Comparing Intended and Real Usage in Web Portal: Temporal Logic and Data Mining

Jérémy Besson[1], Ieva Mitašiūnaitė[2], Audronė Lupeikienė[1], and Jean-François Boulicaut[3]

[1] Institute of Mathematics and Informatics, Vilnius, Lithuania,
[2] Faculty of Mathematics and Informatics, Vilnius University, Lithuania
[3] INSA Lyon, LIRIS CNRS UMR 5205, France

Abstract. Nowadays the software systems, including web portals, are developed from a priori assumptions about how the system will be used. However, frequently these assumptions hold only partly and are defined also partially. Therefore one must be capable to compare the a priori assumptions with the actual user behavior in order to decide how the system could be improved. To tackle this problem, we consider a promising approach to employ the same formalism to express the intended usage, the web portal model and the frequent real usage patterns, extracted from the experimental data by data mining algorithms. This allows to automate the verification whether the frequent real usage patterns satisfy the intended usage in the web portal model. We propose to use temporal logic and Kripke structure as such a common formalism.

Key words: Intended Usage, Real Usage, web Portal Model, Linear Temporal Logic, Pattern Mining

1 Introduction

Nowadays the software systems, including web portals, are developed from a priori assumptions about how the system will be used. However, frequently these assumptions hold only partly and are defined only partially. Therefore one must be capable to compare the a priori assumptions with the actual user behavior in order to decide how the system should be improved. We intend to develop a tool that helps to compare the intended usage of a web portal specified by the web portal designers, and its real (frequent) usage. This would allow to point out some design decisions that led to the development of a web portal whose real usage does not correspond to the intended usage.

We focus on corporate portals that present information about an organization, its activities, people, etc. Such a portal is a set of client-oriented web sites that personalize the portals tools and information to the specific needs and characteristics of the person who is visiting this web site, searching information that is presented directly in the web sites or in the associated databases. As a rule, it provides a standard set of tools. Corporate portals promote the gathering, sharing, and dissemination of information throughout the intranet, extranet and

Proceedings BIS 2010, Springer LNBIS 47, pp. 83-93.

Internet. However, the traditional approaches to portal design often ignore the information needs, searching practices and personalities of users. As a result, the corporate portals suffer from quality problems that prevent or inhibit their use (for example, impossibility to find information, poor navigation or inappropriate display of information). To overcome this problem, one not only need to consider the structure of the portal in terms of web pages and their links between one another, but also additional knowledge and information, e.g., what is the goal of the user while navigating through a given path of the portal and what category the user belongs to.

To allow automatic reasoning and verification on the indented usage and the real usage, the system must be represented in a formal way. A formal representation of a system called Model must check certain Properties of the System in order to reproduce its behavior under various conditions. Considerable work has been done to develop methods of constructing such models. In most of these approaches (e.g., [1], [2], [3]) properties are already known, i.e., they come from literature and/or intuition and insights of the application domain experts. Much less approaches in the domain of system modeling consider the issue of finding out the Properties of the System, necessary to build a Model. Generally, knowledge about the System is acquired by performing the Experiments on it, and thereby it is the Experiments that inform on Properties. We suggest that there is a place here for Data Mining - an art of finding knowledge (e.g., patterns and Properties) from data (e.g., experimental data and experiments).

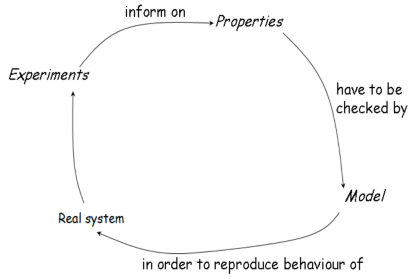

Fig. 1. Model - Properties - Experiments

We consider that the integration of the two issues, i.e., "Properties have to be checked by the Model in order to reproduce the behavior of the real System" and "Experiments inform on Properties" into one cycle of "Experiments or Experimental data inform on Properties that have to be checked by the Model in order to reproduce the behavior of the real System" (see Figure 1) is a promising research goal. This goal can be achieved by employing the same formalism to express the properties, extracted from experimental data by data mining algorithms, and to reason on/verify/learn models using these properties.

In our context, three different elements must be considered: the model, real usage (Experiments) and intended usage (Properties). The model represents the possible usages (behavior) of the system and can be seen as a set of local transitions between states of the user browsing. A real usage is a sequence of transitions in the model. An intended usage is a global constraint on the possible sequences of transitions in the real system. In this context, we seek to be able to express a web portal in terms of both local transitions and global behavior (intended usage), and then to check if they are compatible, i.e., if the local transitions lead to the intended global behavior. Intended usage are here considered as requirements and must be satisfied by the real usage. If it is not the case, then some design decisions was spurious and must be modified, in order to converge toward a web portal model that exhibits the needed global behavior.

