UNIVERSITY OF NICE - SOPHIA ANTIPOLIS

DOCTORAL SCHOOL STIC

SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

P H D T H E S I S

to obtain the title of

PhD of Science

of the University of Nice - Sophia Antipolis

Specialty : Computer Science

Defended by

Olivier COMMOWICK

Design and Use of Aatomical Atlases for Radiotherapy

Thesis Advisor: Grégoire MALANDAIN

prepared at INRIA Sophia Antipolis, ASCLEPIOS Team

defended on February 14, 2007

Jury :

Reviewers :	Patrick CLARYSSE	-	CNRS (CREATIS)
	Louis COLLINS	-	McGill University
Advisor :	Grégoire MALANDAIN	-	INRIA (Asclepios)
President :	Nicholas AYACHE	-	INRIA (Asclepios)
Examinators :	Pierre-Yves BONDIAU	-	Centre Antoine Lacassagne (Nice)
	Guido GERIG	-	University of North Carolina
	Vincent GRÉGOIRE	-	Université Catholique de Louvain
Invited :	Hanna KAFROUNI	-	DOSISoft S.A.

Acknowledgments

Last thing to do :-)

Contents

CHAPTER 1

Introduction

Contents

1.1 Illustration Example

1.1.1 A subsection just for fun

Sorry I won't write your PhD here ;) This small text just to mention that this style supports writing with accents such as in french words (thèse, définir, ...). Also I put here a simple way to include an image. This is standard latex. For pdflatex compilation, the extension of the images is jpg. For latex compilation, this is ps or eps. The base folder containing images is set in formatAndDefs.tex, as well as the default extensions added to the image names.

1.2 An equation

Just to show argmin and partial derivative commands.

$$T = \arg\min_T E(T, R, F) \tag{1.1}$$

Regularization:

$$\frac{\partial T}{\partial t} = \Delta T \tag{1.2}$$

1.3 An other section

Showing a great bullet list environment:

- First point
- Second point

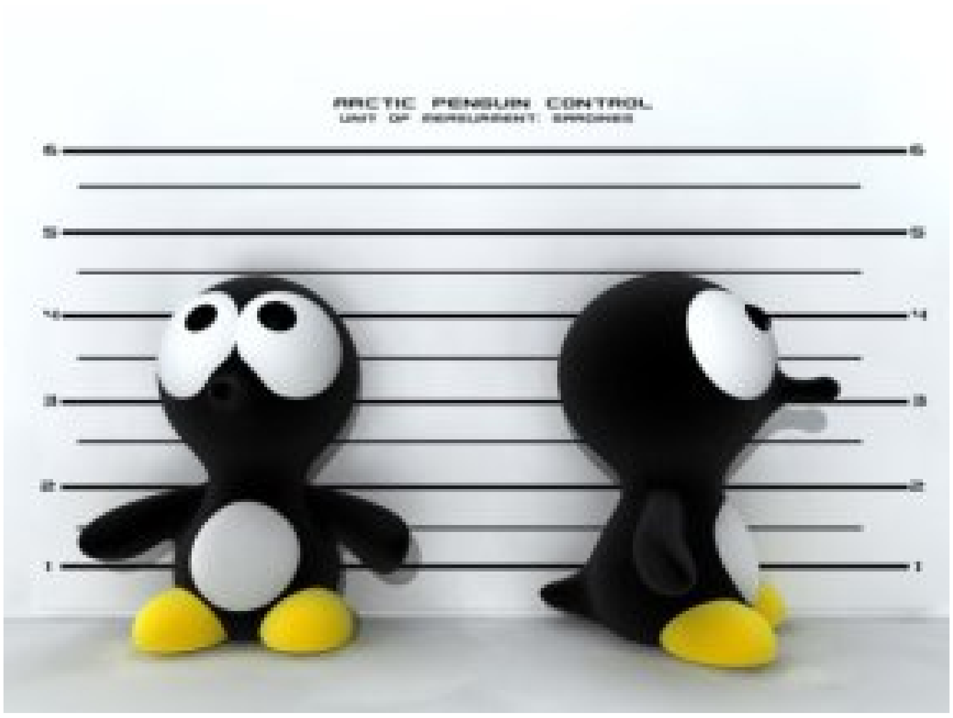


Figure 1.1: A nice image...

