OntoReST: A RST-based Ontology for Maintaining Semantic Consistency in Collaborative Writing

Charbel Rahhal, Nishadi Desilva, Hala Naja-Jazzar, Hala Skaf-Molli, Pascal Molli

▶ **To cite this version:**

Charbel Rahhal, Nishadi Desilva, Hala Naja-Jazzar, Hala Skaf-Molli, Pascal Molli. OntoReST: A RST-based Ontology for Maintaining Semantic Consistency in Collaborative Writing. [Research Report] 2007, pp.18. inria-00194604v1

HAL Id: inria-00194604

https://inria.hal.science/inria-00194604v1

Submitted on 6 Dec 2007 (v1), last revised 13 Dec 2007 (v3)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

OntoReST: *A RST-based Ontology for Maintaining Semantic Consistency in Collaborative Writing*

Charbel Rahhal — Nishadi De Silva — Hala Naja-Jazzar — Hala Skaf-Molli — Pascal Molli

N° ????

December 2007

Thème COG

ISSN 0249-6399 ISRN INRIA/RR--????--FR+ENG

OntoReST: A RST-based Ontology for Maintaining Semantic Consistency in Collaborative Writing

Charbel Rahhal*, Nishadi De Silva †, Hala Naja-Jazzar ‡ , Hala Skaf-Molli§, Pascal Molli ¶

Thème COG — Systèmes cognitifs
Projet ECOO

Rapport de recherche n° ???? — December 2007 — 15 pages

Abstract: Collaborative writing is the process by which more than one author contributes to the content of a document. Multi-synchronous collaboration is very efficient for reducing task completion time but is known to produce inconsistent documents. Most existing collaborative writing environments do not really check the semantic consistency of documents. They rely on authors to verify the coherence of the document. This introduces a severe overhead for authors to achieve efficient collaboration. To address this lack, we use semantic web technologies and a discourse theory called Rhetorical Structure Theory (RST) to reduce the overhead of consistency checking. We develop *OntoReST*, an ontology based on RST that helps detect incoherent texts automatically. *OntoReST* also provides authors with valuable information about the semantic structure of texts which contributes towards more coherent documents.

Key-words: Collaborative writing, Ontology, Semantic consistency, Rhetorical Structure Theory (RST)

* charbel.rahal@loria.fr, ECOO Project, Nancy-University, LORIA, INRIA Centre - Nancy Grand Est
† n.desilva@ecs.soton.ac.uk, School of Electronics and Computer Science, University of Southampton, UK
‡ hjazzar@ul.edu.lb, Faculty of Science - Branch 3 - Lebanese University, Lebanon
§ skaf@loria.fr, ECOO Project, Nancy-University, LORIA, INRIA Centre - Nancy Grand Est
¶ molli@loria.fr, ECOO Project, Nancy-University, LORIA, INRIA Centre - Nancy Grand Est

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois, Campus scientifique,
615, rue du Jardin Botanique, BP 101, 54602 Villers-Lès-Nancy (France)
Téléphone : +33 3 83 59 30 00 — Télécopie : +33 3 83 27 83 19

OntoReST: Une ontologie basée sur RST pour le maintien de la cohérence sémantique dans l'édition collaborative

Résumé : L'édition collaborative est un processus durant lequel plusieurs auteurs contribuent au contenu d'un document. La collaboration multi-synchrone est très efficace car elle permet de réduire le temps de complétude d'une tâche. Cependant elle est connue pour produire de documents incohérents. La plupart des environnements d'édition collaborative existants ne vérifient pas réelement la cohérence sémantique des documents. La vérification de la cohérence reste à la charge des auteurs. Ceci constitue un obstacle majeur pour garantir une collaboration efficace. Pour adresser ce problème, nous utilisons les technologies du web sémantique et une théorie du discours nommée Rhetorical Structure Theory (RST). Nous développons OntoReST, une ontologie basée sur RST qui aide à la détection automatique des textes incohérents. OntoReST offre aussi aux auteurs une information utile sur la structure sémantique des textes en conduisant à des documents plus cohérents.

Mots-clés : Travail collaboratif, Ontologie, Cohérence sémantique, RST

Contents

1 Introduction

Collaborative writing is the process by which more than one author, in addition to sharing opinions, also contributes to the content of a document [10]. Collaborative writing is standard practice in technical and scientific settings; some examples include research papers, software development, proposals for funding and user manuals. When collaboration is efficiently managed, the advantages of working in a group include increased efficiency, reduced errors and the benefits of different viewpoints and expertise [21, 14]. If collaboration is poorly supported, it can lead to inconsistencies, misunderstandings, conflicts, redundant work and coordination problems. The nature of collaboration varies extensively in terms of the group writing strategies, proximity and synchronicity of group activities [21]. For instance, collaborative writing can be done in parallel synchronously, asynchronously or multi-synchronously [4, 13, 17]. Synchronous work with a joint-writing strategy can give good results but is limited to small groups working for short periods of time. Asynchronous work with turn-taking strategies allow to work distributed in time but does not allow task parallelization. The multi-synchronous interaction mode allows people to work in parallel synchronously or asynchronously while being distributed in time and space. It is the least restrictive collaborative writing strategy. This working mode is well known in software engineering. Software engineers commonly use version control systems or distributed version control system to achieve high parallelization of tasks and reduce development time. This high level of concurrency is potentially risky and can lead to software inconsistencies. Fortunately, software engineers can compile software and run automatic tests in continuous integration strategy [5] to limit the risk of inconsistencies.

Unfortunately, we cannot reuse these proven efficient collaborative strategies outside the software engineering world. Currently, we do not dispose of automatic testing mechanisms for checking document consistency according to some specifications. Text documents do not have typed objects to reason about, and they cannot be "compiled" in order to verify some type safety violation.

Multi-synchronous collaboration mode greatly increases the risk of misaligned contributions by individual authors. While each section may be well constructed, they may not 'fit' logically when placed together. While this is easy to correct in short texts, the problem is much harder in large, multi-authored documents. This is what we refer to as **semantic inconsistency** and is the focus of this paper.