We propose the use of linear temporal logic (LTL) and Kripke structure as such a common formalism. It is widely used in many application domains and well adapted to express dynamic properties of non-deterministic finite transition systems. Examples of interesting properties of real systems, such as attainability of a state or property, inevitability of a property, the invariants, chaining of properties can be easily expressed in temporal logic.

The rest of the paper is organized as following. Section 2 presents the Linear Temporal Logic and Kripke structure formalisms that are employed in this article. Section 3 presents the method of modeling a web portal, its intended and real usages. Afterwards, in Section 4 we describe how intended and real usage are compared. Finally, Section 5 concludes the work.

2 LTL Over a Kripke Structure as a Common Formalism

Temporal logic extends propositional logic (used for describing the states) with operators for reasoning over time and non-determinism. Systems of temporal logic are classified along a number of axes: propositional versus first order, branching versus linear, points versus intervals, global versus compositional, and past versus future tense [4]. Temporal logic was shown to be a powerful formalism for expressing the dynamic properties of a system [5, 6], thus we consider LTL (Linear Temporal Logic) as a convenient formalism to express intended usage.

Several temporal operators are introduced in LTL: X ϕ means ϕ is true at next transition, G ϕ means ϕ is always true, F ϕ means ϕ finally true, and ϕ U ψ means ϕ is always true until ψ becomes true. A LTL formula is true if every path of the Kripke structure, starting from a starting node, satisfies the LTL formula.

With LTL one can express the interesting properties, such as reachability, safety, mutual exclusion and responsiveness, thus it can typically be used to express intended usage. In the following we give several examples. A state is called reachable if there is a computation path from an initial state leading to a state that satisfies a propositional expression P, what can be written in LTL as $F\ P$. A system is called safe if one cannot reach a state for which an undesirable

property holds ($!P$), what can be written as $G\ !P$. Mutual exclusion means that a system is not allowed to hold two (or more) critical properties (P_1 and P_2), what can be written in LTL as $G\ !(P_1\ \wedge\ P_2)$. A system is called responsive if, when a property $P1$ holds, then there will be a given answer $P2$, what can be written in LTL as $G(P_1 \rightarrow F(P_2))$. To express that P_1 must remain true until the answer P_2 is given, one can use the formula $G(P_1\ \rightarrow\ P_1\ \bigcup\ P_2)$. The formula $G(P_1\ \rightarrow\ (P_1 \wedge !P_2) \bigcup (!P_1\ \wedge\ P_2))$ means that P_1 and P_2 must be exclusive once P_1 holds.

Semantics of LTL formulas can be implemented on a Kripke Structure. A Kripke structure is a finite-state automaton. A set of discrete variables are defined (Boolean, enumeration, bounded integer [a... b]), such that an instantiation of all the variables defines a state in the automaton. Then the transitions between the variable instantiations (states) are defined. The semantics of the Kripke structure is given by the set of its trajectories (also named paths or executions), i.e., the set of sequences of states of the Kripke structure. For example, two discrete variables named "B" and "E" can be defined in a Kripke structure where "B" is a Boolean variable and "E" is an enumeration variable that can take one of the two values: either "e1" or "e2". Three states (S1, S2 and S3) may be defined with the following instantiations: S1= (B=true,E=e1), S2 = (B=false,E=e1) and S3 = (B=false,E=e2). Transitions in the Kripke structure can be defined, e.g., from S1 to S2, from S2 to S3 and from S3 to S3. This Kripke structure defines the model of the system for which one can check its dynamic properties.

Interestingly, with such a formalism, we can at the same time (a) model a (web portal) system with a set of discrete variables and a finite-state automaton and (b) define LTL constraints over this model. For example, one can model a web portal as a Kripke structure, where the set of states is the set of all tuples of values denoting the set of characteristics of a web portal navigation. Then, intended usage and real usage can be expressed with LTL formulas.

Here are examples of a priori interesting system properties that can be defined in LTL over a Kripke structure:

- A user is of the category X;
- In the initial state, every user of the category X has some information, but the users goal (i. e., what information he/she needs) is unknown;
- Each user of the category X has a goal to find definitions of some terms;
- Any reachable (from initial) state, in which the property p_1 is TRUE, has a path from it on which the property p_2 is eventually TRUE, and until the property p_3 is TRUE;
- Any portal state should always be characterized as follows: no links to already deleted documents.