CHAPTER 2

No-name yet

2.1 Query taxonomy and re-usability formalization

This section presents the description and formalization of the different types of queries which *(i)* can be processed by our integration approach; and *(ii)* can be compared to previous integration requests in order to take advantage from previous integration plans. The query definition is introduced below.

Definition 1 *A query is defined as a n-tuple:*

$$Q := \langle s, t, A, R, S, C, w \rangle$$

where: A is a set of abstract services defining the query Q; R is a set of user preferences that can be defined over the data services or the entire query; S is a set of data services that were selected satisfying the restrictions defined by R to potentially rewrite the query Q; C is a set of compositions that were produced using the data services in S and satisfying the restrictions defined by R that potentially can answer the query Q; and w is the composition that were selected and executed to answer the query Q.

The query taxonomy proposed below is defined according to the type of relation that can be established between two queries. Queries are classified in four groups:

- *Group 1:* The data denoted by the answer of Q_1 is the same data expected by the answer of Q_2. For example, Q_1 and Q_2 retrieve patients that were infected by pneumonia.
- *Group 2:* The data denoted by the answer of Q_1 is a subset of the data denoted by the answer of Q_2. For example, Q_2 retrieves patients that were infected by pneumonia and Q_1 retrieves patients that were infected by pneumonia and treated by the doctor Lucas.
- *Group 3:* The data denoted by the answer of Q_1 is a superset of the data denoted by the answer of Q_2. For example, Q_2 retrieves patients that were infected by pneumonia and treated by the doctor Lucas, and Q_1 retrieves patients that were infected by pneumonia.
- *Group 4:* The data denoted by the answer of Q_1 is different of the data denoted by the answer of Q_2. For example, Q_2 retrieves patients that were infected by pneumonia and treated by the doctor Lucas, and Q_1 retrieves patients that were infected by pneumonia with admission in the hospital Edouard Herriot.

To understand the different types of query, basic concepts regarding *(i)* user requirements, *(ii)* requirements domain, *(iii)* requirements evaluation and *(iv)* comparable requirements should be introduced:

Definition 2 *An user requirement r is in the form $x \otimes c$, where x is an identifier; c is a constant; and $\otimes \in \{\geq, \leq, =, \neq, <, >\}$. The user requirement r could concern* *(i) the entire query, in this case noted as r_Q; or (ii) a single service, noted as r_S. For instance, the total response time is obtained by adding the response time of each service involved in the composition.*

Definition 3 *A requirement domain is a set of possible values which can be assumed by an user requirement r, represented by $Dom(r)$. For instance, a requirement domain "response time" includes the possible values associated to the response time user requirement. Each user requirement r_i has its own requirement domain D_i.*

Definition 4 *The evaluation of an user requirement r, indicated by $eval(r)$, returns a set of values $\{v_1, .., v_i\}$ that can be assigned to r such that $\{v_1, .., v_i\} \subset Dom(r)$.*

Definition 5 *Given two user requirements r_1 and r_2, both can be comparable, denoted by $r_1 \perp r_2$, if and only if: $Dom(r_1) = Dom(r_2)$.*

The thirteen types of queries included in the taxonomy described in the following sections are organized according to their groups.

2.1.1 Queries that can potentially be completely reusable

There are two types of queries belonging to this group. Given a previous query Q_1 stored in the query history and an incoming query Q_2, the types are: *(i)* Q_1 and Q_2 are completely equivalents (the simplest case); and *(ii)* Q_1 and Q_2 comprehend the same abstract services but Q_2 specifies user requirements less restrict than Q_1. The characteristics of these queries are described below:

