a solution to overcome the identified problem: the extended p-sensitive k-anonymity model and an algorithm to enforce this property.

2. Concepts and Notations

Let IM be the initial microdata and IM be the released (a.k.a. masked) microdata. IM consists in a set of tuples over an attribute set. The attributes characterizing microdata are classified into the following three categories:

- $I_1, I_2, \ldots, I_m$ are *identifier* attributes such as *Name* and *SSN* that can be used to identify a record. These attributes are present only in the initial microdata because they express information which can lead to a specific entity.
- $K_1, K_2, \ldots, K_n$ are *key* or *quasi-identifier* attributes such as *ZipCode* and *Age* that may be known by an intruder. Quasi-identifier attributes are present in the masked microdata as well as in the initial microdata.
- $S_1, S_2, \ldots, S_r$ are *sensitive* or *confidential* attributes such as *Principal-Diagnosis* and *ICD9Code* that are assumed to be unknown to an intruder. Confidential attributes are present in the masked microdata as well as in the initial microdata.

While the identifier attributes are removed from the released microdata, the quasi-identifier and confidential attributes are usually kept in the masked microdata and released to the researchers.

A general assumption, as noted, is that the values for the confidential attributes are not available from any external source. This assumption guarantees that an intruder can not use the confidential attributes values to increase his/her chances

of disclosure. Unfortunately, an intruder may use record linkage techniques [18] between quasi-identifier attributes and external available information to glean the identity of individuals from the masked microdata. To avoid this possibility of disclosure, one frequently used solution is to modify the initial microdata, more specifically the quasi-identifier attributes values, in order to enforce the k-anonymity property.

Definition 1. (***k-anonymity property***): The ***k-anonymity property*** for a masked microdata (MM) is satisfied if every combination of quasi-identifier attribute values in MM occurs k or more times.

Based on this definition, in a masked microdata that satisfy k-anonymity property, the probability to correctly identify an individual is at most $1/k$. By increasing k the level of protection increases, along with the changes to the initial microdata.

To achieve k-anonymity, existing k-anonymization algorithms generally proceed by using generalization and suppression [13, 15]. Generalization of the quasi-identifier attributes is used widely for k-anonymization. It consists in replacing the actual value of an attribute with a less specific, more general value that is faithful to the original [15]. Generalization is either based on predefined (static) domain and value generalization hierarchies [15], or is conducted using a hierarchy-free model [10].

The k-anonymity property ensures protection against identity disclosure, i.e. the identification of an entity (person, institution). However, as we will show next, it does not protect the data against attribute disclosure, which occurs when the intruder finds something new about a target entity. The two disclosure types are independent. None of them does imply the other.

Consider the masked microdata example below, where the set of quasi-identifier attributes is composed of *Age*, *ZipCode* and *Gender*, and *Illness* is the sensitive attribute:

TABLE 1. Patient masked microdata satisfying 2-anonymity

Tuples	**Age**	**ZipCode**	**Gender**	**Illness**
r_1	50-60	43102	Male	Colon Cancer
r_2	30-40	43102	Female	Breast Cancer
r_3	30-40	43102	Female	HIV
r_4	20-30	43102	Male	Diabetes
r_5	20-30	43102	Male	Diabetes
r_1	50-60	43102	Male	Heart Disease

Identity disclosure does not happen in this masked microdata, as its construction guarantees that for every existing combination of values for *Age*, *ZipCode* and *Gender* there are at least two tuples that have the respective combination of values. However, assuming that external information in Table 2 below is available, attribute disclosure can take place. If the intruder knows that in the masked microdata the *Age* attribute was generalized to multiples of 10, he can deduce that both Sam and Eric have Diabetes, even he doesn't know which tuple, r_4 or r_5, corresponds to what person. This example shows that k-anonymity fails to protect sometimes against attribute disclosure, even if it protects from identity disclosure.

TABLE 2. External information for Patient example

Name	**Age**	**Gender**	**ZipCode**
Sam	29	Male	43102
Gloria	38	Female	43102
Adam	51	Male	43102
Eric	29	Male	43102
Dana	34	Female	43102
Don	51	Male	43102

For dealing with this flaw in privacy, another model, called p-sensitive k-anonymity was introduced in [16]. A similar privacy model, called l-diversity, is described in [11].

Definition 2. (***p-sensitive k-anonymity property***): The masked microdata (MM) satisfies ***p-sensitive k-anonymity property*** if it satisfies k-anonymity and for each group of tuples with the identical combination of key attribute values that exists in MM, the number of distinct attributes for each confidential attribute is at least p within the same group.

Sometimes, similar to the quasi-identifier attributes, the domain of the sensitive attributes, especially the categorical ones, can also be organized according to some hierarchies. For example, in medical datasets, the *Illness* attribute has values as specified by the ICD9 codes (see Figure 2). The different types of diseases are organized in a tree hierarchy of values. The attribute values are very specific, for example they can represent different types of cancer, which are all descendants of cancer value. The initial microdata contain as values for the *Illness* attribute values from the lowest level of the hierarchy (i.e. from the leaf nodes). In these conditions, the data owner can be interested in protecting not only these most specific values, but also information found at higher levels. For example, the information that a person has cancer needs to be protected, regardless of the

cancer type she has. If p-sensitive k-anonymity property is enforced for masked microdata, it is possible that in a group with p distinct *Illness* attribute values, all of them to be descendants of the cancer node in the corresponding hierarchy. To avoid such situations, we introduce the concept of extended p-sensitive k-anonymity, which is aware of the existence of protected values not only at the ground level.

3. Extended p-sensitive k-anonymity Property

Let S be a categorical confidential attribute we want to protect against attribute disclosure. S has associated predefined (static) domain and value generalization hierarchies [15]. HD_S is the domain generalization hierarchy of attribute S. The values from different domains of this hierarchy HD_S are represented in a tree HV_S called value generalization hierarchy. We illustrate domain and value generalization hierarchy in Figure 1 for attributes *ZipCode* and *Gender*, which are quasi-identifier attributes.

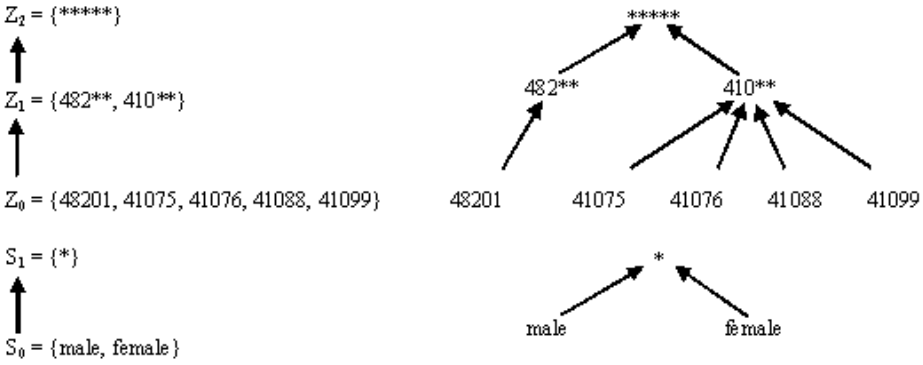


FIGURE 1. Examples of domain and value generalization hierarchies

Figure 2 shows a part of the ICD9 value generalization hierarchy.

Some zones of a value generalization hierarchy HV_S, associated to the sensitive attribute S, need to be protected.

Unlike the quasi-identifier attributes, the values of a sensitive attribute cannot be generalized in the masked microdata for protection, because this would affect the quality of the released data w.r.t. subsequent tasks that will be performed on it, such as data mining tasks.

The protection will be achieved by enforcing k-anonymity (for identity disclosure protection) while ensuring the extended p-sensitivity (for attribute disclosure protection). The heterogeneity of the confidential attributes values in each of the

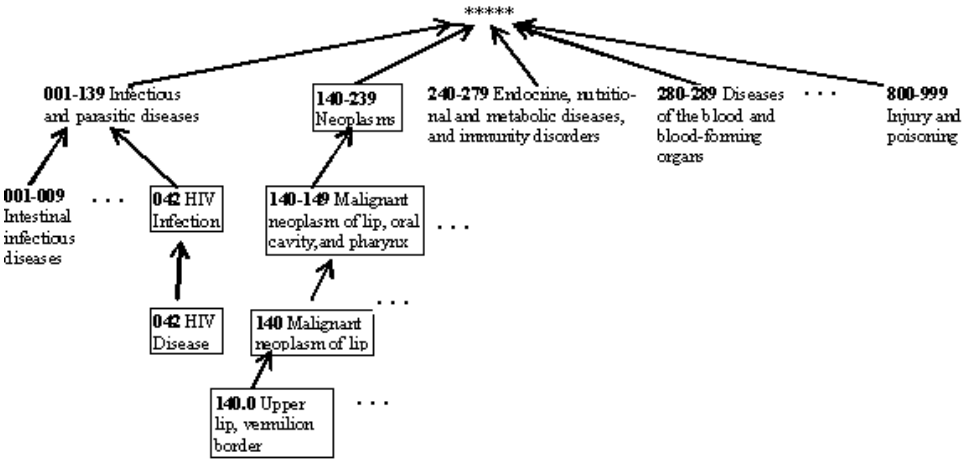


FIGURE 2. ICD9 disease hierarchy and codes

groups formed by k-anonymizing the data is to be achieved not only at the ground values level, but for all the values declared protected in HV_S. The data owner has to mark (declare) which are the protected "zones" in a confidential attribute hierarchy. In Figure 2, the protected values in the value generalization hierarchy of attribute *Illness* are bordered. We require that all the descendants of a protected value to also be protected. In other words, if an internal node of a value generalization hierarchy is protected, the entire subtree rooted in that node needs to be protected. All values at the ground level are considered to be protected. The semantics of a node (its value) being protected is as follows: if extended p-sensitivity is enforced for a microdata w.r.t. the confidential attribute S, this means that each group of tuples with the identical combination of quasi-identifier attributes values contains at least p distinct values for S that respect the condition that, any two of them are not descendants of a common protected value (i.e. any two of these values do not have a common protected ancestor). For example, if Neoplasms is a protected value, no group will contain only descendant values of Neoplasms, even if there are p distinct such values in that group. So, every group containing descendant values of Neoplasms will also contain at least p - 1 different values that are not descendants of Neoplasms. Of course, for these values also functions the same condition. We will refer to the property enounced here informally as extended

p-sensitive k-anonymity. To define the extended p-sensitive k-anonymity property we need to introduce several other concepts.

Requirements: Let S be a confidential attribute and HV_S its value generalization hierarchy. The following two requirements must be met by the protected values in HV_S:

- All ground values in HV_S are protected.
- All the descendants of a protected internal value in HV_S are protected.

Definition 3. A protected value in the value generalization hierarchy HV_S of a confidential attribute S is called ***strong*** if none of its ascendants (including the root) is protected.

Property 1. A protected value is strong if its parent is not protected.

This property results from the definition of strong values and the first requirement imposed to HV_S.

Definition 4. We call ***protected subtree*** of a hierarchy HV_S a subtree in HV_S that has as root a strong protected value.

Definition 5. (***extended p-sensitive k-anonymity property***): The masked microdata (MM) satisfies ***extended p-sensitive k-anonymity property*** if it satisfies k-anonymity and for each group of tuples with the identical combination of key attribute values that exists in MM, the values of each confidential attribute S within that group belong to at least p different protected subtrees in HV_S.

Extended p-sensitive k-anonymity can not be enforced for any microdata set. We give next several necessary conditions that must be satisfied by a microdata set in order to be possible to enforce extended p-sensitive k-anonymity for it. These conditions are adapted from [16], where they were enounced w.r.t. the basic p-sensitive k-anonymity property.

Condition 1. p must be less than or equal to k (i.e. $p \leq k$).

Justification: In a group of k tuples there can not be more than k different values for a confidential attribute S.

Condition 2. The value generalization hierarchy HV_S of every confidential attribute S must contain at least p different protected subtrees.

We use the following notations for a microdata IM:

- n - the number of tuples in IM;
- q - the number of confidential attributes in IM;

- s_j - the number of distinct strong protected values in HV_{S_j} that are ascendants of all the values that the confidential attribute S_j has in IM, $1 \leq j \leq q$;
- f_i^j- the descending ordered frequency set for the confidential attribute S_j, $1 \leq i \leq s_j$, $1 \leq j \leq q$. The frequency set is computed after the confidential values in the microdata are generalized to their corresponding strong protected values;
- cf_i^j - the cumulative descending ordered frequency set for the confidential attribute S_j, $1 \leq j \leq q$. The frequency set is computed after the confidential values in the microdata are generalized to their corresponding strong protected values;
- $cf_i = max_{j=1,q}(cf_i^j)$, $1 \leq i \leq min_{j=1,q}(s_j)$.

Condition 3. The maximum allowed number of combinations of quasi-identifier attribute values in the masked microdata MM is $min_{i=1,p-1}\frac{n-cf_{p-i}}{i}$.

The proof of this property for basic p-sensitive k-anonymity can be found in [16]. For extended p-sensitivity, the confidential attributes values are first generalized in the initial microdata, to their strong ancestors, and then the property for basic p-sensitivity is true for the resulted dataset.

4. Enforcing Extended p-sensitive k-anonymity Property to Microdata

At a closer look, extended p-sensitive k-anonymity for a microdata is equivalent to p-sensitive k-anonymity for the same microdata where the confidential attributes values are generalized to their first protected ancestor, starting from the hierarchy root (their strong ancestor). Consequently, in order to enforce extended p-sensitive k-anonymity to a dataset, the following two-steps procedure can be applied:

- Each value of a confidential attribute is generalized (only temporarily) to its first protected ancestor (including itself), starting from the hierarchy root, i.e. to its strong ancestor.
- Any algorithm which can be used for p-sensitive k-anonymization is applied to the modified dataset. Such an algorithm is indicated in [16]. In the resulted masked microdata the original values of the confidential attributes are restored.

The dataset obtained following these steps respects the extended p-sensitive k-anonymity property.

5. Experimental Results

We performed a set of experiments to test how the existing k-anonymizing algorithms break the p-sensitivity and extended p-sensitivity properties. These experiments show that attribute disclosure can happen when only k-anonymity is enforced for microdata and, therefore, emphasize the need to protect the data against disclosure, beyond the k-anonymity.

In our experiments we used data based on the Adult database from the UC Irvine Machine Learning Repository [12]. This database has become the benchmark in data privacy field, being used by many researchers [10]. We considered *Age*, *Marital_Status*, *Race* and *Sex* from adult data as being the set of quasi-identifier attributes. The confidential attributes are *Pay*, *Capital_Gain*, *Capital_Loss* and *Tax_Amount*. The *Pay* attribute is considered to have two distinct values, $\leq$50K, >50K, and describes whether a person makes or not over 50K a year. The *Capital_Gain* attribute can have three distinct values (1000, 2000, 3000), *Capital_Loss* has four distinct values (1000, 2000, 3000, 4000), and *Tax_Amount* has ten distinct values (100, 200, ..., 1000). The *Tax_Amount* attribute is the only confidential attribute that has an associated generalization hierarchy with more than one level. The value generalization hierarchy is depicted in Figure 3, and the protected values are bordered, the strong protected values are bold bordered.

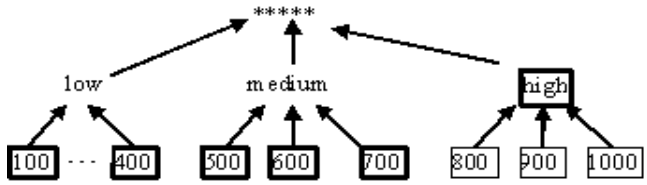


Figure 3. Value generalization hierarchy for *Tax_Amount*

We k-anonymized 400 records randomly chosen from adult database, for k=3 and k=5, using: the anonymization algorithm based on clustering which is described in [6]; the binary search algorithm presented in [13]. The quasi-identifier attributes were generalized w.r.t. the generalizations outlined in Table 3.

The produced masked microdata respect of course the requirements imposed by the k-anonymity property, but it contains several records that contradict the conditions in p-sensitive k-anonymity and in extended p-sensitive k-anonymity. Table 4 summarizes the results of our experiments: the number of tuples and the number of groups of tuples sharing common values for the quasi-identifier attributes that contradict the two properties. So, this experiment shows that

TABLE 3. Adult database quasi-identifier attributes generalization

Attribute	**First Generalization**	**Second Generalization**	**Third Generalization**
Age	10-years range	$\leq$50 and >50 groups	One group
Marital_Status	Single or Married	One group	-
Race	White, Black or Other	White or Other	One group
Sex	One group	-	-

for microdata masked to satisfy the k-anonymity property, disclosure channels still exist so that confidential attributes values can be inferred. P-sensitive k-anonymity property, basic or extended, need to be enforced to the microdata in order to avoid such disclosure situations. We used for k-anonymization two different algorithms, reported in [13], and respectively in [6].

TABLE 4. Attribute disclosures for a masked microdata set with k-anonymity property

***k*-anonymity with [13] algorithm**	**No of attribute disclosures w.r.t. *p*-sensitivity**
2-anonymity	6
3-anonymity	2

***k*-anonymity with [6] algorithm**	***Pay***		***Capital_Gain***		***Capital_Loss***		***Tax_Paid***	
3-anonymity	2-sensitivity disclosures							
	Tuples	Groups	Tuples	Groups	Tuples	Groups	Tuples	Groups
	38	12	36	12	15	5	0	0
5-anonymity	3-sensitivity disclosures							
	Tuples	Groups	Tuples	Groups	Tuples	Groups	Tuples	Groups
	-	-	164	31	30	6	11	2

***k*-anonymity with [6] algorithm**	***Tax_Paid* extended *p*-sensitivity disclosures**	
3-anonymity	extended 2-sensitivity disclosures	
	Tuples	Groups
	3	1
5-anonymity	extended 3-sensitivity disclosures	
	Tuples	Groups
	11	2

6. Conclusions and Future Work

In this paper, we introduced a new privacy protection property, called extended p-sensitive k-anonymity, which is an extension of the p-sensitive k-anonymity property. Next, we presented three necessary conditions a masked microdata must satisfy in order to have extended p-sensitive k-anonymity property. Last, we indicated how an algorithm that generates k-anonymous microdata can be modified to enforce extended p-sensitive k-anonymity property. Our experiments showed that p-sensitive k-anonymity property, basic or extended, need to be enforced to the masked microdata in order to avoid attribute disclosure situations.

In future work, we will create masked microdata that satisfy extended p-sensitive k-anonymity using the existing algorithms for k-anonymity with the addition of the three necessary conditions, and we will compare the running time of these modified algorithms against the existing algorithms that search for k-anonymity only.

References

[1] Adam N. R., Wortmann J. C. (1989), "Security Control Methods for Statistical Databases: A Comparative Study." ACM Computing Surveys, Vol. 21, No. 4, 515–556.

[2] Aggarwal G., Feder T., Kenthapadi K., Khuller S., Panigrahy R., Thomas D., and Zhu A. (2006), "Anonymizing Tables", Proceedings of the ACM PODS Conference, 153–162.

[3] Aggarwal G., Feder T., Kenthapadi K., Motwani R., Panigrahy R., Thomas D., and Zhu A. (2005), "Achieving Anonymity via Clustering", Proceedings of the 10th International Conference on Database Theory.

[4] Agrawal R., Kiernan J., Srikant R., Xu Y. (2002), "Hippocratic Databases", Proceedings of the 20th International Conference on Very Large Databases (VLDB), Hong Kong, 143–154.

[5] Bayardo R.J, Agrawal R. (2005), "Data Privacy through Optimal k-Anonymization", Proceedings of the IEEE International Conference of Data Engineering, 217–228.

[6] Byun J.W., Kamra A., Bertino E, Li N. (2006), "Efficient k-Anonymity using Clustering Technique", CERIAS Tech Report 2006-10.

[7] GLB (1999), "Gramm-Leach-Bliley Financial Services Modernization Act", Available online at http://banking.senate.gov/conf/.

[8] HIPAA (2002), "Health Insurance Portability and Accountability Act", Available online at http://www.hhs.gov/ocr/hipaa.

[9] LeFevre K., DeWitt D., Ramakrishnan R. (2005), "Incognito: Efficient Full-Domain K-Anonymity", Proceedings of the ACM SIGMOD, Baltimore, Maryland, 49–60.

[10] LeFevre K., DeWitt D., Ramakrishnan R. (2006), "Mondrian Multidimensional K-Anonymity", Proceedings of the IEEE International Conference of Data Engineering, Atlanta, Georgia.

[11] Machanavajjhala A., Gehrke J., Kifer D. (2006), "l-diversity: privacy beyond k-anonymity", Proceedings of the 22nd IEEE International Conference on Data Engineering.

[12] Newman D.J., Hettich S., Blake C.L., Merz C.J. (1998), "UCI Repository of Machine Learning Databases", available at www.ics.uci.edu/ mlearn/MLRepository.html, University of California, Irvine, 1998.

[13] Samarati P. (2001), "Protecting Respondents Identities in Microdata Release", IEEE Transactions on Knowledge and Data Engineering, Vol. 13, No. 6, 1010–1027.
[14] Sweeney L., (2002), "k-Anonymity: A Model for Protecting Privacy", International Journal on Uncertainty, Fuzziness, and Knowledge-based Systems, Vol. 10, No. 5, 557–570.
[15] Sweeney L., (2002), "Achieving k-Anonymity Privacy Protection Using Generalization and Suppression", International Journal on Uncertainty, Fuzziness, and Knowledge-based Systems, Vol. 10, No. 5, 571–588.
[16] Truta T.M., Bindu V. (2006), "Privacy Protection: P-Sensitive K-Anonymity Property", Proceedings of the Workshop on Privacy Data Management, In Conjunction with 22th IEEE International Conference of Data Engineering (ICDE), Atlanta, Georgia.
[17] Willemborg L., Waal T. (ed) (2001), "Elements of Statistical Disclosure Control", Springer Verlag.
[18] Winkler W.E. (1994), "Advanced Methods for Record Linkage", Proceedings of the Section on Survey Research Methods, American Statistical Society, 467–472.

Babeş Bolyai University, Cluj Napoca, Romania
E-mail address: alina@cs.ubbcluj.ro

Northern Kentucky University, Highland Heights, USA
E-mail address: trutat1@nku.edu

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LI**, Number 2, 2006

RECOGNIZING TEXTUAL ENTAILMENT BY THEOREM PROVING APPROACH

DOINA TĂTAR AND MILITON FRENŢIU

Abstract. We present two original methods for recognizing textual inference. First, is a modified resolution method, used in theorem proving, such that some linguistic considerations are introduced in unification of two atoms. Some recent methods of transforming texts in logic forms are used. Second, is based on semantic relations in text, as presented in WordNet. Both methods provide comparable results.

1. Introduction

The recognition of textual inference is one of the most complex task in Natural Language Understanding. Thus, a very important problem in some computational linguistic applications (as Question Answering, summarization, segmentation of discourse, coherence and cohesion of a discourse and others) is to establish if a sentence *follows* from a text. That means in many applications it is important to establish if some sentences which are not existing in text are logicaly implied (can be inferred) by this text. The importance of text inference in computational linguistic is proved by the fact that in TREC (Text REtrieval Conference) conference (http://trec.nist.gov/) and in RTE conference (Recognizing Textual Entailment, http:// www.pascal-network.org/ Challenges/ RTE/) a permanent task is to establish the textual entailment relation. The RTE contest data set includes 1367 English T, H pairs (567 for training stage in learning methods and 800 for test) . Here the task is to determine if the meaning of one text (the entailed hypothesis, H) can be inferred from the meaning of the other text (the entailing text, T).

On the other hand is well known that a linguistic text can be represented by a set of logical formulas, called logic forms. Various method were given for associating a logical formula with a text: [5, 12, 15, 2]. From logical point of view, if each sentence is represented as a formula, proving a textual inference consists

Received by the editors: September 1, 2006.

2000 *Mathematics Subject Classification.* 68T27, 68T30.

1998 *CR Categories and Descriptors.* I.2.3 Deduction and Theorem Proving, I.2.7 Natural Language Processing.

in showing that a logical formula is deducible from a set of others formulas. The problem is a classical (semidecidable) problem. In last years, when text mining is very important in many AI applications, text inference from both point of view, theorem proving and from linguistics perspective, is a very active field of research. In [13] is presented a system participating in the RTE competition, using some world knowledge axioms and a theorem proving tool. The logic method proposed by us in this paper suppose the modification of classical theorem proving task such that it contains a lexical-chain component.

Let us denote entailment relation between a text T and a sentence or a group of sentences H as $T \Rightarrow H$.

In this paper we propose two methods to solve the problem of establishing if $T \Rightarrow H$: first is obtained from the classical resolution refutation method, completing the unification of two atoms with some linguistic considerations (Lexical Resolution Method or LRM). Our method differs of [11] by the fact that it does not need learning stage and it does not need a graph representation and evaluation. The weight (cost) of a deduction is obtained only from the weights (costs) of each resolution steps. At his turn, the cost of a step of resolution is obtained by similarity considerations using some linguistic tools as WordNet [4] and Word::Similarity [8]. No background knowledge [2] is needed.

The second method is based on lexical chains (paths) for entailment spanning the text T and the text H (Lexical-chains Based Method or LBM). A system of rules for construction of lexical rules corresponding to entailment is established. We claim that LRM and LBM produce similar results.

In section 2 we will define our modified unification of two atoms method, our resolution rule and lexical resolution method (LRM).

In section 3 we will describe LBM method and we will propose another definition for text inference based on the cohesion of texts.

2. Text inference as theorem proving.

Consider a knowledge base formed by a set of natural language sentences, K. Let define a set of inferences rules which is sound, in the sense that it derive true new sentences when the initial sentences in K are true. It is a long debate about formalisms to represent knowledge such that above desiderata be fulfilled [15]. We will use here the method proposed by [12] of obtaining logical forms (in fact, logical formulas) from sentences expressed in natural language. In this method each *open* word in a sentence (that means noun, verb, adjective, adverb) is transformed in a logic predicate (atom). We consider, additionally, that the constants are denoted by the names of words they represent (they are real lexical units). For these atoms we propose a new algorithm for unification which modifies the classical Robinson unification algorithm by adding some lexical relaxations.

The semantic information is used in the way we define unification between two atoms, as described in the following section.

2.1. **Unification lexical method for two atoms.** Unification lexical method of two atoms supposes that we have a lexical knowledge base where the similarity between two words is quantified. Such a lexical knowledge base is WordNet [4], a lexical resource which, from its construction in 1998 at Princeton University, is largely used in many linguistic applications. Moreover, some connected resources are constructed (also free) which make use of WordNet easier. For example, Word::similarity is an on-line interface which calculates the similarity between two words using some different similarity measures, all these starting from WordNet facilities [9, 10] It offers the possibility to calculate similarity between two words, two words annotated with POS, or even two words annotated with POS and sense (in WordNet notation). Measures used to calculate similarity could be nine, the most well known are Path lenght, Leacok and Chodorow, Wu and Palmer and Resnik [4]. Of course, a maximal similarity is between words belonging to the same synset (concept).