3 Modeling a Web Portal to Verify Intended and Real Usages

We seek to propose a tool that can answer the question: Are the frequent usages of a web portal coherent w.r.t. the web portal model and the intended usage? For doing that, we propose to model the intended usage, the real usage and the web portal model with the same formalism. If succeeded, one could specify and compare the intended and real usage in a unified way. Thereby, our approach is qualitatively different from the ad-hoc methods that glue different applications solving distinct parts of the problem. We also underline that a web portal analysis is only one of the possible applications of the proposed method, and thus we seek genericity.

In the process mining research domain, important work was realized on discovery, conformance and extension of the models w.r.t. experimental data. The main addressed problems are the following: "How to discover a model from the real usage data?", "Is the model conform w.r.t. the real usage data?" and "How to extend the model w.r.t. to the real usage data?". In these questions, two entities are considered: the model and the real usage data. We seek to add another entity in our analysis: the intended usage. In our context, the model represents all the possible usages of the system. The system-usage data is the concrete real usage of the system (i.e., how the system behaves). The intended usage states how the system should be used (i.e., what the expected behavior is).

We model a web portal as a set of discrete variables whose instantiations form the states of the browsing process, and the possible transitions between these states. Indeed, employing such a formalism, the model can be transformed into a Kripke structure on which LTL formulas that represent intended and real usage can be evaluated. The intended usage represents the correct usages of the system, among the possible usages (where all possible usages make a web portal model). By definition, the web portal cannot restrict the usage of the web portal to only those of the intended usages.

Summing up, to compare intended and real usage, we propose the following methodology:

- To specify a web portal model (see Section 3.1)
- To specify intended usage (see Section 3.2)
- To extract frequent usage employing frequent sequence pattern mining algorithms (see Section 3.3)
- To transform the web portal model, intended usage and frequent usage patterns into a Kripke structure and LTL formulas, and finally to apply a symbolic model checker to verify the intended and real usage w.r.t. to a web portal model (see Section 4)

3.1 Web Portal Model

To express the possible usages of a web portal, we propose to model web portals by the means of a Kripke structure, i.e., a finite-state transition system. Kripke

structure contains discrete variables whose instantiations form the states of the browsing process. Among these variables we distinguish a principal one that represents the current web page browsed by the user. However we need to consider not only the web portal structure in terms of web pages and their links between each other, but also additional information, such as the goal of the navigation through a given path and his/her category. Other examples are period of the day, day of the week, quality of services, and meta-data on the browsed pages. All this additional information can be taken into account through additional discrete variables in the Kripke structure.

Following are several examples of web portal possible usages that we seek to represent:

- Ex1: From the page "P1", a user can go to pages "P2", "P3" and "P4".
- Ex2: When a user arrives to the web page "P1", he/she should have the goal named "G".
- Ex3: When a user sends an email and has the goal named "G", then it is probable that the goal "G" is reached.
- Ex4: When a user of the category "C" goes to the page "P1", he/she has the goal "G2".
- Ex5: The web pages "P1" and "P2" are pages of the same category "C1"

Interestingly, all these usages can be defined in a Kripke structure. More generally, any information that can be expressed by the means of discrete variables can be included in such a model.

Example 1. Figure 2 presents a web portal model with four web pages (P1, P2, P3 and P4) and two user's goals (G1 and G2) that can become new or reached for some web pages (G1 is new in P1 and reached in P4 and G2 is new in P2 and reached in P3). This web portal model can be formalized by a Kripke structure with three state variables: $page \in \{P1, P2, P3, P4, P5\}$, $G1 \in \{True, False\}$ and $G2 \in \{True, False\}$ where "True" means that the user has this goal and "False" means that the goal is reached or that the user does not have this goal. The Kripke structure is defined as following:

- start($page$) = P1, P2, P3, P4 or P5
- start($G1$) = False
- start($G2$) = False
- next($page$) = (P2 or P3) if $page = P1$, P4 if ($page = P2$ or $page = P3$), $P2$ if $page = P4$ or $P5$ if $page = P2$
- next($G1$) = True if $page = P1$, False if $page = P4$ else $G1$
- next($G2$) = True if $page = P2$, False if $page = P3$ else $G2$

The "start" function gives the possible initial values of a state variable and the "next" function provides the possible values of a state variable for the next Kripke state.