Semantic consistency is poorly supported by existing Multi-synchronous collaborative writing environments. In these environments, each author works on her own copy of the shared data. The system is correct if: (1) it eventually converges to an idle state where all copies are identical (2) user intentions are preserved [20]. 'Intention' means that if an operation produced an effect when generated, this effect must be observable in the same way by all users. For instance, if the author has the text AC and she wants to insert B exactly between A and C, the system will ensure that all the authors will see B between A and C regardless of other concurrent operations. Of course, intention preservations do not prevent two authors to insert the same idea twice or contradicting ideas at the same place in the document. It is clear that automatic detection of such problems is very difficult and current collaborative environments rely on authors to verify the logical connections of the content. Collaborative environments help authors by providing awareness about concurrent changes. Next, authors have to verify that each local operation is compatible with all other concurrent operations. We want to leverage this stage by providing more information about the context and impact of modifications. This requires us to define more clearly what is meant by "semantic consistency".

"Semantic consistency" or "coherence" (the two words are used interchangeably in this paper) is a subjective phenomenon because different readers perceive texts differently. What we mean by coherence is the ease with which a document can be read and understood. While multiple factors such as grammar and punctuation can affect this, the logical progression of the ideas presented is perhaps the one with the most impact. The mere sequence in which the sentences and sections are laid out can significantly affect how they are understood [9, 12]. (See examples in our previous papers [16, 3]).

Research into this area revealed that linguists had developed discourse theories to guide the analysis and synthesis of text. In particular, Mann and Thompson (1988) developed Rhetorical Structure Theory (RST) [11]. This theory attributes the coherence of a text to the implicit logical relationships that exist between its segments. In previous work [16], we combine RST and merging algorithm based on the operational transformation approach [15] to address document coherence during collaborative writing.

In this paper, we use techniques from the semantic web domain to address the problem of semantic coherence during collaborative writing. More precisely, we define an ontology called *OntoReST* based on RST that allows to improve the quality of documents. *OntoReST* turns the document contents into a machine readable and structured form which allows the detection of semantic inconsistencies. It also allows document querying, reasoning and searching. Such ontologies can be used to turn text into typed objects. This allows software to be written (equivalent to compilers for programs) to check some properties about the document. Consequently,

multi-synchronous collaboration interaction mode can be used in order to achieve more efficient collaboration for producing text documents.

So, first we give a brief description of RST and show how the structure of a text can be analysed using it. In section 3, we define *OntoReST* formally using OWL and description logic DL. Next, in section 4, we detail the required steps to manipulate this ontology during collaborative writing and describe the applied merging algorithm using an example. We also discuss related work in the field and finally, present our conclusions and directions for future work.

2 Rhetorical Structure Theory (RST)

There are several discourse theories developed by linguists to analyse the structure of texts. We have chosen Rhetorical Structure Theory (RST) [11] for its simplicity, precise relationship definitions and its ability to render itself to formal descriptions. RST attributes the coherence of a text to implicit logical relationships such as 'Motivation', 'Background' and 'Elaboration' that exist between sections of the text. The rest of this section gives a brief overview of how a RST analysis can be done and highlights parts of the process essential to the discussions in this paper.

2.1 Analysing a Text Using RST

The first step in a RST analysis is to divide the text into non-overlapping, functionally independent segments[11]. As an example, we use the text below to demonstrate the segmentation.

[**Text 1:**] [*1:The problem with existing writing software is their inability to detect semantic problems in documents.*] [*2:OntoReST is the result of combining the techniques behind onotologies and those related to RST.*] [*3:When combined with existing writing tools, it can help improve the quality of documents by alerting authors to possible semantic inconsistencies.*]

During a bottom-up analysis, the second step is to identify logical relationships that exist between pairs of segments. For instance, in the above example, we see segment 3 to be providing motivating information to the statement in segment 2 (i.e. Motivation relationship).

Segments in a relationship can play one of two roles: a **nucleus** or a **satellite**. A nucleus is considered to be an important segment, essential to the understanding of the text. A satellite is not as critical but does provide supporting material.

More information about RST can be found in Mann and Thompson's paper (1988). Mann and Thompson defined 23 relationships, each with precise descriptions of what should go in each end of the relationship and the its expected effect on the reader. Henderson and De Silva [7], however, considered 23 to be too many for technical writing and began selecting a subset of relationships that were sufficient for analysing technical documents. In [2], a user study has shown that technical authors found a set of 9 relationships adequate for their analysis.

In the analysis, segments involved in a relationship collectively form a **span**. A span can in turn become part of another relationship. For instance, in our example, the span of segments 2 and 3 is identified as being in a BACKGROUND relationship with segment 1 (i.e. segment 1 provides background information that helps understand the significance of BOTH segments 2 and 3). Hence, the analysis is a recursive process and continues until all the segments are assembled into a tree of relationships as shown. This is called a **RSTree**. In the RSTree the arrowhead points towards the Nucleus as shown in the figure 1.

2.2 RSTrees Properties

The important point about RSTrees is that Mann and Thompson (1988) [11] conjecture that producing a well-formed RSTree for a text indicates that a text is coherent. This is a useful measure in our work where we apply RST to detect incoherent texts. They define four properties that determine if a RSTree is well formed. They are:

- ***Completedness*** One schema application (the root) should cover the entire text.
- ***Connectedness*** Each text span/segment, apart from the span that covers the entire text, should be a minimal unit in the tree or part of another schema application.
- ***Uniqueness*** Each text span/segment should have only one parent (i.e. each schema application consists of a different set of text spans/segments).