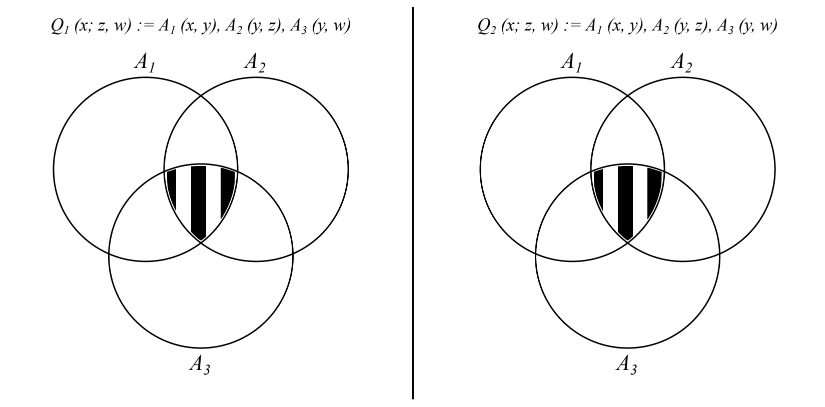
a) *Query type 1:* the *first* type is the simplest case. The figure 2.2 illustrates the manner this query is represented. Given a previous query Q_1 and an incoming query Q_2, Q_1 is equivalent to Q_2 when: *(1)* both queries expect the same data as answer, which means they cover the same abstract services (Figure 2.2 - Data point of view). In this sense, the set of abstract service of Q_1, denoted as $Q_1.A$, is equals to the set of abstract services of Q_2, denoted as $Q_2.A$.

$$Q_1.A = Q_2.A$$

(2) For each user requirement r_i in $Q_1.R$, there is a user requirement r_j in $Q_2.R$ such that the evaluation of r_i is equal to the evaluation of r_j. Consequently, the score of $Q_1.R$ is equals to the score of $Q_2.R$. The *query type 1* and the equivalence between requirements are formally defined below.

Definition 6 *A set of user requirements R_1 is equivalent to a set of user requirements R_2, represented by $R_1 \equiv R_2$, if and only if: $\forall r_i \in R_1,\ \exists r_j \in R_2 \mid eval(r_i) = eval(r_j)$ and $|R_1| = |R_2|$.*

Definition 7 *Query Type 1 – a query Q_1 is equivalent to a query Q_2, if and only if: $Q_1.A = Q_2.A$ and $Q_1.R_1 \equiv Q_2.R_2$.*



Q_1 (x; z, w) := A_1 (x, y), A_2 (y, z), A_3 (y, w)
A_1
A_2
A_3
Q_2 (x; z, w) := A_1 (x, y), A_2 (y, z), A_3 (y, w)
A_1
A_2
A_3

Figure 2.1: Query type 2 representation.

From the re-usability point of view, everything from Q_1 could be reused to answer Q_2. All data services filtered to the query Q_1, denoted $Q_1.S$, potentially could be reused in the query Q_2, excepting the ones that are not online in the exact moment.

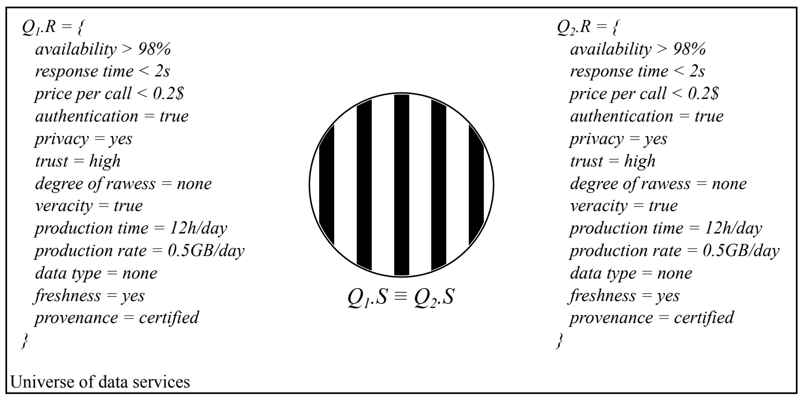


Figure 2.2: Query type 1 representation.

The set of compositions produced to the query Q_1, denoted as $Q_1.C$, potentially could be used to answer the query Q_2, excepting the ones using offline data services. Following, the reusability function for data services, rewritings and queries are presented.

The service reusability function – denoted as $reuse\_services(q_1,\ q_2)$ where q_1 is a stored query and q_2 an incoming query – returns a set of reusable data services for q_2 based on the data services of q_1.