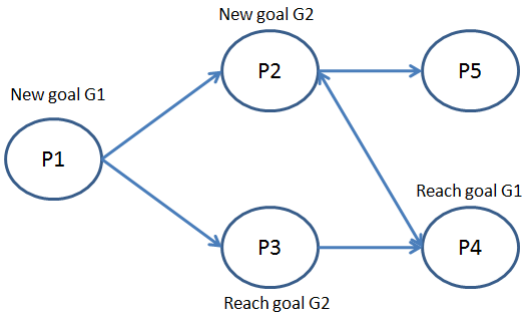

Fig. 2. Example of a web portal model

3.2 Intended Usage

To express the intended usage, we use LTL formulas over the Kripke structure, i.e., the web portal model. In general, because of design decisions and incompatible choices to fulfill the requirements and specifications, a model contains navigational paths that do not satisfy the intended usage. Thus by intended usage we mean the constraints that must be satisfied by the frequent usage of the portal.

Activity diagrams and use cases are documents commonly written while specifying a software system. They can be used as the basis of the intended usage. Activity diagrams are used to model the behaviors of a system and the way in which these behaviors are related in an overall flow of the system. Activity diagrams are similar to flowcharts, but they allow the specification of concurrent parallel processing paths. Activity diagrams are used to model the behavior of a system and the way in which these behaviors are related in an overall flow of the system, while state machine diagrams illustrate how an element can move between states classifying its behavior, according to transition triggers, constraining guards, and so on.

In addition, behavioral constraints can be defined to specify the acceptable behavior without considering the specific web portal. For example, each new goal must be reachable, user can have only a maximum given number of goals to reach, user should not go too many times to the same web page and any browsing should contain a reached goal.

Example 2. Referring to Example 1, following are examples of use cases:

- Use case "UC1": User has the new goal G1, then has the new goal G2 then reaches the goal G2 and finally reaches the goal G1
- Use case "UC2": User has the new goal G1, then reaches the goal G1 and then goes to the web page P5

One can define an intended usage in LTL that states that any browsing should follow one of the two use cases "UC1" or "UC2": $F(G1 = True \wedge F(G2 = True \wedge$

$F(G2 = False \wedge F(G1 = False)))) \vee F(G1 = True \wedge F(G1 = False \wedge F(page = P5)))$.

3.3 Frequent Usage Patterns Discovery Through Frequent Sequential Pattern Mining

To extract frequent usage patterns from event files (log files), one can use sequential pattern mining algorithms. Indeed, event files can be transformed into sets of sequences of nominal values. Each session can be seen as an ordered sequence of pages (nominal values) that was browsed by the user during its session. Sequences of nominal values can be mined to extract patterns of different pattern languages. Portal usages patterns can be revealed by extracting, e.g., episode rules [7], frequent sub-strings (e.g., [10] and [12]) and frequent sub-sequences ([8]). For example, a pattern "20% of users that reached the page P1 also go to the page P2" can be used as a frequent real usage and compared with the intended usage.

We propose to express the extracted frequent usage patterns using LTL. In the following we provide examples of different sequential patterns and their corresponding LTL formulas. We consider that the event file being mined contains three sequences {abaabcb, baacba, cbaa}. In the LTL formulas we use the notation S[1...n] that represents a pattern of n symbols where S[i] is its i-th symbol.

Frequent sub-string:

- Pattern to extract: all the strings (consecutive elements) that appear in at least X% of the input sequences
- Examples: "a", "b", "c", "aa", "baa" and "cb" appear in 100% of the input sequences. "cba" appears in 2/3 of the input sequences.
- LTL formula of a frequent sub-string S[1...n]: F(page=S[1] $\wedge$ X(page=S[2] $\wedge$ X(page=S[3] $\wedge$ X(... $\wedge$ X(page=S[n])))))).

Frequent sub-sequence:

- Pattern to extract: all the sequences (not necessary successive elements) that appear in at least X% of the input sequences and such that the consecutive elements of the extracted sequence appear in the input sequences within a window of maximal size Y
- Example: "ca" appears in 2/3 of the input sequences with a maximal windows size of 2.
- LTL formula of a frequent sub-sequence S[1...n]: F(page=S[1] $\wedge$ X F(page=S[2] $\wedge$ X F(page=S[3] $\wedge$ X F(... $\wedge$ X F(page=S[n])))))

Episode rules:

- Pattern to extract: all the couples of sequences (S1,S2) such that if the sequence S1 is present then S2 is present afterwards with a confidence of at least X% and it appears in at least Y% of input sequences

