NeOn: Lifecycle Support for Networked Ontologies

Integrated Project (IST-2005-027595)

Priority: IST-2004-2.4.7 — "Semantic-based knowledge and content systems"

D3.3.5: Context-based matching revisited

Deliverable Co-ordinator: **Patrick Hoffmann**

Deliverable Co-ordinating Institution: **INRIA**

Other Authors: **Mathieu d'Aquin (Open university), Jérôme Euzenat (INRIA), Chan Le Duc (INRIA), Marta Sabou (Open university), François Scharffe (INRIA)**

Matching ontologies can be achieved by first recontextualising ontologies and then using this context information in order to deduce the relations between ontology entities. In Deliverable 3.3.1, we introduced the Scarlet system which uses ontologies on the web as context for matching ontologies. In this deliverable, we push this further by systematising the parameterisation of Scarlet. We develop a framework for expressing context-based matching parameters and implement most of them within Scarlet. This allows for evaluating the impact of each of these parameters on the actual results of context-based matching.

Document Identifier:	NEON/2010/D3.3.5/v1.0	Date due:	January 31st, 2010
Class Deliverable:	NEON EU-IST-2005-027595	Submission date:	January 31st, 2010
Project start date	March 1, 2006	Version:	v1.0
Project duration:	4 years	State:	Final
		Distribution:	Public

NeOn Consortium

This document is part of the NeOn research project funded by the IST Programme of the Commission of the European Communities by the grant number IST-2005-027595. The following partners are involved in the project:

Open University (OU) – Coordinator Knowledge Media Institute – KMi Berrill Building, Walton Hall Milton Keynes, MK7 6AA United Kingdom Contact person: Martin Dzbor, Enrico Motta E-mail address: {m.dzbor, e.motta}@open.ac.uk	**Universität Karlsruhe – TH (UKARL)** Institut für Angewandte Informatik und Formale Beschreibungsverfahren – AIFB Englerstrasse 11 D-76128 Karlsruhe, Germany Contact person: Peter Haase E-mail address: pha@aifb.uni-karlsruhe.de
Universidad Politécnica de Madrid (UPM) Campus de Montegancedo 28660 Boadilla del Monte Spain Contact person: Asunción Gómez Pérez E-mail address: asun@fi.ump.es	**Software AG (SAG)** Uhlandstrasse 12 64297 Darmstadt Germany Contact person: Walter Waterfeld E-mail address: walter.waterfeld@softwareag.com
Intelligent Software Components S.A. (ISOCO) Calle de Pedro de Valdivia 10 28006 Madrid Spain Contact person: Jesús Contreras E-mail address: jcontreras@isoco.com	**Institut 'Jožef Stefan' (JSI)** Jamova 39 SL–1000 Ljubljana Slovenia Contact person: Marko Grobelnik E-mail address: marko.grobelnik@ijs.si
Institut National de Recherche en Informatique et en Automatique (INRIA) ZIRST – 665 avenue de l'Europe Montbonnot Saint Martin 38334 Saint-Ismier, France Contact person: Jérôme Euzenat E-mail address: jerome.euzenat@inrialpes.fr	**University of Sheffield (USFD)** Dept. of Computer Science Regent Court 211 Portobello street S14DP Sheffield, United Kingdom Contact person: Hamish Cunningham E-mail address: hamish@dcs.shef.ac.uk
Universität Kolenz-Landau (UKO-LD) Universitätsstrasse 1 56070 Koblenz Germany Contact person: Steffen Staab E-mail address: staab@uni-koblenz.de	**Consiglio Nazionale delle Ricerche (CNR)** Institute of cognitive sciences and technologies Via S. Marino della Battaglia 44 – 00185 Roma-Lazio Italy Contact person: Aldo Gangemi E-mail address: aldo.gangemi@istc.cnr.it
Ontoprise GmbH. (ONTO) Amalienbadstr. 36 (Raumfabrik 29) 76227 Karlsruhe Germany Contact person: Jürgen Angele E-mail address: angele@ontoprise.de	**Food and Agriculture Organization of the United Nations (FAO)** Viale delle Terme di Caracalla 00100 Rome Italy Contact person: Marta Iglesias E-mail address: marta.iglesias@fao.org
Atos Origin S.A. (ATOS) Calle de Albarracín, 25 28037 Madrid Spain Contact person: Tomás Pariente Lobo E-mail address: tomas.parientelobo@atosorigin.com	**Laboratorios KIN, S.A. (KIN)** C/Ciudad de Granada, 123 08018 Barcelona Spain Contact person: Antonio López E-mail address: alopez@kin.es

Work package participants

The following partners have taken an active part in the work leading to the elaboration of this document, even if they might not have directly contributed to the writing of this document or its parts:

- Jožef Stefan Institute
- Open university
- iSoCo
- Universität Karlsruhe
- INRIA
- Ontoprise
- CNR
- Universidad Politécnica de Madrid

Change Log

Version	Date	Author	Changes
0.1	15.04.2008	Jérôme Euzenat	initial version
0.2	18.02.2009	Jérôme Euzenat	first framework version
0.3	27.10.2009	Patrick Hoffmann	merge with learning document
0.4	22.12.2009	Patrick Hoffmann	completion of Chapter 4
0.5	15.01.2010	Jérôme Euzenat	revision of the first part
0.51	18.01.2010	Patrick Hoffmann	partial correction of Chapter 3
0.6	27.01.2010	Patrick Hoffmann	correction of the second part
0.7	28.01.2010	Jérôme Euzenat	revision of executive summary, Chapter 1 and 2
0.8	3.01.2010	Jérôme Euzenat	added Figure 3.1, implemented comments from QC
1.0	10.02.2010	Patrick Hoffmann	implemented comments from QA

Executive Summary

If one considers that the context of an ontology can be given by its relations with other ontologies, decontextualisation is the operation that separates an ontology from its context and recontextualisation is the opposite operation that establishes relations between ontologies. Ontology matching can be used for putting ontologies in context, but, in the other direction, providing more context to ontologies can improve ontology matching.

Indeed, ontology matching can be achieved by first recontextualising ontologies and then using this context information in order to deduce the relations between ontology entities. In Deliverable 3.3.1, we introduced the Scarlet system, developed at The Open University, which uses ontologies on the web as context for matching ontologies. Other systems using similar techniques have been developed.

In this deliverable, we push this further by systematising the parameterisation of Scarlet. We develop a framework for expressing context-based matching parameters and implement most of them within Scarlet. The different dimensions along which Scarlet can be more parameterised are:

- The kind of ontologies to use: upper level ontologies, domain-dependent ontologies, any ontology;
- The way to select context ontologies ontologies: a priori, globally (e.g., maximum common words), locally (e.g., pair of words), glocally (maximum common words among the neighbours of a pair of word);
- The method used for matching concepts with the context ontology: string-based matching, synonyms, more elaborate matching technique;
- The type of relations between concepts to extract: asserted, inherited, consequences;
- The composition graph traversal: only when there is no ontology containing the common terms, always traverse, use the first traversal that works, use them all;
- The strategy for evaluating correspondences: select the first one, select them all;
- The aggregation operator for correspondences: confidence, conjunction, majority.

Each combination of these parameters defines a new context-based matcher. We have extended Scarlet based on the available NeOn contextualisation plugin and support tools. Our method provides several improvements over the initial prototype:

- parameterisation of the matcher;
- selection of ontologies to consider through various criteria (size, types, specific ontologies).
- integration within the Alignment API framework;

Our ambition was to evaluate the impact of parameters in this new matcher through a set of systematic experiments. Hence, we designed such experiments.

However, we have not been able to provide results from these experiments so far.

Contents

Chapter 1

Introduction

In this chapter, we recast the intuition presented in deliverables D3.1.1 and D3.3.1 in the context of networked ontologies. In particular, we explain in what sense contexts are networks of ontologies and how this can be used for defining a new kind of matchers: context-based matchers (§1.1).

Then we present the purpose of this deliverable as systematising the design of context-based matchers such that they can be parameterised along many dimensions and that the resulting configurations can be tested (§1.2). This is achieved through the redesign of the Scarlet matcher developed at Open university.

1.1 Contextualising ontologies through ontology matching

Domain ontologies are designed to be applied in a particular context and use terms in a sense that is relevant to this domain, e.g., *Ontology* in computer science, and which may not be related to similar concepts in other domains. They do not fully specify concepts because part of their specification is implicit from the context. We use here the word “constraint” for this specification in a broad sense: any axiom constrains the meaning of the terms it uses.

NeOn has in its core the ambitious scenario that ontologies are developed in an open environment in a distributed fashion. This involves that ontologies are used in settings that are not those that have led to their design. In order to use them properly, it is necessary to be able to identify this context so as to take the appropriate action, namely rejecting them or adapting them to the new context.

The context of some knowledge or ontology is given by additional knowledge in which or in the perspective of which this knowledge has been elaborated. This knowledge can vary in nature and expression. For instance, in work package 2, the context of elaboration of ontologies is expressed as argumentative structures about ontology design rationale. In work package 3 [Haase *et al.*, 2006], the context of ontologies is prominently placed in the framework of networked ontologies: the context of an ontology is given by the network of other ontologies with which it is related.

From this definition, a pair of dual operations can be associated with context (see Figure 1.1): contextualisation and decontextualisation. Contextualisation or recontextualisation is the action of finding the relations of an ontology with other ontologies which express its context. Decontextualisation, as the opposite, extracts one particular ontology from its context. These operations can be combined, for instance, if someone wants to transfer one ontology from a domain to another one, by first decontextualising it and recontextualising it to another domain.

In Deliverable 3.3.1 [Euzenat *et al.*, 2007], we have considered expressing context with two main constructions from the NeOn model for networked ontologies: Ontologies and mappings (or alignments). The NeOn model already offers the meta-model for these two kinds of entities. The Alignment API can be considered as an implementation of this meta-model.

