PLOTS OF NARRATIVES OVER TEMPORAL DATABASES

Antonio L. Furtado, Angelo E. M. Ciarlini

E-mail: {furtado,angelo}@inf.puc-rio.br
PUCRioInfMCC 6/97

Abstract- An operation-based notion of plots of narratives is introduced, to summarize events happening to objects in the temporal database of an information system. The method proposed for modelling and handling plots, involving the application of a Plan-recognition and Plan-generation facility, is shown to be useful for various purposes, including problem-detection and decision support. An interdisciplinary approach is adopted, utilizing results from studies on literary theories.

Keywords- Object-Oriented Temporal Databases, Plan Recognition and Generation, Literary Theories, Plots of Narratives.

Resumo- Uma noção operacional de enredos de narrativas é introduzida, para resumir eventos ocorridos com objetos representados no banco de dados temporal de um sistema de informações. O método proposto para modelar e manipular enredos, o qual envolve uma facilidade para reconhecer e gerar planos, é comprovadamente útil para várias finalidades, inclusive deteção de problemas e apoio à decisão. Uma abordagem interdisciplinar é adotada, utilizando resultados de estudos em teorias literárias.

Plavras-chave- Bancos de Dados Temporais com Orientação de Objetos, Reconhecimento e Geração de Planos, Teorias Literárias, Enredos de Narrativas.

1. Introduction

The concept of *narrative* has been treated in depth along the years by scholars working on the field of *literary theories* [Ge80]. However, the concept also stands out as a topic of major interest for information systems research, once we visualize a database -- especially an *object-oriented temporal database* -- as a medium to register, albeit in a compact and rigidly formatted way, *events* happening to objects in the mini-world of interest. When expressed under the usual form of a natural language text, the narrative of an event appears far richer in terms of semantics and pragmatics. But it is possible to extract from the database a faithful *plot*, summarizing the essence of the full-fledged textual narrative.

We propose to examine in the present paper how a database application can be designed and used to function as a repository of events, based on the notion of plots of narratives. Our method involves the definition of application-oriented operations and applies a *Plan-recognition and Plan-generation facility* (Pr/Pg), which is shown to be useful for various purposes, including problem-detection and decision support. Section 2 outlines the method, and its use is illustrated through a very simple example in section 3. Section 4 briefly refers to literary studies, especially by the Russian scholar V. Propp, which have motivated our work. Section 5 indicates the status of the project and aspects for future investigation.

2. Defining and utilizing plots

Objects in a database can be viewed *statically*, in terms of their properties, or *dynamically*, by considering the events in which they are involved. The dynamic view is naturally associated with object-oriented temporal database organizations. As part of the database design, it is convenient to define a number of application-oriented *operations* as functions between states. The definitions give their pre- and post-conditions, in a style consistent with the STRIPS formalism [FN71]. Both the pre- and post-conditions are logical expressions involving *predicates* expressing properties of objects. States are in turn characterized by the set of predicates (more precisely, *predicate instances*) holding at a given time. Intuitively, pre-conditions impose restrictions on the execution of an operation, whereas the difference between pre- and post-conditions indicates the effects of the operation, i.e. how the current state s_i is transformed into a new state s_j by negating and/or affirming certain predicates. The state-changes must be such that no static or dynamic integrity constraint can be violated.

Without going into detail about the choice of temporal data structures [OS95], we stress that they should allow to query not only the predicate instances but also the executions of operations. One possibility is that, in addition to the database itself, a log be maintained recording the entire sequence of executions of operations with the respective times of execution. Another option is to reconstitute this sequence from the temporal records of the predicate instances, possibly with the help of the definition of the operations. In [FC90], we have adopted an *update-oriented organization* for temporal databases, where the executions of operations are directly represented as time-stamped relational tuples.

We should then be in a position to query what happened to a given object (or group of objects) within a certain time interval. The answer would simply consist of a subsequence of the operations executed. If expanded in natural language, the answer would assume instead the aspect of a textual *narrative*. In keeping with the literary terminology, we shall say that what a sequence provides by itself is a *plot* of the narrative.