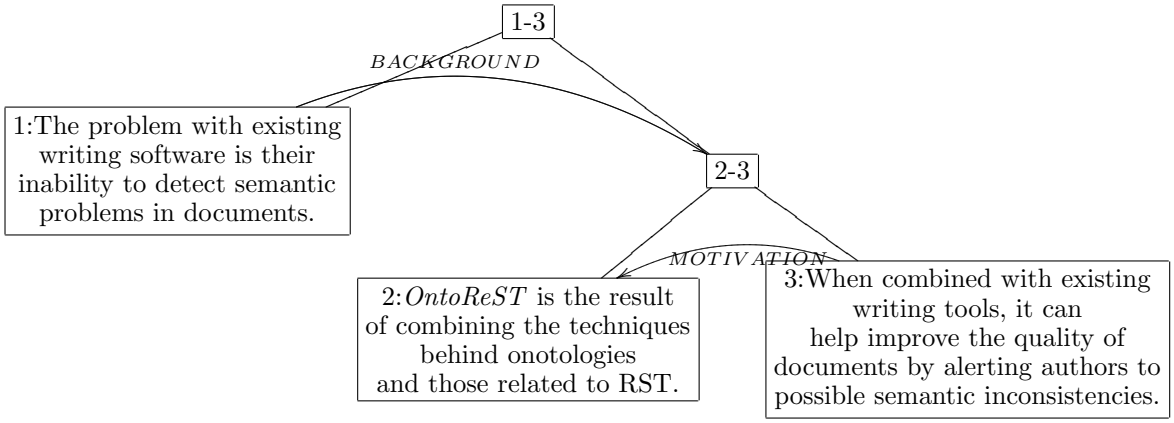


Figure 1: RST for Text1

- ***Adjacency*** Only adjacent text spans/segments can be grouped together to form larger spans.

We make use of these properties in *OntoReST* to evaluate the coherence of documents written collaboratively.

3 Ontologies for Semantic Collaborative Writing

An ontology describes basic concepts in a domain and defines relations among them [19]. It is composed of concepts, properties, relations and restrictions on properties. We formalize the RST theory as an ontology. This allows to take advantages of the semantic Web by turning the content of document into a machine readable and structured form. It provides also a common knowledge base for the authors. Moreover, it is possible to detect automatically semantic problems and to make interesting queries on the document. For example, by selecting all the **Nucleus** we can produce a summary of the document.

We have defined three ontologies (see figure 2):

- **Document structure ontology:** captures the internal structure of the document (sections, sentences, etc).
- **Rhetorical ontology *OntoReST***: models the document in terms of its rhetorical elements (*i.e.* segments, spans and RST relationships). This allows the detection of semantic inconsistencies in documents.
- **Annotation ontology**: annotates the sentences and the sections of the document. It also captures additional meta data about the document. This is helpful in classifying the documents according to their types, authors and topics.

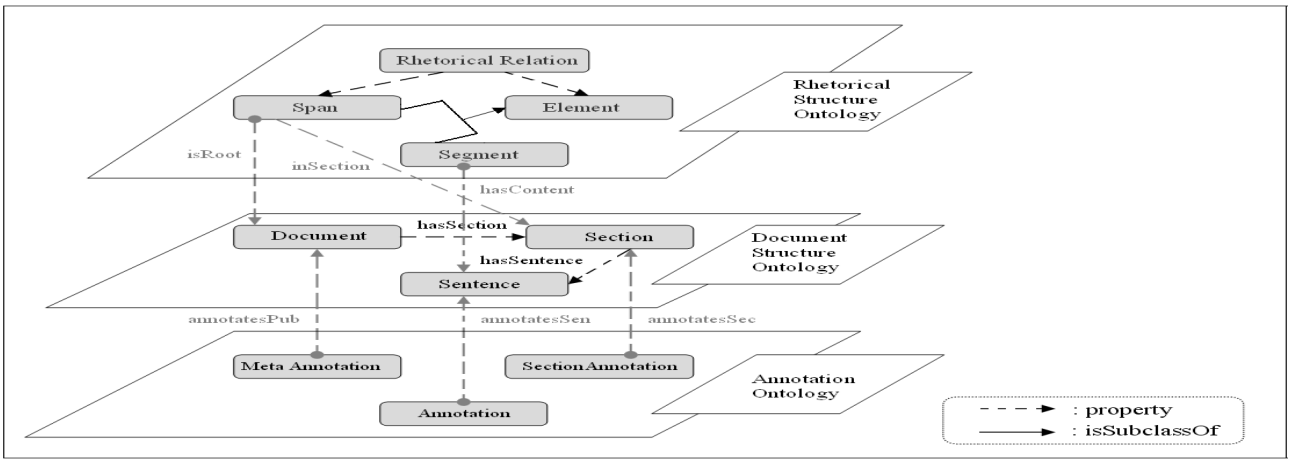


Figure 2: Ontology Layers

The definition of three separate ontologies allow flexible modification of the ontologies. For instance, we can modify the structure of the document, without modifying the RST information. The idea of defining three different ontologies is inspired from [22]. As our major interest is detecting semantic inconsistency, in this paper we will focus just on the rhetorical ontology *OntoReST*.

3.1 Rhetorical Ontology *OntoReST*

The rhetorical ontology captures the semantics of the text using RST. It models the segments, spans and rhetorical relations. It also uses the four properties for well-formed RSTrees to detect semantic discrepancies in the document. We do not use all 23 rhetorical relations defined by Mann and Thompson, but we only use the subset of 9 relations identified in [7]. This ontology is not tied to a specific document but it is a generic ontology applicable to several types of technical documents.

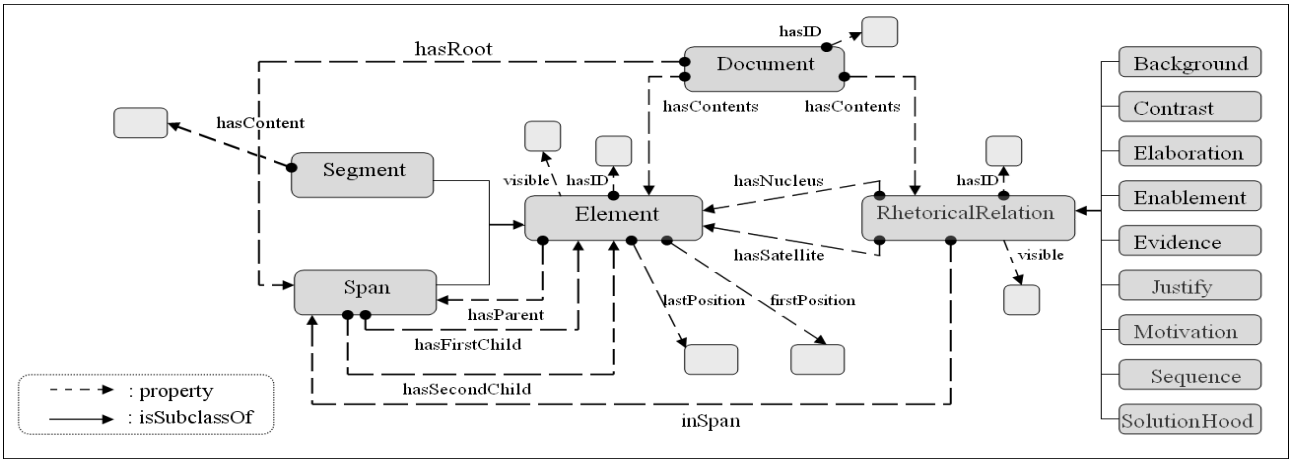


Figure 3: The Rhetorical Ontology *OntoReST*

We identify five main concepts: *Document, Element, Span, Segment and RhetoricalRelation* as shown in figure 3. In this figure, we omit inverse properties for simplicity.