Context-based matching contrasts with content-based matching. Content-based matching uses the description of the ontologies (axioms, concepts, properties, individuals, comments, labels, etc.) in order to match

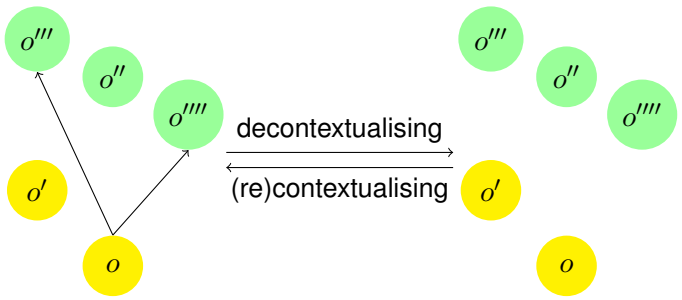

Figure 1.1: Contextualising/decontextualising ontologies in the framework of networked ontologies.

them. Context-based matching, for the same purpose, uses the context of these ontologies (resources that they annotate, message exchanges between agents that use them, etc.).

In Deliverable 3.3.1, we introduced the link between context-based matching and contextualisation. In such a setting, one uses an intermediate ontology to contextualise the two ontology to be matched. For instance, when trying to find the relation between “Beef” in the Agrovoc terminology and “Food” in the NAL terminology, it is possible to recontextualise both ontologies in the TAP ontology. This ontology provides two concepts called “Beef” and “Food” which can be matched with the initial ones. It also provides a relationship between them: “Beef” is a specialisation (subclass) of “Food”. This relation can be considered as a correspondence between these terms. This principle has been implemented in a system, Scarlet. This is recalled in Chapter 2.

1.2 Extending Scarlet

Scarlet is at the crossroad of these two trends: it is an ontology matcher which operates by contextualising ontologies with ontologies that can be found on the web. But Scarlet is more complex than this definition since it involves selecting ontologies for context, matching entities from the initial ontologies and those of the context, composing the relations obtained after matching, etc.

So the goal of this deliverable is to redesign Scarlet by making the various options explicit and offer the possibility to parameterise the system, and to test the various parameter combinations so as to evaluate their impact. We proceed with the following steps:

- Determine atomic parameters that constitute these configurations,
- Develop a version of Scarlet that can be configured to adopt some most significant variations of the context-based matching approach,
- Design experiments for a few real world cases to assess the advantages of each configuration,
- Compare the approach with other ontology matching approaches to establish its strengths and weaknesses, and suggest conditions in which the context-based approach, with some specific combinations of parameters, could be a commendable alternative to traditional matching approaches.

A brand new version of Scarlet has been developed for this deliverable, for the purpose of assessing the validity of the context-based matching approach, notably by evaluating its performance under numerous configurations. It also includes the possibility to launch a series of tests by launching the matcher several times with configurations automatically generated. Configurations consist each of distinct parameter combinations, and their generation is based on a configuration file that defines the limited set of values that each parameter can take.

1.3 Outline of the deliverable

In Chapter 2 we briefly summarise the state of the art in context-based ontology matching. Then we introduce the framework that allows for parameterising a system like Scarlet (Chapter 3). The implementation principles of the new version of Scarlet are presented in Chapter 4. Chapter 5 explain the design of experiments of combinations of parameters.

In addition, Appendix A describes how to use the resulting system.

Chapter 2

State of the art

We cover here briefly the state of the art in two directions: context-based matchers (§2.1) and their evaluation (§2.2).

2.1 Context-based matchers

There have been various experiments, from several teams, of the idea of recontextualising ontologies in order to match them better. This section complements and updates Deliverable 3.3.1 [Euzenat *et al.*, 2007]. We present them depending on the type of ontology which is used, e.g., top-level ontology, specific purpose ontology, any ontology, and not in chronological order.

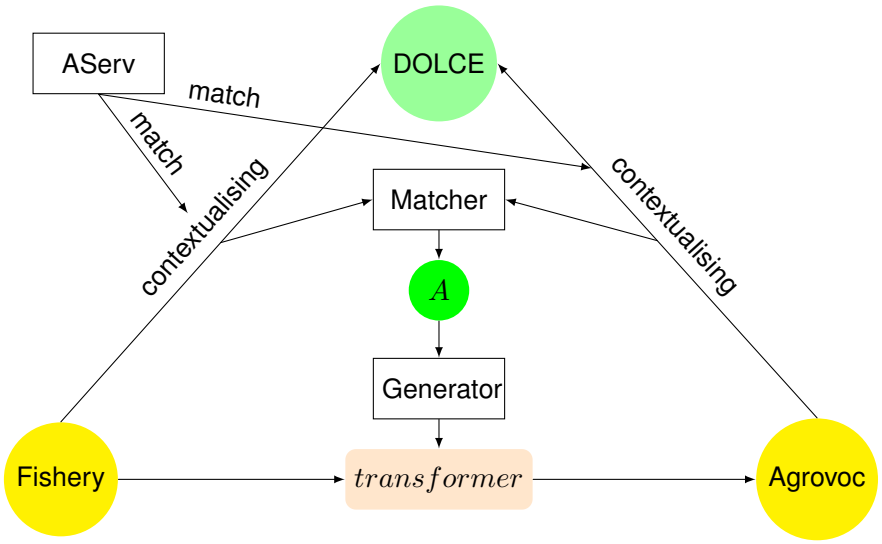

Figure 2.1: Simple recontextualisation using Dolce

2.1.1 Manually contextualising with upper-level ontologies

In the Wonderweb project, Aldo Gangemi used the upper-level ontology DOLCE in order to provide external definitions to resources from FAO such as the Fishery database and Agrovoc [Gangemi, 2004]. It has been proposed in NeOn that we could use the alignments which lack definitions in order to help with matching the resources (see Figure 2.1). Matching would be facilitated because the anchoring to the upper-level ontology makes it possible to know whether two entities are in exclusive branches or in similar ones.

2.1.2 Recontextualisation through specific ontologies

Zharko Aleksovski [Aleksovski *et al.*, 2006] pioneered the use of various kinds of background knowledge in ontology matching. In particular, he used a specific medical ontology (DICE) to match lists of controlled terms in Dutch hospitals (OLVG and AMC hospital). The process goes in two steps:

- Recontextualising the list of terms (and the words comprised in the terms) is achieved first with simple lexical matching (and also an available alignment).
- Specific reasoning is used for obtaining and composing the relations between matched terms. This can be that the two terms are in the ontology and a relation between them can be obtained or this can be through a more explicit reasoning, based, for instance, on the relations between words composing the term, e.g., if crisis is more specific than problem and asthma more specific than respiratory affection, then asthma crisis is more specific than respiratory problem.

He also showed that using several ontologies instead of only one, would improve the results.

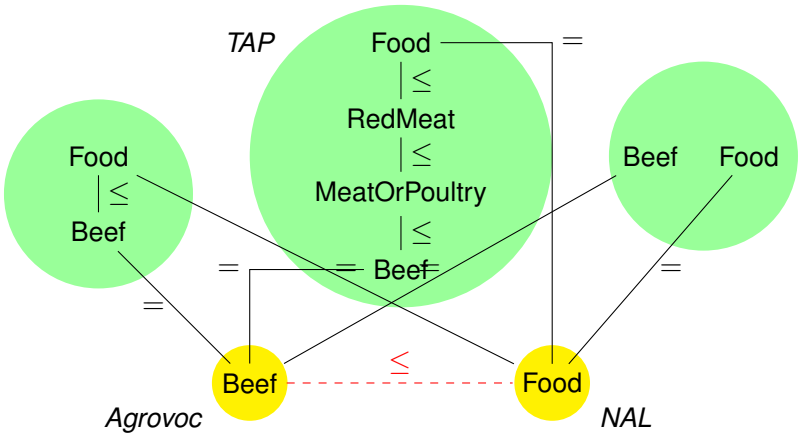

Figure 2.2: Scarlet example: several results are returned and must be aggregated.

2.1.3 Recontextualisation with all the ontologies of the web

Since using more ontologies improves the results, why not using all the ontologies of the web? After all, Google solves the problem raised by the huge amount of data by taking advantage of this huge amount of data. Here, the problems raised by the heterogeneity of ontologies is solved by taking advantage of these many heterogeneous ontologies. This is the rationale behind Scarlet [Sabou *et al.*, 2006; Sabou *et al.*, 2008a; Sabou *et al.*, 2008b] which is based on the following principles:

- Using the ontologies on the web as context. Finding them requires crawling the web. This is the reason why Watson was developed (Scarlet started with Swoogle but it was too limited);
- Composing the relations obtained through these ontologies: this covers reasoning within the ontology for deducing the relations between entities or reasoning across ontologies (see Figures 2.3 and 2.2).

In more details, Scarlet processing roughly consists of the following steps:

1. Harvest ontologies on the web with Watson;
2. Select those which are related to the ontologies to match: usually this is achieved by selecting, for each pair of named entities, the ontologies which contain both names;
3. Match them with the found ontologies: here Scarlet uses simple string equivalence;

4. Compose the relations between entities through the intermediate ontologies: this is usually returning the relation found in the ontology (see Figure 2.3);

5. Aggregate the obtained results (see Figure 2.2): Scarlet stops as soon as it has a relation).

Another implemented variation was, when no ontology would contain the pair of terms, to look for ontologies containing each of these terms and bridges between these ontologies (see Figure 2.3). This can become a very complex procedure so it is restricted to finding, for each pair of ontologies, the intersection between the entities subsuming one term and those subsumed by the other, which helps to quickly find subsumption relations (see Figure 2.3).