A study of plots thus obtained will normally reveal that such "stories" are not fortuitous. They are brought about by agents who are intent on reaching relevant *goals*, denoted, as before, by logical expressions involving predicates. So a plot can also be seen as a *plan*, in which the overall effect of executing the constituent operations is to lead from an initial state s_0, passing by a number of intermediate states, to a final state s_f where the intended goal holds. Plans follow a partial order, because pre-conditions of some operations may need to be established as part of the post-conditions of other operations which, therefore, must be executed first.

A more detailed study will often expose the interplay of *cooperative* and *competitive* goals inside the same plot, especially when more than one agent is involved with the execution of the constituent operations. One way to treat this multiplicity is to regard the subsequences of operations whose initiative belongs to different agents as separate mutually interfering plans. If we prefer to take the plot as a single plan, we may concentrate on the viewpoint of one agent, such as the (database-owner) corporation itself.

After a database has been put to use, a vital *knowledge discovery* [MCP93] opportunity is offered by the study of plots: to create a *library of typical plans* employed by users to reach successfully (and also, optionally, without success) their identifiable goals. Several techniques are applicable for combining analogous plots to build the patterns of the typical plans, including most specific generalization [Fu92]. Since, being recursive structures, sequences are composed of subsequences which are sequences on their own right and can eventually consist of a single operation, the distinction between operation and plan is blurred and each typical plan in the library can be regarded as a (complex) operation, with pre- and post-conditions compatible with its goal as a plan. The library is consequently organized hierarchically with **part-of** links, where the plans at the bottom correspond to the initially defined simple operations. In addition, **is-a** links are established whenever a plan is introduced as a generalization of two or more plans aiming at the same goal. The library is rooted at an "end" node, to which are connected by **is-a** links those plans (to be called *end-plans* or *end-operations*) encompassing in its entirety some activity of interest to users.

The pre-conditions and post-conditions of complex operations can be defined in terms of those of their constituent simple operations. An obvious preliminary requirement for the correct specification of each complex operation is that there must exist some reachable database state starting from which every one of its constituent operations be in turn applicable, there being no need to interpolate additional constituent operations to establish pre-conditions at intermediate states of the execution. If this requirement were not satisfied, the specification of the complex operation would be incomplete (i.e. the operation would not be executable as a *transaction* in the usual database sense); operations obtained from plots effectively executed at some occasion should normally meet the requirement.

For complex operations introduced by composition (**part-of** links), we first assume, to simplify the explanation, a total rather than a partial order. The post-conditions are then expressed by "C_1 & ~ C_2", where C_1 and C_2 consist of conjunctions of predicates: C_1 comprises the predicates holding at the final state s_f which did not already hold at the initial state s_0, whereas C_2 comprises the predicates which held at s_0 and no longer hold at s_f. Both C_1 and C_2 combine the effects of each of the constituent operations that are not transitory (i.e. limited to intermediate states, being undone by subsequent constituent operations). The pre-conditions are expressions of the form P_1 & Q_2 & ... & Q_n, where P_1 denotes exactly the pre-conditions of the first operation in the sequence, and the Q_i are conjunctions of the (affirmed or negated) predicate instances in the pre-conditions of each of the other operations that are not already present in the post-conditions of the sub-sequence of the i - 1 previously executed operations. Whenever the composition is only partially ordered, both the pre- and post-conditions will be disjunctions of the expressions corresponding to each possible sequence.

For complex operations formed by a generalization (via **is-a** links) of n simpler operations, both pre- and post-conditions are expressions of the form "P_1 & (P_2 | P_3 | ... | P_n)", where P_1 factors out all asserted and negated predicates common to the expressions of the n operations, whereas P_2, ..., P_n contains those exclusively present in each operation.

However a more crucial feature of operations -- especially simple or complex end-operations -- is what motivates their execution, namely their goals. The goal of an operation may coincide exactly with its post-conditions or, more generally, be a logical consequence of these post-conditions. In the case of operations formed by generalization, only part P_1 of the post-conditions needs to be considered, since it expresses the effects that will be achieved regardless of which alternative specialized operation is executed.