A *Document* is composed of an ordered sequence of segments and it has a set of spans and a set of rhetorical relations. It has also the following properties:

- *hasID*: a unique identifier given by the system for the document.
- *hasContents*: links a document to its segments, spans and rhetorical relations. The textual content of a document is the *hasContent* values of its all ordered segments. This property is the inverse property of *containingDoc*.
- *hasRoot*: indicates the root of a document. Its inverse property is *isRoot*. For example, the span $1-3$ is the root of the text in the figure 1. A root is a span that has the *isRoot* property.

An *Element* can be either a Segment or a Span. It is used to avoid repetition of common properties in Segments and Spans. It has the following properties:

- *hasID*: is a unique identifier given by the system for each element.
- *firstPosition* and *lastPosition*: are the position of the element in the document. They have equal value for a segment. For a span they indicate the position of the first and the last segments covered by the span in the document. We use them for the adjacency property. For example, the firstPosition of the Element $1-3$ of the figure 1 is 1 and the lastPosition is 3. The elements 1 and 3 are not adjacent because their positions are not consecutive. While the Element $2-3$ has two adjacent children.
- *visible*: is the status of the element. Its range is boolean and has "true" as default value. It turns to "false" when the element is deleted. We add this property because we do not delete physically the element, we just mark it as invisible. This property will be set by the merging algorithm as we will see later.

- *hasParent*: indicates the parent of an element. Each element in the document has a parent property except the root (which is the span that covers the entire text). For example, the span $2-3$ of the figure 1 is the parent of the Elements 2 and 3 and the span $1-3$ is the root of the text, it has no parent.
- *containingDoc*: indicates the document containing the elements.

A *Segment* is a sentence. It inherits the properties in element and has an additional property *hasContent*.

A **Span** covers two adjacent segments. It is always linked to one and only one rhetorical relation. Span inherits the properties in element and has the additional following properties:

- *hasFirstChild* and *hasSecondChild*: are the first and the second child elements of a Span.
- *isRoot*: indicates the parent document of the root span. Its inverse property is *hasRoot*.
- *hasRstRelation*: indicates the rhetorical relation that exists between the children of the span. The children of a span are the nucleus and the satellite of that relation.
- *changed*: is the status of the span. It is "false" by default. It is set to "true", after deleting one of its children or the hasRstRelation property.

The *changed* property allows to propagate modifications to the concerned spans and relations in the RSTree. This property gives some awareness that helps the authors to localize the modified parts of the documents. This saves time and effort for the authors during the revising and the reviewing phase, and especially after integrating the modifications of other authors.

A *Rhetorical Relation* is the rhetorical relation that holds between two elements. It is always linked to a span. It has the following properties:

- *hasID*: is a unique identifier given by the system for each relation.
- *hasName*: is the name of the relation such as *Motivation* or *Elaboration*. Its domain is a Relation and its range is a list of relations' names. According to the RST theory, the name of the relation specifies the order of the nucleus and the satellite i.e. nucleus is before satellite or the opposite.
- *hasNucleus* and *hasSatellite*: represent the nucleus and the satellite respectively of the relation. Their values are the first and the last child of the linked span i.e. they reference the first and last child properties of the related span. We need to define the *hasNucleus* and *hasSatellite*: properties since we make separation between span and relation. This allows to avoid to reconstruct the RSTree in case of minor modifications (such as correct misspelling or rephrasing a segment without changing its meaning).
- *visible*: is the status of the relation. Its range is boolean and has "true" as default value. It turns to "false" when the relation is deleted.
- *inSpan*: this property is the inverse property of the *hasRstRelation*.

3.2 Formal Specification of *OntoReST*

We use both OWL (Web Ontology Language) and Description Logic to formalize *OntoReST*. We write the class axioms, the property axioms and the constraints in OWL DL which is the most investigated species of OWL [1]. OWL DL has different syntaxes. However, the normative syntax for OWL DL is the abstract syntax. OWL can be seen as an alternate notation for Description Logic Language $\mathcal{SHOIN}(\mathtt{D})$. Table 1 presents the concepts of the *OntoReST*.

Table 2 presents the object and data properties. In this table, we skipped some identical properties for simplicity. Both tables represent a mapping between the OWL DL abstract syntax and the syntax of the Description Logic $\mathcal{SHOIN}(\mathtt{D})$.

4 *OntoReST* in Collaborative Writing

Using *OntoReST* to maintain semantic consistency during collaborative writing requires the following steps:

OWL Abstract syntax	**DL syntax**
Class axioms	
SubClassOf(Document Thing)	Document $\sqsubseteq \top$
SubClassOf(Element Thing)	Element $\sqsubseteq \top$
SubClassOf(Span Element)	Span $\sqsubseteq$ Element
SubClassOf(Segment Element)	Segment $\sqsubseteq$ Element
EquivalentClasses(Element unionOf(Span Segment))	Element $\equiv$ Span $\sqcup$ Segment
DisjointClasses(Span Segment)	Span $\sqcap$ Segment $\sqsubseteq \bot$
EquivalentClasses(RheRelation)	RheRelation $\equiv$
unionOf(Background ... SolutionHood))	Background $\sqcup$ ... $\sqcup$ SolutionHood
DisjointClasses(Background, Contrast)	Background $\sqcap$ Contrast $\sqsubseteq \bot$, ...
DisjointClasses(Sequence, SolutionHood)	Sequence $\sqcap$ SolutionHood $\sqsubseteq \bot$