In the following algorithm we consider that each word of a natural language sentence is transformed in atom as in [12]. See our section 2.3. The classical unification of atoms is replaced by *lexical unification*, which depends on the similarity in the dictionary WordNet. In the following algorithm we consider that $sim(p, p')$ between two words p, p' is that obtained by the Word::similarity interface.

INPUT: Two atoms $a = p(t_1, ..., t_n)$ and $a' = p'(t'_1, ..., t'_m)$, $n \leq m$, threshold τ, threshold for a step τ' . The names p and p' are also words in a lexical knowledge base.

OUTPUT: Decision: The atoms are lexical unifiable with a calculated score W and the unificator is σ, OR they are not unifiable (the score W of unification is less than τ). The steps of the algorithm are:

Step 1. σ = empty substitution, W=0.

Step 2. If $p \equiv p'$ (similarity is maximal) or $sim(p, p') \geq \tau'$
then $W := W + sim(p, p')$; go to Step 3
else Print : " a and a' are not lexical unifiable"; STOP

Step 3. If (for each $t_i, i = 1, ..., n$ exists t'_j in $\{t'_1, ..., t'_m\}$ such that t_i and t'_j are lexical unifiable and the composition of all unificators is σ' OR for each $t'_j, i = 1, ..., m$ exists t_i in $\{t_1, ..., t_n\}$ such that t_i and t'_j are lexical unifiable) , the composition of all unificators is σ', the score is greater than threshold τ
then

Print: " a and a' are lexical unifiable and $\sigma := \sigma$ *composed with* σ'"
else
Print: " a and a' are not lexical unifiable"
STOP

Let us observe that the two terms t_i and t'_j are unifiable in the following two cases.

1. First one refers to the regular cases in FOPC:

- terms are equal constants;
- one is a variable, the other is a constant;
- both are variables.

2. In the second case, if t_i and t'_j are two different constants, as they are words in KB, then they are unifiable if $sim(t_i, t'_j) \geq \tau'$. In the method of obtaining logic form, on which we are based, the arguments of predicate are only variable or constants.

3. Additionally, the similarity $sim(p, p')$ is maximal when p, p' are from the same synset in Wordnet.

The similarity between two words is used to calculate a score for unifiability of two atoms. The test in this case is that the score is larger than a threshold τ. The "assumption cost model" presented in [6] uses a similarity measure for some dependency graphs matching. The difference with our method is that they calculate all unificators and choose the best one (which minimizes a given cost). For the resolution method, we need to obtain the empty clause once. The "cost" of resolution is restricted to be low, while the condition of step threshold is applied.

2.2. **Modified resolution or lexical resolution method.** The modified resolution, called also *lexical resolution method*, LRM, consists in considering of lexical unification of two atoms as replacing regular unification:

Definition

Two (disjunctive) clauses c_i and c_j provide by *lexical resolution* the (disjunctive) clause c_k with the weight τ, written as

$$c_i, c_j \models_{lexical\,resolution} c_k \quad or \quad , shortly, \; c_i, c_j \models_{lr} c_k$$

if $c_i = l \vee c_i', c_j = \neg l' \vee c_j'$, l and l' are lexical unifiable with the weight τ and the unificator σ. The resulting clause is $c_k = \sigma(c_i') \vee \sigma(c_j')$.

Remark: by disjunctive clause we mean a disjunction of literals (negated or not negated atoms).

The following theorem is a translation of Robinson's theorem of resolution method:

Theorem

A set of disjunctive clauses C (obtained from formulas associated to sentences of a text) is contradictory if the empty clause [] is obtained from the set of formulas C by the modified resolution:

$$C \models_{lr}^{*} []$$

Definition

A set of disjunctive clauses C obtained from formulas associated to sentences of a text is contradictory with the weight τ if the empty clause [] is obtained from the set of formulas C by the modified resolution, and the sum of all steps of resolution is τ.

Definition

A set C of clauses which are proved contradictory when modified resolution is used will be denoted as *lexical contradictory*.

Let us resume the steps of demonstrating by lexical resolution method that a text T entails the sentence H with the weight τ, property denoted by $T \Rightarrow_{LRM,\tau} H$:

- Translate T in a set of logical formulas T' and H in H' (as in the following subsection).
- Consider the set of formulas $T' \cup neg(H')$, where by $neg(H')$ we mean the logical negation of formula H'
- Find the set C of disjunctive clauses of the set of formulas T' and $neg(H')$
- Verify if the set C is lexical contradictory with the weight τ. In this case

$$T \Rightarrow_{LRM,\tau} H$$

2.3. **Logical form derivation from sentences.** We will use the method established by [12] which is applied to texts which are part of speech tagged and syntactic analyzed.

The method is the following:

- A predicate is generated for every noun, verb, adjective and adverb (possibly even for prepositions and conjunctions). The name of a predicate is obtained from the morpheme of word.

- If the word is a noun, then the corresponding predicate will have as argument a variable, as individual object. Ex: *person(x2).*
- If the word is a verb, then the corresponding predicate will have as first argument an argument for the event (or action denoted by the verb). Moreover, if the verb is intransitive it will have as arguments two variables: one for the event and one for the subject argument. If the verb is transitive it will have as arguments three variables: one for the event, one for the subject and one for the direct complement. If the verb is ditransitive it will have as arguments four variables: two for the event and the subject and two for the direct complement and the indirect complement.
- The arguments of verb predicates are always in the order: event, subject, direct object, indirect object (the condition is not necessary for modified unification).
- If the word is an adjective (adverb) it will introduce a predicate with the same argument as the predicate introduced for modified noun (verb).

  Example: *man-made object* is translated as: *object(x1) AND man-made(x1)*
- If the word is a preposition or a conjunction it will introduce a predicate with the same argument as the modified word.

Some transformation rules that create predicates and assign them arguments are presented in [12]. These are obtained from the set of rules of the syntactic analyzer. For example, the rule for introduction of noun predicate is $ART\ NOUN \longrightarrow noun(x_1)$. The rule for introduction of adverb predicate is: $VERB\ ADVERB \longrightarrow verb(e_1, x_1, x_2)\ AND\ adverb(e_1)$.

Let us consider the following example from [13]:

T: John and his son, George, emigrated with Mike, John's uncle, to US in 1969
H: George and his relative, Mike, came to America

The logical form obtained for T is:

$$John(x_1) \wedge son(x_2) \wedge George(x_2) \wedge emigrated(e_1) \wedge Agent(x_1, e_1)$$
$$\wedge Agent(x_2, e_1) \wedge Mike(x_3) \wedge uncle(x_1, x_3) \wedge Location(e_1, x_4)$$
$$\wedge US(x_4) \wedge Time(e_1, x_5) \wedge 1969(x_5)$$

The logical form obtained for H is:

$$George(x_1) \wedge relative(x_2) \wedge Mike(x_2) \wedge came(e_1) \wedge Agent(x_1, e_1)$$
$$\wedge Agent(x_2, e_1) \wedge America(x_3) \wedge Location(e_1, x_3)$$

Applying the unification lexical method for two atoms and modified resolution for the obtained disjunctive clauses, we obtain empty clause, as follows.

First, the set of clauses for $neg(H))$ is formed by only one disjunctive clause:

$$\neg George(x_1) \vee \neg relative(x_2) \vee \neg Mike(x_2) \vee \neg came(e_1) \vee \neg Agent(x_1, e_1)$$
$$\vee \neg Agent(x_2, e_1) \vee \neg America(x_3) \vee \neg Location(e_1, x_2)$$

Then, if we apply modified unification between the following pairs of atoms, the empty clause is obtained:

$relative(x_2), uncle(x_1, x_3)$
$America(x_3), US(x_4)$,
$emigrated(e_1), came(e_1)$.

The similarities for the pair $relative, uncle$, for the pair $America, US$ and for the pair $emigrated, came$ are calculated with Word::similarity. So $T \Rightarrow_{LRM,\tau} H$ where the weight τ is the sum of these similarities.

Let us remark that in [13] the result is obtained using additionally 6 axioms.

3. Entailment on linguistic bases

In this section we will introduce another definition for entailment between a text T and a sentence H. This definition is based on the concept of lexical paths and on the semantical relations presented on WordNet.

In the huge knowledge base which is WordNet there are many semantic relations which are defined between synsets of nouns, verbs, adverbs and of adjectives. Synsets in WordNet (or concepts) are set of words which are:

a) with the same POS and

b) are similar as meaning (or synonyms).

The most well known semantical relation is the relation *IS-A* between synsets of nouns (or of verbs). The relations *ENTAIL* and *CAUSE-TO* defined only between synsets of verbs, are the most suited for purposes of entailment study.

We will define a *lexical path for entailment* between two words w_1 and w_2, denoted by $LPE(w_1, w_2)$, a path of the form:

$$LPE(w_1, w_2) = c_1 r_1 c_2 r_2 ..... r_{k-1} c_k$$

where w_1 is from the synset c_1, w_2 is from the synset c_k and each relation r_j is a semantical WordNet relation of the form *IS-A* or *ENTAIL* or *CAUSE-TO* between synsets c_j and c_{j+1}. A *lexical path for entailment*, $LPE(w_1, w_2)$, can be described as a regular expression of the form:

$c_1 r_1 c_2 r_2 ..... r_{k-1} c_k \in ((< concept > (IS - A))^* (< concept > (ENTAIL))^* \mid ((< concept > (IS - A))^* (< concept > (CAUSE - TO)^*)^* < concept >$

The relations *IS-A*, *ENTAIL* and *CAUSE-TO* are transitive and no simetric. Thus the paths $LPE(w_1, w_2)$ and all the concepts defined using them have an orientation from w_1 to w_2.

Definition

$T \Rightarrow_{LPE,\tau} H$ if card$(\{LPE(w_1, w_2) \mid w_1 \in T,\ w_2 \in H\})$ is greater than a given threshold τ.

A method to construct a path $LPE(w_1, w_2)$ is to apply the following rules:

- From $c_1\ IS - A\ c_2$ and $c_2\ IS - A\ c_3$ it results $c_1\ IS - A\ c_3$
- From $c_1\ IS - A\ c_2$ and $c_2\ ENTAIL\ c_3$ it results $c_1\ ENTAIL\ c_3$
- From $c_1\ ENTAIL\ c_2$ and $c_2\ IS - A\ c_3$ it results $c_1\ ENTAIL\ c_3$
- From $c_1\ ENTAIL\ c_2$ and $c_2\ ENTAIL\ c_3$ it results $c_1\ ENTAIL\ c_3$
- From $c_1\ IS{-}A\ c_2$ and $c_2\ CAUSE{-}TO\ c_3$ it results $c_1\ CAUSE{-}TO\ c_3$
- From $c_1\ CAUSE{-}TO\ c_2$ and $c_2\ IS{-}A\ c_3$ it results $c_1\ CAUSE{-}TO\ c_3$
- From $c_1\ CAUSE{-}TO\ c_2$ and $c_2\ CAUSE{-}TO\ c_3$ it results $c_1\ CAUSE{-}TO\ c_3$
- From $c_1\ CAUSE{-}TO\ c_2$ and $c_2\ ENTAIL\ c_3$ it results $c_1\ ENTAIL\ c_3$
- From $c_1\ ENTAIL\ c_2$ and $c_2\ CAUSE{-}TO\ c_3$ it results $c_1\ ENTAIL\ c_3$

We claim that the following theorem holds:

Theorem

For each given threshold τ there exists a threshold τ' such that the relation $T \Rightarrow_{LPE,\tau} H$ holds iff $T \Rightarrow_{LRM,\tau'} H$ holds.

Also, we can introduce another frame for text inference which is very promising to use: the coherence of a text.

Let define a lexical path $LP(w_1, w_2)$ as a path

$$LP(w_1, w_2) = c_1 r_1 c_2 r_2 ..... r_{k-1} c_k$$

were all semantical relations in WordNet are permitted as r_i [3] and c_i are synsets. The semantical relations in WordNet are:

- hypernymy and his reverse hyponymy,
- meronymy and his reverse holonymy,
- entailment, cause-to and reverse of they,
- antonimy.

Definition

The coherence $coh(TE)$ of a text TE is equal to the number of lexical paths which link two different words from text TE.

In the case of entailment of H from T, the coherence of text $T + H$ (the text H and the text T considered as a single text) is larger than the sum of coherences of the separate texts T, H. In other words, we claim that the following theorem holds:

Theorem
$T \Rightarrow H$ iff $coh(T) + coh(H) \leq coh(T + H)$.

4. Conclusions and further work

In this paper we presented two methods for recognizing textual inference: one is from the logic resolution area, using a modified unification algorithm, the second is a pure semantic lexical method and uses the big facilities offered by the huge semantical dictionary WordNet. We consider that the meaning of these methods has common roots: the similarity between two atoms in unification algorithm and the lexical path for entailment are calculated considering semantical relations which exist between concepts (synsets) in WordNet. A study of the relation between τ, τ' is in our attention.

The combined methods in Artificial Intelligence between approaches so different, as Logic and Linguistics, are very largely developed in the last time. The present paper belongs to this category of combined methods.

References

[1] J.Allen: "Natural language understanding", Benjamin/Cummings Publ., 2nd ed., 1995.
[2] J.Bos, K. Markert: "Recognising Textual Entailment with logical inference", Proceedings of HLT/EMNLP. Vancouver, October 2005, pages 628-635.
[3] G. Miller: "WordNet: a lexical database for english", Communications of the ACM, 38(11), pages 39-41 1995
[4] ed. C. Fellbaum: "WordNet: an electronic lexical database", MIT Press, 1998.
[5] S. Harabagiu, D.Moldovan: "A parallel system for Textual Inference", IEEE Transactions parallel and distributed systems, vol 10, no 11, nov 1999 pages 254-270.
[6] A. Haghighi, A. Ng, C. Manning: "Robust textual Inference via graph matching", Proceedings of HLT/EMNLP. Vancouver, October 2005, pages 387-394.
[7] G. Morrill: "Type LogicalGrammar.Categorial Logic of Signs", Kluwer Academic Publishers, 1994.
[8] T. Pedersen, S. Patwardhan, J. Michelizzi: "WordNet::Similarity - Measuring the Relatedness of Concepts", Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI-04), July 25-29, 2004, San Jose, CA (Intelligent Systems Demonstration)
[9] T. Pedersen, A. Kulkarni: "Identifying Similar Words and Contexts in Natural Language with SenseClusters", Proceedings of the Twentieth National Conference on Artificial Intelligence, Pittsburgh, 2005. (Intelligent Systems Demonstration)

[10] http://www.d.umn.edu/ tpederse/similarity.html.
[11] R. Raina, A. Ng, C. Manning: "Robust textual inference via learning and abductive reasoning", AAAI, 2005, http:www.aaai.org.
[12] V. Rus: "Logic form transformation for WordNet glosses and its applications". PhD Thesis, Southern Methodist University, CS and Engineering Department, March 2001.
[13] M.Tatu, D. Moldovan: "A semantic approach to recognizing Textual Entailment", Proceedings of HLT/EMNLP, Vancouver, October 2005, pg 371-378.
[14] D. Tatar: "Unification Grammars in Natural Language Processing", in "Recent topics in mathematical and computational linguistic", ed. C. Martin-Vide, G. Paun, Editura Academiei, 2000, pg 289-300
[15] A. Thayse (editor): "From natural language processing to logic for expert systems", John Wiley and Sons, 1990.

E-mail address: dtatar@cs.ubbcluj.ro

E-mail address: mfrentiu@cs.ubbcluj.ro

Department of Computer Science, Babes-Bolyai University, Cluj-Napoca, Romania

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LI**, Number 2, 2006

DYNAMIC PROGRAMMING AND d-GRAPHS

KÁTAI ZOLTÁN

Abstract. In this paper we are going to introduce a special graph, which we have called d-graph, in order to provide a special tool for such an optimum problem's analysis which breaks down into two or more subproblems by every decision.

1. Introduction

The dynamic programming as a method for resolution of optimizing problems was worked out by Richard Bellman. His first book about dynamic programming was published in 1957 [1]. Since then until his death in 1982 he wrote several books and articles in this area. In 1962 Bellman together with Dreyfus published the book Applied Dynamic Programming [2]. In this publication they drew attention to the fact that dynamic programming can be formulated as a graph search problem. Later this subject was largely analyzed in some papers. For example Georgescu and Ionescu introduced the PD-tree notion [3]. In this paper we are going to introduce a special graph, which we have called d-graph (from division-graph), in order to provide a special tool for such an optimum problem's analysis which breaks down into two or more subproblems by every decision (see later).

2. Optimizing problems

The efficient solving of numerous programming problems implies their optimal breaking down into subproblems. In the present paper we are dealing with such optimizing problems where the following conditions are true:

- There is a target function which has to be optimized.
- The optimizing of the target function implies to break down the problem into subproblems.
- This involves a sequence of decisions.

Received by the editors: May 20, 2006.

2000 *Mathematics Subject Classification.* D.1.0 [Programming Techniques]: General, G.22. [Discrete Mathematics]: Graph Theory - Graph Algorithms, Path and Circuit Problems, Trees.

1998 *CR Categories and Descriptors.* code [**Topic**]: Subtopic – *Detail*; code [**Topic**]: Subtopic – *Detail* .

- Concerning the division into subproblems, with each decision (cut) the problem is reduced to one (I. type optimizing problems) similar, but smaller size subproblem, or breaks into two or more (II. type optimizing problems) similar, but smaller size subproblems.
- The target function is defined on the set of the problem's subproblems.
- The principle of optimality is valid for the problem, according to which the optimal solution of the problem can be built from the optimal solutions of its subproblems (the optimal value of the target function referring to the problem can be determined from the optimal values referring to the subproblems).
- Out of the different possibilities of breaking down the problem, we consider optimal that one (or that sequence of decisions) which - in accordance with the basic principle of optimality - involves the optimal construction of the solution of the problem.
- We call a subproblem trivial when the value of the target function referring to it is given by the input data of the problem in a trivial way.

Such an optimizing problem is solved efficiently with the so called dynamic programming technique.

Example 1. *Let's calculate the result of the product of matrixes* $A_1 \times A_2 \times \cdots \times A_n$ *(the dimensions of the matrixes are:* $d_0 \times d_1, d_1 \times d_2, \ldots, d_{n-1} \times d_n$*). Due to the associativity of the multiplication, we can perform this in several ways. Let's determine such a parenthesis of the product (its breaking down into subproblems) where the corresponding order of the multiplication of matrixes involves a minimal number of basic multiplications (the target function).*

For example, if

$$n = 4,\ A_1(1 \times 10),\ A_2(10 \times 1),\ A_3(1 \times 10),\ A_4(10 \times 1)$$

The optimal breaking down into subproblems: $(A1 \times A2) \times (A3 \times A4)$, which involves 21 basic multiplications. A worst solution would imply 210 multiplications: $((A_1) \times (A_2 \times A_3)) \times (A_4)$.

The structure of an optimizing problem can be described by a d-graph (division graph) , defined in the followings.

3. d-GRAPHS

Definition 1. *We call the connected weighted digraph* $G_d(V, E, C)$ *a d-graph if the following conditions are fulfilled:*

(1) $V = V_p \cup V_d$ *and* $E = E_p \cup E_d$
(2) V_p *- the set of the p type nodes of the graph (p-nodes).*
$V_p = \{p_1, p_2, \ldots, p_{nr\_p}\}$, nr_p *- number of p-nodes.*
(3) *Exactly one element of the set* V_p *is a source node (f).*

(4) *We assign the set of p type sink nodes of G_d with $S(G_d)$ (nr_s marks the number of sink nodes).*
(5) *V_d - the set of the graph's d type nodes (d-nodes); nr_d - number of d-nodes.*
(6) *All the neighbours of the d-nodes are p type and inversely, all the neighbours of the p-nodes are of d type. Each d-node has exactly one in-neighbour of p type, which we call p-father. The out-neighbours of the p-nodes are called d-sons. Each d-node has at least one p type out-neighbour and we are going to refer to these as p-sons.*
(7) *The d-nodes are identified with two indexes: For example the notation d_{ik} refers to the d-son identified as the k^{th} d-sons of the p-node p_i.*
(8) *E_p - the set of p type arcs of the graph (p-arcs).*
$E_p = \{(p_i, d_{ik})/p_i \in V_p, d_{ik} \in V_d\}$.
(9) *E_d - the set of d type arcs of the graph (d-arcs).*
$E_d = \{(d_{ik}, p_j)/d_{ik} \in V_d, p_j \in V_p, i < j\}$. *We should notice that the p type descendent of any p-node have bigger indexes. So in case of any d-graph the source is the 1 node.*
(10) *The $C : Ep \to R$ function associates a cost to every p-arc. We consider the d-arcs of zero cost.*

Theorem 1. *Every d-graph is acyclic.*

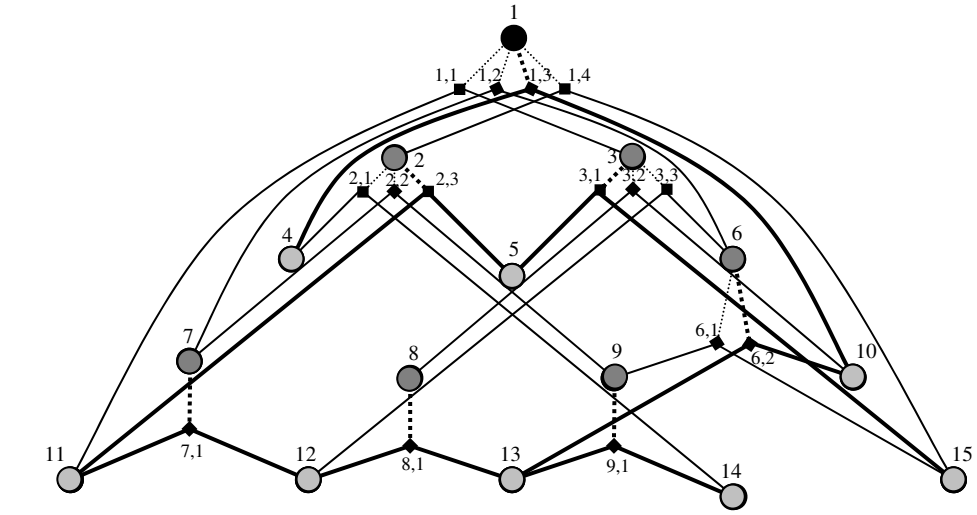
Proof. Let's assume that an oriented cycle exists in one of the d-graphs. According to the sixth item of the definition the p and d type nodes alternate on the cycle. Should the cycle consist of one p-node and one d-node, then the p-node is the p-father and also the p-son of node d in the same time. But this contradicts the ninth item of the definition according to which the p-sons of a d-node have always bigger indexes than its p-father. In case there are at least two nodes of both types, then let's consider p_i and pj two consecutive p-nodes of the cycle. As pi is the ancestor and in the same time the descendent of p_j - also according to the ninth item of the definition - i should be smaller and also bigger than j, which is obviously impossible. So every d-graph is acyclic. □

Conclusion 1. *The p-nodes of any d-graph can be arranged in topological order.*

The following picture presents such a d-graph where each d-node has exactly two p-sons.

Definition 2. *We call the d-graph $g_d(v, e, c)$ the d-subgraph of the d-graph $G_d(V, E, C)$, if*

- *$v_p \subseteq V_p$, $v_d \subseteq V_d$, $e_p \subseteq E_p$, $e_d \subseteq E_d$ and $S(g_d) \subseteq S(G_d)$*
- *$c : e_p \to R$ and $c(x) = C(x)$ for any $x \in e_p$*
- *the set of the d, respectively p type sons of any p, respectively d type node of g_d are similar in the g_d and G_d d-graphs.*



- p-nodes (the black one is source, the light grey one is sink)
- d-nodes
- oriented p-arcs
- oriented d-arcs

(we haven't drawn the directions of the arcs explicitely in order to avoid making the picture too packed. They indicate always from top to bottom.)

FIGURE 1. d-graph

It results from the above definition that every p-node of a d-graph unequivocally identifies the d-subgraph for which the respective node is its source.

Definition 3. *We call d-tree the d-graph where every p-node (except the sinks) has exactly one son. The source of a d-tree is called d-root and its sinks are called d-leaves. The set of leaves of the T_d d-tree are marked with $L(T_d)$.*

Definition 4. *We call the d-tree $t_d(vt, et, c)$ the d-subtree of the d-tree $T_d(Vt, Et, C)$ if*

- *$vt_p \subseteq Vt_p$, $vt_d \subseteq Vt_d$, $et_p \subseteq Et_p$, $et_d \subseteq Et_d$ and $L(t_d) \subseteq L(T_d)$*
- *$c : et_p \to R$ and $c(x) = C(x)$ for any $x \in et_p$*
- *the set of p-sons of any d-node of t_d corresponds in the t_d and T_d d-trees.*

Definition 5. *We call a d-tree $T_d(Vt, Et, c)$ the d-subtree of the d-graph $G_d(V, E, C)$ if*

- $Vt_p \subseteq V_p$, $Vt_d \subseteq V_d$, $Et_p \subseteq E_p$, $Et_d \subseteq E_d$ *and* $L(T_d) \subseteq S(G_d)$
- $c : Et_p \to R$ *and* $c(x) = C(x)$ *for any* $x \in Et_p$
- *the set of p-sons of any d-node of* T_d *corresponds in the d-tree* T_d *and the d-graph* G_d.

If the root of T_d corresponds to the source of G_d, then we can speak about a spanning d-subtree.

Definition 6. *By the costs of a d-tree we mean the total costs of its p-arcs.*

Definition 7. *We call the spanning d-subtree of a d-graph with the lowest costs minimal cost spanning d-subtree.*

Definition 8. *(the basic principle of optimality): We say that a d-graph has an optimal structure if every d-subtree of its optimal (having minimal costs) spanning d-subtree is itself an optimal spanning d-subtree of the d-subgraph determined by its root.*

4. Optimal Structure d-graphs

Let $G_d(V, E, C)$ be a d-graph. In the followings we are going to define a function C of p-arc-costs where every d-graph will be of optimal structure. Before doing that we are defining the node-weighing functions w_p and w_d. We mark the set of d-sons of the p-node p_i with d_son_set(p_i) and the set of p-sons of the d-node d_{ik} with p_son_set(d_{ik}).

The weight-function w_p :

$w_p : V_p \to R$

for every $p_i, i = 1 \dots nr_p$ p-node corresponds

$w_p(p_i) =$ optimum $\{w_d(d_{ik})\}$, if $p_i \notin S(G_d)$

$d_{ik} \in d\_son\_set(p_i)$

$w_p(p_i) = h_r$, if p_i is the r^{th} sink of the d-graph

where $\{h_1, h_2, \dots h_{nr\_s}\} \subset R$ is an input set which characterizes the G_d d-graph

The w_p weight of every p-node (except the sinks) is equal to the w_d weight of its "optimal d-son".