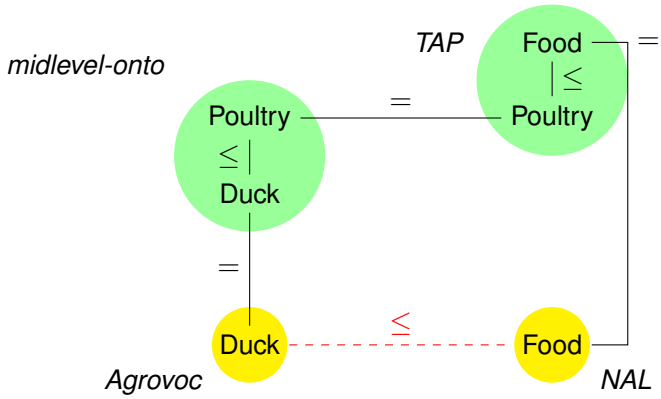

Figure 2.3: Scarlet composition example.

One of the beneficial aspects of these procedures is that they work for matching resources without any structure, like the list of terms in the first example. They may even be used with only one ontology (or simple resource) in order to find new relations within it.
Deliverable 3.3.1 identified:

- Mapping within one ontology, subdivided in two variants:
  - stopping at the first correspondence found;
  - finding all correspondences and combining them (hereby called aggregation);
- Cross-ontology mapping (hereby called composition);

2.2 Evaluation of context-based matchers

Here we do not consider the general evaluation of matchers as developed in the Ontology Alignment Evaluation Initiative [Euzenat *et al.*, 2009]. Two evaluations of Scarlet have been carried out within OAEI 2008 [Caraciolo *et al.*, 2008] as the Spider system involving a special anchoring technique developed by Jorge Gracia [Sabou and Gracia, 2008].
Instead, we focus on experiments which compare various modalities of context-based matchers.

2.2.1 The initial Scarlet experimentations

Deliverable 3.3.1 reported on the evaluation of three variants of Scarlet against Agrovoc (FAO) and NAL (US DoA). The considered variants were:

S1 works with only one intermediate ontology: it retrieve the ontologies covering both candidate terms and delivers all the correspondence that it finds between matched concepts;

S1' is like S1 but it stops at the first correspondence that it finds;
S2 implements composition in the graph of ontologies, but only through direct superclasses (and no subclasses).

In all cases, the search for anchors was provided by strict string matching on terms as bags of words and candidate ontologies were provided by Swoogle.

Because of the lack of a full reference alignment in the data set, results were manually assessed and only reported on precision. They provide an average value of 70% precision. This is expected with the given anchoring strategy, indeed, anchoring with string equivalence usually provides high precision. However, this is rather good given that Scarlet returns subsumption relations.

2.2.2 The uo_match experiment

[Mascardi *et al.*, 2009] compared the use of several upper-level ontologies for context-based matching. They have developed their own system, uo_match, and compared three configurations:

- uo_match which combines several terminological matchers (including using Wordnet),
- structural_uo_match which composes the relationships obtained by the same matchers against an upper level ontology, and
- mixed_match which simply combines both approaches,

and parameterised it with three different upper-level ontologies: OpenCyc, SUMO-OWL and DOLCE.

The tests have been run against a collection of ontologies with a reference alignment designed by hand.

As expected, results show that structural matching increases recall. In particular, this is true when the upper-level ontologies are OpenCyc or SUMO-OWL. In this case, precision is not significantly degraded, while DOLCE seems too small for preserving this result. The mixed approach also increases recall while not significantly decreasing precision whatever upper-level ontology used.

2.3 Synthesis

Context-based matching is not new but it has been developed in a relatively ad hoc manner. The goal of the present work is to decompose such methods into the features they are made of and develop a fully customisable matcher.

A secondary objective in doing this is to be able to run tests depending on the various parameters in order to isolate those parameters having the most influence.

Chapter 3

Framework for parameterising context-based matching

The literature demonstrates that networked ontologies can be put into context in many conceivable ways. Possibilities for variations of the context-based matching approach are so numerous that a framework is required to evaluate them rigorously. Among them are:

- the kind of ontologies that will be considered: upper-level ontologies, domain-dependent ontologies;
- the method to choose ontologies to anchor concepts: comparison of the maximum common terms between ontologies, of the concepts local names;
- the method to contextualise: string-based, lexicon-based, or based on ontology-alignment;
- the method to aggregate distinct results: confidence, conjunction, majority.

Each combination of answers to these questions defines a new context-based matcher. The purpose of this chapter is to provide a framework in which these parameters fit naturally.

This chapter presents some definitions (§3.1), as well as an overview of a general algorithm (§3.2) and of its parameters (§3.3).

3.1 Definitions

For this work, we understand the context of an ontology to be the network of other ontologies which describe some comparable knowledge (Deliverable 3.3.1). We therefore denote by *context-based matching* the process of finding an ontology alignment based on the knowledge described in such ontologies.

Context-based matching involves seven main processes: *ontology arrangement*, *ontology selection*, *ontology contextualisation*, *exploration of paths in ontologies*, *path selection*, *composition*, *aggregation* and *extraction* of alignments. Figure 3.1 provides an illustration of these different steps.

We define precisely these processes here:

Definition 1 (Ontology arrangement) Ontology arrangement *is the organisation of ontologies into a sequence, which establishes the order in which to contextualise and explore ontologies. The process takes as input ontologies as well as the maximum number of intermediary ontologies desired.*

Definition 2 (Ontology selection) Ontology selection *aims at acquiring the ontologies that serve as intermediary ontologies. It involves searching and retrieving ontologies that are good candidates to serve as intermediary ontologies, by evaluating the knowledge that they describe.*

Definition 3 (Contextualisation) Contextualisation *consists of aligning an ontology with other ontologies to find the relations that express its context; it results in a set of correspondences, or alignment. The comparable process of seeking in other ontologies all concepts equivalent to a given concept, is called* anchoring.

Definition 4 (Path exploration) Path exploration *consists of traversing an ontology to retrieve all local paths (lists of semantic relations) by which two sets of concepts can be connected. It takes as input an ontology and two sets of its concepts, and returns a set of local paths.*

Definition 5 (Path selection) Path selection *involves forming a path between two concepts by concatenating various local paths from distinct ontologies, such that the concept at the end of each local path is anchored to the concept at the beginning of the next local path. It therefore takes as input a list of local paths, and returns a single path.*

Definition 6 (Composition) Composition *of the path relations, reduces mathematically the relations along a path to one relation or set of relations holding between the two extremities.*

Definition 7 (Aggregation) Aggregation *involves combining various alignments obtained from distinct sources. It can be used for either combining several relations into one single relation or to choose among several possible relations the one to retain. For that purpose, external considerations, such as trust in sources, may be used.*

3.2 Algorithm

With these definitions, we can define a general algorithm as illustrated in Figure 3.2.

Let O and O' be two ontologies to be matched using Scarlet.

The first *ontology arrangement* consists of ordering the possible intermediate ontologies. This is usually achieved by using preferences (upper-level ontologies, ontologies on a particular domain, etc.). Then the *ontology selection* selects those ontologies that contain some knowledge comparable to the knowledge described in both O and O'. Then, O and O' are *contextualised* with the selected ontologies $S_{(O,O')}$, by applying an ontology matcher multiple times between O and $S_{(O,O')}$ and between O' and $S_{(O,O')}$. The matching can also be done by *anchoring* concepts of O and O' to any concept in $S_{(O,O')}$. For each of the ontologies $S \in S_{(O,O')}$, we will have to connect the concepts $s \in S$ that have been anchored to any concept in $o \in O$ with concepts $s' \in S$ anchored to any concept $o' \in O'$. This is done by examining whether there exist a sequence of semantic relations from s to s', by *path exploration*. Now, there might be in S many paths that relate the two given concepts $o \in O$ and $o' \in O'$. We might therefore have to do some *path selection*, to limit ourselves to one path.

Using more intermediary ontologies is achieved by iterating this process as long as possible (usually parameterised by a maximum depth). The iteration will match one of the ontology to match to a supplementary ontology and this new ontology will be contextualised to the intermediary ontology. These new anchors allow for extracting new paths between concepts of the two initial ontologies.

Once all this is done, we have to summarise all paths by *composition* into the shortest possible equivalent lists of semantic relations. There may be different semantic relations (or list of semantic relations) that relate two concepts $o \in O$ and $o' \in O'$, that have been retrieved by the means of distinct intermediary ontologies or ontology sequences; they have to be compared and *aggregated* to decide which list of semantic relations is the more likely.

3.3 Categories of parameters

We identified four types of parameters:

Strategies identify in which order the different parts of the algorithm may be organised. For example, ontology selection can be done once for the whole context-based ontology alignment, or it can be done each time one searches to relate two concepts.