Table 1: The Rhetorical concepts in OWL and DL

OWL Abstract syntax	**DL syntax**
Property axioms	
ObjectProperty(hasFirstChild domain(Span) range(Element))	$\top \sqsubseteq \forall$ hasFirstChild$^-$.Span $\top \sqsubseteq \forall$ hasFirstChild.Element
restriction(hasFirstChild maxCardinality(1) minCardinality(1))	Span $\sqsubseteq$ (= 1 hasFirstChild)
ObjectProperty(hasNucleus domain(RheRelation) range(Element))	$\top \sqsubseteq \forall$ hasNucleus$^-$.RheRelation $\top \sqsubseteq \forall$ hasNucleus.Element
restriction(hasNucleus minCardinality(1) maxCardinality(2))	RheRelation $\sqsubseteq$ ($\geq$1 hasNucleus) $\sqcap$ ($\leq$2 hasNucleus)
ObjectProperty(hasParent domain(Element) range(Span))	$\top \sqsubseteq \forall$ hasParent$^-$.Element $\top \sqsubseteq \forall$ hasParent.Span
ObjectProperty(inSpan domain(RheRelation) range(Span) inverseOf(hasRstRelation))	$\top \sqsubseteq \forall$ inSpan$^-$.RheRelation $\top \sqsubseteq \forall$ inSpan.Span inSpan $\equiv$ hasRstRelation$^-$
DatatypeProperty(firstPosition domain(Element) range(String))	Element $\sqsubseteq \exists$ firstPosition.String
DatatypeProperty(hasID domain(unionOf(Element RheRelation Document)) range(Integer))	$\top \sqsubseteq \forall$ hasID$^-$.(Element $\sqcup$ RheRelation) Element $\sqcup$ RheRelation $\sqcup$ Document $\sqsubseteq \exists$ hasID.Integer
DatatypeProperty(visible range(Boolean))	Element $\sqcup$ RheRelation $\sqsubseteq \exists$ visible.Boolean

Table 2: The Rhetorical properties in OWL and DL

- *Ontology instantiations*: Each instance of concept is created locally at a user's site. So, when the author adds a sentence, the system will detect this modification as an operation and create an instance of the segment concept. The operation is then sent and integrated at all others users' sites. Eventually, when there is no modifications, the replicated instances will be the same at all sites. In the section 4.1, we define operations to instantiate this ontology.
- *Merging algorithms*: To integrate remote modifications, we use the *Tombstone Transformation Functions* algorithm [15] (TTF) as detailed in the section 4.3.
- *Inconsistency checker*: There is a significant difference between using RST in collaborative writing and traditional applications of RST. Usually, RST is applied to a 'static' text. However, in collaborative writing, the text continually changes and its corresponding RSTree changes too. We have to check that the new RSTree respects the four properties defined in the RST theory as detailed in section 4.4.

4.1 Populating Ontology

In this section, we describe the process of ontology's instantiation during the edition.

During the edition, the changes made by the authors are detected by the system as the following operations:

- *addSegment(position, hasID, content, sid)* adds an instance of segment with the specified position and the text content. *sid* is the identifier of the site that generates the operation. The *sid* is necessary for the merging algorithm.
- *delSegment(position)* deletes logically a segment at the given position. The *visible* property of the segment is set to false. There is no physical deletion of segments to ensure the convergence [15]. If the segment has a parent span, then the *changed* property of its parent is set to true.
- *addSpan(hasID, hasFirstChildID, hasSecondChildID)* creates an instance of span with the required properties.
- *delSpan(hasID)* deletes logically a span. The *visible* property of the span is set to false. If the span has a parent span, then the *changed* property of its parent is set to true.
- *addRelation(hasID, NucleusID, SatelliteID, SpanID, hasName)* adds a rhetorical relation between the children of its linked span. For example, the hasFirstChild (hasSecondChild) of the span could be the satellite(nucleus) or the nucleus (satellite) of that relation.
- *delRelation(hasID)* deletes logically a relation. The *visible* property of the relation is set to false. The *changed* property of the span linked to this relation is set to true.

We define only *add* and *delete* operations. Because during the merge the *update* operation is detected as *delete* followed by *add* by the diff algorithms.

Let us consider a scenario where an author u_1 is working on *site*1. She wants to write the *Text 1* of the section 2 but this time with rhetorical annotations.

She modifies her local copy by generating operations. The system will detect these changes as the following sequence of operations:

```
S =[addSegment(1, 1A, "The problem with..",1);addSegment(2, 1B, "OntoReST is..", 1);
 addSegment(3, 1C, "When combined with..", 1); addSpan(2-3s, 1B, 1C );
 addRelation(2-3, 1C, 1B, 2-3s, "Motivation"); addSpan(1-3s, 1A, 2-3s);
 addRelation(1-3, 1A, 2-3s, 1-3s, "Background")]
```

The system will build the RSTree for *AnnotatedText1* as depicted in the figure 4.

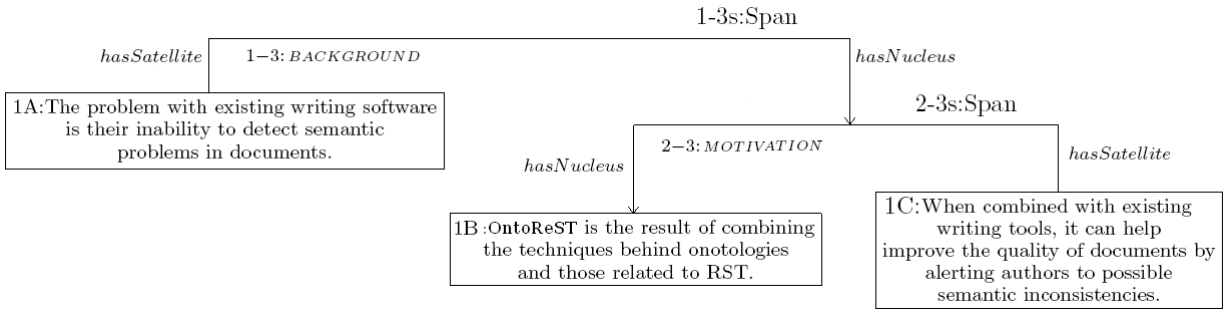


Figure 4: *RSTree of AnnotatedText1*

4.2 Concurrent Writing

Now consider that two authors A and B working on *site*2 and *site*3 respectively are writing the *AnnotatedText1* of the figure 4. Each author has his own copy of the text.