The weight function w_d :

$w_d : V_d \to R$

for every d_{ik} d-node corresponds

$w_d(d_{ik}) = \varphi(\{w_p(p_j)/p_j \in p\_son\_set\ (d_{ik})\})$

The function φ describes mathematically how the w_d weight of a d-node can be calculated from the w_p weights of its p-sons. The function φ also characterizes the G_d d-graph

After having introduced the above weight functions, we define the cost function C^* in the following way:

$$C^* : E_p \to R,\ C^*((p_i, d_{ik})) = |w_p(p_i) - w_d(d_{ik})|$$

Theorem 2. *Every d-graph $G_d(V, E, C^*)$ has optimal structure.*

Proof. As we have chosen the weight of the optimal d-sons as the weight of the p-nodes, every p-node is adjacent to at least one zero cost p-arc. It derives from this that the minimal cost spanning d-subtree and its every d-subtree will have zero costs. As C^*, by its definition, assigns positive costs to the p-arcs, it is natural that every d-subtree of the minimal cost spanning d-subtree will be a minimal cost spanning d-subtree of the d-subgraph which has a corresponding source of its root. □

5. Determination of the optimal spanning d-subtree with the implementation of the basic principle of optimality

Let $G_d(V, E, C^*)$ be an optimal structure d-graph. According to the basic principle of optimality, the optimal spanning d-subtree of any g_d d-subgraph of G_d can be determined from the optimal spanning d-subtrees of the son-d-subgraphs of g_d. Consequently we are going to determine the optimal spanning d-subtrees belonging to the nodes $p_i \in V_p (i = 1 \dots nr\_p)$ in a reversed topological order. This order can be ensured if at the depth-traversing, we deal with the certain nodes at the moment we are leaving them.

We use the arrays $WP[1 \dots nr\_p]$ and $WD[1 \dots nr\_d]$ in order to store the weights of the p, respectively d type nodes of the d-graph G_d. At the beginning we fill up the elements of array WP corresponding to the sinks with their $h_i (i = 1 \dots nr\_s)$ weights, the other elements with the value NIL. For the storage of the optimal spanning d-subtree we take array $ODS[1 \dots nr\_p]$, which stores the optimal d-sons of the p-nodes. We initialize this array with the value NIL. The **initialization** procedure, depending on the nature of the optimum to be calculated, gives a suitable starting value to the array-element $WP[p_i]$ received as a parameter. The function **is_ better** analysis whether the first parameter is better than the second one, according to the nature of the optimum.

optimal_division(p_i)
    initialization(WP[p_i])
    for all $d_{ik} \in d\_son\_set\ (p_i)$ do
        for all $p_j \in p\_son\_$ set (d_{ik}) do
            if $WP[p_j] = NIL$ then optimal_division(p_j)
            endif
        endfor
        $WD[d_{ik}] = \varphi(\{WP[p_j]/p_j \in p\_son\_set\ (d_{ik})\})$
        if is_better($WD[d_{ik}], WP[p_i]$) then
            $WP[p_i] = WD[d_{ik}]$
            $ODS[p_i] = d_{ik}$
        endif
    endfor

end optimal_division

Of course we call the **optimal_division** procedure for the source node, presuming that it is not a sink in the same time. The OSD values of the sinks remain NIL. The following recursive procedure, based on the ODS array prints the p-arcs of the optimal spanning d-subtree in a preorder order.

optimal_tree (p_i)
    write $(p_i, ODS[p_i])$
    for all $p_j \in p\_son\_set\ (ODS[p_i])$ do
        if $ODS[p_j] \notin NIL$ then optimal_tree (p_j)
        endif
    endfor
end optimal_tree

6. The optimizing problems and the d-graphs

A d-graph can be associated to any optimizing problem described in the introduction.

- The p-nodes represent the different subproblems given by the breaking down of the problem. The source represents the original problem, the sinks the trivial ones.
- The numbering of the p-nodes and the acyclicity given by this go hand in hand with the fact that, in the course of the breaking down, we reduce the problem to simpler and simpler subproblems.
- A p-node will have as many d-sons as the number of possibilities in which the subproblem represented by it can be broken down to its subproblems, by the respective decision. These decision possibilities are represented by the p-arcs.
- The d-nodes represent the way the respective subproblem breaks down into its subproblems with the choices given by the different decisions.
- A d-node will have as many p-sons, as the number of subproblems resulted after the disintegration - with the occasion of the decision represented by it - of the subproblem described by its p-father. This breaking down into subproblems is described by the d-arcs.
- If different sequences of decisions taken at the breaking down of a problem lead to the same subproblem, then the respective p-node will have identical p-descendents on different descent branches.
- The d-subgraphs of a d-graph express the way in which the subproblems represented by its sources can be broken down onto further, smaller subproblems.
- A certain subtree of a d-graph describes one of the breaking downs onto subproblems of the subproblem represented by its root. The spanning

subtrees of a d-graph represent the possibilities of breaking down the original problem onto its subproblems.

- The optimal structure of the d-graphs expresses the fact that the optimal solution of the problem is built from the optimal solutions of the subproblems. In other words, the corresponding subsequences of the optimal sequence of the decisions are also optimal.
- The optimal spanning d-subtree represents the optimal breaking down of the problem into subproblems (its every p-arc represents one of the decisions of the optimal sequence of decisions.).
- The wp function is nothing else but the returning of the target function to be optimized to the G_d d-graph.
- $h_1, h_2, \ldots, h_{nr\_s}$ real values are the optimal values referring to the trivial subproblems of the target function, represented by the sinks.
- The nature of the optimum function is directly given by the target function of the problem and is often one of the minimum or maximum functions.
- The function φ is determined by the structure of the problem, the general rule according to which the solution of a subproblem is built from the solutions of its subproblems.

Hereby, an optimizing problem can be regarded as the determination of the weight of the source of a d-graph (the optimal value of the target function concerning the original problem) and of its optimal spanning d-subtree (optimal sequence of decisions, respectively optimal breaking down into subproblems).

We call the procedure **optimal_division**, which implements the basic principle of optimality, dynamic programming.

7. Solving a problem given as an example

Compression: A bit-sequence of n elements is given. We also have m other "shorter" sequences of bits, where there are sequences containing only one bit of 0 respectively of 1, too. Let's replace the first bit-sequences with the minimal number of short bit-sequences.

Example:

Let the first sequence of bits be 01011.

Further the short sequences are: 1:0, 2:1, 3:11, 4:010, 5:101. We can see that the original sequence of bits can be broken down to the given short sequences in several ways:

$$(0)(1)(0)(1)(1), (0)(1)(0)(11), (010)(11), (0)(101)(1), (010)(1)(1)$$

The optimal solution is of course represented by the third version, whose compressed code is 43.

How can the problem be broken down into its subproblems? In so far as the original bit-sequence is not one of the given short sequences, we cut it in two, thus

reducing its optimal compression to the compression of the sub-sequences of bits gained at the left and right side of the cut. We continue this until we get sequences of bits which appear in the given short sequences (trivial subproblems replaceable one short sequence's code).

The general subproblem is represented by the optimal compression of the subsequence $i \dots j$ of the original sequence of bits. These indexes will identify the p-nodes of the d-graph which can be assigned to this problem. The source of the problem given as an example is the p-node 15 (read one-five). The role of the WP array is played by the part of a bidimensional array $a[1 \dots n, 1 \dots n]$ situated on and above its diagonal. This part of the array can be interpreted as the implicit representation of the d-graph of the problem. The p-nodes are represented by the corresponding array-elements and we can consider as their w_p weight the length of the code of the optimal compression. The ij p-node -in so far as it is not a sink (the sequence $i \dots j$ is not part of the given short sequences)- will have a number of $(j - i + 1)$ d-sons, whose p-son-pairs will be the p-node-pairs $(ik, (k+1)j)(k = i \dots j-1)$. So the d-nodes, respectively the p and d type arcs are only implicitly present in this representation of the d-graph. Of course this also implies that we do not use a WD array. This is not necessary, as the weighing of the d-nodes of the d-graphs can be avoided by merging formulas (1) and (2) (the weight of any p-node which is not a sink can be determined from the weight of its direct p-descendents):

$$w_p(p_i) = \underset{d_j \in d\_\text{son\_set}\ (p_i)}{\text{optimum}} \{\varphi(\{w_p(p_k)/p_k \in p\_\text{son\_set}(d_j)\})\},\ \text{if } p_i \notin S(G_d)$$

In the role of the ODS array the part of the bidimensional array situated above the diagonal can be used. The array-element $a[j, i](i < j)$ implicitly represents the optimal d-son of the p-node ij by the storage of the optimal k value belonging to the optimal cut of the sequence of bits $i \dots j$. If the p-node $ij(i < j)$ is a sink, then the element $a[j, i]$ will get the value zero. The p-nodes $ii(i = 1 \dots n)$ are obviously all sinks.

The following picture (see Figure 2.) represents the d-graph of the sample problem, as it is hidden in the array **a** storing the optimal values of the subproblems. We have highlighted the optimal spanning d-subtree of every d-subgraph with the source ij.

With such a representation of the d-graph the traversing of the non-sink p-nodes in a reversed topological order can be achieved by the simple traversing of the array-elements situated above the main diagonal (for example row by row from below upwards left to right). As the optimal code of any sequence of bits is the concatenation of the optimal codes of the subsequences given by its optimal cut, the function φ is a simple additive function. As we are looking for the compression with the shortest code, the function **optimum** will calculate a minimum. The input data is stored by the variables $n, m, b[1 \dots n]$ and $sequence[1 \dots m]$. The

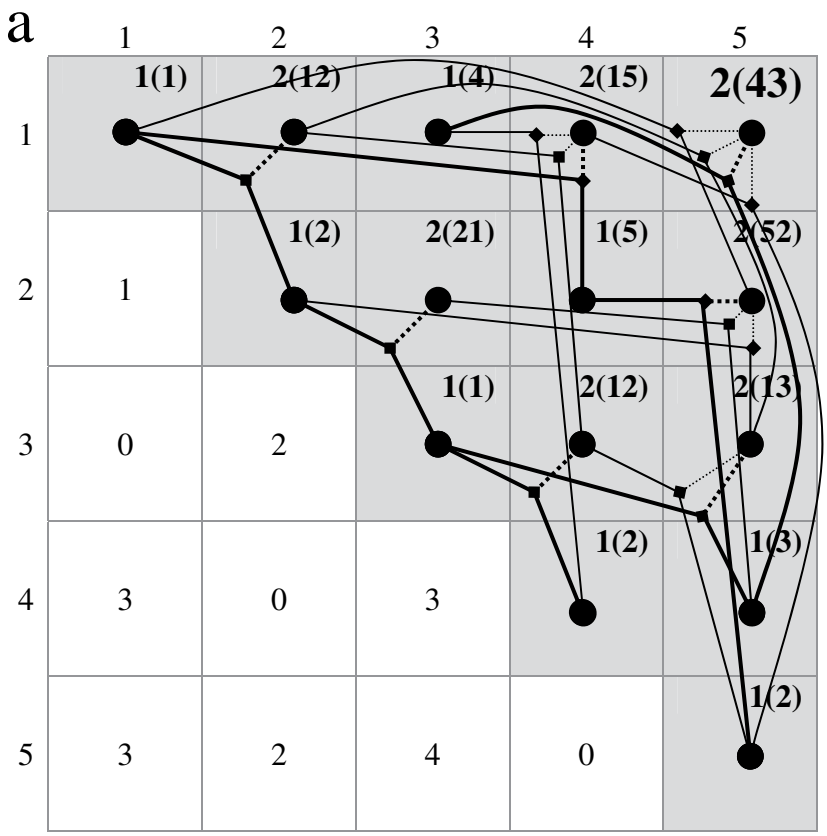


FIGURE 2. The d-graph stored in an implicit way in the array **a**

function **nr_sequence** $(\mathbf{i}, \mathbf{j})$ checks whether the sequence of bits $b[i \dots j]$ is present in the given short sequences. If yes, then it returns its code (its index from the array *sequences*), if no, it returns zero. In parallel with the filling up of the array a we store the optimal codes themselves, too, in an array $\mathrm{COD}[1 \dots n, 1 \dots n]$ (in the above picture we have represented them in brackets). The function **concate(cod1,cod2)** concatenates the codes received as parameters.

```
for i=1,n,1 do
      a[i,i]=1
      COD[i,i]=nr_sequence(i,i)
endfor
```

```
for i=n-1,1,-1 do
    for j=i+1,n,1 do
        cod=nr_sequence(i,j)
        if cod>0 then
            a[i,j]=1
            COD[i,j]=cod
            a[j,i]=0
        else
            a[i,j]=0
            for k=i,j-1,1 do
                if a[i,k]+a[k+1,j]>a[i,j] then
                    a[i,j]=a[i,k]+a[k+1,j]
                    COD[i,j]=concate(COD[i,k],COD[k+1,j])
                    a[j,i]=k
                endif
            endfor
        endif
    endfor
endfor
```

The optimal code of the original sequence of bits gets into the array COD[1,n]. In case we would also like the optimal parenthesis of the bit-sequence, we can obtain this by traversing in depth the optimal spanning d-subtree of the d-graph based on array a.

```
depthfirst(i,j)
    write '('
    if i=j OR a[j,i]=0 then
        for k=i,j,1 do
            write b[k]
        endfor
    else
        depthfirst(i,a[j,i])
        depthfirst(a[j,i]+1,j)
    endif
    write ')'
end depthfirst
```

8. Conclusions

First of all it is interesting to remark that in case of the I. type optimizing problems the attached d-graphs can be reduced to a "normal graphs" (since every d-node has an unique p-son they can be left out from the graph by matching their

p-father directly with their unique p-son). In this special situation the optimal solution will be represented by the optimal root-leaf path of the graph. By introducing the d-graphs, the consistent discussion of several optimizing problems has become possible, and also the theoretical basis of the dynamic strategies related to them. The relation is similar to the one between the greedy algorithm and the theory of matroids.

References

[1] R. Bellman, Dynamic Programming, Princeton University Press, New Jersey, 1957.
[2] R. Bellman, S. Dreyfus, Applied Dynamic Programming, Princeton University Press, New Jersey, 1962.
[3] H. Georgescu, C. Ionescu, The Dynamic Programming Method, a New Approach, STUDIA Universitatis Babes-Bolyai, Cluj, 43, 1999, pp. 23-38.

E-mail address: katai_zoltan@ms.sapientia.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LI**, Number 2, 2006

A STUDY ON DISTANCE METRICS FOR PARTITIONING BASED ASPECT MINING

GRIGORETA SOFIA MOLDOVAN AND GABRIELA ŞERBAN

Abstract. The aim of this paper is to make a study on the influence of distance metrics for partitioning based aspect mining. For this purpose, we comparatively present, from the aspect mining point of view, the results of three algorithms in Aspect Mining, *kAM* ([3]), *HAM* ([6]) and *GAAM* ([5]), for different distance metrics. The evaluation is based on a set of quality measure that we have previously defined in [1] and [2], and a case study is also reported. We introduce three different criteria on which our study is based.

Keywords: aspect mining, distance metrics, clustering.

1. Introduction

1.1. **Aspect Mining.** *Separation of concerns* ([13]) is a very important principle of software engineering that, in its most general form, refers to the ability to identify, encapsulate and manipulate those parts of software that are relevant to a particular concept, goal, or purpose. Some of the benefits of a good separation of concerns are reduced software complexity, improved comprehensibility, limited impact of change, easy evolution and reuse.

Aspect Oriented Programming (AOP) ([10]) provides means to encapsulate concerns which cannot be modularized using traditional programming techniques. These concerns are called *crosscutting concerns*. Logging and exception handling are well known examples of crosscutting concerns. Aspect oriented paradigm offers a powerful technology for supporting the separation of crosscutting concerns. Such a concern is explicitly specified as an *aspect*. Aspects encapsulate the implementation of a crosscutting concern. A special tool, called *weaver*, integrates a number of aspects to obtain the final software system.

In order to apply AOP principles to legacy software systems, it is necessary to analyze the existing implementation to discover the crosscutting concerns and

Received by the editors: November 5, 2006.

2000 *Mathematics Subject Classification.* 68N99, 62H30.

1998 *CR Categories and Descriptors.* D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement –*Restructuring, reverse engineering, and reengineering*; I.5.3 [**Computing Methodologies**]: Pattern Recognition – *Clustering*.

refactor them into aspects. The research on *aspect mining* refers to the identification and analysis of non-localized crosscutting concerns throughout an existing legacy software system ([9]). The goal of aspect mining is to support aspect-oriented refactoring to improve software comprehensibility, reusability and maintainability.

1.2. **Related Work.** In [4] a vector space model based clustering approach in aspect mining is proposed. This approach is improved in [3], by defining a new *k-means* based clustering algorithm in aspect mining (*kAM*).

In [1], a part of a formal model for clustering in aspect mining is introduced and a set of quality measures for evaluating the results of clustering based aspect mining techniques is presented. This model is extended in [2].

A Hierarchical clustering algorithm in Aspect Mining (*HAM*) is introduced in [6]. In [5], the problem of identifying crosscutting concerns is defined as a search problem in a *graph* and *GAAM* algorithm (*Graph Algorithm in Aspect Mining*) is introduced for this purpose.

Each of *kAM*, *HAM* and *GAAM* algorithms make use of distance metrics between multi-dimensional vectors in order to determine the distance (dissimilarity) between the methods from a software system to be mined.

In this paper we study the influence of distance metrics on the results obtained by the above mentioned algorithms. We intend to identify the most suitable distance metric between methods. The comparison of the results is from the aspect mining point of view and is made based on some quality measures that were previously introduced in [1] and [2].

The paper is structured as follows. A theoretical model for the problem of crosscutting concerns identification is given in Section 2. Section 3 presents a vector space model based partitioning approach in aspect mining. The comparative results for different distance metrics, based on some quality measures, is presented in Section 4. Some conclusions and further work are given in Section 5.

2. Background

In [5] the problem of identifying *crosscutting concerns* is defined as a problem of identifying a partition of a software system.

Let $S = \{s_1, s_2, \ldots, s_n\}$ be a software system, where $s_i, 1 \leq i \leq n$, is an element from the system. An *element* can be a statement, a method, a class, a module, etc. We denote by n ($|S|$) the number of elements of the system.

In the following, we will consider a crosscutting concern as a set of elements $C \subset S$, $C = \{c_1, c_2, ..., c_{cn}\}$, elements that implement this concern. Let $CCC = \{C_1, C_2, ..., C_q\}$ be the set of all crosscutting concerns that exist in the system S. The number of crosscutting concerns in the system S is $q = |CCC|$. Let

$NCCC = S - (\bigcup_{i=1}^{q} C_i)$ be the set of elements from the system S, elements that are not used to implement any crosscutting concerns.

Definition 1. ([2]) ***Partition of a system*** S.
The set $\mathcal{K} = \{K_1, K_2, ..., K_p\}$ *is called a* ***partition*** *of the system* S *iff* $1 \leq p \leq n$, $K_i \subseteq S, K_i \neq \emptyset, \forall\ i, 1 \leq i \leq p$, $S = \bigcup_{i=1}^{p} K_i$ *and* $K_i \cap K_j = \emptyset,\ \forall\ i, j, 1 \leq i, j \leq p, i \neq j$.

In the following we will refer to K_i as the i-th *cluster* of $\mathcal{K}$.

In fact, the problem of aspect mining can be viewed as the problem of finding a partition $\mathcal{K}$ of the system S such that $CCC \subset \mathcal{K}$. Definition 2 introduces the notion of **partitioning aspect mining technique**, that is used in this paper.

Definition 2. ([5]) ***Partitioning aspect mining technique.***
Let $\mathcal{T}$ *be an aspect mining technique and* S *a software system to be mined. We say that* $\mathcal{T}$ *is a* ***partitioning aspect mining technique*** *if the result obtained by* $\mathcal{T}$ *is a* ***partition*** (Definition 1) $\mathcal{K}$ *of* S.

We mention that *kAM*, *HAM* and *GAAM* algorithms determine partitions of a software system, but using different approaches, and are used in the partitioning aspect mining techniques introduced in [3], [6] and [5], The first two algorithms use clustering ([11]) approaches and the last algorithm uses a graph based approach.

3. Vector Space Model based Partitioning in Aspect Mining

Let us consider a software system S to be mined.

In approaches [3], [5] and [6], the software system S is composed of a set of methods $m_1, m_2, \ldots, m_n$, so the objects to be grouped (partitioned) are the methods from S . The methods belong to the application classes or are called from the application classes.

Based on the vector space model, each method is considered as an l-dimensional vector: $m_i = (m_{i1}, \ldots, m_{il})$.

Crosscutting concerns in non AO systems have two symptoms: *code scattering* and *code tangling*. *Code scattering* means that the code that implements a crosscutting concern is spread across the system, and *code tangling* means that the code that implements some concern is mixed with code from other (crosscutting) concerns.

We have considered a vector-space model that illustrate only the *scattered code* symptom. Future development will also consider the *code tangling* symptom.

The vector associated with a method m is $\{FIV, B_1\ B_2, ...B_{l-1}\}$, where FIV is the fan-in value ([12]) of m and B_i $(1 \leq i \leq l-1)$ is 1, if the method m is called from a method belonging to the application class AC_i, and 0, otherwise.

As in a vector space model based clustering ([11]), we consider the *distance* between two methods m_i and m_j as a measure of dissimilarity between them.

In our approach we will consider three possible distance metrics between methods:

- *Euclidian Distance.* The distance between m_i and m_j is expressed as:

$$d_E(m_i, m_j) = \sqrt{\sum_{k=1}^{l}(m_{ik} - m_{jk})^2} \tag{1}$$

- *Hamming Distance.* The distance between m_i and m_j is expressed as:

$$d_H(m_i, m_j) = |\{k | 1 \leq k \leq l,\ m_{ik} \neq m_{jk}\}| \tag{2}$$

- *Manhattan Distance.* The distance between m_i and m_j is expressed as:

$$d_M(m_i, m_j) = \sum_{k=1}^{l} |m_{ik} - m_{jk}| \tag{3}$$

4. Experimental Evaluation

In order to evaluate the results of *kAM*, *HAM* and *GAAM* algorithms for different distance metrics, from the aspect mining point of view, we use four quality measure defined in [1] (*DISP*, *DIV*, *PREC* and *PAM*) and two quality measures defined in [2] (*ACTE* and *PANE*). We mention that the last two measures are considered for the case in which the software system consists of a set of methods.

These measures are applied on a case study and the comparative results are reported in Subsection 4.1.

We make the comparison of the obtained results based on three criteria:

(1) **Partitioning criterion**. The degree to which each crosscutting concern is well placed in the partition. For this criterion we use measures *DISP* and *DIV* ([1]).
(2) **Selection criterion**. How well the clusters to be analyzed are chosen. For this criterion we use measures *PREC* and *ACTE* ([1, 2]).
(3) **Ordering criterion**. How relevant is the order in which the clusters are analyzed. For this criterion we use measures *PAM* and *PANE* ([1, 2]).

In order to compare two partitions of a software system S from the above defined criteria, we introduce Definitions 3, 4 and 5. The definitions are based on the properties of the quality measures defined in [1] and [2].

Definition 3. *If $\mathcal{K}_1$ and $\mathcal{K}_2$ are two partitions of the software system S, CCC is the set of crosscutting concerns in S, then $\mathcal{K}_1$ is **better** than $\mathcal{K}_2$ from the **partitioning** criterion point of view iff the following inequalities hold:*

$$DISP(CCC, \mathcal{K}_1) \geq DISP(CCC, \mathcal{K}_2), DIV(CCC, \mathcal{K}_1) \geq DIV(CCC, \mathcal{K}_2).$$

Definition 4. *If $\mathcal{K}_1$ and $\mathcal{K}_2$ are two partitions of the software system S, CCC is the set of crosscutting concerns in S and $\mathcal{T}$ is a partitioning aspect mining technique, then $\mathcal{K}_1$ is **better** than $\mathcal{K}_2$ from the **selection** criterion point of view iff the following inequalities hold:*

$$PREC(CCC, \mathcal{K}_1, \mathcal{T}) \geq PREC(CCC, \mathcal{K}_2, \mathcal{T}),$$

$$ACTE(CCC, \mathcal{K}_1, \mathcal{T}) \geq ACTE(CCC, \mathcal{K}_2, \mathcal{T}).$$

Definition 5. *If $\mathcal{K}_1$ and $\mathcal{K}_2$ are two partitions of the software system S, CCC is the set of crosscutting concerns in S and $\mathcal{T}$ is a partitioning aspect mining technique, then $\mathcal{K}_1$ is **better** than $\mathcal{K}_2$ from the **ordering** criterion point of view iff the following inequalities hold:*

$$PAM(CCC, \mathcal{K}_1) \leq PAM(CCC, \mathcal{K}_2), PANE(CCC, \mathcal{K}_1) \leq PANE(CCC, \mathcal{K}_2).$$

Remark 1. *If at least one of the inequalities from Definitions 3, 4 and 5 is not satisfied, we cannot decide which of the partitions $\mathcal{K}_1$ or $\mathcal{K}_2$ is better related to its corresponding criterion.*

4.1. **Results.** In order to evaluate the results of the algorithms presented in [3], [6] and [5] we have considered as case study Carla Laffra's implementation of Dijkstra algorithm ([8]).

This case study is a Java applet that implements Dijkstra algorithm in order to determine the shortest path in a graph. It was developed by Carla Laffra and consists of **6** classes and **153** methods.

In this subsection we comparatively present the results obtained after applying the selected algorithms, for the vector space model and distance metrics defined in Section 3, with respect to the quality measures, for the case study presented above.

We mention that in the *analysis* step ([3], [5], [6]) for identifying the crosscutting concerns from a software system only a part of the obtained clusters are analyzed, i.e., the clusters whose distance from 0_l point is greater than a given threshold, α. Because α depends on the distance metric used for partitioning, in Table 1 we give the values for this threshold and in Table 2 we present the comparative results.

The reasons for choosing the values from Table 1 for the threshold α, from the aspect mining point of view, are as follows:

- For the *Euclidian* distance metric we analyze only the methods that are called from at least two different contexts.

Distance metric	α
d_E	2
d_H	3
d_M	3

TABLE 1. The values of the threshold α for the distance metrics.

- For the *Manhattan* distance metric the inequality (4) holds:

$$d_M(m, 0_l) \geq \frac{3}{2}\dot{d}_E(m, 0_l), \forall m \in S \tag{4}$$

- For the *Hamming* distance metric we analyze only the methods that are called from at least two different classes.