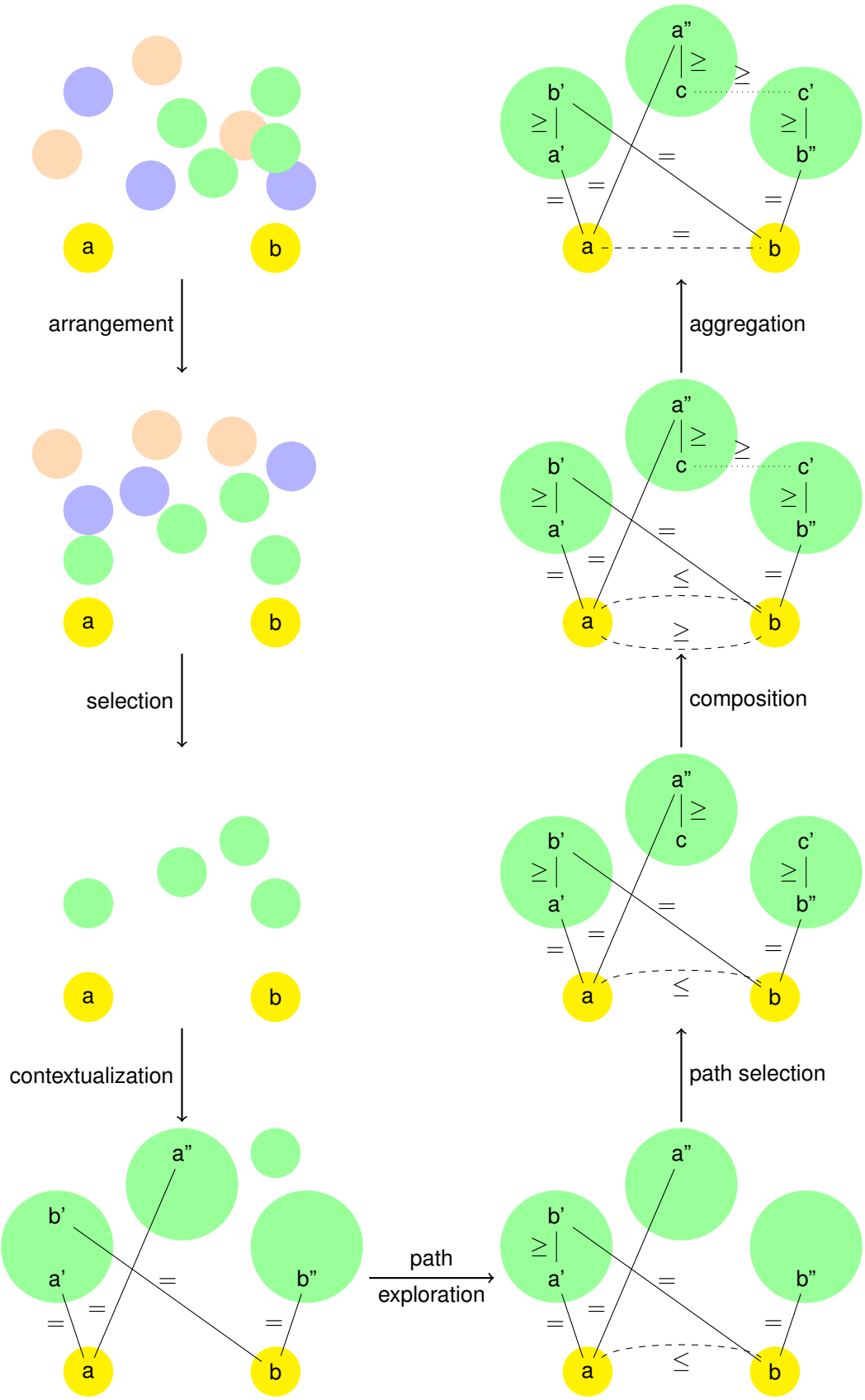

Figure 3.1: Example of the different steps of context-based matching.

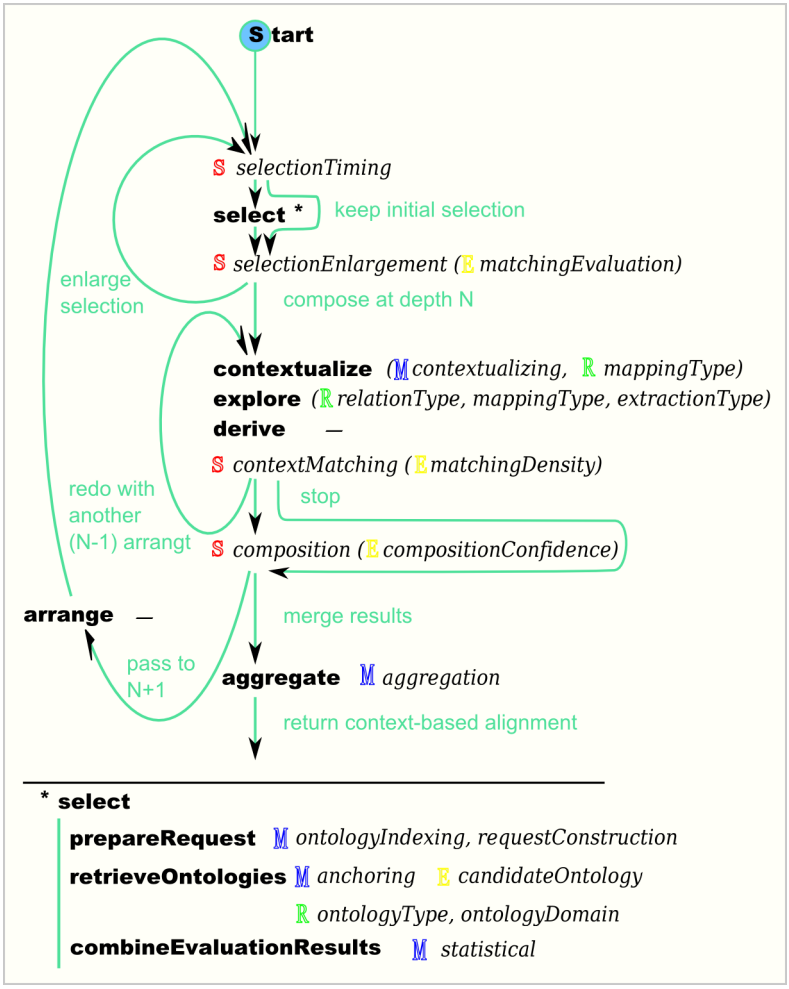

Figure 3.2: A general context-based matching algorithm. This diagram shows the ordering of the different steps of the algorithm. Arrows express the possible transitions and parameter values changing along these transitions. The type of parameters, (S)election method, other (M)ethod, (R)elation type, (E)valuation threshold, is displayed with a different letter prefixing the parameter.

Type restrictions are restrictions made on the data manipulated. For example, only some type of semantic relations might be considered when exploring the ontologies

Methods are concrete algorithms to accomplish each of the steps defined in §3.1. For example, anchoring may be done by strict label matching, using the alignments available on an alignment server, or by a specific ontology matcher.

Evaluations establish how to make qualitative judgments. They are often based on methods, and serve to help some strategies decide what behaviour to adopt.

The various approaches discussed within §2.1 can be characterised with respect to a few value parameters as illustrated in Table 3.1.

Option	Matching method		Ontology selection type restriction			Composition Method	
Approach	based on concepts local names	based on token comparison	upper ontologies	specific ontologies	ontologies on the web	functional (equivalence only)	relational
Contextualising with upper-level ontologies (§2.1.1)	X	-	X	-	-	X	-
Contextualising through specific ontologies (§2.1.2)	X	-	-	X	-	-	X
Contextualising with any ontology on the web (§2.1.3)	X	X	-	-	X	-	X
Scarlet 2.0	X	X	X	X	X	-	X

Table 3.1: Characterisation of approaches presented in section 2 in terms of framework parameters. Only parameters that differ among these systems are presented in the table.

3.4 An additionnal note about alignments

Recontextualising consists of providing alignments between the resources to be matched and the context ontologies. If one considers that alignments could be made available on the web, for instance, through alignment servers, then the whole context-based matching simply consists of composing alignments between the ontology to match and the context ontologies with several intermediate ontologies if needed. Moreover, depending on the kind of inference used within the ontologies, this corresponds to full composition or to the "reasoning from alignments alone" introduced in Antoine Zimmermann's thesis [Zimmermann, 2008].

3.5 Synthesis

We have shown that context-based matchers can be characterised by a series of processes instantiated through different parameters. Existing context-based matching approaches can be characterised with regard to these parameters. There are, however, many more possible parameter values that will be illustrated in the next chapter. The next chapter will consider how to reengineer Scarlet with regard to the proposed parameters.

Chapter 4

Implementing the framework

We describe how Scarlet has been redesigned in order to implement the framework introduced in the previous chapter. Section 4.1 describes the actual implemented algorithm. Section 4.2 provides the extensive list of proposed parameters, while §4.3 indicates some necessary optimisations.
A tutorial to demonstrate how to make the call within various implementations is presented in Appendix A.

4.1 Context-based matcher

The algorithm depicted in Figure 4.1 is a refinement of the one of Figure 3.2.
Contrary to the initial Scarlet implementation, the new one performs the process with whole ontologies at a time: ontologies are selected and arranged into ontology sequences, and are contextualised with one another, according to these sequences; the aim is to relate any concepts of the ontologies to match, passing by the anchors previously found as an articulation of the ontologies selected, and to explore for paths between them. After a selection of the best path to relate two concepts through an ontology sequence, the paths are transformed into alignments by composition. The semantic relations produced are aggregated to form one single alignment, which may be extracted into a smaller alignment that summarises it.

4.2 Implementation of parameters

4.2.1 Ontology arrangement

Ontology arrangement strategies

Decide under which conditions to stop the context-based matching :

- stop when the alignment of the two ontologies O and O' is satisfying (see 4.2.1)
- stop when there is evidence that there is no possible alignment between the two
- stop when all has been done (complete).