Author A wants to delete the segment in position 2. So, he deletes this segment. In order to preserve local consistency of the RSTree, the system will propagate this deletion to its parent span and its associated relation. The span $2-3s$ and the relation $2-3$ are logically deleted. And also, the parent of the span of $2-3s$ will be replaced with $1-3's$ and the relation $1-3$ with $1-3'$ as shown in the figure 5.

The changes performed by *Author*A will be detected by the system as the following sequence of operations $P1$:

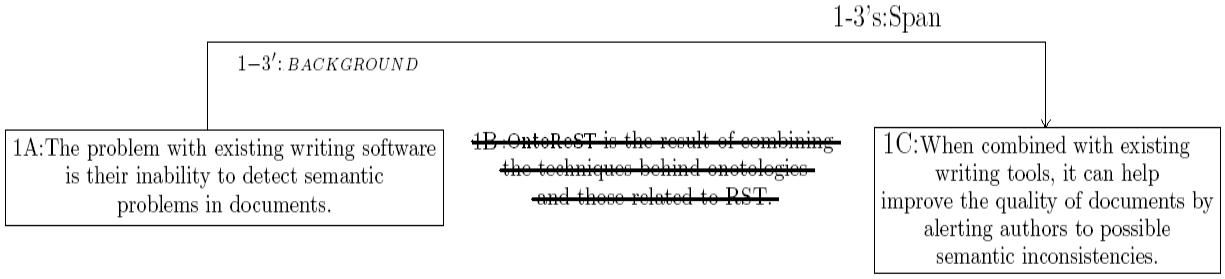
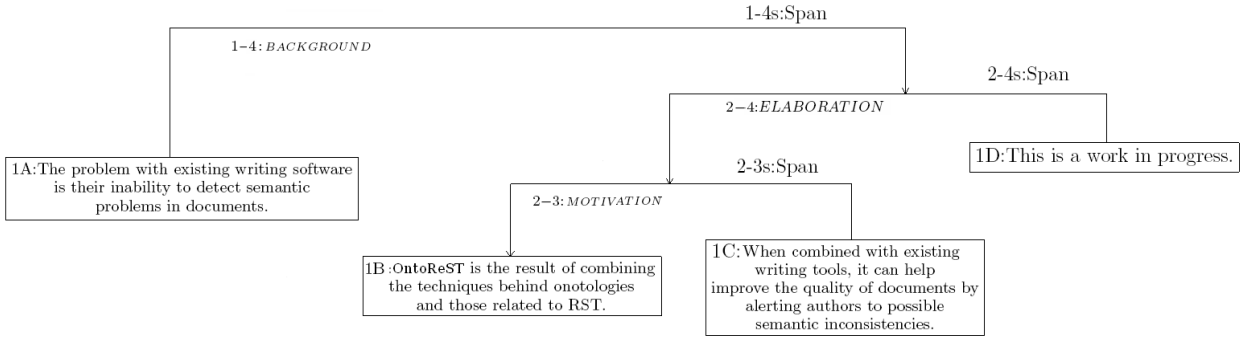


Figure 5: AnnotatedText1 of the Author A

```
P1= [delSegment(2); delSpan(2-3s); delRelation(2-3); delSpan(1-3s);
delRelation(1-3); addSpan(1-3's, 1A, 1C); addRelation(1-3', 1A, 1C, "Background")]
```

At the same time, the author B performs concurrent operations. She adds a new segment:"This is a work is in progress." at the position 4 and the appropriate relations as shown below:



The system produces the following set of operations $P2$:

```
P2=[addSegment(4, 1D, "This...", 3); delSpan(1-3s); delRelation(1-3);
addSpan(2-4s, 2-3s, 1D);addRelation(2-4, 1D, 2-3s, "Elaboration");
addSpan(1-4s, 1A, 2-4s); addRelation(1-4, 1A, 2-4s, "Background");]
```

4.3 Merging Ontological Data

In this section, we will detail through an example how we merge the above concurrent operations. *AuthorA* has generated P_1 and *AuthorB* has generated P_2, so the copies hosted on $site2$ and $site3$ are diverging now *i.e.* they have different content. Both sites exchange their operations and run the integration process as depicted in figure 6. In order to converge, the system has to ensure that $Merge(P1, P2) = Merge(P2, P1)$. Unfortunately, this property is not ensured by traditional merge algorithms.

This problem is well-know in CSCW community. The Operation Transformation (OT) framework [6] has been developed to ensure convergence in these conditions. In [16], we defined a set of all transformation functions that deal with concurrent operations and ensure convergence of semantically annotated documents with RST.

As shown in the figure 7, the final state is converged towards a value that has both **RSTree**. This value is inconsistent since it does not respect the rhetorical properties. For example, the segment at the position 1 violates the uniqueness property. Having both **RSTree** will help the authors to better understand the reasons of the semantic inconsistency and to locate exactly the segments of the text that are responsible of this inconsistency. In the next section, we detail our *inconsistency checker*.

4.4 Inconsistency Checker

In our approach we consider a document is semantically coherent if it respects the four properties of the RST (see section 2). Therefore, we formalize these properties as constraints in the *OntoReST* ontology. These