Algorithm	Distance metric	DISP	DIV	PAM	PREC	ACTE	PANE
kAM	d_E	0.75	0.8854	0.1486	1	0.6667	0.2009
kAM	d_H	0.75	0.8910	0.3316	0.25	0.25	0.6846
kAM	d_M	0.75	0.8854	0.1633	1	0.66	0.2058
HAM	d_E	0.75	0.8981	0.4199	0.5	0.5	0.4493
HAM	d_H	0.75	0.8988	0.3545	0.25	0.25	0.6944
HAM	d_M	0.75	0.8981	0.4183	0.5	0.5	0.4673
GAAM	d_E	0.75	0.8333	0.4133	0.5	0.5	0.4493
GAAM	d_H	0.75	0.8583	0.6111	0.25	0.25	0.6601
GAAM	d_M	0.75	0.8333	0.4117	0.5	0.5	0.4477

TABLE 2. The values of the quality measures for LaffraGraph case study.

From Table 2, based on Definitions 3, 4 and 5 we observe the following:

- For *kAM* algorithm we conclude that:
  - **Partitioning criterion**. Better results are obtained for *Hamming distance*, followed by *Euclidian distance* and *Manhattan distance* (the last two metrics provide the same results).
  - **Selection criterion**. Better results are obtained for *Euclidian distance* and *Manhattan distance*, followed by *Hamming distance* (the first two metrics provide the same results).
  - **Ordering criterion**. Better results are obtained for *Euclidian distance*, followed by *Manhattan distance*, and then by *Hamming distance*.
- For *HAM* algorithm we conclude that:
  - **Partitioning criterion**. Better results are obtained for *Hamming distance*, followed by *Euclidian distance* and *Manhattan distance* (the last two metrics provide the same results).

  - **Selection criterion**. Better results are obtained for *Euclidian distance* and *Manhattan distance*, followed by *Hamming distance* (the first two metrics provide the same results).
  - **Ordering criterion**. We cannot decide which of the distance metrics provide better results, because not all the inequalities from Definition 5 are simultaneously satisfied. This lack of decidability, in our opinion, may be determined by the vector space model and the method chosen for ordering the clusters.
- For *GAAM* algorithm we conclude that:
  - **Partitioning criterion**. Better results are obtained for *Hamming distance*, followed by *Euclidian distance* and *Manhattan distance* (the last two metrics provide the same results).
  - **Selection criterion**. Better results are obtained for *Euclidian distance* and *Manhattan distance*, followed by *Hamming distance* (the first two metrics provide the same results).
  - **Ordering criterion**. Better results are obtained for *Manhattan distance*, followed by *Euclidian distance*, and then by *Hamming distance*.

We observe that, for the *partitioning* and *selection* criteria, the classification of distance metrics is the same for all algorithms. But, for the *ordering* criterion a general classification cannot be decided.

In order to solve the lack of decidability for the *ordering* criterion, we intend:

- To improve the vector space model by taking into account the *code tangling* symptom.
- To find a better order for the clusters analysis.

5. Conclusions and Future Work

In this paper we have comparatively present the results obtained by three algorithms in aspect mining (*kAM* ([3]), *HAM* ([6]) and *GAAM* ([5])) for different distance metrics.

In order to evaluate the obtained results from the aspect mining point of view, we have used a set of quality measures defined in [1] and [2].

We have also given definitions in order to compare two partitions from the aspect mining point of view, based on three different criteria. Based on these definitions, we have comparatively analyzed the influence of distance metrics on the results obtained by the selected partitioning aspect mining techniques.

Further work can be done in the following directions:

- To improve the vector space model by also considering the *code tangling* symptom.
- To identify the most suitable values for the threshold α and the proper order for analyzing the clusters in a partition.

- To identify new distance metrics suitable in aspect mining, considering weighted attributes, too.

References

[1] Moldovan, G.S., Serban, G., "Quality Measures for Evaluating the Results of Clustering Based Aspect Mining Techniques", In: Proceedings of Towards Evaluation of Aspect Mining(TEAM), ECOOP, 2006, pp. 13–16.

[2] Moldovan, G.S, Serban, G., "Clustering Based Aspect Mining Formalized", WSEAS Transactions on Computers, Issue 2, Vol.6, 2007, pp. 199–206

[3] Serban, G., Moldovan, G.S., "A new k-means based clustering algorithm in aspect mining", In: 8th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC'06), 2006, pp. 60–64.

[4] Moldovan, G.S., Serban, G., "Aspect Mining using a Vector-Space Model Based Clustering Approach", In: Proceedings of Linking Aspect Technology and Evolution (LATE) Workshop, 2006, pp. 36-40.

[5] Serban, G., Moldovan, G.S., "A graph algorithm for identification of croscutting concerns", Studia Universitatis "Babes-Bolyai", Informatica, LI(2), 2006, to appear.

[6] Serban, G., Moldovan, G.S., "A New Hierarchical Agglomerative Clustering Algorithm in Aspect Mining", The 24th International Symposium on Theoretical Aspects of Computer Science, Aachen, Germany, 2007, submitted.

[7] Marin, M., van, A., Deursen, Moonen, L., "Identifying Aspects Using Fan-in Analysis", In: Proceedings of the 11th Working Conference on Reverse Engineering (WCRE2004), IEEE Computer Society, 2004, pp. 132–141.

[8] Laffra, C.: "Dijkstra's Shortest Path Algorithm", http://carbon.cudenver.edu/∼hgreenbe /courses/dijkstra/DijkstraApplet.html, 1996.

[9] Magiel Bruntink, Arie van Deursen, Remco van Engelen, and Tom Tourwe. On the Use of Clone Detection for Identifying Crosscutting Concern Code. *IEEE Transactions on Software Engineering*, 31(10), 2005, pp. 804–818.

[10] Kiczales, G., Lamping, J., Menhdhekar, A., Maeda, C., Lopes, C., Loingtier, J.M., Irwin, J., "Aspect-Oriented Programming", In *Proceedings European Conference on Object-Oriented Programming*, volume 1241, Springer-Verlag, 1997, pp. 220–242.

[11] Jain, A., Murty, M.N., Flynn, P., "Data clustering: A review", *ACM Computing Surveys*, 31(3), 1999, pp. 264–323.

[12] Marin, M., van Deursen, A., Moonen, L., "Identifying Aspects Using Fan-in Analysis", In *Proceedings of the 11th Working Conference on Reverse Engineering (WCRE2004)*, IEEE Computer Society, 2004, pp. 132–141.

[13] David. L. Parnas. On The Criteria To Be Used in Decomposing Systems Into Modules. *Communications of the ACM*, 15(12), 1972, pp. 1053–1058.

Babeş-Bolyai University, Faculty of Mathematics and Computer Science, Cluj-Napoca, Romania

E-mail address: grigo@cs.ubbcluj.ro

Babeş-Bolyai University, Faculty of Mathematics and Computer Science, Cluj-Napoca, Romania

E-mail address: gabis@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LI**, Number 2, 2006

ALGORITHME DE CONSTRUCTION D'UN GRAPHE PERT À PARTIR D'UN GRAPHE DES POTENTIELS DONNE

NASSER EDDINE MOUHOUB, HOCINE BELOUADAH, AND ABDELHAK BOUBETRA

Résumé : On présente dans ce papier, dans les problèmes d'ordonnancement de projet, un algorithme original de construction d'un graphe PERT à partir d'un graphe des potentiels donné en utilisant les notions de graphes adjoints de graphes.

Mots clés : Graphe adjoint de graphe, ordonnancement, méthode des potentiels, méthode PERT, méthode de construction de graphes.

1 **Introduction** [KEY 61] [CRA 97] [DAL 01] [ESQ 99]

Les problèmes d'ordonnancement sont définis par la donnée d'un certain nombre d'opérations (les tâches) et des contraintes de succession entre ces tâches, ainsi que les durées de ces tâches.
Plusieurs méthodes de modélisation existent actuellement. On peut citer entre autres: le diagramme de Gantt , la méthode des potentiels et la méthode PERT.Ces deux dernières utilisent comme moyen de modélisation la théorie des graphes ; et plus particulièrement le réseau.
Dans le graphe des potentiels (appelé également graphe potentiels tâches), les tâches sont symbolisées par des sommets auxquels on donne le même code, 2 sommets u et v sont reliés par un arc de u vers v si et seulement si la tâche u précède la tâche v.

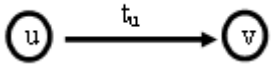


FIGURE 1. La tâche u, de durée t (u), précède la tâche v

Dans le graphe PERT alors, appelé graphe potentiels-étapes, une tâche est représentée par un arc auquel on donne le même code, deux arcs u et v tels que T(u) = I(v) si et seulement si la tâche u précède la tâche v. Les extrémités initiale et terminale d'un arc sont respectivement les évènements début de tâche

Received by the editors: March 22, 2006.

et fin de tâche, elles sont appelées étape. Les durées sont portées sur les arcs correspondants.

FIGURE 2. La tâche u précède la tâche v dans le graphe PERT.

Si le graphe de la méthode des potentiels et celui de la méthode PERT sont très proches, ce n'est pas toujours le cas. La construction du graphe PERT pose des problèmes qui amènent à ajouter des arcs fictifs (virtuels ou artificiels) qui ne correspondent à aucune tâche [ROY 70]. L'introduction des tâches fictives permet de solutionner certaines situations et de lever des ambigutés. Elles ne mettent en jeu aucun moyen matériel ou financier.

2 Liens entre graphe des potentiels et graphe PERT

Tenant compte de la simplicité de dessin du graphe des potentiels qui est unique, on est amené à étudier la construction d'un graphe PERT à partir du graphe des potentiels. L'opportunité de cette idée réside dans le fait que les praticiens préfèrent travailler avec le graphe PERT qui est plus clair (chaque tâche est représentée par un arc), alors que le graphe des potentiels est encombrant vu le nombre important des arcs.
Le problème de passage du graphe des potentiels au graphe PERT a été largement étudié citons entre autres, A.C FISHER [FIS 68], M.HAYES en 1969 et F.STERBOUL [STE 81] qui se sont basés sur la notion de graphe arc-dual.
Nous présenterons alors, une nouvelle méthode pour la construction du PERT à partir du graphe des potentiels, qui est basée sur le principe de graphes adjoints des graphes.

3 Le graphe adjoint de graphe

Soit G=(X, U) un graphe orienté simple ou multiple. On construit à partir de G un graphe ou 'line graphe' noté L(G), appelé graphe adjoint (ou graphe représentatif des arcs) de G comme suit : Les sommets de L(G) sont en correspondance biunivoque avec les arcs de G. Pour des raisons de simplicité, on donne le même nom aux arcs de G et aux sommets correspondants de L(G).
2 sommets u et v de L(G) sont reliés par un arc de u vers v si et seulement si les arcs u et v de G sont tels que l'extrémité terminale de u coincide avec l'extrémité initiale de v c.à.d. T(u)= I(v) [AIG 67] .
Par la définition, tout graphe G admet un graphe adjoint L(G) unique. Par contre, deux graphes non isomorphes peuvent avoir le même graphe adjoint.

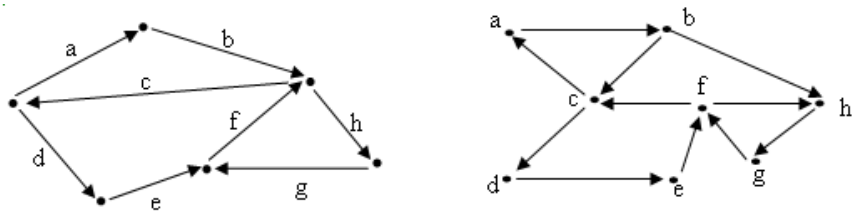


FIGURE 3. Un graphe G et son graphe adjoint L(G)

3.1 Le problème inverse : [MOU 02]

On pose le problème inverse suivant:
Etant donné un graphe H, est-il le graphe adjoint d'un graphe? Autrement dit, existe-t-il un graphe G tel que L(G) soit isomorphe à H, où H = L(G)? Avant de répondre à cette question, donnons la définition d'une configuration " Z ".

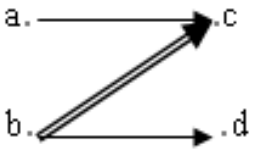


FIGURE 4. La configuration"Z"

3.1.1 ***Définition :*** G admet une configuration "Z" (qui est un sous graphe de G) si G contient 4 sommets a,b,c et d tels que si (a,c), (b,c) et (b,d) sont des arcs de G, alors (a,d) n'est pas un arc de G.
Dans le seul but de simplicité, on donnera le nom de barre du "Z" l'arc (b,c). La configuration "Z" apparaît lorsque 2 sommets ont des successeurs communs et des successeurs non communs ou par symétrie lorsque 2 sommets ont des prédécesseurs communs et des prédécesseurs non communs.

3.2 Quelques caractérisations des graphes adjoints : [AIG 67] [BER 73] [MOU 02]

Les graphes adjoints ont été très étudiés mais nous ne donnons dans cet article que les résultats qui nous intéressent.
1. H est le graphe adjoint d'un graphe si et seulement si H ne contient aucune configuration 'Z'.
2. H est le graphe adjoint d'un graphe G si et seulement si les arcs de H peuvent

être partitionnés en bipartis complets Bi=(Xi,Yi) , i=1.., m, tels que

(1) $$Xi \cap Xj = \emptyset \quad et \quad Yi \cap Yj = \emptyset, \quad \forall i \neq j$$

Les bipartis Bi de H sont alors en bijection avec les sommets notés aussi Bi qui ne sont ni sources ni puits.
Deux sommets Bi et Bj de G étant reliés par un arc de Bi vers Bj si et seulement si les bipartis complets Bi et Bj de H sont tels que Yi∩ Xj=∅

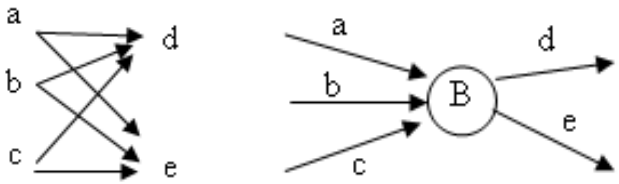


FIGURE 5. Un biparti complet B de H et l'étoile de G associée à B

Exemple:

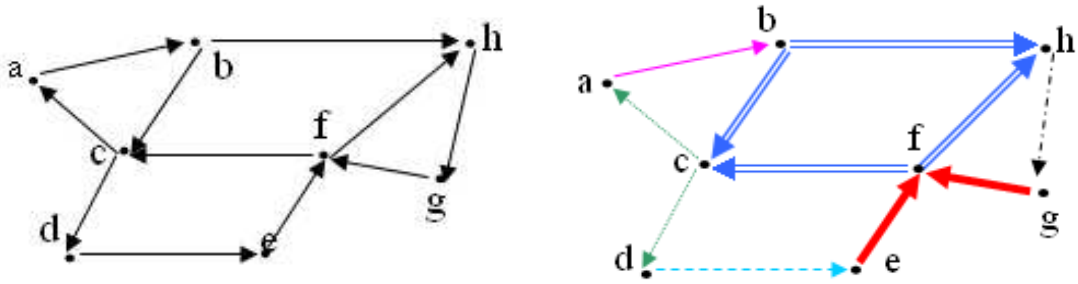


FIGURE 6. un graphe H et la partition de ses arcs en bipartis complets. Les arcs de chaque couleur représentent un biparti complet.

Supposons que le graphe H est le graphe adjoint d'un graphe G qu'on doit chercher. Pour cela, particionnons les arcs de H en bipartis complets (figure6) et qui sont :

```
B1 = { (a), (b)  }    B2 = { (c), (a,d)}
B3 = { (d), (e)  }    B4 = { (e,g), (f)}
B5 = {(b,f),(c,h)}    B6 = { (h), (g)}
```

Le graphe G résultant tel que H=L(G) est: (Figure7).

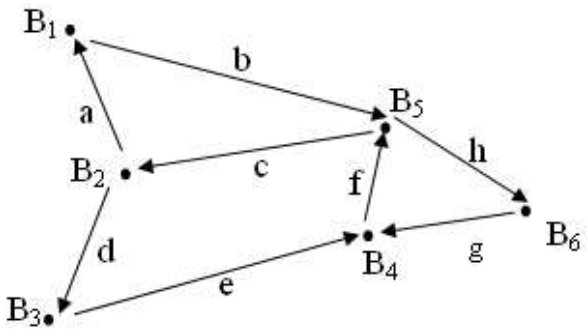


FIGURE 7. Le graphe G tel que H = L(G)

3. H est le graphe adjoint d'un graphe sans boucles si et seulement si H ne contient aucune configuration " Z ".
4. H est le graphe adjoint d'un graphe si et seulement si toute paire de sommets ayant des successeurs communs ont tous leurs successeurs communs.
5. H est le graphe adjoint d'un graphe si et seulement si toute paire de sommets ayant des prédécesseurs communs ont tous leurs prédécesseurs communs.

Ainsi H n'est le graphe adjoint d'aucun graphe si est seulement s'il existe une paire de sommets ayant des successeurs communs et des successeurs non communs ou des prédécesseurs communs et des prédécesseurs non communs (présence de Z).

4 Passage du graphe des potentiels au graphe PERT

A cause de la facilité d'utiliser le graphe PERT, on doit se concentrer sur l'étude de la possibilité de transformer le graphe des potentiels (nombre d'arcs important) au graphe PERT (nombre d'arcs réduit). On se pose alors le problème de savoir comment transformer H (qui est le graphe des potentiels) pour en faire un nouveau graphe qui est le graphe G (graphe PERT).
Le problème qui se pose, est ce que H contient des configurations Z ou non ? S'il ne contient pas des Z il est alors adjoint et la transformation est immédiate. Mais s'il contient des Z on est amené à éliminer la barre de chaque Z préservant naturellement les contraintes de succession. Etudions chaque cas à part :

4.1 Le graphe des potentiels est un graphe adjoint :

Construisons le graphe PERT à partir du graphe des potentiels dans le cas où celui-ci est un graphe adjoint (absence des Z).
En vertu des résultats du paragraphe 3.2., on procède comme suit : On partitionne

les arcs du graphe des potentiels en bipartis complets Bi = (Xi, Yi). Dans le graphe PERT que l'on veut construire, chaque Bi est représenté par un sommet encore noté Bi et sera le centre de l'étoile (voir exemple et figures 6,7.).

4.2 Le graphe des potentiels n'est pas un graphe adjoint

La construction du graphe PERT est cependant plus complexe dans le cas général où le graphe des potentiels n'est pas un graphe adjoint : il n'admet pas de partition des arcs en bipartis complets (à cause de la présence des Z). C'est dans ce cas qu'on doit le modifier afin de le transformer en graphe adjoint en préservant les contraintes d'antériorités.

Supposons que les tâches a1,..,am précèdent les tâches b1,..,bn .

Dans le graphe des potentiels,ces contraintes d'antériorité sont représentées par un biparti complet. Dans le graphe PERT, elles sont représentées par une étoile.

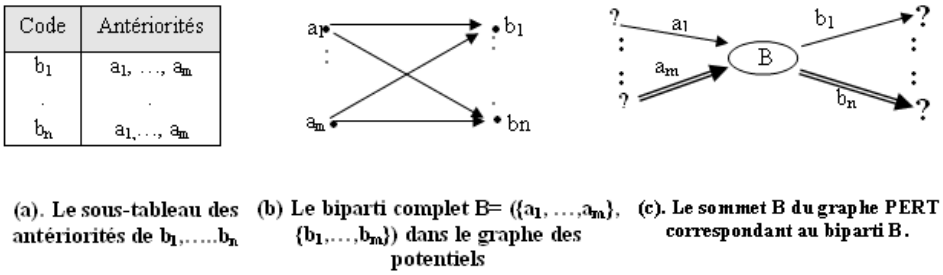


FIGURE 8. Correspondance entre un biparti complet du graphe des potentiels et une étoile du graphe PERT.

Revenons au problème de tâche fictive dans le graphe PERT. Si on a par exemple 4 tâches a,b,c et d avec les contraintes d'antériorité suivantes : a et b précèdent c, mais d est précédée par b uniquement. Dans le graphe des potentiels, il n'y a aucun problème pour la représentation de ces tâches. Elle est faite comme dans la figure 9 [CAR 88] . Or, pour le passage du graphe des potentiels (qui est considéré comme le graphe adjoint H), on est obligé à éliminer toutes les configurations " Z ". On introduit alors, dans le graphe des potentiels une tâche fictive f dans tout Z :

L'introduction des tâches fictives vise donc à éliminer toutes les configurations " Z " du graphe des potentiels, les contraintes restant inchangées. Il faut rappeler que les tâches fictives ne sont nullement nécessaires dans le graphe des potentiels mais ne sont introduites que pour construire le graphe PERT.

Une technique simple d'élimination consiste à remplacer la barre (b,c) de tout " Z " par deux arcs (b,f) et (f,c), selon la figure 9. f étant un sommet fictif.

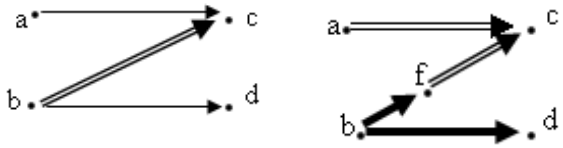


FIGURE 9. Représentation de Z et sa transformation en bipartis complets dans le graphe des potentiels.

5 Algorithme

Soit Gv un graphe des potentiels qui doit être orienté, valué, connexe et sans circuit. Gv étant un graphe conjonctif, organisé en niveaux. On veut construire le graphe PERT correspondant qui est appelé Ge.

Début

Si Gv contient des configurations Z **Alors**
Repérer les Zi (i = 1,2,…, m)
Pour i allant de 1 à m **Faire**
- Créer le sommet fi dans Gv
- Remplacer la barre (b_i, c_i) de Zi dans Gv par (b_i, f_i), (f_i, c_i)

Fpour
Fsi
- Repérer les bipartis dans Gv
- Représenter chaque biparti Bi dans Gv par un sommet Bi dans Ge
- Représenter les arcs de façon que :
Un arc est dessiné entre 2 sommets Bi et Bi dans Ge ssi les 2 bipartis Bi et Bi dans Gv sont tels que $Yi \bigcap Xj = \phi$.

Fin

FIGURE 10

L'algorithme se termine puisque la boucle pour n'est exécutée que dans le cas de présence de Z et le nombre de Z dans Gv est fini il ne peut en aucun cas être infini. L'élimination d'un Z est immédiate et elle est faite en une seule étape . Il suffit de repérer la barre de Z et la remplacer par deux arcs.

Les trois étapes suivantes ne traitent que la réorganisation des arcs et des sommets en bipartis, ensuite les transformer en étoiles comme on l'a vu précédemment (voir section 4.2).
Pour la complexité de l'algorithme, cela dépend de la structure de données proposée au départ pour la représentation du graphe ainsi que la structure de donnée qui héberge le graphe PERT.

6 Exemple

Considérons le tableau des contraintes suivant et le graphe des potentiels associé, les durées n'étant pas représentées:
Ayant repéré les " Z " dans le graphe des potentiels, on introduit les tâches fictives

Codes	Antériorités
α	-
A	α
B	α
C	A,B
D	A ,B
E	B
F	C
G	D,E
H	D,E
I	F
J	F,,G,H
K	G,H
w	I,J,K

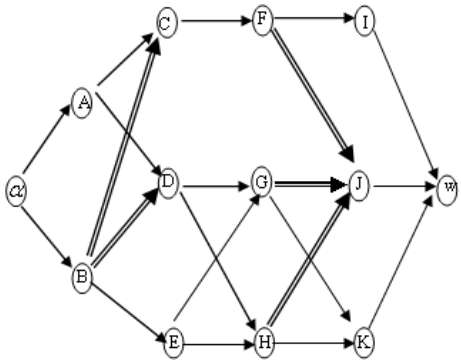


FIGURE 11. Table des antériorités T et Le graphe des potentiels

selon la figure 12, puis l'on réorganise le graphe modifié en niveaux: Cherchons les bipartis complets du graphe des potentiels :

```
B1 =({ }, {A, B}),   B2 ={B}, {E, f1}),  B3 = ({A,f1},{C,D}),
B4 ={C }, {F}),      B5 = ({D,E},{G,H}),  B6 ={F}, { f2, I}),
B7 ={G,H},{f3,K}),  B8 = ({f2,f3},{J}),  B9 = ({I,J,K}, {w}),
```

7 Conclusion

Ce travail vient d'élaborer un algorithme original qui introduit les graphes adjoints dans les problèmes d'ordonnancement de projet avec ou sans la présence des " Z " dans le graphe de potentiels et ceci pour la construction d'un graphe PERT. Il ouvre la voie à des perspectives, telles que l'optimisation de l'algorithme proposé

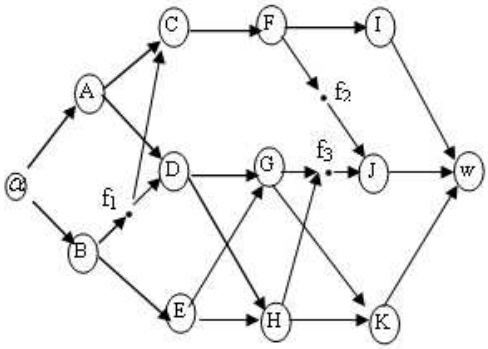


FIGURE 12. Le graphe des potentiels modifié par l'introduction des tâches fictives fi.

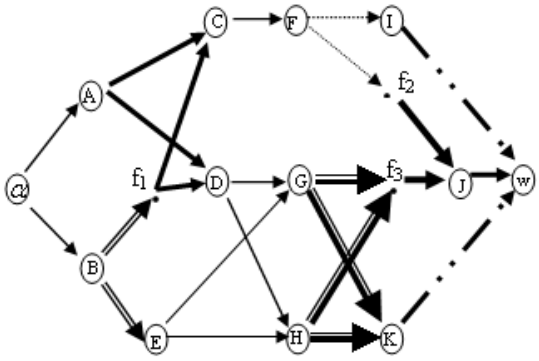


FIGURE 13. Le graphe des potentiels modifié avec réorganisation des tâches en niveaux et partition des arcs en bipartis complets.

pour réduire le nombre de tâches fictives, le traitement des contraintes de localisation temporelle (appelées également contraintes de durée) et l'algorithme de recherche du graphe PERT minimal en nombre de tâches fictives et/ou en nombre de sommets.

Bibliographie

[AIG 67] M. AIGNER, On the linegraph of a directed graphs, 1967.
[BER 73] C. BERGE, Graphes et hypergraphes, DUNOD, Paris, 1973.

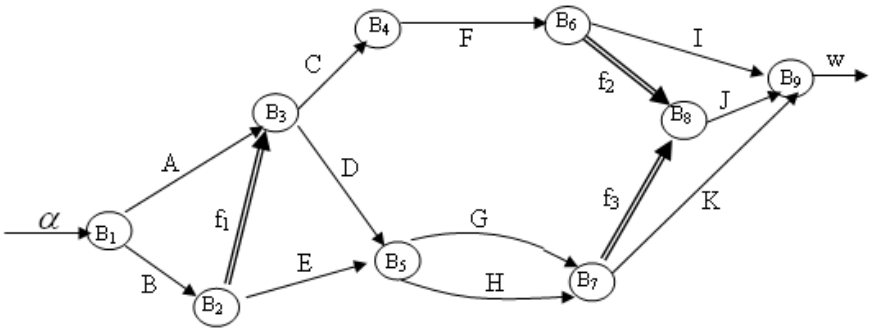


FIGURE 14. Construction du graphe PERT à partir du graphe des potentiels potentiels.