To take full advantage of the library of typical plans, a Pr/Pg facility must be available. Our present prototype integrates the plan-recognition algorithm proposed by Kautz [Ka91] with the Abtweak algorithm [Wo91] for plan-generation. By matching a few *observations* (concerning operations executed thus far involving a given object or group of objects) against the library, the plan-recognition component can anticipate what plan is being tried. If the plan is conducive to failure or if, even being successful in normal circumstances, there exist obstacles in the current state, the plan-generation component may propose either to *adapt* the recognized library plan (in the spirit of the "reuse" concept from software engineering [GHJV95]) or to *generate* an entirely new plan able to reach the intended goal. New well-accepted plans are candidates for later inclusion in the library. These uses of the facility are geared to provide *assistance* [CF90, GGM92] to users in terms of problem prevention and decision support, but other purposes can be served as well, such as detecting illegal or undesirable actions or attempts. Checking observations against the library can be done occasionally in a manual fashion, or systematically in active database environments.

3. A simple example

Consider, as an overly simplified example, the temporal database of Company Alpha, containing information about **employees** of Alpha - who are hired, trained, have their salary levels raised, are assigned to a client's account, replaced in the client's account by another employee, and fired; and information about **clients** - which are companies that may open an account with Alpha and, if not satisfied, complain about the service. Companies and employees are thus among the objects represented. Company Alpha's managers and employees also appear as users of the database, with different levels of access and update authorization. An employee being hired is an event involving the employee and company Alpha. As explained in the previous section, we shall characterize events by operations able to cause their occurrence; for the event just mentioned, this would be the hire operation.

The operations must be defined in such a way that they preserve the established integrity constraints. These prescribe that "employees serving clients" is a one-to-one relationship, and that no employee serving a client can be fired. Employees become eligible to a higher salary level by improving their skills through training programs, but only if they are currently serving a client in a way that the latter finds satisfactory. The definitions follow:

open(C)
    pre-conditions: company(C) &
        ~ is-client(C)
    post-conditions: is-client(C)

hire(E)
    pre-conditions: person(E) &
        ~ is-employee(E)
    post-conditions: is-employee(E) &
        has-level(E,1)

fire(E)
    pre-conditions: is-employee(E) &
        ~ ∃C serves(E,C)
    post-conditions: ~ is-employee(E) &
        ~ ∃L has-level(E,L)

train(E,T)
    pre-conditions: is-employee(E) &

```
                          course(T) &
                          ~ took-course(E,T)
      post-conditions:  took-course(E,T) &
                          ~ ∃C dissatisfied(C,E)

raise-level(E,)
      pre-conditions:   ∃L has-level(E,L) &
                          ∃C serves(E,C) &
                          ~ dissatisfied(C,E) &
                          ∃T took-course(E,T) &
                          ~ credited(E,T)
      post-conditions:  ~ has-level(E,L) &
                          has-level(E,L + 1) &
                          credited(E,T)

assign(E,C)
      pre-conditions:   is-employee(E) &
                          is-client(C) &
                          ~ ∃C1 serves(E,C1) &
                          ~ ∃E1 serves(E1,C)
      post-conditions:  serves(E,C)

complain(C,E)
      pre-conditions:   serves(E,C)
      post-conditions:  dissatisfied(C,E)

replace(E1,E2,C)
      pre-conditions:   serves(E1,C) &
                          is-employee(E2) &
                          ~ ∃C1 serves(E2,C1)
      post-conditions:  ~ serves(E1,C) &
                          serves(E2,C) &
                          ~ dissatisfied(C,E1)
```

Suppose now that, besides others, the following events have occurred:

"Beta became a client of Alpha. John was hired, and then assigned to the Beta account. Later, Beta complained about the service. John participated in training program c135. John's level was raised"

Suppose further that it is possible, somehow, to ascertain from the information stored in the database that the plot consisting of the following sequence of operations has been executed:

Plot 1:

a. open(client: Beta)
b. hire(employee: John)
c. assign(employee: John, client: Beta)
d. complain(client: Beta, employee:John)
e. train(employee: John, course: c135)
f. raise-level(employee: John)

One will readily verify that **Plot 1** provides a compact summary of the narrative presented before as a natural language text.