Definition 8 *The service reusability function $reuse\_services(q_1,\ q_2)$ – for queries of group 1 – returns a set of data services S_o, which are online in this exact moment in the data services of q_1.*

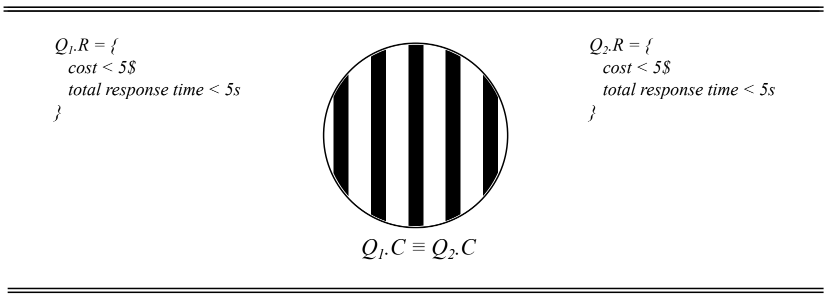


Figure 2.3: Query type 1 - relation between composition sets.

$$S_o \leftarrow S_o \,\cup\, \{ds_i\} \textit{ such that } \{\forall ds_i \in q_1.S \mid ds_i \textit{ is online}\}.$$

The rewritings reusability function – denoted as $reuse\_rewritings(q_1,\ q_2)$ where q_1 is a stored query and q_2 an incoming query – returns a set of reusable compositions to answer q_2 based on the data services and compositions of q_1.

Definition 9 *The rewritings reuse function* **reuse_rewritings($\mathbf{q_1}$, $\mathbf{q_2}$)** *returns a set of rewritings C_o, which uses the data services S_o returned by $reuse\_services(q_1,\ q_2)$.*

$$C_o \leftarrow C_o \,\cup\, \{c_i\} \mid \{\forall c_i \in q_1.C,\ \exists ds_j \in c_i,\ \nexists ds_k \in c_i \mid ds_j \in S_o \textit{ and } ds_k \notin S_o\}.$$

b) *Query type 2* comprehends a query case including less restrict user requirements. This type of query is a general case of *query type 1* as introduced in the figure 2.1. Given a previous query Q_1 and an incoming query Q_2, Q_2 is a subset case of Q_1 when: *(1)* both queries expect the same data as answer, which means they cover the same abstract services. For example, in the figure 2.1 (Data point of view), it possible to note that both queries includes the abstract services A_1, A_2 and A_3 in their definition. In this sense, the set of abstract service of Q_1, denoted as $Q_1.A$, is equals to the set of abstract services of Q_2, denoted as $Q_2.A$.

$$Q_1.A = Q_2.A$$

(2) There exists at least one user requirement r_i in $Q_2.R$ and r_j in $Q_1.R$ such that the evaluation of r_i contains the evaluation of r_j. *(3)* There not exists a user requirement r_k in $Q_2.R$ such that the evaluation of r_k is contained in the evaluation of r_j. Consequently, the score of $Q_1.R$ is higher than the score of $Q_2.R$. The *query type 2* and the less restrict user requirements are formally defined below.

Definition 10 *Given a set of user requirements R_1 and R_2, R_1 is less restrict than R_2, represented by $R_1 \lhd R_2$, if and only if: $\forall r_i \in R_1$, $\exists r_j \in R_2$, $\nexists r_k \in R_2 \mid eval(r_i) \supset eval(r_j)$ and $eval(r_i) \subset eval(r_k)$ and $|R_1| = |R_2|$.*

Definition 11 *Query Type 2 – a query Q_2 is a subset case of a query Q_1, if and only if: $Q_1.A = Q_2.A$ and $Q_2.R \lhd Q_1.R$.*

Once queries of *type 2* are general cases of queries of *type 1*, Q_2 could profit from the entire data (data services and compositions) collected/stored to answer Q_1. The set of data services selected to Q_1 ($Q_1.S$) – respecting the user requirements specified in Q_1 – is a potential subset of the data services selected to Q_2 ($Q_2.S$). In other words, $Q_1.S$ is contained in the set $Q_2.S$ (see figure 2.4). Although $Q_2.S$ could accept other data services which are not in $Q_1.S$ (if the rewriting process was launched from the beginning), all data services from $Q_1.S$ are reusable in $Q_2.S$ excepting the services offline in the exact moment. The service reusability function for queries *type 2* is the same defined for the queries *type 1* (see definition 8).