[CAR 88] J. CARLIER et P. CHRETIENNE, Problèmes d'ordonnancement Modélisation, Complexité, Algorithmes, MASSON, Paris, 1988.
[CRA 97] Y. CRAMA, L. DUPONT, G. FINKE, Recherche opérationnelle et gestion de la production, Nouvelles de la science et des technologies, 1997.
[DAL 01] J. M. DALBARADE et B. PETIT-JEAN, Ordonnancement et gestion des projets, 2001.
[ESQ 99] P. ESQUIROL et P. LOPEZ, L'ordonnancement, ECONOMICA, Paris, 1999.
[FIS 68] A.C. FISHER, J.S. LIEBMAN et G.L. NEMHAUSER, Computer construction of project networks, Communications of ACM, volume 11, N 7, juillet 1969.
[KEY 61] J. E.KELLEY, Critical-path planning and scheduling mathematical basis, 1961.
[MOU 02] N. MOUHOUB, Vers le graphe PERT minimal, thèse de Magister, Université de Sétif, 2002
[ROY 70] B. ROY, Algèbre moderne et théorie des graphes, tome 2, fascicule 3, Problèmes d'ordonnancement et ensembles de potentiels sur un graphe, DUNOD, Paris, 1970.
[STE 81] F.STERBOUL, et D.WERTHEIMER, Comment construire un graphe PERTminimal, RAIRO, 1981.

INSTITUT D'INFORMATIQUE, CENTRE UNIVERSITAIRE DE BORDJ BOU ARRÉRIDJ, ALGERIE
E-mail address: n_mouhoub@yahoo.fr

DÉPARTEMENT D'INFORMATIQUE, UNIVERSITÉ MED BOUDIAF M'SILA, ALGERIE

INSTITUT D'INFORMATIQUE, CENTRE UNIVERSITAIRE DE BORDJ BOU ARRÉRIDJ, ALGERIE

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LI**, Number 2, 2006

INDEXING MOBILE OBJECTS USING BRICKR STRUCTURES

ANDREEA SABĂU

Abstract. A growing number of applications manage mobile objects. The storage and the organization of data within databases is an open challenge. A new indexing method is proposed in this paper. The BrickR structure organizes continuously evolving objects with no extent using two structures: a temporary structure that manages recent data and a permanent structure that stores data belonging to the past. This access method treats identically the spatial and temporal dimensions, thus allowing spatial, temporal and spatio-temporal queries to be answered.

1. Introduction

There is a tendency today to study systems for management of objects with spatial evolution in time, particularly for management of mobile objects. These objects are also known as spatio-temporal objects.

The spatial attributes of which values have evolution in time can represent the shape and / or the location of objects, and these evolutions may be discrete or continuous. For example: land parcels have a discrete evolution in time of the positions and extents; the cars on a road are continuously changing their position, but not the shape. The most significant challenge is to store data about continuous changing.

A new spatio-temporal access method called BrickR is proposed in this paper. The spatial objects that BrickR is organizing are point objects (objects with no extent) with continuous spatial evolution. Such kind of object is a car on the road. This car might move with a variable speed and along the both senses.

Two sub-structures work together so as to manage efficiently the recently received and past data: a permanent physical structure designed as an R*-tree [2] and a temporary structure designed as a grid which evolves in time like a brick wall.

Received by the editors: November 10, 2006.

2000 *Mathematics Subject Classification.* 68P05, 68P20.

1998 *CR Categories and Descriptors.* H.2.2 [**Information Systems**]: Database Management – *Physical Design*; H.2.4 [**Information Systems**]: Database Management – *Systems*; H.2.8 [**Information Systems**]: Database Management – *Database Applications* .

The paper is organized as follows: Section 2 is an overview of some spatio-temporal access methods. Characteristics of spatio-temporal data referenced in this paper are presented in Section 3. Section 4 presents the structures of BrickR, the manner in which data received by the system is managed. In Section 5 some experimental results are provided. There are also conclusions and proposed ideas as future work in Section 6.

2. Related Work

A lot of work has been done on organizing spatio-temporal objects in index structures. These structures may be classified as structures which index past data, store data about present and past or index data about present and future (predictions) [8].

Some index structures which organize past information are the following:
STR-tree, TB-tree [10] - are R-tree like structures, with the main characteristic that they attempt to achieve trajectory preservation for the same object by storing trajectory segments of the same object in the same tree node. One advantage is in answering object-oriented queries, like "Which was the object's X trajectory?". On the other hand, spatial window queries are not efficiently solved due to the fact that even if some trajectory segments belonging to two different objects are spatially closed, they are stored in different tree nodes.
SETI [4] - divides the spatial domain into a static partition and the data corresponding to one cell of partition are organized into an R-tree; one segment which intersects two cells is divided and the resulting pieces are stored into the corresponding cells; one major drawback is the lack of efficiency solving temporal queries, because all the cells tree must be searched.

There are spatio-temporal access methods that manage present (and past) data:
2+3 R-tree [9], 2-3 TR-tree [1] - are index structures that contain two R-trees: one tree for points that represent present data and another R-tree for the trajectories from the past; the search is possible to be done in both R-trees.
LUR-tree [7] - indexes only current positions of objects, which means that historical queries are not supported; this is a structure adapted to frequent updates.
Hashing [13] - stores only current data; the space is partitioned into zones that may be overlapped; an object belongs to one zone and its accurate position is acquired using an auxiliary layer.

The following structures belong to the third class of spatio-temporal access methods, which manage present data and data for prediction of future movement.
Duality transformation [6] - transforms a line segment from the time-space domain into a 2D point (an equation x_t = at + b is represented by the point (a, b)); indexing some trajectory segments spatially closed is not a guarantee that the representative points are closely stored in tree nodes.
TPR-tree [12], PR-tree [3], STAR-tree [11], TPR*-tree [14] - are included into a

new category of STAM; these structures index the original time-space domain using parametric bounding rectangles; this means that the rectangles are functions of time and are built to enclose moving objects trajectories; these access methods employ different optimization strategies of the parametric boxes.

3. Spatio-Temporal Data

As it was mentioned earlier in this paper, a new indexing method for one-dimensional objects with continuous evolution in time is proposed. A mobile object of which extent is not for interest is represented as point. In order to facilitate the representation and implementation of the system the movement of these objects in one-dimensional space is considered. The extension to the movement in R^n space, $n \geq 2$, is straightforward, and the indexing of objects with shape (for example polygonal shape) is proposed as future work.

The motion of a spatio-temporal object in 1D space is continuous in time: the objects cannot disappear or be teleported. It is natural for that object to travel with variable speed during different time intervals. Therefore the movement on a time interval is represented as a function of time defined and continuous on that interval, and the trajectory of a mobile object is built using these functions of time. Graphically representing these functions, we obtain a set of connected segments (the end of a line is the start of the next trajectory line), except the situation when, during a time interval, the information about the object's motion is unknown. The trajectory segments of an object do not intersect (an object cannot travel back in time and cannot exist simultaneous in two different locations). The only common points are the end points of the two temporal consecutive segments.

Therefore, an essential feature of a spatio-temporal object is the arbitrary movement in spatial domain, but only the chronological travel in the temporal domain. In this paper we consider that the object's speed stays invariably on a certain time interval, so that the time function representing a trajectory segment is linear.

It is not imposed a certain unique manner of receiving the spatio-temporal data sent by the mobile objects. The objects may possess GPS equipment and may send positioning data regularly or after the change of the motion parameters (speed, direction), or they may have a local implementation of a process unit of data and send to the server the trajectory segments. In any of these situations, the input data for BrickR structure are linear functions of time, and each of them is defined on a certain time interval and represented as line segment. The temporal feature of motion is the foundation of the design of BrickR structure (see section 4.2).

4. BrickR Structure

The permanent structure of the BrickR-tree is based on the R-tree design, therefore in the next section the R-tree family access methods are presented.

4.1. **R-Trees.** The R-trees [5, 2] are index structures for point objects and spatial objects with extent in R^d, d=2. These spatial access methods are based on the division of space depending on data adapting to the position of objects in space. The inclusion relation is used to establish the hierarchy between tree nodes [5, 2].

The objects stored in the R-tree nodes are called Minimum Bounding Rectangles (MBR). An MBR is the smallest d-dimensional rectangle including one or more given objects. The leaf nodes contain pointers to objects stored in the database, but represented inside the index by its MBR. The internal nodes of the tree contain pointers to child nodes and the smallest rectangles that include all the entries of those child nodes. Generally, a record stored within an R-tree node has the structure (P, MBR). If the node is a leaf node, P is a pointer to an object and MBR is the MBR of that object. Otherwise, P points to a child node and MBR is the minimum rectangle that encloses the child node's entries.

Two parameters of an R-tree are the maximum and the minimum number of entries within a node. These are noted here by M and m respectively. Their values are chosen by taking into account the size of the page that stores a node, and on the request of minimizing the number of nodes accessed within a query. Usually $m = \left\lceil \frac{M}{2} \right\rceil$, but it can be even smaller. The only exception is the root node, which can contain a minimum of one entry.

The well-known structures of the R-tree family are the R-tree, R*-tree and R+-tree. The main difference between the R-tree, R*-tree and R+-tree is the insertion algorithm of a new object into the tree. In the case of R-tree and R*-tree, the first step is the selection of the leaf node into which the new object will be inserted, based on some heuristic. If necessary, the MBR of that node will be enlarged, to enclose the whole object. The insertion into R+-tree is done using the clipping method: an object is divided intro pieces regarding the leaf nodes that intersect the object to be inserted. Therefore, comparing to R-tree, the R+-tree has the advantage that between leaf nodes are no overlapping areas, but the main disadvantage is that there are indexed more objects because of the clipping method (there may be more pieces for a single object).

In the case of R-tree and R*-tree, the nodes of the same level can overlap, so that the division of space cannot be necessary a disjunctive one. An MBR can intersect or be included within the MBR of more nodes, but it is associated with only one node. On the other side, on the same level in an R+-tree, the MBRs of nodes define disjunctive partitions of space.

The R*-tree is actually an optimized R-tree, mainly by the insertion and deletion algorithms [2], therefore the structure used for BrickR-tree is the R*-tree.

4.2. **Structures of BrickR.** One of the drawbacks of using an R-tree as a STAM [1, 9] is the possible extension of an MBR because of a trajectory segment too long. It is known that a MBR having large margin and area may cause large overlapping areas among tree nodes and dead space within nodes, which decreases

the performances solving queries.

The BrickR structures are mainly designed to minimize the overlapping in the tree structure. More then overlapping area, the experimental results (see Section 5) prove a better value of the sum of perimeters and areas of the tree nodes MBRs.

The design of the BrickR structures had as start ideas the following facts:

(1) the forced enlargement of an MBR because of a too long trajectory segment;
(2) the objects random movement within the spatial domain;
(3) the strictly chronological evolution on the temporal domain.

BrickR is an index structure and is composed of two structures:

- the permanent structure: an R*-tree type structure used to index spatio-temporal data from the remote past; it is assumed to be stored in secondary memory, therefore the paginated storage of the nodes is facilitated;
- the temporary structure: a grid structure which indexes the newest ST data received by the system; this structure is stored in main memory, having the advantage of a short time accessing data and performing operations.

The Temporary Structure

The temporary structure is designed to offer an extra "thinking" moment of how to group objects into MBRs. The main idea is not to insert an object into the main structure as soon as the data is received by the system. Storing more recent objects allows a better clustering of these, having almost void chances to get overlapping areas between the resulting MBRs.

The grouping of objects is accomplished with the help of one grid-type layer on the spatial domain. This grid is obtained by dividing the n-dimensional space using (n-1)-dimensional hyper-planes parallel with the Ot axis. As it was earlier mentioned, this paper discus the indexing of 1D data, but the extrapolation in a space of a greater dimension is straightforward. Therefore, in order to index one-dimensional ST objects, the 2-dimensional space is divided using a set of lines parallel with the temporal axis.

In this case, this grid is composed by a set of strips. The setting of the dividing lines (the margins of strips) may take into account the spatial distribution and the density of mobile objects or these lines may be equally distanced.

As data is arriving into BrickR system, each segment data is inserted into intersecting grid strips, using the clipping method. That means that if a segment intersects two or more strips, it is divided into pieces, each piece being totally enclosed within a strip. These segments are later grouped into rectangular bounding boxes and then such a MBR is sent to the permanent structure to be inserted.

The construction operation of a new MBR occurs when a strip satisfies the

adequacy criterion, meaning the number of segments stored in that strip. The adequacy criterion can vary, in accordance with the density of objects or their trajectories. In most of the tests, it was checked whether the strip contains at least fan-out-T * stripOccupancy segments, where fan-out-T is defined as the maximum capacity of a node of BrickR-T structure and stripOccupancy is equal to 3. It has been considered that a greater value than 3 may have as effect the management of a too big number of segments within a strip. On the other side, based on test results, it has been found out that the value 3 is large enough not to get overlapping areas between the MBRs cut out from the grid strips.

he value of the stripOccupancy parameter and the division of grid can be set according to the known or predicted movement of objects. This stripOccupancy's value is seen as a compromise between the number of trajectory segments stored within a strip and the chance that some overlapping areas may appear.

It can be noticed in figure 1(a) that, when stripOccupancy = 1, overlapping areas are obtained even within the temporary structure (the shadowed areas). In the situation when a greater value for stripOccupancy parameter is used (eg. 2), the segments are clipped and no overlapping area is obtained (see figure 1(b)).

The algorithm for the MBR shaping operation is based on the clipping method, as well as the insert operation. To get a new MBR from a strip, the algorithm tries to find a 1D line perpendicular on Ot axis, or a (d-1)-dimensional hyper-plane in R^d space, which cuts the strip so that the new rectangle to contain at most fan-out-T segments. Each line segment from the current strip is clipped if the cutting line intersects it: one piece is stored in the new MBR and the rest of it remains in the strip.

An observation has to be made about the flexibility of the grid structure. Let consider tStart(S) to be the smallest timestamp of a start point segment by the strip S. It may happen that the number of the alive objects at time tStart(S) to be greater than fan-out-T. This means that a new MBR cannot be obtained by cutting the strip with a line perpendicular on Ot axis. In such a case, the strip is divided using a hyper-plane parallel to Ot axis, resulting two narrower strips (see figure 1(b)). The division operation may continue until a new MBR can be built. The reverse operation of joining two neighbor strips may be included in the management algorithms of the grid structure, depending on implementation or input data.

The Permanent Structure

The permanent structure has the architecture of an R*-tree [2], but the nodes are organized into two different sub-structures. The terminal nodes containing the segment data are stored into BrickR-T and are organized into linked lists. A linked list corresponds to a strip from the temporary structure. The other structure called BrickR-I stores the internal nodes and follow the organization of an

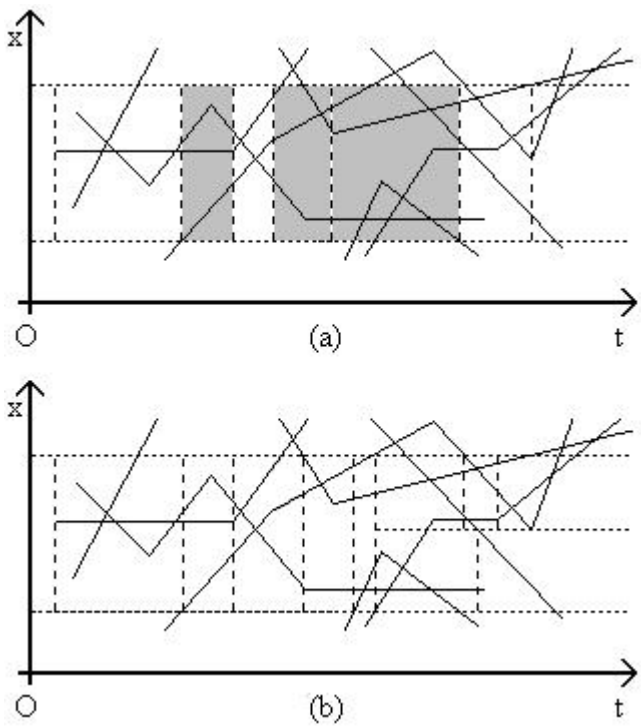


FIGURE 1. The construction of MBRs cut out from a grid strip, if fan-out-T = 4; the segment data is arriving in chronological order. (a) stripOccupancy = 1: the shadowed surface represents overlapping areas between MBRs (b) stripOccupancy = 2: no overlapping area is obtained.

R*-tree. Therefore, the insert operation into the permanent structure has two steps: the addition of the node containing segment data at the end of a linked list and the insertion of the node's MBR into the R*-tree like structure. In this way, the insertion of the terminal node is simplified.

The main advantages of the two structures of BrickR are emphasized:

- it does not have to insert separately each segment into the physical structure (many I/O operations are skipped), but the already grouped segments are sent as a node to the BrickR-T structure; the next step is the insertion of the new node's MBR into the upper structure (the BrickR-I)

- the odds to result overlapping areas between the MBRs sent to the permanent structure are minimized.

5. Experimental Results

A Delphi 2005 application using tables and stored procedures on a MS-SQL server has been developed to manage spatio-temporal data using 2D R*-tree, respectively BrickR. Tests has been run on two sets of data, containing different number of objects and using a fan-out of nodes of value 10 and a minimum occupied space of 40%. A record does not use a lot of space, therefore a paginated node stores usually more records than the fan-out considered. Some trial tests using a greater fan-out have showed an improvement of the BrickR efficiency and this demonstrates that the structure presented in this paper works well for realistic values of fan-out.

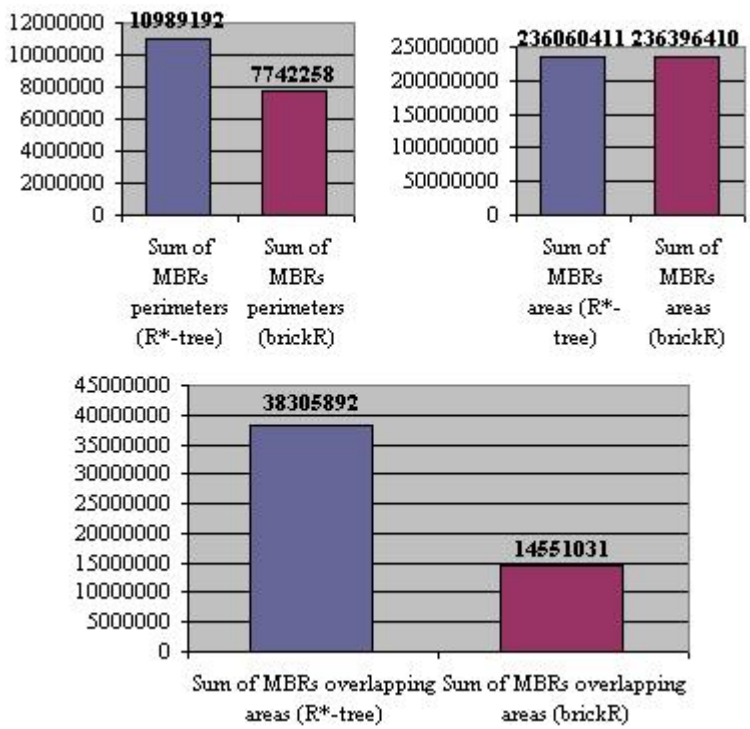


Figure 2. The comparisons between the sums of MBRs margins, the sums of the total areas, and the sums of the overlapping areas of the R*-tree and BrickR structures.

The point data signifying the end points of the trajectory segments have been randomly generated. Their coordinates (x, t) belong to a well-determined interval and the lengths of the resulting segments are not greater than some threshold (4%, 8.5%, 13% and 17% by the working space).

The parameters evaluated for the two tested access methods are the number of indexed objects, the number of nodes included into permanent structure, the sum of nodes MBR margin, the sum of nodes MBR area and the total value of the overlapping area between the nodes on the same tree level.

It was mentioned earlier that the clipping method has as consequence the growth of indexed objects number. This fact is pointed out by the tests: the amount of indexed objects during all tests within BrickR-I and BrickR-T structures is greater than the number of objects indexed using the R*-tree by approximately 45%. Nevertheless, the number of nodes occupied within BrickR structure is less than the number of nodes within R*-tree. This proves a much better usage of space occupied by BrickR structure: on the average, the node occupancy in the R*-tree is 58.61% and in the BrickR is 90.1%.

Figure 2 shows a lower value for the sums of MBRs margins and the overlapping area between the MBRs of BrickR structure's nodes, than the values obtained for R*-tree.

Another result noticed during tests is a substantial improvement of insertion operations running time: BrickR structure works 9 times faster than the R*-tree.

6. Conclusions and Future Work

The arguments and the experimental results show that the presented spatio-temporal indexing structure outperforms other ST index structures. Some improvements can be made for the insertion algorithm in BrickR. The optimization of temporal queries and the adjustment of the BrickR structure to allow mobile objects with shape to be indexed are proposed as future work.

References

[1] M. Abdelguerfi, J. Givaudan, K. Shaw, R. Ladner, *The 2-3 TR-tree, A Trajectory-Oriented Index Structure for Fully Evolving Valid-time Spatio-temporal Datasets*, In Proc. of the ACM Workshop on Adv. in Geographic Information Systems, ACM GIS, 29-34, 2002.

[2] N. Beckmann, H. P. Kriegel, R. Schneider, B. Seeger, *The R*-tree: An Efficient and Robust Access Method for Points and Rectangles*, In Proc. of the Intl. Conf. on Management of Data, SIGMOD, 322-331, 1990.

[3] M. Cai, P. Revesz, *Parametric R-Tree: An Index Structure for Moving Objects*, In Proc. of the Intl. Conf. on Management of Data, COMAD, 57-64, 2000.

[4] V. P. Chakka, A. Everspaugh, J. M. Patel, *Indexing Large Trajectory Data with SETI*, In Proc. of the Conf. on Innovative Data Systems Research, CIDR, 164-175, 2003.

[5] A. Guttman, *R-Trees: A Dynamic Index Structure for Spatial Searching*, In Proc. of the ACM Intl. Conf. on Management of Data, SIGMOD, 47-57, 1984.

[6] G. Kollios, D. Gunopoulos, V. J. Tsotras, *On Indexing Mobile Objects*, In Proc. of the ACM Symp. on Principles of Database Systems, PODS, 261-272, 1999.

[7] D. Kwon, Sj. Lee, S. Lee, *Indexing the Current Positions of Moving Objects Using the Lazy Update R-tree*, In Mobile Data Management, MDM, 113-120, 2002.

[8] M. F. Mokbel, T. M. Ghanem, W. G. Aref, *Spatio-temporal Access Methods*, IEEE Data Eng. Bull., 26(2), 40-49, 2003.

[9] M. A. Nascimento, J. R. O. Silva, Y. Theodoridis, *Evaluation of Access Structures for Discretely Moving Points*, In Proc. of the Intl. Workshop on Spatio-Temporal Database Management, STDBM, 171-188, 1999.

[10] D. Pfoser, C. S. Jensen, Y. Theodoridis, *Novel Approaches in Query Processing for Moving Object Trajectories*, In Proc. of the Intl. Conf. on Very Large Data Bases, VLDB, 395-406, 2000.

[11] C. M. Procopiuc, P. K. Agarwal, S, Har-Peled, *STAR-Tree: An Efficient Self-Adjusting Index for Moving Objects*, In Proc. of the Workshop on Alg. Eng. and Experimentation, ALENEX, 178-193, 2002.

[12] S. Saltenis, C. S. Jensen, S. T. Leutenegger, M. A. Lopez, *Indexing the Positions of Continuously Moving Objects*, In Proc. of the ACM Intl. Conf. on Management of Data, SIGMOD, 331, 342, 2000.

[13] Z. Song, N. Roussopoulos, *Hashing Moving Objects*, In Mobile Data Management, 161-172, 2001.

[14] Y. Tao, D. Papadias, J. Sun, *The TPR*-Tree: An Optimized Spatio-temporal Access Method for Predictive Queries*, In Proc. of the Intl. Conf. on Very Large Data Bases, VLDB, 790-801, 2003.

Faculty of Mathematics and Computer Science, Babes-Bolyai University, Cluj-Napoca, Romania

E-mail address: deiush@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LI**, Number 2, 2006

OPERATIONAL SEMANTICS OF TASK MODELS

ADRIANA TARŢA AND SIMONA MOTOGNA

Abstract. This paper proposes an operational semantic approach for the task models. Task models are used in the process of user centered software design to represent the work structure and the sequence of steps needed to reach a goal. The goal of our approach is to develop inference rules which will reflect the changes on the presentation aspects of an application when a certain temporal operator occurs. By using the proposed inferece rule we will show that when the deductive process is ended the enabled task sets are determined.

1. Introduction

Today, computers are used in almost every domain of our life. The success or failure of a software system is given by the support the system gives to its user in performing their tasks efficiently and with satisfaction. These features of a software systems are components of a software quality measure, called *usability*. In order to obtain a high usability level, a user centered approach in the design of interactive system should be used. In this paper we will discuss a method used in the design of interactive system called task-based design. The task-based design relies on the analysis of the tasks the users perform. The result of the task analysis is represented by task models. In this paper we will present a formal approach of task models based on operational semantics [10]. The paper is structured as follows: Section 2 presents the basic elements of task-based design, Section 3 presents some introductory notions about operational semantics followed by the presentation of our approach in describing task models using operational semantics, Section 4 presents few examples of applying operational semantics on some task trees and Section 5 presents the conclusions of our research and further work on our approach.

Received by the editors: November 28, 2006.

2000 *Mathematics Subject Classification.* 68-xx, 68Qxx.

1998 *CR Categories and Descriptors.* [**F.4.1**]: Theory of Computation – *Mathematical Logic*; [**H.5.2** [Information Systems]: Information Interfaces and Presentation – *User Interfaces*.

2. Task-based design

The first step in task based design of interactive systems is *task analysis*. Task analysis is the process of gathering data about tasks people perform and acquiring a deep understanding of it. The process of structuring data and gaining insight into the data is called *task modeling* [14]. Work structure is one of the most important aspects in task analysis. The design of an interactive system usually means restructuring the work and removing or adding tasks. Work structure can be captured in a task decomposition tree [15]. The tree forms a hierarchy where the high level tasks are found at the top of the tree and the most basic tasks are at the leaf nodes. In the representations proposed over time ([14], [7]), the tree are enhanced *constructors* that indicate time relationship between tasks. In the work structure model, the root of the tree is a goal with possibly some subgoals. Connected to goals are tasks and tasks can be connected on the same level by temporal operators. When designing for usability, the work structure is important for developing the most appropriate interaction structure and functionality.