Now consider another plot, which clearly corresponds to a different narrative:

Plot 2:

a. <u>open</u>(client: Delta)
b. <u>hire</u>(employee: Robert)
c. <u>assign</u>(employee: Robert, client: Delta)
d. <u>complain</u>(client: Delta, employee(Robert))
e. <u>hire</u>(employee: Laura)
f. <u>replace</u>(employee: Robert, employee: Laura, client: Delta)
g. <u>fire</u>(employee: Robert)

Upon close examination, **Plot 2** is seen to have much in common with **Plot 1**. From Alpha's viewpoint, both are about reacting to the complaint of a client. In **Plot 1**, John, who served company Beta, took course c135, whereas, in **Plot 2**, the employee in charge, Robert, was replaced by a new employee, Laura, and then fired. An event of interest from an employee's point of view was that John had his level raised after taking a course.

The study of the temporal database may reveal that the main goals of company Alpha are to obtain clients and to improve the service offered to them in case they communicate their dissatisfaction; as to employees, their goals are related with stability in their jobs and level (which implies salary) increase. Typical plans to reach these goals may be derived from plots successfully leading to them, extracted from the database. In particular, suppose that, among others, a significant number of plots similar to **Plot 1** and **Plot 2** are extracted and that five typical plans are finally obtained; as argued in the previous section, they can be regarded as complex operations, which may be appropriately named and then added to the library of typical plans. Figure 1 below shows the resulting structure of the library, following the conventions in [Ka91]. Notice the five end-operations (one of which is the initially defined <u>complain</u> operation), marked by being connected by an **is-a** link to the node "end"; **is-a** links are also present to indicate the two ways to upgrade customer service, so as to placate a dissatisfied client. All other links are of the **part-of** type and describe operations formed by composition.

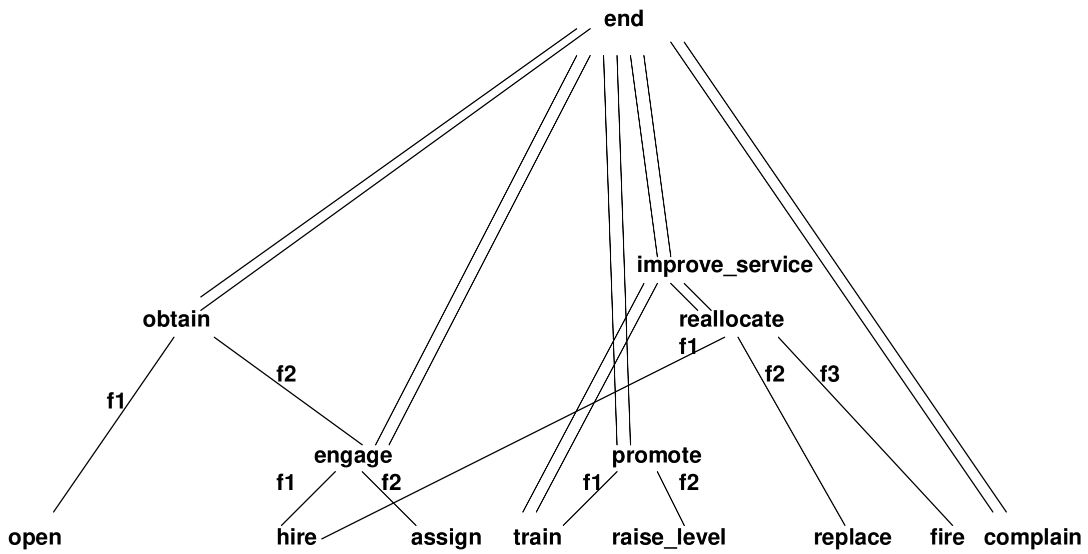

Figure 1 - Library of typical plans (operations)