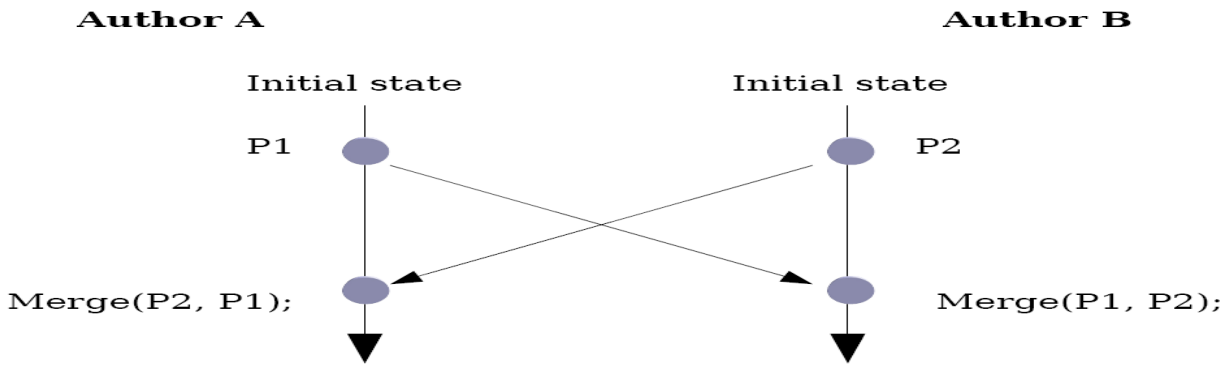


Figure 6: Global merging scenario

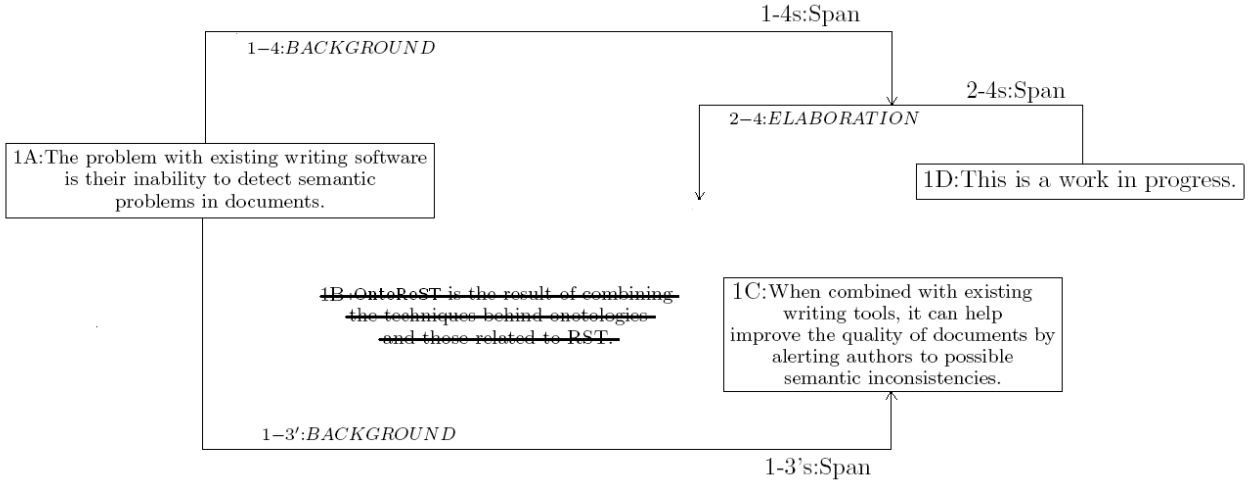


Figure 7: Final Content of the *AnnotatedText1*

constraints are checked continuously. The inconsistency checker allows individual author to produce coherent documents and to detect semantic inconsistency after merging concurrent modifications.

To check the inconsistent instances of the *OntoReST*, we formalize the violation of the coherence properties of the RST in DL on an instance d of the document $Document(d)$ as follows :

- **Connectedness Violation (ConV)** For the document d, an inconsistent element that violates the connectedness property is a visible element that has no parent and that is not a root of d.

  $\texttt{ConV} \equiv \texttt{Element} \sqcap \exists$ visible.{true} $\sqcap \exists$ $\texttt{containingDoc.}\{\texttt{d}\} \sqcap \forall$ hasParent.$\bot \sqcap \forall$ isRoot.$\neg\{\text{d}\}$

- **Uniqueness Violation (UniV)** For the document d, an inconsistent element that violates the uniqueness property is a visible element that has a number of parents different than one.

  $\texttt{UIE} \equiv \texttt{Element} \sqcap \exists$ visible.{true} $\sqcap \exists$ containingDoc.{d} $\sqcap \neg$ `(=1 hasParent)`

- **Completedness Violation (CompV)** The document d has a number of roots different than one.

  $\texttt{CompV} \equiv \neg$ `(= 1 hasRoot)`

- **Adjacency Violation (AdjV)** For a visible span in the document d, there exists a visible element (segment or span) between the last position of its first child and the first position of its second child.

  $\texttt{C1} \equiv \texttt{Span} \sqcap \exists$ visible.{true} $\sqcap \exists$ containingDoc.{d}

  $\texttt{C2} \equiv \texttt{Element} \sqcap \exists$ visible.{true} $\sqcap \exists$ containingDoc.{d}

  Let s and e be $C1(s)$ and $C2(e)$:

$$\texttt{AdjV} \equiv (> \texttt{firstPosition.\{e\}} \quad \texttt{lastPosition.hasFirstChild.\{s\}}) \sqcap (< \texttt{lastPosition.\{e\}} \quad \texttt{firstPosition.hasSecondChild.\{s\}}).$$

We have implemented and verified our *OntoReST* in Protégé and our rhetorical constraints in Protégé Axiom Language (PAL).

5 Related works

In the collaborative writing domain, most work on semantic consistency are based on constraints. Some approaches allow the violation of the constraints. In case of their violation, the reparation is done either automatically like in [18] or manually. Others prevent the violation of the constraints like in [8]. If an operation violates the constraints, the operation is canceled. Both approaches in [18] and [8] bring about lost updates which is not ideal.

Constraints can be specific to an application and concern more the document structure. However, they cannot capture the co-author's understanding and logical reasoning about the text. RST provides this need by attributing relationships between its segments. These relationships create an overall effect on the reader which contributes to understanding the text better. Therefore, when authors exchange documents, they also pass on their understanding of it via the attached RST relationships. In our work, we evaluate the semantic consistency of documents using the RST properties as our constraints.

The project SALT (Semantically Annotated Latex) has some common features with our work. In [22], the authors propose a framework for authoring and annotating LaTeX documents. They develop ontology based on RST. The authors add RST-based semantic tags to their LaTeX documents while editing. SALT does not consider collaborative work. In our work, the *OntoReST* ontology is not just used to add semantic annotations within the document but also as a tool to evaluate the document's level of coherence during collaborative writing. By constantly maintaining and checking the four properties, we are able to detect inconsistencies and alert the authors to such areas. We anticipate that we can integrate our work easily into the SALT framework such that our RST capability can be extended into LaTeX documents too.