Many task analysis methods have been developed and used in the design of software systems: HTA [1], GOMS [3], TAG [9], MAD [11], most of them using a textual formal notation. For complex systems, these methods were impossible to be used. In 1996, a new approach in task based design has been developed, consisting in a method for analysing the groupware work globally, not individually. This new method has been called GTA (Groupware Task Analysis) [13] and based on GTA a design method called DUTCH (Designing for Users and Tasks from Concepts to Handles) [16] has been developed. The task models (trees) developed using GTA used *constructors* to specify the time relationships between tasks. For the average usage, the following time relationships have proved sufficient: Concurrent, Choice, AnyOrder, Succesive and * (iteration) [14]. These constructors were: SUCC for sequential tasks, PAR for concurrent tasks, CHOICE for alternate task, * for iterative tasks and ANY for independent order tasks [14]. CTT (ConcurTaskTrees), another approach in task modeling, uses a more formal time relationship specification using LOTOS (Language Of Temporal Ordering Specification) operators [5]. The operators taken from LOTOS are: enabling, disabling, parallel composition (interleaving and full synchronization), choice, order independence and iteration. In the following, we will present the definition of each of the operators mentioned above. The original definitions are expressed in terms of processes [2]. For the subject of our paper, we will use the term *task* instead of using the term *process*.

The *choice operator* - denoted by [] - is defined as follows: if T1 and T2 are two tasks, then T1[]T2 denotes a task which behaves like T1 or T2.

The *interleaving operator* - denoted by ||| - if T1 and T2 are two tasks ready for an action (t1 and t2), then both action orderings (t1 after t2 ot t2 after t1) are possible.

The *synchronization operator* - denoted by || - T1 and T2 have to synchronize on some actions in order to exchange information.

The *sequential composition* (enabling) operator - denoted by - $\gg$ - T1 $\gg$ T2 - if T1 terminates successfully, then the execution of T2 is enabled.

The *disabling operator* - denoted by [> - T1 [>T2 - if T2 is started, then T1 is never performed.

The *iteration operator* - denoted by * - $T1^*$ - means that task T1 is iterative.

In the following subsection we will use operational semantics in order to systematically derive the actions (subtasks) that a task may perform from the structure of expression itself. The goal of this approach is to determine the Enabled Task Sets (ETS) [4], which will lead us to a *presentation model* of the user interface of an interactive systems.

3. Operational semantics for temporal operators

Operational semantics has been successfully used in specifying different programming languages. The principles can be applied with a few modifications for our goals.

We will denote the set of temporal operators by $\mathcal{O} = \{[], >>, [>, ||, |||, |=|, *\}$. The prority order among operators is: choice operator > parallel composition operators (interleaving, synchronization) > disabling operator > order independence operator> enabling operator [8].

Our aim is that starting from a task tree to obtain in a rigourous way the *enabled tasks sets* (tasks which are enabled at the same time) that will correspond to the user interface of an interactive system. Having the enabled tasks sets, we will able to determine the widgets of the user interface based on task types. Tasks types can be: editing, monitoring, selection of a single choice or selection of multiple choices and control [14]. In the following we will describe our approach in identifying the enabled tasks sets based on the operational semantics.

We will give the definition of a task tree starting from the definition of the tree concept [6]. A *task tree* has a root node (representing the goal of the task performance). Each node is either a leaf (a unit task) or an internal node (representing a subtask). An internal node has one or more children nodes and is called the parent of its child nodes. All children of the same node are siblings. Every two siblings are related by a temporal operator belonging to the set $\mathcal{O}$.

We will use Haskell-like data definitions to define abstract syntax formally:

```
Op=[](ST,ST) | ||(ST,ST)| [> (ST,ST)| >> (ST,ST)| ||| (ST,ST)| *(ST)
```

A task sequence may be defined as

$ST \rightarrow T\ op\ T \mid T\ op\ ST \mid T$

$T \rightarrow task$

$op \rightarrow \gg \mid [] \mid \parallel \mid ||| \mid [> \mid *.$

The *state* of the abstract machine for task trees has a *stack of tasks* and an *enabled task sets (ETS) collection* denoted by $\mathcal{E}$ where the enabled tasks sets generated by the execution of the abstract machine are saved.

The structured operators semantics for task models defines a relation "$\rightarrowtriangle$" which means "is transformed by a single execution step into" [12]. We define this relation by means of inference rules. An expression like "$[\![t_1]\!]\mathcal{E} \rightarrowtriangle [\![skip]\!]\mathcal{E}$" can be read as "execution of task t_1 with the enabled task sets $\mathcal{E}$ means execution of *skip* (no action) with the enabled task sets $\mathcal{E}$". An expression like "$[\![t_1]\!]\mathcal{E} \rightarrowtriangle [\![t_1']\!]\mathcal{E}'$" can be read as "execution of task t_1 with the enabled task sets $\mathcal{E}$ means execution of subtask t_1' with the enabled task sets $\mathcal{E}'$".

3.1. **Enable operator semantics.** When two tasks are related by the enabling operator it means that after the completion of the first task, the second task will start. This means that the two tasks will belong to different enabled tasks sets. This aspect will be described by the changes that will affect the enabled task sets $\mathcal{E}$. In the following we will describe the inference rules for the enable operator:

$$\frac{[\![t_1]\!]\mathcal{E} \rightarrowtriangle [\![t_1']\!]\mathcal{E}}{[\![t_1 \gg t_2]\!]\mathcal{E} \rightarrowtriangle [\![t_1';\ t_2]\!]\ \mathcal{E}}\ (ER_1)$$

The first rule refers to the situation when the task t_1 is evaluated to one of its subtasks t_1'. In this case, the expression $[\![t_1 \gg t_2]\!]\mathcal{E}$ is evaluated to the expression $[\![t_1';\ t_2]\!]\ \mathcal{E}$. In this case the ETS remains unchanged, because further processing must be done.

$$\frac{[\![t_1]\!]\mathcal{E} \rightarrowtriangle [\![skip]\!]\mathcal{E} + \{t_1\}}{[\![t_1 \gg t_2]\!]\mathcal{E} \rightarrowtriangle [\![skip;\ t_2]\!]\ \mathcal{E} + \{t_1\}}\ (ER_2)$$

The second rule handles the situation when the enabling task t_1 is already accomplished (the next step in its performance is skip (no action)). In this case, the expression $[\![t_1 \gg t_2]\!]\mathcal{E}$ is evaluated to the sequential execution of skip followed by the execution of the enabled task t_2. The ETS will be enriched with the accomplished task t_1.

$$\frac{}{[\![skip \gg t_2]\!]\mathcal{E} \rightarrowtriangle [\![t_2]\!]\ \mathcal{E}}\ (ER_3)$$

The third rule describes the way the expression $[\![skip \gg t_2]\!]\mathcal{E}$ is evaluated. In this situation, only task t_2 must be executed and the ETS remains unchanged.

$$\frac{}{[\![t_1 \gg skip]\!]\mathcal{E} \rightarrowtriangle [\![t_1]\!]\ \mathcal{E}}\ (ER_4)$$

The fourth rule describes the situation when the enabling task is a final task (it is the last child of a node). In this case the expression $[\![t_1 \gg skip]\!]$ is evaluated to the execution of the enabling task.

3.2. **Choice operator semantics.** If two tasks t_1 and t_2 are related by the choice operator, the performance of the task depends on users option. That is why we have added o in the middle of the operator's notation, representing the users' option. We make the convention that if o evaluates to the constant 1, then the first task is selected for execution (see ChR_1). If o evaluates to 2, then the second task will be selected for execution (see ChR_3). In order to be able to select one of two options, both tasks should be available at the same time. This aspect is reflected on the updates which affect the store by adding both tasks to the ETS. In the following we will present the inference rules for the choice operator:

$$\frac{[\![o]\!]\mathcal{E} \twoheadrightarrow 1\ \mathcal{E} \qquad [\![t_1]\!]\mathcal{E} \twoheadrightarrow [\![skip]\!]\mathcal{E}'}{[\![t_1\ [o]\ t_2]\!]\mathcal{E} \twoheadrightarrow [\![skip]\!]\ \mathcal{E}' + \{t_1,\ t_2\}}\ (ChR_1)$$

$$\frac{[\![o]\!]\mathcal{E} \twoheadrightarrow 1\ \mathcal{E} \qquad [\![t_1]\!]\mathcal{E} \twoheadrightarrow [\![t_1']\!]\mathcal{E}'}{[\![t_1\ [o]\ t_2]\!]\mathcal{E} \twoheadrightarrow [\![t_1']\!]\ \mathcal{E}' + \{t_1',\ t_2\}}\ (ChR_2)$$

$$\frac{[\![o]\!]\mathcal{E} \twoheadrightarrow 2\ \mathcal{E} \qquad [\![t_2]\!]\mathcal{E} \twoheadrightarrow [\![skip]\!]\mathcal{E}'}{[\![t_1\ [o]\ t_2]\!]\mathcal{E} \twoheadrightarrow [\![skip]\!]\ \mathcal{E}' + \{t_1,\ t_2\}}\ (ChR_3)$$

$$\frac{[\![o]\!]\mathcal{E} \twoheadrightarrow 2\ \mathcal{E} \qquad [\![t_2]\!]\mathcal{E} \twoheadrightarrow [\![t_2']\!]\mathcal{E}'}{[\![t_1\ [o]\ t_2]\!]\mathcal{E} \twoheadrightarrow [\![t_2']\!]\ \mathcal{E}' + \{t_1,\ t_2'\}}\ (ChR_4)$$

3.3. **Disable operator semantics.** If t_1 and t_2 are two tasks and t_2 is a disabling task, this means that when t_2 performance starts t_1 is disabled no matter which is the execution state of t_1 (i.e. t_2 is able to suspend the execution of any subtask of t_1). That is why the rule does not have an hypothesis part of the inference rule. This means that t_2 must belong to every enabled tasks set generated by t_1. We have described this fact by updating the ETS $\mathcal{E}$ by adding the disabling task to each set of the ETS:

$$\frac{}{[\![t_1\ [>\ t_2]\!]\ \mathcal{E} \twoheadrightarrow [\![skip;\ t_2]\!]\ \mathcal{E}'}\ (\text{DR})$$

where $\mathcal{E}' = \{\{l_1, \ldots, l_n, t_2\}\},\ \forall\{l_1, \ldots, l_n\} \subset E$.

3.4. **Parallel composition operator semantics - Pure interleaving.** The pure interleaving operator relating the task T1 and T2 expresses nothing but any interleaving of the actions of T1 with the actions of T2:

$$\frac{}{[\![skip \ ||| \ skip]\!]\ \mathcal{E} \rightarrow\!\!\!\!\circ\ [\![skip]\!]\ \mathcal{E}}\ (ConcR_1)$$

The first rule expresses the fact that by interleaving no actions (skip), the result will be also skip.

$$\frac{[\![t_1]\!]\mathcal{E} \rightarrow\!\!\!\!\circ\ [\![t_1']\!]\mathcal{E}'}{[\![t_1 \ ||| \ skip]\!]\ M \rightarrow\!\!\!\!\circ\ [\![t_1']\!]\ \mathcal{E}'}\ (ConcR_2)$$

$$\frac{[\![t_2]\!]\mathcal{E} \rightarrow\!\!\!\!\circ\ [\![t_2']\!]\mathcal{E}'}{[\![skip \ ||| \ t_2]\!]\ M \rightarrow\!\!\!\!\circ\ [\![t_2']\!]\ \mathcal{E}'}\ (ConcR_3)$$

The rules $ConcR_2$ and $ConcR_3$ describe the situation when one of the interleaving tasks is evaluated to one of its children and the ETS is changed to $\mathcal{E}'$, then interleaving the task with skip (no action), the result will be the execution of the child task and the ETS will be $\mathcal{E}'$.

$$\frac{[\![t_1]\!]\mathcal{E} \rightarrow\!\!\!\!\circ\ [\![t_1']\!]\mathcal{E}'}{[\![t_1 \ ||| \ t_2]\!]\ \mathcal{E} \rightarrow\!\!\!\!\circ\ [\![t_1' \ ||| \ t_2]\!]\ \mathcal{E}'}\ (ConcR_4)$$

$$\frac{[\![t_2]\!]\mathcal{E} \rightarrow\!\!\!\!\circ\ [\![t_2']\!]\mathcal{E}'}{[\![t_1 \ ||| \ t_2]\!]\ \mathcal{E} \rightarrow\!\!\!\!\circ\ [\![t_1 \ ||| \ t_2']\!]\ \mathcal{E}'}\ (ConcR_5)$$

$$\frac{\begin{array}{c}[\![t_1]\!]\mathcal{E} \rightarrow\!\!\!\!\circ\ [\![t_1']\!]\mathcal{E}' \\ [\![t_2]\!]\mathcal{E} \rightarrow\!\!\!\!\circ\ [\![t_2']\!]\mathcal{E}''\end{array}}{[\![t_1 \ ||| \ t_2]\!]\ \mathcal{E} \rightarrow\!\!\!\!\circ\ [\![t_1' \ ||| \ t_2']\!]\ \mathcal{E}\mathcal{R}}\ (ConcR_6)$$

where $\mathcal{E}\mathcal{R}$ is the enabled tasks set formed as follows: $\forall\{X\} \subset \mathcal{E}', \forall\{Y\} \subset \mathcal{E}''$, $\mathcal{E}\mathcal{R} = \{\{X, Y\}\}$

The last three rules $ConcR_4$, $ConcR_5$ *and* $ConcR_6$ describe the situation when the execution of one (or both) of the interleaving tasks is evaluated to the execution of one of its children with changes on the ETS ($\mathcal{E}'$)(i.e. $[\![t_1]\!]\mathcal{E} \rightarrow\!\!\!\!\circ\ [\![t_1']\!]\mathcal{E}'$). In this case, the result of the evaluation of interleaving of the two tasks will be evaluated to the interleaving of the child task(s) with implications on the ETS (i.e. $[\![t_1 \ ||| \ t_2]\!]\ \mathcal{E} \rightarrow\!\!\!\!\circ\ [\![t_1' \ ||| \ t_2]\!]\ \mathcal{E}'$).

3.5. **Parallel composition operator semantics - Full Synchronization.** The rules for the parallel composition operator are very similar to those for the interleaving operator and will be presented in the following:

$$\frac{}{[\![\textbf{sync } t_1 \mid\mid \textbf{sync } t_2]\!]\ \mathcal{E} \twoheadrightarrow [\![t_1 \mid\mid t_2]\!]\ \mathcal{E}}\ (SyncR_1)$$

$$\frac{[\![t_2]\!]\mathcal{E} \twoheadrightarrow [\![t_2']\!]\mathcal{E}'}{[\![skip \mid\mid \textbf{sync } t_2]\!]\ \mathcal{E} \twoheadrightarrow [\![t_2]\!]\ \mathcal{E}'}\ (SyncR_2)$$

$$\frac{[\![t_1]\!]\mathcal{E} \twoheadrightarrow [\![t_1']\!]\mathcal{E}'}{[\![\textbf{sync } t_1 \mid\mid skip]\!]\ \mathcal{E} \twoheadrightarrow [\![t_1]\!]\ \mathcal{E}'}\ (SyncR_3)$$

$$\frac{[\![t_1]\!]\mathcal{E} \twoheadrightarrow [\![t_1']\!]\mathcal{E}'}{[\![t_1 \mid\mid \textbf{sync } t_2]\!]\ \mathcal{E} \twoheadrightarrow [\![t_1' \mid\mid \textbf{sync } t_2]\!]\ \mathcal{E}'}\ (SyncR_4)$$

$$\frac{[\![t_2]\!]\mathcal{E} \twoheadrightarrow [\![t_2']\!]\mathcal{E}'}{[\![\textbf{sync } t_1 \mid\mid t_2]\!]\ \mathcal{E} \twoheadrightarrow [\![\textbf{sync } t_1 \mid\mid t_2']\!]\ \mathcal{E}'}\ (SyncR_5)$$

$$\frac{\begin{array}{c}[\![t_1]\!]\mathcal{E} \twoheadrightarrow [\![t_1']\!]\mathcal{E}' \\ [\![t_2]\!]\mathcal{E} \twoheadrightarrow [\![t_2']\!]\mathcal{E}''\end{array}}{[\![\textbf{sync } t_1 \mid\mid \textbf{sync } t_2]\!]\ \mathcal{E} \twoheadrightarrow [\![t_1' \mid\mid t_2']\!]\ \mathcal{E}\mathcal{R}}\ (SyncR_6)$$

where $\mathcal{E}\mathcal{R}$ is the enabled tasks set formed as follows: $\forall\{X\} \subset \mathcal{E}', \forall\{Y\} \subset \mathcal{E}''$, $\mathcal{E}\mathcal{R} = \{\{X, Y\}\}$

3.6. **Order independence operator semantics.** The expression $t_1 \mid = \mid t_2$ means that t_1 and t_2 can be executed in any order, but both tasks must be executed (for example filling in the user name and the password in a login form). In terms of enabled task sets, both tasks will belong to the same enabled tasks set, fact illustrated by adding both tasks to the same enabled tasks set of ETS. The rules will be presented in the following:

$$\frac{[\![t_1]\!]\mathcal{E} \twoheadrightarrow [\![t_1']\!]\mathcal{E}}{[\![t_1 \mid = \mid t_2]\!]\ \mathcal{E} \twoheadrightarrow [\![t_1';\ t_2]\!]\ \mathcal{E} + \{t_1,\ t_2\}}\ (OIR_1)$$

$$\frac{[\![t_2]\!]\mathcal{E} \twoheadrightarrow [\![t_2']\!]\ \mathcal{E}}{[\![t_1 \mid = \mid t_2]\!]\mathcal{E} \twoheadrightarrow [\![t_1;\ t_2']\!]\ \mathcal{E} + \{t_2,\ t_1\}}\ (OIR_2)$$

The first two rules describe the situation when one of the two tasks is evaluated to the execution of one of its children (let's say t_1 is evaluated to the execution of t_1'). In this case, the expression $[\![t_1 \mid = \mid t_2]\!]\ \mathcal{E}$ is evaluated to the expression

$[\![t_1';\ t_2]\!]\ \mathcal{E} + \{t_1,\ t_2\}$ which means the execution of t_1' followed by the execution of t_2.

The following rules (OIR_3 and OIR_4) regard the situation when one of the tasks is skip and the other is evaluated to the execution of one of its subtasks. In this case the expression evaluates to the execution of the subtask and the ETS is updated by adding the parent task.

$$\frac{[\![t_2]\!]\mathcal{E} \twoheadrightarrow [\![t_2']\!]\ \mathcal{E}}{[\![skip \mid = \mid t_2]\!]\mathcal{E} \twoheadrightarrow [\![t_2']\!]\ \mathcal{E} + \{t_2\}}\ (OIR_3)$$

$$\frac{[\![t_1]\!]\mathcal{E} \twoheadrightarrow [\![t_1']\!]\ \mathcal{E}}{[\![t_1 \mid = \mid skip]\!]\mathcal{E} \twoheadrightarrow [\![t_1']\!]\ \mathcal{E} + \{t_1\}}\ (OIR_4)$$

3.7. **Iteration operator semantics.** The iteration operator associated to a task means that the task is iterative (after completing an execution, the task can start its execution again).

The first inference rule say that if a task execution is accomplished ($[\![t_1]\!]\mathcal{E} \twoheadrightarrow [\![skip]\!]\mathcal{E}$) then task's iteration is evaluated to skip and the ETS is updated by adding a new set containing the iterative task:

$$\frac{[\![t_1]\!]\mathcal{E} \twoheadrightarrow [\![skip]\!]\mathcal{E}}{[\![t_1^*]\!]\ \mathcal{E} \twoheadrightarrow [\![skip]\!]\ \mathcal{E} + \{t_1\}}\ (ItR_1)$$

If the iterative task is not accomplished (one of its subtasks is running), then the iteration of the task is evaluated to the iteration of its child and the ETS is updated by adding the parent task.

$$\frac{[\![t_1]\!]\mathcal{E} \twoheadrightarrow [\![t_1']\!]\mathcal{E} + \{t_1\}}{[\![t_1^*]\!]\ \mathcal{E} \twoheadrightarrow [\![t_1'^*]\!]\ \mathcal{E} + \{t_1\}}\ (ItR_2)$$

3.8. **Complementary rules.** In addition to the rules associated to each operator we will need some rules for sequential execution. The need for these rules appears when composed temporal operators should be handled. The evaluation of such kind of expressions is reduced to the evaluation of some sequential tasks. The rules are presented in the following:

$$\frac{}{[\![skip; t]\!]\ \mathcal{E} \twoheadrightarrow [\![t]\!]\ \mathcal{E}}\ (SR_1)$$

The rule SR_1 says that skip (no action) followed by the execution of the task t is evaluated to the execution of task t and the ETS is not affected.

$$\frac{[\![t_1]\!]\mathcal{E} \rightarrow [\![t_1']\!]\mathcal{E}'}{[\![t_1; t_2]\!]\ \mathcal{E} \rightarrow [\![t_1'; t_2]\!]\ \mathcal{E}'}\ (SR_2)$$

If the task t_1 is evaluated to the execution of one of its children t_1', the expression $[\![t_1; t_2]\!]\ \mathcal{E}$ is evaluated to the execution of the child task (t_1') followed by the execution of t_2. The ETS is updated with the changes introduced by the execution of t_1'.

$$\frac{t_1 \rightarrow skip}{[\![t_1]\!]\ \mathcal{E} \rightarrow [\![skip]\!]\ \mathcal{E} + \{t_1\}}\ (ExR_1)$$

The rule ExR_1 is used when t_1 is a leaf in the task tree and its execution is completed (is transformed in skip). In this case the expression $[\![t_1]\!]\ \mathcal{E}$ where $\mathcal{E}$ is the ETS is evaluated to skip and the ETS is updated with a new task (t_1).

4. Examples

In the following we will present some examples of generating enabled tasks sets using the operational semantics. The first example is related to a task tree using the enabling operator. The task tree is presented in Figure 1. We have to evaluate the expression $[\![B \gg D \gg E]\!]\mathcal{E}$ which is the frontier of the tree. Using the rule

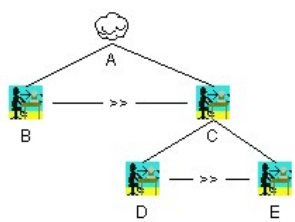


Figure 1. Task tree with enabling operators

ER_2 we will make the first step in our deduction as follows.

$$\frac{[\![B]\!]\mathcal{E} \rightarrow [\![skip]\!]\mathcal{E} + \{B\}}{[\![B\ \gg\ D\ \gg\ E]\!]\mathcal{E} \rightarrow [\![skip;\ D\ \gg\ E]\!]\ \mathcal{E} + \{B\}}$$

Let us denote $\mathcal{E}' = \mathcal{E} + \{B\}$. By applying the rule SR_1 we will evaluate $[\![skip;\ D \gg E]\!]\mathcal{E}'$.

$$\frac{}{[\![skip;\ D\ \gg\ E]\!]\ \mathcal{E}' \rightarrow [\![D\ \gg\ E]\!]\ \mathcal{E}'}$$

Now, using ER_2 we have:

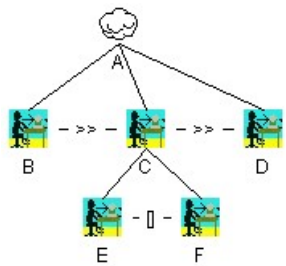


FIGURE 2. Task tree with choice operator

$$\frac{[\![D]\!]\mathcal{E}' \rightarrow [\![skip]\!]\mathcal{E}' + \{D\}}{[\![D \gg E]\!]\mathcal{E}' \rightarrow [\![skip;\ E]\!]\ \mathcal{E}' + \{D\}}$$

Let us denote $\mathcal{E}'' = \mathcal{E}' + \{D\}$. Using The rule SR_1, $[\![skip; E]\!]\mathcal{E}''$ is evaluated to $[\![E]\!]\ \mathcal{E}''$.

$$\frac{}{[\![skip; E]\!]\mathcal{E}'' \rightarrow [\![E]\!]\ \mathcal{E}''}$$

Now, we will apply ExR_1 to evaluate $[\![E]\!]\ \mathcal{E}''$ and we will obtain:

$$\frac{E \rightarrow skip}{[\![E]\!]\mathcal{E}'' \rightarrow [\![skip]\!]\ \mathcal{E}'' + \{E\}}$$

At the end of the deduction process the final content of the store will be composed by three enabled tasks sets: $\{B\}$, $\{D\}$, and $\{E\}$.

The second example handles a task tree using the choice operator. We will consider that the user selects the first option from the available ones (see Figure 2). We have to evaluate the expression $[\![B \gg E[o]F \gg D]\!]\mathcal{E}$. Using ER_2 for the task B we will obtain:

$$\frac{[\![B]\!]\mathcal{E} \rightarrow [\![skip]\!]\ \mathcal{E} + \{B\}}{[\![B \gg E\ [o]\ F \gg D]\!]\mathcal{E} \rightarrow [\![skip;\ E\ [o]\ F \gg D]\!]\ \mathcal{E} + \{B\}}$$

Let us denote $\mathcal{E}' = \mathcal{E} + \{B\}$. Applying ChR_1 and ER_2we will obtain:

$$\frac{\dfrac{[\![o]\!]\mathcal{E}' \rightarrow\ 1\ \mathcal{E}' \qquad E \rightarrow skip}{[\![E\ [o]\ F]\!]\mathcal{E}' \rightarrow [\![skip]\!]\mathcal{E}' + \{E, F\}}}{[\![E\ [o]\ F \gg D]\!]\mathcal{E}' \rightarrow [\![skip; D]\!]\mathcal{E}' + \{E, F\}}$$

Let us denote $\mathcal{E}'' = \mathcal{E}' + \{E, F\}$. We will apply SR_1 and the next step in the deductive process will be:

$$\frac{}{[\![skip; D]\!]\mathcal{E}'' \rightarrow [\![D]\!]\mathcal{E}''}$$

Applying ExR_1 we will obtain:

$$\frac{D \rightarrow skip}{[\![D]\!]\mathcal{E}'' \rightarrow [\![skip]\!]\mathcal{E}'' + \{D\}}$$

At the end of the deductive process the content of the ETS is $\mathcal{E}'' + \{D\}$ which means $\{E, F\},\ \{D\}, \{B\}$.

The following example illustrates the process of building ETS when the task model contains parallel (interleaving) tasks. The expression we must evaluate is $D \gg E[]F \ ||| \ G[]H$(see Figure 3).