Decide under which conditions to use more than one intermediary ontology:

- never
- compose until a given maximum number of intermediary ontologies

Ontology arrangement evaluations

Evaluate the quality of the context-based ontology matching:

- by proportion of concepts matched in O and O'

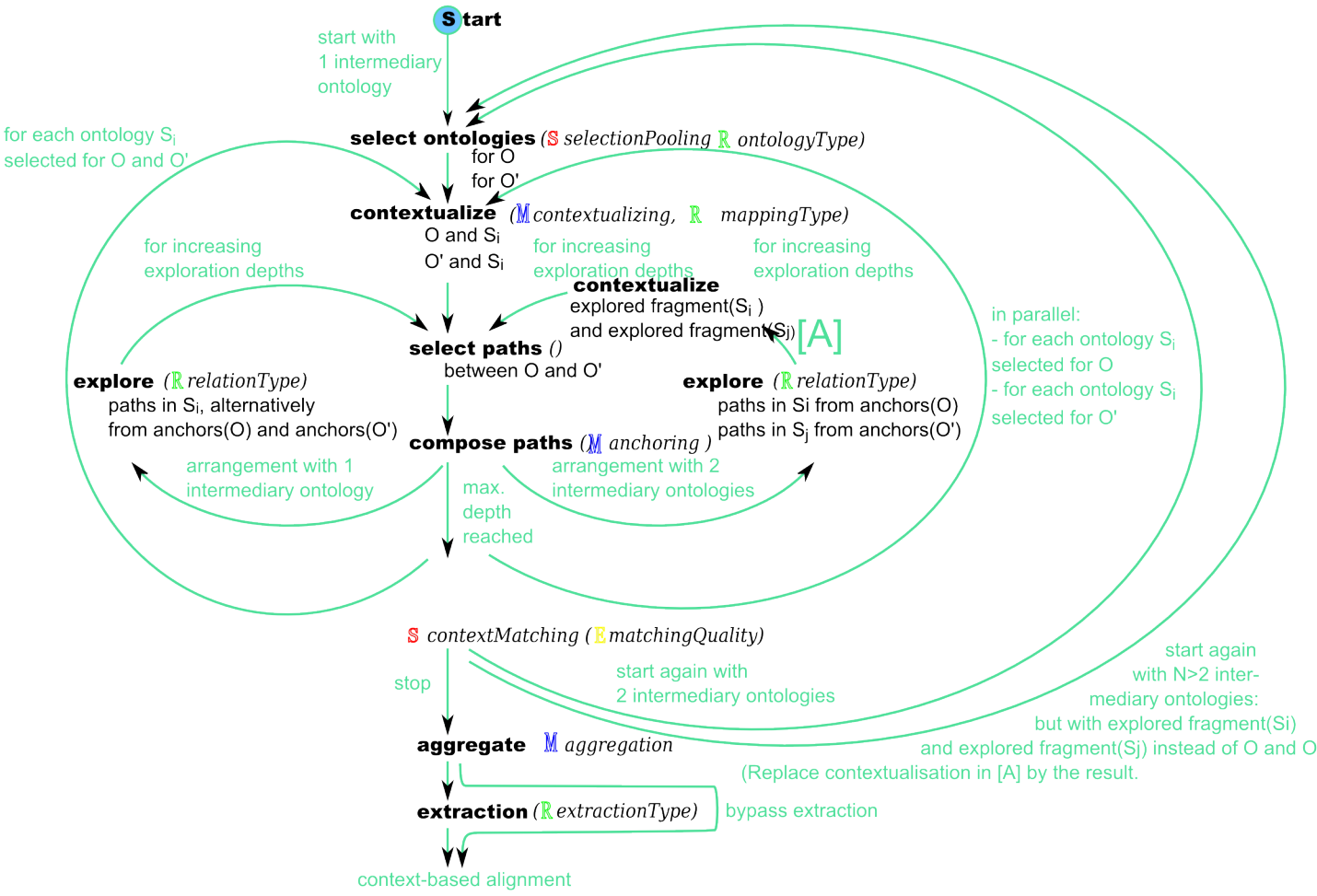

Figure 4.1: The more precise algorithm implemented in Scarlet 2. The transitions are basically those of the general algorithm of Figure 3.2 but their control conditions are made explicit. The path exploration is also highlighted through the inner loops exploring possible new intermediate ontologies.

- by density and homogeneity
- by proportion of concepts matched in O
- by proportion of concepts matched in O'

Estimation of the gain of increasing the depth of the composition from N to $N+1$, to help the decision to do it or not

- by proportion of concepts matched in O and O'
- by proportion between the concepts matched at the levels N and those matched at level $N-1$,
- by the pertinence of mappings found at level N
- by the confidence of mappings found at level N

4.2.2 Ontology pooling

We call ontology pooling a selection done beforehand, which is used to restrict the ontologies that will be used, depending on information specific to the context-based matching process.

Ontology pooling strategies

This parameter serves to decide whether the ontologies should be selected from a pool, or not, and in the case a pool is used, how this pool should be constituted. Values can be any of the following:

- No pooling
- The pool is created at the beginning, and not changed afterwards. It is constituted from the ontologies selected from the two original ontologies that have to be context-matched
- The pool is created for the two original ontologies first, and then is renewed (restricted) at each new ontology matching, to serve as pool for the following anchoring.

Ontology pooling methods

This parameter serves to offer various methods to select the ontologies that will serve as intermediary ontologies. Values can be any of the following:

- Select for the pool all ontologies where there is at least an anchor for one concept of any of the two ontologies to match
- select for the ontology pool that have more than a customisable minimum number of anchors for distinct concepts in the ontologies to match
- Use ontology distance measures to determine the ontologies that will serve for the ontology pool

4.2.3 Ontology selection

Ontology selection consists in choosing from the pool (when it exists) ontologies that will serve to anchor a given concept.

Ontology selection method

This parameter serves to offer a few methods to select ontologies in which to find anchors for any given ontology concept. Values can be any of the following:

- Select all ontologies for which an anchor exists for the concept
- Select ontologies for which an anchor exist for the concept, and that have a (customisable) minimum of anchors for distinct concepts from the neighborhood of the concept to anchor. One may keep only a (customisable) number of such ontologies, for which the number of anchors is the highest.

Ontology selection type restriction

This parameter serves to restrict the type of ontologies to select as intermediary ontologies. Values can be any of the following:

- Use all ontologies of the Web (no restriction of type)
- Use only ontologies that belong to the type given; ontologies need to be pre-defined, among for example, upper ontologies, domain dependent ontologies(such as medical, biological, competencies), popular ontologies, recommended ontologies, or any customised set of ontologies.
- Use only ontologies that belong to the domain given
- Use only ontologies that belong to the given selection
- Use only small ontologies
- Use only medium ontologies
- Use only large ontologies (considered to be upper ontologies)

4.2.4 Ontology contextualisation

Ontology matching is done either with an ontology matcher, or by searching for all concepts of a source ontology, all the anchors that they have in the ontology to match.

Ontology contextualisation methods

This parameter serves to offer a few methods to match ontologies. Values can be any of the following:

- anchoring method, where concepts are selected as anchors when they have the same URI as a concept in the source ontology.
- anchoring method, where concepts are selected as anchors when they have the same local name as a concept in the source ontology.
- anchoring method, where concepts are selected as anchors when they have a local name judged similar to the local names of a concept in the source ontology. The string-based similarity measure is customisable, as well as the threshold below which the concepts are not retained as anchors. Only the first (customisable) concepts with the highest similarity are kept.
- anchoring method, where concepts are selected as anchors when at least one word from their local name is contained in the local name and label of the concept in the source ontology.
- use an external matcher to match source and target ontologies.

Restriction of relation type for Ontology contextualisation

Restriction on the type of relations taken into account for the ontology alignments that serve as source, among:

- equivalence
- subsumption and inverser subsumption
- any named relation

4.2.5 Path exploration

Ontology exploration methods

This parameter serves to offer a few methods to explore ontologies from two sets of anchors. Values can be any of the following:

- relate an anchor to any other concept with the first path found
- relate an anchor to any other concept with the path that contains the preferred relations, by considering the weight of relations
- relate an anchor to any other concept by all possible paths

Ontology exploration strategies

This parameter serves to offer a few methods to match ontologies. Values can be any of the following:
When to stop exploration between two sets of concepts in an ontology

- do full exploration until the number of paths reaches a given length
- the same, but when at a given path length, some paths are found between two concepts for any given number of relations, stop searching to relate them by longer paths
- the same, but when an anchor is connected by a path, stop to search for any other path to connect this anchor.

Restriction of relations types considered for path exploration

Restriction on the type of relations taken into account when connecting two concepts with one another.

- equivalence (similarity value is not taken into consideration)
- subsumption and inverser subsumption
- any named relation

4.2.6 Path selection

Path selection Evaluation

Evaluation of the confidence that one can have in a path based on:

- quality of the ontologies traversed
- confidence in composition
- quality of the alignments used
- confidence value of mappings used
- similarity value of mappings used

4.2.7 Composition

Composition methods

Compose relations with one another (or not, depending on their type)

- Functional: the simple composition of equivalence relations (which always provide equivalence as a result),
- Relational: the mathematical composition of relations, e.g., $>$ composed with $=$ is $>$,
- Structural: the composition is not restricted to ontology language relations, but can take advantage of domain relation, e.g., is-part-of composed with $=$ yields is-part-of.

4.2.8 Aggregation

Aggregation methods

When the composition of the various paths between two concepts $o \in O$ and $o' \in O'$ return distinct semantic relations (or list of semantic relations), how to compute a unique semantic relation that is the synthesis of all?

- all results are supposed to be true (logical conjunction)
- at least one result should be true (logical disjunction)
- the semantic relation that comes from the path in which there is the greatest confidence is selected (confidence based)
- the semantic relation on which the most of results agree (popularity based)

4.2.9 Extraction

Extraction methods

Extract a minimal alignment from a series of relations between concepts.

- None: no treatment is made, the correspondences are returned as one alignment;
- Stability: it is not possible to expand the alignment (through composing with its inverse);
- Consistency: a consistent alignment is extracted from the set of correspondences;
- Minimal conflict: in addition the consistent alignment should be maximal.

4.3 Optimisations

The initial version of Scarlet was highly network-bound, due to calls to Watson web services. For example, with the previous version of Scarlet, launching the “path searcher"’ RelationComposer v1.3[1], the time taken for benchmark 101 compared to 202 was 7 hours.

By accessing a local server, we expect to gain a factor 2 of performance. We chose to install a Watson server at INRIA with its indexes for the tests, but we encountered some technical problems that are not yet solved. Simply having the service and the indexes here would also have the advantage of letting us compare various solutions against a same data set.