– Example: When "baa" is present then later there is "cb" with a confidence of at least 2/3 and it appears in at least 2/3 of input sequences
– LTL formula of an episode rule pattern (S1[1...n],S2[1...n]): G (page=S1[1] ∧ X (page = S1[2] ∧ ... ∧ page = S1[n] ∧ X F(page = S2[1] ∧ X (page = S2[2] ∧ ... ∧ X(page = S2[m]))))) ∨ ! (page=S1[1] ∧ X (page=S1[2] ∧ ... ∧ page=S1[n])))

Example 3. Referring to Example 1, following are examples of user browsing that can be obtained from the log file of the web portal:

– P1, P3, P4, P2, P5
– P1, P2, P4, P2
– P1, P2, P5
– P1, P2, P5

$\{P1, P4, P2\}$ and $\{P1, P2, P5\}$ are examples of frequent usage patterns in user browsing with a frequency threshold of 50% that can be transformed into LTL formulas: $F(page = P1 \wedge XF(page = P4 \wedge X(page = P2)))$ and $F(page = P1 \wedge X(page = P2 \wedge X(page = P5)))$.

4 Comparing Real Usage and Intended Usage

To verify the extracted frequent usage patterns w.r.t. the intended usage from a web portal model, we check if it exists a path in the Kripke structure that satisfies all the intended usages and the LTL formulas obtained from the frequent usage patterns. Thereby one can verify if the frequent usage patterns are coherent w.r.t. the intended usage.

Let C1, C2, ..., Cn be the LTL formulas specifying the intended usage, and Cp be the LTL formula obtained from a frequent usage pattern. To verify the intended usage w.r.t. the frequent usage we generate the following LTL formula "! (C1 ∧ C2 ∧ ... ∧ Cn ∧ Cp)". It is important to notice that a LTL formula is true if it is true in all initial states and for all the paths of the Kripke structure. But our objective is to check only if it exists one path that satisfies the LTL formula. Note that a LTL formula A holds on some computation path if and only if it is not true that formula !A holds on all computation paths. Vice versa, A holds on all computation paths if and only if it is not true that formula !A holds on some computation path. Thus we will check if "! (C1∧ C2 ∧ ... ∧ Cn ∧ Cp)" is false, i.e., if it is not true that the formula is true for all paths, what is equivalent to check if it exists a path such that the formula (C1 ∧ C2 ∧ ... ∧ Cn ∧ Cp) is true.

This LTL formula, generated from the intended usage and frequent usage, expresses the following "Does it exist a path in the Kripke structure model of a web portal that satisfies the intended usage and that follows the frequent usage pattern?". To be more concrete, we do not want to check if it exists a real usage (extracted from the event file) that satisfies the generated LTL formula, because the paths that satisfy the LTL formula may not be present in the event

files. On the contrary, we want to check whether a frequent usage pattern leads to coherent or incoherent paths w.r.t. the intended usage. A frequent usage pattern is incoherent w.r.t. the intended usage, if no path can follow the frequent usage pattern while satisfying the intended usage. The frequent usage pattern is incoherent w.r.t. the intended usage, if the generated LTL formula is true. To check whether the generated LTL formula is true over the Kripke structure model of a web portal we use the NuSMV symbolic model checker.

Example 4. Referring to the web portal model of Example 1, the intended usage of Example 2 ($F(G1 = True \wedge F(G2 = True \wedge F(G2 = False \wedge F(G1 = False))))) \vee F(G1 = True \wedge F(G1 = False \wedge F(page = P5))))$ and of the frequent usage patterns of Example 3 ($F(page = P1 \wedge XF(page = P4 \wedge X(page = P2)))$ and $F(page = P1 \wedge X(page = P2 \wedge X(page = P5))))$, one can check if the real usage and the intended usage are compatible. Using our method, we can verify that the frequent usage pattern$F(page = P1 \wedge XF(page = P4 \wedge X(page = P2)))$ satisfies the intended usage. Indeed, the path (P1, P2, P4, P2, P5) follows the frequent usage pattern and satisfies the intended usage. The pattern $F(page = P1 \wedge X(page = P2 \wedge X(page = P5)))$ does not satisfy the intended usage. Indeed, it cannot reach any goal and then does not follow neither "UC1" nor "UC2".