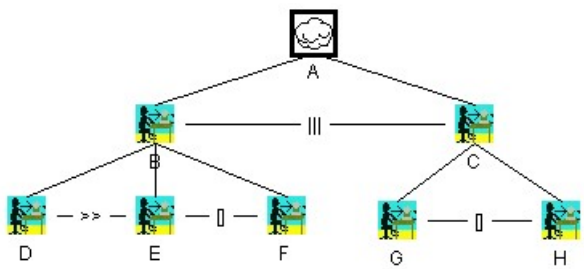


FIGURE 3. Task tree using the pure interleaving operator

$$\frac{[\![D]\!]\mathcal{E} \twoheadrightarrow [\![skip]\!]\mathcal{E} + \{D\}}{[\![D \ \gg \ E\ [o_1]\ F \ ||| \ G\ [o_2]\ H]\!]\mathcal{E} \twoheadrightarrow [\![skip;\ E\ [o_1]\ F \ ||| \ G\ [o_2]\ H]\!]\ \mathcal{E} + \{D\}}$$

Let us denote $\mathcal{E}' = \mathcal{E} + \{D\}$. We will evaluate $[\![skip;\ E\ [o_1]\ F \ ||| \ G\ [o_2]\ H]\!]\ \mathcal{E}'$ using SR_1.

$$\frac{}{[\![skip;\ E\ [o_1]\ F \ ||| \ G\ [o_2]\ H]\!]\mathcal{E}' \twoheadrightarrow [\![E\ [o_1]\ F \ ||| \ G\ [o_2]\ H]\!]\ \mathcal{E}'}$$

Applying SR_1 and ChR_1 we will obtain:

$$\frac{\dfrac{[\![o_1]\!]\ \mathcal{E}' \qquad E \twoheadrightarrow skip}{[\![E\ [o_1]\ F]\!]\ \mathcal{E}' \twoheadrightarrow [\![skip]\!]\mathcal{E}' + \{E, F\}}}{[\![E\ [o_1]\ F \ ||| \ G\ [o_2]\ H]\!]\ \mathcal{E}' \twoheadrightarrow [\![skip\ \ ||| \ G\ [o_2]\ H]\!]\ \mathcal{E}' + \{E, F\}}$$

Let us denote $\mathcal{E}'' = \mathcal{E}' + \{E, F\}$. Using $ConcR_3$ and ChR_2 the result will be:

$$\frac{\dfrac{[\![o_2]\!]\ \mathcal{E}'' \twoheadrightarrow 2\ \mathcal{E}'' + \{G, H\} \qquad G \twoheadrightarrow skip}{[\![G\ [o_2]\ H]\!]\mathcal{E}'' \twoheadrightarrow [\![skip]\!]\mathcal{E}'' + \{G, H\}}}{[\![skip\ \ ||| \ G\ [o_2]\ H]\!]\ \mathcal{E}'' \twoheadrightarrow [\![skip \ ||| \ skip]\!]\ \mathcal{ER}}$$

The final ETS will be $\mathcal{ER} = \{\{D, G, H\},\ \{E, F, G, H\}$.

5. CONCLUSIONS AND FURTHER WORK

In this article we have presented a new approach of task models based on operational semantics. We have given a definition of task trees and we have written inference rules for the temporal operators used to describe task models. We have captured on the inference rules aspects regarding the updates that take place at

the enabled tasks sets collection ($\mathcal{E}$). The process of building the presentation model will be based on the obtained ETS collection.

As future research directions our goals are: to build the presentation model based on the enabled tasks sets obtained using operational semantics and to extend our approach in order to build a dialog model also (transitions between states) starting from task models.

References

[1] J. Annett and K.D. Duncan. Task analysis and training design. *Journal of Occupational Psychology*, 41:211–221, 1967.

[2] T. Bolognesi and E. Brinksma. Introduction to the iso specification language lotos. *Comput. Netw. ISDN Syst.*, 14(1):25–59, 1987.

[3] S. Card, T. Moran, and A. Newell. *The Psychology of Human-Computer Interaction*. Cariere. Lawrence Erlbaum Associates, 1983.

[4] L. Marucci, F. Paterno, and C. Santoro. *Multiple and Cross-Platform User Interfaces: Engineering and Application Frameworks*, chapter Supporting Interactions with Heterogeneous Platforms Through User and Task Models, pages 217–238. H. Javahery and A. Seffah (eds.), 2003.

[5] G. Mori, F. Paternò, and C. Santoro. CTTE: Support for developing and analyzing task models for interactive system design. *IEEE Transactions on Software Engineering*, 28(9):1–17, 2002.

[6] NIST. National institute for standardization and technology. http://www.nist.gov/dads/HTML/tree.html.

[7] F. Paternò. Model-based tools for pervasive usability. *Interacting with Computers*, 2004.

[8] F. Paternò, C. Mancini, and S. Meniconi. ConcurTaskTrees: A diagrammatic notation for specifying task models. In *INTERACT '97: Proceedings of the IFIP TC13 Interantional Conference on Human-Computer Interaction*, pages 362–369. Chapman & Hall, Ltd., 1997.

[9] S.J. Payne. Task Action Grammar. In Bullinger H. J. and Shackel B., editors, *Proceedings INTERACT'84*, pages 139–144, North-Holland, 1984.

[10] G. D. Plotkin. A Structural Approach to Operational Semantics. Technical Report DAIMI FN-19, University of Aarhus, 1981.

[11] D. Scapin and C. Pierret-Golbreich. Towards a method for task description: MAD. *Work with Display Units*, 89:371–380, 1989.

[12] B. Sufrin and R. Bornat. Animating Operational Semantics with JAPE. citeseer.ist.psu.edu/485702.html.

[13] R. van Loo, G. van der Veer, and M. van Welie. Groupware Task Analysis in practice: a scientific approach meets security problems. In *7th European Conference on Cognitive Science Approaches to Process Control*, 1999.

[14] M. van Welie. *Task-based User Interface Design*. PhD thesis, Vrije Universiteit Amsterdam, 2001.

[15] M. van Welie, G. van der Veer, and A. Koster. Integrated representations for task modeling. In *Tenth European Conference on Cognitive Ergonomics*, pages 129–138, 21-23 August 2000.

[16] M. van Welie and G.C. van der Veer. Structured methods and creativity: a happy Dutch marriage. In *Co-Designing 2000*, 2000.

Department of Computer Science, Babeş-Bolyai University, Cluj-Napoca, Romania
E-mail address: adriana, motogna@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LI**, Number 2, 2006

SEQUENT CALCULUS IN COMPUTING DEFAULT EXTENSIONS

MIHAIELA LUPEA

Abstract. Justified and constrained default logics are the versions of default logic that have the property of semi-monotonicity. Based on this property, in this paper we present an iterative approach of the problem of computing the justified and constrained extensions of a propositional default theory. The sequent calculus and its complementary system, the antisequent calculus, are used to check the cautious applicability condition for defaults.

Keywords: default logic, nonmonotonic reasoning, theorem proving.

1. Introduction

The nonmonotonic reasoning is an important part of human reasoning and represents the process of drawing conclusions from incomplete information. Adding new facts may later invalidate these conclusions which are only plausible, not necessarily true. Default logics (classical, justified, constrained, rational) formalize *default reasoning*, a special case of nonmonotonic reasoning. These logical systems overcome the lack of information by making default assumption about a situation. The *defaults* are nonmonotonic inference rules used to model laws which are true with a few exceptions.

A *default theory* ([8]) $\Delta = (\mathcal{D}, \mathcal{W})$ consists of a set $\mathcal{W}$ (the *facts*) of consistent formulas of first order logic and a set $\mathcal{D}$ of *default rules*. $\mathcal{W}$ represents absolute (sometimes incomplete) knowledge about the world while $\mathcal{D}$ represents defeasible knowledge.

A *default* has the form $d = \frac{\alpha:\beta_1,\ldots,\beta_m}{\gamma}$, where: α is called *prerequisite*, $\beta_1, \ldots, \beta_m$ are called *justifications* and γ is called *consequent*.

A default $d = \frac{\alpha:\beta_1,\ldots,\beta_m}{\gamma}$ can be applied and thus derive γ if α is deductible (derivable) and it is consistent to assume $\beta_1, \ldots, \beta_m$ (meaning that $\neg\beta_1, \ldots, \neg\beta_m$ can not be derived).

Default extensions contain all the formulas obtained from the set of facts using the classical inference rules and the defaults. The elements of extensions are called

Received by the editors: November 10, 2006.

2000 *Mathematics Subject Classification.* 03B79, 68T15, 68T27, 68T20.

1998 *CR Categories and Descriptors.* I.2[**Artificial Intelligence**]: Logic in artificial intelligence – *default logics, nonmonotonic reasoning, theorem proving*;

nonmonotonic theorems (*beliefs*). The set of defaults used in the construction of an extension is called the *generating default set* for the considered extension.

The versions (classical([8]), justified ([4]), constrained([9]), rational([7]) of default logic, by different applicability conditions, try to provide an appropriate definition of consistency condition for the justifications of the defaults, and thus to obtain many interesting and useful properties for these logics: existence of extensions, semi-monotonicity, commitment to assumptions, cumulativity, regularity.

In the literature there were developed several methods to solve the problem of computing extensions of the versions of default logic, using different approaches.

In the paper [3], a relaxed stratification of a default theory is the primary search-space pruning technique for computing the classical extensions. The semantic tableaux method is adapted to be used as a general or local prover.

Exten([1]) is a system that computes classical, justified and constrained extensions, based on an operational approach and uses pruning techniques for search tree.

Semantic tableaux method is used in [11] to compute classical extensions for a decidable subset of default logic. An uniform approach, based on a modified version of propositional semantic tableaux method, of computing constrained and rational default is presented in [5].

In paper [10], default reasoning is integrated into existing model elimination in order to solve the query-answering problem for constrained and cumulative default logics.

Based on the semi-monotonicity property of justified and constrained default logics, and their relationship, in this paper we propose an iterative approach for generating these two types of default extensions. The sequent calculus and antisequent calculus, as proof systems, are used to check the cautious applicability condition for defaults.

The paper is structured as follows. In section 2 two complementary proof systems for propositional logic, sequent calculus and anti-sequent calculus, are described. Section 3 presents the main aspects of *justified* and *constrained* default logics. Section 4 explains our approach, introducing the theoretical model for computing justified and constrained extensions. Conclusions and future work are outlined in Section 5.

2. Sequent calculus and anti-sequent calculus in propositional logic

This section presents two complementary systems: the sequent calculus and the anti-sequent calculus, used to check the derivability and non-derivability in propositional logic.

The *sequent calculus* method, as an improvement of Gentzen natural deduction system, is a direct and syntactic proof method.

A *sequent* has the form: $U \Rightarrow V$, where U and V are finite sets of propositional formulas. U is called *antecedent* and V is called *succedent*.

A *basic sequent* contains the same formula, A, in both antecedent and succedent: $U, A \Rightarrow V, A$;

Semantics: The sequent $U \Rightarrow V$ is *true* if each model of U is also a model for at least one of the formulas of V. All basic sequents are true, therefore they are the *axioms* of sequent calculus.

The inference rules of sequent calculus are presented in TABLE 1.

This proof method consists in reducing an initial sequent, by successive applications of sequent inference rules, in order to obtain basic sequents.

TABLE 1. Sequent rules

connective	Introduction into antecedent	Introduction into succedent
$\neg$	$(\neg_l)\ \frac{U \Rightarrow V, A}{U, \neg A \Rightarrow V}$	$(\neg_r)\ \frac{U, A \Rightarrow V}{U \Rightarrow V, \neg A}$
$\wedge$	$(\wedge_l)\ \frac{U, A, B \Rightarrow V}{U, A \wedge B, \Rightarrow V}$	$(\wedge_r)\ \frac{U \Rightarrow A, V \quad U \Rightarrow B, V}{U \Rightarrow A \wedge B, V}$
$\vee$	$(\vee_l)\ \frac{U, A \Rightarrow V \quad U, B \Rightarrow V}{U, A \vee B, \Rightarrow V}$	$(\vee_r)\ \frac{U \Rightarrow A, B, V}{U \Rightarrow A \vee B, V}$
$\rightarrow$	$(\rightarrow_l)\ \frac{U \Rightarrow A, V \quad U, B \Rightarrow V}{U, A \rightarrow B \Rightarrow V}$	$(\rightarrow_r)\ \frac{U, A \Rightarrow B, V}{U \Rightarrow A \rightarrow B, V}$

The *derivability* from propositional logic is expressed in sequent calculus as follows: $U1, U2, \ldots, Un \longmapsto V1 \vee V2 \vee \ldots \vee Vm$ if and only if the sequent $U1, U2, \ldots, Un \Rightarrow V1, V2, \ldots, Vm$ is *true*, meaning that from the conjunction of hypothesis at least one of the formulas from succedent can be proved.

The *anti-sequent calculus* for propositional logic was introduced in [2] as the complementary system of sequent calculus.

An *anti-sequent* has the form $U \not\Rightarrow V$, where U, V are finite sets of propositional formulas.

Semantics: $U \not\Rightarrow V$ is *true* if there is a model M of U in which all the formulas of V are false, and M is an *anti-model* for this anti-sequent.

An anti-sequent $U \not\Rightarrow V$ is called a *basic anti-sequent* if all the formulas of U and V are atomic formulas and $U \cap V = \emptyset$. The basic anti-sequents are true and represent the axioms of this system.

The *non-derivability* from propositional logic is expressed in anti-sequent calculus as follows: $U1, U2, \ldots, Un \not\longmapsto V1 \wedge V2 \wedge \ldots \wedge Vm$ if and only if the anti-sequent $U1, U2, \ldots, Un \Rightarrow V1, V2, \ldots, Vm$ is *true*, meaning that from the conjunction of hypothesis none of the formulas from succedent can be proved.

TABLE 2 contains the inference rules of anti-sequent calculus, used to reduce an initial anti-sequent to an axiom that represents a partial anti-model for the initial anti-sequent.

TABLE 2. Anti-sequent rules

Introduction into antecedent	Introduction into consequent
$(\neg^c_l)\ \frac{U \not\Rightarrow V, A}{U, \neg A \not\Rightarrow V}$	$(\neg^c_r)\ \frac{U, A \not\Rightarrow V}{U \not\Rightarrow V, \neg A}$
$(\wedge^c_l)\ \frac{U, A, B \not\Rightarrow V}{U, A \wedge B, \not\Rightarrow V}$	$(\wedge^c_{r1})\ \frac{U \not\Rightarrow A, V}{U \not\Rightarrow A \wedge B, V} \mid (\wedge^c_{r2}) \frac{U \not\Rightarrow B, V}{U \not\Rightarrow A \wedge B, V}$
$(\vee^c_{l1})\ \frac{U, A \not\Rightarrow V}{U, A \vee B \not\Rightarrow V} \mid (\vee^c_{l2})\ \frac{U, B \not\Rightarrow V}{U, A \vee B \not\Rightarrow V}$	$(\vee^c_r)\ \frac{U \not\Rightarrow A, B, V}{U \not\Rightarrow A \vee B, V}$
$(\rightarrow^c_{l1})\ \frac{U \not\Rightarrow A, V}{U, A \rightarrow B \not\Rightarrow V} \mid (\rightarrow^c_{l2})\ \frac{U, B \not\Rightarrow V}{U, A \rightarrow B \not\Rightarrow V}$	$(\rightarrow^c_r)\ \frac{U, A \not\Rightarrow B, V}{U \not\Rightarrow A \rightarrow B, V}$

We remark that the difference between TABLE 1 and TABLE 2 consists in splitting the rules with two premisses from sequent calculus into pairs of rules in anti-sequent calculus. Thus the exhaustive search in sequent calculus becomes nondeterminism in anti-sequent calculus and the reduction process is a linear one.

The following theorem shows the complementarity of these two proof systems:

Theorem 2.1 ([2])
The anti-sequent $U \not\Rightarrow V$ is true if and only if the sequent $U \Rightarrow V$ is not true.

3. Justified and Constrained default logics

Justified default logic was introduced by Lukaszewicz ([4]). This version of default logic solves the problem of inconsistencies consequents-justifications using a *support set*, but the inconsistencies justifications-justifications are still not detected. *The existence of justified extensions is guaranted and the property of semi-monotonicity is satisfied.*

In *constrained default logic* ([9]), the assumptions (stored in a set of constraints) from the reasoning process are used to express a global consistency condition for justifications. This logic is strongly regular, *semi-monotonic, commits to assumptions and guarantees the existence of constrained extensions.*

The results from [6] show that default theories can be represented by unitary theories (all the defaults have only one justification, $d = \frac{\alpha:\beta}{\gamma}$) in such a way that extensions (classical, justified, constrained, rational) are preserved. In this paper we will use only unitary default theories and the following notations:

$Prereq(d) = \alpha$, $Justif(d) = \beta$, $Conseq(d) = \gamma$, $Prereq(\mathcal{D}) = \bigcup_{d \in \mathcal{D}} Prereq(d)$,
$Justif(\mathcal{D}) = \bigcup_{d \in \mathcal{D}} Justif(d)$, $Conseq(\mathcal{D}) = \bigcup_{d \in \mathcal{D}} Conseq(d)$,
$Th(X) = \{A | X \vdash A\}$ the classical deductive closure of the set X of formulas.

All versions of default logics were introduced using fixed-point operators. The definitions below are the original ones for justified and constrained default logics.

Definition 3.1([4])
Let $\Delta = (\mathcal{D}, \mathcal{W})$ be a default theory. For any pair (S, U) of sets of formulas, let $\Gamma_1(S,U)$ and $\Gamma_2(S,U)$ be the smallest sets of formulas which satisfy:

a) $\mathcal{W} \subseteq \Gamma_1(S,U)$;

b) $\Gamma_1(S,U) = Th(\Gamma_1(S,U))$;

c) For any $\frac{\alpha:\beta}{\gamma}$, if $\alpha \in \Gamma_1(S,U)$ and $\forall \eta \in U \cup \{\beta\}$, $S \cup \{\gamma\} \not\vdash\!\!\!\longrightarrow \neg\eta$ then $\gamma \in \Gamma_1(S,U)$ and $\beta \in \Gamma_2(S,U)$.

A pair (E, J) of sets of formulas is a *justified extension* of Δ if and only if $E = \Gamma_1(E,J)$ and $J = \Gamma_2(E,J)$.

E is the *actual extension* and J is the *support set*. The applicatibility condition c) permits the detection of inconsistencies consequents-justifications, but the support set may be inconsistent, meaning that defaults with contradictory justifications were applied.

Definition 3.2([9])
Let $\Delta = (\mathcal{D}, \mathcal{W})$ be a default theory. For any set T of formulas, let $\Upsilon(T)$ be the pair of the smallest sets (S', T') of formulas which satisfy:

a) $\mathcal{W} \subseteq S' \subseteq T'$;

b) $S' = Th(S')$ and $T' = Th(T')$;

c) For any $\frac{\alpha:\beta}{\gamma}$, if $\alpha \in S'$ and $T \cup \{\gamma\} \not\vdash\!\!\!\longrightarrow \neg\beta$ then $\gamma \in S'$ and $\beta, \gamma \in T'$.

A pair (E, C) of sets of formulas is a *constrained extension* of Δ if and only if $\Upsilon(C) = (E, C)$.

The *actual extension* (E) is embedded in a consistent *context* (C) containing the facts, the consequents and all the justifications assumed to be true in the construction of E.

These definitions are difficult to be used in the process of constructing extensions and thus equivalent characterizations of extensions were proposed.

The *semi-monotonicity* property is defined as follows:

Definition 3.3
Let $\Delta = (\mathcal{D}, \mathcal{W})$ be a default theory and $\mathcal{D}'$ be a set of defaults such that $\mathcal{D} \subseteq \mathcal{D}'$. If (E, C) is a default extension of Δ, then there is a default extension (E', C') of the theory $(\mathcal{D}', \mathcal{W})$, with $E \subseteq E'$ and $C \subseteq C'$.

Justified and constrained default logics are both semi-monotonic ([9]), meaning that new defaults can augment but never destroy previous extensions. The nonmonotonicity in this two versions of default logic is caused by addition of new facts which can invalidate formulas already derived.

The semi-monotonicity property guarantees the existence of justified and constrained extensions. In the worst case $(Th(\mathcal{W}), \emptyset)$ is the only justified extension and $(Th(\mathcal{W}), Th(\mathcal{W}))$ is the only constrained extension of the default theory $(\mathcal{D}, \mathcal{W})$.

Another advantage of this property is from a computational point of view. These two types of default extensions are constructible in a truly iterative way by applying one applicable default rule after another.

The relationship between justified and constrained logics is expressed by the following results from ([5]):

- A constrained extension is a justified extension, satisfying the property that all justifications of the applied defaults are consistent with the actual extension.
- Each constrained extension is included in at least one justified extension.

4. Computing justified and constrained default extensions

This section presents an uniform operational approach for computing justified and constrained extensions. It is inspired from the operational semantics ([5]) which characterizes these two types of extensions. Due to the smi-monotonicity we define the problem of computing the justified and constrained extension as an iterative process, applying the applicable defaults one by one.

Definition 4.1
Let $\Delta = (\mathcal{D}, \mathcal{W})$ be a default theory.

- A triple of the form $\langle E_p, J_p, D_p \rangle$ is called a *j-structure* if $D_p \subseteq \mathcal{D}$, $E_p = \mathcal{W} \cup Conseq(D_p)$, and $J_p = Justif(D_p)$.
- A triple of the form $\langle E_p, C_p, D_p \rangle$ is called a *c-structure* if $D_p \subseteq \mathcal{D}$, $E_p = \mathcal{W} \cup Conseq(D_p)$, and $C_p = \mathcal{W} \cup Conseq(D_p) \cup Justif(D_p)$.

The applicability conditions for defaults in justified and constrained default logics can be expressed by the following definition:

Definition 4.2

- The default $d = \frac{\alpha:\beta}{\gamma}$ is *j-applicable with respect to* $\langle E_p, J_p, D_p \rangle$ if:
  - the sequent $E_p \Rightarrow \alpha$ is true;
  - $\forall \eta \in J_p \cup \{\beta\}$ the anti-sequent $E_p, \gamma \not\Rightarrow \neg\eta$ is true.
- The default d is *c-applicable with respect to* $\langle E_p, C_p, D_p \rangle$ if:
  - the sequent $E_p \Rightarrow \alpha$ is true;
  - the anti-sequent $C_p, \gamma \not\Rightarrow \neg\beta$ is true.

The sequent calculus is used to express the derivability of the prerequisite, while the fact that the justification is believed (its negation is not derivable) is expressed using the anti-sequent calculus.

Definition 4.3
To a default $d = \frac{\alpha:\beta}{\gamma}$ we assign a mapping d^j from the set of j-structures into the set of j-structures as follows:

- $d^j(\langle E_p, J_p, D_p\rangle) = \langle E_p \cup \{\gamma\}, J_p \cup \{\beta\}, D_p \cup \{d\}\rangle$ if d is j-applicable wrt $\langle E_p, J_p, D_p\rangle$;
- $d^j(\langle E_p, J_p, D_p\rangle) = \langle E_p, J_p, D_p\rangle$ otherwise.

To a default $d = \frac{\alpha:\beta}{\gamma}$ we assign a mapping d^c from the set of c-structures into the set of c-structures as follows:

- $d^j(\langle E_p, C_p, D_p\rangle) = \langle E_p \cup \{\gamma\}, C_p \cup \{\beta, \gamma\}, D_p \cup \{d\}\rangle$ if d is c-applicable wrt $\langle E_p, J_p, D_p\rangle$;
- $d^j(\langle E_p, C_p, D_p\rangle) = \langle E_p, C_p, D_p\rangle$ otherwise.

These mappings model a *caution application* of the defaults, meaning that once a default is applied it can not lead to inconsistency further in the process of building an extension.

Definition 4.4
Let $\langle E_p, J_p, D_p\rangle$ be a j-structure, $\langle E_p, C_p, D_p\rangle$ be a c-structure and D be a set of defaults.

- $\langle E_p, J_p, D_p\rangle$ is *j-stable* wrt D if $d^j(\langle E_p, J_p, D_p\rangle) = \langle E_p, J_p, D_p\rangle, \forall d \in D$
- $\langle E_p, C_p, D_p\rangle$ is *c-stable* wrt D if $d^c(\langle E_p, C_p, D_p\rangle) = \langle E_p, C_p, D_p\rangle, \forall d \in D$

A stable structure characterizes the end of the reasoning process in which were used all the applicable defaults.

Definition 4.5
A j-structure $\langle E_p^n, J_p^n, D_p^n\rangle$ is *j-accesible* from the j-structure $\langle E_p^0, J_p^0, D_p^0\rangle$ if there is a sequence of defaults $(d_1, d_2, \ldots, d_n)$ and a sequence o j-structures $(\langle E_p^0, J_p^0, D_p^0\rangle, \langle E_p^1, J_p^1, D_p^1\rangle, \ldots, \langle E_p^n, J_p^n, D_p^n\rangle)$ such that $d_i^j(\langle E_p^{i-1}, J_p^{i-1}, D_p^{i-1}\rangle) = \langle E_p^i, J_p^i, D_p^i\rangle$, for $i = 1, \ldots, n$.

Definition 4.6
A c-structure $\langle E_p^n, C_p^n, D_p^n\rangle$ is *c-accesible* from the c-structure $\langle E_p^0, C_p^0, D_p^0\rangle$ if there is a sequence of defaults $(d_1, d_2, \ldots, d_n)$ and a sequence o c-structures $(\langle E_p^0, C_p^0, D_p^0\rangle, \langle E_p^1, C_p^1, D_p^1\rangle, \ldots, \langle E_p^n, C_p^n, D_p^n\rangle)$ such that $d_i^j(\langle E_p^{i-1}, C_p^{i-1}, D_p^{i-1}\rangle) = \langle E_p^i, C_p^i, D_p^i\rangle$, for $i = 1, \ldots, n$.

If a j-structure $\langle E_p, J_p, D_p\rangle$ is j-accessible from $(\mathcal{W}, \emptyset, \emptyset)$ then it corresponds to a partial justified extension. If a c-structure $\langle E_p, C_p, D_p\rangle$ is c-accessible from $(\mathcal{W}, \mathcal{W}, \emptyset)$ then it corresponds to a partial constrained extension.

Theorem 4.1
Let $\Delta = (\mathcal{D}, \mathcal{W})$ be a default theory. The j-structure $\langle E_p, J_p, D_p\rangle$ corresponds to the justified extension $(Th(E_p), J_p)$ of Δ, with D_p as generating default set if and only if $\langle E_p, J_p, D_p\rangle$ is j-stable wrt $\mathcal{D}$ and $\langle E_p, J_p, D_p\rangle$ is j-accesible from $(\mathcal{W}, \emptyset, \emptyset)$.

Theorem 4.2

Let $\Delta = (\mathcal{D}, \mathcal{W})$ be a default theory. The c-structure $\langle E_p, C_p, D_p \rangle$ corresponds to the constrained extension $(Th(E_p), Th(C_p))$ of Δ, with D_p as generating default set if and only if $\langle E_p, C_p, D_p \rangle$ is c-stable wrt $\mathcal{D}$ and $\langle E_p, C_p, D_p \rangle$ is c-accesible from $(\mathcal{W}, \mathcal{W}, \emptyset)$.