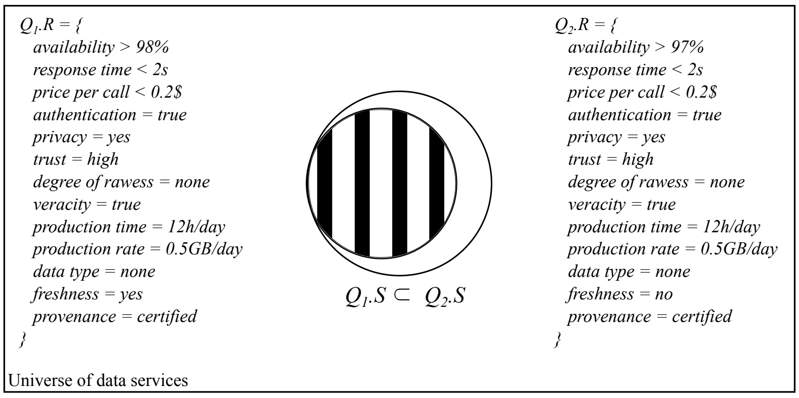


Figure 2.4: Query type 2 - relation between sets of data services.

The set of compositions produced to the query Q_1 ($Q_1.C$) is a potential subset of the compositions to answer the query Q_2 even if the user requirements of Q_2 ($Q_2.R$) are less restrict than the requirements of Q_1 ($Q_1.R$). This means $Q_1.C$ is contained in $Q_2.C$ (see figure 2.5). Although Q_2 could be answered by other compositions which are not in $Q_1.C$ (if the rewriting process was launched from the beginning), all compositions from $Q_1.C$ are reusable in $Q_2.C$ excepting those that uses data services offline in the exact moment. The rewritings reusability function for queries of *type 2* is the same defined for queries *type 1* (see definition 9).

Finally, the query reusability function – denoted as **reuse_query**($\mathbf{q_1}$, $\mathbf{q_2}$) where q_1 is a stored query and q_2 a incoming query – provides the information concerning the data services and compositions that can be used to answer the user request by reusing results from q_1 in q_2.

Definition 12 *The query reusability function* **reuse_query**($\mathbf{q_1}$, $\mathbf{q_2}$) *returns q_2 such that $q_2.S \leftarrow reuse\_services(q_1, q_2)$ and $q_2.C \leftarrow reuse\_rewritings(q_1, q_2)$.*

Atualizar funcoes antes de passar para o outro grupo.
daqui para baixo sera o outro grupo

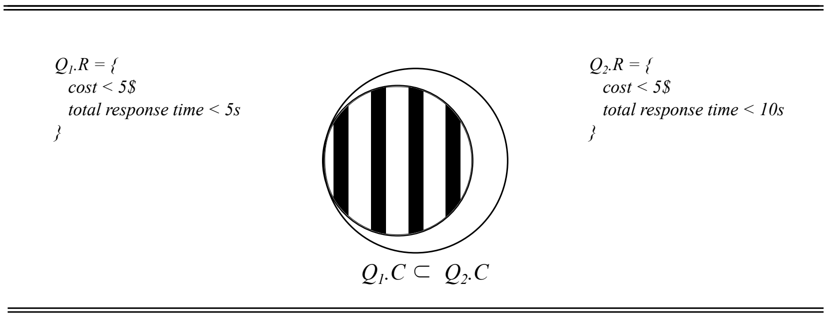


Figure 2.5: Query type 2 - relation between sets of data services.

2.1.2 Group 2

a) Query equivalent with user requirements more restrict

Definition 13 $Q_1.A = Q_2.A$

Definition 14 *Definir servico concreto:*

$$ds := \langle A, R \rangle$$

Definition 15 *Definir QoS aspects of the service respect user requirements, denoted ds satisfies R:*

$$\forall r_i \in ds.R,\ \exists r_j \in R_q \mid eval(r_i) = eval(r_j)\ or\ eval(r_i) \subset eval(r_j).$$

Definition 16 ***RS-2***

$S_o \leftarrow S_o\ \cup\ \{ds_i\}\ such\ that\ \{\forall ds_i \in q_1.S \mid ds_i\ is\ online\ and\ ds_i\ satisfies\ q_2.R\}.$

reuse rewriting: ***RR-1***

b) Query equivalent with user requirements more and less restrict. Definicoes iguais ao anterior.