Let us look at the goals associated with the five end-operations, introducing them, in a semi-formal notation, as new named predicates (the fifth definition contains, in square brackets, an explicit reference to a predicate instance holding at the state before execution):

operation: obtain(C,E) -- primary goal:
assisted(C) ::=
    is-client(C) &
    ∃E serves(E,C)

operation: complain(C,E) -- primary goal:
needing-attention(C) ::=
    dissatisfied(C,E)

operation: improve-service(C,E) -- primary goal:
compensated(C) ::=
    ~ ∃E1 dissatisfied(C,E1)

operation: engage(E,C) -- primary goal:
stable(E) ::=
    is-employee(E) &
    ∃C serves(E,C)

operation: promote(E) -- primary goal:
rewarded(E) ::=
    ∃L [has-level(E,L)/s_0] has-level(E,L + 1)

Notice that constants such as "John", "Beta", etc. do not appear in the above expressions. As similar plots are being matched to build the patterns, the criterion of most specific generalization retains constants wherever they coincide. Only if different constants occur, the criterion introduces a variable in the corresponding parameter place.

A library of typical plans does not necessarily supply the optimal ways to reach the intended goals; the typical plans are those in everyday usage, and tend to show the influence of the company's policies. Once the library has been created, it is ready to be exploited through the Pr/Pg facility. We shall consider reactions to a few different sets of observations:

observations: hire(Mary)
Three end-plans are recognized: engage(Mary,C), obtain(C,Mary) and improve-service(C,E). The answer must display the three possibilities.

observations: hire(Mary)
replace(David,Mary,Epsilon)
Now there is no ambiguity: only improve-service(Epsilon,David) is recognized.

observations: train(Leonard,c280)
The end-plan promote(Leonard) is recognized. Assume that, for no C, serves(Leonard,C) currently holds, which constitutes an obstacle to the execution of the plan, since one pre-condition of raise-level is not satisfied. The plan-generator may suggest that an operation assign(Leonard,Lambda) would fulfill the requirement, assuming that, for no E, serves(E,Lambda) currently holds.

observations: complain(Epsilon,David)
This corresponds to a problem-detection situation, where the availability of an active system might be desirable. As the plan-recognition component indicates the need to achieve improve-

service(Epsilon,David), the two underlying alternative strategies for improving customer service are offered to the user's choice: train or reallocate.

observations: open(Omega)
The end-plan obtain(Omega,E) is recognized, requiring the execution of hire and assign. However, the plan-generator component may be consulted to propose alternative non-typical plans. It may simply generate, for instance, assign(David,Omega) if for no C serves(David,C) currently holds, thus disclosing a non-optimal item in Alpha's policies: it is not necessary to hire a new employee if an unassigned employee is around. Another more involved option is: <replace(Laura,David,Lambda), assign(Laura,Omega)>, which might be better than the simpler ones if, say, to assist company Omega the service of a more experienced employee would be required.

4. Plots in literary narratives

The Russian scholar, V. Propp, one of the forerunners of the literary theory known as Structuralism, characterized the gender of fairy tales by identifying certain typical recurring passages and associating them with 31 "functions", which he proceeded to describe (for details, see [Pr68]): absence, interdiction, violation, reconnaissance, delivery of information, fraud, complicity, villainy or lack, mediation, counteraction, departure, proof, reaction, receipt of magical object, translocation, struggle, marking, victory, liquidation, return, pursuit, rescue, arrival, pretentions, task, solution, recognition, exposure, transfiguration, punishment, and wedding.

Propp performed his analysis over a number of Russian fairy tales, collected by Afanas'ev [Af45]. With each tale, he associated a sequence of executions of his proposed functions, in order to denote the scheme or *structure* of the tale. By what we argued before, the Proppian structures can be said to correspond to plots of the textual narratives of the fairy tales. Continuing Propp's studies, Greimas [Gr66] showed the convenience of reducing the number of functions by gouping some of them, which we did by adopting a **part-of** hierarchy. Likewise, Propp's enumeration of the different forms that can be assumed by each of his functions (e.g. villainy can be either kidnapping, or murder, or mutilation, etc.) corresponds to our use of **is-a** links.

Finally, Propp distinguished 7 personages: villain, donor, helper, victim, dispatcher, hero, false hero. Among them he distributed the execution of the functions, noting that certain functions involve more than one character: Propp's struggle, for instance, involves the hero and the villain. Referring back to the database context, we saw that database objects may include several classes of agents (human beings or organization units), and that agents also figure as users.