5 The Whole Process

Figure 3 presents a global picture of our tool for comparing real usage and intended usage for a web portal. The web portal model is specified by the web portal designer/administrator. Then the intended usage is defined. Afterwards, the event file of the web portal is retrieved and pre-processed (session detection, transformation of pages into categories, removing robots, format transformation, etc). Frequent patterns are extracted from the event file and transformed into LTL formulas. Then, we use the Nusmv symbolic model checker to verify if a real usage fits with the intended usage of a web portal model. To accomplish this, we transform the web portal model, the intended usages and the extracted frequent usage patterns into a NuSMV files (input format of Nusmv).

6 Conclusion

We present a method to compare intended usage and real usage in web portal employing the same unifying formalism: LTL and Kripke structure. We also employ Data Mining techniques to extract frequent usage patterns from event files (system usage data). This method is valid not only for web portal re-engineering but also for the analysis of other software systems.

Acknowledgment: This work is funded by the Lithuanian State Science and Studies Foundation under the grand V-09059 and by the French-Lithuanian bilateral program EGIDE PHC "Gilibert" under the grant 19976RM

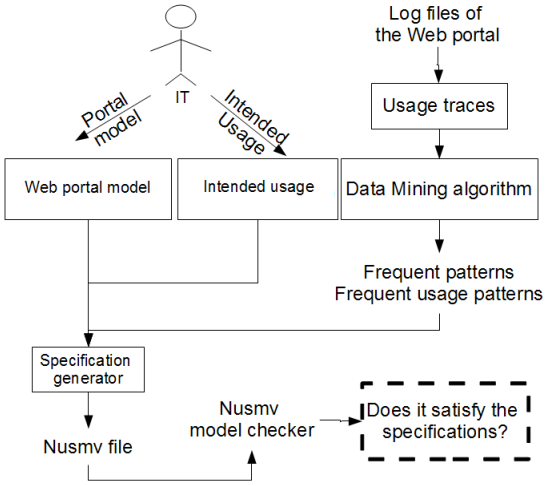

Fig. 3. Comparing intended usage and real usage

References

1. Alexander Bockmayr and Arnaud Courtois, Using Hybrid Concurrent Constraint Programming to Model Dynamic Biological Systems, ICLP '02: Proceedings of the 18th International Conference on Logic Programming, pages 85-99, Springer-Verlag, 2002
2. Aviv Regev, William Silverman and Ehud Shapiro, Representation and simulation of biochemical processes using the pi-calculus process algebra, Proceedings of the 6th Pacific Symposium on Biocomputing, pages 459-470, 2001
3. Gilles Bernot, Jean-Paul Comet, Adrien Richard and Janine Guespin, Application of formal methods to biological regulatory networks: extending Thomas' asynchronous logical approach with temporal logic, Journal of theoretical biology, pages 339-347(229-3), 2004
4. E.A. Emerson, Temporal and Modal Logic, Formal Models and Semantics, pages 995-1072, North-Holland Pub. Co./MIT Press, 1990
5. A. Pnueli, System specification and refinement in temporal logic, Proceedings of Foundations of Software Technology and Theoretical Computer Science, pages 1-38, Springer, 1992
6. C. Dixon, M. Fisher, B. Konev, and A. Lisitsa, Efficient first-order temporal logic for infinite-state systems, Computing Research Repository, Cornell University, 2007
7. Nicolas Méger and Christophe Rigotti, Constraint-Based Mining of Episode Rules and Optimal Window Sizes, PKDD, pages 313-324, 2004
8. Agrawal, Rakesh and Srikant, Ramakrishnan, Mining Sequential Patterns, ICDE '95: Proceedings of the Eleventh International Conference on Data Engineering, pages 3-14, IEEE Computer Society, 1995
9. S. Dan Lee and L. De Raedt, An Efficient Algorithm for Mining String Databases Under Constraints, Proceedings KDID'04, page 108-129, Springer-Verlag, 2004
10. L. De Raedt, M. Jaeger, S. Dan Lee and H. Mannila, A Theory of Inductive Query Answering, Proceedings IEEE ICDM'02, page 123-130, 2002

11. H. Mannila and H. Toivonen, Levelwise search and borders of theories in knowledge discovery, Data Mining and Knowledge Discovery, page 241-258 (1-3), Kluwer Academic Publishers, 1997
12. S. Dan Lee and L. De Raedt, An Efficient Algorithm for Mining String Databases Under Constraints, Proceedings KDID'04, page 108-129, Springer-Verlag, 2004
13. H. Albert-Lorincz and J.-F. Boulicaut, Mining frequent sequential patterns under regular expressions: a highly adaptative strategy for pushing constraints, Proceedings 3rd SIAM SDM'03, page 316-320, 2003