2.1.3 Group 3

a) Query q2 is a subset of q1 with equivalent user requirements

Definition 17 $Q_1.A \subset Q_2.A$

Definition 18 *Igual ao query type 1. O reuse rewritings tb.*

$$S_o \leftarrow S_o\ \cup\ \{ds_i\}\ such\ that\ \{\forall ds_i \in q_1.S \mid ds_i\ is\ online\ \}.$$

Definition 19 *Given two queries Q_1 and Q_1 as follows:*

$$Q_1 = \langle s,\ t,\ A,\ R,\ S,\ C,\ w\rangle$$
$$Q_2 = \langle s,\ t,\ A,\ R,\ S?,\ C?,\ w?\rangle$$

Where query Q_2 contains undefined sets of data services ($Q_2.S?$) and compositions ($Q_2.C?$), and the composition ($Q_2.w?$) selected to answer the request. The query reusability function of Q_2 based on Q_1, denoted as $reuse\_query(Q_1,\ Q_2)$, completes and returns Q_2 such that:

$$Q_2.S \leftarrow S_i\ \cup\ S_{Q_2-Q_1}$$

Where S_i is a set of reusable data services obtained from $Q_1.S$ and $S_{Q_2-Q_1}$ is a set of data services selected by the algorithm X:

$$S_i \leftarrow reuse\_service(Q_1,\ Q_2)$$
$$S_{Q_2-Q_1} \leftarrow algorithm\_X(Q_1,\ Q_2)$$

This algorithm is responsible (i) to identify and (ii) to select data services that cover the abstract services in the set $Q_2.A - Q_1.A$. In other words, it selects services that are necessary to complete Q_1 compositions to satisfy Q_2.

The set of compositions of Q_2 is obtained by invoking the algorithm Y.

$$Q_2.C \leftarrow algorithm\_Y(C_0, S_{Q_2-Q_1}, Q_2)$$

This algorithm is a simple and reduced rewriting step responsible to improve each composition in $Q_1.C$ adding the absent services to satisfy Q_2. The algorithm expects three inputs (i) the set of reusable compositions C_0 from Q_1 obtained from the reuse rewriting function:

$$C_0 \leftarrow reuse\_rewritings(Q_1,\ Q_2)$$

ii) the set of services to complete the compositions $S_{Q_2-Q_1}$; and (iii) the query Q_2. Finally, the composition $Q_2.w$ selected to answer Q_2 is the one with the highest score among the compositions in the set $Q_2.C$. The query Q_2 is now complete.

b) Query q2 is a subset of q1 with user requirements less restrict

Definition 20 *Tudo igual ao anterior...*

2.1.4 Group 4

a) Query q2 is a subset of q1 with user requirements more restrict

b) Query q2 is a subset of q1 with user requirements more/less restrict

Definition 21 ***RS-2 RR-1*** *and* ***RQ-2***

2.1.5 Group 5

a) Query q2 is a superset of q1 with user requirements equivalents

b) Query q2 is a superset of q1 with user requirements less restrict

Definition 22 *Data service ds covers the query q:*

$$\forall a_i \in ds.A,\ \nexists a_j \in ds.A \mid a_i \in q.A \text{ and } a_j \notin q.A$$

Definition 23 ***RS-3**:*

$S_o \leftarrow S_o \ \cup \ \{ds_i\}$ *st.* $\{\forall ds_i \in q_1.S \mid ds_i$ *is online and* ds_i *covers* $q_2\}$.

RR1 igual..

Definition 24 ***RQ-3**:*

$S_0 \leftarrow reuse\_service(q_1, q_2)$.
$C_0 \leftarrow reuse\_rewritings(q_1, q_2)$.
$C_1 \leftarrow algorithm\_Z(C_0, q_2)$.
$q_2.S \leftarrow S_0 and\ q_2.C \leftarrow C_1$.