More recent literary theories criticized Structuralism, mostly on the grounds that its approach is too "syntactical". Even so, several scholars acknowledge the lasting value of its contributions [We96]. It has also been noted that the stress on formality makes its methods particularly appropriate for automatic processing, as Rumelhart's "story grammars" have long ago demonstrated [Ru75]. We do not claim, however, that our extended use of the structural analysis developed by Propp and others exhausts the possibilities for the automatic handling of plots of narratives. We are currently studying contributions from other literary theories [We96, Ea83, LM87], with the intent of applying them, with the continuation of the project, to revise and expand our method.

To finish this section, we should add that the literary use of the method is not limited to the gender of fairy tales. It applies, in principle, to any gender of literary narrative [Jo72], for which we can determine an appropriate repertoire of operations, predicates and personages. And, by introducing other repertoires, many kinds of narratives usually not classified as literary are tractable as well, such as progress reports on engineering projects, bank account transactions, criminal testimonies, business affair statements, newspaper coverage of political events, travel diaries, etc.

5. Concluding remarks

A prototype Pr/Pg facility has been implemented in Prolog, to test the method described in this paper. It was applied, initially, to the structural analysis of Russian fairy tales, based on a hierarchical library of typical plans which we built to formalize Propp's functions. We have been able to recognize, modify and generate a number of plots of tales with relatively complex features, such as the occurrence of repetitions (e.g. both the hero's brothers and the hero himself undertaking the same sequence of actions) and the chaining of separate tales. Sometimes unexpected and interesting variant plots with the same initial and final states have been generated, suggesting that a certain margin of creativity is within reach.

Further experiments with plots of narratives of various kinds will aim at the detection of intertextualities [Th85, Fu95], and also at the implementation of *structural* indexing and retrieval techniques. Such techniques should allow to find texts compatible with a given structure-pattern, assuming a textual database where each full text would be stored together with its plot (elaborated manually or, ideally, extracted when needed by some automatic process). A subsequent implementation in a hypermedia environment [SR95] is being considered.

As we started to work with database applications, the experience gained from adapting Propp's functions proved to be invaluable. The company Alpha database example described in section 3 was designed and tested. Although simple, the experiment served to demonstrate the usefulness of Pr/Pg in the database context, and guided us in the process of tuning the method and the prototype to accomodate specific database requirements, related, for instance, with the treatment of negation within the closed world assumption.

One of the next additions to be made to the prototype will be the ability to impose conditions on intermediate states, so as to provide more control on the choice of alternatives at each point in a narrative. Letting the choice be determined by the intended effect on readers, in the line of Reception theory [Is72], and employing models of classes of readers [Ri83], is another feature to be incorporated especially for the literary applications.

Thus far, the libraries of typical plans we have been using were designed *a priori*. This is a convenient strategy in many practical situations, where plots are known beforehand; Kautz's example library, for instance, gives recipes to cook popular Italian dishes. But in other situations the typical plans must result from the study of plots obtained from the temporal database, as mentioned in sections 2 and 3. We still have to develop the methods for this knowledge discovery task, possibly looking at contributions from the promising field of case-based learning [Ko93].

Both as records of the past and present story of the mini-world represented in the database and as an anticipation of reachable alternative futures, plots offer a perhaps too terse and formal notation. We intend to investigate, at a later phase, the passage from plots to natural language narratives, as a *text-generation* problem [Mck92]. We note that the *syntax* and the *semantics* of plots are appropriately captured by pre- and post-conditions of operations. Futhermore, when the planning perspective attaches to plots the goals they propose to achieve, the *pragmatic* aspect is taken into account as well. An itegrated coverage of all these aspects is indispensable to generate texts focusing on relevant events and reflecting the purposes motivating the various actions [Wi86].

References

[Af45] Afanas'ev, A. - *Russian Fairy Tales*. New York: Pantheon Books, 1945.

[CF90] Casanova, M. A. and Furtado, A. L. - “An information system environment based on plan generation”. in Proc. of the *International working conference on cooperative knowledge based systems* . Univ. Keele, England, 1990.