For lack of space we will not give the proofs of the above theorems.

5. Conclusions and further work

In this paper we defined the problem of computing the justified and constrained extension as an iterative process. Due to the semi-monotonicity property of these two versions of default logics, the applicable defaults have been applied one by one in order to build an extension. The sequent calculus and anti-sequent calculus proof systems were used to check the applicability conditions for defaults.

As further work we will implement an algorithm based on this approach, using a top-down techinque and pruning for efficiency, in order to generate all justified and constrained extensions of a propositional default theory.

References

[1] Antoniou, G., Courtney, A.P., Ernst, J., Williams, M.A., "A System for Computing Constrained Default logic Extensions", Logics in Artificial Intelligence, Lecture Notes in Artificial Intelligence, Vol. 1126, 1996, pp. 237–250.

[2] Bonatti, P., Olivetti, N., "Sequent Calculi for Propositional Nonmonotonic Logics", ACM Trans. Comput. Log., 2002, pp. 226–278.

[3] Cholewinski, P., Marek, W., Truszczynski, M., "Default reasoning system DeReS", Proceedings of KR-96, Morgan Kaufmann, 1996, pp. 518–528.

[4] Lukasiewicz, W., "Considerations on default logic - an alternative approach", Computational Intelligence **4**, 1988, pp. 1–16.

[5] Lupea, M. "Nonmonotonic reasoning using default logics", Ph.D. Thesis, "Babes-Bolyai" University, Cluj-Napoca, 2002.

[6] Marek, W., Truszczynski, M., "Normal form results for default logics", Non-monotonic and Inductive logic, LNAI Vol. 659, Springer Verlag, 1993, pp. 153–174.

[7] Mikitiuk, A., Truszczynski, M., "Rational default logic and disjunctive logic programming", Logic programming and non-monotonic reasoning, MIT Press, 1993, pp. 283–299.

[8] Reiter, R., "A Logic for Default reasoning", Artificial Intelligence **13**, 1980, pp. 81–132.

[9] Schaub, T.H., "Considerations on default logics", Ph.D. Thesis, Technischen Hochschule Darmstadt, Germany, 1992.

[10] Schaub, T.H., "XRay system: An implementation platform for local query-answering in default logics", Applications of Uncertainty Formalisms, Lecture Notes in Computer Science, Vol. 1455, Springer Verlag, 1998, pp. 254–378.

[11] Schwind, C., "A tableaux-based theorem prover for a decidable subset of default logic", Proceedings CADE, Springer Verlag, 1990.

Babeş-Bolyai University, Faculty of Mathematics and Computer Science, Cluj-Napoca, Romania

E-mail address: lupea@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume LI, Number 2, 2006

ON MODEL-DRIVEN DEVELOPMENT FOR WEB APPLICATIONS

IOAN LAZĂR AND DAN COJOCAR

Abstract. The importance of requirements engineering for web systems is increasing today. Only few methodologies provides a systematic approach for the specification of web systems through requirements models. New results that address model transformation from requirements to web system design were recently obtained in the context of using QVT language.

In this paper we propose an approach for deriving web system design from requirements models in the context of a model-driven development process. We propose an extension of AndroMDA basic development process that can be suitable also for other model-driven processes. We also extend the AndroMDA presentation profile for modeling conversational flows for web applications.

1. Introduction

Current trends in software development focus on the specification of models and model transformations. Model-driven development is a successful methodology for model transformations based on Model Driven Architecture [8]. Model-driven development starts at the computational independent level (CIM) with a business model that capture system requirements. Then the initial CIM model is refined and a platform independent model (PIM) is obtained. Finally, the code is generated by transforming the PIM model into a platform specific model (PSM).

In this paper we focus on early steps of model-driven development: obtaining a PIM model from business requirements. Model transformations from requirements to web system designs were recently investigated by Koch et al [16]. In order to capture web process models they used activity diagrams and introduced an UML profile based on requirements metamodels introduced in [4]. Our approach is different because we focus on using *UML 2.0 state machines* to express the process models in web applications. In this paper we propose to use standard UML and

Received by the editors: November 20, 2006.

2000 *Mathematics Subject Classification.* 68U07, 68U35.

1998 *CR Categories and Descriptors.* D.2.1 [**Requirements/Specification**]: Methodologies; D.2.2 [**Design Tools and Techniques**]: Computer-aided software engineering (CASE), Object-oriented design methods, State diagrams; D.2.11 [**Software Architectures**]: Domain-specific architectures.

the extension mechanism provided through stereotypes [10]. The intent is to allow the modeling process to be performed using any modeling tool that conforms to UML 2.0 and UML extension mechanism.

State machine models were also used by AndroMDA organization[1] which developed several profiles and transformation modules called cartridges. AndroMDA provides also transformation cartridges for Struts [5] (see [2]) and Java Server Faces. We also introduce a new profile that extends AndroMDA profile for presentation layer. This profile allows to model conversational flows in web applications. Moreover the profile is designed such that to allow transformations for other frameworks, such as Spring MVC [1] and Spring WebFlow [3].

The remainder of this paper is structured as follows. Section 2 introduces a new model-driven development process for web applications. A new UML profile for presentation layer is defined in Section 3. Section 4 discusses how the profile and the model-driven development process can be used together. Finally, in Section 5 some conclusions and future work are outlined.

2. Model-Driven Development for Web Applications

The basic idea of model-driven development is to build platform independent designs and to generate code for specific platforms. Recently Koch et al [16, 4] introduced a model-driven approach for web systems. Their approach are structured around OO-H method [7, 14] and UWE [12, 11, 13, 15]. The PIM models used by the authors are:

- *Process model* – defines business processes/workflows using UML activities. This is the main model from which the content and navigation models can be derived using Query/View/Transformation [9].
- *Content model* – contains objects needed for the construction of web pages content. The classes of this model can be generated from the process model.
- *Navigation model* – defines page flow navigation and menus organization. Also this model can be generated from the process model using QVT transformations.
- *Presentation model* – represents pages layout and design.

The basic idea in [16, 4] is to use activity diagrams for describing web business processes and then to derive other PIM artifacts using QVT transformation rules. Our approach is to use state machine diagrams instead of activity diagrams. UML state machine semantic is more suitable for modeling web user interface than UML activity diagrams. Other differences between our approach and the approach from [16, 4] are: (a) the *Content model* will be derived from signal properties associated with transitions between states; (b) our approach is suitable for processes that produces web systems having the modern layer architecture presented in Figure 1.

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

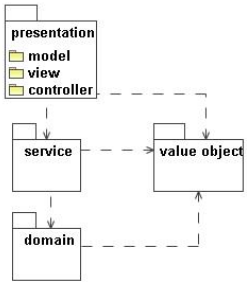


FIGURE 1. Web Application Architecture

The *presentation* package from Figure 1 corresponds to the presentation layer and includes web pages, required form models, and controllers. The *service* package corresponds to the business layer and contains business services. The *domain* package will include entities and business objects. The *value object* package includes coarse grained objects used to transfer data between *domain* and *presentation*. As Figure 1 shows, *presentation* will use only *services* and *value objects*.

In this paper we refine the basic MDA process steps presented in Figure 2 (A). This process was introduced by AndroMDA organization as its basic development approach. It is a model-driven, agile, test-driven, and iterative development process. Each iteration starts with *modeling the current iteration scenarios* (as PIM) – agile and iterative approach. Then automatic generation of *entities and services* (PSM) follows – model-driven approach. Then, *writing unit tests* for services (before implementing the service logic) – test-driven approach, *implementing the service logic*, and *running units tests* follows. *Implementing the front-end* is the last step in this development process.

The focus of this paper is on web requirements engineering, that is on the first step of the model-driven process from Figure 2 (A) – *model current iteration scenarios*. This step indicates what is needed to be build for the current iteration. Emphasizes on this step was also considered in [16]. Figure 2 (B) presents our extension to the basic model-driven approach from Figure 2 (A). Figure 2 (B) shows an activity diagram that refines the first step from Figure 2 (A).

Short descriptions of these tasks and their relationships with OpenUP [6] disciplines are given below:

- *Model use cases*. The primary focus is to obtain a detailed *use case model* for the current iteration scenarios. This step belongs to the OpenUP

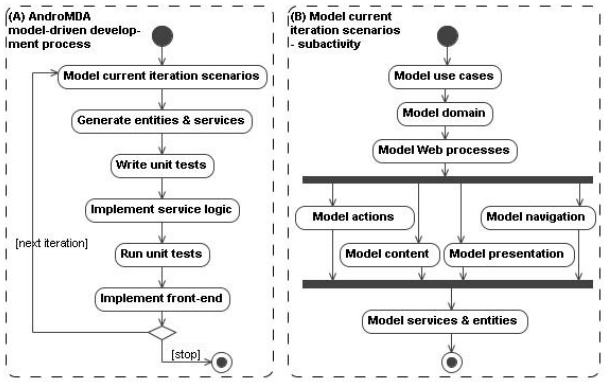


FIGURE 2. A model-driven development process

requirements discipline. The remaining steps that follows below belongs to the OpenUP analysis and design discipline.

- *Model domain.* In this step we identify the elements (classes, subsystems, etc) that collaborate together to provide the required behavior. The artifact produced after this step is a *conceptual/domain model.*
- *Model web processes.* In this step we determine how elements collaborate to realize the scenarios at high level. In order to do that we define web user interface flows by attaching *state machine diagrams* to use cases. This is an analysis step and not a design step. So, at this stage the state machines do not capture *actions* needed to be performed in order to obtain information from the system. The state diagrams express only *views* and *flows* between them.
- *Refine web processes.* The scope of this step is to detail how elements collaborate to realize the scenarios – that is, a design step. Major design decisions are made in this stage. Ideally the following steps are performed, sequentially:
  - *Model actions.* For each use case a *controller* class is attached. Now the state machines are refined by adding *action states* that captures the required system operations to be performed. The action states

will defer execution to controller operations. The controller operations will be manually implemented by developers.
  - *Model content. Presentation form model* must be discovered at this step, that is what data need to be presented in *view states*. In order to keep things simple and independent of specific frameworks we follow the basic idea from [2], that is, we do not model explicitly presentation form models. Instead we model data propagation between *action states* and *view states* within our state machine diagrams. At this stage we introduce *value objects* that will carry data from *domain* to *presentation*. Data propagation between action and view states are modeled as parameters assigned to the transition events. PIM to PSM transformation processes need to generate the required presentation form models from data propagation between states.
  - *Model navigation*. Web pages navigation will be directly generated from transitions between states. Special considerations require global transitions and menus.
  - *Model presentation*. This refinement step refers to marking event parameters sent between *action* and *view states* so that the generated code includes the required page layout and controls.
- *Model services and entities*. The connection between *presentation* layer and *service* layer is designed at this stage. The required services are designed and the relationships between each controller and required services are established. Moreover, the refinement of domain entities occurs also.

3. A New Presentation Profile for Conversational Flows

We extend AndroMDA profiles for modeling web applications [2] by adding the concept of a *conversational* scope to support the execution of use cases that span multiple use cases [17, Chapter 11]. Many web applications have use cases that do not normally fit into the request, session, and application scopes defined by the Java servlet specification. Such applications have use cases that span more than one page but do not require the longevity of the session.

Spring Web Flow framework [3] treats conversational flows as first level citizens. In this section we introduce new UML stereotypes for modeling conversational flows in UML. The UML models marked with these stereotypes will allow transformation tools to extract flow information for specific target platforms like [3]. These new stereotypes are presented below.

ConversationalFlow: This stereotype can be applied on state machines and indicates that the contained front end view states represents a single user conversation. During this user conversation the front end views can

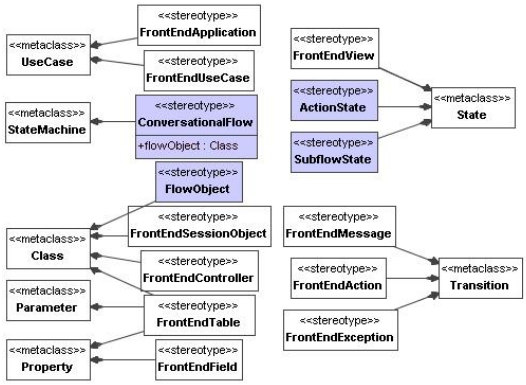


FIGURE 3. A new profile for conversational flow

share some information. The *flowObject* tag can be used to indicate a class that encapsulate the conversation shared information.

FlowObject: This stereotype is used to mark classes that represents information used in conversational flows. When transforming the PIM model into PSM models the dependency relationships between flow object classes and front end controllers classes can be used to generate convenient methods to access flow object properties within controller classes.

ActionState: This stereotype can be used to mark server side actions that belongs to conversational flows. AndroMDA profiles do not provide a stereotype for server side actions, by default a state not marked with *FrontEndView* stereotype being considered a server side action state.

SubflowState: The submachine states within a *ConversationalFlow* state machine that span a subflow that is part of the conversation will be marked using *SubflowState*.

The other stereotypes presented in Figure 3 comes from AndroMDA profiles – *ValueObject* stereotype being part of the common profile and the remaining stereotypes being part of the presentation profile. The next section refers almost all these stereotypes.

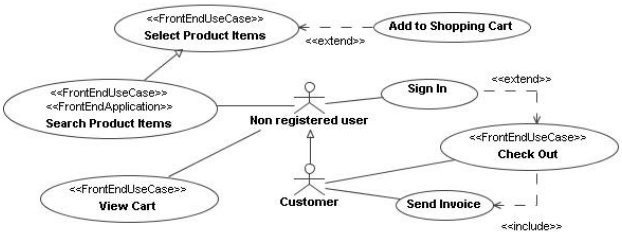


FIGURE 4. Partial use case diagram of Amazon web application

4. Modeling Example

In this section we outline the steps presented in Figure 2 (B) on Amazon[2] running example (also used as case study in [13]). The following actors interact with the application: (a) non registered user – search and select products; add products to the shopping cart, and login (b) customer - inherits from the non registered user and it is allowed (after logged-in) to start the checkout proces.

The subsections of this section correspond to the tasks presented in Figure 2 (B). Content, navigation, and presentation modeling tasks will not be discussed below, these tasks being well documented in [2]. We consider that all requirements outlined above are allocated for the iteration described in the subsections that follow. Each task includes several marking steps that indicate how to build the model in order to transform later the model into a PSM model.

4.1. **Model use cases task.** The purpose of this task is to detail the current iteration requirements, and the result is a use case model. This task includes the following marking steps [2]:

Step 1: Mark with *FrontEndView* stereotype those use cases that require user interaction.

Step 2: Mark with *FrontEndApplication* a single use case that will represent the application entry point.

Figure 4 presents a use case model for our example and the given requirements above. *Search product items*, *View cart*, and *Check out* use cases are marked with *FrontEndView*, and *Search product items* is also marked with *FrontEndApplication* stereotype.

[2]http://www.amazon.com

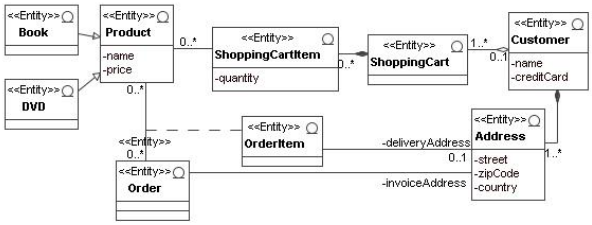


FIGURE 5. Domain model

4.2. **Model domain task.** The primary purpose of this task is to identify the concepts used to provide the required behavior. At this stage we produce a conceptual model presented in Figure 5. This is a common feature in most web modeling approaches including UWE and OO-H. The marking step of this task is [2]:

Step 3: Apply the *Entity* stereotype to all domain persistent objects.

When transforming the PIM model, the UML entities will be mapped to entities of specific object relational mapping frameworks.

4.3. **Model web processes.** The primary purpose of this task is to describe at high level the web business processes using UML 2.0 state machines. At this level of abstraction the state machines will include only the view state. The marking steps of this task are:

Step 4: For each *FrontEndUseCase* attach a state machine that describes the required views (web pages) and the navigation between them.

Step 5: Use UML submachine states in order to model navigation between use cases.

Step 6: Mark state machines with *ConversationalFlow* stereotype where appropriate. Mark where appropriate the submachine states of a *ConversationalFlow* with *SubflowState* stereotype.

This task is not indicated in [2]. In fact this is an intermediary step, that is, a requirements analysis step. The resulting state machine diagrams will be refined during the next design task.

Figure 6 and 7 presents the result of applying the steps 4–6 described above. *Checkout* web process is defined as conversational flow, that is all contained views will share some information (e.g. payment information is used when the invoice is submitted). Note different types of navigation between *Search product items* web process and: (a) *Show cart* state machine attached to the use case View cart, and (b) *Checkout* conversational flow.

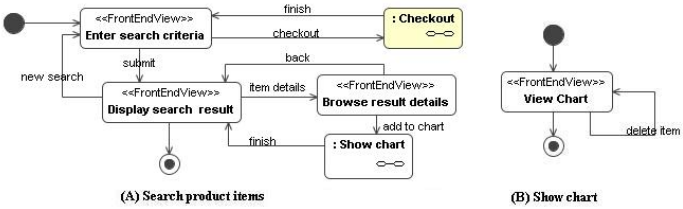


FIGURE 6. *Search product items* and *View cart* web processes

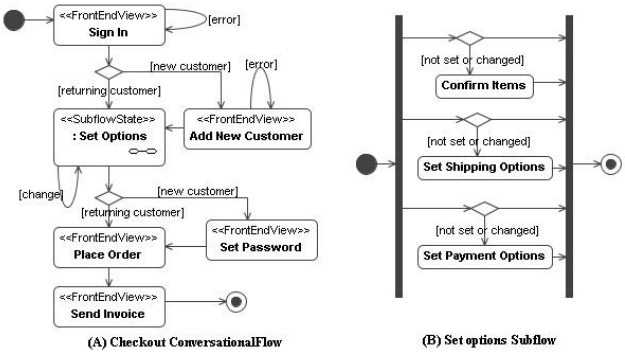


FIGURE 7. *Checkout* web process

Instead on relying on submachines for navigation, the reference documentation of AndroMDA for Struts [2] indicates the usage of labels attached to the ingoing transitions of final states.

4.4. **Refine Web processes.**

Model actions task. The purpose of this task is to determine the system operations. In order to do that we refine the web processes by adding action states corresponding to server-side action needed to be executed. Because current server-side frameworks do not follow a standard defined architecture we follow the guideline

from [2] and the actions will be encapsulated in front-end controllers. The relationships between front-end controllers and session/conversational flow objects will be generated according to the targetted platform.

Step 7: For each *FrontEndUseCase* attach a controller through a tag defined by this stereotype.

Step 8: Optionally, mark each state that represents server-side actions with *ActionState* stereotype.

Step 9: Indicate the actions that need to be executed when entering action state using deferrable trigger property of a UML state. The triggers should refer a UML CallEvent pointing to a controller operation.

Step 10: For each transition between action and view states use signal parameters to indicate the data flow between those states. At this step usually we introduce classes that captures data extracts from domain – those classes will be marked with *ValueObject* stereotype.

Step 11: Define session objects and mark their classes with *FrontEndSession*. Add dependency relationships between controllers and session object classes.

Step 12: Define objects with conversational scope and link them with conversational state machine using the tag *flowObject* (see Figure 3).

The steps 7–11 are standard steps indicated in [2]. Only the last step is new and this is needed only for the new introduced conversational flows.

5. Conclusions

The extension of AndroMDA model-driven development process with emphasizes on requirements engineering provides a simple approach close to traditional requirements engineering for desktop applications. The main advantage of using such a process consist in the ability to perform requirements engineering tasks at a high level of abstraction, independent of specific platform.

Currently we are analyzing the possibilities to extend AndroMDA profile in order to support full mapping to other frameworks such as SpringMVC, and Spring Web Flow.

References

[1] ***. Spring framework reference documentation. Technical report, 2006. http://static.springframework.org/spring/docs/2.0.x/spring-reference.pdf (06/07/06).

[2] AndroMDA. *Business Process Management for Struts Cartridge*. 2006. http://galaxy.andromda.org/docs/andromda-bpm4struts-cartridge/index.html.

[3] Keith Donald and Ervin Vervaet. *Spring WebFlow Reference Documentation*. 2006. http://static.springframework.org/spring-webflow/docs/1.0.x/spring-webflow-reference.pdf (06/07/06).

[4] María Josè Escalona and Nora Koch. Metamodeling the requirements of web systems. In *Proc. of the 2nd Int. Conf. on Web Information Systems and Technologies*, Setubal, Portugal, April 2006.

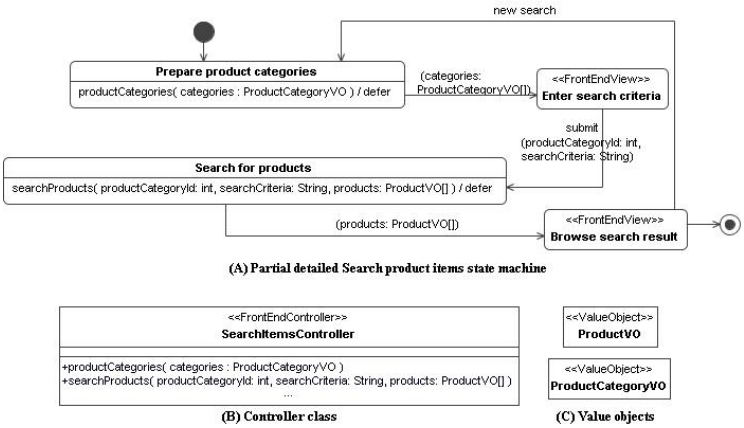


FIGURE 8. Detailed web process - action and view states

[5] Apache Foundation. *Struts Framework, Version 1.3.5.* 2006. http://struts.apache.org/1.3.5/index.html.
[6] Eclipse Process Framework. *OpenUP/Basic.* 2006. http://www.eclipse.org/epf/ (06/07/06).
[7] Jaime Gomez and Cristina Cachero. Oo-h method: Extending uml to model web interfaces. In *Information Modeling for Internet Applicaions*, pages 144–173. Idea Group Publishing, 2002.
[8] Object Management Group. *MDA Guide Version 1.0.1.* 2003. http://www.omg.org/docs/omg/03-06-01.pdf (06/07/06).
[9] Object Management Group. *MOF 2.0 Query/Views/Transformations RFP.* 2004. http://www.omg.org/cgi-bin/apps/doc?ad/02-04-10.pdf (06/07/06).
[10] Object Management Group. *UML 2.0 Superstructure.* 2004. http://www.omg.org/cgi-bin/apps/doc?formal/05-07-04.pdf (06/07/06).
[11] Alexander Knapp, Nora Koch, and Gefei Zhang. Modeling the behavior of web applications with argouwe. In *Lecture Notes in Computer Science*, pages 624–626. Springer Verlag, 2005.
[12] Nora Koch and Andreas Kraus. The expressive power of uml-based web engineering. In *Second Int. Workshop on Web-oriented Software Technology*, 2002.
[13] Nora Koch and Andreas Kraus. Integration of business processes in web application models. *Journal of Web Engineering*, 1(1):22–49, 2002.
[14] Nora Koch and Andreas Kraus. Modeling web business processes with oo-h and uwe. In *Third Int. Workshop on Web Oiented Software Technology*, 2003.
[15] Nora Koch and Andreas Kraus. Towards a common metamodel for the development of web applications. In *Second Int. Conference on Web Engineering*, pages 497–506. Springer Verlag, 2003.

[16] Nora Koch, Gefei Zhang, and Mar'ia Jos'e Escalona. Model transformations from requirements to web system design. In *Proc. of the 6th Int. Conf. on Web Engineering*, pages 281–288. ACM Press, 2006.

[17] Seth Ladd and Keith Donald. *Expert Spring MVC and Web Flows*. Apress, 2006.

E-mail address: ilazar@cs.ubbcluj.ro

E-mail address: dan@cs.ubbcluj.ro

Babeş-Bolyai University, Faculty of Mathematics and Computer Science, 400084 M. Kogalniceanu 1, Cluj-Napoca, Romania

În cel de al LI-lea an (2006) *STUDIA UNIVERSITATIS BABEŞ-BOLYAI* apare în următoarele serii:

matematică (trimestrial)
informatică (semestrial)
fizică (trimestrial)
chimie (semestrial)
geologie (trimestrial)
geografie (semestrial)
biologie (semestrial)
filosofie (semestrial)
sociologie (semestrial)
politică (anual)
efemeride (semestrial)
studii europene (trei apariții pe an)
dramatica (semestrial)
business (semestrial)
psihologie-pedagogie (anual)
ştiințe economice (semestrial)
ştiințe juridice (trimestrial)
istorie (trei apariții pe an)
filologie (trimestrial)
teologie ortodoxă (semestrial)
teologie catolică (trei apariții pe an)
teologie greco-catolică - Oradea (semestrial)
teologie catolică - Latina (anual)
teologie reformată (semestrial)
educație fizică (semestrial)

In the LI-th year of its publication (2006) *STUDIA UNIVERSITATIS BABEŞ-BOLYAI* is issued in the following series:

mathematics (quarterly)
computer science (semesterily)
physics (quarterly)
chemistry (semesterily)
geology (quarterly)
geography (semesterily)
biology (semesterily)
philosophy (semesterily)
sociology (semesterily)
politics (yearly)
ephemerides (semesterily)
European studies (three issues / year)
business (semesterily)
dramatica (semestrial)
psychology - pedagogy (yearly)
economic sciences (semesterily)
juridical sciences (quarterly)
history (three issues / year)
philology (quarterly)
orthodox theology (semesterily)
catholic theology (three issues / year)
greek-catholic theology - Varadiensis (semesterily)
catholic theology - Latina (yearly)
reformed theology (semesterily)
physical training (semesterily)

Dans sa LI-ème année (2006) *STUDIA UNIVERSITATIS BABEŞ-BOLYAI* paraît dans les séries suivantes:

mathématiques (trimestriellement)
informatiques (semestriellement)
physique (trimestriellement)
chimie (semestriellement)
géologie (trimestriellement)
géographie (semestriellement)
biologie (semestriellement)
philosophie (semestriellement)
sociologie (semestriellement)
politique (annuellement)
éphémérides (semestriellement)
études européennes (trois apparitions / année)
dramatica (semestrial)
affaires (semestriellement)
psychologie - pédagogie (annuellement)
études économiques (semestriellement)
études juridiques (trimestriellement)
histoire (trois apparitions / année)
philologie (trimestriellement)
théologie orthodoxe (semestriellement)
théologie catholique (trois apparitions / année)
théologie greco-catholique - Varadiensis (semestriellement)
théologie catholique - Latina (annuellement)
théologie réformée - (semestriellement)
éducation physique (semestriellement)