2.1.6 Group 6

a) Query q2 is a superset of q1 with user requirements more restrict

b) Query q2 is a superset of q1 with user requirements more/less restrict

Definition 25 ***RS-4**:*

$S_o \leftarrow S_o \ \cup \ \{ds_i\}$ *st.* $\{\forall ds_i \in q_1.S \mid ds_i$ *is online and* ds_i *covers* q_2 *and* ds_i *satisfies* $q_2.R\ \}$.

RR1 and RQ-3

2.1.7 Group 7

a) Query q2 is different from q1 but both have abstract services in common

Definition 26 $\forall a_i \in q_1.A,\ \exists a_j \in q_1.A,\ \exists a_k \in q_2.A \mid$
$a_j \subset q_2.A$ *and* $a_j \subsetneq q_2.A$ *and* $a_k \subsetneq q_1.A\}$.
$n(Q_1.A \cap Q_2.A) > 1$
caso contrario eh melhor reescrever

Definition 27 ***RS-4*** *and* ***RR-1*** *iguais....*

Definition 28 ***RQ-4**:*

$S_0 \leftarrow reuse\_service(q_1, q_2)$.
$S_1 \leftarrow algorithm\_X(q_1, q_2)$.
$C_0 \leftarrow reuse\_rewritings(q_1, q_2)$.
$C_1 \leftarrow algorithm\_w(C_0, S_1, q_2)$.
$q_2.S \leftarrow S_0 \cup S_1$ *and* $q_2.C \leftarrow C_1$.

A case outside of this taxonomy requires a full rewriting process.

2.1.7.1 Query type 2: Q_2 is a subset of Q_1

The *second* type deals with *query subsets* due to more restrict user requirements. Given two queries Q_1 and Q_2, Q_2 is a subset of Q_1 when:

a) They expect the same data as answer, which means they cover the same abstract services. For instance, the set of abstract service of Q_1, denoted as $Q_1.A$, is equals to the set of abstract services of Q_2, denoted as $Q_2.A$.

$$Q_1.A = Q_2.A$$

b) For all user requirement r_i in $Q_2.R$, there is at least one r_j in $Q_1.R$ such that the evaluation of r_i is contained in the evaluation of r_j. For all r_k in $Q_2.R$, there is no r_l in $Q_1.R$ such that the evaluation of r_l is contained in the evaluation of r_k. Consequently, the score of $Q_1.R$ is lower than the score of $Q_2.R$. The definition of more restrict requirements is presented below.

Definition 29 *Given a set of user requirements R_1 and R_2, R_1 is more restrict than R_2, represented by $R_1 \rhd R_2$, if and only if: $\forall r_i \in R_1$, $\exists r_j \in R_2$, $\nexists r_k \in R_2 \mid eval(r_i) \subset eval(r_j)$ and $eval(r_k) \subset eval(r_i)$ and $|R_1| = |R_2|$.*

From the re-usability point of view, a subset of the data services filtered to the query Q_1 which are *online* in the moment, *online*$(Q_1.S)$, could be reused in the query Q_2. This fact occurs due to the more restrict requirements imposed by Q_2. With respect to the compositions, a subset of the rewritings produced to the query Q_1 could also be used to answer the query Q_2. These rewritings should use the data services in *online*$(Q_1.S)$, denoted as *available*$(Q_1.C)$, and respect the more restrict requirements defined in Q_2. The query type 2 definition is presented below.

Definition 30 *Query Type 3 – a query Q_1 is a subset of a query Q_2, if and only if: $Q_1.A = Q_2.A$ and $Q_1.R_1 \rhd Q_2.R_2$*

Definition 31 *Given a set of requirements R_1 and R_2, R_1 contains mixed types of requirements compared to R_2, represented by $R_1 \neq R_2$, if and only if: $\forall r_i \in R_1$, $\exists r_j \in R_2$, $\exists r_k \in R_2 \mid eval(r_i) \subset eval(r_j)$ and $eval(r_k) \subset eval(r_i)$ and $|R_1| = |R_2|$.*

Definition 32 *Query Type 4 – a query Q_2 is a subset of a query Q_1, if and only if: $Q_1.A = Q_2.A$ and $Q_1.R_1 \neq Q_2.R_2$*

Definition 33 *Query Type 5 – a query Q_2 is a superset of a query Q_1, if and only if: $Q_1.A \subset Q_2.A$ and $Q_1.R_1 \equiv Q_2.R_2$*