Here are some of the improvements done, that should improve the global performance:

4.3.1 Breadth-first contextualisation and search

The initial version of Scarlet was designed for finding the relation between two given concepts, through one or two intermediary ontologies, instead of matching two full ontologies. To match two ontologies of respectively N and M concepts, using this initial version, we had to try to launch the “path searcher" $N * M$ times. This version performs a breadth-first contextualisation and exploration, and we expect that because of this reason, it should be about $(N + M)/2$ times faster, when having to match medium and large ontologies.

4.3.2 Cache implementation

We implemented a cache solution, to reduce the calls to Watson. Performance is also notably increased: with the cache, we gain more than a factor 6.

[1] one of the two classes offered at the time in Scarlet, to relate two given concepts

4.3.3 Concurrency

We also made use of concurrency, and the preliminary results show a gain of a factor 2 when testing on a bi-processor machine, with 3 threads.

4.4 Synthesis

We have seen how Scarlet can be configured and presented the considered parameters with their possible values. Appendix A explains how the Scarlet 2.0 implementation can be used and how to pass parameters to it as well as how to package it for testing.

Some of the parameters above have not yet been implemented. This is the case for all evaluation measures, Ontology arrangement strategies, aggregation and extraction methods, restriction of relation type for ontology contextualisation, and anything about path selection. For some parameters, all options have not been implemented. This is the case for the selection of ontologies with an ontology distance (method), for the restriction of ontology type by an ontology domain (which requires to fill first the URI of the ontologies of the various domains), for contextualising ontologies with an external matcher (method), for exploring paths in ontologies by exhaustive exploration (method), and for functional and structural composition methods. For each of these non implemented values, the matcher uses the default value for the parameter.

In the next chapter, we will describe how configurations of Scarlet, made of a combination of these parameters, could be tested.

Chapter 5

Experiments

A benefit of having a parameterised matcher is to be able to evaluate the various configurations of this matcher and to determine in which context they perform well. We describe below a series of tests that can be used for evaluating these parameters. Instead of considering a systematic sampling of the parameters, we propose two series of tests.

First, we consider generic trials (§5.1) aiming at testing the context-based matcher for various typical usages of ontology matching. Then, we choose some specific groups of configurations to be tested one-to-one (§5.2). These comparisons aim at confronting specific combinations of parameters that describe current “extreme" context-based matching approaches, and are expected to show significant disparity in performance.

These experiments have to be performed against pairs of ontologies to match and the resulting alignments are evaluated against a reference alignment for providing precision, recall and processing time.

5.1 Trials with specific assignments

For each of the experiments described below, the ontologies to match and the set of ontologies available to help the context-based alignment may vary. Generally speaking, one should have different categories of matched ontology size : small (≈10 concepts), medium (≈100 concepts) and 1-2 larges (>500 concepts).

Specific kinds of background ontologies may be filtered among a few categories, to see whether it has an influence on the results.

We give for each trial the ontologies used, the rationale behind it, the configuration used, and possibly discuss a few questions.

5.1.1 Best current output

This trial aims at testing the context-based matching approach in the “best possible conditions” available, to get a good idea of what to expect, especially for recall.

We therefore consider a domain where Watson has indexed many ontologies. The medical domain seems adequate, all the more as labels representing similar concepts do not vary much from an ontology to another. We chose the anatomy track of OAEI, version 2009.

We expect therefore that anchoring methods will do pretty well, but the choice of the label comparison method will be decisive. In particular, the recall is likely to be the best among all trials, for a comparable precision.

5.1.2 Regularity of the matcher

This trial aims at testing the regularity of the context-based matcher, and comparing it to the one of matchers based on other approaches.

The ontologies used as input are taken from the 2009 version of the benchmark from the Ontology Alignment Evaluation Initiative[1]. This trial consists therefore of about 60 alignments to perform between an ontology and mostly its variants.

We expect that the trial will almost completely fail due to the structure of the data set: either the labels have not been altered and a string-based matcher will work correctly or they have been altered and there is not much chance that the context-based matcher can find proper anchors. The results will probably be much better in the cases where the results of other ontology matchers are available for the context-based matcher to use (on the alignment server). In such a case, it would be interesting to note the result of the various aggregation strategies.

5.1.3 Matching with the help of upper ontologies

This trial aims at testing the relative impact of the type of ontologies used. A second advantage of this trial is that we are able to confront the results with other experiences of alignment by the means of an upper ontology.

It seems interesting here to reproduce the kind of setting that others have used. We therefore propose to reproduce here the experiments from Section 2.2.2. We select only the three upper ontologies that gave the best results: OpenCyc[2] (26,965 concepts), DOLCE[3] (242 concepts), and SUMO[4] (4,393 concepts). As for the alignments, we kept only those whose ontologies were still online in January 2010, and either described in OWL or in RDF(S). To compare with the usual context-based approach, we plan to perform also all alignments with any ontologies, and with only upper ontologies (but all together). The experiments consist of matching the ontologies (between parentheses the number of concepts):

test1 Ka[5] (96) and bibtex[6] (15),

test2 Biosphere[7] (88) and Top-Bio [8] (65),

test4 Restaurant[9] (164) and Food[10] (65),

test6 Travel[11] (84) and Vacation[12] (32),

test7 Resume [13] (167) and Agent[14] (130),

test8 Resume[13] (167) and HL7_RBAC[15] (60),

test10 Vertebrate[16] (19) and TopBio[8] (65).

Our expectation is that the average precision should be higher, and the standard deviation between the precision results for all alignments should be smaller. As it is unlikely that upper ontologies may have corresponding concepts for most concepts in the ontologies to align, the distribution of correspondences is likely

[1] cf. http://oaei.ontologymatching.org/2009/benchmarks/
[2] http://www.cyc.com/opencyc
[3] http://www.loa-cnr.it/DOLCE.html
[4] http://www.ontologyportal.org/
[5] protege.cim3.net/file/pub/ontologies/ka/ka.owl
[6] oaei.ontologymatching.org/2004/Contest/304/onto.rdf
[7] sweet.jpl.nasa.gov/ontology/biosphere.owl
[8] www.co-ode.org/ontologies/basic-bio/top-bio.owl
[9] guru-games.org/ontologies/restaurant.owl
[10] silla.dongguk.ac.kr/jena-owl1/food
[11] lsdis.cs.uga.edu/projects/meteor-s/downloads/Lumina/ontology/travelontology.owl
[12] www.guru-games.org/ontologies/vacation.owl
[13] http://statistic.gunadarma.ac.id/research/WorkGroupInformationSystem/DownLoad/onto_colection/resume.owl
[14] 212.119.9.180/Ontologies/0.2/agent.owl
[15] lsdis.cs.uga.edu/projects/meteor-s/wsdl-s/ontologies/HL7_RBAC.owl
[16] www.co-ode.org/ontologies/basic-bio/basic-vertebrate-gross-anatomy.owl

to be heterogeneous, with many alignments in some reduced portions of the ontologies. We expect that the recall and precision will be higher on average when considering all upper ontologies at the same time, but that at the same time the performance will be poorer.

5.1.4 Sensitivity to context

This trial aims at testing whether the context-based matching actually “recontextualises", that is, whether the approach is good to differentiate concepts whose name is similar but that are used in different contexts, or not.
We align ontologies where there might be ambiguity on a concept (or a few concepts) if the surrounding concepts are not considered, and where labels taken separately may not be explicit. For example, we align two ontologies that have nothing to do with one another, but they have some concepts which have similar labels. If possible, some might represent different concepts, some might represent the same concept, but in such a way that the mapping should not be done, etc.

character Project LT4eL[17]() tap[18]()

space [19](NOT AVAILABLE) Shuttle Ontology[20]()

score The Music Ontology[21]() Mid-level-ontology[22](NOT AVAILABLE)

square [23]() Brand City journal[24]()

We expect the strategy used for ontology selection to be particularly important. Our intuition is that anchoring will probably give bad results, unless restricted to a pool of ontologies selected according to the number of terms that they share.

5.1.5 Scaling test

This trial aims at determining the conditions under which the context-based approach can scale, to align bigger ontologies. Hence it uses the huge ontologies Agrovoc-NAL which have already tested with Scarlet (§2.2.1).
It seems to us that there are not many ontologies available where the relation is present. So the recall will probably be very low. Because of this, it might scale correctly, if it is well developed.

5.2 One-to-one performance comparison of Scarlet configurations

In this section, we present various series of parameter combinations to test. Each combination constitutes an ontology matcher. For each series, we detail the aim, specify the methodology to be used — including the parameters that vary and those that are fixed, give the expected results, and possibly discuss a few questions.

[17] http://watson.kmi.open.ac.uk/ontologies/LT4eL/CSnCSv0.01Lex.owl
[18] http://139.91.183.30:9090/RDF/VRP/Examples/tap.rdf
[19] http://frot.org/space/0.1/
[20] http://semspace.mindswap.org/2004/ontologies/System-ont.owl
[21] http://pingthesemanticweb.com/ontology/mo/musicontology.rdfs
[22] http://reliant.teknowledge.com/DAML/Mid-level-ontology.owl
[23] http://www.ontologyportal.org/translations/SUMO.owl
[24] http://brandcity.livejournal.com/data/foaf

5.2.1 Original version of Scarlet, to serve as a baseline

The aim of this series is to establish a baseline of the results that we can obtain with the initial version of Scarlet developed at Open University, with the various parameters that were associated to it.

The parameters that vary are therefore: `checkDeepInheritance`, `depth`, `checkForInheritedRelations`, `checkForNamedRelations`, `checkForDisjointRelations`, `stopAtFirstRelation`.

5.2.2 Comparison of the two versions of Scarlet in comparable configurations

The aim of this series is to test Scarlet 2.0 in a configuration similar to the one of the initial version, and to do it with various methods for label comparison. In comparison, we test the original version of Scarlet with the best parameters used in the previous series, and varying between the strict label comparison, or the token-based similarity measure.