[Ea83] Eagleton, T. - *Literary Theory: an Introduction*. Oxford: Basil Blackwell, 1983.

[FC90] Furtado, A. L. and Casanova, M. A. - "Plan and schedule generation over temporal databases". in Proc. of the *9th International Conference on the Entity-Relationship Approach*, 1990.

[FN71] Fikes, R. E. and Nilsson, N. J. - "STRIPS: A new approach to the application of theorem proving to problem solving". *Artificial Intelligence* , 2(3-4), 1971.

[Fu92] Furtado, A.L. - "Analogy by generalization and the quest of the grail". *ACM/SIGPLAN Notices*, 27, 1, 1992.

[Fu95] Furtado, A.L. - "From Alexander of Macedonia to Arthur of Britain". *Arthuriana*, 5,3,1995.

[Ge80] Genette, G. - *Narrative Discourse: An Essay in Method*. translated by J. E. Lewin. Ithaca: Cornell Univ. Press, 1980.

[GGM92] Gaasterland, T., Godfrey, P e Minker, J. - “An overview of cooperative answering”. *Journal of Intelligent Information Systems*, v. 1, n. 2, 1992.

[GHJV95] Gamma, E., Helm, R., Johnson, R., and Vlissides, J. - *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading: Addison Wesley Publishing Company, 1995.

[Gr66] Greimas, A. J. - *Sémantique Structurale*. Paris: Librairie Larousse, 1966.

[Is72] Iser, W. - "The reading process: a phenomenological approach". *New Literary History*, 3, 1972.

[Jo72] Jolles, A. - *Formes Simples*.. Paris: Éditions du Seuil, 1972.

[Ka91] Kautz, H. A. - “A formal theory of plan recognition and its implementation”, in *Reasoning about plans*. J. F. Allen et al (eds.). Morgan Kaufmann, 1991.

[Ko93] Kolodner, J. L. - *Case-Based Reasoning*. San Mateo: Morgan Kaufmann Publishers, 1993.

[LM87] Lambropoulos, V. and Miller, D. N. (eds.). - *Twentieth-Century Literary Theory*. Albany: State University of New York Press, 1987.

[Mck92] McKeown, K. R. - *Text generation: using discourse strategies and focus constraints to generate natural language text*. Cambridge: Cambridge University Press, 1992.

[MCP93] Matheus, C. J., Chan, P. K. and Piatesky-Shapiro, G. - "Systems for knowledge discovery in databases". *IEEE Transactions on Knowledge and Data Engineering*, 5, 6, 1993.

[OS95] Ozsoyoglu, G. and Snodgrass, R. T. - "Temporal and real-time databases: a survey". *IEEE Transaction on Knowledge and Data Engineering*, 7, 4, 1995.

[Pr68] Propp, V. - *Morphology of the Folktale*. translated by Laurence Scott. Austin: University of Texas Press, 1968.

[Ri83] Rich, E. - "Users are individuals: individualizing user models". *International Journal of Man-Machine Studies*, 18, 1983.

[Ru75] Rumelhart, D. E. - "Notes on a schema for stories". in *Representation and Undertanding - Studies in Cognitive Science*. D. G. Bobrow and A. Collins (eds.). New York: Academic Press, 1975.

[SR95] Schwabe, D. and Rossi, G. - "Building hypermedia applications as navigational views of information models". in Proc. of the *28th. Hawaii International Conference on System Sciences*, Jan, 1995.

[Th85] Morgan, T. E. - "Is there an intertext in this text?: literary and interdisciplinary approaches to intertextuality". *American Journal of Semiotics*, 3, 1985.

[We96] Webster, R. - *Studying Literary Theory*. London: Arnold, 1996.

[Wi86] Wilensky, R. - "Points: a theory of the structure of stories in memory". in *Readings in Natural Language Processing*. B. J. Grosz, K. S. Jones and B. L. Webber (eds.), Los Altos: Morgan Kaufmann, 1986.

[Wo91] Woods, S. G. - *An Implementation and Evaluation of a Hierarchical Non-linear Planner*. Master's thesis. Computer Science Department of the University of Waterloo, 1991.