Definition 34 *Query Type 6 – a query Q_2 is a superset of a query Q_1, if and only if: $Q_1.A \subset Q_2.A$ and $Q_2.R \lhd Q_1.R$*

Definition 35 *Query Type 7 – a query Q_2 is a superset of a query Q_1, if and only if: $Q_1.A \subset Q_2.A$ and $Q_2.R \rhd Q_1.R$*

Definition 36 *Query Type 8 – a query Q_2 is a superset of a query Q_1, if and only if: $Q_1.A \subset Q_2.A$ and $Q_2.R \neq Q_1.R$*

Definition 37 *Query Type 9 – a query Q_2 is a subset of a query Q_1, if and only if: $Q_1.A \supset Q_2.A$ and $Q_2.R \equiv Q_1.R$*

Definition 38 *Query Type 10 – a query Q_2 is a subset of a query Q_1, if and only if: $Q_1.A \supset Q_2.A$ and $Q_2.R \lhd Q_1.R$*

Definition 39 *Query Type 11 – a query Q_2 is a subset of a query Q_1, if and only if: $Q_1.A \supset Q_2.A$ and $Q_2.R \rhd Q_1.R$*

Definition 40 *Query Type 12 – a query Q_2 is a subset of a query Q_1, if and only if: $Q_1.A \supset Q_2.A$ and $Q_2.R \neq Q_1.R$*

Definition 41 *Query Type 13 – a query Q_2 is different of a query Q_1, but sharing abstract services, if and only if: $Q_1.A \neq Q_2.A$ and $n(Q_1.A \cap Q_2.A) > 1$.*

Appendix A

Appendix Example

A.1 Appendix Example section

And I cite myself to show by bibtex style file (two authors) [Commowick 2007].

This for other bibtex stye file : only one author [Oakes 1999] and many authors [Guimond 2000].

Bibliography

[Commowick 2007] Olivier Commowick and Grégoire Malandain. *Efficient Selection of the Most Similar Image in a Database for Critical Structures Segmentation.* In Proceedings of the 10th Int. Conf. on Medical Image Computing and Computer-Assisted Intervention - MICCAI 2007, Part II, volume 4792 of *LNCS*, pages 203–210. Springer Verlag, 2007. (Cited on page 15.)

[Guimond 2000] A. Guimond, J. Meunier and J.-P. Thirion. *Average Brain Models: A Convergence Study.* Computer Vision and Image Understanding, vol. 77, no. 2, pages 192–210, 2000. (Cited on page 15.)

[Oakes 1999] David Oakes. *Direct Calculation of the Information Matrix via the EM Algorithm.* J. R. Statistical Society, vol. 61, no. 2, pages 479–482, 1999. (Cited on page 15.)

Design and Use of Numerical Anatomical Atlases for Radiotherapy

Abstract: The main objective of this thesis is to provide radio-oncology specialists with automatic tools for delineating organs at risk of a patient undergoing a radiotherapy treatment of cerebral or head and neck tumors.
To achieve this goal, we use an anatomical atlas, i.e. a representative anatomy associated to a clinical image representing it. The registration of this atlas allows to segment automatically the patient structures and to accelerate this process. Contributions in this method are presented on three axes.
First, we want to obtain a registration method which is as independent as possible w.r.t. the setting of its parameters. This setting, done by the clinician, indeed needs to be minimal while guaranteeing a robust result. We therefore propose registration methods allowing to better control the obtained transformation, using outlier rejection techniques or locally affine transformations.
The second axis is dedicated to the consideration of structures associated with the presence of the tumor. These structures, not present in the atlas, indeed lead to local errors in the atlas-based segmentation. We therefore propose methods to delineate these structures and take them into account in the registration.
Finally, we present the construction of an anatomical atlas of the head and neck region and its evaluation on a database of patients. We show in this part the feasibility of the use of an atlas for this region, as well as a simple method to evaluate the registration methods used to build an atlas.
All this research work has been implemented in a commercial software (Imago from DOSIsoft), allowing us to validate our results in clinical conditions.
Keywords: Atlas-based Segmentation, non rigid registration, radiotherapy, atlas creation