For Scarlet 1.3, the varying parameters are: `labelComparison` and a few significant combinations of parameters determined in the previous subsection. For Scarlet 2.0, we use no composition, and vary the same kinds of parameters: `labelComparisonMethods`, `ASSERTIONVERSUSINHERITANCEtype`, `conceptConnectionStrategy`, and `relationType`.

The results should be comparable.

5.2.3 Comparison of contextualisation by ontology matching and by anchoring

The aim of this series is to establish the impact of the use of ontology matchers for the contextualisation versus a simple label comparison.

From this time on, we consider only Scarlet 2.0 for the experiments. Still, we do not consider any composition, but the ontology matcher varies, including AROMA (very fast matcher based on simple label and structure comparison). We match ontologies that have a minimum percentage of common terms, and redo the tests with a few different threshold values. For the anchoring version, we use either ontologies from this selection, or select the ontologies for each couple of concepts to match.

We expect the contextualisation by ontology matching to produce more precise alignments, but anchoring after a first selection of ontologies may not be far behind, as this should limit the number of concepts taken "out of context".

This experiment is also concerned with the performance of the system: how long will the version with ontology matching take? Will it generally take more time than the other, or less? (and with the alignment server?) In which circumstances does it take more time?

5.2.4 Test of the validity of relations found through composition

The aim of this series is to test the influence of the composition strategy on the validity of the resulting alignment. Can we have confidence in the results found by composition?

We consider the most significant combinations of parameters from the previous experiment, and change the strategy, to compose relations coming from up to 2, 3 or 4 ontologies. For each of these experiments, we consider the performances, and get the number and percentage of relations found by adding composition with one more ontology. We also measure the precision of the relationships added by increasing the composition level. We finally measure the overall precision (after aggregation).

5.2.5 Test of the usefullness of exploring when a path has been found

The aim of this series is to see whether there may be an interest to continue the exploration of ontologies when two concepts have been related (keeping the same alignments).

We test with the most significant combinations of parameters found previously, for composition level varying between 0 and 2 or 3, according to what is still relevant. The type of relations considered for the exploration, the maximum length of the path to consider and the methods used for deriving such relations will vary during this test. The result will be evaluated through the precision of relations found that have increasing lengths of ontology paths, and the precision of relations, according to the total length of the path found (including correspondences).

We expect the precision of the final relation decreases with the overall length of the path, but also increases quickly when the path in an ontology is made of more than 3 relations.

5.2.6 Test the usefullness of confronting results from various ontologies

The aim of this series is to test whether it is useful to consider other ontologies to find how to connect two concepts once a path has been found between them.

For this experiment, the strategy of concept connection and aggregation methods will vary, notably the method where we reject the results of 1-2 ontologies that give divergent results, or the opposite, consider only ontologies that give “unusual” results.

5.2.7 Test the originality of relations found

The aim of this series is to determine whether the relations found through this approach could have been retrieved by other means.

We consider a few significant parameter combinations, and match the ontologies directly using the same method that was used (also anchoring). We then remove from the set of relationships found through the context-based approach the ones that were also found through the application of the same matcher used to contextualise for the same parameter combination. We measure the percentage of relations from the context-based approach that were original compared to the traditional approach, the precision of the correspondences found by the two methods, and the precision of the correspondences that are original for the context-based approach method.

5.3 Synthesis

We have provided a testing plan for evaluating the relevance of parameters within the possible configurations of Scarlet 2.0.

We initially had more ambitious plans. However, exhaustively testing the set of parameters has revealed too resource consuming, even for small ontologies. The main reasons for this were that we were testing the high-end parameters (propagating through all the ontologies of the semantic web) and that the naïve processing of Scarlet was continuously calling Watson, saturating the network.

For these reasons, we reengineered both the tests and the Scarlet implementation (§4.3).

However, this prevented us from completing the evaluation.

Conclusion

The goal of this work was to revisit context-based matching as we defined it at the beginning of the project.

For that purpose, we provided a framework in which the various parameters governing context-based matching could clearly be expressed. We reimplemented Scarlet within this framework making it literally a matcher generator. We also included it within the Alignment API so that it can be called through the NeOn toolkit Alignment plug-in and other tools supporting the Alignment API.

We have also defined a set of realistic tests for exploring all those matchers that the new Scarlet can generate.

Further work along this path will require completing the implementation of the missing parameters within the new Scarlet. This will be done progressively. We would also like to pass all the tests that have been defined and get a deeper insight into context-based matching.

Bibliography

[Aleksovski *et al.*, 2006] Zharko Aleksovski, Michel Klein, Warner ten Kate, and Frank van Harmelen. Matching unstructured vocabularies using a background ontology. In S. Staab and V. Svatek, editors, *Proc. EKAW, Praha (CZ)*, LNAI. Springer-Verlag, 2006.

[Caraciolo *et al.*, 2008] Caterina Caraciolo, Jérôme Euzenat, Laura Hollink, Ryutaro Ichise, Antoine Isaac, Véronique Malaisé, Christian Meilicke, Juan Pane, Pavel Shvaiko, Heiner Stuckenschmidt, Ondrej Svab, and Vojtech Svátek. Results of the ontology alignment evaluation initiative 2008. In Pavel Shvaiko, Jérôme Euzenat, Fausto Giunchiglia, and Heiner Stuckenschmidt, editors, *Proc. 3rd ISWC workshop on ontology matching (OM), Karlsruhe (DE)*, pages 73–119, 2008.

[Euzenat *et al.*, 2007] Jérôme Euzenat, Antoine Zimmermann, Marta Sabou, and Mathieu d'Aquin. Matching ontologies for context. deliverable 3.3.1, NeOn, 2007.

[Euzenat *et al.*, 2009] Jérôme Euzenat, Alfio Ferrara, Laura Hollink, Antoine Isaac, Cliff Joslyn, Véronique Malaisé, Christian Meilicke, Andriy Nikolov, Juan Pane, Marta Sabou, François Scharffe, Pavel Shvaiko, Vassilis Spiliopoulos, Heiner Stuckenschmidt, Ondrej Sváb-Zamazal, Vojtech Svátek, Cássia Trojahn dos Santos, George Vouros, and Shenghui Wang. Results of the ontology alignment evaluation initiative 2009. In Pavel Shvaiko, Jérôme Euzenat, Fausto Giunchiglia, Heiner Stuckenschmidt, Natasha Noy, and Arnon Rosenthal, editors, *Proc. 4th ISWC workshop on ontology matching (OM), Chantilly (VA US)*, pages 73–126, 2009.

[Gangemi, 2004] Aldo Gangemi. Restructuring semi-structured terminologies for ontology building: a realistic case study in fishery information systems. Deliverable D16, WonderWeb, 2004.

[Haase *et al.*, 2006] Peter Haase, Pascal Hitzler, Sebastian Rudolph, Guilin Qi, Marko Grobelnik, Igor Mozetič, Damjan Bojadžiev, Jerome Euzenat, Mathieu d'Aquin, Aldo Gangemi, and Carola Catenacci. Context languages – state of the art. Deliverable D3.1.1, NeOn, 2006.

[Mascardi *et al.*, 2009] Viviana Mascardi, Angela Locoro, and Paolo Rosso. Automatic ontology matching via upper ontologies: a systematic evaluation. *IEEE Transactions on knowledge and data engineering*, 2009. to appear.

[Sabou and Gracia, 2008] Marta Sabou and Jorge Gracia. Spider: Bringing non-equivalence mappings to oaei. In Pavel Shvaiko, Jérôme Euzenat, Fausto Giunchiglia, and Heiner Stuckenschmidt, editors, *Proceedings of the 3rd International Workshop on Ontology Matching (OM-2008) Collocated with the 7th International Semantic Web Conference (ISWC-2008), Karlsruhe, Germany, October 26, 2008*, volume 431 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.

[Sabou *et al.*, 2006] Marta Sabou, Mathieu d'Aquin, and Enrico Motta. Using the semantic web as background knowledge for ontology mapping. In Pavel Shvaiko, Jérôme Euzenat, Natalya Fridman Noy, Heiner Stuckenschmidt, V. Richard Benjamins, and Michael Uschold, editors, *Proceedings of the 1st International Workshop on Ontology Matching (OM-2006) Collocated with the 5th International Semantic Web Conference (ISWC-2006), Athens, Georgia, USA, November 5, 2006*, volume 225 of *CEUR Workshop Proceedings*, 2006.

[Sabou *et al.*, 2008a] Marta Sabou, Mathieu d'Aquin, and Enrico Motta. Exploring the semantic web as background knowledge for ontology matching. *Journal on data semantics*, 11:156–190, 2008.

[Sabou *et al.*, 2008b] Marta Sabou, Mathieu d'Aquin, and Enrico Motta. Scarlet: Semantic relation discovery by harvesting online ontologies. In Sean Bechhofer, Manfred Hauswirth, Jörg Hoffmann, and Manolis Koubarakis, editors, *The Semantic Web: Research and Applications, 5th European Semantic Web Conference, ESWC 2008, Tenerife, Canary Islands, Spain, June 1-5, 2008, Proceedings*, volume 5021 of *Lecture Notes in Computer Science*, pages 854–858. Springer, 2008.

[Zimmermann, 2008] Antoine Zimmermann. *Sémantique des réseaux de connaissances: gestion de l'hétérogénéité fondée sur le principe de médiation*. Thèse d'informatique, Université Joseph Fourier, Grenoble (FR), 2008.

Appendix A

Configuring Scarlet — How To

Scarlet was originally a tool to search for paths between two concepts defined in two distinct ontologies. It relies for this on the ontology search engine Watson, which also provides for simple anchoring methods, so as to search for all concepts having a similar label to the concept searched for. It has now been embedded in an ontology matcher compatible with the Alignment API.