6 Conclusions and Future Work

The problem we have tackled in this paper is the lack of support in current collaborative writing tools to detect semantic problems in texts. We have combined ideas from a discourse theory called Rhetorical Structure Theory (RST) and ontologies to develop OntoReST. RST has provided us with a way to formally identify what it means for a text to be coherent (or semantically sound). RST attributes the coherence of a text to underlying relationships between its segments such as ELABORATION or SEQUENCE. These relationships create an overall effect on the reader which contributes to understanding the text better. The creators of RST also define some properties which we have formalised and used in our work to evaluate if the text is semantically sound.

OntoReST offers a novel application of semantic annotations where the focus is to achieve more coherent documents in collaborative writing. It benefits from having the advantages of ontologies such as providing a common knowledge base for authors, the ability to easily search for data through queries. We make use of a merging technique to ensure the convergence of the OntoReST, *i.e.* getting identical RSTrees for documents on all the authors' sites leaving no room for confusion.

This combination of different strands of research (RST, ontologies and collaborative writing) to bridge a gap in existing tools is novel and is the main contribution of our paper. As future work, we intend to evaluate its usability and identify its shortcomings.

Acknowledgments

We thank the Orpailleur team at LORIA, especially Jean Lieber for improving and revising the OntoReST and Yannick Toussaint for simulating discussions.

References

[1] J. de Bruijn, R. Lara, A. Polleres, and D. Fensel. Owl dl vs. owl flight: conceptual modeling and reasoning for the semantic web. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 623–632, New York, NY, USA, 2005. ACM.

[2] N. De-Silva. *A narrative-based collaborative writing tool for constructing coherent technical documents.* PhD thesis, University of Southampton, 2007.

[3] N. De-Silva and H. Skaf-Molli. Narratives to preserve coherence in collaborative writing. In *The Eight International Workshop on Collaborative Editing, With CSCW 2006*, Banff, Canada, November 2006.

[4] P. Dourish. The Parting of the Ways: Divergence, Data Management and Collaborative Work. In *Proceedings of the European Conference on Computer-Supported Cooperative Work - ECSCW'95*, pages 215–230, Stockholm, Sweden, September 1995. Kluwer Academic Publishers.

[5] P. Duvall, S. Matyas, and A. Glover. *Continuous Integration: Improving Software Quality and Reducing Risk (The Addison-Wesley Signature Series).* Addison-Wesley Professional, June 2007.

[6] C. A. Ellis and S. J. Gibbs. Concurrency Control in Groupware Systems. 18:399–407, May 1989.

[7] P. Henderson and N. De-Silva. A narrative approach to collaborative writing: A business process model. In *8th International Conference on Enterprise Information Systems (ICEIS)*, Cyprus, 2006.

[8] A.-M. Kermarrec, A. Rowstron, M. Shapiro, and P. Druschel. The IceCube Approach to the Reconciliation of Divergent Replicas. In *Proceedings of the ACM Symposium on Principles of Distributed Computing - PODC 2001*, pages 210–218, Newport, Rhode Island, USA, August 2001. ACM Press.

[9] A. Knott. *A Data-Driven Methodology for Motivating a Set of Coherence Relations.* PhD thesis, University of Edinburgh, 1996.

[10] P. B. Lowry, A. Curtis, and M. R. Lowry. Building a Taxonomy and Nomenclature of Collaborative Writing to Improve Interdisciplinary Research and Pratice. *Journal of Business Communication*, 41(1):66–99, January 2004.

[11] W. C. Mann and S. A. Thompson. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8(3):243–281, 1988.

[12] D. Marcu. *The Theory and Practice of Discourse Parsing and Summarization.* The MIT Press, 2000.

[13] P. Molli, H. Skaf-Molli, G. Oster, and S. Jourdain. SAMS: Synchronous, Asynchronous, Multi-Synchronous Environments. In *Proceedings of the Conference on Computer-Supported Cooperative Work in Design - CSCWD 2002*, pages 80–85, Rio de Janeiro, Brazil, September 2002.

[14] S. Noël and J.-M. Robert. Empirical study on collaborative writing: What do co-authors do, use, and like? *Computer Supported Cooperative Work - JCSCW*, 13(1):63–89, 2004.

[15] G. Oster, P. Urso, P. Molli, and A. Imine. Tombstone transformation functions for ensuring consistency in collaborative editing systems. In *The Second International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2006)*, Atlanta, Georgia, USA, November 2006. IEEE Press.

[16] C. Rahhal, H. Skaf-Molli, P. Molli, and N. D. Silva. Semcw: Semantic collaborative writing using rst. In *The 3rd International Conference on Collaborative Computing:Networking, Applications and Worksharing - CollaborateCom 2007*, New York, USA, nov 2007.

[17] H. Skaf-Molli, C.-L. Ignat, Rahhal, and P. Molli. New Work Modes for Collaborative Writing. In *International Conference on Enterprise Information Systems and Web Technologies- EISWT 2007*, Orlando, Florida, jul 2007.

[18] H. Skaf-Molli, P. Molli, and G. Oster. Semantic Consistency for Collaborative Systems. In *Proceedings of the International Workshop on Collaborative Editing Systems - CEW 2003*, Helsinki, Finlande, September 2003.

[19] S. Staab and R. Studer, editors. *Handbook on Ontologies.* International Handbooks on Information Systems. Springer, 2004.

[20] C. Sun, X. Jia, Y. Zhang, Y. Yang, and D. Chen. Achieving Convergence, Causality Preservation, and Intention Preservation in Real-Time Cooperative Editing Systems. *ACM Transactions on Computer-Human Interaction*, 5(1):63–108, March 1998.

[21] S. G. Tammaro, J. N. Mosier, N. C. Goodwin, and G. Spitz. Collaborative Writing Is Hard to Support: A Field Study of Collaborative Writing. *Computer-Supported Cooperative Work - JCSCW*, 6(1):19–51, 1997.

[22] K. M. Tudor Groza, Siegfried Handschuh and S. Decker. SALT - Semantically Annotated LaTeX for scientific publications. In *4th European Semantic Web Conference (ESWC 2007).*, jul 2007.

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399