We will first give a reminder of the syntax to call an ontology matcher through the Alignment API (Section 1). We present then the parameters available for Scarlet (Section 2 and 3 respectively for the new and old version). In Section 4, we describe the syntax for generating series of tests where parameter values vary. Finally, we show in Section 5 how to configure the parameters to use and the values that they can take.

A.1 Reminder on the use of the Alignment API

The Alignment API offers a façade that offers to match ontologies possibly reusing previous alignments, to render the resulting alignments and evaluate them. The only requisite is for ontology matchers to include a public class that implements the interface `org.semanticweb.owl.align.AlignmentProcess`.

Then, the ontology matching is executed by executing a jar file, where the main class is `fr.inrialpes.exmo.align.util.Procalign`, and the syntax is as following.

A.1.1 Syntax

The syntax is: `java -jar <filename>.jar <options> <arguments>`.

Options are formatted with the syntax `-<parameter>=<value>` for all parameter that we want to pass to `fr.inrialpes.exmo.align.util.Procalign`, which is the main class of the jar file. Parameters for the ontology matcher can be set by `-D<parameter name>=<value>` . If any parameter is declared which was not defined by the matcher, it is ignored.

Options for Procalign

The only parameter required for `Procalign` is:

impl full name of the class that implements the interface `AlignmentProcess`;

Other optional parameters include:

debug value between 0 and 9, indicating level of debug information, 0=none.

renderer full name of the class corresponding to the alignment renderer chosen. By default, an RDF file is generated.

alignment alignment file given as input, from which to proceed the alignment (not used with Scarlet)

threshold double value under which to filter the similarity values

cutmethod method for computing the threshold, chosen among hard|perc|prop|best|span

Arguments

The arguments consist of the URI of the two ontologies to align.

A.2 Configuring Scarlet 2.0

A.2.1 Syntax

Properties to be set for Scarlet 2.0

This is the current configuration:

ContextMatchingStrategy takes only one value: `COMPLETE`.

CompositionMethod takes only one value: `RELATIONAL`

CompositionDeepeningStrategy takes values among `NEVER` and `UNTIL_LEVEL_MAX`.

PoolingMethod takes values among `GET_ALL_ONT_WHERE_THERE_ARE_ANCHORS` (default), `SELECTION_BASED_ON_ANCHOR_NB`, `GET_ALL_ONT_WHERE_THERE_ARE_ANCHORS`

SelectionMethod takes values among `CONCEPT_BASED_SELECTION` (default), `GLOCAL_SELECTION`, `CONCEPT_BASED_SELECTION`

OntologySelectionStrategy takes values among `WEB_SCALE`, `FIXED_SET_OF_ONT`

OntologyPoolingStrategy takes values among `NONE`(default), `POOL_RESTRICTED_FOR_EACH_MATCHING`, `ONCE`

MatchingMethod takes values among `TOKEN_BASED_ANCHORING` (default), `URI_BASED_ANCHORING`, `LOCAL_NAME_BASED_ANCHORING`, `SIM_LOCAL_NAME_BASED_ANCHORING`.

ExplorationMethod takes values among `WEIGHT_BASED` (default) and `EDGE_BASED`.

ExplorationStrategy takes values among `NONE`, `UNTIL_PATHLENGTH`, `UNTIL_FIRST_SUCCESS`(default)

If the value chosen for a parameter is not correct, then it is ignored, and the default choice is set.

A.2.2 Example

A.3 Configuring Scarlet 1.3

Since July 2009, Scarlet is an ontology matcher, and is compatible with the Alignment API. It can therefore be launched through a call of the class `Procalign`, with all its parameters.

A.3.1 Syntax

The syntax is the one of the Alignment API.

Options for Procalign

The only parameter required for `Procalign` is:

impl `fr.inrialpes.exmo.scarlet.matcher.ConfigurableCtxtBasedMatcher`

If one wants to launch the original (< 2.0) version of Scarlet, one should use instead `AAPCMatcher`, in the same package,.

Parameters to be set for Scarlet <2.0, with RelationFinder

Here below are the current properties for Scarlet, with the default configuration `-Dconf=fr.inrialpes.exmo.scarlet.searcher.RelationFinderAdapter`. Properties that are not set take a default value as fixed in the class `AAPCMatcher`.

CheckDeepInheritance true or false

Depth 1, 2, 3 ou 4

CheckForInheritedRelations consider not only assertions but also relations that are inherited (true or false)

CheckForAllRelationTypes consider various relation types, such as named relations, to relate the concepts with one another (true or false)

StopAtFirstRelation stop when a path has been found to relate the two concepts (true or false)

Parameters to be set for Scarlet <2.0, with RelationComposer

Here below are the current properties for Scarlet, with another configuration `-Dconf=fr.inrialpes.exmo.scarlet.searcher.RestrictedRelationComposerAdapter`. Properties that are not set take a default value as fixed in the class `AAPCMatcher`.

depth 1, 2, 3 ou 4

subsumptionRelations consider subsumption relations (true or false)

namedRelations consider named relations, to relate the concepts with one another (true or false)

StopAtFirstRelation stop when a path has been found to relate the two concepts (true or false)

A.3.2 Example

```
java -jar Scarlet.jar
    --impl=fr.inrialpes.exmo.scarlet.matcher.AAPCMatcher --debug=1
    http://hoffmannp.free.fr/onto/reviewingProcess.rdf
    http://hoffmannp.free.fr/onto/publication.rdf
```

This call launches the initial version of Scarlet with the options by default, which are `depth` at 4, `CheckDeepInheritance` at true, `CheckForInheritedRelations` at true, `CheckForAllRelationTypes` at true and `StopAtFirstRelation` at false. It generates a printout on the main output stream, in RDF format.

A.4 Generating a series of tests for Scarlet

Scarlet is compatible with the Alignment API. As a result, it is also compatible with the test generator, which aims at launching any ontology matcher multiple times, with various combinations of parameters.
The test generator calls the ontology matcher several times, according to a set of parameter combinations which is determined at the command.

A.4.1 Syntax

The syntax is: `java -jar scarlet-test.jar <options> <arguments>`.
The main class is `fr.inrialpes.exmo.scarlet.tests.trial.TestGenerator`, so there is no need to specify the name of the class to call. The main difference from previous versions is that options may also be directed to the test generator. These options are not passed on to `Procalign`, except for the `debug` option.

Options for the test generator

There are a few parameters for the test generator:

log filename for the logs

threads number of threads to be used

destdir directory where to write the alignments

debug This option is passed to `Procalign`, but its value is also used for the test generator. The test generator uses log4j, and the correspondence used is: 0 = OFF, 1 = FATAL, 2 = ERROR, 3 = WARN, 4-7= INFO, 8 = DEBUG, 9= ALL, default = 2.

Properties

The difference from previously is that when a property is not present in the command, the test generator will generate a command for each value that the property can actually take (according to the property definition in the matcher). If, for example, five properties are defined in the matcher, but none is set in the command, then there will be 2^5 calls to `Procalign`.

A.4.2 Arguments

Again, the arguments consist of the URI of the two ontologies to align.

A.4.3 Output

The alignments computed will be stored in files whose names format will conform to `align_[parameter-value]*.rdf`. An exception to that rule is that when a property takes the value "true", it is supposed that the property is boolean, and the part `-true` is omitted.
Also, as the file is generated by the Alignment API itself, it respects the standard. Each property is inserted in the alignment file, along with its value.

A.4.4 Example

```
java -jar scarlet-test.jar
    --impl=fr.inrialpes.exmo.scarlet.matcher.AAPCMatcher
```

```
--destdir=test/output --threads=1
--debug=1 -DCheckDeepInheritance=true
http://hoffmannp.free.fr/onto/reviewingProcess.rdf
http://hoffmannp.free.fr/onto/publication.rdf
```

Here only the property `CheckDeepInheritance` has been set. There will be 4*2*2*2=32 calls generated, where the other properties defined by Scarlet will take all the values that they can possibly take, as defined. There will be as many alignments computed, which will be stored in the directory `test/output`.

A.5 Changing the parameters that vary in the series of tests

To set up new configurations, one will have to create a class where parameters are defined, with all possible values that they can take, the value by default, and the function that is called to actually take this parameter into consideration for the alignment.

A.5.1 Create a new configuration class

The class has to implement the interface `Configurable`, which can be done almost automatically by extending the class `AbstractConfigurable` (both of them are in the package `fr.inrialpes.exmo.scarlet.matcher.conf`).
This class must define all properties that the matcher will take into consideration, and all the values that we want to allow these properties to take. These properties are of the type `Parameter` with the modifiers `static final private`. The arguments to give for the parameter constructor are, in the mandated order:

1. the parameter name; it should not be too long, and contain only regular characters, as it actually serves for the name of the alignment file generated.

2. an instance of the class `Parameter.Function` that indicates the function that needs to be run to put into effect the value chosen for the parameter. The function takes as arguments the instance of the resource that has to be called to change the parameter value, and the value. This is due to the fact that parameters are defined statically.

3. the default value for the parameter.

4. all the possible values for the parameter (including the default one), in the order when they will be selected for the automatic generation of parameter combinations.

The constructor of the class should initialise the resource instance in the constructor, and return it when in the function `getConfiguredResource`.

Example

```
static private final Parameter setAnchoringMethod=
    new Parameter("AnchoringMethod",   new Parameter.Function(){
   public void run(Object configTool, Object option){
((CtxtMatcher)configTool).setAnchoringMethod((AnchoringMethod)option);}},
   AnchoringMethod.LABEL\_COMPARISON,
   AnchoringMethod.LABEL\_COMPARISON);
```

A.5.2 Use of a new configuration

Once the configuration class is ready, run the `ant` task *testjar* to generate a new jar file `scarlet-test.jar`. With it, you can do the same as described earlier, but with the parameters that you have defined, simply by adding the option `-conf=<fullname of the configuration class>`. To set some fixed values for the parameters that you have defined, use the name that you have set for the parameter `-D<parameter name>=<value>`.